Ph.D. Thesis

APPLICATION OF DEEP GENERATIVE MODELS FOR THE DESIGN OF PHARMACEUTICAL MANUFACTURING PROCESSES

Diego Alvarado Maldonado

A thesis submitted in fulfilment of the requirements for the degree of Doctor of Philosophy

Strathclyde Institute of Pharmacy and Biomedical Sciences

University of Strathclyde

Glasgow, UK

2024

Declaration of Authenticity and Author's Rights

This thesis is the result of the author's original research. It has been composed by the author and has not been previously submitted for examination which has led to the award of a degree. The copyright of this thesis belongs to the author under the terms of the United Kingdom Copyright Acts as qualified by University of Strathclyde Regulation 3.50. Due acknowledgement must always be made of the use of any material contained in, or derived from, this thesis.

Signed: D. Alvarado

Date: 29-08-2024

Abstract

Designing processes for pharmaceutical product manufacturing is a complex and resource-intensive task. With increasing research costs and quality standards, the pharmaceutical industry seeks innovative technologies to enhance productivity and maintain competitiveness. While a variety of tools exist in the process design domain for optimizing conditions or selecting materials, options for guiding the selection of manufacturing operations remain limited.

In this context, deep generative models (DGMs) emerge as a promising approach. DGMs, known for learning the probability distribution of data, have gained popularity for their ability to generate realistic examples, commonly applied in text and image generation. In drug discovery, DGMs have successfully generated new active substances with desirable properties. However, their application in the manufacturing space remains unexplored. These models have the potential to assist in operation selection and experimental targeting, thereby reducing development time.

This thesis aims to investigate the applicability of DGMs in pharmaceutical manufacturing process design, developing DGMs capable of generating plausible sequences of operations for product manufacturing, taking input information about the target product. A significant challenge in developing DGMs is the requirement for large datasets. To address this, two datasets were constructed using natural language processing (NLP) applied to primary and secondary manufacturing data extracted from patents. The primary processing dataset comprises over 385K manufacturing processes, while the secondary processing dataset includes approximately 9K procedures for various dosage forms and active ingredients.

The study involved training and comparing several architectures based on generative adversarial networks (GAN) and variational autoencoder (VAE) using different metrics. Real and generated sequences were contrasted manually to evaluate how closely the model outputs resemble typical manufacturing sequences. This research contributes to the exploration of DGMs' application in pharmaceutical manufacturing, offering insights into their potential for operation selection and product development. In the end, DGMs were successfully trained and their potential for the generation of plausible sequences was demonstrated. A survey assessed by a panel of experts yielded that the models

generated sequences at least as good as the actual procedures in 38% of occasions for the primary domain. While this shows the potential of generative modelling in this field, it also remarks there is room for improvement to make it applicable in real-world scenarios.

Acknowledgments

I would like to begin by expressing my deepest gratitude to my supervisor, Dr. Cameron Brown. His guidance, patience, knowledge, and unwavering support throughout this project have been invaluable. This experience allowed me to explore and apply knowledge in machine learning and integrate it with pharmaceutical sciences, a field I am deeply passionate about. Without Dr. Brown's mentorship, this project would not have been possible.

I would also like to recognize my co-supervisor, Professor Blair Johnston, for his guidance and feedback on multiple occasions. His insights have been crucial to the development of this work.

Furthermore, I wish to acknowledge all the staff involved in the PhD program at the University of Strathclyde. Their assistance, and collaboration over the years have greatly contributed to the completion of this thesis.

Finally, my heartfelt thanks go to my parents, Aide and Hector, and my sister, Daniela. Their constant support, encouragement, and motivation have been the driving force that helped me persevere throughout this journey.

Contents

Declaration of Authenticity and Author's Rights i

Abstract ii

Acknowledgments iv

Contents v

List of Figures ix

List of Tables xiv

Chapter 1. Introduction 1

1.1 Introduction 2

1.2 Pharmaceutical Manufacturing Development 5

1.2.1 Primary manufacturing 5

1.2.2 Secondary manufacturing 7

1.3 AI/ML Role in Pharmaceutical Manufacturing 10

1.4 Overview on Deep Generative Models 12

Chapter 2. Aims 16

2.1 Aim 17

2.2 Objectives 17

Chapter 3. Methods 19

3.1 Data Collection 20

3.2 Natural Language Processing for Information Extraction 20

3.2.1 Topic modelling 21

3.2.2 Text classification 25

3.2.3 Named entity recognition (NER) 26

3.2.4 Dataset Cleaning and Curation 28

3.3 Sequence Generation with Deep Learning 31

3.3.1 Recurrent Neural Networks 34
3.3.2 Transformers 36
3.4 Deep Generative Models 38
3.4.1 Generative Adversarial Networks 39
3.4.2 Variation Autoencoders 41
3.4.3 Energy-based models 42
3.4.4 Autoregressive models 42
3.4.5 Flow-based models 43
3.4.6 Performance Metrics in Generative Models 43
3.5 Equipment and Software 45
Chapter 4. Data collection 47
4.1 Introduction 48
4.2 Methods 50
4.2.1 Search strategy and documents retrieval 50
4.2.2 Patents selection 52
4.2.3 Content analysis 62
4.3 Results and discussion 63
4.3.1 Search results 63
4.3.2 Patent Classifier 65
4.3.3 Error analysis 68
4.3.4 Pharmaceutical manufacturing corpus 72
4.4 Summary 78
Chapter 5. Natural Language Processing (NLP) for Pharmaceutical Manufacturing Data Extraction 79
5.1 Introduction 80
5.2 Methods 82

5.2.1 Preliminary experiments 82
5.2.2 Relevant section detection 83
5.2.3 Named entity recognition (NER) 88
5.3 Results and discussion 93
5.3.1 Relevant section identification 93
5.3.2 Named entity recognition 102
5.4 Summary 114
Chapter 6. Deep Generative Models for Primary Manufacturing Process Design 117
6.1 Introduction 118
6.2 Methods 120
6.2.1 Dataset development 120
6.2.2 Output representation 123
6.2.3 Input representation 124
6.2.4 Modelling 128
6.2.5 External Validation 138
6.3 Results and discussion 138
6.3.1 Data description 138
6.3.2 Generative modelling 148
6.3.3 Conditional generation of manufacturing sequences 165
6.3.4 Expert assessment 169
6.4 Summary 170
Chapter 7. Deep Generative Models for Secondary Manufacturing Process Design 173
7.1 Introduction 174
7.2 Methods 176
7.2.1 Dataset development 176

7.2.2 Input and Output representation 178
7.2.3 Generative modelling 179
7.3 Results and discussion 183
7.3.1 Data description 183
7.3.2 Generative modelling 193
7.4 Summary 198
Chapter 8. Conclusions 200
8.1 Conclusions and future work 201
References 210
Appendices 230
Appendix A: List of standardised operations 230
Appendix B: Topics keywords for LDA analysis 233
Appendix C: Ranges for clusters of conditions 250
Appendix D: Survey for assessment of the quality of generated procedures for primary manufacturing by CVAE and Invariant CVAE 252

List of Figures

Figure 1-1. Product formulation and manufacturing process development scheme. Taken and adapted from Osakwe et al work [31]. 8
Figure 3-1 Example NER output for a secondary manufacturing paragraph. 27
Figure 3-2 Example dependency tree for a sentence. *nsubjpass*: passive nominal subject; *auxpass*: auxiliar passive; *cc*: coordinating conjunction; *conj*: conjunction; *prep*: preposition; *pobj:* object of a preposition 29
Figure 3-3 Visualisation of representations generated by a convolutional neural network for different layers trained on images. Taken from Zeiler et al work [104] 31
Figure 3-4 Multilayer Perceptron (MLP). 32
Figure 3-5 Architecture classes based on input-output data [107]. 34
Figure 3-6 Architecture recurrent neural network. 34
Figure 3-7 Diagram of the most common types of cells used in RNN. From left to right, Simple RNN, Long-short term memory (LSTM), and gated recurrent unit (GRU) [107]. ... 35
Figure 3-8 Transformer Architecture. Taken from Vaswani et al [112]. On the left, the encoder module can be seen which includes two sublayers self-attention, and a feed-forward network. On the right, the decoder is similar to the first sublayers, but instead of feed-forward, a cross-attention sublayer is introduced. This sublayer is fed with information on the encoder output and self-attention of the sequence. Then, the results are input into another feed-forward network that produces a representation used for a particular task. 37
Figure 3-9 GAN architecture for generator and discriminator networks during training. Taken and adapted from Alom et al [69] 39
Figure 4-1. Overview pharmaceutical corpus development. 50
Figure 4-2 Example CPC scheme for patent technical classification. Taken from USPTO [145] 52
Figure 4-3 word2vec architectures. On the left hand-side continuous bag-of-words (CBOW) and on the right skip-gram (SG). For a sequence of words, CBOW mode will predict word t using as inputs the words t-2, t-1, t+1 and t+2 and word. Whereas SG mode will output the context words t-2, t-1, t+1 and t+2 given the word t. Taken and adapted from Mikolov et al [88] 56
Figure 4-4 doc2vec architecture. Taken and adapted from Le Q et al [153]. 57

Figure 4-5 General scheme of a RNN architecture for binary classification. Text Vectorization layer assigns an index to each word. This index is used by the embedding layer to locate the corresponding vectorial representation for each word in a lookup table. Then, these embeddings are fed into the RNN. In this case, there are two stacked RNN layers where each process the information in different directions (bidirectional). The output of these layers is subsequently concatenated to be handled by a dense layer. This final layer will outcome the probability of the input text belonging to the category $y = 1$. Taken from Tensorflow website [157] 60
Figure 4-6 Patent Classifier architecture using abstracts. For claims, output_sequence_length is 1001 instead of 116, as in abstracts. Hyperparameters were based on Risch J. et al work [151]. 61
Figure 4-7 Word clouds with top 50 most important words for all the search results (A), irrelevant results (B), and pharmaceutical patents (C)..... 64
Figure 4-8 Classification performance for different models. A) comparison of AUC-PR for several model trained for different inputs and text representations. B) Precision-recall curve for the best model – RNN + Fasttext using abstract and claims as inputs 66
Figure 4-9 Comparison of Abstract + RNN performance using custom and pretrained embeddings (fasttext) as a word representation during training time..... 67
Figure 4-10 Histogram (A) and cumulative frequency (B) of the number of words for abstracts and claims. 68
Figure 4-11 Confusion matrix for the best model. PH: pharmaceutically relevant patents and NP: non-relevant patents. 69
Figure 4-12 Distribution of CPC classes (A) and keywords for FN (B) and FP (C). 71
Figure 4-13 Top 5 keywords for subtopics 0 to 8 obtained using NMF. 73
Figure 4-14 Top 5 keywords for subtopics 9 to 17 obtained using NMF. 74
Figure 4-15 Overall distribution of main topic in the corpus (A), evolution of topics across time (B) and evolution of the patents most related to small molecules or biological products (C). To note, data for 2021 only covers until September, which is why a sudden drop in the number of patents is seen this year. 75
Figure 4-16 Number of NCE and biologics approved by FDA between 1997 and 2021. Taken and adapted from de la Torre et al [164] 77
Figure 5-1. Overview information extraction (IE) using natural language processing (NLP) tools. 82

Figure 5-2 Sequence of steps for text preprocessing for topic modelling. 84
Figure 5-3 Example errors of labelling. Case I presents example of labelling error. Case II displays error in prediction. 90
Figure 5-4 Common architecture for NER using DL. 91
Figure 5-5 Paragraph topic classifier performance for the assigned labels..... 94
Figure 5-6 Example of undesirable results of paragraph classification approach. 95
Figure 5-7 Distribution of the number of tokens per section (A) and Perplexities determined for LDA models evaluating shuffling (B) and truncation effect (C)..... 96
Figure 5-8 Top 10 keywords for truncated (A) and non-truncated (B) LDA models. 97
Figure 5-9 Performance for k-Means model changing the number of clusters..... 98
Figure 5-10 t-SNE visualization of document representations generated using LDA. ... 99
Figure 5-11 Example documents with similar content and classified in the same cluster (primary manufacturing) referenced in Figure 9. 100
Figure 5-12 Normal plot of hyperparameters effects on F1-score for NER models. A: sequence max length (number of tokens), B: number of Conv1D layers, C: kernel size, D: number of filters, E: dropout rate, F: LSTM units, G: spatial dropout 1D rate, H: number of heads (MHA layer), I: keys dimension (MHA), J: embeddings dimensions, K: embeddings windows size, L: learning rate, M: weight decay, and N: initializer. 103
Figure 5-13 Normal plot (A) and magnitude (B) of hyperparameters effects on F1-score from Confirmatory Placket Burman design. B: number of Conv1D layers, C: kernel size, D: number of filters, E: dropout rate, F: LSTM units, G: spatial dropout 1D rate, H: number of heads (MHA layer), I: keys dimension (MHA), J: embeddings dimensions, K: embeddings windows size, and M: weight decay. 105
Figure 5-14 Normal plot (A) and magnitude (B) of hyperparameters effects on F1-score from Factorial design. B: number of Conv1D layers, F: LSTM units, and H: number of heads (MHA layer). 107
Figure 5-15 F1-score micro average vs the Number of Heads in MHA layer 108
Figure 5-16 Contour plot Weight Decay vs Learning Rate. 109
Figure 5-17 Best model architecture for NER..... 111
Figure 5-18 Confusion matrix for NER 113
Figure 6-1. Steps for generative model development for manufacturing routes generation for primary domain. 120

Figure 6-2 Top 10 most common substances in the different categories of materials. THF: tetrahydrofuran, DIPEA: N-ethyl-N-isopropylpropan-2-amine, Brine: typically, a saturated solution of sodium chloride in water. 141
Figure 6-3. Comparison of properties distributions between target molecules in the training set (n = 359,912) and approved drugs by FDA recorded in ZINC15 database (n = 1,615) [203]. LogP, log-partition coefficient calculated using RDkit library [205]; LogS, log-molar solubility calculated using SolTranNet [206]; MW: molecular weight. 142
Figure 6-4 Distribution of conditions reported in the training set. Ar: Argon, H2: Hydrogen, N2: Nitrogen. 143
Figure 6-5 Most frequent operations by (A) standardised actions and (B) type of operation. Purify refers to chromatography. Dry Solution consists of the removal of water from a mixture usually through a desiccant agent. Make Solution refers to the combination of two or more materials to form a solution, a suspension, or other type of mixture. .. 145
Figure 6-6 Evolution across time of the number of patents mentioning the most common separation methods. RECRYSTALLISATION refers to crystallisation from a solid material, while CRYSTALLISE starts from a solution. DRY indicates that the method of drying is unspecified. 147
Figure 6-7. Evolution across time of the number of patents mentioning the most common separation methods with respect to the total number of patents. 147
Figure 6-8. Distribution of the amount of target molecule obtained in the respective process in log-scale (n = 274,956) 148
Figure 6-9 Comparison of sequence prediction accuracy in terms of BLEU scores resulting from distinct molecular representations of a target substance. 149
Figure 6-10 Reconstruction Accuracy (A) and condition importance (B) for different generative models for output A generation using target molecular descriptors. CD: condition included in discriminator; st: stochastic encoder; de: deterministic encoder. 155
Figure 6-11 Learning curves of feature importance (FI), BLEU and generated sequence validity for A) CVAE ($\beta = 0.05$) and B) Invariant CVAE ($\beta = 0.01$, $\lambda = 0.02$) trained with different number of layers. 160
Figure 6-12. Effect of β-control method on CVAE performance during the training process calculated on development set. 163

Figure 7-1. Steps for generative modelling for secondary manufacturing route generation. 176
Figure 7-2 Top 10 most mentioned substances for the secondary manufacturing dataset. 184
Figure 7-3 Comparison of properties distributions between secondary manufacturing dataset molecules in (n = 2,178) and approved drugs by FDA recorded in ZINC15 database (n = 1,615)[203]. LogP, log-partition coefficient,t calculated using RDkit library 31; LogS, log-molar solubility calculated using SolTranNet [206]; MW: molecular weight. ... 186
Figure 7-4 Top 10 most frequent dosage forms found in the extracted patents. 187
Figure 7-5. Distribution of conditions pH, temperature, and time obtained. 189
Figure 7-6 Top 10 most frequent operations found for the entire dataset (A) and some common dosage forms such as tablets (B), emulsion (C), and solutions (D). 190
Figure 7-7 Evolution of the number of records obtained across time (A) and breakdown by form (B). Other refers to non-conventional dosage forms such as microspheres, microparticles, nanoparticles, among others. 191
Figure 7-8 Distribution of material weighted for secondary manufacturing in log-scale. Median weight = 49.9 g (N = 2,576) 192
Figure 7-9 Effects of hyperparameters on reconstruction (BLEU) and dosage form importance (FI_{df}). Plots A and C represent the absolute effect for BLEU and FI_{df}, respectively. Plots B and D correspoind to the normal plots of the effects for BLEU and Fi_{df}, respectively. The hyperparameters codes are dropout rate (A), token dropout (B), batch size (C), and KL target (D). 195
Figure 7-10. Most important effect for reconstruction (A – dropout rate and B – Target KL) and dosage form importance (C – Interaction Token dropout + Batch size) 196

List of Tables

Table 1-1. Typical unit operations in synthesis. Taken from Am Ende et al work [3] 6
Table 1-2. Typical unit operations for solid dosage forms manufacturing. Taken and adapted from Mittal B's work [32] 9
Table 1-3. Types of DGMs: advantages and disadvantages [47,63]. 14
Table 3-1. Libraries and equipment used for data analysis. 45
Table 4-1 Criteria of exclusion for search terms. 51
Table 4-2 CPC Scheme sections. Taken and adapted from USPTO [145] 52
Table 4-3 Hyperparameters for assessed text representations training. Each representation was determined for abstracts and claims, separately. 54
Table 4-4 Hyperparameters for assessed models. [a] class weights were calculated using scikit learn package for imbalance data. [b] Separate models were trained for abstracts and claims. [c] Recurrent neural networks were trained using pre-trained fasttext and custom embeddings as word representations. 58
Table 4-5 Excluded terms in content analysis. 62
Table 4-6 Distribution of the search results by sources and search terms. 63
Table 4-7 Distribution of the search results by relevant (PH) and irrelevant (NP) patents with CPC. 65
Table 4-8 Assignation of subtopics to main topics 74
Table 5-1 Exclusion criteria for token in vocabulary for BoW representation. 84
Table 5-2 Hyperparameters assessed for LDA model development 85
Table 5-3 Definition of labels. *Category added for the final model. 88
Table 5-4 Assessed hyperparameters for selection and optimization of NER model. 92
Table 5-5 Degree of agreement between labels assigned based on LDA keywords and manually revised sections 100
Table 5-6 Initial Placket Burman design and results for hyperparameters selection. A: sequence max length (number of tokens), B: number of Conv1D layers, C: kernel size, D: number of filters, E: dropout rate, F: LSTM units, G: spatial dropout 1D rate, H: number of heads (MHA layer), I: keys dimension (MHA), J: embeddings dimensions, K: embeddings windows size, L: learning rate, M: weight decay, and N: initializer. 102
Table 5-7 Confirmatory Placket Burman design and results for hyperparameters selection. B: number of Conv1D layers, C: kernel size, D: number of filters, E: dropout

rate, F: LSTM units, G: spatial dropout 1D rate, H: number of heads (MHA layer), I: keys dimension (MHA), J: embeddings dimensions, K: embeddings windows size, and M: weight decay.....104
Table 5-8 Full Factorial design and results for hyperparameters selection. B: number of Conv1D layers, F: LSTM units, and H: number of heads (MHA layer).....106
Table 5-9 Summary selected values for NER model hyperparameters. *Layers to confirm 109
Table 5-10 F1-scores (micro-average) for confirmatory models in development and test sets. BiLSTM: Bidirectional Long Short-Term Memory, Conv1D: unidimensional convolution, Attention: multihead self-attention, CE: Character embeddings.....110
Table 5-11 Breakdown of the performance for best NER model. 112
Table 6-1. Exclusion criteria of manufacturing procedures from the dataset for model training. [a] Duplicates corresponded to patents that reported the same molecules and procedures but with a different scope. 122
Table 6-2. Examples of manufacturing sequences representations. $$: material; %% condition; T: temperature; t: time; if no letter is included within %%, it refers to pH. ...123
Table 6-3. Assessed molecular representations for generative modelling. 124
Table 6-4. Sequence prediction networks. Architecture taken and adapted from the decoder proposed by Dollar et al[127]. [a] transformer blocks were implemented as proposed by Vaswani et al[112].....126
Table 6-5. Sequence prediction networks for representation B. Adapted from Vaucher et al [43].....127
Table 6-6. Initial metrics used for assessing sequence generation performance. [a.] Validity criteria for representation B was limited to referencing all the materials in the output sequence and the correct identification of target molecule.....132
Table 6-7. Encoder and decoder architecture using Transformers layers. [a.] If encoder is deterministic only one linear layer was used and μz and $\log\sigma z2$ were excluded. [b.] Transformer decoder block was only included in CVAE. The other model did not process information on materials.....133
Table 6-8. Material encoder architecture using Transformers layers. [a.] If the encoder is deterministic, only one linear layer was used, whereby μz and $\log\sigma z2$ were excluded. [b.] Transformer decoder block was only included in CVAE. The other model did not process information on materials.....134

Table 6-9. Survey design 138
Table 6-10. Extraction performance metrics per entity type for primary manufacturing dataset. 139
Table 6-11. Performance metrics for prediction of actions associated with materials and conditions (Output B, n = 1000). [a.] Results from reference model developed by Vaucher et al[43]. Architecture B corresponded to an adaptation of Vaucher's Transformers trained on our dataset with different representations. Architecture A is a modified version of B with half transformer blocks and heads. [b] It refers to the quantiles of accuracies obtained by sequence matching evaluated using Levenshtein similarity. For instance, 3.6% of the sequences had 100% matching between actual and predicted in Vaucher's Transformer. [c] it only considers architecture A and B. 152
Table 6-12. Examples of randomly generated sequences by different models. 154
Table 6-13. Performance metrics for variations in CVAE architecture. [1.] Percentage of unique sequences obtained from new generated sequences; [2.] Percentage of generated sequence found in the training set. *Pooling by Multihead Attention, implemented as proposed by Lee et al [199] 157
Table 6-14 Performance metrics for architecture modification in CVAE model in the encoder, pooling method, and latent space injection trained using PI for β dynamic adjustment with KL target = 35. 161
Table 6-15. Performance metrics for CVAE trained under different β-control strategies. 164
Table 6-16. Effect of KL target on performance metric for Invariant CVAE. 164
Table 6-17. Example generated vs actual manufacturing sequence. CVAE and Invariant CVAE were trained using PI with target KL = 35. 165
Table 6-18. Determination coefficient (R^2) for process yield prediction for different models. Yield-BERT's authors trained that model using datasets: Gram and subgram. The former compile values of yield obtained when the final amount of product was in the order of grams. Subgram uses data with amounts in the order of miligrams. 168
Table 6-19. Results survey. 169
Table 7-1. Exclusion criteria of manufacturing procedures for secondary manufacturing dataset. 177
Table 7-2. Output representation for secondary processing. $$: condition; T: temperature; t: time; if no letter is included within $$, it refers to pH. 178

Table 7-3. Proposed CVAE architecture based on Transformers for secondary manufacturing sequence generation. 179
Table 7-4. Automatic performance metrics employed for generation of manufacturing sequences for secondary domain. 182
Table 7-5. Performance metrics assessed over the test set for CVAE using different values of dropout rate (A), token dropout (B), batch size (C), and KL target (D). 193
Table 7-6. Examples of generated sequence for different APIs and dosage forms. ... 197

Chapter 1. Introduction

1.1 Introduction

In recent years, the pharmaceutical industry has embarked in a transformation process aiming to meet patient's needs and regulatory requirements as well as adapt itself to the changes in health market [1]. Historically, the industry has been distinguished by having a business model strongly centred on the discovery of new chemical entities that are patented and commercialized throughout patent expiry time [2,3]. This focus has made the largest pharmaceutical companies invest most of their efforts and resources in the design and development of novel active pharmaceutical ingredients (API) [4]. However, the innovation in other processes of value chain, such as manufacturing, seems not to maintain the same rate as in research and development (R&D) for new API, which has resulted in inefficacious processes with many opportunities for improvement [5].

Nonetheless, the investment in R&D has not stopped increasing. Between 2001 and 2020, on average, leading companies have expanded their spending in this area annually by 6% [6]. This has been translated into more than 200 new molecules approved for commercialisation within the mentioned period [6,7]. However, profits have been reported to exhibit a declining trend since 2013 [8]. Aligned with this, analyses point out that less than half of the new approved substances are profitable, thereby highlighting the risks and the challenges in productivity that the industry must face [6]. It has been argued that constantly rising research costs is the main contributing factor to this trend [4,8–11]. In turn, tendencies of cost have been associated with increasing regulatory requirements and development times [8,10]. As a result, new APIs are not enough to cover the losses caused by patents expiry and cost reduces new products profit, which increases the need of implementing strategies that enhance productivity and make companies more sustainable [6,10,11]. Thus, the industry has sought for solutions to tackle inefficacies in different steps in the value chain, and particularly there has been a special interest in manufacturing [1,4,12].

Aspects to improve in pharmaceutical manufacturing include flexibility in scale-up/down, production speed, process control and reliability, and inventory management [3,5,13–15]. Thus, initiatives such as Quality by Design (QbD), continuous manufacturing (CM), and process analytical technologies (PAT) have been proposed and implemented throughout the last decades. The impact of the implementation of QbD for process optimization has

been estimated to reduce costs between 10 and 25% [3]. In the same line, large companies such as Jansen have estimated that the implementation of CM could reduce costs up to a 50% [3]. These examples show the conveniency in financial terms of a change towards a paradigm oriented in process optimization and increasing in manufacturing productivity. In addition, the employment of these initiatives is not only reflected in more profit and a cost reduction, but it also has a result more affordable prices for patients, easier accessibility to medicines, ensuring acceptable standards of quality and safety.

Despite the advantages that QbD or CM offer, the transition to these approaches in manufacturing have been slow [1,15]. Reviews published in 2020 and 2022 pointed out that although there have been advancements in the acceptance of QbD for the design and development of products and processes, it is not yet a standard practice in the pharmaceutical industry [16,17]. It has been claimed that some challenges to progress towards a widespread adoption include the extensive requirement of data, experiments, documentation, and specialised tools [16]. From a regulatory perspective for complex products, the lack of data and the complexity of the results to be analysed are additional concerns [16].Nonetheless, the publication of research papers and guidance for the implementation of QbD in several types of products, as well as the increasing implementation of techniques such as multivariate data analysis and different approaches for the design of experiments are expected to help QbD usage [16,17].

In this context, one of the concerns for the adoption of frameworks, such as QbD, is that a large number of experiments may be required for a deep understanding of a process or product [1]. In addition, this may generate a substantial amount of data that, when combined with other sources such as process control observations, needs the use of tools to facilitate their understanding and usage in process optimisation [13,15]. Consequently, techniques that aid data analysis and guide decisions at various stages of research, including experiment selection, have emerged and become popular in the pharmaceutical sector. In this regard, artificial intelligence (AI) has gained special relevance in addressing the aforementioned challenges [15].

AI refers to the application of a series of mathematical models and computational capabilities to assist and guide human decision making [15]. There exist numerous reported applications of AI in pharmaceutical manufacturing. To mention some examples, deep learning (DL) has been employed to automate visual inspection of drug products, conduct

automated risk analysis to predict equipment maintenance, and forecast dissolution profiles, among others [18,19]. Nonetheless, within AI, a group of algorithms have become increasingly popular due to offering a different perspective compared to traditional predictive tools which are Deep Generative Models (DGMs).

DGMs are neural networks with varied architectures that aim to estimate or approximate the probability distribution of high dimensional data [20]. Upon knowing the probability distribution of the data, it is possible to draw samples from that distribution. The drawn samples are expected to have similar characteristics to the original data. In the medical domain, they have been found helpful in tumour classification and COVID-19 diagnosis with good accuracy [21]. In the pharmaceutical context, DGMs have been employed mainly on drug discovery to find out potential drug-like molecules showing promising results [22]. However, to the best of our knowledge, the application of DGM have not been explored deeply in pharmaceutical manufacturing.

DGMs can potentially be used to generate plausible sequences of unit operations to manufacture a product. Resuming pharmaceutical industry challenges in terms of process optimization, this ability to generate sequences might help to explore different combinations of operations that suits with user needs. In addition, these models could assist in targeting the process towards a reduced group of combinations by conditioning the selection based on the properties of input materials or target product. In this manner, the application of DGMs might reduce time and costs in experimentation, complementing other workflows. Nevertheless, for deep generative modelling in a specific domain, a large and reliable dataset is required.

Currently, although a vast amount of information on primary and secondary processing is available in documents such as scientific papers and patents, to the best of our knowledge, there is not a database covering both domains in an appropriate format for their usage in modelling. Due to the substantial efforts that this task requires, automated solutions are needed to obtain as much information as possible. Subsequently, once a suitable dataset is available, an architecture for DGM can be defined and trained. Thus, this thesis aims to develop deep generative models for the design of pharmaceutical manufacturing processes taking molecular descriptors as inputs and outputting a plausible chain of unit operations. To achieve this, a database on pharmaceutical manufacturing domain will be built through the utilisation of natural language processing

(NLP). To finally proceed with the design, training, and validation of the generative models. To contextualise more in several subjects covered thus far, this chapter will dive into traditional approaches in manufacturing development in both primary and secondary production, current state of applications of AI tools in manufacturing, and DGM's types and applications.

1.2 Pharmaceutical Manufacturing Development

Manufacturing involves all the processes carried out to transform inputs (raw materials) into outputs (API/drug product) in order to have a continuous supply. Depending on the outcome, pharmaceutical manufacturing can be primary or secondary [5]. Primary manufacturing comprises the obtention of API which can be achieved by synthesis or extraction from a natural source and further purification steps [5]. On the other hand, secondary manufacturing encompasses the conversion of one or several APIs to a dosage form that can be administrated to a patient [5]. This distinction is convenient regarding development since the considerations in synthesis vary significantly compared to a dosage form production.

Another convenient distinction is the mode of manufacturing. In this regard, the product fabrication can be done in continuous (CM) or as a batch process (BP). Traditionally, BP has been extensively used in both primary and secondary manufacturing [23,24]. This mode is characterized by generating a fixed amount of product and usually, the process can be divided into a chain of sequential steps also known as unit operations. In each step, intermediate products are generated such that feed the next stage. In contrast, in CM, the amount of generated product depends on the time a process is run and some or all the unit operations are integrated in such a way that the transformation process occurs partially or completely in the same equipment [23]. For this work purposes, there will be a focus on batch operations since this mode of manufacturing is the most commonly applied for the different types of products.

1.2.1 Primary manufacturing

In primary manufacturing process design, API origin is determinant to select the process conditions and unit operations. APIs can be classified depending on its origin as small

molecules and biologics [13]. Small molecules can be obtained usually by synthesis, whereas biologics are produced by extraction from natural sources or biotechnology [13].

The following workflow for the development of an API synthesis process is based on Siegfried Labs study case and includes route selection, initial optimization, purification steps, process refining and scale-up and commercial production [25]. The three first steps can be defined at laboratory scale and usually are explored at a scale of hundreds of g to kg [26]. In route selection, synthesis steps are chosen considering aspects such as reduction of the number of steps, avoidance of hazardous transformations, materials cost, and consecutive or parallel reactions. Afterwards, the selected route is fine-tuned by checking every step and select the most appropriate reactants in terms of cost, quality and safety. Additionally, reaction conditions are optimized such as temperature, stoichiometry and concentration to get the best yield.

Table 1-1. Typical unit operations in synthesis. Taken from Am Ende et al work [3]

Unit operation	Considerations
Reaction/Quench	Chemical kinetics, mass transfer, heat transfer, addition, removal
Distillation	Mass transfer, heat transfer, addition, removal, time, stability
Extraction	Mass transfer, heat transfer, addition, removal, time, stability
Crystallization	Chemical kinetics, mass transfer, heat transfer, addition, time, stability, crystallization kinetics, method, stirring
Filtration	Addition, removal, time, stability
Drying	Mass transfer, heat transfer, addition, removal, time, stability

Next, purification steps are defined. At this stage, intermediates and product should be analysed to evaluate whether purification is needed since excessive purification steps will affect yield. For instance, intermediates may have a purity that is good enough for the succeeding process step and so purification is not needed. Other factors to consider are shown in Table 1-1. Once all the synthetic and purification steps are determined, scale-up is performed.

Scale-up can go from kilograms to industrial-scale of produced material. In this phase, the feasibility of the process developed at laboratory scale is assessed in terms of

availability of equipment, conditions, compliance of good manufacturing practice (GMP), safety and risk management. As a result of this assessment, changes in the originally proposed procedure can be done, for instance, new critical parameters can be identified to adjust and control, or some operations have to be reconsidered as might not be feasible for safety issues, scale-up effects or costs. Once the process has been standardised for the intended batch size, validation is performed to report to regulatory bodies to manufacture for commercial purposes.

Other aspects to consider throughout the development process are control strategies and API critical attributes. In solid products, certain the type of salt or polymorph can have a strong influence on API therapeutical efficacy or toxicity as well as its manufacturability as a drug product. In these cases, if a particular type of salt is required, additional reaction steps might be needed; or if a specific polymorph is sought, further purification steps such as crystallization become more critical in the development. Similarly, all the process steps must be controlled appropriately by which certain parameters should be monitored in-process such as temperature, pressure or mixture composition. Finally, in certain steps, API critical attributes like assay, particle size distribution, and impurities should be tested.

1.2.2 Secondary manufacturing

Secondary manufacturing aims to transform an API into a dosage form to be administrated to a patient. A dosage form is a vehicle that facilitates the delivery of an API to the end-user, and contributes to ensuring product stability (physical, chemical and microbiological), efficacy and safety [27]. To accomplish these objectives, a drug product is primarily composed of API, excipients and packaging system [27]. Excipients are defined as materials without biological activity that are added to a product to improve aspects such as manufacturability, stability, patient acceptability and/or product performance [28]. In the same line, container closure systems protect a dosage form from environmental and mechanical risks [28].

In Figure 1-1, usual pharmaceutical development steps are displayed. During the first stage of development a characterization of the API physical, chemical and biopharmaceutical properties take place. However, the development must also be guided by pre-set objectives such as the type of dosage form that will be manufactured and special conditions that the route of administration can have [29]. In this way, in the

formulation stage, appropriate excipients and closure system are chosen based on the dosage form and API stability and biopharmaceutical performance. In this step, the selection of manufacturing process will be established based on API physicochemical properties and dosage forms requirements. To exemplify the importance of this, the product profile indicates that the product is an injectable solution that is administrated through a parenteral route. This product profile points out that the solution requires to be sterile, so the ideal is to work in aseptic conditions during the manufacturing process. In addition, upon being a liquid, the type of unit operations will be limited to those typical for the preparation of this type of dosage forms [30]. In addition, excipients can be added to improve manufacturability for a certain process. In this manner is possible to notice the interdependency among process, excipient, and dosage form, revolving around API.

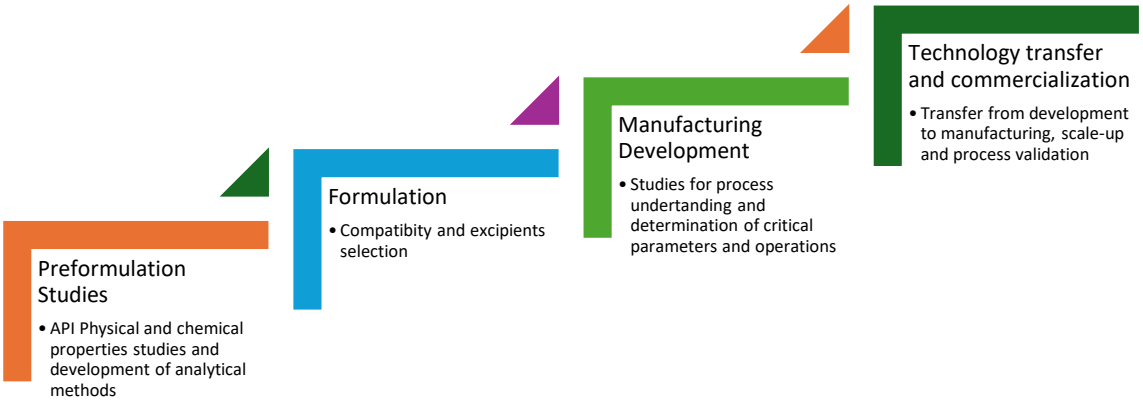


Figure 1-1. Product formulation and manufacturing process development scheme. Taken and adapted from Osakwe et al work [31].

As a result of the pre-formulation and formulation stages, there is a set of excipients and a series of unit operations that can be employed in the fabrication of the dosage form. It is then an analysis is done where the available option of manufacturing, equipment and conditions are taken into consideration to design the process and select unit operations [27]. Subsequently, it is necessary to define process conditions and equipment settings. As can be seen in Table 1-2, every unit operation has several parameters and considerations that can be affected by additives or process conditions. Given the number of variables that can affect process performance and therefore product quality, it is necessary to define what variables or parameters are critical and a range of working values. This task

can be accomplished by employing tools such as risk analysis and design of experiments (DoE) [27,29]. Thus, it is possible to have a complete understanding of process and unit operations to establish the best working conditions.

Table 1-2. Typical unit operations for solid dosage forms manufacturing. Taken and adapted from Mittal B's work [32]

Unit operation	Explanation	Consideration
Granulation	Increase particle size to improve powder flow properties	Technique (dry or wet granulation), additives, particle size distribution, flowability, physical properties of material to granulate
Drying	Remove moisture from a wet powder	Temperature profile, batch size, initial water contents, air flow
Particle size reduction	Reduce particle size	Method (attrition, shear, impact), type of material, moisture
Blending	Mix components, direct impact on drug product uniformity	Blender type (agitator, tumbling or high-shear mixers), mixing time, speed, addition order
Encapsulation	Powder mixture is filled into a hard empty capsule	Filling volume, flowability, capsule size
Compression	Compaction of powder mixture to convert to tablet	Compression force, press type and speed, formulation, die geometry, die filling, moisture, flowability
Coating	Addition of thin film onto granules or tablet surface.	Product flowability, moisture, additives, product geometry and shape, equipment settings

Process understanding leads to the determination of optimal parameters as well as the establishment of process robustness. In addition, with the identification of critical parameters, control strategies can be implemented, prioritizing critical points [27]. Once the robustness and reproducibility of the process have been verified, scale-up proceed. At this stage, changes in equipment and process equipment due to scale change should be incorporated into risk analysis, additional adjustments may be required to standardize the process [33]. Once the process has been defined, it can be validated.

1.3 AI/ML Role in Pharmaceutical Manufacturing

Driven by increasing dataset sizes and enhanced computing capabilities, AI has emerged as a technology with a great potential to improve all the pharmaceutical supply chain leading to more self-managing and autonomous processes [34]. These benefits have been recognised by regulatory bodies such as Food Drug Administration (FDA) and European Medicines Agency (EMEA), which have promoted the adoption and development of frameworks for a good and responsible use of AI [35,36]. Although applications have been reported in the entire lifecycle of pharmaceutical products, there has been a special emphasis in early stages of research. An analysis of FDA submissions between 2016 and 2021 revealed that there is an increasing use of AI tools, showing a significant growth in 2021 with more than 100 submissions using this technology[37]. It has also been highlighted that AI has been mostly utilised in clinical development and drug discovery [37] . This is aligned with previous statements remarking that pharmaceutical industry innovation is more present in R&D.

Although AI includes a wide range of fields, machine learning (ML) is perhaps that most popular area. ML gathers algorithms and models that can learn to identify patterns on data that allow them to perform a task [38]. The set of data used during learning are usually known as training set. Depending on the type of task, these algorithms can be classified into supervised and unsupervised learning [38]. In supervised learning, a ML method aims to learn to map a set of input features $\mathbf{x}$ onto a response or outcome space $\mathbf{y}$ [38]. In turn, depending on the nature of the response $\mathbf{y}$, a task can also be catalogued as regression (continuous) or classification (discrete). These groups may well correspond the most numerous usages of AI in the pharmaceutical context. It can be found how supervised approaches support the prediction toxicity, dose, and other relevant properties in clinical development [37,39,40]. In pharmaceutical manufacturing, models such as neural networks have been integrated into PAT to monitor critical attributes like particle size distribution or composition in both primary and secondary manufacturing [41]. Aligned with the design and development of processes, in the primary domain, supervised approaches have also been used to predict synthesis routes and experimental procedures [42,43]. Route prediction have been achieved using transformers models, that are neural architectures used in novel large language models such as generative pre-trained transformers (GPT) [43].

On the other hand, the applications of unsupervised algorithms seem to be more limited in manufacturing process design. Contrary to its supervised counterpart, these models learn from unlabelled data whereby the final output is not clearly defined [38]. Therefore, they aim to extract underlying or latent information, thereby usually being associated with knowledge discovery. There exists a wide diversity of tasks for which unsupervised learning is relevant, and some include clustering analysis, dimensionality reduction, and density estimation [38]. Particularly, it is common to find multidimensional data in the context of monitoring and control which may be complex to analyse directly. In this scenario, techniques oriented to represent information in a few variables are useful for further analysis. Due to this, methods like principal components (PCA) or autoencoders (AE) have been employed to obtain representations that can be combined with other approaches to perform a task of interest. Examples include again PAT, where PCA is a popular technique employed for visualisation as well as regression by combination with supervised approaches using as inputs spectral data [44]. In the same way, Cacciarelli et al proposed the use of AE for simultaneous monitoring of multiple process parameters and fault detection [45].

Another interesting application of unsupervised learning is feature extraction. Forbes et al applied non-negative matrix factorisation (NMF) to create meaningful features from mass spectra that enables the characterisation of polymorphs and different components in mixed particles samples [46]. At this point, most applications focus on the interpretation of data and characterization of materials or processes that, although crucial for manufacturing development, they do not directly provide information on the process itself. Namely, most information must be interpreted by developers to define operations and conditions of the process, contrary to some supervised examples mentioned above whose output points out more explicitly this information. In that direction, generative models, another type of unsupervised approaches capable of extracting features and estimating probability distributions, have become a promising tool.

Generative models (GM) encompass a set of algorithms that aim to estimate the joint probability distribution for a set of variables representing a group of examples [20,47]. This approach, contrary to traditional classification or discriminative models, such as support vector machines or random forest, are limited to learn a decision boundary for a group of classes. On the other hand, GM requires learning more complex correlations among the

variables under consideration [47]. This distinction has implications for training complexity, as generative modelling can require more time to attain satisfactory results [47]. Most applications of GM are centred on synthesis of data samples that are difficult to differentiate from real samples [20,47–50]. Several algorithms can be used for generative tasks, such as gaussian mixture models (GMM) or Latent Dirichlet Allocation (LDA) [47]. The latter has interesting applications in text analytics which will be explored in the next chapters. Nonetheless, one of the most remarkable group of GMs currently correspond to Deep Generative Models (DGMs). DGMs have gained significant relevance in recent years across many areas, particularly in drug discovery considering the pharmaceutical context. Earlier in this chapter, the potential applications in manufacturing were briefly mentioned. Thus, in the following section, more details about these models, their applications, and variants will be explored.

1.4 Overview on Deep Generative Models

DGMs estimate the underlying probability distribution of a training data making use of a variety of neural network architectures [20,47]. Thus, during the training process, these models learn a generator function able to synthesise examples with similar characteristics to the original data. Compared to other generative modelling approaches, the use of neural networks provides a greater flexibility that allows to handle different kinds of data structures. In other models, when data does not correspond to numeric representations, preprocessing steps are needed to make data “usable”. On the other hand, neural networks can learn these representations and, simultaneously, tailor them to execute a task, without needing the same level of processing. This versatility has led DGMs to be deployed in a wide variety of fields including the generation of images, text, molecules, among many others [51]. Similarly, the versatility and flexibility of these models have led to the development of multiple variants.

To compare different approaches, Goodfellow at al proposed a taxonomy considering models whose method of parameter estimation is closely related to maximum likelihood [52]. In this method, a model, represented as $p_{model}(x; \boldsymbol{\theta})$, assigns a probability to a sample x that depends on model parameters $\boldsymbol{\theta}$. In this manner, the optimisation process aims to find a set of values for $\boldsymbol{\theta}$ that maximises $\log p_{model}(x; \boldsymbol{\theta})$, which corresponds to the likelihood of the model under the training data x. The models can define p_{model} in an

explicit or implicit manner, depending on the tractability of the distribution. When the data distribution has a high complexity, so that may not be computed, is considered as intractable. In this scenario, for parameter estimation, the density function p_{model} can be assumed implicit, as in generative adversarial networks (GAN), or approximated to a simpler distribution, as in the case of variational autoencoders (VAE). Finally, with tractable cases, autoregressive models and normalising flows can be applied. However, the tractability of data imposes strong constraints on the models, which limits the type of data that can be used and demands a higher computational capability to be trained. Thus, it is possible to notice that the complexity of data distribution influences the selection of the model and affect its performance.

Among all the different variants, GAN and VAE constitute perhaps the most representative models. The availability of implementations and their ability to model data, without major constraints, compared to approaches with an explicit density, might explain their widespread diffusion. As has been reiterated, a considerable number of models for drug discovery can be found, in particular, VAE and GAN-based [22,53,54]. Typically, in these models, molecules are represented as SMILES (simplified molecular-input line-entry system) or molecular graphs. Then, models are trained to generate drug candidates in the form of these representations. Upon learning molecules distribution, it is possible to generate and explore a large diversity of chemical structures. It is also worth noting that applications using autoregressive models, normalising flows, and energy-based models such as denoising diffusion have also been reported, increasing the range of options in this area [55–57]. An interesting characteristic of this application is the need of targeting the search of candidates based on a particular group with desirable properties such as partition coefficient (logP) or solubility. In this scenario, DGMs are particularly useful as the architectures allows the incorporation of information on properties to condition the outputs and guide the search.

All these benefits have also been extended to other domains. In material science, there are usages for the discovery of new materials not only in terms of compositions and chemical structures, but it is also used to find out new polymorphs. In this regard, Noh et al developed a VAE to generate synthesisable compositions and polymorphs of vanadium oxide [58]. Another interesting application was proposed by Kim et al where a conditional GAN was used to predict possible crystal structures for a given molecular

composition in ternary systems of Mg – Mn – O [59]. In chemical synthesis design, DGMs, again using VAEs or GANs, have been recently proposed as methods to search novel synthetic pathways [60,61]. Thus far, it is possible to see how generative models has been employed in applications of pharmaceutical interest, where there is a special emphasis on novel materials discovery. More recently, DGMs have been trained as a tool to explore alternative chemical reactions, which can aid primary manufacturing design [62]. Nonetheless, it is worth mentioning that, to the best of our knowledge, there are no additional reports of other applications for manufacturing design, whereby this is an area whose potential is yet to be researched more deeply. Similarly, different types of architectures have been briefly mentioned; however, VAE and GAN may well be regarded as the most popular approaches, possibly explained by the advantages these offer compared to other models. The advantages and disadvantages of several approaches are summarised in Table 1-3. Finally, the different variants will be discussed in the methods section.

Table 1-3. Types of DGMs: advantages and disadvantages [47,63].

Model	**Density**	**Advantages**	**Disadvantages**
Generative Adversarial Networks (GAN)	Implicit	High quality sample generated	Difficult to train. Low diversity in generated samples Limitation for sequential and discrete data. More sensitive to hyperparameters selection. Slow convergence.
Variational Autoencoders (VAE)	Approximated	Fast and easy to train. Explicit and more controllable latent space.	Generated samples quality not good. In image generation, samples tend to be blurry. Prone to collapse when training on sequential data.
Energy-based models	Architecture-dependent (implicit or explicit)	High quality samples and controllable latent space.	Dimensionality constraints. Require long training and sampling times and may be more

			unstable during training. Difficulties for sampling
Autoregressive model	Explicit	Powerful density estimator able to model complex distribution	Restricted to certain types of data. Slow training and generation process.
Flow-based models	Architecture-dependent (explicit but approximation can also be applied)	Powerful density estimator able to model complex distribution.	Dimensionality constraints. Performance tends to degrade with high dimensional data. Limitation for sequential and discrete data. High computational cost.

Chapter 2. Aims

2.1 Aim

The main purpose of this work is to develop deep generative models (DGM) for the design of pharmaceutical manufacturing processes with special focus on variational autoencoders (VAE) and generative adversarial networks (GAN) since these represent the approaches more widely employed. Specifically, the research aims to train models capable of generating multiple plausible sequences of unit operations conditioned on given final product molecular descriptors for both primary and secondary processing. By leveraging the potential benefits of DGM, this study seeks to enable the exploration of diverse manufacturing routes, facilitating more targeted experimentation and reducing the time and resources required for research and development (R&D) efforts in the pharmaceutical industry at early stages. Drawing upon insights from other domains where DGM have shown promising results, such as drug discovery, this research endeavours to demonstrate the implications for process design and innovation.

2.2 Objectives

In order to achieve the aforementioned aims, the following objectives will be pursued:

a. Build a dataset suitable for training deep generative models. The dataset comprises information on chemical identity of the final product, unit operations, dosage form (if applicable) for API synthesis and/or purification route (primary manufacturing) and drug product fabrication (secondary manufacturing) obtained from documents such as patents. Given models based on deep learning require a substantial amount of data for them to be able to capture meaningful information, a number of observations in the order of thousands will be collected. The construction of the dataset implies the processing and analysis of the text to verify the relevancy (Chapter 4).

b. Extract and structure data for model training using automated approaches such as natural language processing (NLP). This will involve training and evaluation of NLP models to effectively extract target information (Chapter 5). Subsequently, the extracted data will undergo cleaning and curation to prepare structured datasets for primary and secondary manufacturing applications.

c. Analyse datasets to identify trends, patterns, and biases. Employing statistical analysis, and visualization techniques, the objective is to establish the type of data available and limitations for the design of neural network architectures in both domains, primary (Chapter 6) and secondary (Chapter 7). By gaining insights into the characteristics and biases present in the datasets, adjustments to data preprocessing, network structures or parameters can be made to enhance model performance and robustness.

d. Design and train a generative network architecture that can generate a set of plausible unit operations for the manufacturing of an API/drug product by assessing different frameworks based on VAE and GAN models (Chapter 6 and 7). The assessment will encompass the effect of various data representations, hyperparameters, and network arrangements on model performance. Given the domain-dependent nature of generative modelling, relevant metrics from other domains will be adapted for sequence generation tasks to evaluate model performance effectively. The experimental setup will include comparisons of different frameworks and configurations to identify optimal settings for generating sequences of unit operations.

e. Validate models for process design. Given the lack of a reference or ground truth due to the unsupervised nature of generative approaches, validation will focus on qualitative assessment of the plausibility of generated sequences. This will involve conducting a survey with domain experts, who will evaluate the generated sequence of unit operations in terms of their suitability for execution in a lab environment. In addition, performance metrics, calculated using a test set and defined in the previous point, will complement the analysis. These metrics will provide quantitative insights into the performance of the generative models for each domain studied (Chapter 6 and 7).

Chapter 3. Methods

3.1 Data Collection

As a source of manufacturing information, Patents were employed. Patents from 1976 to 2021 were obtained from the United States Patent and Trademark Office (USPTO) website. A list of relevant patents was built based on keywords searched on USPTO API and Patent View website, and cooperative patent classification system (CPC). Although the selection of patents was assisted by machine learning tools. The specific approaches applied to gather documentation and build a corpus are discussed in more detail in Chapter 4.

Patents were chosen as a primary source for mining due to ease of access and the extensive literature available addressing the extraction of information from these documents. In this regard, existing databases used for the development of machine learning models contained a substantial number of records, in the order of thousands. This makes manual extraction impractical by which methods to automate this process were used.

3.2 Natural Language Processing for Information Extraction

Natural Language Processing (NLP) is a field related to artificial intelligence (AI) that comprises a set of approaches that enables computing systems to interpret natural language (i.e., text or speech) [72]. In recent years, there has been significant progress and an increase in its usage promoted by developments and improvements in other areas such as big data, deep learning, and computing capabilities. As a result, NLP has become extensively employed in several tasks. Some include neural translation, text generation, and information extraction. For the purposes of this work, we will focus on this later usage.

Information extraction (IE) refers to the process of gathering data of interest from unstructured sources and providing it with a structure to make it usable for machine learning applications [73]. Contrary to tabulated data, unstructured data does not possess a pre-defined organization or format, for example, text. It has been suggested that most available data is characterized by some lack of structure [74]. This limits its usability in different areas which has been a strong driver for the development of strategies to make full use of it.

There exist various approaches that allow the obtention of information from documents. In this regard, ruled-based and machine learning (ML) approaches have been deployed to extract and organize data. The former involves the utilization of linguistic rules based on regular patterns found in texts. These can be based on word morphology, syntaxis, or other features. ML, on the other hand, does not depend on preset rules. Instead, by using data, it identifies or learns patterns that help to a task completion. In turn, ML techniques can be grouped into supervised and unsupervised. Supervised approaches require labelled data to indicate what information to identify, and thus, learn relevant patterns. In NLP context, these are usually employed in text classification or named entity recognition. In contrast, unsupervised techniques do not make use of label data to train models. These can be used in areas such as topic modelling.

For the extraction of data in this thesis, it was necessary to deploy multiple techniques. Text classification and topic modelling were applied to select patents and fragments of text containing relevant information. This is explained thoroughly in Chapter 4. Named entity recognition (NER) was crucial in the identification of keywords associated with manufacturing processes. The details of how a NER system was trained are described in Chapter 5. Finally, to organize the information, different sets of rules and clustering techniques were used as explained in Chapters 6 and 7. Thus, theoretical aspects considered during the application of all these approaches are discussed below.

3.2.1 Topic modelling

Topic modelling encompasses algorithms that aim to capture the prevalent subjects in a set of documents [75]. Typically, these algorithms represent a collection of documents in terms of relevant words or keywords [75]. These words are expected to be related to one another in a meaningful and coherent manner. Consequently, the keywords can be interpreted as a topic of the collection of documents. Among the most common algorithms, non-negative matrix factorization (NMF), latent semantic analysis (LSA) and Latent Dirichlet Allocation (LDA) can be found [75–77].

All these algorithms share the following characteristics. They are unsupervised approaches that output a set of topics, and each topic has a distribution of words. The relevancy of each word in a topic is defined by a weight assigned by the algorithm. In the same manner, each document can be associated with each topic by another set of

weights. These values enable the assessment of how related a topic is to a document by ranking the topics and their set of words. Thus, each document can be labelled with a respective topic. As documents can be naturally grouped into topics, this also can be considered as a text clustering methodology.

Another important aspect to highlight is how words are input into these models. As with any ML algorithm, a numerical representation is necessary to carry out all the required calculations. For cases such as LSA, LDA, and NMF, the collection of documents can be transformed into a frequency matrix. This matrix has as many columns as unique words (V) and the number of rows is equal to the number of documents considered in the analysis (D). In this manner, the frequency matrix will have $D \times V$ dimensions. The value of each cell $w_{i,j}$ represents the frequency of a particular word i in a document j. There exist two variants of this matrix: count-based and term frequency-inverse document frequency (*TF-IDF*). In the former, also known as bag-of-words (BOW), frequency is calculated as the count of each word in each document. TF-IDF, on the other hand, uses the relative frequency for the words by dividing the counts into the number of words in the document (tf) and multiplying by a correction factor related to the number of documents that contain the word (idf). idf factor reduces the effect of words with a high frequency that may not have a significant contribution to the meaning of a document, for instance, articles. Equations 1-3 illustrate the calculation of TF-IDF.

$$tf_{i,j} = \frac{counts\ of\ word\ iin\ document\ j}{number\ of\ words\ of\ the\ document\ j} \quad (1)$$

$$idf_i = 1 + \log \frac{1 + number\ of\ documents}{1 + number\ of\ documents\ containing\ the\ word\ i} \quad (2)$$

$$w_{i,j} = tf_{i,j} \cdot idf_i \quad (3)$$

The main purpose of using topic modelling in this work was to cluster documents and extract their keywords to identify whether the content was relevant for further analysis. Throughout this thesis, NMF and LDA were utilized in different scenarios. Next, these two techniques are described.

3.2.1.1 Non-negative matrix factorization

NMF is a dimensionality reduction method that uses as inputs non-negative matrices [78]. In NLP context, NMF has been used to extract meaningful topics from a collection of

documents. In the same manner, this allows to group documents based on topics and, for a given topic, it is possible to rank how related a document is to a determined topic using output weights. As an input, TF-IDF matrix is frequently used. NMF decomposes the input matrix $\boldsymbol{X}^{\top} \in \mathbb{R}^{V \times D}$ into two positive matrices $\boldsymbol{W} \in \mathbb{R}^{V \times K}$ and $\mathbf{H} \in \mathbb{R}^{K \times D}$, as shown in Equation 4, respectively, where K corresponds to the number of topics.

$$\boldsymbol{X}^{\top} = \boldsymbol{W}\boldsymbol{H} \tag{4}$$

As a result of the decomposition, $\boldsymbol{X}^{\top}$ can be reconstructed by multiplying $\boldsymbol{W}$ and $\boldsymbol{H}$. Consequently, the weights of both, $\boldsymbol{W}$ and $\boldsymbol{H}$**,** represent the contributions of the words to the topics and the topics to the documents, respectively. Thus, a document j is usually assigned with the topic k whose weight is the highest based on their contribution from $\boldsymbol{H}$. For topic interpretation, the top 10 words with the highest weights in each topic derived from $\boldsymbol{W}$ were used.

Regarding the choice of K, this is a hyperparameter that must be specified before training the model. It has been suggested that NMF performs best with values lower than 20 [77]. It has also been recommended that manual revision is important to ensure the quality of the topics since there seems not to be a consensus about metrics for automatic evaluation of topic quality [77]. This thesis uses NMF as a text clustering approach whereby the selection of the optimal number of clusters is based on typical clustering metrics, such as Davies-Bouldin score [79]. However, a revision of the keywords was done to ensure coherence and interpret each topic. The settings used to train this model are specified in Chapter 4.

3.2.1.2 Latent Dirichlet Allocation

LDA is a generative probabilistic model that assumes a document is a mixture of topics [76,80]. Each topic, in turn, follows a characteristic distribution of words. In this approach, documents are represented using a Bag-of-words (BOW) representation. As a result of these assumptions, a document can be generated by a probabilistic process [75,76]. Similar to NMF, LDA requires the specification of the number of topics (K). Nonetheless, it has two additional hyperparameters, $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$. Taking into consideration these assumptions and hyperparameters, a document can be generated through a generative process as described in Algorithm 3-1. When the LDA model is fitted, it yields two main outputs: the probability distribution of documents over topics and the distribution of topics over words.

Algorithm 3-1. Generative Process LDA. Taken and adapted from Ng. A. et al[80]

For each topic $k \in \{1, 2, \dots, K\}$:

1. Sample the distribution of words for each topic $\boldsymbol{\phi_k} \sim Dir(\boldsymbol{\beta})$, with $\boldsymbol{\phi}_k$ corresponding to the distribution of each word in the *k*th topic.

For each document j in a collection of D documents:

1. Draw a sample with the mixture topics for a document j: $\boldsymbol{\theta_j} \sim \boldsymbol{Dir}(\boldsymbol{\alpha})$. $\boldsymbol{\theta_j}$ corresponds to the distribution of topics in the document j.
2. Choose the number of words for $\boldsymbol{j}$: $n_j \sim Poisson(\boldsymbol{\xi})$
3. For each word $i \in \{1, 2, \dots, n_j\}$ the document j,
    a. Choose a topic k from the mixture of topics for the document j $k \sim Multinomial(\boldsymbol{\theta_j})$
    b. Choose a word i from a mixture of words for the document j. word $i \sim Multinomial(\boldsymbol{\phi_k})$

Where $\boldsymbol{Dir}(\boldsymbol{\gamma})$ is a Dirichlet distribution that has the following probability function:

$$p(\boldsymbol{\theta}|\boldsymbol{\gamma}) = \frac{\Gamma(\Sigma_{\mathrm{i}}^{K} \gamma_i)}{\Pi^K \Gamma(\gamma_i)} \theta_1^{\gamma_i - 1} \dots \theta_k^{\gamma_k - 1}$$

With $\boldsymbol{\gamma}$ being a vector with K dimensions, where each element is positive. $\Gamma(\cdot)$ corresponds to the Gamma function. For the generative process, there are two independent Dirichlet distributions involved, one parameterised with $\boldsymbol{\alpha}$ which defines the distribution of topics per document by which has K components. On other hand, the Dirichlet distribution with parameter $\boldsymbol{\beta}$ has as many components as words in the collection of documents and controls the contribution of words over topics.

Since its publication in 2001, LDA has been widely used in topic modelling applications and numerous variants have been proposed [75]. It has also been combined with other algorithms such as K-means [81], as was done in this thesis. The topic probability distribution of each document served as a numeric representation for clustering documents [81]. Other important considerations include the selection of hyperparameters and text preprocessing. It has been advised to use LDA where more than 20 topics are expected for the collection [77]. In our case, default values for α and β from gensim library

were used. The optimization of hyperparameters and text preprocessing is discussed in detail in Chapter 5.

3.2.2 Text classification

Text classification consists of the assignment of a label to a block of text through supervised techniques. This task has been applied widely in different domains. For instance, it has been used to automatically detect misuse/abuse of medication [82]. In the biomedical domain, there are proposals to apply NLP techniques to associate clinical records with classes of diseases [83]. Another example includes the automatic classification of patents into CPC classes [84,85]. The input can be found in different forms, going from sentences to full documents.

Traditionally, these algorithms employ a combination of techniques to transform words into numerical representations, which are then used with traditional ML techniques. The process to develop a model for this task involves the following steps: text preprocessing, vectorization, and training a ML model [86]. Text preprocessing usually involves tokenization (splitting text into tokens), removal of irrelevant tokens, and vocabulary normalisation. In this context, the term “tokens” refers to individual words and punctuation marks. Different approaches for processing text applied in this work are described more deeply in Chapters 4 and 5. It is important to note that the extent of preprocessing varies depending on the task. For instance, there are some scenarios where words such as “not” or “neither” may be considered irrelevant, while in others, such as sentiment analysis, these words are crucial for the task. Following preprocessing, the resulting tokens are vectorised, i.e., they are converted to numerical representations suitable for ML training.

Various approaches have emerged to properly represent words for ML. Historically, frequency-based techniques were widely employed. In this category, it is worth mentioning Bag-of-words (BOW) and TF-IDF matrices, which were mentioned in topic modelling section [86,87]. More recently, in 2013, the development of word embeddings was a breakthrough for NLP [88]. By means of neural networks, continuous representations that capture semantic similarity could be created. Over time, advancements in neural network architectures and learning methods have led to the development of new representations, resulting in the creation of word embeddings with performance improvements. These

advancements have not only benefited text classification but also favoured NLP tasks in general [87]. Different word embeddings used in this work as well as drawbacks and advantages are reviewed in Chapter 4.

Once numerical representations are generated, conventional ML algorithms can be applied. Classification techniques such as logistic regression, support vector machines, and Naïve Bayes are commonly trained for text classification [86]. Nonetheless, deep learning (DL) has emerged as the state-of-the-art approach for this task due to its tendency to outperform more traditional models [89]. More complex DL models involving convolution neural networks (CNN), recurrent neural networks (RNN) or Transformers networks are typically employed [89]. Further details about DL models will be explored in posterior sections.

However, ML models are often characterised by their need for large datasets to effectively learn complex patterns and attain acceptable performances. Considering the supervised nature of text classification, this implies that a significant number of labelled examples are required for model development. To address this limitation, there exist label databases for predefined purposes. For instance, in patent cases, websites, such as PatentView, offer datasets with records of CPC classes for documents from 1976 up to current date. When there is no available dataset for the specific task, it is necessary to generate the data. This can be achieved by manual labelling or using semi-supervised approaches where techniques such as topic modelling can be used to generate labelled data to subsequently train a classification model. In the context of this thesis, text classification was used for patent classification. A significant portion of the collected documents did not report CPC class, whereby the relevance of many could not be verified. In this manner, a DL model was trained to classify unlabelled patents. This application is covered in Chapter 4.

3.2.3 Named entity recognition (NER)

Previously, several NLP tasks, such as text classification and topic modelling, have been explored. These tools have helped distinguish irrelevant from relevant data when extracting information on manufacturing. In the same manner, these methodologies have allowed, to some extent, the summarisation of the data by identifying keywords which enable a more in-depth analysis of the initial collection of documents. However, up to this

point, no specific information has been extracted and organised to fulfil the project's primary objective. To tackle this particular issue, named entity recognition (NER) has played a crucial role.

NER is an NLP task that classifies individual words or tokens into predefined categories. These categories are determined based on specific information targeted for extraction, and this set of terms is known as named entities. An example of a NER output can be observed in Figure 3-1. Concerning the availability of models, generic NER models have been designed to identify places, and names, among other types of information [90]. Additionally, a wide variety of domain-specific NER systems exists, focused on extracting data on chemical substances, diseases, symptoms, etc [91,92]. In order to develop these models, multiple methods can be employed.

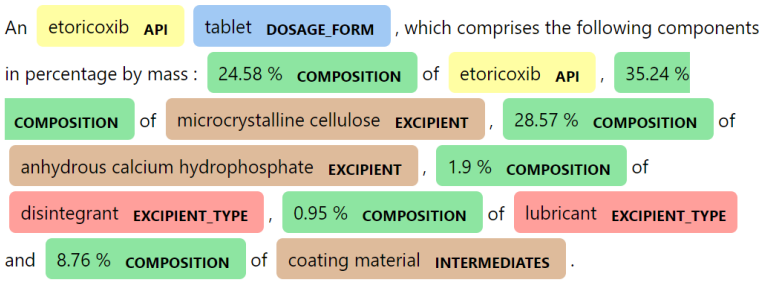


Figure 3-1 Example NER output for a secondary manufacturing paragraph.

As in several NLP tasks, rule-based and ML techniques can be applied in the identification of entities [90]. However, the application of one approach does not exclude the other. ML models combined with rules for refinement have also been reported [90]. Nonetheless, for NER, DL has also provided outstanding results compared to other approaches in several fields [90,93].

When employing DL, currently, the most widely adopted architectures centred around bidirectional long short-term memory (BiLSTM) networks and transformers [90]. Both are sequence-to-sequence models designed to capture the dependency between words. For a sequence of words, these networks are interconnected in a manner that allows information on preceding or subsequent words or steps to be processed at every time step. In this manner, these networks can learn word representations that incorporate meaning and context, to a certain extent.

BiLSTM networks have been used for a longer time compared to transformers. Typically, BiLSTMs are combined with a conditional random field (CRF) model that acts as a decoder to convert word representation into class [90]. However, more recently transformer architecture has emerged as the most representative model for this task in several domains. Apart from offering a higher performance when pretrained in large datasets, transformers are generally faster as they allow parallelisation [90]. Nonetheless, the selection of a model tends to be case-specific. It has been shown that BiLSTM can still outperform pretrained transformers such as SciBERT or BioBERT in domain-specific NER such as material science [93].

For this thesis, a DL model for NER was trained on pharmaceutical manufacturing data. The primary requirement was labelled tokens containing the relevant information. This study aimed to collect data on operations, materials, and conditions for both primary and secondary manufacturing. While there are datasets and models trained on primary processing [94,95], to the best of our knowledge, there was no available dataset with labelled data on both domains was not available, nor was there an existing NER system for extracting the specific terms required. In this manner, a dataset with labelled data was first created to proceed with model training. The process of elaboration of the dataset and the NER system is detailed in Chapter 5.

3.2.4 Dataset Cleaning and Curation

With the NER model, relevant keywords are identified and categorised. The categories enable to organise terms into groups, which in turn help identify the type of information to process. Consequently, materials, conditions, operations, and other types of entities can be separated and analysed. Concerning operations, assuming they occur in the order mentioned in a section allows depicting a procedure as a sequence of actions. In this manner, inputs (materials) and outputs (sequences) are available to train a model; however, some aspects must be addressed to prepare the dataset. Firstly, there are multiple ways to denominate the same term, whereby entities must be normalised to reduce noise and facilitate the identification of relevant subsequences. Secondly, ideally, conditions and materials should be associated with an operation. This aids in setting rules to standardise actions with ambiguous terms and provides a more comprehensive view of the manufacturing process.

To achieve this, a hybrid approach combining ML and ruled-based tools was applied. Initially, the association between materials and conditions with the respective operations was established through dependency trees to set rules. A dependency tree represents the semantic relationships between pairs of words [96]. These relationships are organised in a hierarchical structure, comprising a head and a dependent [96]. In turn, each type of relation can be defined based on grammatical functions [96]. This is illustrated in Figure 3-2. In the example provided, there are 2 actions: “add” and “stir”. In addition, there is a chemical substance (“HCl”) and a condition (“30 minutes”). The dependencies show that “HCl” can be associated directly with “add”, while “stir” can be related to “30 minutes” by descending to a lower level in the tree. Thus, this example highlights the subordination between materials/conditions and operations. Using this concept, it is possible to establish logical instructions to find out the association among entities. To automate this process, along with the developed NER model, Spacy library was used to build the dependency trees for each sentence [97].

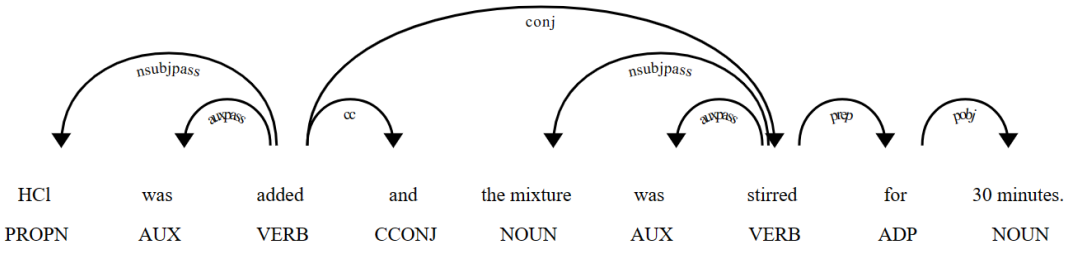


Figure 3-2 Example dependency tree for a sentence. *nsubjpass*: passive nominal subject; *auxpass*: auxiliar passive; *cc*: coordinating conjunction; *conj*: conjunction; *prep*: preposition; *pobj:* object of a preposition

From the extracted data, two distinct datasets were created for primary and secondary manufacturing. The normalisation of terminology was conducted separately for materials, conditions, and operations in each dataset. For materials, dictionaries from several sources were utilised such as Pubchem, Drugbank, ChEMBL, and Pipeline Pilot software [98–100]. The list of obtained materials was cross-referenced with these dictionaries and their corresponding InChI (international chemical identifier) was extracted [101]. In cases where the retrieval of InChI was not feasible, alternative identifiers were used for normalisation such as UNII (unique ingredient identifier). UNII corresponds to an alphanumeric code used to identify a substance considering its properties, based on ISO11238 guidelines [102]. This approach was particularly valuable in secondary

processing for substances, such as excipients, that may not be pure but rather a mixture of various materials. To facilitate this task, removing some subtokens from material terms was necessary as, in some cases, these hindered the search. On the other hand, terminology related to generic entities of intermediary subproducts encompassing terms such as “solution”, “suspension”, and “mixture” was standardised differently.

For intermediates, operations, and conditions, words were grouped based on semantic similarity. To measure this, word embeddings trained for NER were used to represent terms. Principal components analysis (PCA) was applied to representations and the components explaining at least 80% variance were used for clustering analysis. K-means models were then fitted to create clusters. To determine the most appropriate number of groups, Davie-Bouldin score was determined. Thus, the number of groups with the lowest score value was chosen for each type of entity. Then, the top 10 closest and farthest words to the centroid were revised to assign a label to each group. For instance, if a cluster contained terms like “30 °C”, “25 deg.C”, and “80 degrees C”, the cluster was labelled as temperature. In this manner, it was possible to identify types of conditions, operations, and materials and discard terms that might not be relevant. In the case of intermediate materials, the assigned labels served as the standardised form for the terms within the respective clusters.

In the case of conditions, additional steps were undertaken to standardise units and magnitudes, especially for conditions related to temperature, pressure, repetitions, pH, and atmosphere. These were selected since they were the most frequently reported and were important later for model development. Moving on to operations, they were manually revised thoroughly to define a standard way of expressing manufacturing actions. The definition of action categories was adapted from the works of Vaucher et al and Wang et al [95,103]. Among the clusters, it was found categories that were ambiguous. For instance, the term “increase” was found very often but, in isolation, it did not provide sufficient information for a better understanding of a process. Consequently, conditions and materials associated with these terms were also assessed to reassign these entities into a more specific class. The list of the standardised terminology for operations can be found in Appendix A.

Other types of process-related entities such as yield, amount, and concentration were also revised. They were only checked for consistency in units and magnitude. However,

they were not considered for modelling. Finally, a random sample of 100 procedures from the final version of the datasets was drawn. Each sample was compared to the source to validate the extraction accuracy of materials, conditions, and operations. Additional considerations were taken into account depending on the dataset. Details about the cleaning and curation process for the dataset are provided in Chapters 6 and 7.

3.3 Sequence Generation with Deep Learning

DL is a subfield of machine learning that belongs to a family of methods known as representation learning (RL) [64]. DL enables the generation of meaningful features for either supervised or unsupervised learning. This is achieved by performing a series of nonlinear transformations on data. With each transformation, different features are extracted, capturing different levels of detail. This is illustrated in Figure 3-3. As can be seen, in the initial transformations (low-level), the model learns generic information related to colours, eyes, and so forth. As the transformations progress deeper (high-level), the extracted features reveal more complex details, such as shapes and figures. These capabilities make DL a powerful tool for various applications in tasks related to classification, regression, or unsupervised learning. DL excels in fields such as computer vision and NLP, where it finds applications in image detection, captioning, text generation, among many others.

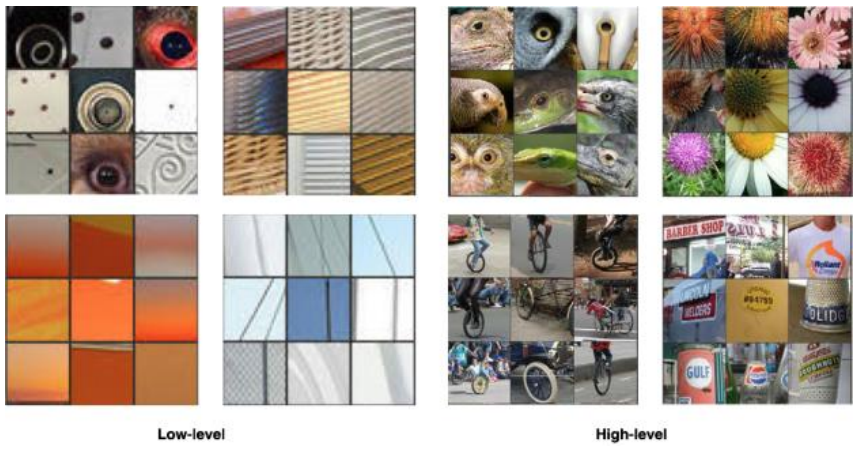


Figure 3-3 Visualisation of representations generated by a convolutional neural network for different layers trained on images. Taken from Zeiler et al work [104]

A MLP graphical representation is shown in Figure 3-4. This can be expressed as a series of nested functions, as indicated in Equation 5 [105]. $\mathbf{x}$ is the input data, $\boldsymbol{\theta} = (\mathbf{W_1}, \mathbf{W_2})$ represents the parameters, σ is a non-linear function also known as activation, and $f(\mathbf{x}; \boldsymbol{\theta})$ is the output. During training, the parameters $\boldsymbol{\theta}$ are learnt by minimizing a loss function l through backpropagation, often using optimisation algorithms such as gradient descent [105]. The MLP in the example consists of two hidden layers, indicating the network depth. The layers have 4 and 2 units, respectively. Within each unit, input data is multiplied by a set of weights, the products are summed. This sum adjusted with a bias term before undergoing activation. Thus, each layer produces as many outputs as units, which feed into the next layer.

$$f(\mathbf{x}; \boldsymbol{\theta}) = \sigma_2\big(\mathbf{W_2}\sigma_1(\mathbf{W_1}\mathbf{x})\big) \tag{5}$$

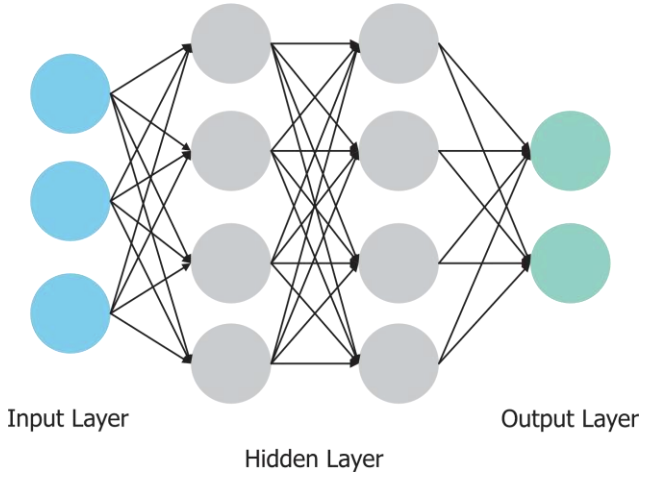


Figure 3-4 Multilayer Perceptron (MLP).

It is important to highlight that MLP, like any neural network, can be configured in various ways. The number of hidden layers, the number of units per layer, the type of unit, the activation function, and the connections between layers, among others are factors that affect the performance of a model for a particular task and define its architecture. In addition, several other variables can impact the model during training. Some to remark include the use of regularisation like normalisation or dropout, as well as the chosen training strategy. The training strategy involves the selection of an optimisation technique and critical parameters such as learning rate. Regarding the optimisers, stochastic

gradient descent (SGD) and adaptative momentum estimation (Adam) are very popular in deep learning community [106]. It is worth noting that many of these factors cannot be learnt and must be chosen carefully to achieve a good performance. This set of factors are known as hyperparameters.

Resuming the concept of architecture, neural network flexibility in terms of how structural hyperparameters can be arranged has led to the design of different models suitable for various specific kinds of data. For instance, convolutional neural networks (CNN) have had a big impact on image-related analysis. However, for this thesis, there will be a special emphasis on sequential data, specifically, sequences of discrete variables.

This type of data refers to sets of variables with a temporal relationship. More exactly, there is an ordinal arrangement among them. Examples of such data can be found in several domains. in NLP, language can be seen as a sequence of words organised in an specific order which gives cohesion and coherence. In the field of chemistry, DNA and proteins can be expressed as an ordered chain of nitrogenous bases and amino acids, respectively. Manufacturing processes also fall into this category, comprising a sequence of unit operations where each step depends on the outcome of the previous one. This class of dependency has allowed the usage of specific architectures designed to tackle temporal relationships. Particularly, architectures such as recurrent neural networks (RNNs) and transformers have shown remarkable results. These will be explored in the next sections.

Finally, the modelling of sequential data has led to the definition of different architectural designs tailored to different input/output scenarios as depicted in Figure 3-5. In the first class, exemplified by a typical MLP, a single input produces a single output. The dimensionality of this output varies depending on model design and task requirements. Then, the subsequent types address various applications with sequential data. In the one-to-many model, a single input generates a sequence, a common scenario in image captioning. Here, the input image, represented by a vector, is transformed into a sequence of words. Conversely, the many-to-one reverses this process with a sequence producing a single output, typically applied in text classification. Lastly, many-to-many architectures, also known as sequence-to-sequence models, generate new sequences based on another one. This approach is commonly seen in language translation or NER. This work places particular emphasis on the generation of sequences. Thus, in

subsequent chapters, one-to-many and many-to-many models will be explored for manufacturing procedures generation.

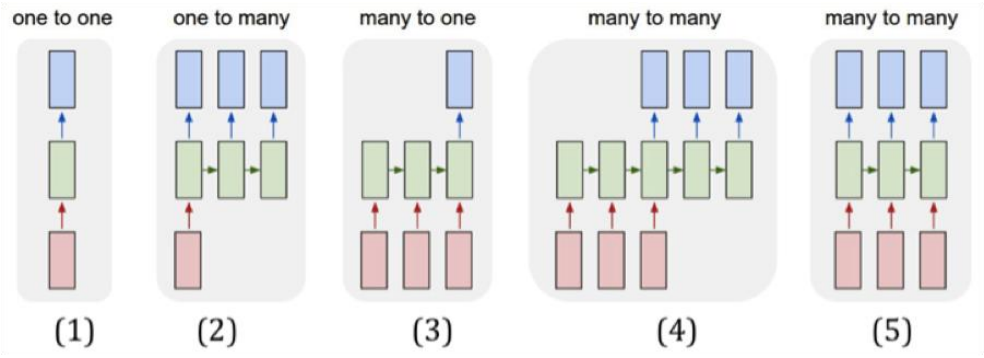


Figure 3-5 Architecture classes based on input-output data [107].

3.3.1 Recurrent Neural Networks

The architecture of a RNN is displayed in Figure 3-6. At each time step t, the input $\mathbf{x}_t$ is fed into a cell. This cell comprises a certain number of units and receives inputs $\mathbf{x}_t$ and information from the previous step $t-1$. This information is operated to generate a hidden state for the timestep t ($\mathbf{h}_t$). The hidden state is then passed through the next cell, generating the hidden state for the succeeding step, and can also be processed for an output layer $\mathbf{o}_t$ to produce features for $\mathbf{x}_t$ that, to a certain extent, consider information from previous time steps. In this manner, the generated features take into account the context of the inputs. This process is repeated until T number of steps have been completed.

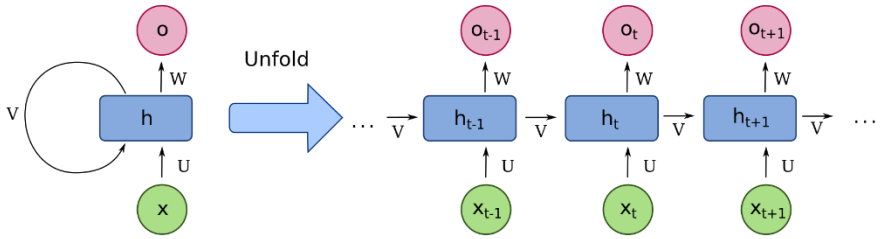


Figure 3-6 Architecture recurrent neural network.

A key factor influencing the efficiency of an RNN in processing data is the type of cell. Figure 3-7illustrates common cell architectures used in DL. The mathematical expression for a Simple RNN, or Elman network, can be seen in Equation 6. This architecture, proposed by Elman in 1990, marked a significant milestone in RNN development [108]. However, the practical applications of this cell are limited due to its susceptibility to vanishing and exploding gradient problems. Vanishing gradient happens when gradients have small values during training. As a results, these values become much smaller during back-propagation, which hinders neural networks from learning effectively. On the other hand, gradient explosion exhibits the opposite behaviour; gradients with large values tend to grow rapidly. This instability in gradients can prevent model from converging.

$$\begin{aligned} \mathbf{h}_t &= f(\mathbf{x}_t, \mathbf{h}_{t-1}) \\ &= \tanh(\mathbf{U}_h \mathbf{h}_{t-1} + \mathbf{W}_h \mathbf{x}_t + \mathbf{b}_h) \end{aligned} \tag{6}$$

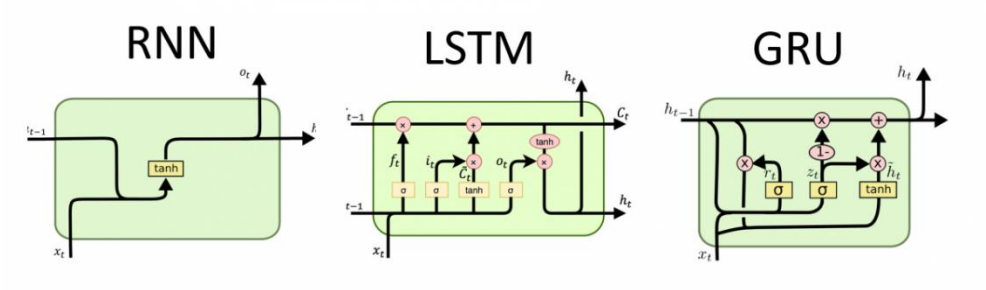


Figure 3-7 Diagram of the most common types of cells used in RNN. From left to right, Simple RNN, Long-short term memory (LSTM), and gated recurrent unit (GRU) [107].

To mitigate Elman network weaknesses, long short-term memory (LSTM) and gated recurrent unit (GRU) cells were proposed. LSTM was designed in 1997 with various improvements [109]. The mathematical formulation is described in Equations 7-12. In summary, LSTM consists of 3 gates: an input gate $\mathbf{i}_t$, a forget-gate $\mathbf{f}_t$, and an output gate $\mathbf{o}_t$. These three components receive inputs including the hidden state from the previous cell and the current input, computed independently. Each gate is activated with a sigmoid function to scale values between 0 and 1. These calculate the memory state $\mathbf{c}_t$, which determines how much information from the previous and new memory states is retained or forgotten for use in the next cell. Finally, the new hidden state $\mathbf{h}_t$ is calculated, serving as output along with $\mathbf{c}_t$. While both $\mathbf{h}_t$ and $\mathbf{c}_t$ are propagated towards the next steps, $\mathbf{h}_t$ values can also be used to feed further layers. As a remark, $\odot$ operator represents element-wise multiplication, also known as Hadamard product.

$$\mathbf{f}_t = \sigma(\mathbf{W}_f \mathbf{x}_t + \mathbf{U}_f \mathbf{h}_{t-1} + \mathbf{b}_f) \quad (7)$$

$$\mathbf{i}_t = \sigma(\mathbf{W}_i \mathbf{x}_t + \mathbf{U}_i \mathbf{h}_{t-1} + \mathbf{b}_i) \quad (8)$$

$$\mathbf{o}_t = \sigma(\mathbf{W}_o \mathbf{x}_t + \mathbf{U}_o \mathbf{h}_{t-1} + \mathbf{b}_o) \quad (9)$$

$$\tilde{\mathbf{c}}_t = \tanh(\mathbf{W}_c \mathbf{x}_t + \mathbf{U}_c \mathbf{h}_{t-1}) \quad (10)$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t \quad (11)$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh \mathbf{c}_t \quad (12)$$

LSTMs have outperformed simple RNNs in various applications. However, it requires a larger number of parameters to be learnt. In consequence, LSTMs are more computationally expensive, especially when dealing with long sequences. Aiming to address both the limitations of Elman networks and LSTM computational cost, GRUs were introduced by Cho et al in 2014 [110]. The calculations for GRU are illustrated in Equations 13-16. This cell makes use of an update gate z_t and a reset gate $\mathbf{r}_t$. The first gate defines how much information is updated, while the reset gate establishes how much information to omit or forget. Unlike LSTMs, GRU cell generates a single new hidden state $\mathbf{h}_t$.

$$\mathbf{z}_t = \sigma(\mathbf{W}_z \mathbf{x}_t + \mathbf{U}_z \mathbf{h}_{t-1}) \quad (13)$$

$$\mathbf{r}_t = \sigma(\mathbf{W}_r \mathbf{x}_t + \mathbf{U}_r \mathbf{h}_{t-1}) \quad (14)$$

$$\tilde{\mathbf{h}}_t = \tanh\big(\mathbf{W}_h \mathbf{x}_t + \mathbf{U}_h (\mathbf{r}_t \odot \mathbf{h}_{t-1})\big) \quad (15)$$

$$\mathbf{h}_t = (1 - \mathbf{z}_t) \odot \mathbf{h}_{t-1} + \mathbf{z}_t \odot \tilde{\mathbf{h}}_t \quad (16)$$

Overall, Both LSTM and GRU have demonstrated a superior performance compared to simple RNN [110]. Nonetheless, there is no consensus regarding which cell offers a better performance. Therefore, the choice tends to be case-specific in this regard [110,111]. On the other hand, from a computational cost perspective, using GRU cells can be trained faster than LSTM in large datasets or complex architectures, which in many cases can guide the decision [111].

3.3.2 Transformers

Although RNNs have been a significant advancement in the modelling of sequential data with neural networks, they still possess some aspects to improve. The sequential nature of RNNs, where the calculation of step t depends on the previous step $t-1$, imposes substantial computational demands when training on large datasets. In response to this limitation, Transformers architecture was developed. Unlike RNNs, Transformers do not

require processing a sequence step by step; instead, it can process the entire sequence at once. This is achieved by introducing two types of layers: positional embeddings (PE) and attention mechanism to replace recurrent cells. PE capture information on a step position in a sequence, while attention estimates the dependencies of the steps within and between sequences.

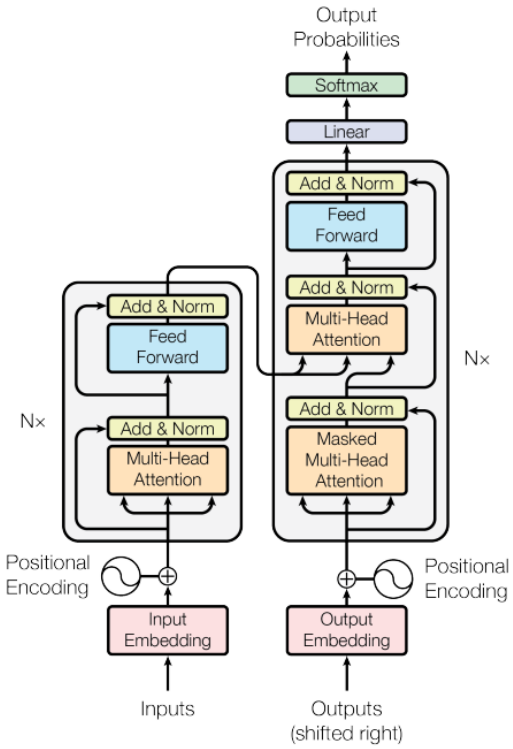


Figure 3-8 Transformer Architecture. Taken from Vaswani et al [112]. On the left, the encoder module can be seen which includes two sublayers self-attention, and a feed-forward network. On the right, the decoder is similar to the first sublayers, but instead of feed-forward, a cross-attention sublayer is introduced. This sublayer is fed with information on the encoder output and self-attention of the sequence. Then, the results are input into another feed-forward network that produces a representation used for a particular task.

Figure 3-8 shows the architecture of the transformers model. This network has two main layers an encoder and a decoder. Both layers receive the sum between token embeddings and positional embeddings for a sequence and perform *self-attention* by using a scaled-dot product as shown in Equation 17. In self-attention, the matrices Q, K, and V correspond to the same input-embedded sequences [112]. On the other hand, in the decoder, cross-attention is performed. In this case, K and V correspond to the encoder output, while Q is derived from the output of the self-attention in the first sublayers of the decoder [112]. As mentioned previously, attention mechanism learns dependencies

between sequence steps, producing scores that fall into the range of 0 and 1. The attention scores quantify the relationship among the steps of a sequence and allow the model to decide what steps to focus on.

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^{\mathrm{T}}}{\sqrt{\mathrm{d_k}}}\right)\mathbf{V} \tag{17}$$

All these changes have accelerated the training process, allowing the use of large amounts of data. Consequently, transformers architecture has become the main driver in the development of large language models (LLM). LLMs are transformers pre-trained on extensive corpora usually, in a self-supervised fashion, to predict the next word or sentence based on previous context [113]. Then, LLM weights can be reuse by fine-tuning the parameters for specific applications. These models are characterised by having a vast number of parameters learnt. For instance, GPT-3, a LLM trained by OpenAI, has approximately 135 billion parameters [113]. In practice, training LLM from scratch demands substantial technical resources in terms of data and hardware [113]. However, this limitation has been addressed using approaches such as model fine-tuning and transfer learning for specific tasks, which has facilitated their widespread adoption.

3.4 Deep Generative Models

Deep generative models (DGM) comprise neural network architectures that aim to estimate the probability distribution of a set of variables for a given set of examples [47,66]. Among the most frequently deployed models, autoregressive approaches (AR), variational autoencoders (VAE) and generative adversarial networks (GAN) can be pointed out. Overall, these models aim to obtain a generator function g such that $g(\boldsymbol{z}) \approx \boldsymbol{x}$, where $\boldsymbol{x}$ represents samples of interest that are independent and identically distributed (i.i.d.), and $\boldsymbol{z}$ is a vector of latent variables that can map values from $\mathbf{x}$ through the generator g, whose probability distribution is approximately $p_\theta(\boldsymbol{x})$ [66].

Regarding the usage of DMGs for sequence generation, a range of models have been reported. To the best of our knowledge, most of these have focused on text generation. More related to the pharmaceutical domain, applications in the generation of drug candidates represented as SMILES or molecular graphs can be found [114,115]. Although not in the generative domain, Transformers have been used to predict synthesis

procedures from reaction SMILES [43]. As can be seen, given the nature of manufacturing sequences, models applied for sequences of discrete variables, such as text or sequential molecular representations, can be adapted for pharmaceutical procedure generation. This work will primarily focus on investigating the use of VAE and GAN, as they represent some of the most extensively studied approaches in this context. In addition, there exists a gap in exploring how well these approaches perform in manufacturing sequence generation. Below, different approaches will be discussed further than GANs and VAEs, however, the present work will focus on these two models.

3.4.1 Generative Adversarial Networks

GANs were introduced by Goodfellow et al in 2014 [67], having shown remarkable success in image generation. Unlike VAE, GANs do not require an explicit specification of a probability distribution or rely on additional assumptions [47]. The architecture consists of a generative network $g_\theta(\boldsymbol{z})$ and a discriminative network $d_\phi(\boldsymbol{x})$, both of which are trained concurrently [68]. The first component $g_\theta(\boldsymbol{z})$ creates samples in $\boldsymbol{x}$-space, while the discriminative network $d_\phi(\boldsymbol{x})$ distinguishes between real (1) or fake (0) samples. The parameters of $g_\theta(\boldsymbol{z})$ are adjusted to maximise the probability of the generated samples be classified as real [68]. The adversarial training process is illustrated in Figure 3-9.

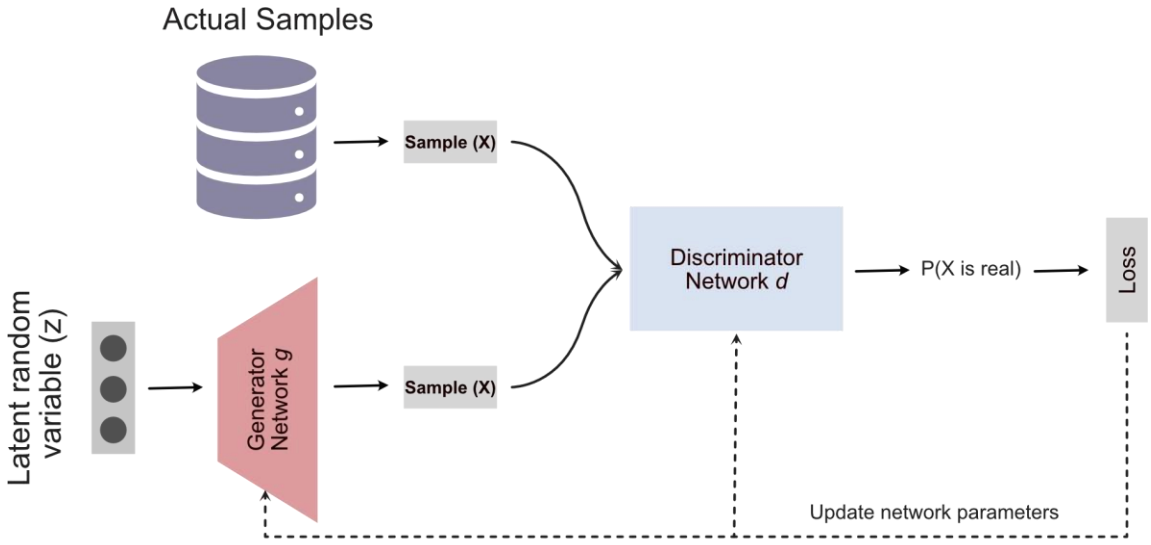


Figure 3-9 GAN architecture for generator and discriminator networks during training. Taken and adapted from Alom et al [69]

The objective function depends on both discriminative and generative network $f(g_\theta, d_\phi)$ and is defined as shown in Equation 18 [67]:

$$\min_{\theta} \max_{\phi} f(g_\theta, d_\phi) = \mathbb{E}_{\boldsymbol{x}\sim p_{data}(\boldsymbol{x})}\left[\log d_\phi(\boldsymbol{x})\right] + \mathbb{E}_{\boldsymbol{z}\sim p(\boldsymbol{z})}\left[\log(1 - d_\phi\left(g_\theta(\boldsymbol{z})\right)\right] \quad (18)$$

The optimal value is reached at a saddle point where the discriminator is not able to distinguish between real and generated samples [66]. While the model performance is generally good, training and tuning can be complex since the system can be unstable and prone to collapse [47]. To tackle training difficulties, various modifications have been adopted into the loss function. In this regard, a widely used variant is Wasserstein GANs with Gradient Penalty (WGAN-GP) [70,71]. Wasserstein's loss is illustrated in Equation 19. In WGAN, one of the main changes is that the discriminator network does not output probabilities directly, instead, this network is trained to yield unbounded values that represent the "quality" of the input sample [71]. As a result, the discriminator network, in this context, is also known as critic and measures the distance between probability distributions [71].

$$\max_{\phi} L_{WGAN} = \mathbb{E}_{x\sim p_{data}(x)}\left[d_\phi(\boldsymbol{x})\right] - \mathbb{E}_{z\sim p(z)}\left[d_\phi\left(g_\theta(\boldsymbol{z})\right)\right] \quad (19)$$

For sequence generation, GANs poses additional challenges due to the discrete nature of the outputs. Normally, sequence generation models include an output layer that assigns probabilities to each possible token or operation, in our case, at every time step. In this manner, the sequence can be constructed step by step by sampling the operation with the highest probability. In adversarial networks, this output is then fed into a discriminator for training the networks. However, this procedure of sampling is non-differentiable, thereby hindering weights from being updated through backpropagation [116]. To overcome this, various strategies to use adversarial training have been proposed. These can be oriented to modify network architecture to produce a continuous and differentiable output, while others adjust the loss function using reinforcement learning (RL). Architectural-based approaches, for instance, may employ techniques such as relaxing the sampling procedure by applying Gumbel distribution or incorporating autoencoders (AE) [116,117].

AEs are commonly used architectures in deep learning applications [118]. These models enable data compression into a continuous latent space through an encoder [118]. In turn,

this projection can be restored back into the original data by a decoder [118]. This approach allows sequences to be represented as a continuous vector that is differentiable and can serve as an input into a discriminator network, facilitating adversarial training. In addition, by knowing the distribution of the latent space, samples can be generated in this space and, subsequently, transformed into sequences by the decoder. Building upon this concept, architectures such as adversarial autoencoders (AAE) and adversarially regularized autoencoders (ARAE) have been proposed [119–121].

3.4.2 Variation Autoencoders

VAEs belong to the family of autoencoders that learn a latent space that follows a specific probability distribution. In this framework, the encoder is known as recognition model, and projects a sample $\boldsymbol{x}$ onto a latent space $\boldsymbol{z}$ [65]. Next, the decoder, or generative network, uses as input the latent projection and tries to reconstruct $\boldsymbol{x}$ [65]. To point out, the dimensionality of the latent space $\boldsymbol{z}$ dimensions is lower than input data $\boldsymbol{x}$, by which this technique can also be used as a dimensionality reduction approach. [65,66].During training, VAE learns a joint distribution $p_\theta(\mathbf{x}, \mathbf{z})$ and which can be decomposed into $p_\theta(\mathbf{x}, \mathbf{z}) = p(\mathbf{z})p_\theta(\mathbf{x}|\mathbf{z})$, where the conditional distribution $p_\theta(\mathbf{x}|\mathbf{z})$ acts as a decoder model and $_\theta p(\mathbf{z})$ is a prior distribution that characterises the latent space [65]65. The encoder $q_\phi(\mathbf{z}|\mathbf{x})$ is a function that approximates the posterior $p_\theta(\mathbf{z}|\mathbf{x})$ since this distribution cannot be estimated given its intractability [65,66]. Finally, the loss function is known as the evidence lower bound (ELBO) and is defined as shown in Equation 20. [65]. The first term in the loss function maximised the reconstruction accuracy, while the second term, corresponding to Kullback-Leibler divergence, minimises the difference between the prior and the learnt posterior distribution of $\boldsymbol{z}$.

$$L_{ELBO} = \mathbb{E}_{z\sim q_{\phi(Z|X)}}[\log p_\theta(x|z)] - D_{KL}\left[q_\phi(z|x)||p(z)\right] \quad (20)$$

VAEs have also been used to generate sequences. Contrary to GANs, VAEs tend to be easier to train and less prone to mode collapse. However, when training on sequential data, VAE may suffer from a problem known as posterior collapse. This occurs when the model disregards the information provided by the latent space, resulting in uninformative sequence representation incapable of reconstructing the input data [122]. Consequently, it tends to produce repetitive and limited data, failing to learn meaningful features for data representation. To tackle posterior collapse, a balance between reconstruction and KL

divergence in the loss function is essential [123,124]. VAEs with this variation in the loss function is referred to as β-VAE, which was introduced by [125]. This approach has been frequently applied in the generation of SMILES sequences in literature [49,126,127].

3.4.3 Energy-based models

Energy-based approaches can be regarded as the first generative models using neural networks. The first architectures dates to 1980s with Boltzman Machines (BM) and Restricted Boltzman Machines (RBM) [64]. In the original formulations, $p_{model}(x)$ depends on an energy function $E(x)$ and a partition function Z, which is also related to $E(x)$, as shown in Equations 21 and 22 [51]. Z acts as a scaling factor to ensure that the sum of all the probabilities equals 1. $E(x)$ can be estimated by using neural networks and its analytical form depends on the architecture. Due to this, there are no constraints regarding the type of data to model [51]. However, a limitation is that Z may not be tractable [51]. In contrast to other models, there is no generative function learnt, by which the generative process may be challenging and require some approximation [51]. Difficulties in Z tractability and sampling restrict their usages in several domains.

$$p_{model}(x) = \frac{e^{-E(x)}}{Z} \tag{21}$$

$$Z = \int_X e^{-E(x)} dx \tag{22}$$

3.4.4 Autoregressive models

In AR models, $p_{model}(\mathbf{x})$ is explicitly defined by decomposing each observation $\mathbf{x}$ as a sequence of variables $x_1, x_2, \ldots, x_D$ [63]. In this manner, the density function can be determined calculating the product of a set of conditional probabilities as illustrated in Equation 23. This approach allows likelihood estimation as the sum of the likelihoods of each conditional probability. Nonetheless, it is evident that under the assumption which allows variable decomposition, it should be possible to express data as an ordered set of variables. Although this is more natural in some scenarios such as modelling of time series or text, it is not a trivial problem in tabular data or images [63]. In the same manner, given that sequences are usually generated step by step, sampling can be slow [51]. Another limitation is related to dimensionality. Higher dimensions increase the number of

terms to estimate, thereby leading to a more complex model [63]. Finally, more common architectures to estimate these probabilities include neural autoregressive density estimator (NADE), masked autoregressive density estimator (MADE), recurrent neural networks (RNN), transformers-based models [51,63].

$$p_{model}(\boldsymbol{x}) = p_{model}(x_1, \dots, x_D) = \prod_{i=1}^{D} p(x_i | x_1, \dots, x_{i-1}) \tag{23}$$

3.4.5 Flow-based models

Flow-based models or normalising flows (NF) is another approach that can estimate exact density without approximations. NF also can be classified as a deep latent variable model (DLVM) since it assumes a random variable $\boldsymbol{x} \sim p(\boldsymbol{x})$ can be obtained by transforming a group of latent variables $\boldsymbol{z} \sim \boldsymbol{p}(\boldsymbol{z})$ through a generator. Considering a smooth and invertible function $f: \mathbb{R}^d \rightarrow \mathbb{R}^d$, $\boldsymbol{z}$ and $\boldsymbol{x}$ are related in the following manner $\boldsymbol{z} = f(\boldsymbol{x})$ [65,51]. As a result, $p_{model}(\boldsymbol{x})$ can be derived through the change of variable rule using the density function $p(\boldsymbol{z})$ as can be seen in Equation 24 [51]. Thus, $p(x)$ corresponds to $p_z(z)$ adjusted using the absolute value of the determinant of the Jacobian matrix of f^{-1} with respect to x with f being parameterised using a deep neural network. Nevertheless, while it has been remarked that NF are powerful density estimators, the invertibility of f only permits the use of a reduced type of neural networks [63]. In the same way, another condition for their application is that $\boldsymbol{x}$ and $\boldsymbol{z}$ should have the same dimensionality [66]. This fact affects the scalability of NF in high dimensional data and reduce the control over the latent space. Thus, although several modifications have been introduced to make NF more flexible, their applicability in several domain is an area that seems to require further development.

$$p_{model}(\boldsymbol{x}) = p_z\big(z = f(x)\big) \left| \det \left(\frac{\partial f^{-1}(z)}{\partial x} \right) \right| \tag{24}$$

3.4.6 Performance Metrics in Generative Models

Generative and discriminative models differ significantly in terms of evaluation due to the absence of a reference or ground truth. Since DGMs goal is to produce realistic samples that are not necessarily equal to the training set, they cannot be compared directly to the reference set to estimate error as in classification/regression models. In the evaluation of

generative models, domain-specific metrics are often employed, i.e., the choice of metrics depends on the model and the type of data being generated. For autoencoders, assessing reconstruction accuracy from the latent is a common practice. This helps to determine how well the latent space is related to the data and how reliable a model is in generating data. In sequential data, metrics like BLEU and Levenshtein distance can be used to measure reconstruction [128,129].

On the other hand, with only reconstruction, it is not possible to ensure the quality of generated examples. Reconstruction is limited to see whether a decoder can restore an example to its original form using example projection onto latent space. In this case, there exists a reference, which does not occur when new samples are generated. Thus, in several domains, specific metrics have been proposed to assess high-level properties that indicate that the model generates examples that belong to a similar distribution to the dataset of interest. A more concrete example of this is in drug discovery. Multiple approaches have been used to formulate indicators of the validity of a newly generated molecule. Along the same line, it is often sought that generated samples are novel and relevant to the domain.

In this work, since autoencoders are used, BLEU and Levenshtein similarity are used to measure reconstruction accuracy, both of which are explained in more detail below. On the other side, other attributes of manufacturing sequences related to sequence validity are explored based on logical rules that are discussed in Chapters 6 and 7, for primary and secondary manufacturing, respectively.

3.4.6.1 Bilingual Evaluation Understudy (BLEU)

BLEU score is a metric widely used in machine translation and NLP [128]. It is applied in the evaluation of similarity between generated and reference sequences. This metric computes n-grams precision, that is the ratio between the matching n-grams in generated and reference sequences, and the total number of n-grams in the reference. A n-gram is a subsequence of n consecutive steps drawn from a sequence. For instance, the sequence [“ADD”, “STIR”, “HEAT”, “PURIFY”] can be split into 2-grams in the following manner: [“ADD”, “STIR”], [“STIR”, “HEAT”], and [“HEAT”, “PURIFY”]. In the standard BLEU score, individual precisions are determined for up to 4-grams and the geometric mean between the four ratios is calculated. Finally, this is multiplied by a factor that considers the difference in length between candidate and reference. BLEU is a bounded

metric between 0 and 1, where 1 means perfect matching between the reference and the candidate.

3.4.6.2 Levenshtein Distance

Levenshtein or edit distance is a metric used to measure the difference between two strings. However, it has also been used to compare sequences like DNA or manufacturing actions [43,129]. Lev counts the number of insertions, deletions or substitutions needed to convert a sequence to the other. This score is usually normalised by dividing the counts by the lengths of the largest sequence between reference and candidate. In the same manner, lev can be expressed between 0 and 1. The similarity can then be measured by subtraction between 1 – normalised distance. Lev distance focuses mostly on comparing individual components in a sequence, whereas BLEU considers subsequence thereby considering order and length.

3.5 Equipment and Software

The specifications of equipment and software used for data analysis and model development are recorded in Table 3-1.

Table 3-1. Libraries and equipment used for data analysis.

Model	Software	Library	Ref.	Equipment
Non-negative matrix factorization (NMF)	Python 3.8	Scikit-learn 1.1	[130]	Processor: Intel(R) Xeon(R) CPU E5-2667 v3 @ 3.20GHz 3.20 GHz (12 cores), Memory: 128 GB RAM installed, Graph card: NVIDIA Quadro K6000 GPU Memory 12.0 GB
Latent Dirichlet Allocation (LDA)	Python 3.8	Gensim 4.1	[131]	
FasText embeddings	Python 3.8	Genism 4.1	[131]	
Named Entity Recognition	Python 3.8	Tensorflow 2.8 (deep neural network)	[132]	Processor: Intel(R) Core(TM) CPU i9-

				12900K @ 3.20GHz 3.20 GHz (12 cores), Memory: 64 GB RAM installed, Graph card: NVIDIA GeForce RTX3090 GPU Memory 24.0 GB
		Spacy 3.3 (text processing)	[97]	
		Chemdataextractor (text processing)	[133]	
Dataset cleaning and curation	Python 3.8	Spacy 3.3.	[97]	
		Chemdataextractor	[133]	
		Scikit-learn 1.1	[130]	
		Pubchempy	[99]	
	Pipeline Pilot			
Deep Generative Models	Python 3.9	Tensorflow 2.11	[132]	
Visualization	R 4.1			

Chapter 4. Data collection

4.1 Introduction

Over time, information on the design and development of medicines has undergone an intensive democratization process. During this process, a vast amount of data has been made accessible by the private sector, regulatory agencies and academic institutions. As a result, a large amount of information is available to be consulted in digital sources [134]. Information available covers the whole pharmaceutical product lifecycle, ranging from design and development of new chemical entities to pharmacovigilance and safety aspects [134–136]. Consequently, pharmaceutical information can be said to be varied and abundant. However, another important feature is that much of this information is unstructured [134,136]. This means that data are usually not found in a tabulated and organized way such that it cannot be easily used for analysis and machine learning development. This leads to the capture and selection of relevant information on a large scale that cannot be done manually, instead, requires a certain degree of automation. It is at this point that natural language processing (NLP) comes into play.

In the last decades, there have been many initiatives to use NLP to mine data from unstructured documents and build usable databases for diverse fields. In particular, the biomedical field can be highlighted to have numerous related datasets obtained through text mining [137]. For example, Roche Diagnostics constructed a disease marker dataset by extracting information from 50 million abstracts [138]. Other examples are the development of drug-drug interactions data sets and methodologies to extract drug side effects [139,140]. In turn, all these data sets have provided comprehensive access to information and could enable the use of artificial intelligence (AI) to accelerate the discovery of new molecules and improve therapies for patients [138]. In pharmaceutical manufacturing, most works building databases has focused on primary processing for the extraction of chemical reactions and materials. Nonetheless, independently of the field, to build any of these applications enabling data extraction, a fundamental starting material is a corpus.

A corpus comprises a large collection of documents focused on a specific domain [141]. As a major input for the development of NLP applications, it is usually found that many resources have been allocated on gathering and curating documents to have reliable corpora to be used. Thus, similarly to the aforementioned datasets, annotated corpora

have been made available to serve as gold standards and aid NLP tasks. Aligned with this, examples in primary preprocessing can be found like a chemical reaction corpus built by Cheminformatics Elsevier Melbourne University (ChEMU) in 2020 and chapati corpus, which is a product of a collaboration between the European Patent Office (EPO) and ChEBI (Chemical Entities of Biological Interest) organization [94,142]. The former uses complete patents and includes annotations of materials, operations and conditions, whereas the latter provides information on only chemical substances [94,142]. These corpora have been used to develop named entity recognition tools which enable the automatic extraction of information from multiple documents. Thus, it is possible to notice how significant a corpus is as a starting point for NLP applications.

Regarding corpora development for pharmaceutical manufacturing, most efforts have been concentrated in the primary manufacturing and biomedical domains. However, to the best of our knowledge, there are no corpora available for secondary manufacturing. As to primary processing, while there exists a gold standard developed by ChEMU team [94] as previously discussed, this is limited to a few hundred patents and was developed for a specific task. On the other hand, with the current available patents, a larger dataset could potentially be built, nonetheless, it is necessary to establish a sampling frame of documents which can be used to train generative models. Considering these gaps, the present work collects pharmaceutical patents containing information related to either primary or secondary manufacturing to be used in downstream NLP modelling and dataset construction for training generative models. This task is accomplished as can summarised in Figure 4-1 by web scraping documents from the available web application programming interfaces (API) of the United States Patents and Trademark Office (USPTO) and using model-based approaches to select the information of interest. Finally, content relevance is validated by identifying main topics and keywords from the selected files. The scope of this work is limited to patents obtained from USPTO due to ease of accessibility and high availability of literature for information extraction in this type of documents.

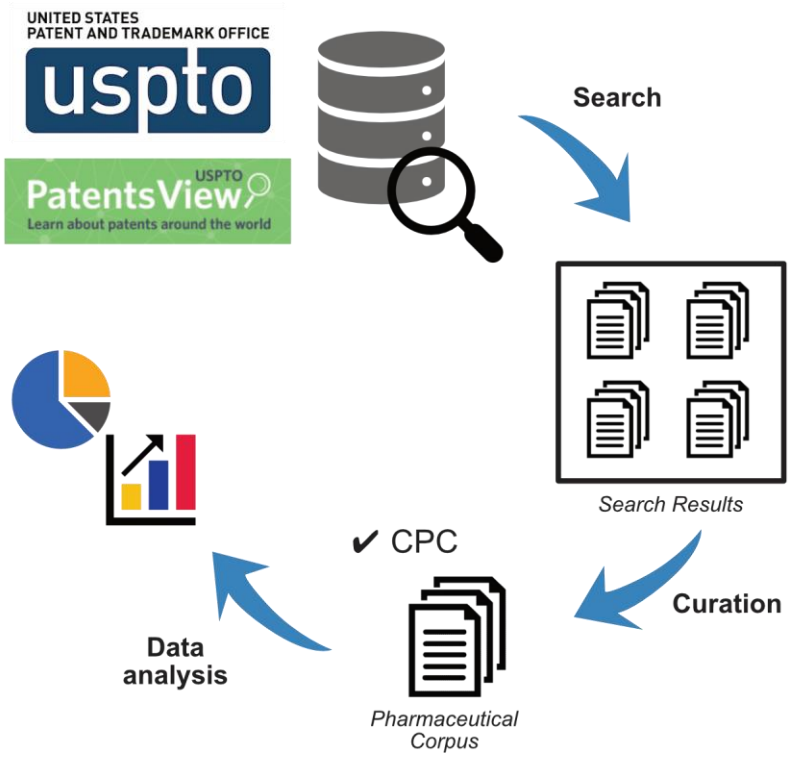


Figure 4-1. Overview pharmaceutical corpus development.

4.2 Methods

4.2.1 Search strategy and documents retrieval

All the granted patents between January 1976 and September 2021 were downloaded from United States Patent and Trademark Office website (https://www.uspto.gov/). These patents were available in XML and TXT format. To select the patents that were pharmaceutically relevant, search results were scraped from the following websites: Bulk Search and Download API (uspto.gov) and PatentsView. The terms employed in the searches included the official dosage forms established by the FDA and the synonyms of drug substances obtained from DrugBank database [143,144]. Some terms were excluded from the latter list. The criteria of exclusion for synonyms can be seen in Table 4-1. These terms were discarded since they corresponded in most cases to acronyms that may have several meanings depending on the field. In this manner, if these were included, irrelevant results would have been obtained.

Table 4-1 Criteria of exclusion for search terms.

Criteria	Regular expression	Example of Excluded Terms
Numeric expressions	"^\d+[\.\,]?\d+$"	33355
Alphanumeric expressions with fewer than 3 consonants	"^[a-z]{1,3}\-?\d+$"	bb1111
Terms with more than 10 words	N/A	HAEMOPHILUS INFLUENZAE TYPE B CAPSULAR POLYSACCHARIDE MENINGOCOCCAL OUTER MEMBRANE PROTEIN CONJUGATE ANTIGEN
Term with fewer than 5 characters	N/A	E265

After filtering, there were a total of 43,538 search terms. These were distributed in 157 and 43,381 corresponding to dosage forms and active ingredients synonyms, respectively. The search was automatised for each term on both websites using Python 3.8. The search criterion in USPTO API was based on the appearance of search terms in patent claims, while the presence in abstract was applied for PatentsView. The results were stored and preprocessed. The preprocessing consisted of removing duplicates and incomplete records. In the final list, patent number, and Cooperative Patent Classification (CPC) - when available - were stored. The patent number was subsequently used to trace back and extract abstracts, claims, and detailed descriptions from XML/TXT files.

CPC is a classification system that assigns a category to a patent depending on the technical field. In this system, patents are divided into 9 sections (A-H and Y). In turn, every main section is subdivided in classes, subclasses, groups, and subgroups. As a result, an alphanumeric code is assigned. An example of this is illustrated in Figure 4-2. A description of every main category is displayed in Table 4-2. It is worth noting that a patent may fall into more than one category. Thus, CPC subclasses A61K *Preparations for Medical, Dental, or Toilet Purposes*, A61P *Specific Therapeutic Activity of Chemical Compounds or Medicinal Preparations*, and A61Q *Specific Use of Cosmetics or Similar Toilet Preparations* were considered as relevant since they may contain information on either primary or secondary manufacturing.

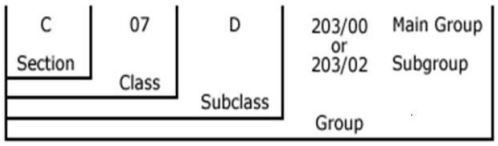


Figure 4-2 Example CPC scheme for patent technical classification. Taken from USPTO [145]

Table 4-2 CPC Scheme sections. Taken and adapted from USPTO [145]

Section	Technical Field
A	Human necessities
B	Performing operations; transporting
C	Chemistry; metallurgy
D	Textiles; paper
E	Fixed constructions
F	Mechanical engineering; lighting; heating; weapons; blasting engines or pumps
G	Physics
H	Electricity
Y	General tagging of new technological developments; general tagging of cross-sectional technologies spanning over several sections of the IPC; technical subjects covered by former USPC cross-reference art collections [XRACs] and digests

With the search results, an exploratory analysis of abstracts was carried out to evaluate how related the patents were to the intended content. Thus, keywords were extracted by using term-frequency inverse document frequency (*tf-idf*) technique. The abstracts were preprocessed as indicated in section *4.2.2.1*. Words that were presented in fewer than 10 documents were discarded. This algorithm was run using scikit-learn library. The top 50 most important terms based on *tf-idf* scores were revised to define whether or not the selected patents were pharmaceutically relevant.

4.2.2 Patents selection

Although relevant search results were selected by means of CPC, not all of them had a category assigned. Therefore, a patent classifier was developed to select pharmaceutical

documents from those whose CPC was not available using machine learning. As a training set, all the patents retrieved with CPC were employed. As input in model development, abstracts and claims were evaluated individually. 70% of the data was used to train the model. The classifier performance was assessed using the remaining 30%. Usually, text has to go through several preprocessing steps to be converted to a numerical representation. Then, conventional machine learning models are applied to achieve the task. In the following sections, these steps are described in detail.

4.2.2.1 Text Preprocessing

The preprocessing steps comprised converting text to lowercase, tokenization, removing stop-words and punctuation, and normalising words [146,147]. In tokenization step, text is split into a list of meaningful tokens, which are substrings that can be words and punctuation [146,148]. Then, words that did not provide relevant meaning, also known as stop-words [146], and punctuation were discarded. Examples of stop-words include articles such as “the” and “a”, or common verbs such as “is” and “have”. A complete list of these words was obtained from spacy 3.0 library [97]. Additionally, the words “hereby” and “said” were included.

Words normalization consists of transforming words into their base form to reduce redundance and ease analysis [146,147]. In this manner, morphological changes of a word or inflections done to express, for instance, tense or number, are removed [146]. This task is normally accomplished by the application of two approaches: stemming and lemmatization. Stemming, through a set of rules, changes a word into a stem form, without considering context [146]. Depending on the rules employed, there exist several stemmers, being Porter and Snowball some of the most popular [146,148]. To illustrate the result, the strings “formulation”, “formulated” and “formulate” will be converted to the stem “formul”, using Porter stemmer. On the other hand, lemmatization is dependent on the context. This technique considers word syntactic function (verb, noun or adjective) in a sentence to convert a word to its base form, in this case known as lemma [146]. To exemplify this method, we will consider the following sentences:

- “The active ingredients may be ***formulated*** as compositions containing several active ingredients in a single dose form and/or as kits containing individual active ingredients in separate dose forms.”,
- “The following can be used as additional ***formulation*** aids”, and

- "The invention discloses a method for controlling a production process of a *formulated* product".

While the word "formulation" is a noun, "formulated" can be an adjective or a verb. By lemmatizing, the lemma of "formulation" will be the same word. On the other, for the string "formulated", there will be two possible outputs: "formulate", when it is a verb, and "formulated", in the other case. As can be seen above, both methods provide different results. A major drawback seen with stemming is that stems may not always be actual words [148]. This makes legibility more difficult, for instance, in scenarios when keywords are required like topic modelling [148]. Nonetheless, lemmatization may be less robust as it is sensitive to the method used to identify word functionality in a sentence [148]. In the end, abstracts and claims were lemmatized considering the advantage that lemmatization offers in terms of interpretability.

4.2.2.2 Numerical representation

Numerical representations for textual data provide a notion of similarity between texts and enable the use of textual data in machine learning modelling. There exist several ways to extract features from this kind of data which include count-based representations and document/word embeddings [146,147]. For the selection of the most convenient representation for classification, term frequency – inverse document frequency (*tf-idf*) matrix, fasttext and doc2vec representations were assessed. The parameters employed to calculate every representation can be seen in the Table 3. The packages employed to learn these representations are described in section 3.5. These techniques are discussed in more detailed in the next subsections.

Table 4-3 Hyperparameters for assessed text representations training. Each representation was determined for abstracts and claims, separately.

Representation	**Parameters**
tf-idf	min_df = 10
Doc2vec[149,150]	dm=0, dbow_words=1, vector_size=300, negative=10, min_count=10, sample = 1e-5, workers=cores, window= 20
Fasttext[151,152]	window = 20, min_count = 10, sample= 1e-5, ngram= (3, 6), alpha= 0.05, epochs = 20

4.2.2.2.1 Count-based representations

Count-based methods build a frequency matrix of all the words in a corpus. These methods include bag-of-words (BoW) and term frequency – inverse document frequency matrix (*tf-idf*) [146,147]. In BoW, a document is represented by a vector whose number of elements equals the number of unique words in the corpus. In turn, each element represents the number of times a specific term is found in a document. In the end, a matrix containing as many columns as unique words and as many rows as documents is formed. This approach has as a disadvantage that tend to overestimate the importance of the most recurrent words in the corpus [147].

Nonetheless, this drawback can be mitigated by using *tf-idf* matrix [147]. In this method, term frequency (tf) is first defined in relative terms as a function of the total number of words in the document. Then, tf is multiplied by a correction factor which is inversely proportional to the number of times that a word appears in all the corpus. In this manner, when a term is present in many documents, it will have a low value for this factor. By doing this, those terms that are very frequent and do not provide much information are assigned a lower value. The calculation is detailed in the section 3.2.1[146].

Finally, for this work, *tf-idf* representation was selected for assessment over bag-of-words since ponders terms more accurately [146,147]. However, although these approaches are relatively easy to calculate, they are not able to represent words semantics and context properly [147]. Furthermore, these methods are limited by the vocabulary of the training set, thereby ignoring information given by unseen words or out-of-vocabulary (OOV) in new documents to classify. To tackle some of these limitations, word embeddings have shown very promising results.

4.2.2.2.2 Text embeddings

In 2013, Mikolov et al developed word2vec [88]. This is a self-supervised algorithm based on fully connected neural networks where vectorial representations are generated for each word in a corpus [88]. There are two ways of training this algorithm: continuous bag-of-words (CBOW) and skip-gram (SG) [88]. The architecture for both approaches is illustrated in Figure 4-3. In both methods, the weights generated in the hidden layer of the model for each word provide a relative notion of meaning. This means words that have a similar meaning are expected to have similar values for their respective vectors.

This approach, to a certain extent, considers words semantics [146,147]. In addition, vectors can be used as inputs in machine learning models such as neural networks, individually, or calculating average to represent a sentence or a block of text in other techniques like support vector machines or logistic regression. However, this method does not generate a representation for unseen words. This may be an issue for new documents that contain words not included in the training set or with spelling mistakes. Thus, in 2015, P. Bojanowski et al proposed fasttext [152].

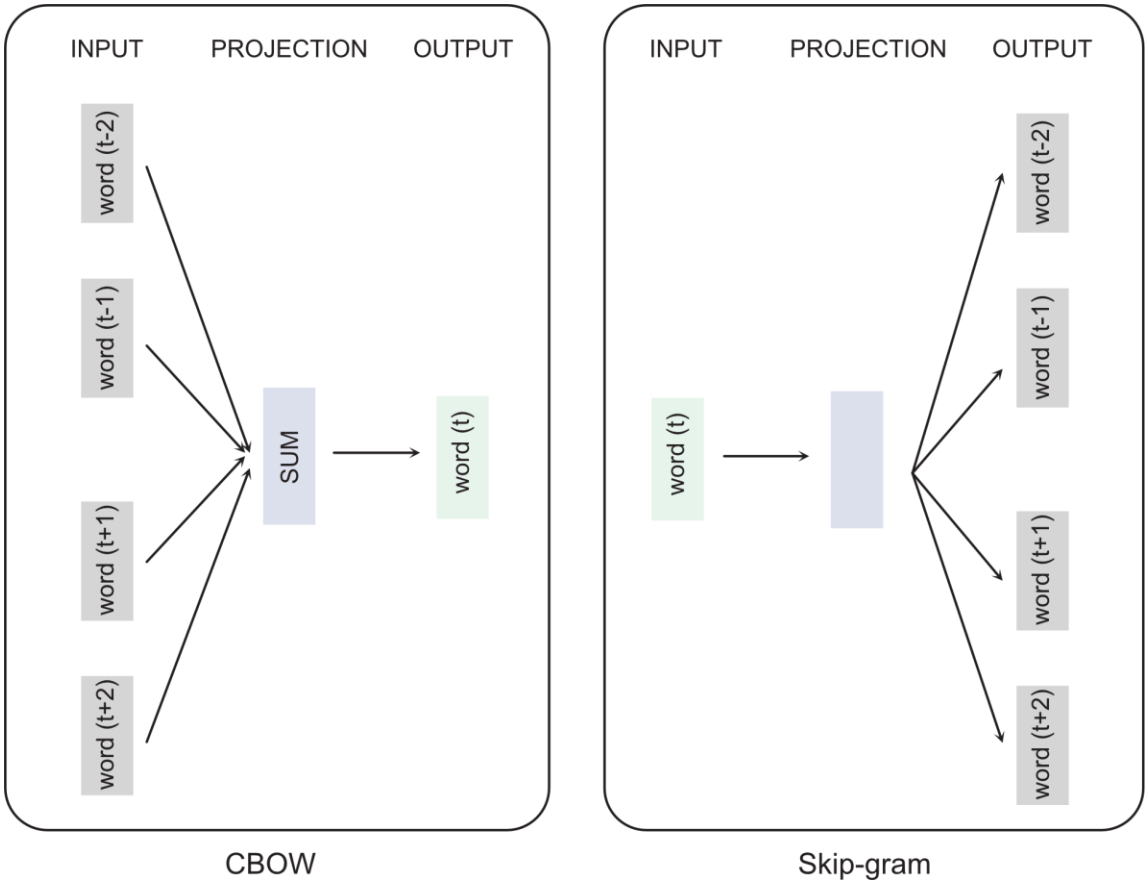


Figure 4-3 word2vec architectures. On the left hand-side continuous bag-of-words (CBOW) and on the right skip-gram (SG). For a sequence of words, CBOW mode will predict word t using as inputs the words t-2, t-1, t+1 and t+2 and word. Whereas SG mode will output the context words t-2, t-1, t+1 and t+2 given the word t. Taken and adapted from Mikolov et al [88]

Instead of using words to generate representations, fasttext uses subwords. In this approach, words are divided into n-grams [152], for instance, the word “octanol” can be split into subwords in the following manner <oct, cta, tan, ano, nol> where n = 3. Then, fasttext generates representations for each individual subword in such a way that a vectorial representation for each word will be equivalent to the sum of the respective subwords

vectors [152]. Like word2vec, there are two modes of training CBOW and SG, whose inputs and outputs are analogous [152]. Moreover, the word vectors generated by fasttext can also be used in the same way as word2vec [152]. Going back to octanol example, it is then expected in the case of another alcohol not included in the training set, to give an example, dodecanol, fasttext can still provide a representation for this word by summing up its subwords vectors. Additionally, this vector maintains similar values compared to resembling words.

Nonetheless, even though these two algorithms can generate good representations for individual words, this might not be the case for blocks of text such as sentences or paragraphs. A usual approach to represent text is to average word vectors [149]. However, the resulting vector may be biased towards the most frequent words and information may be lost upon ignoring sequence order [149,153]. Due to these constraints, algorithms to obtain embeddings for an entire text have been developed. An example of this is doc2vec [153]. This technique follows a similar architecture to word2vec approach [153]. However, in addition to word embeddings, each document is allocated a vector which is concatenated with word vectors to return the outputs [153]. The architecture of doc2vec representation can be observed in Figure 4-4. To conclude this section, to choose the best representation doc2vec and fasttext algorithms were evaluated. Hyperparameters for the training of these models in this work are detailed in Table 4-3.

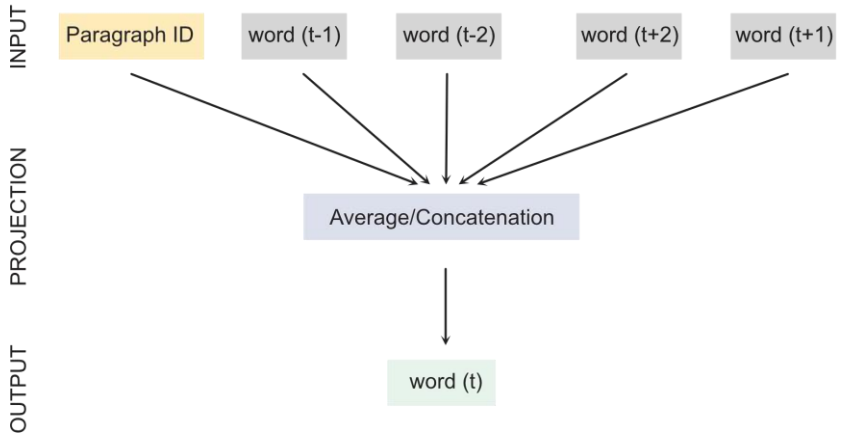


Figure 4-4 doc2vec architecture. Taken and adapted from Le Q et al [153].

4.2.2.3 Machine Learning Methods

In the current work, support vector machines, logistic regression, and recurrent neural networks were assessed for patent selection. These models have been widely used for

text classification [86]. Although there are several alternatives, the main motivation to choose these three models are related to their computational cost and previous reports indicating the use of these for patent classification [84,85,151]. Employed hyperparameters are shown in Table 4-4 and models are described the next subsections. The packages employed for model training are cited in section 3.5.

Table 4-4 Hyperparameters for assessed models. [a] class weights were calculated using scikit learn package for imbalance data. [b] Separate models were trained for abstracts and claims. [c] Recurrent neural networks were trained using pre-trained fasttext and custom embeddings as word representations.

Model	Hyperparameters
Logistic Regression (LR)	class_weight[a], max_iter = 1000, solver = "saga", random_state = 42
Support Vector Machines (SVM)	class_weight[a], random_state = 42, max_iter = 1000
Recurrent Neural Networks (RNN) [c]	spatial_dropout = 0.5, dropout = 0.5, optimizer = adam learning_rate = 0.001 epochs = 20 batch_size = 256, patience early_stopping = 3 epochs class_weight[a]

4.2.2.3.1 Logistic Regression (LR)

Among the selected models, LR is the simplest and fastest model as it has fewer parameters [84]. LR can be used to model the conditional probability of a patent to belong to a specific category ($y = 1$) given a text vectorial representation $\boldsymbol{x}$ and model parameters $\boldsymbol{\theta}$ ($P(y = 1|\boldsymbol{x};\boldsymbol{\theta})$). In this way, the model assigns a probability to a text – abstract or claims – to belong to a relevant ($y = 1$) or irrelevant group ($y = 0$) [86]. However, it does not return directly the class, by which it is necessary to set a threshold to define the category. In Equation 1, the relationship between conditional probability and vectorial representation is shown [86].

$$P(y = 1|\boldsymbol{x};\boldsymbol{\theta}) = \frac{e^{\boldsymbol{\theta}^T\boldsymbol{x}}}{1 + e^{\boldsymbol{\theta}^T\boldsymbol{x}}} \quad (1)$$

4.2.2.3.2 Support Vector Machines (SVM)

SVM projects features onto a hyperplane in so that the separation between classes is maximised [154]. As a result, a boundary can be formed, enabling to set a decision region to classify data depending on the position with respect to the hyperplane [154]. SVM can be linear and non-linear in accordance with a kernel [154,155]. This kernel is an auxiliar function which provides an additional dimension to determine the hyperplane [154]. Linear SVM was selected for this assessment as this approach is more appropriate to handle data of high dimensionality like text representations [155]. In this case, a document with a representation vector $\boldsymbol{x}$ will be classified in the class 1 if $\boldsymbol{\theta}^T\boldsymbol{x} \geq 0$, where $\boldsymbol{\theta}^T\boldsymbol{x}$ is the distance between $\boldsymbol{x}$ and the hyperplane [155].

4.2.2.3.3 Recurrent Neural Network (RNN)

RNN comprise a deep neural network architecture capable of handling sequential inputs like word sequences [156]. Contrary to LR and SVM, in the case of text classification, RNN is not fed using a representation for an entire document, instead, vectors for each word are input into different cells, which process word representations separately. In addition, these cells are interconnected to one another in such a way that the sequence order of the words is maintained. In this manner, not only is individual information of the words processed by a cell, but this cell also receives the output of a previous or posterior cell; therefore, the cell ponders information on surrounding states thereby considering word context to generate a result. Figure 4-5 illustrates a general scheme for a RNN used for classification.

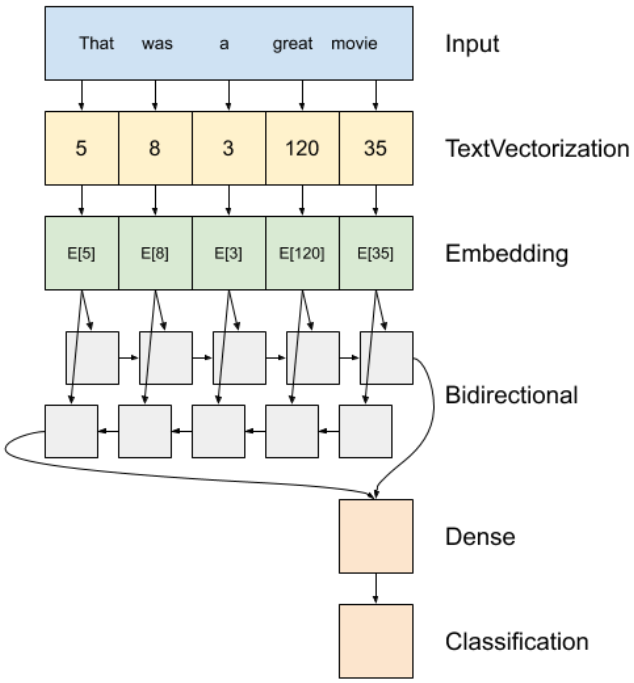


Figure 4-5 General scheme of a RNN architecture for binary classification. Text Vectorization layer assigns an index to each word. This index is used by the embedding layer to locate the corresponding vectorial representation for each word in a lookup table. Then, these embeddings are fed into the RNN. In this case, there are two stacked RNN layers where each process the information in different directions (bidirectional). The output of these layers is subsequently concatenated to be handled by a dense layer. This final layer will outcome the probability of the input text belonging to the category $y = 1$. Taken from Tensorflow website [157]

In the last decade, RNN have undergone a significant increase in popularity due to their performance in natural language modelling [86]. In particular, two kinds of cells have contributed for this approach to outperform other models: long short terms memory (LSTM) and gated recurrent unit (GRU) [86]. These cells basically module how much information from neighbour cells should be “forgotten” or “retained”. These cells utilise different approaches to accomplish the modulation task, whereby differences in parameters are observed, where GRU has fewer parameters than LSTM [86,156]. In some cases, this fact plays in favour of GRU as usually having fewer parameters is associated with lower overfitting [151]. More details about the functioning of each cell are provided in Chapter 3. Thus, GRU cells were chosen as model architecture in this work. An analogous architecture to what is shown in Figure 4-5 was employed for the patent classifier. The number of units of each layer is specified in Figure 4-6.

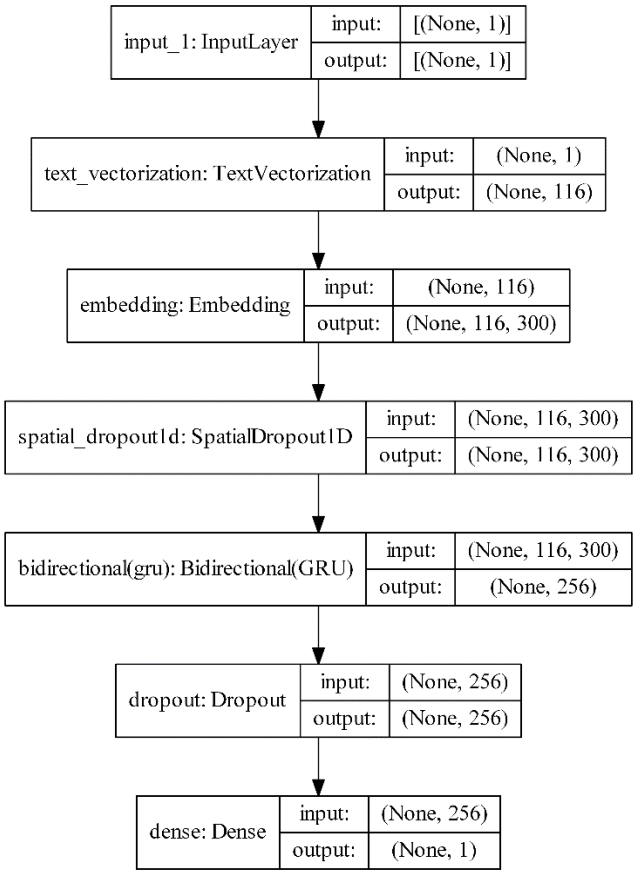


Figure 4-6 Patent Classifier architecture using abstracts. For claims, output_sequence_length is 1001 instead of 116, as in abstracts. Hyperparameters were based on Risch J. et al work [151].

4.2.2.4 Metrics

Precision (P), recall (R) and F1-score were used to assess model performance. In the context of information retrieval, precision would measure how many of the retrieved documents were relevant using a model, in relative terms [158]. On the other hand, recall measures how effective a method is at retrieving relevant documents [158]. F1-score summarises these two metrics by returning the harmonic mean [158]. The formal calculation for these metrics is shown in Equations 2 to 4 [158]. However, as indicated in previous sections, the majority of models do not return directly a class, instead, they yield a probability or a value which influences this decision. As a result, a threshold must be set to categorise a document into a class and depending on this value, the aforementioned metrics may vary.

$$P = P(relevant|retrieved) = \frac{\#\ relevant\ documents\ retrieved}{\#\ retrieved\ documents} \quad (2)$$

$$R = P(retrived|relevant) = \frac{\#\ relevant\ documents\ retrieved}{\#\ relevant\ documents} \quad (3)$$

$$F_1 = \frac{2PR}{P+R} \quad (4)$$

In this manner, these metrics are not appropriate to summarise the overall performance of a model, but they are more useful to aid an optimal threshold selection. Thus, a complementary metric was used which was area under the precision-recall curve (AUC-PR). AUC-PR has been recommended as the first choice to evaluate performance in binary classification for imbalanced data [159]. This metric seems to be more sensitive to changes in the minority class, allowing to appreciate more easily the effect of class size differences [159]. Considering that a preliminary analysis suggested that the obtained data were imbalanced, AUC-PR was also evaluated. In the end, the model with the highest AUC-PR was chosen as the best. Then, P, R and F1-score were calculated over a range of values to select the parameter that yields the largest F1-score.

4.2.3 Content analysis

Once the best model for patent classification was selected, this was used to retrieve the pharmaceutically relevant patents from those whose CPC was not available. This led to have a final corpus which contains patents with pharmaceutical information that can be extracted. To evaluate the type of information, which is expected to be found in the corpus, topic modelling was carried out. *tf-idf* matrix was calculated for preprocessed abstracts. At this stage, terms that contained digits, with less than three characters or included in Table 4-5 were excluded. Similarly, terms that appeared in less than 50 documents were discarded. In addition, bigrams were included, and sub-linear transformation was applied to the values.

Table 4-5 Excluded terms in content analysis.

Term		
Relate	provide	include
present	useful	describe
invention	include	e.g.
comprise	[NUM]	example
disclose	have	select

Later, minibatch nonnegative matrix factorization (NMF) was used to group patents in topics and extract keywords. NMF was trained with the following hyperparameters: n_components=18, batch_size=16000, random_state=32, beta_loss="kullback-leibler", l1_ratio=0.5, and max_no_improvement = 10. Eventually, top 20 keywords were extracted for each topic and revised to define the main relationship. Then, topics of each document were identified, and distribution was analysed.

4.3 Results and discussion

4.3.1 Search results

3'238,764 patents were obtained in the search results. The patent distribution based on search terms can be seen in Table 4-6. The majority of the results were obtained by using dosage form terms compared to active ingredients synonyms. Keywords for the results obtained from abstracts can be seen in Figure 4-7. Based on that, the obtained patents seem not to be pharmaceutically relevant. Thus, CPC classification was revised in more detail to identify patents that contain information on pharma.

Table 4-6 Distribution of the search results by sources and search terms.

Database	Terms	Results
Patentsviews	Active pharmaceutical ingredients drugbank	544267 (16.80%)
	Dosage forms	1048039 (32.36%)
USPTO Bulk Search API	Active pharmaceutical ingredients drugbank	244553 (7.55%)
	Dosage forms	1401905 (43.29%)

18.2% of the results did not report CPC. Of the remaining patents, only 7.5% corresponded to pharmaceutical patents A61K, A61P, and A61Q; this group will be denominated as PH. As for the majority group (NP), 61% of patents corresponded to sections B, G, and H. These categories enclosed inventions related to devices or pieces of equipment related to, but not limited to, the generation of electricity, measurement of

physical properties or execution of unit operations [145], which is consistent with the keywords shown in Figure 4-7A and B. On the other hand, keywords observed in Figure 4-7C showed patents in group PH contained information about therapeutical substances and formulations.

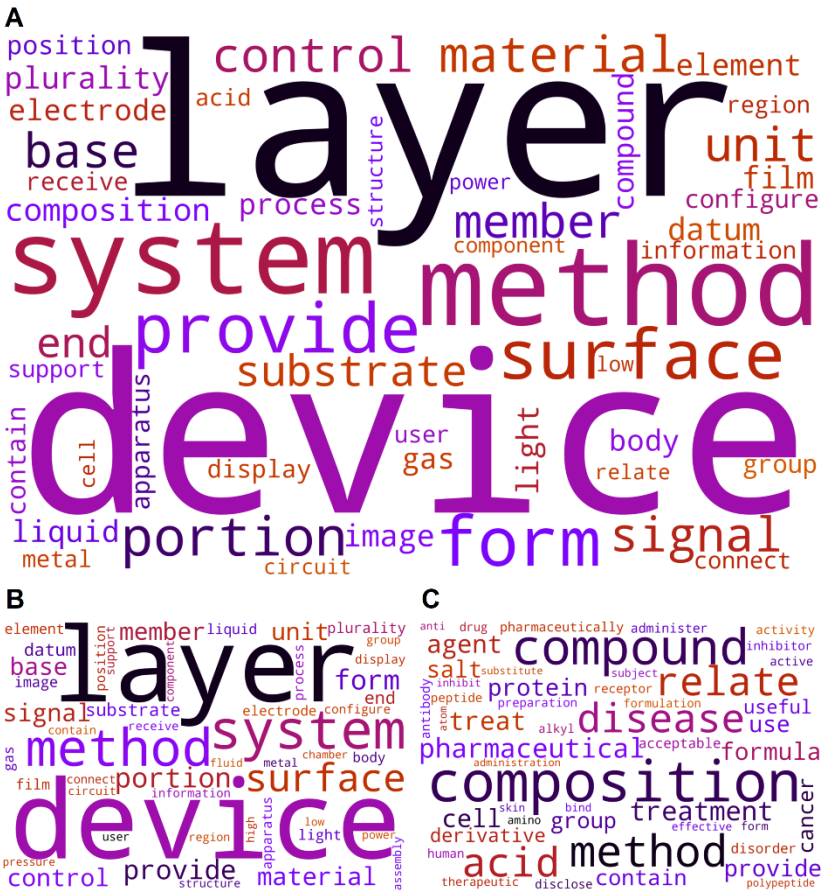


Figure 4-7 Word clouds with top 50 most important words for all the search results (A), irrelevant results (B), and pharmaceutical patents (C).

The main source of irrelevant information was terms related to dosage forms. This can be seen in Table 4-7, where most non-pharmaceutical patents were obtained by using dosage forms as search terms. This result was expected as several words employed to describe dosage forms have several meanings in different fields and were not exclusive to the pharmaceutical context, as in active ingredients case. In this case, it was necessary to apply an additional filter to the dosage form to obtain information on secondary manufacturing whereby CPC becomes useful to bound the results to pharmaceutical

data. However, a limitation of using this classification was that all the results did not count in this class. As stated previously, 18.2% of results do not have CPC class, which is equivalent to 590.761 patents whose relevance was uncertain. Thus, a method to classify patents into either group PH or NP was developed and applied.

Table 4-7 Distribution of the search results by relevant (PH) and irrelevant (NP) patents with CPC.

Database	Terms	Class	Number of patents
Patents View	Active ingredients	NP	406103 (12.5%)
		PH	137027 (4.23%)
	Dosage forms	NP	1019339 (31.5%)
		PH	27147 (0.83%)
USPTO API	Active ingredients	NP	74739 (2.31%)
		PH	25120 (0.78%)
	Dosage forms	NP	950098 (29.3%)
		PH	8430 (0.26%)
Total			3238764

4.3.2 Patent Classifier

Abstract and claims were retrieved for all the patents with an assigned CPC. Approximately, 97.9% of the information could be extracted from files. Most patents whose abstracts or claims were not available corresponded to years from 2002 to 2004. This period coincided with the first years that XML format started being used by USPTO, resulting in documents with varying syntaxis that hindered information extraction. Due to this limitation, a small portion of the data from all the patent files could not be obtained. In the end, with retrieved information, models for patent classification were trained and the results are discussed below.

The performance of the trained models can be seen in Figure 4-8A. Overall, the best result was achieved by RNN in both cases, using abstract and claims as inputs. This model yielded an average AUC-PR of 0.926. This result was followed by LR and linear SVM, with average AUC-PR of 0.871 and 0.869, respectively. The outcomes obtained presented a certain agreement with the literature for similar tasks [84,85,151]. RNN are usually the first choice for text classification. The reason behind that is RNN have been reported to outperform other algorithms like the other models employed in this work. Unlike LR and SVC, RNN not only considers individual words but also captures the connection between words. In this way, RNN can identify more complex patterns.

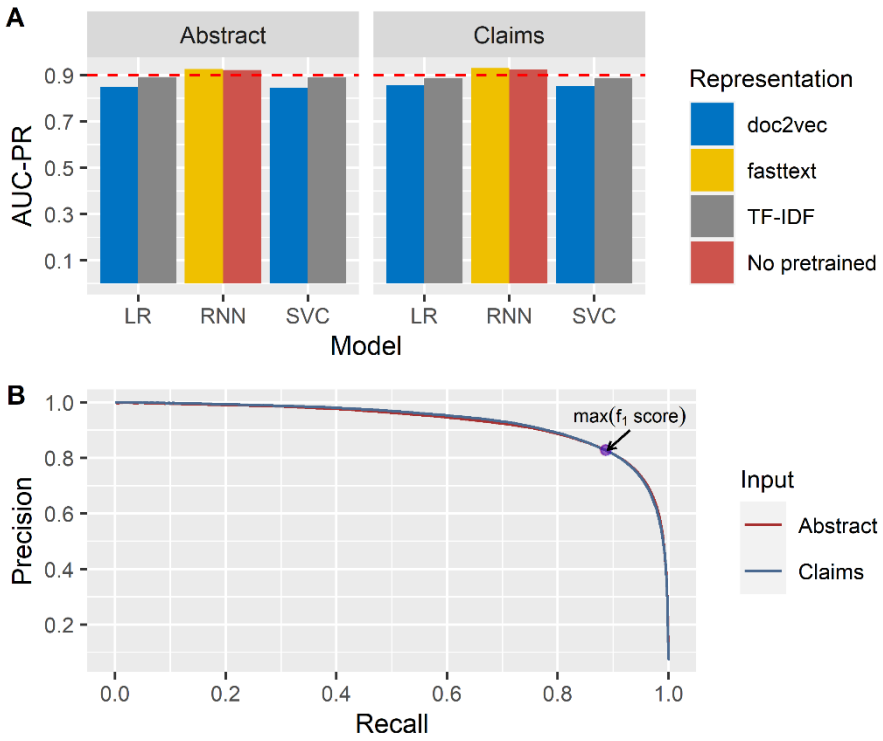


Figure 4-8 Classification performance for different models. A) comparison of AUC-PR for several model trained for different inputs and text representations. B) Precision-recall curve for the best model – RNN + Fasttext using abstract and claims as inputs

Upon comparing word representations, doc2vec showed the lowest performance in all the models, being lower than TF-IDF method. This suggests that doc2vec trained under the employed hyperparameters did not capture the differences between both groups and is not the best representation for the classification of this kind of document. On the other, fasttext had a higher performance compared to custom word embeddings trained along with RNN with AUC-PR 0.929 and 0.923, respectively. This difference could be caused by overfitting, although this different might not be large enough to conclude so further testing is required to confirm.

The evolution of model performance across training is shown in Figure 4-9. For the model with custom embeddings, AUC-PR values for the training set increased rapidly, whereas the validation set decreased after epoch 2. In addition, the difference between losses of training and validation set tended to increase rapidly. In opposition to this behaviour, although training took longer when fasttext embeddings were used, the difference in performance and loss between the test and training set was stable. The first behaviour indicates that the model is “memorizing” how to classify the training sample, instead of

learning any latent information that yields good results in unseen samples, which may indicate the model overfit [160]. As a result, the generalization of this model is poorer than using fasttext embeddings. In this manner, pre-trained embeddings provide more reliable results as it is less prone to overfitting.

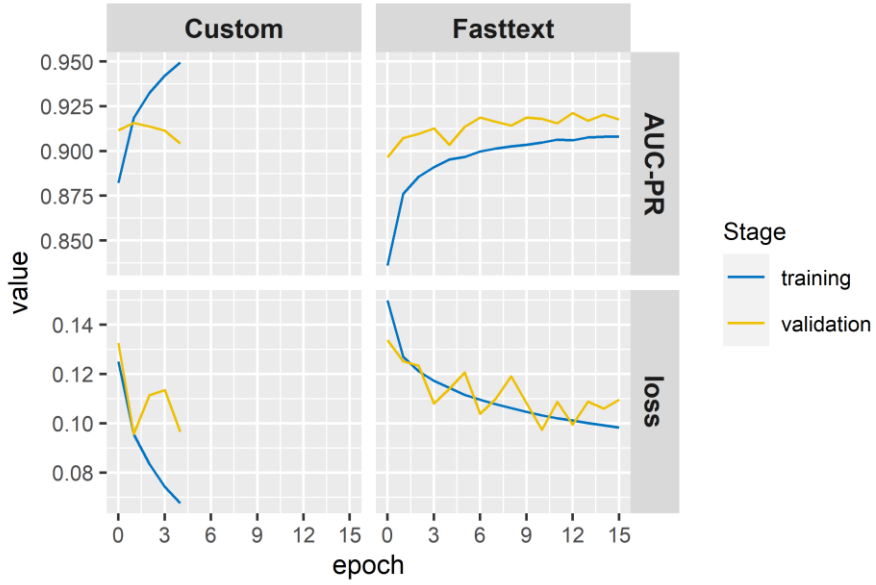


Figure 4-9 Comparison of Abstract + RNN performance using custom and pretrained embeddings (fasttext) as a word representation during training time.

Regarding the effect of using abstracts or claims as an input for classification, the results are compared for RNN-Fasttext models in Figure 4-8B. The AUC-PR of both models are similar with values 0.927 and 0.930 for abstract and claims, respectively. Although, the usage of claims to classify patents provided a slightly higher performance, when other metric like F1-score was compared the results were the opposite. F1-score for RNN with abstract was 85.7%, while a value of 85.6% was obtained for claims. By seeing closely, the most visible difference between abstract and claims is the number of words. The distributions of number of words in abstract/claims are shown in Figure 4-10. While 90% of abstracts are described with 98 words or fewer, this figure increases to 1001 words for claims. From the results, it is demonstrated that even though claims provide additional information, this is not helpful to define whether or not a patent is related to pharmaceutical manufacturing. Therefore, claims contain information that is redundant and, to a certain extent, is already compressed in abstract for this particular task.

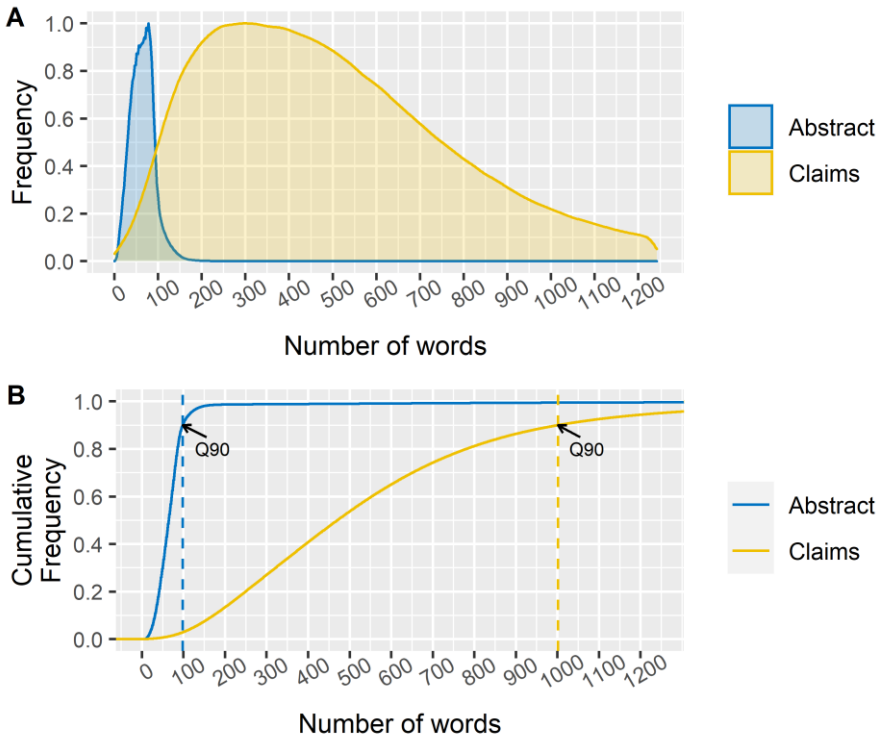


Figure 4-10 Histogram (A) and cumulative frequency (B) of the number of words for abstracts and claims.

To conclude the best model selection, although the utilization of claims for classification might not harm performance, this may lead to a greater computational cost. This could be observed at the training stage, where the duration of an epoch using abstract and claims was approximately 13 and 92 minutes, respectively. In this manner, since claims did not improve performance significantly and could cause prediction or training to take a longer time, Abstract + RNN + Fasttext was chosen as the best model for patent classification.

4.3.3 Error analysis

The confusion matrix in Figure 4-11 describes the performance of the final model in more detail. Performance in terms of precision and recall was 82.9% and 88.7%, respectively. As can be seen, the data were imbalanced, where the group of interest (PH) represented the minority group. 1.4% of the predictions corresponded to false positives (FP). This meant that 17.1% of the retrieved patents were misclassified into PH class. In the scenario of false negatives (FN), the model did not recognise 11.3% of the relevant patents as such. Consequently, approximately 1 in 9 pharmaceutical patents are

expected to not be identified as relevant, thereby not being considered for further analyses.

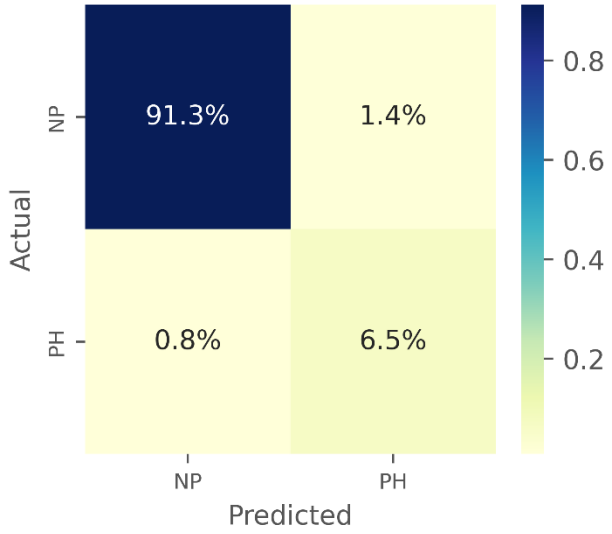


Figure 4-11 Confusion matrix for the best model. PH: pharmaceutically relevant patents and NP: non-relevant patents.

Having considered the two types of errors that the model presents, FP are of particular significance in this case. On the one hand, while FN cause a loss of data to be analysed, these may not impact directly on conclusions as these patents are simply omitted and the proportion is minor compared to FP. On the other hand, FP would be analysed with true positives, which may mislead conclusions depending on their content. In consequence, content profiling for misclassified patents was performed to assess what data FP and FN are expected to provide.

The most common CPC classes derived from the profiling of FP and FN are displayed in Figure 4-12A. 73.4% of FP patents belong to main class C, which groups inventions related to chemical entities and metallurgy. In particular, classes C07D, C07K, and C12N stood out as they are the most frequent and comprise heterocyclic, peptides, and acyclic compounds. Similarly, C07 inventions, in general, cover organic compounds, while enzymes and microorganisms' compositions are grouped into C12N class. Thus, it is possible to see the majority of FP led to retrieve patents about chemical substances that can be regularly found in pharmaceutical context, and these may provide data about synthesis and/or purification process.

As for the remaining classes, A61L, G01N and Y10S corresponded to 24.3%. The latter is a generic category which involves new technological patents. Y10S belongs to a recently implemented section (Y) that covers a wide range of inventions. Usually, the patents that fall into this category have another assigned class, whereby Y10S class may not specify clearly the invention field. Therefore, a more accurate definition of the technical field is obtained from additional CPC. Regarding G01N and A61L, these comprise patents about methods to test or analyse materials and methods or apparatus for sterilising materials. Even if these could still be of interest in other fields in pharma, these types of inventions may not offer specific information on manufacturing. Consequently, G01N and A61L categories might be of concern for information extraction of manufacturing data as these documents may introduce noise to the collected corpus. However, their impact is not expected to be significant since these represent a minority within all FP and can be discarded by analysing in more detail other sections such as detailed description which will be the focus in further chapters.

Another aspect to highlight was that multiple classes in common could be observed among the top 10 most frequent categories of FN and FP. For instance, C07D, C12N and C07K were also found in FN. Nonetheless, even though these classes co-occurred with relevant classes - A61K, A61P, and A61Q, patents were not recognized as relevant by the algorithm.

Upon observing keywords for FN and FP in Figure 4-12B and C, terms related to application area were the main difference. While FN keywords provided an idea of patents were about compounds and chemical substances, these did not suggest what the specific application of the compounds might be. In contrast, FP keywords mentioned terms related to diseases or directly words such “pharmaceutical” or “treatment”. Consequently, this points out that the algorithm requires the abstract to contain words emphasizing that the invention has a therapeutical application, apart from indicating that it is about a chemical substance or drug product. This fact shows that the algorithm is consistent about the identification of patents of pharmaceutical interest as one of the most important aspects to achieve this is the field of invention. In addition, this suggests that the abstract might not provide all the necessary data to have a higher performance.

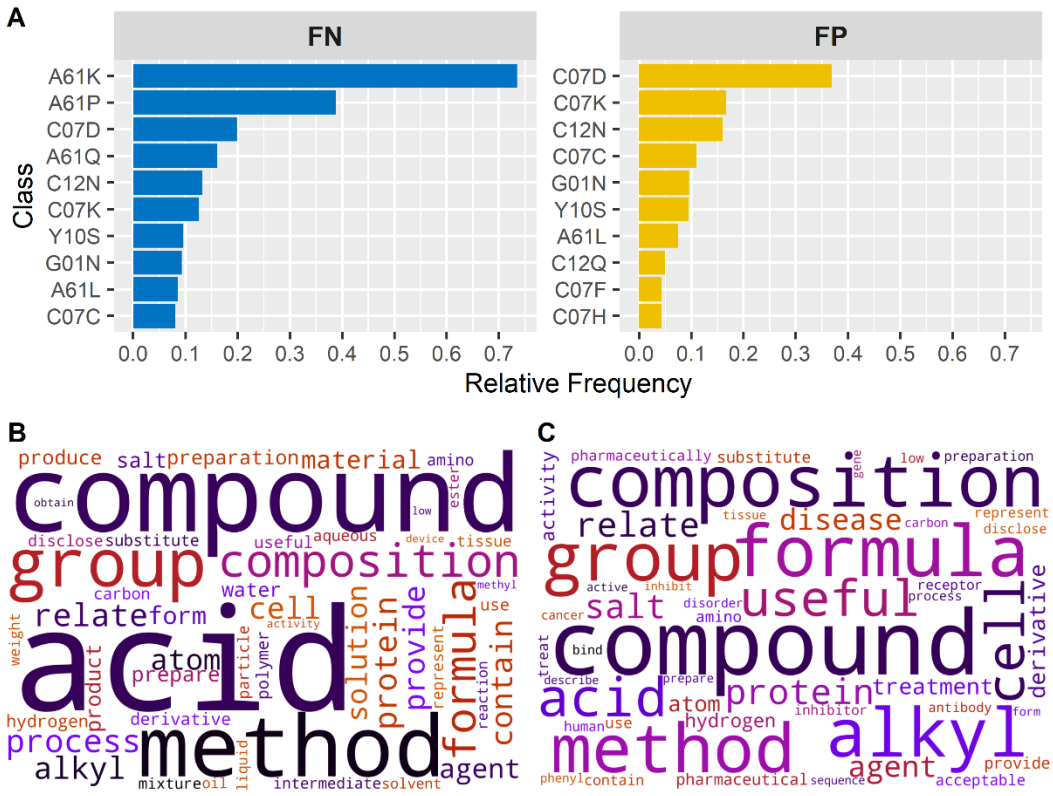


Figure 4-12 Distribution of CPC classes (A) and keywords for FN (B) and FP (C).

To conclude this section, most errors found could be considered reasonable and are not expected to yield a substantial number of off-topic patents. FN could be caused by limitations of the information provided by the abstract since this seems to not offer insight into the invention field on all occasions. As shown in the previous section, claims did not help to improve performance significantly, by which using detailed descriptions might help to retrieve more data but with a much higher computational cost. On the other hand, an important portion of FP turned out to be relevant. This idea is supported by the fact that most frequent classes in FP were related to new organic chemical entities, usually found in pharma, and keywords suggest these molecules may be used in the treatment of some diseases. However, there still is a minority group that is unlikely to contain manufacturing information such as patents of classes G01N and A61L. Although it was shown these represent a small proportion of all classified patents and might not have a big impact on conclusions, it would be convenient to consider further filters to discard these patents during the information extraction of manufacturing data.

4.3.4 Pharmaceutical manufacturing corpus

Initially, 197,724 patents were identified as pharmaceutically relevant and there were 590,761 patents whose content was unknown. After applying the patent classifier in unknown patents, 18,299 documents (3.1%) were found to potentially have content related to pharmaceutical products. As expected, a significant portion of unknown patents were unrelated to the field of interest in this study. This result is consistent with the proportion of documents extracted in the search results, which was around 6.1%. In the end, after discarding additional patents due to not including abstract, a total of 208.664 documents were gathered

The collection of documents accounted for information from January 1976 to September 2021. Using non-negative matrix factorization (NMF) over abstracts, patents were categorized into 18 subtopics and their content was explored in more detail. As a note, NMF is a soft-clustering technique, by which it does not assign a document to a unique topic. Instead, it allocates weights to every topic in such a way that those weights measure the relatedness of the documents to each topic. In addition, NMF provides the most important terms to define every topic. The top 5 keywords for each group can be observed in Figures Figure 4-13 and Figure 4-14. As can be seen, keywords confirmed documents fitted well with subjects related to pharmaceutical manufacturing in most cases and also showed aspects usually highlighted in the inventions.

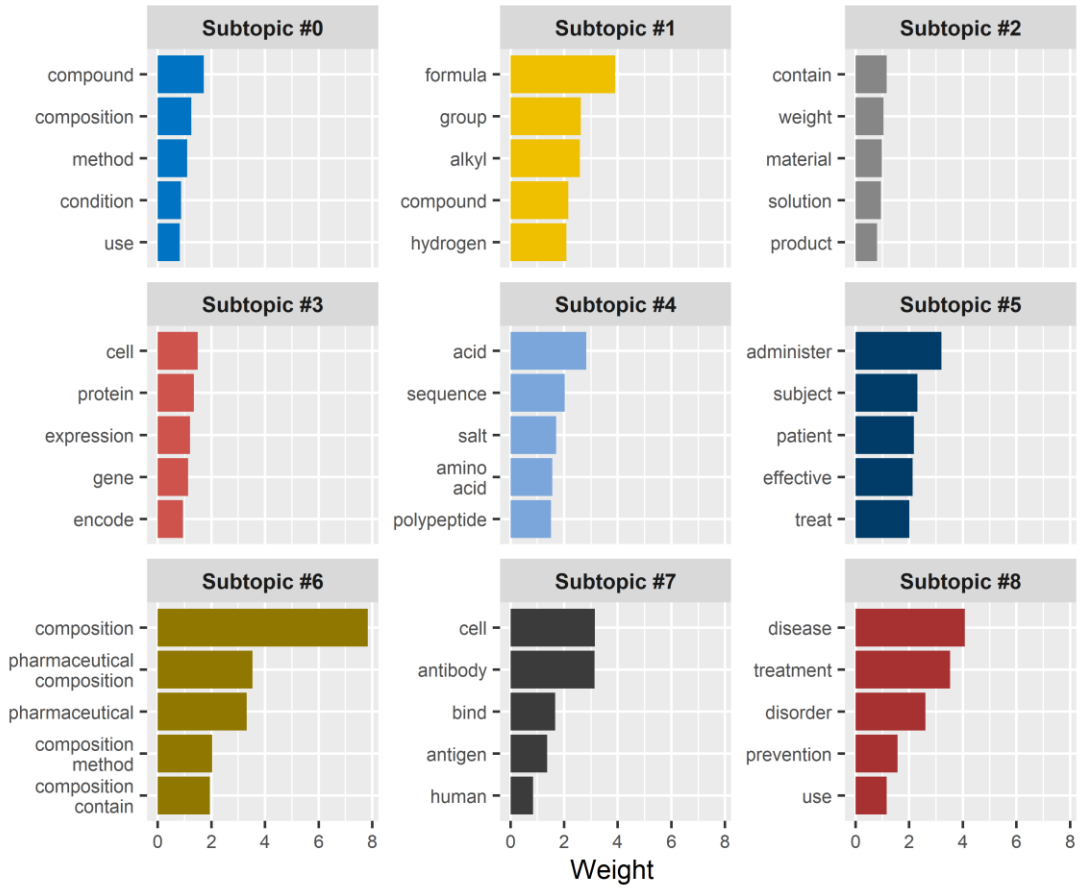


Figure 4-13 Top 5 keywords for subtopics 0 to 8 obtained using NMF.

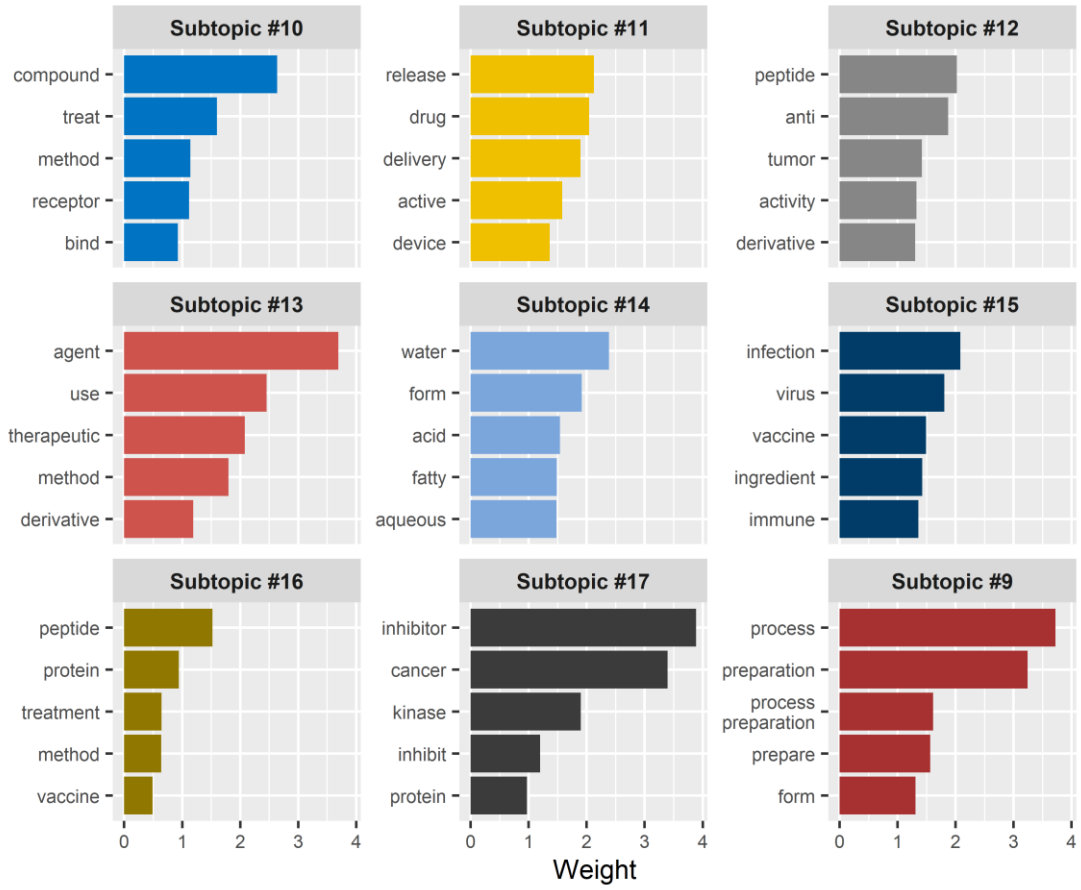


Figure 4-14 Top 5 keywords for subtopics 9 to 17 obtained using NMF.

Table 4-8 Assignation of subtopics to main topics.

Topic	Subtopics
Pharmaceutical Composition	0, 2, 6, 11, 14
Small Molecules	1
Biological Products	3, 4, 7, 16
Therapeutical Application	5, 8, 10, 12, 13, 15, 17
Manufacturing	9

Subsequently, through subtopics keywords, 5 major topics were identified: Biological Products, Therapeutical Applications, Small Molecules, Pharmaceutical Compositions, and Manufacturing. The assignation of subtopics to major topics is shown in Table 4-8. To remark, 68.1% of patent abstracts mainly focused on the health benefits or pharmaceutical products and the majority of patents correspond to the last decade as observed in Figure 4-15A. This latter covers inventions involving novel delivery systems

for conventional active pharmaceutical ingredients, natural products, and cosmetics, whereby all of these may not be necessarily associated with the treatment of a disease. As for the therapeutical applications, these may comprise well-known substances repurposing or new chemical entities usages. Particularly in this topic, when the top 20 keywords were observed, some of the most common terms were related to pathologies, namely “cancer”, “pain”, “infections”, or “skin”. This may denote what groups of diseases and pharmacological groups were having more attention during the assessed period.

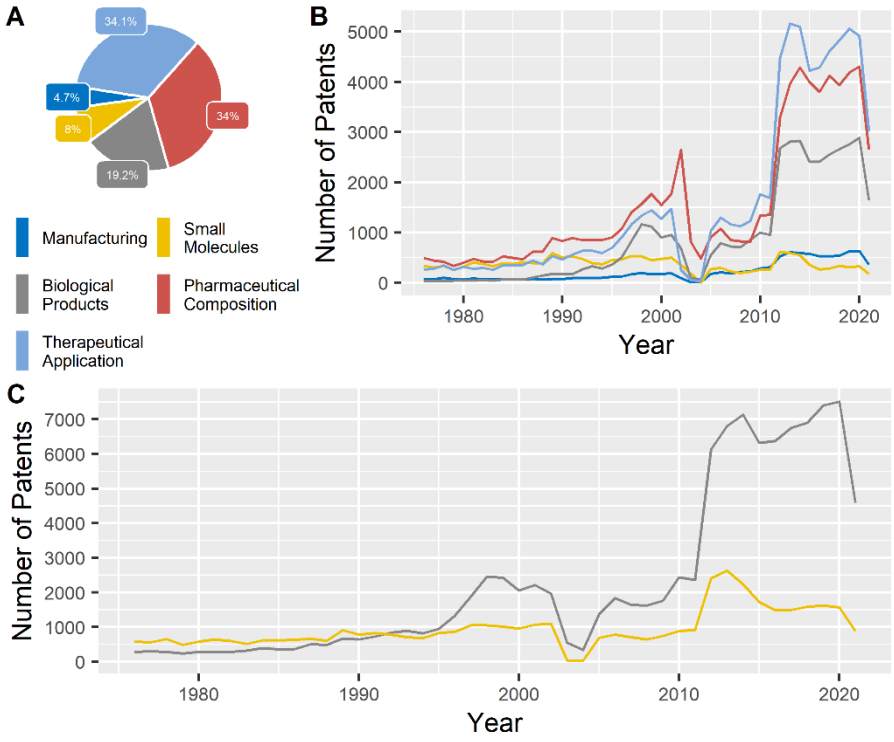


Figure 4-15 Overall distribution of main topic in the corpus (A), evolution of topics across time (B) and evolution of the patents most related to small molecules or biological products (C). To note, data for 2021 only covers until September, which is why a sudden drop in the number of patents is seen this year.

Regarding new substances, biologics and new chemical entities (NCE) represented 27.2% of the total. However, it is worth noting that when the top 5 topics with the greatest weights were considered, new substances came to account for 70.6% of all the documents. This suggests that even though it was not the main point of the patent, all of these were related to some substance. This result was expected considering a previous study performed by Ouellette L. et al in 2010. This shows usually a substance is protected by several patents, where each covers different aspects of the invention[161]. During that study, an important outcome was that by 2005, a NCE could have between 3.5 and 5

patents related on average [161]. In addition, the authors showed that this number tended to increase [161]. This would explain the trend shown in Figure 4-15B, where most topics have undergone a significant increase in the last decades, except for small molecules. In addition, all the topics different to biological products and small molecules were more frequent; thereby pointing out that currently more patents are being used to protect a new substance. However, this would not explain the large difference observed between patents related to small molecules and biological products.

Going into more details about patents evolution across time, the number of patents related to either biological products or small molecules taking the top 5 most important topics is shown in Figure 4-15C. It was observed that until 1992, small molecules were the focus of the inventions. However, after that time, terms related to biological products such as “proteins” and “antibodies” started to have a greater weight. From this time onwards, the difference between these topics has kept increasing, to a greater or lesser extent, until the most recent date. In this regard, this trend had been pointed out in 2012 by Jones et al [162]. In their findings, the authors indicated that at least before the 2000s, biologics had surpassed small molecules and, between 2005 and 2008, the gap between these had been widen[162]. Some of the factors that favoured biologics evolution were lack of strong regulation in comparison to small molecules, finding new small molecules has become harder as this field has been exploited for a longer time, and technological development has eased biologics manufacturing [162,163].

Interestingly, biologics dominance has not been translated into a greater market share. Figure 4-16 reveals that, even though biologics approvals were raised, small molecules kept being higher[164]. Similarly, it has been estimated that around 90% of the worldwide market share corresponds to small molecules[163]. This has been explained by the fact that the transition to a biologics-based market is a long-term process since the majority of infrastructure has been developed to manufacture small molecules[162]. Similarly, the high costs of biologics hinder the accessibility to all the population[163]. Thus, the disparity between marketed and patented products shows that many inventions are not commercialised. This may be caused by market projections, technical feasibility or other decisions that a company could make[162]. In terms of information retrieval, these facts suggest that patents are a good source of novel information as it is expected to find many more compounds compared to what is available in the market. However, the trade-off is

that the information found in patents might not always be as complete as for marketed products.

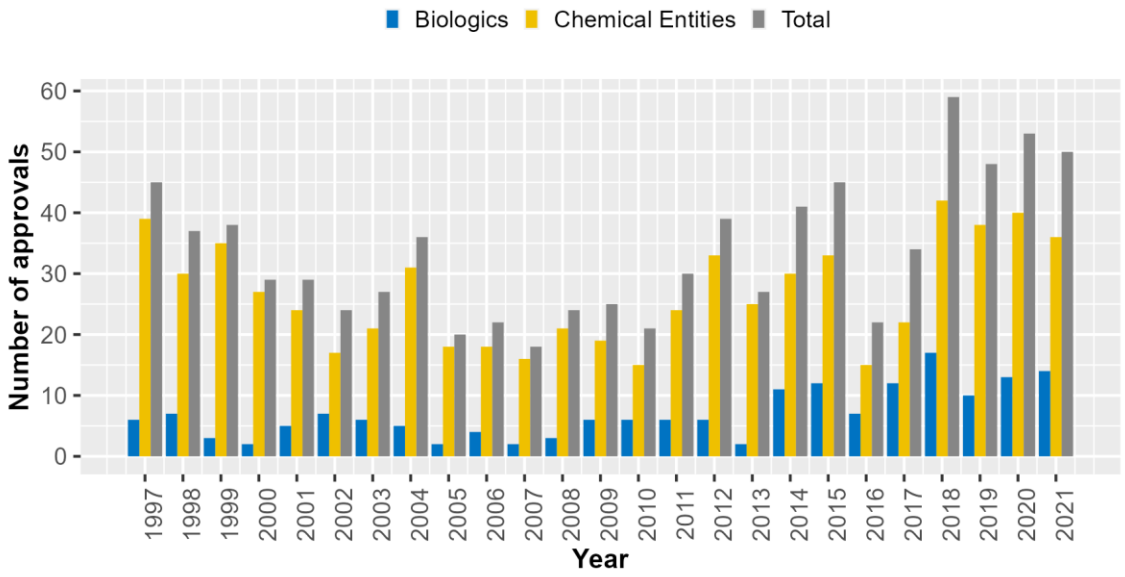


Figure 4-16 Number of NCE and biologics approved by FDA between 1997 and 2021. Taken and adapted from de la Torre et al [164]

To conclude, a corpus about pharmaceutical manufacturing containing 208,664 patents was built. This corpus covers a period of 46 years. Through exploratory analysis, the content of the documents was validated. This information can be summarised, but not limited, in 5 main topics: Biological Products, Therapeutical Applications, Small Molecules, Pharmaceutical Compositions, and Manufacturing. Additionally, some trends could be observed where most patents corresponded to recent years and related to pharmaceutical compositions and therapeutical applications. The former trend might generate some bias towards recent files. Regarding substances class, most documents had content associated with biological products followed by small molecules. Finally, from this corpus, pharmaceutical manufacturing information is expected to be extracted. For instance, information on secondary manufacturing is likely to be found in patents belonging to Pharmaceutical Compositions; also, primary manufacturing data such as synthesis or purification method is to be in topics related to small molecules or biologics.

4.4 Summary

Although several corpora related to biomedical topics can be found, that is not the case for pharmaceutical manufacturing. Using patents as a main source of textual data, a corpus to be used for mining manufacturing information was built. Throughout the process, more than 3 million of patents were collected but these were filtered using initially CPC classification, as the majority were irrelevant. Nonetheless, a significant number of these did not have CPC. Therefore, collected documents were used to generate one of the outcomes of this chapter which is the patent classifier. The patent classifier consists of a deep neural network which can identify pharmaceutically relevant patents through abstract text. The performance of this model measured as recall and precision was 82.9% and 88.7%, respectively. Furthermore, the error analysis reveals that classification error made by the model did not have a serious impact on the retrieved documents.

Using the CPC and model, more than 200K pharmaceutical patents were gathered. The content of the corpus was then validated by using a topic modelling technique known as non-negative matrix factorization (NMF). This method grouped the patents into topics and output keywords for every group. With model outputs, it was possible to identify first inventions main topics based on abstracts and later describe the evolution of patents content across time. Thus, most patents information can be clustered into the following topics: Biological Products, Therapeutical Applications, Small Molecules, Pharmaceutical Compositions, and Manufacturing. The topics about Therapeutical Applications and Pharmaceutical Composition turned out to be the most frequents. As for substances, Biologics patens are more dominant than small molecules. This trend started to be significant from 1992. The patents are expected to contain data of primary and secondary manufacturing according to these topics and keywords.

Chapter 5. Natural Language Processing (NLP) for Pharmaceutical Manufacturing Data Extraction

5.1 Introduction

Information extraction (IE) aims to identify and summarise information of interest from documents for a specific domain [73]. This task is a multistep process that usually requires the deployment of several components. A first step usually involves preprocessing, which prepares text to be used downstream [147,165]. Preprocessing steps can include methods to filter out irrelevant data to improve the reliability of the information [147]. Then, components such as a part-of-speech (POS) tagger can be applied to individual words to assign semantic functions (verb, noun, or adjective). In a similar way, named entity recognition (NER) systems categorise words into a class, which hints at the type of data provided by each token [90,147]. Contrary to POS tagger, the NER component is more domain-specific. Thus, the development of an IE system involves the design and elaboration of a pipeline making use of different approaches.

In this regard, several workflows have been developed to extract information from documents in the chemical domain. O. Kononova *et al* proposed a methodology to mine inorganic synthesis procedures from papers [166]. In this work, the authors developed methods that first identified the paragraphs that contained information on certain types of reactions. Then, these paragraphs in turn were input into other components to recognize materials, operations and conditions [166]. Continuing with primary manufacturing, Lowe's work developed a framework to extract chemical reactions and structures from patents [167], where NER models are highlighted to play a key role. Similarly to the aforementioned examples, many others found as a common factor the emphasis on the development of NER models [91,92,168–171].

NER systems can be considered the core of IE task [165]. This component identifies and enables the extraction of key information (entities). Depending on the methodology applied, NER can be based on rules, or machine learning (ML) approaches [90]. Rule-based NER uses dictionaries or semantic/syntactic patterns to label entities [90]. It has been shown that this approach offers very good precision but low recalls, due to the requirement of exhaustive dictionaries and the need for a wide variety of rules to cover all the possible variations in word patterns [90]. Due to these limitations, ML methods have become more popular since they provide greater flexibility, achieving good recall and precision for several applications, particularly when deep learning is used [90]. In the

pharmaceutical context, the development of NER models has been focused on the biomedical and primary processing domain [91,92,94,95,167–170,172]. In the latter case, models for the recognition of materials, operations or both can be found. However, to the best of our knowledge, models for secondary processing have not been found.

Returning to IE workflows, pharmaceutical patents have special considerations to extract manufacturing data. In a previous chapter, a corpus was built with pharmaceutical manufacturing patents. However, contrary to what happens in scientific articles, these patents are written in such a way that their procedures are difficult to reproduce and do not have well-defined sections that indicate whether or not the specific content of interest is available [142]. In addition, the whole document does not necessarily cover a specific area, but instead, involves several aspects of an invention that may include clinical information, analytical data, manufacturing procedures, etc. Therefore, the first step for data extraction should revolve around the selection of relevant sections or text fragments. Subsequently, these sections can be inputted into a NER model.

However, as mentioned previously, current available NER models are mostly applied to primary processing data extraction for pharmaceutical manufacturing applications. Considering that secondary processing is out of the scope of these models, a new model for drug product fabrication data mining would be necessary to collect data for both modes of manufacturing. An additional aspect to consider would be efficiency given the volume of data. A unique model capable of performing the recognition of entities for both primary and secondary domains would allow extracting simultaneously all the targeted information. In addition, it would facilitate the distinguishing between synthesis/purification procedures and drug product manufacturing. In this manner, this chapter aims to develop models that assist in data extraction of manufacturing data of small molecules from patents using natural language processing (NLP) tools from patents. These tools consist of a relevant section selector and a NER model for both primary and secondary processing. These models will then be used to build a database through the corpus developed in the previous chapter. Both proposed model will work in conjunction for IE as illustrated in Figure 5-1.

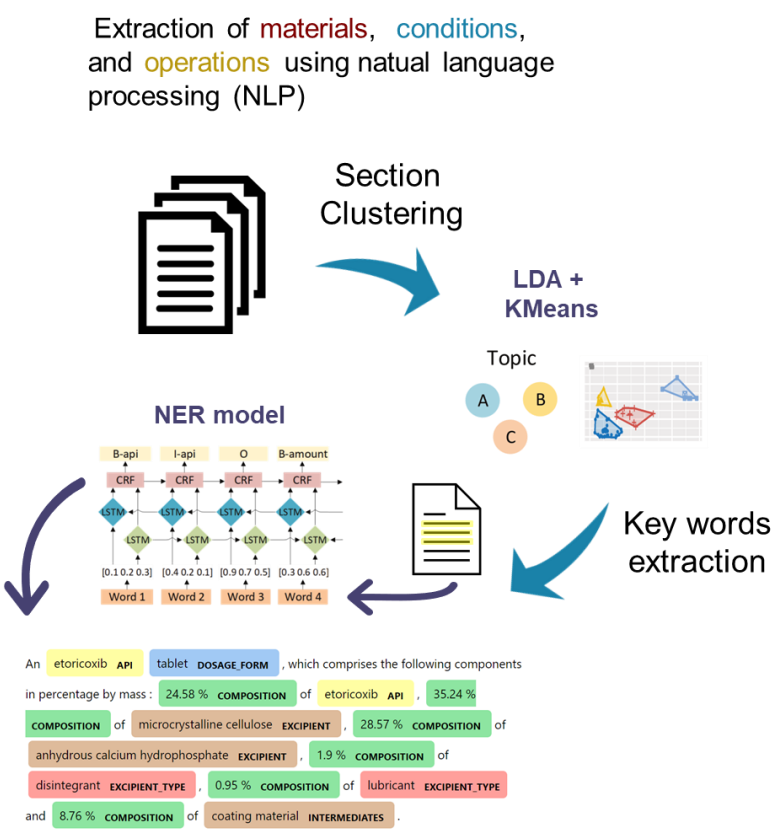


Figure 5-1. Overview information extraction (IE) using natural language processing (NLP) tools.

5.2 Methods

5.2.1 Preliminary experiments

Preliminary tests were carried out as a first approach to develop models and improve the understanding of certain NLP tools. Initially, a sample of around 49K patents was selected to determine relevant paragraphs. These patents contained around 3.6M of elements summing paragraphs, headings and tables. At this point, all the elements were preprocessed as indicated in section 5.2.2.1, excluding truncation, and then Latent Semantic Analysis (LSA) coupled with k-Means were used to extract the topics.

LSA is a topic modelling and dimensionality reduction technique which takes as an input a matrix representing documents [173]. This method decomposes document representation, such as bag-of-words (BoW) or term frequency-inverse term frequency (TF-IDF) ($\boldsymbol{C_k} \in \mathbb{R}^{D \times V}$) with D number of documents and a vocabulary size V, into three matrices applying truncated singular value decomposition (SVD) as shown in Equation 1

[173]. The three matrices $\boldsymbol{U} \in \mathbb{R}^{D\times K}$, $\boldsymbol{\Sigma_k} \in \mathbb{R}^{K\times K}$, and $\boldsymbol{V}^{\top} \in \mathbb{R}^{K\times V}$ correspond to a lower dimensional representation of the documents in a reduced K-dimensional space, a diagonal matrix containing the singular values with K-latent dimensions , and a matrix that associates new dimensions and terms, respectively [173]. This technique in particular requires a pre-set value for the rank K of $\boldsymbol{\Sigma_k}$, which is usually in the order of hundred[173]. For the present words values of K 100, 200, 300, and 500 were evaluated.

$$\boldsymbol{C_k} = \boldsymbol{U\Sigma_k V^T} \quad (1)$$

The matrix U obtained from LSA was then used as input in k-Means analysis. The implementation of k-Means employed for this experiment is described in detail in section 5.2.2.3. At this point, for the selection of the optimal number of clusters only Davies-Bouldin index was employed. Various numbers of clusters were tested ranging from 40 to 390. Afterwards, with the optimal number of clusters, the top 20 keywords per cluster were extracted from centroids. Finally, keywords were revised to firstly assign an arbitrary label which summarises the content and, secondly, define the cluster that may contain information related to manufacturing of small molecules. The clusters that contained mostly keywords related to dosage forms, operations, conditions, or materials were considered relevant. Although this approach turned out to be useful to segment relevant vs irrelevant content, some weaknesses constrain this application, which are discussed in detail in results section. Considering these limitations, this approach was then applied at text sections with additional modifications, instead of paragraphs.

5.2.2 Relevant section detection

5.2.2.1 Preprocessing

From the pharmaceutical corpus, detailed descriptions were retrieved for each patent. Patents without description were discarded. At this stage, descriptions were separated into headings, paragraphs and tables. Thus, texts were preprocessed as illustrated in Figure 5-2. Tokenization was achieved using chemdataextractor 1.3, while the remaining steps were performed with spacy 3.3.[97,133]. Afterwards, the preprocessed text was grouped into sections based on headings. Then, a BoW representation was determined for each section. Additional words were then discarded following the exclusion criteria shown in Table 5-1. In the end, a vocabulary with approximately 62K tokens was built. Finally, sections with a number of tokens greater than $Q_{outlier}$, which was calculated as

indicated in Equation 2 [174], were truncated to this length to ensure these were not longer than the majority of the sections.

$$Q_{outlier} = Q_3 + 1.5\,(Q_3 - Q_1) \quad (2)$$

Where Q_1 and Q_3 is the first and third quantile in text length distribution.

Table 5-1 Exclusion criteria for token in vocabulary for BoW representation.

Criteria	Regular expression
Present in less than 200 hundred sections,	N/A
Present in more than 70% of all the sections,	N/A
Less than 3 characters,	N/A
Containing digits or punctuation,	“\d" and "(?u)\b\w\w+\b"

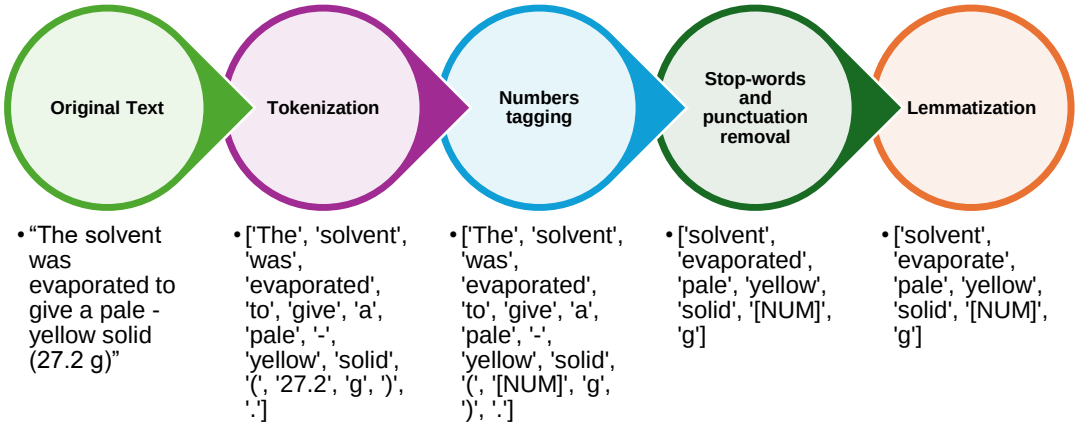


Figure 5-2 Sequence of steps for text preprocessing for topic modelling.

5.2.2.2 Topic modelling

The preprocessed sections were divided into 90% and 10% to train and test LDA models, respectively. Several LDA models were built varying the expected number of topics. In addition, models were run with and without shuffling and truncating the training set. For every model, Perplexity was calculated over the test set to select the optimal hyperparameters. Perplexity is defined as the inverse of the per-word likelihood geometric mean as shown in Equation 3 [80]. A lower perplexity is related to a model with

a better generalization capability [80]. Thus, the model with the lowest perplexity for the test set was chosen as the best. Lastly, once the optimal number of topics was chosen, additional experiments were performed changing hyperparameters alpha and number of iterations. The hyperparameters employed in the development of LDA models are summarised in Table 5-2.

$$Perplexity = exp\left[-\frac{\Sigma_{j=1}^{D} log\, p(\boldsymbol{w}_j)}{\Sigma_{d=j}^{D} N_j}\right] \quad (3)$$

For a sample of D documents, $\log p(w_j)$ equals to per-word likelihood of the document j, and N_j is the number of words of the document j. More specifically, $p(w_j)$ is the probability of generating the set of words $\boldsymbol{w_j}$ in the document j as defined in LDA generation process. Thus, the better the LDA model fits the data, the higher probability it assigns to a set of words associated with a document from a test set. This results in a lower perplexity, as perplexity is inversely related to the log-likelihood of the test data. Lower perplexity values indicate a better fit, meaning the model can more accurately predict the structure of the unseen data.

Table 5-2 Hyperparameters assessed for LDA model development.

Hyperparameter	Tested values
Alpha	Symmetric and asymmetric
Iterations	50 and 200
Number of topics	5, 10, 15, 20, 25, 30, 35, 40, 50, 60, 70, 80, 90, 100
Passes	1
Chunk size	4096

In addition, the top 10 most important words per topic were determined for the best model. These words were first revised to identify meaningful associations within the topics and to validate how coherent the model outputs were. On the other hand, keywords were also employed to assign arbitrary topic labels for the cluster analysis which is discussed in more detail in the next section.

5.2.2.3 Text clustering

With LDA, latent topics and their contributions were determined for all the patents. In this way, the next step was to group documents that shared a similar idea. Thus, to define clusters and allocate documents to each group, k-Means algorithm was employed using as inputs the document representations outputted by LDA. k-Means is an iterative clustering technique. The algorithm works by generating k centroids, where k is a pre-set

number of clusters [81]. Then, the assignment of documents to a particular cluster is defined by the distance between documents and centroids, in such a way that every document is assigned to the cluster whose centroid is the closest [81]. Once all documents have been assigned, centroids are recalculated by averaging document representations[81]. Subsequently, documents are reassigned using the updated centroids. This process is repeated until no significant changes in the centroids are observed or after a predefined number of iterations has been completed [81].

In this work, due to the large number of data, a variation called minibatch k-Means was applied. This version mainly differs from the original k-Means in that, instead of using all the data in every iteration, subsamples known as mini-batches are employed [175]. However, as in k-Means, the optimal number of clusters must be selected. Therefore, various models with k ranging from 5 to 60 were trained. To choose the best model, Davies-Bouldin (DB) and Silhouette (S) scores were calculated as described in Equations 4 and 5 [176,177]. To calculate of these metrics, random samples of 10,000 documents were drawn. The calculations were done with six different samples to estimate the variability of the scores. In the case of the DB score, the optimal number of clusters is reached when the minimum value is found, whereas the opposite applies for the S score.

$$R_{ij} = \frac{s_i + s_j}{d_{ij}} \qquad (4)$$

$$DB = \frac{1}{k}\Sigma_{i=1}^{k} \max_{i \neq j} R_{ij}$$

Where s_i represent the average distance between the centroid of the cluster i and all its members. d_{ij} is the distance between the clusters i and j.

$$s_i = \frac{b_i - a_i}{\max(a_i, b_i)} \qquad (5)$$

$$S = \frac{1}{n}\Sigma_{i=1}^{n} s_i$$

Where a_i is the average distance between the document i and the remaining documents belonging to the same cluster, and b_i is the mean distance between the ith documents and the documents that belongs to the closest cluster.

As for the implementation of Minibatch k-Means, the scikit learn python module was employed [178]. One of the limitations of this implementation is that it only works with a Euclidean distance. In this case, the inputs were the LDA document representations which are probabilities. As such, Euclidean distance has been proven to not be the best

choice for this kind of data [81]. Alternatively, distances such as cosine and Hellinger seem to have shown to measure document similarity in a better way for clustering purposes [81,179]. Consequently, the data were previously transformed for k-Means training. As proxies of the cosine and Hellinger distance, two transformations were assessed: L2-normalization and element-wise square root. The chosen distances are strongly the related to Euclidean distance as illustrated in Equations 6 – 8 [179].

$$D_e^2(\mathbf{x},\mathbf{y}) = \sum_{i=1}^{n}(x_i - y_i)^2 \tag{6}$$

$$D_h^2(\mathbf{p},\mathbf{q}) = \sum_{i=1}^{n}\left(\sqrt{\frac{p_i}{2}} - \sqrt{\frac{q_i}{2}}\right)^2 \tag{7}$$

$$D_c(\mathbf{x},\mathbf{y}) = \frac{D_E^2(\mathbf{x},\mathbf{y})}{2}, if\ \left||\mathbf{x}|\right| = 1\ and\ \left||\mathbf{y}|\right| = 1 \tag{8}$$

Where $D_e, D_c, and\ D_h$ represent Euclidean, cosine and Hellinger distances. $D_e = D_h$ if $x_i = \sqrt{\frac{p_i}{2}}$ and $y_i = \sqrt{\frac{q_i}{2}}$.

With the centroids of the best model, the latent topic with the highest contribution was extracted for each cluster. Then, the keywords obtained from LDA were revised and a label that fit the keyword information was set. Subsequently, all sections were assigned to a cluster. To validate the model, a random sample of 5 sections per cluster was drawn and each document was revised. If the section was clearly related to the arbitrary label, it was marked as 1 or, otherwise, 0. In the case where there was no absolute certainty about the content, a value of 0.5 was given. At the same time, the sections whose content was more related to pharmaceutical manufacturing, for instance, having information on operations, methods, dosage forms or composition, were separately labelled with 1, while if the documents did not meet this condition, were assigned 0. Lastly, the agreement between the results of the algorithm and the manual assessment of relevant documents was estimated using Cohen's kappa and the percentage of agreement. The first metric was used to measure agreement between manufacturing information and the latter to assess concordance between assigned label and actual section content. It is worth mentioning that two percentages of agreement were calculated, one for the worst case and another for the best. In the worst case, the sections with an assigned values of 0.5 were rounded to 0, whereas 1 was used for the other case.

5.2.3 Named entity recognition (NER)

5.2.3.1 Preprocessing

Contrary to section 5.2.2.1, the preprocessing for NER was limited to tokenization. This was performed using chemdataextractor 1.3 [133]. However, it was noticed the package did not deal very well with punctuation in some particular cases. By which, additional regular expressions were added to the preprocessing to mitigate this effect as much as possible. In addition to this step, sentences were segmented using the same package.

5.2.3.2 Training set preparation

Initially, a set of 2,000 paragraphs considered relevant using the methodology described in preliminary experiments were selected. These paragraphs were segmented into sentences and then tokenized. Subsequently, using the IOB scheme, every token was manually labelled using the entity class defined in Table 5-3. The IOB scheme is a label methodology widely used in named entity recognition applications and stands for inside (I), outside (O), and beginning (B) [180]. This is particularly helpful to identify entities that are composed of more than one token. In this approach, irrelevant tokens are labelled as O. As for the remaining, the token that begins the entity is marked with B followed by the respective label. Then, the rest of the entity tokens are marked with I-[Label]. A tagging example can be seen in Figure 5-3.

Table 5-3 Definition of labels. *Category added for the final model.

Label	Definition
AMOUNT	Amount of material employed in a manufacturing process. Normally expressed in terms of mass or volume units
DOSAGE_FORM	Final product of a secondary manufacturing process
SOLVENT	Material used generally to disperse or dissolve another material as a part of a synthesis, extraction or dosage form preparation
REACTANT	Material used in primary manufacturing to synthetise or purify a compound. This category includes: reagents, catalysers, substances to adjust pH, or materials used in extraction such as chromatographic columns. Solvents are not included as there is an independent class for this type of materials
EXCIPIENT	Materials forming part of a dosage form formulation, usually do not have biological activity or are not responsible for the product's therapeutic effect. Solvents

	are not included as there is an independent class for this type of materials
YIELD	Resulting amount or percentage respect to theoretical expected amount of target product obtained from a synthesis/purification process
TARGET	Final product obtained from a synthesis/purification process with a chemical structure clearly defined. This category excludes biological products such as proteins and antibodies.
OPERATION	Action or intermediary step carried out to manufacture a product.
INTERMEDIATES	Materials or mixture of materials which are the result of an operation of a previous process and intervene in a later operation or process. These materials are reported in a generic manner, for instance, some common terms that fall into this category within an appropriate context are mixture, solution, suspension, etc.
CONDITION	Specific settings under which an operation is performed. This includes, but is not limited to, temperature and pressure.
BIOLOGICAL_MATERIAL	Materials originated from either extracts or parts of plants, animals, bacteria or other organisms.
EXCIPIENT_TYPE*	Class assigned to an excipient depending on its chemical properties or function within a formulation.
API	Active pharmaceutical ingredient
PACKAGING	Materials employed as a primary or secondary packaging for the storage of pharmaceutical products.
COMPOSITION	Composition or concentration of the components of a mixture. This can be reported as a proportion, percentage or other concentration units like molar (M).
EQUIPMENT	Instruments, vessels, or specialised machines used in an operation.

During model development step, additional paragraphs were also added. The added paragraphs were characterised by a poor performance to get their entities recognized by the trained models. Thus, 2,069 sentences were included. In the end, the training set consisted of 7,440 sentences. As a final step, sentences with one token and duplicates were removed to obtain 7,215 (221,257 tokens). These sentences were further corrected by comparing actual and predicted labels by models and manually correcting. Example of errors are illustrated in Figure 5-3, where the corrected error only corresponded to case I. The correction procedure was repeated twice, randomly selecting the order of the examples during training stage.

Case I

word	tag	pred
extract	B-INTERMEDIATES	B-INTERMEDIATES
was	O	O
washed	B-CONDITION	B-OPERATION
with	O	O
NaCl	B-EXCIPIENT	B-EXCIPIENT
solution	B-INTERMEDIATES	O
twice	B-CONDITION	B-CONDITION
and	O	O
dried	B-OPERATION	B-OPERATION

Case II

word	tag	pred
acid	I-REACTANT	I-REACTANT
salt	I-REACTANT	I-REACTANT
of	I-REACTANT	O
2	I-REACTANT	B-AMOUNT
(dimethyla	I-REACTANT	B-REACTANT
(	O	O
4.25	B-AMOUNT	B-AMOUNT
g	I-AMOUNT	I-AMOUNT
)	O	O

Figure 5-3 Example errors of labelling. Case I presents example of labelling error. Case II displays error in prediction.

Finally, examples were split into training, development, and test set where sentences were distributed in the following percentages 80, 10 and 10, respectively. The sampling was randomised and stratified by paragraph labels. This sampling strategy was used to ensure the different types of paragraphs employed were represented during training and test stage. To select the best model, the performance of the NER model in the development set was monitored during training. As a performance metric, f1-score micro average was utilised. Finally, the performance over development and test set was reported.

5.2.3.3 Model architecture and initial settings

In the literature, various approaches for NER have been reported; however, methods based on deep learning (DL) have provided the best performance [90]. DL-based NER consists of a system of three components: an input representation, a contextualiser encoder, and a label decoder [90]. The input representation generates a numerical representation for each word or token. As representations, word embeddings such as word2vec, fasttext, or transformers-based models are frequently used. Nonetheless, other features can also be included such as character embeddings or part-of-speed information [90]. Then, the contextualiser processes information from inputs as a sequence, thereby considering the token order, to generate a contextual representation that feeds a decoder [90]. Typically, this component corresponds to bidirectional (Bi) recurrent neural networks with long short-term memory (LSTM) or gated recurrent unit (GRU) cells [90]. Finally, the latter component translates the input into entity types, assigning a label to each token. This task is normally achieved using a conditional random field (CRF) layer [90]. This architecture is illustrated in Figure 5-4.

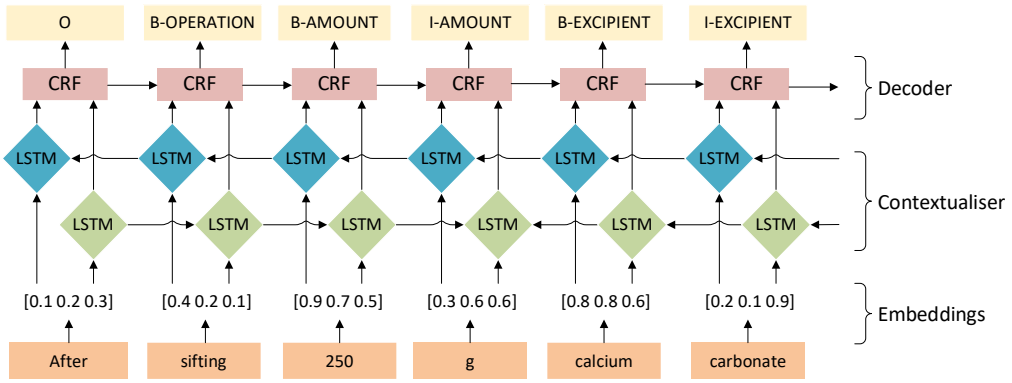


Figure 5-4 Common architecture for NER using DL.

As a base architecture in this work, fasttext embeddings-BiLSTM-CRF was employed. To establish initial settings, word embeddings were pre-trained using gensim package with vector size and windows size in ranges be 100 – 300 and 5 – 20, respectively [131]. As for the remaining hyperparameters, gensim default settings were used. Then, the embeddings were connected to a convolutional layer (Conv1D). This information fed a BiLSTM, which in turn, was input in a multihead attention (MHA) layer. This latter layer was included as some works in this area have reported to provide good results [170,181]. MHA applies an attention mechanism over the features which enable the algorithm to focus in the most important features for the task. Finally, these results input a CRF layer, which assigned the final label.

The deep neural networks were trained using Tensorflow 2 [132]. AdamW was selected as the optimizer. As a performance measurement, the micro-average f1-score was utilised. Then, the hyperparameters were evaluated to choose the optimal model. The values employed for each are listed in Table 5-4. In the selection of the most important factors, sequential sets of experiments were run. Firstly, the most important variables were filtered out, and values were selected using Placket Burman and Factorial designs. Then, the 2 most important variables were optimised by using a central composite design (CCD). These experiments were performed using paragraphs from the 49K patents employed in section 5.2.1. This text was used for embeddings training. As for the labelled data, the dataset which has the initial 2,000 paragraphs, without curation, was employed. Once, the best initial conditions were selected, confirmatory experiments were run to verify and optimise the final model.

Table 5-4 Assessed hyperparameters for selection and optimization of NER model.

Variable	Code	Layer	Range	Comment
max_len	A		83 to 113	When a RNN is trained using batches, it is necessary all the sequences have the same number of tokens. For this reason, a sequence max length is specified. When a sentence is shorter than max length, this will be padded. Otherwise, sentences are truncated, only considering the first tokens.
numb_conv 1d	B	Conv1D	1 to 5	Number of convolution layers
kernel_size	C	Conv1D	1 to 5	
filters	D	Conv1D	128 to 512	
dropout	E		0.3 to 0.7	
lstm_units	F	LSTM	100 to 300	Number of recurrent units in a LSTM layer
spatial_drop out	G		0.3 to 0.7	
mha_heads	H	Attention	1 to 8	Number of heads in a self-attention layer
mha_keys	I	Attention	32 to 128	Dimension of the key in a self-attention layer
emb_dim	J	Fasttext emebddins	100 to 300	Embeddings dimension for the pretrained fasttext model
emb_win	K	Fasttext emebddins	5 to 20	Embeddings dimension for the pretrained fasttext model
lr (log)	L		-2 to -3	Learning rate (optimizer)
wd (log)	M		-3 to -4	Weight decay (optimizer)
init	N		A, B	Initializer. Neural networks weights are usually initialised by taking values from a random distribution. However, depending on the library employed, the initializer changes, potentially affecting results. Which is why 2 methodologies were assessed tensorflow default (A) and pytorch default (B)

5.2.3.4 Model architecture optimization

Confirmatory experiments were conducted varying some additional conditions not included in the initial experimental designs. At this stage, the full training set was used.

In addition, fasttext embeddings were trained, varying windows size and embedding dimension, with the detailed description paragraphs of around 208K patents. The main objective at this point was to evaluate the effect of removing Conv1D and MHA layers. Similarly, the experiments included additional features such as character-level embeddings (CE), along with variations of dropout rates, as suggested elsewhere [180]. The best performing model was finally selected based on the highest micro-average f1-score for both development and test set. With the chosen model, an error analysis was carried out to evaluate the most common mistakes. Finally, code and embeddings employed for this work can be found in https://github.com/Diego-Alvarado/nlp_for_pharma_manufacturing.git and https://doi.org/10.15129/d688a529-cb11-4081-89a1-76bbd6a03f80.

5.3 Results and discussion

5.3.1 Relevant section identification

5.3.1.1 Preliminary results

Using LSA + k-Means on approximately 3.6 million paragraphs, the text could be grouped into 220 clusters. Based on cluster keywords, a total of 11 labels were established and assigned to each. This analysis initially provided a good insight into the type of information that might be found in patents. As expected, the diversity of data related to pharmaceuticals was abundant. Many aspects, in general, were covered in these documents, including invention description and background, pharmacological and therapeutical application, clinical evaluation, analytical techniques, manufacturing, among others. In information extraction applications, it has been suggested that models provide better results when trained in a domain-specific field [147]. As can be seen, many areas could be identified through paragraph analysis, whereby filtering information to only manufacturing-related data may be the best approach to ensure the best efficiency and quality at extracting information before carrying out a named entity recognition task over each piece of text. Thus, the idea of developing a methodology that enables the identification of relevant paragraphs was explored.

Since only a portion of all the collected patents was being analysed at this point, a neural network was trained with the 49K labelled patens to predict labels for unseen examples.

The labels and classification performance for each class are summarised in Figure 5-5. While a high f1-score was reached surpassing 90%, a paragraph-based approach showed to not be the most appropriate to select manufacturing data. The reason behind this is illustrated in Figure 5-6. Even though the algorithm selected the first two paragraphs accurately as relevant due to containing information on manufacturing, the remaining paragraph was classified into topic 9, which corresponded to chemical characterization. It can be noticed that although the third paragraph contained data on a purification process, there was also a description of mass spectrometry (MS) and nuclear magnetic resonance (NMR) results. Under these considerations, it is understandable why the algorithm classified the third paragraph as that topic. Nonetheless, if this decision was followed, this paragraph would have been omitted for entity extraction, thereby losing data related to the purification process. Like this example, many others were observed where paragraphs contain information on several topics which might cause a loss of data.

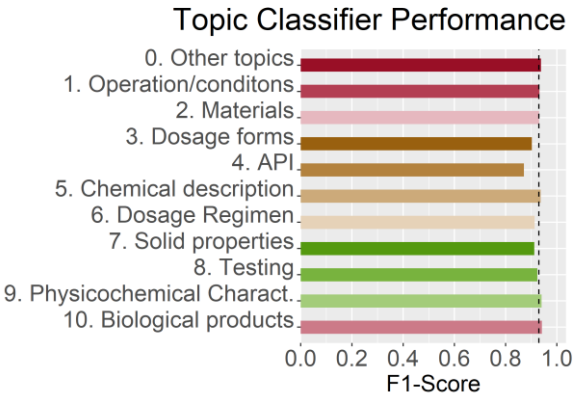


Figure 5-5 Paragraph topic classifier performance for the assigned labels.

On the other hand, it was also noticed that filtering by sections might be more helpful at the moment to collect all the relevant information. Most revised patents were structured in such a way that a section usually covered a specific topic, mentioning most steps and conditions. In the manufacturing case, this means the entire manufacturing process for a product is frequently described in one unique section. Going back to the example in Figure 5-6, if the entire section had been chosen, there would not have been a loss of data and, the purification data could have been included as well in all the process data.

Nonetheless, although the use of sections may be better, some drawbacks may still be found. For instance, there are situations when several sections cover different steps of

the process, or irrelevant content could be included. In consequence, this might hinder the structuring of the data. In this manner, the use of sections to filter have still some limitations, but still, being more promising than paragraphs. Thus, this highlights the importance of postprocessing to mitigate these constraints. Finally, a section-based selection was explored for filtering information of interest instead of paragraphs as more data could be retrieved and ease procedures organization, despite the limitations.

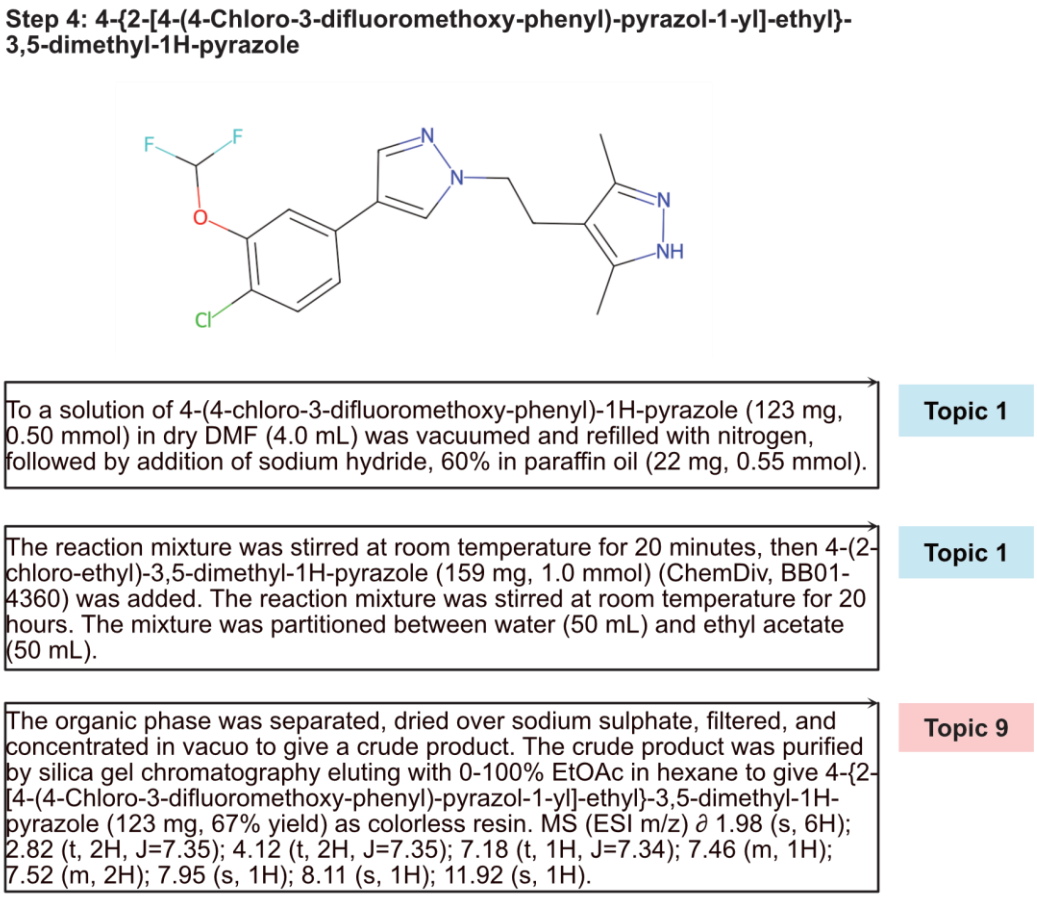


Figure 5-6 Example of undesirable results of paragraph classification approach.

5.3.1.2 Sections modelling

A total of 208,596 patents were used for topic modelling. These were equivalent to 5,542,816 different sections. With these data, LDA models were trained to determine potential topics. During this process, the effect of the order of the examples and inclusion of long texts was assessed. Regarding the first, it has been reported that LDA may suffer from “order” effect, which means that the order of the examples in the training set may

affect the results [182]. This was confirmed in Figure 5-7B where perplexity tended to increase every time the number of topics was greater. In contrast, when the order was randomized, the opposite behaviour was exhibited. Perplexity tends to decrease as a function of the number of topics, whereby shuffled data provided a behaviour that fits more with theory [80]. In this way, the results suggested that shuffling data is necessary to have reliable LDA models.

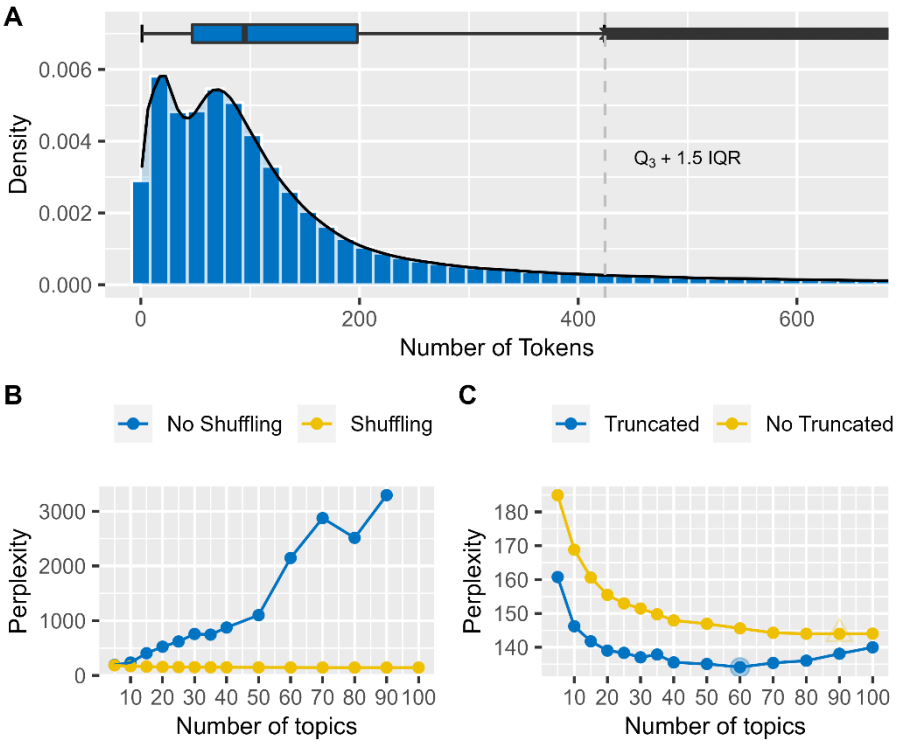


Figure 5-7 Distribution of the number of tokens per section (A) and Perplexities determined for LDA models evaluating shuffling (B) and truncation effect (C)

Regarding the effect of text length on model performance, the distribution for the number of words per preprocessed section and the effect of truncation can be seen in Figure 5-7A and C. Nearly 87% of all the sections contained 425 tokens or fewer. This point corresponded to the superior limit of the boxplot, by which, based on the IQR rule, values greater than this threshold can be suspected to be outliers. Thus, texts whose length was higher than $Q_{outlier}$ were defined as long. Observing Figure 5-7C, truncated texts showed lower perplexities throughout the assessed range, compared to texts without truncation. This difference even led to a different optimal number of topics, with values of 60 and 90 for truncated and non-truncated models, respectively. Furthermore, apart from the

difference in perplexity, the output quality was also affected, impacting on the topic interpretability as illustrated in Figure 5-8.

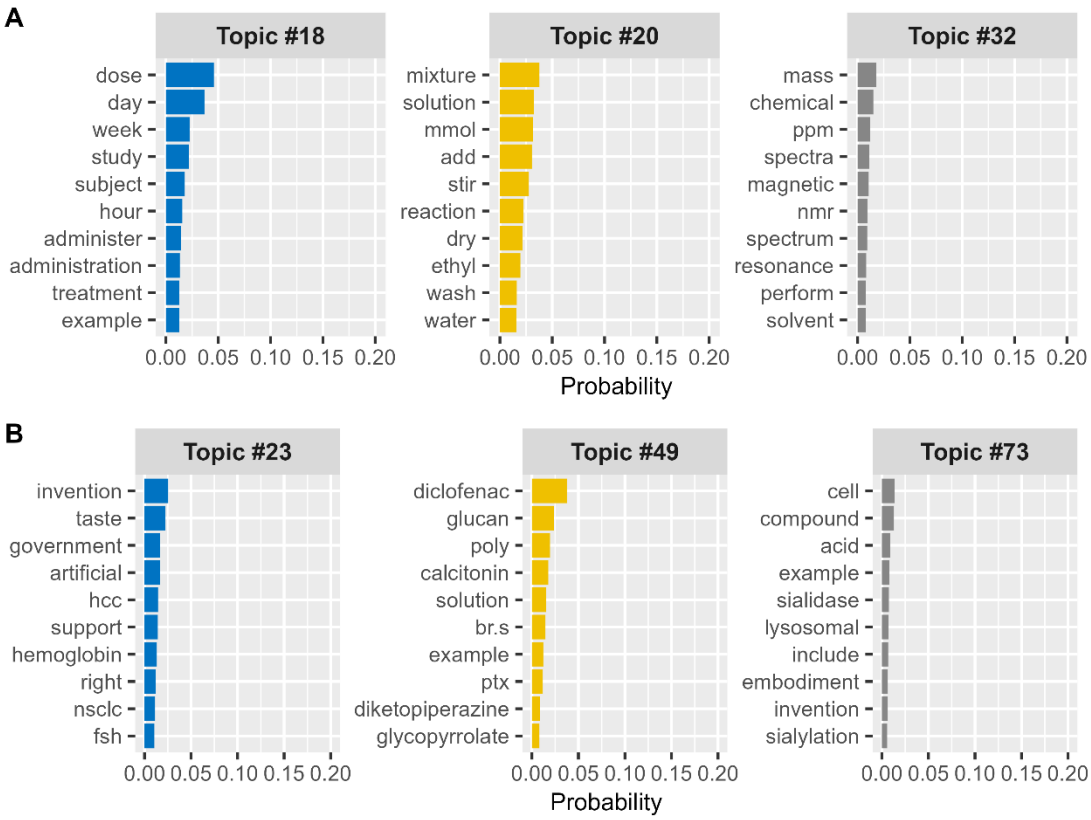


Figure 5-8 Top 10 keywords for truncated (A) and non-truncated (B) LDA models.

Overall, when the texts were not truncated, very specific words were found within the most important words. For instance, active ingredients or uncommon operations seemed to have a greater relevance (Topic 49 and 73 - Figure 5-8B). Another aspect observed in the sections without truncation was that the LDA model yielded topics with keywords that were difficult to interpret more often (Figure 5-8B – Topic 23). Lastly, while there were topics with a distribution that clearly favour certain words, some groups had keywords with more uniform probabilities (Figure 5-8B – Topics 32 and 73). In consequence, it is possible that these keywords did not generalise well the topics, by which the documents in these groups might not necessarily be related to what keywords suggested, reducing model interpretability and reliability. Once again, although this happened in both models, these kinds of topics were less frequent for truncated data. In this manner, the truncated model was concluded to be the best, with 60 latent topics. Furthermore, section truncation

was also demonstrated to generate more understandable topics. Keywords for both models can be found in Appendix B.

Using document representation for the optimal LDA model, k-Means algorithm was applied to cluster documents. The results for the selection of the optimal number of groups are displayed in Figure 5-9. The first factor analysed was the effect of data transformation. Across the assessed range and for both metrics, L2-norm outperformed square root transformation. Knowing this, the analysis next focused on selecting the optimal number of clusters for normalised data. The optimal points given by the DB and S scores differed. However, one of the main differences between both scores was the variability. the S score showed values with a higher scattering compared to the DB score. Therefore, the differences in S score might not have been as significant as with DB index. Due to this consideration, the selected number of clusters for k-Means was 60, which was the optimal considering the DB index.

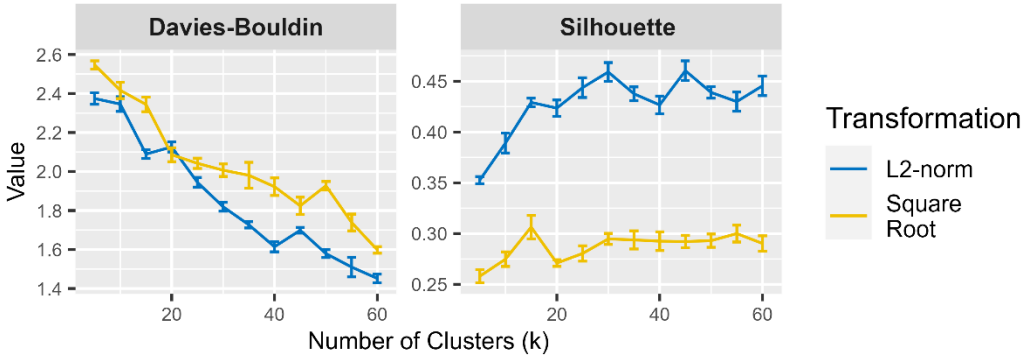


Figure 5-9 Performance for k-Means model changing the number of clusters.

5.3.1.3 Section selector performance

Figure 5-10 depicts the LDA representation for a sample of 2000 sections, using t-distributed stochastic neighbour embedding (t-SNE) to reduce dimensionality. A total of 17 labels were assigned to the clusters. It was possible to visualise how documents belonging to the same cluster and label tend to be close to each other. An example of this can be seen in the references 1 and 2 (black points) in Figure 5-10, whose texts can be seen in Figure 5-11. These two sections were about to synthesis procedures, and it can be seen how the algorithm correctly classified them into a primary manufacturing topic, having similar values. In the same manner, upon selecting labels related to manufacturing, as listed in Table 5-5, LDA representations allowed the agglomeration of

documents into two clearly defined regions about pharmaceutical manufacturing. These results supported that LDA enabled the numerical comparison of documents which are semantically similar. In addition, LDA + k-Means enabled the separation of information into interpretable topics, considering this work scope.

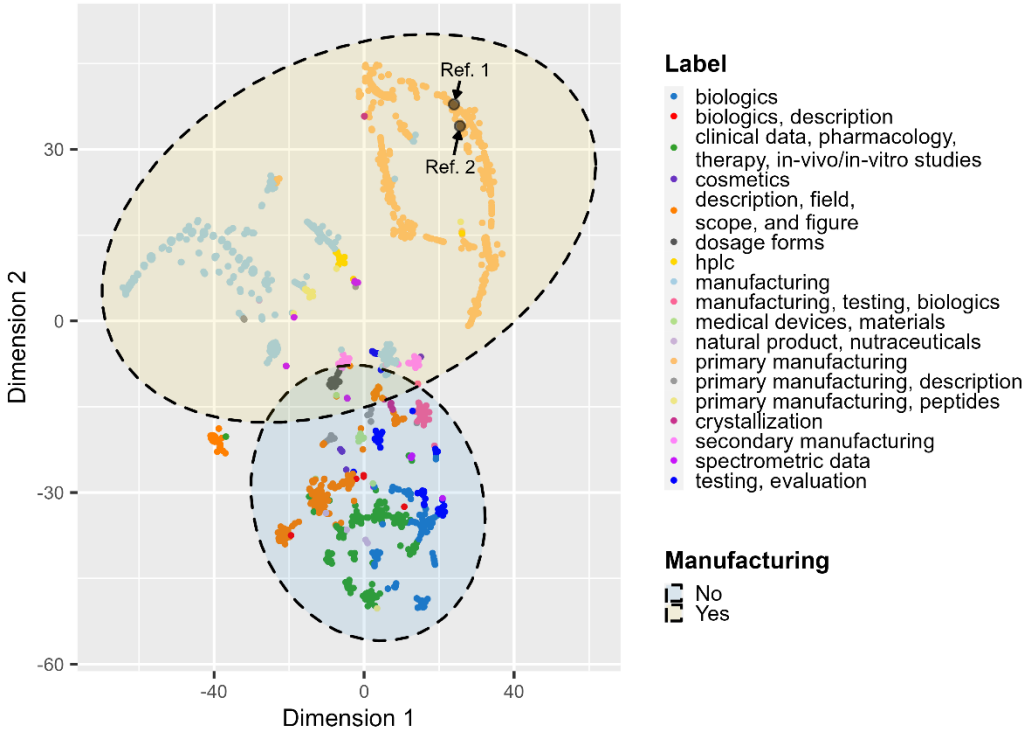


Figure 5-10 t-SNE visualization of document representations generated using LDA.

The results of the performance assessment are summarised in Table 5-5. In this case, the agreement between sections and labels assigned can be seen for the best and worst case. The overall percentage of agreement in the worst case was around 81%. The lowest degree of agreement was found for the cluster assigned with the spectrometric data label. The LDA topic most important for this label corresponded to the number 32, which was characterised for having keywords with uniform probabilities, as shown in Figure 5-8A. As discussed previously, this fact results in keywords with a poor power of characterisation, which results in low generalization. As a result, these keywords were not helpful to define a reliable label. Another unique aspect of the documents classified under this label was that LDA representations were widely spread with respect to the others. This could be evidenced in Figure 5-10where documents belonging to analytical topics were very sparse, and it was difficult to appreciate a dense cluster. In this manner,

this cluster in particular was not reliable and grouped sections that were not related to the assigned label. Nonetheless, despite this cluster behaviour, it is important to note that the cluster did not contain sections on manufacturing considering the revised sample. Likewise, the label was assigned as nonrelevant since it is focused on spectrometric results. Therefore, for manufacturing information retrieval, the sections belonging to this cluster were not considered for the subsequent data extraction step, which should not affect final results.

Ref. 1 (Patent Number US10590109B2)

Step 9

4-((2-(dimethoxymethyl)-6-(methylamino) pyrid-3-yl)methyl) morpholin-3-one

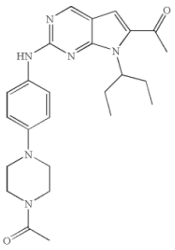
Compounds t-butyl-(6-(2-methoxyethyl)-5-((3-carbonylmorpholine)methyl) pyrid-2-yl)(methyl) aminocarboxylate 9i (70 mg, 0.18 mmol), trifluoroacetic acid (1 mL) and dichloromethane (4 mL) were mixed, and stirred for 6 h at room temperature. The mixture was alkalified with triethyl amine, and subjected to exsolution under reduced pressure. The residuals were purified through a preparative silica gel plate (petroleum ether/ethyl acetate 1:1), to obtain the target product 4-((2-(dimethoxymethyl)-6-(methylamino) pyrid-3-yl) methyl) morpholin-3-one 9j (46 mg, colorless solid), at a yield of 86%.

MS m/z (ESI): 296 [M+1].

Ref. 2 (Patent Number US8324225B2)

EXAMPLE 377

1-[2-[4-(4-Acetyl-piperazin-1-yl)-phenylamino]-7-(1-ethyl-propyl)-7H-pyrrolo[2,3-d]pyrimidin-6-yl]-ethanone



To a solution of 1-[2-chloro-7-(1-ethyl-propyl)-7H-pyrrolo[2,3-d]pyrimidin-6-yl]-ethanol (61 mg, 0.2 mmol) in CH_2Cl_2 (2 mL) is added Dess-Martin periodinane (242 mg, 0.5 mmol). The reaction mixture is stirred for 1 h, quenched with 10% NaS_2O_3:saturated $NaHCO_3$ (1:1) aqueous solution, and extracted with CH_2Cl_2. The extracts are washed with water and brine, dried over Na_2SO_4, and concentrated in vacuo. The residue is purified by flash chromatography (SiO_2, EtOAc/Hexane 1:3) to afford 58 mg of 1-[2-chloro-7-(1-ethyl-propyl)-7H-pyrrolo[2,3-d]pyrimidin-6-yl]-ethanone.

LCMS: 266 $(M+H)^+$.

By repeating the procedures described in example 201, using 1-[2-chloro-7-(1-ethyl-propyl)-7H-pyrrolo[2,3-d]pyrimidin-6-yl]-ethanone as a starting material, 1-[2-[4-(4-acetyl-piperazin-1-yl)-phenylamino]-7-(1-ethyl-propyl)-7H-pyrrolo[2,3-d]pyrimidin-6-yl]-ethanone is obtained.

LCMS: 449.4 $(M+H)^+$.

Figure 5-11 Example documents with similar content and classified in the same cluster (primary manufacturing) referenced in Figure 9.

Table 5-5 Degree of agreement between labels assigned based on LDA keywords and manually revised sections

Label	Related to manufacturing?	Worst Case	Best Case	Number of Clusters
Spectrometric data	No	20.0%	20.0%	1
Primary manufacturing, description	No	50.0%	60.0%	2
Biologics, description	No	60.0%	80.0%	1
Biologics	No	75.6%	86.7%	9
Testing, evaluation	No	76.0%	80.0%	5

Clinical data, pharmacology, therapy, in-vivo/in-vitro	No	80.0%	95.6%	9
Medical devices, materials	No	80.0%	80.0%	1
Natural product, nutraceuticals	No	80.0%	80.0%	1
Description, field, scope, figure	No	92.5%	95.0%	8
Manufacturing, testing, biologics	No	100.0%	100.0%	1
Cosmetics	Yes	60.0%	80.0%	1
Dosage forms	Yes	60.0%	80.0%	1
Primary manufacturing, purification, crystallization	Yes	60.0%	100.0%	1
Hplc	Yes	80.0%	80.0%	1
Manufacturing	Yes	80.0%	90.0%	4
Primary manufacturing, peptides	Yes	80.0%	100.0%	1
Secondary manufacturing	Yes	90.0%	90.0%	2
Primary manufacturing	Yes	100.0%	100.0%	11
Weighted Average by Number of Clusters		81.7%	89.3%	60

The remaining labels presented values greater than or equal to 50%, leading to an overall performance greater than 80% for both cases. This suggests that there is a high agreement between the information provided by the documents and the interpretation provided by the topic keywords and clusters, at least being better than a random guess. Given that there was a good level of agreement between the cluster interpretation and the grouped sections, it was possible to generalise to a higher level by directly relating the labels with the possibility of containing information on manufacturing, with a special emphasis on small molecules. Therefore, all the clusters with a label about manufacturing was marked as relevant. The list of labels with relevant and irrelevant information can be seen in Table 5-5. In this manner, this methodology enabled to indirectly determine what sections were of interest.

In the end, 22 out of 60 clusters were considered relevant. Although, as can be seen in Table 5-5, there were a few labels considered irrelevant that still mentioned aspects about fabrication and composition, these were discarded as pharmaceutically relevant because they only provided generic information; for instance, some sections listed all the possible dosage forms. Interestingly, when the agreement was calculated by relevancy, the group

of manufacturing had a higher level with 88.1% for the worse case, while 77.8% was obtained for the other group. Some possible reasons behind that were the presence of few sections that were related to manufacturing, and perhaps the most frequent, sections that could be associated with other similar labels. Finally, with model output and manual revision, agreement was measured using Cohen's kappa for manufacturing data. This resulted in a value of 91.1%, which can be considered acceptable [183]. In this way, the results suggested that the algorithm was able to distinguish sections with information of interest from those that were irrelevant. Nonetheless, it is worth to highlight that even though this approach may help to distinguish relevant information in an efficient manner, there still sections in the sample which were difficult to define thereby the final selection might still contain a reduced number of irrelevant information.

5.3.2 Named entity recognition

5.3.2.1 Experimental Designs for NER model optimization

The initial results and the effect of each factor can be observed Table 5-6 and Figure 5-12. The results showed the most important factor was learning rate (L), with the highest magnitude. The rest of the factors did not deviate significantly from normality. A possible reason was that the effect of L was so strong, particularly at a high level, that most of the scores were close to zero. In this manner, L obscured the effect of the other factors to a certain extent. Thus, in this run, the learning rate was concluded to be a factor that required to be optimized, given its magnitude. On the other hand, to confirm the effect of the remaining factor, a second Placket Burman design with 11 factors was performed.

Table 5-6 Initial Placket Burman design and results for hyperparameters selection. A: sequence max length (number of tokens), B: number of Conv1D layers, C: kernel size, D: number of filters, E: dropout rate, F: LSTM units, G: spatial dropout 1D rate, H: number of heads (MHA layer), I: keys dimension (MHA), J: embeddings dimensions, K: embeddings windows size, L: learning rate, M: weight decay, and N: initializer.

#	A	B	C	D	E	F	G	H	I	J	K	L	M	N	F1-score
1	113	1	3	512	0.3	128	0.3	6	96	100	20	0.001	0.0001	B	74.3%
2	113	3	1	512	0.3	256	0.5	6	32	100	5	0.001	0.0001	A	74.6%
3	83	3	3	128	0.3	256	0.5	2	32	300	20	0.001	0.0001	B	70.4%
4	113	3	1	128	0.3	128	0.3	2	32	100	20	0.01	0.001	B	42.2%
5	83	1	1	512	0.3	256	0.3	2	96	300	5	0.001	0.001	B	68.1%
6	113	1	1	512	0.5	128	0.5	2	32	300	5	0.01	0.0001	B	0.0%
7	113	3	3	512	0.5	256	0.5	6	96	300	20	0.01	0.001	B	0.0%
8	83	1	3	512	0.5	256	0.3	2	32	100	20	0.01	0.0001	A	6.0%

9	83	3	3	512	0.3	128	0.5	2	96	100	5	0.01	0.001	A	0.0%
10	83	3	1	512	0.5	128	0.3	6	32	300	20	0.001	0.001	A	65.3%
11	83	3	1	128	0.5	256	0.3	6	96	100	5	0.01	0.0001	B	0.0%
12	113	1	1	128	0.5	256	0.5	2	96	100	20	0.001	0.001	A	55.1%
13	113	1	3	128	0.3	256	0.3	6	32	300	5	0.01	0.001	A	0.0%
14	113	3	3	128	0.5	128	0.3	2	96	300	5	0.001	0.0001	A	72.6%
15	83	1	3	128	0.5	128	0.5	6	32	100	5	0.001	0.001	B	61.5%
16	83	1	1	128	0.3	128	0.5	6	96	300	20	0.01	0.0001	A	0.0%

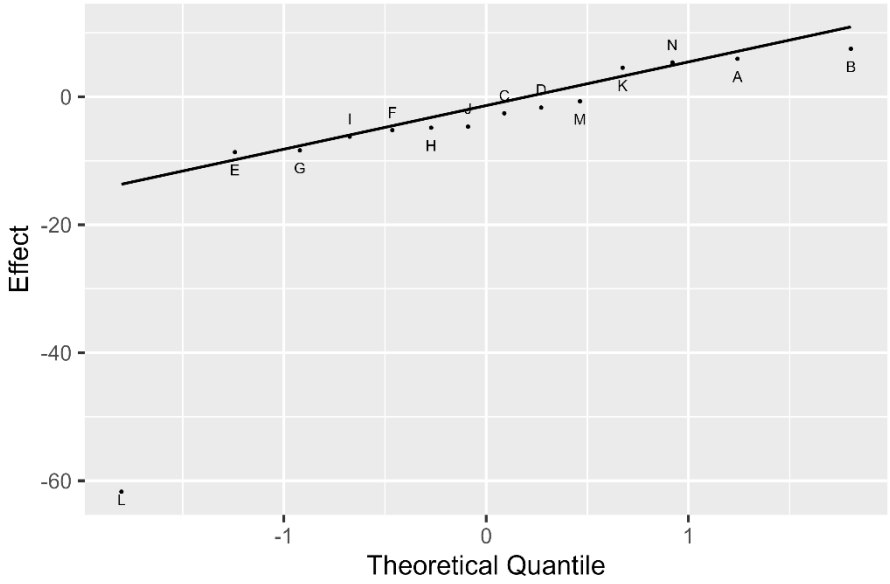


Figure 5-12 Normal plot of hyperparameters effects on F1-score for NER models. A: sequence max length (number of tokens), B: number of Conv1D layers, C: kernel size, D: number of filters, E: dropout rate, F: LSTM units, G: spatial dropout 1D rate, H: number of heads (MHA layer), I: keys dimension (MHA), J: embeddings dimensions, K: embeddings windows size, L: learning rate, M: weight decay, and N: initializer.

For the second Placket Burman, the factor L was fixed to 0.001. Similarly, the variables corresponding to max_len (A) and init (N) were also maintained constant with values of 113 and default initializer A. The max sequence length was adjusted at the highest level as the more tokens are included, the more likely is to retrieve entities which are mentioned in the last part of a string, for long sentences. In addition, this variable did not have a negative impact on model performance. Regarding the initializer, the default for TensorFlow library was set since, even though it has been pointed out that this factor may affect model convergence [64], it is not a parameter which is usually reported to be optimized for named entity recognition applications [92,172,180,181]. Apart from this, in the same manner as sequence length, this parameter did not affect significantly the

performance either. As for the remaining variables, some of the ranges were widened to confirm whether the variable definitely has no effect, or the levels employed in the previous experiment were not broad enough to observe a significant difference. The second design is shown in Table 5-7.

Table 5-7 Confirmatory Placket Burman design and results for hyperparameters selection. B: number of Conv1D layers, C: kernel size, D: number of filters, E: dropout rate, F: LSTM units, G: spatial dropout 1D rate, H: number of heads (MHA layer), I: keys dimension (MHA), J: embeddings dimensions, K: embeddings windows size, and M: weight decay.

Run	B	C	D	E	F	G	H	I	J	K	M	F1-score
1	4	1	128	0.3	300	0.3	6	128	100	20	0.001	40.7%
2	1	5	128	0.7	300	0.3	6	128	300	5	0.0001	69.1%
3	4	1	512	0.7	300	0.3	1	32	300	5	0.001	51.3%
4	1	5	512	0.3	300	0.7	6	32	100	5	0.001	48.8%
5	1	1	128	0.7	100	0.7	6	32	300	20	0.001	38.4%
6	4	5	128	0.3	100	0.7	1	128	300	5	0.001	28.0%
7	4	5	128	0.7	300	0.7	1	32	100	20	0.0001	34.2%
8	1	1	512	0.3	300	0.7	1	128	300	20	0.0001	69.6%
9	1	1	128	0.3	100	0.3	1	32	100	5	0.0001	67.8%
10	4	1	512	0.7	100	0.7	6	128	100	5	0.0001	58.0%
11	4	5	512	0.3	100	0.3	6	32	300	20	0.0001	64.9%
12	1	5	512	0.7	100	0.3	1	128	100	20	0.001	44.6%

The results of the second design are shown in Table 5-7 and hyperparameters effects can be seen in Figure 5-13. In this case, factor M (weight decay) had the greatest effect on performance, by which M was further optimized in a final CCD, along with the learning rate. The next hyperparameters were G (spatial dropout), which is related to E (dropout). Dropout is usually applied to deactivate randomly a pre-set proportion of units at the training stage, which mitigates overfitting [184]. However, an excess of deactivated units may hinder model training, whereby this value should be chosen carefully[184]. For both types of dropouts, the effect was negative by which a lower dropout favours model performance. Thus, this was fixed at 0.3.

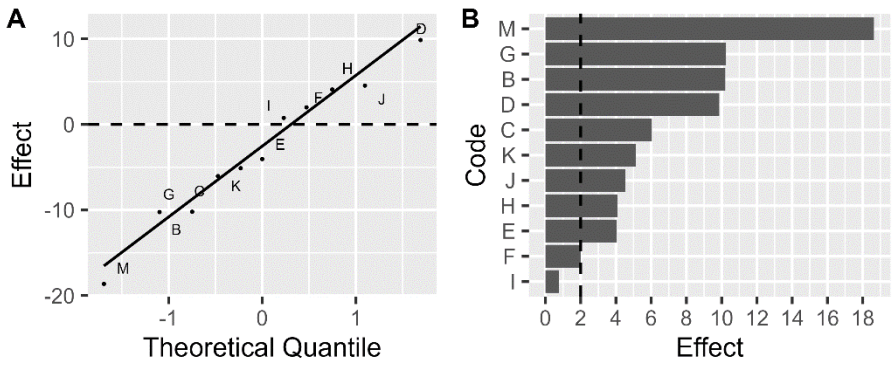


Figure 5-13 Normal plot (A) and magnitude (B) of hyperparameters effects on F1-score from Confirmatory Placket Burman design. B: number of Conv1D layers, C: kernel size, D: number of filters, E: dropout rate, F: LSTM units, G: spatial dropout 1D rate, H: number of heads (MHA layer), I: keys dimension (MHA), J: embeddings dimensions, K: embeddings windows size, and M: weight decay.

Next, the hyperparameters related to the convolutional layer such as B, C, and D also had an important effect. Whereas the effect of B and C was negative, D had the opposite. This indicates the highest performance can be obtained at the lowest levels of kernel size (C) and the number of layers (B). A convolutional layer has been suggested to help the model to detect semantic features that are more relevant to text classification tasks [185]. However, in this work architecture, Conv1D followed the embedding layer, by which many layers or a higher kernel size may cause excessive filtering resulting in a loss of information from word embeddings. This also leads to the question of whether a convolutional layer actually helps to retrieve more information. To evaluate this, a confirmatory experiment was carried out. As for the layer settings, it was also shown that a high number of filters (D) seems to be favourable for the model. In the end, the best kernel size and the number of filters were 1 and 512, respectively.

Regarding attention layer parameters H and I, although they seemed to increase f1-score, the magnitude of their effects was not significant compared to the others. In particular, the size of each attention head for keys (I) had an effect of around 0.7%. Since no major benefit was obtained from this parameter, for the next experiments, the middle point (64) was chosen. On the other side, the number of heads (H) had a greater effect with nearly 4.1%, therefore, the highest level was selected. However, the overall effect that this layer provides is lower considering the number of parameters that requires, thus, this layer was subsequently evaluated to see whether it improves NER performance.

The influence of hyperparameters related to NER main components such as embeddings and LSTM layer was moderate to low. The embeddings parameters windows size (K) and embeddings dimensions (J) behaved inversely. Pretrained embedding with a higher number of dimensions provided better results. This is aligned with previous works that claim that embeddings with higher dimensions tend to represent better semantic information of the words [88]. On the other hand, a lower window size favoured model performance. This parameter defines how many words behind and ahead are evaluated with respect to a particular word in a sentence to train embeddings. A wide window might cause the algorithm to lose focus on the target word, degrading the representation of semantic information for this data. LSTM number of units in turn had a behaviour where the highest number of units seemed to give a better contextual representation for entity recognition, although the effect was expected to be greater given the importance of this component, by which this was verified subsequently. In this manner, the best pre-trained embeddings had parameters windows size 5 and dimension 300.

To confirm the effect of the variables related to Conv1D, MHA and LSTM layers, whose magnitude was low, a third batch of experiments was run. For this case, a full factorial design was employed with two centre points. The number of headings for the attention layer was widened to ensure a larger range was covered. Conv1D parameters were set as aforementioned. The levels for LSTM units used in the previous experiments were maintained. The matrix design can be seen in Table 5-8 and results in Figure 5-14.

Table 5-8 Full Factorial design and results for hyperparameters selection. B: number of Conv1D layers, F: LSTM units, and H: number of heads (MHA layer).

Run	B	F	H	F1-score
1	2	200	5	75.8%
2	1	300	8	75.7%
3	1	300	2	75.9%
4	3	300	8	74.9%
5	3	300	2	76.0%
6	1	100	8	75.6%
7	1	100	2	75.7%
8	3	100	8	74.6%
9	3	100	2	75.0%
10	2	200	5	75.7%
			Average	75.5%

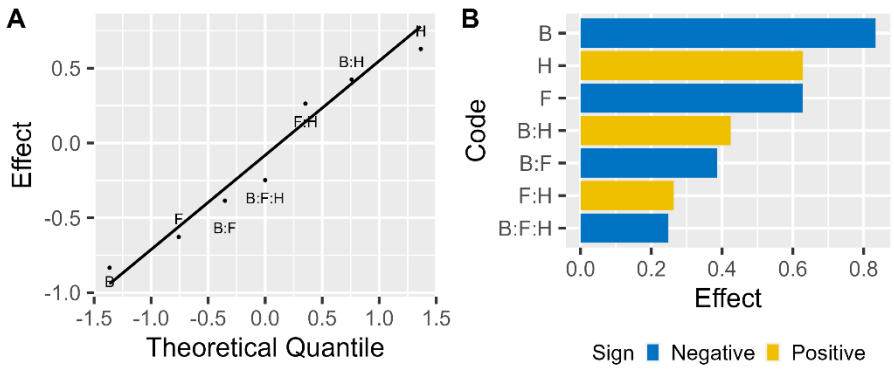


Figure 5-14 Normal plot (A) and magnitude (B) of hyperparameters effects on F1-score from Factorial design. B: number of Conv1D layers, F: LSTM units, and H: number of heads (MHA layer).

Factorial experiments yielded the overall effects were not significantly different. This can be seen in Figure 5-14, where no significant deviations from normal distribution were found. By observing effect magnitude, the negative impact of the addition of convolutional layer (B) on f1-score was confirmed, with this factor having the highest effect. Similar to the previous set of experiments, more LSTM units (F) seemed to increase recognition performance, but it only generated a marginal difference. Nonetheless, the number of heads (H) in the attention layer effect resulted in the opposite trend to what was observed, where more heads improved performance. There are two possible hypotheses about this behaviour.

On the one hand, the levels employed in factorial design were expanded to 2, 5, and 8, whereas previously 1 and 6 heads were assessed. The maximum performance was reached with 5 and 6 heads, which meant that a reduced number of heads may not have a positive impact, in the same way as an excessive number, as illustrated in Figure 5-15. As a result, the optimal value of this hyperparameter would be around 5. On the other hand, the obtained effects in this experiment were not particularly large, which might be due to the observed difference being random. The rest of the factors yielded a similar behaviour as exhibited in previous models. This led to conclude that the best hyperparameters for B, F, and H were 1, 300, and 5, respectively. However, the trends shown by the B and H did not confirm the benefits of the convolutional and attention layer on the improvement of entity recognition, reinforcing the need to carry out confirmatory experiments without these layers.

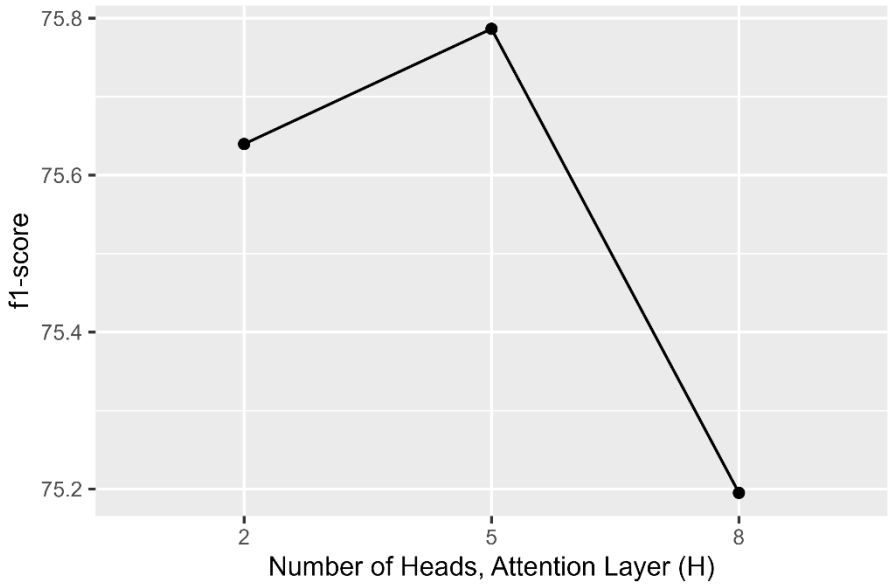


Figure 5-15 F1-score micro average vs the Number of Heads in MHA layer

Going back to the previous experiments, the parameters learning rate and weight decay turned out to be the most important. Thus, a central composite design (CCD) was used for optimization. The results of weight decay and learning rate optimization can be seen in Figure 5-16. The contour plot showed that extreme values of both hyperparameters were harmful to model performance. In addition, it was confirmed that the values employed in the previous tests were within the area where the highest f1-scores could be obtained (learning rate 0.001 and weight decay 0.0001). Learning rate has been reported in the literature to be perhaps the most important hyperparameter to optimize [64]. This statement is in line with the results obtained in this work. Finally, the optimal values yielded by the results were$10^{-4.1305}$ and $10^{-2.8566}$ for learning rate and weight decay, calculated through the fitted curve.

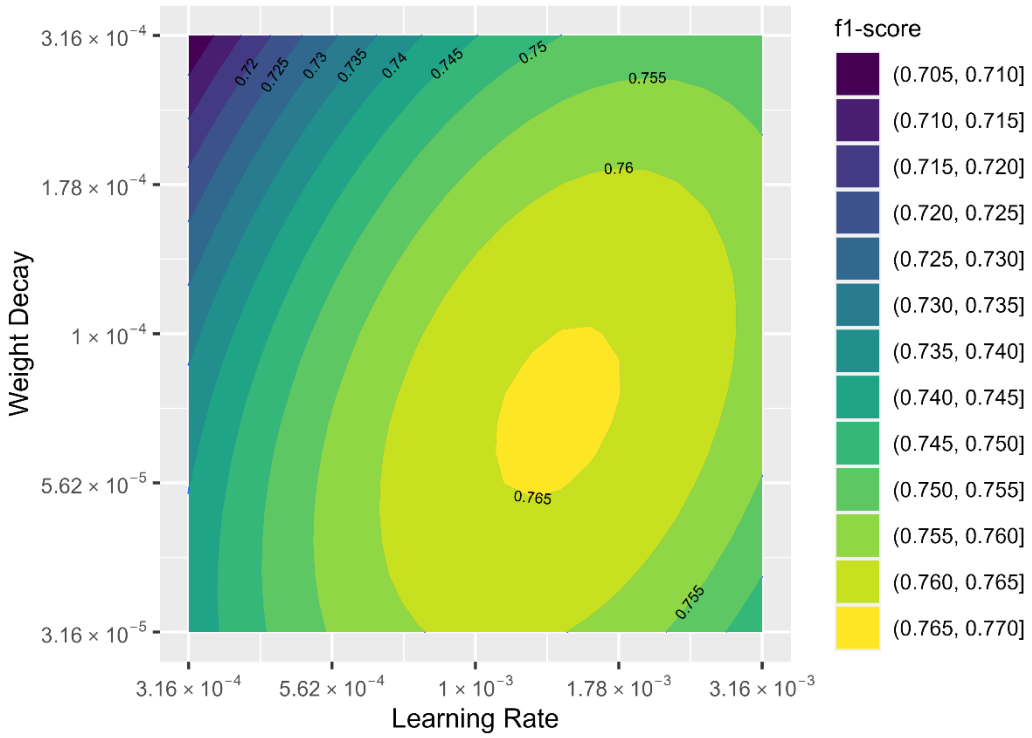


Figure 5-16 Contour plot Weight Decay vs Learning Rate.

To sum up this section, the best values for the majority of hyperparameters were defined using different sets of experiments. The most important variables were learning rate and weight decay, which are related to the optimizer component for training the model. As for the other features, the set values can be seen in the Table 5-9. To highlight, most of the variables had a consistent effect throughout all the experiments, however, there were still questions about the effect of attention and convolutional layers. This led to verifying the actual benefit of including these layers, by which additional tests were carried out to confirm this. In the end, at this point, a base architecture was defined, but the inclusion/exclusion of some parts of the model might improve performance further which is explored in the following sections.

Table 5-9 Summary selected values for NER model hyperparameters. *Layers to confirm

Hyperparameters	Selected Value
A: Sequence max length	113
B: Number of Conv1D layers*	1
C: Kernel size*	1
D: Number of filters*	512
E: Dropout rate	0.3

F: LSTM units	300
G: Spatial dropout 1D rate	0.3
H: Number of heads (MHA)*	5
I: Keys dimension (MHA)*	64
J: Embeddings dimensions	300
K: Embeddings windows size	5
L: Learning rate	$10^{-4.1305}$
M: Weight decay	$10^{-2.8566}$
N: Initializer.	A

5.3.2.2 Confirmatory experiments

The summary of the results can be seen in Table 5-10. Using as a base model the architecture without convolution or attention layers (BiLSTM), it can be seen that the self-attention layer did not enhance f1-score in any case. Instead, a reduction of 2.6% on average was observed when used. On the other hand, Conv1D seems to have a significant improvement in model performance, thereby confirming the effect of this layer discussed in the previous section, helping to filter relevant semantic features [185]. A similar effect was expected for the attention layer; however, this was not the case. In this manner, for the best model, the attention layer was discarded, whereas the Conv1D layer was demonstrated to be needed to have the best performance possible.

Table 5-10 F1-scores (micro-average) for confirmatory models in development and test sets. BiLSTM: Bidirectional Long Short-Term Memory, Conv1D: unidimensional convolution, Attention: multihead self-attention, CE: Character embeddings.

Model	Dev	Test	Average
BiLSTM	81.7%	79.6%	80.6%
BiLSTM + Conv1D	84.6%	82.0%	83.3%
BiLSTM + Attention	79.4%	76.7%	78.0%
BiLSTM + Attention + Conv1D	83.8%	82.0%	82.9%
BiLSTM + Conv1D + CE + Dropout (0.3)	84.7%	**83.5%**	84.1%
BiLSTM + Conv1D + CE + Dropout (0.5)	84.9%	83.2%	84.0%
BiLSTM + Conv1D + CE + Dropout (0.7)	**85.4%**	83.0%	**84.2%**

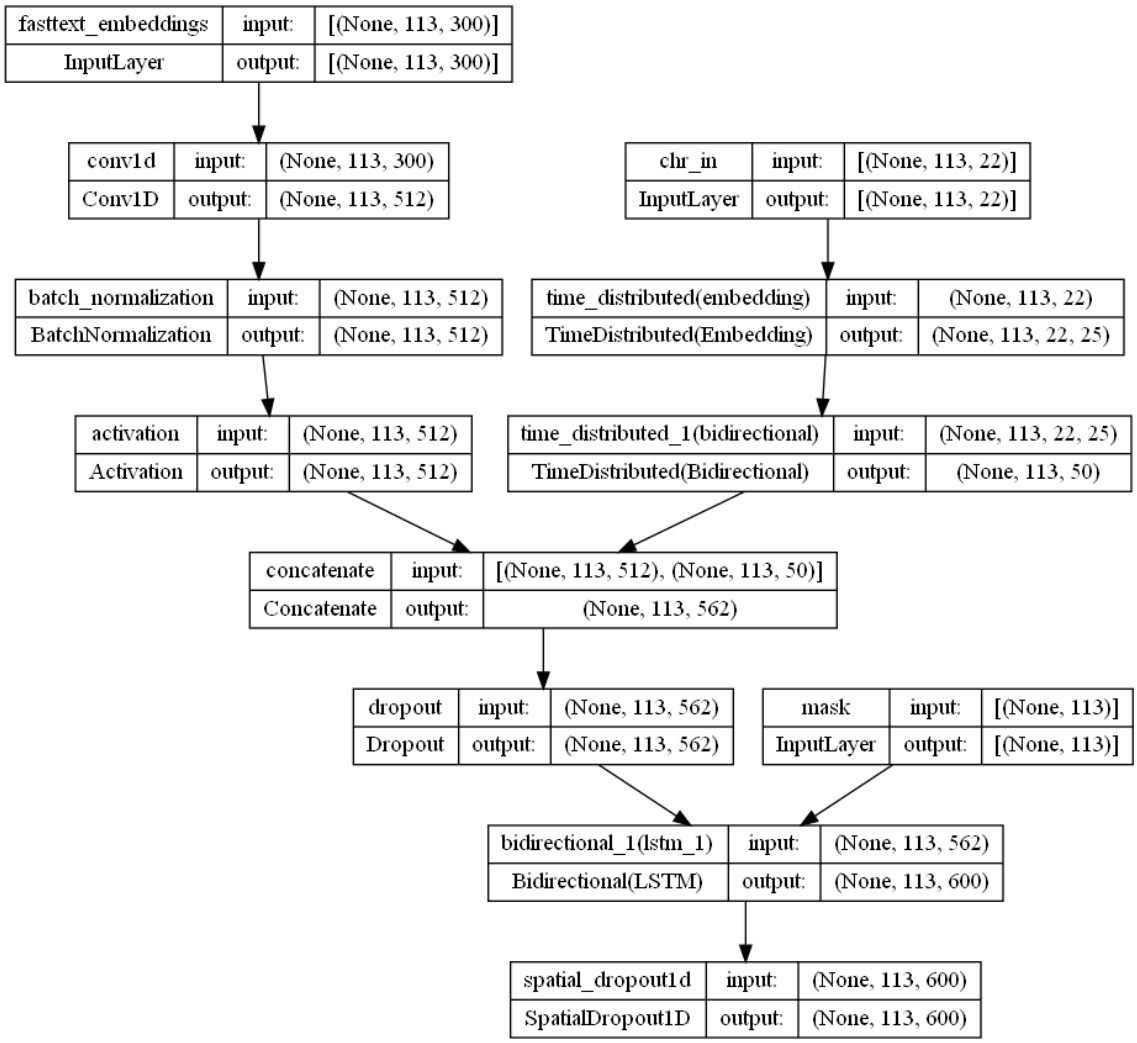


Figure 5-17 Best model architecture for NER

In addition to the inclusion or exclusion of specific layers, the confirmatory test also included the evaluation of the use of additional features such as characters embeddings (CE). CE have been proven to enhance NER performance in several domains [180]. To some extent, this is achieved by modelling other features not considered by word embeddings such as word morphology and spelling [180]. However, the effect of CE might be obscured by the effect of word features [180]. Due to this, some authors recommend adjusting dropout to maximise the effect of CE and model performance. Therefore, tests using CE with several levels of dropout were also assessed [180]. As observed in Table 5-10, the additional features increased model performance independently of the rates of dropout used, with f1-scores surpassing 83.0% in all cases. When comparing the

behaviour of performance metrics using different dropouts, a disparity in trends was observed. For the development set, the higher the dropout, the better the performance, while the test set resulted in the opposite. However, on average, the former trend was dominant, whereby the model with the highest dropout was selected as the best. The best model architecture is illustrated in Figure 5-17.

5.3.2.3 Error analysis

The performance of the final model for each entity is broken down in Table 5-11. Overall performance, in terms of precision and recall, was 84.9% and 83.5%. These values were comparable to other works in similar domains, where results revolved around 60 and 98% [147]. To highlight, the 3 most difficult entities to recognize were related to packaging materials, excipient type, and targets, having the lowest values for the different metrics. A key aspect of these types of entity is the low recall, which is translated into a loss of information during extraction. From the confusion matrix shown in Figure 5-18 Confusion matrix for NER, the most common error for these entities was not to be recognized or classified as irrelevant ("O"). Additionally, this error was also the most frequent among the other types of entities.

Table 5-11 Breakdown of the performance for best NER model.

Entity	Precision	Recall	F1-Score
AMOUNT	94.1%	95.0%	94.5%
API	88.1%	88.2%	88.1%
BIOLOGICAL_MATERIAL	87.2%	83.6%	85.3%
COMPOSITION	75.8%	81.1%	78.1%
CONDITION	89.3%	83.3%	86.2%
DOSAGE_FORM	80.7%	82.4%	81.5%
EQUIPMENT	80.2%	81.3%	80.6%
EXCIPIENT	83.3%	79.1%	81.1%
EXCIPIENT_TYPE	70.9%	64.6%	67.6%
INTERMEDIATES	78.5%	75.5%	76.9%
OPERATION	89.6%	89.1%	89.4%
PACKAGING	66.5%	58.0%	61.8%
REACTANT	81.2%	83.4%	82.3%
SOLVENT	88.7%	88.6%	88.7%
TARGET	73.2%	72.9%	72.6%
YIELD	97.4%	94.4%	95.7%
micro avg	84.9%	83.5%	84.2%

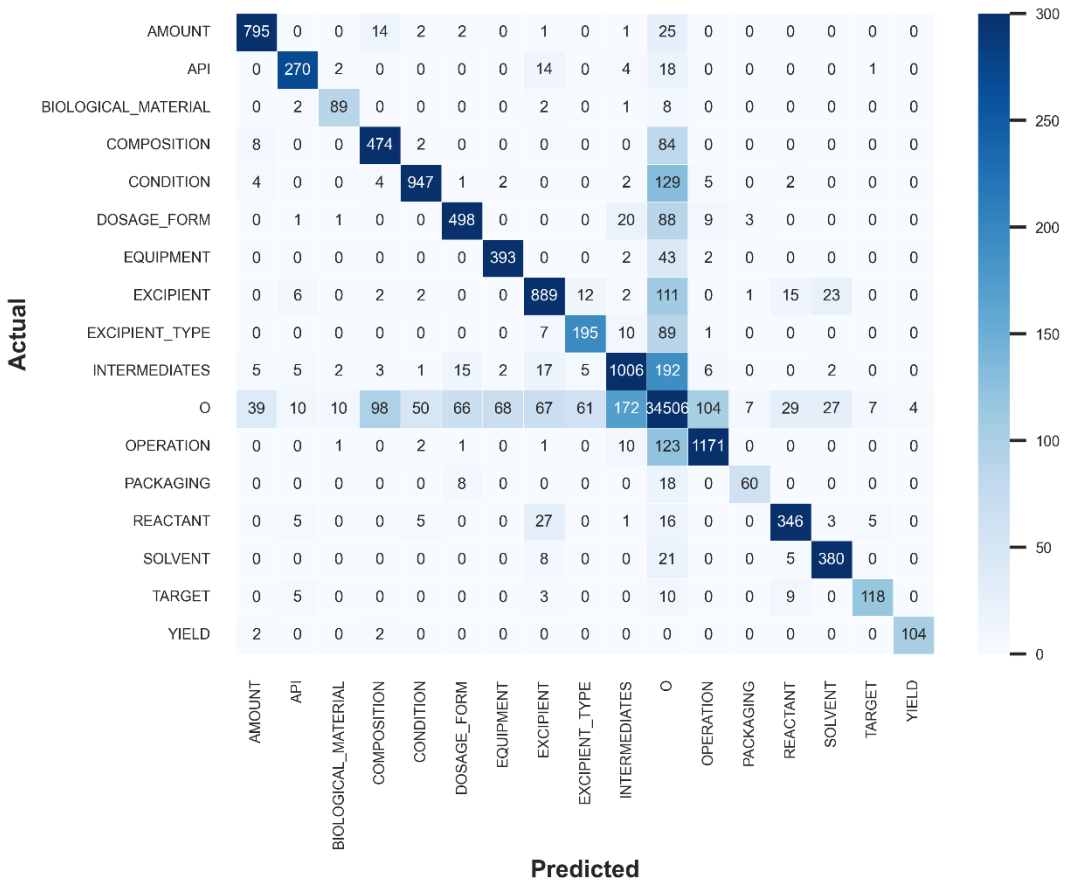


Figure 5-18 Confusion matrix for NER

Interestingly, in the specific cases of material-related entities such as targets or packaging, these were also misclassified into another material class such as reactant or dosage form, respectively. It was noticed that this behaviour could be extrapolated to any other similar type of entity, for instance, API, excipients, or solvents. Consequently, it can be concluded that the model was associating the materials with material-concept tags. Going to other types of entities such as amount, composition, and yield, these tended to be classified into one another, when misclassified. However, despite the error, the model still associated entities with similar concepts. In contrast, this trend was not observed in operations or conditions. Operations were sometimes tagged as intermediates or dosage forms were classified as operations. However, these cases were marginal compared to operations falling into the “O” class, representing 0.8% and 1.5% of the predictions, respectively. Furthermore, other types of errors were even less frequent. Thus, overall, the model yielded errors that could be considered reasonable in most cases. However,

for information retrieval and machine learning development, further processing and cleaning steps may be required to mitigate the effect of these errors on the reliability of the extracted data. As for the possible causes of these errors, these might be explained by two sources: context modelling and training set size.

In the first instance, sentences were used as inputs for NER model and, for this type of input, the surrounding words might have not sufficed to know whether or not a token belongs to a specific category occasionally. An example of this was seen with materials that can be used as excipient or reactant like Sodium Hydroxide or Hydrochloric Acid, where in a synthesis procedure, the model recognized them as EXCIPIENT. Another possible reason may be that the employed embeddings do not completely capture the differences between the entities. In this regard, the embeddings deployed for the present work generate a unique representation for each token. Recent developments have provided embeddings based on transformers that generate representations for tokens considering context [94,147,186]. In this way, the same word may have several representations depending on the other surrounding words. This approach has shown good results in other domains [94,147,186], which can be worth exploring as an opportunity for improvement in future works.

Going back to the cases of targets and packaging materials, a characteristic of these types of entity was that there were not many examples in the training set. Nearly 2.0% of the sentences had terms related to packaging materials, while around 2.6% contained tokens about targets. In this manner, the inclusion of examples containing this type of entity with a greater diversity of terms might help the model learn to identify these better. Although it was observed that these entities were not mentioned very often throughout the patents compared to not being identified. In particular, the main focus of the studied patents was formulations and packaging materials were not mentioned. In the case of targets, a common pattern in the analysed document was the implicit reference; thus, chemical structure or name was described in different sections. Additionally, these entities were not abundant in the corpus, hindering model learning.

5.4 Summary

In this chapter, a set of tools was developed to aid the extraction of information on primary and secondary manufacturing from patents. The process by which these tools were

trained and validated is discussed throughout the chapter. The first component comprises a section selector. This model filters sections that potentially contain information on manufacturing, with special emphasis on small molecules. This made use of unsupervised approaches such as LDA and k-Means to group documents and provide interpretation for each cluster based on keywords. The interpretability and agreement of the cluster topics were validated by manual revision of a sample of sections. The overall degree of agreement between assigned topic labels based on the cluster keywords and the actual section topic was 81.7%, in the worst case. For the identification of relevant documents, the agreement was measured using Cohen's kappa, obtaining a value of 91.1%. This suggests that there is a good level of agreement between the output of the model and the actual section topic. In this manner, this approach has been proven to be useful to select relevant fragments of text. Then, once the fragments of text containing the relevant information have been filtered, it is necessary to apply another algorithm that recognizes the specific information we are interested in extracting.

To accomplish this, a NER model was trained. This model consisted of a deep neural network with a base architecture comprising word embeddings, a contextualiser (BiLSTM) and a decoder (CRF). After optimizing hyperparameters and neural network architecture, the best architecture was determined. During optimization, the hyperparameters related to the optimizer (learning rate and weight decay) were found to be the most important. In addition to this, the inclusion of a convolutional layer and additional features like character embeddings proved to be helpful to enhance entity recognition performance. The overall performance for the NER model, measured using the micro-average f1-score, was 84.2%. The most common errors were associated with the inability to detect an entity, followed by confusion between recognized entities, to a lesser extent. For instance, a material which is a final product of a synthesis (TARGET) could be confused with a reactant. However, these kinds of error seem to be occasional. Future work focused on improving the NER algorithm may focus on using word embeddings based on transformers. Approaches such as bidirectional encoder representations from Transformers (BERT) are more context-sensitive and have provided promising results in other domains helping models to characterise words whose meaning is dependent on context. However, the developed model showed to be able to recognize the required information with a high level of accuracy, although some errors

may still occur. Due to this, when applying, post-processing steps are recommended to mitigate the effect of errors in dataset construction or machine learning application reliability using the data extracted by these approaches. This will be discussed in more detail in the next chapter.

Chapter 6. Deep Generative Models for Primary Manufacturing Process Design

6.1 Introduction

In primary manufacturing, deep learning (DL) has become a powerful tool to assist process design. With novel emerging techniques and substantial efforts done to build large datasets, multiple models have been trained to achieve tasks in specific areas in material discovery, and synthesis route prediction, being the latter perhaps the most relevant in manufacturing. Specifically, it is possible to find commercial platforms based on DL for retrosynthetic planning such as IBM RXN [187]. In addition, these developments are not only limited to this task but there are also tools to predict sequences of operations to perform a particular reaction [43]. Complementing this, deep neural network architectures in the prediction of reaction yield, conditions, and optimal solvent can also be found [43,188,189]. In this manner, providing a set of models, that in conjunction, can cover various considerations in reaction planning. This could in turn allow a more informed, rapid, and autonomous process design, by targeting feasible chemical reactions with their respective process and conditions, saving time and resources.

To highlight, many of these options employ sequence-to-sequence or translation-based approaches to predict a synthesis path for a target molecule. On the other hand, there have been proposals using deep generative models (DGMs). In this sense, various models have been cited previously in this work such as the developments performed by R. Tempke et al and S. Li et al [60,61]. Although sequence-to-sequence models have shown promising results, it has been argued that these are more likely to inherit biases from datasets [60]. Typically, these biases can be related to that sources mainly report successful experiments [60]. In this manner, the possibility of discovering new routes or alternatives can be restricted to interpolation based on existing data. This has become a strong driver to explore the applicability of DGMs in different aspects of synthesis planning given their capacity to generate examples not observed in the training set and provide diverse options. Thus, apart from chemical reaction generation, the application of DMGs, such as variational autoencoder (VAE), has also been tested to predict reaction conditions [190,191].

Nonetheless, as discussed throughout this work, the availability of models for the generation of sequences of operations is limited. The main development in this area, proposed by A. Vaucher et al, utilises translation-based approaches [43]. This model

involves transformers architectures that, processing information on a chemical reaction expressed as a reaction SMILES, outputs a sequence of actions and conditions to produce a given target molecule. Thus, the model has demonstrated the potential of DL in generating experimentally valid procedures and has established a foundation in this area [43]. Aligned with this, DGMs could expand the application of DL for manufacturing sequence generation, offering solutions to the limitations faced by traditional models. As discussed previously, the low degree of generalisation of predictive approaches could be overcome by generative modelling as these would allow learning more complex relationships and distributions. These might facilitate a wider exploration in the manufacturing space to discover different routes and attain more generalisability.

Thus, this chapter presents the steps taken for the design and training of DGMs. These are summarised in Figure 6-1. In particular, the following aspects are discussed: the development of a training set, evaluation of different architectures, and applicability and limitations of these models. As with any DL model, a large number of data is required. In this manner, making use of the tools developed, a training set for sequence prediction is built. Previously, it has been mentioned models can inherit biases from the data. Therefore, the training set is described, and their limitations are explored. As for the modelling, considering the selection of operations must be customised according to the chemical reaction materials, there is a special focus on conditioned models. Going more into detail about the models, frameworks such as adversarial autoencoder (AAE) and variational autoencoder (VAE) are assessed along with the effect of variations in architecture and hyperparameters. Finally, performance metrics and limitations are also analysed to provide insights into the feasibility of employing these models in real-world manufacturing scenarios.

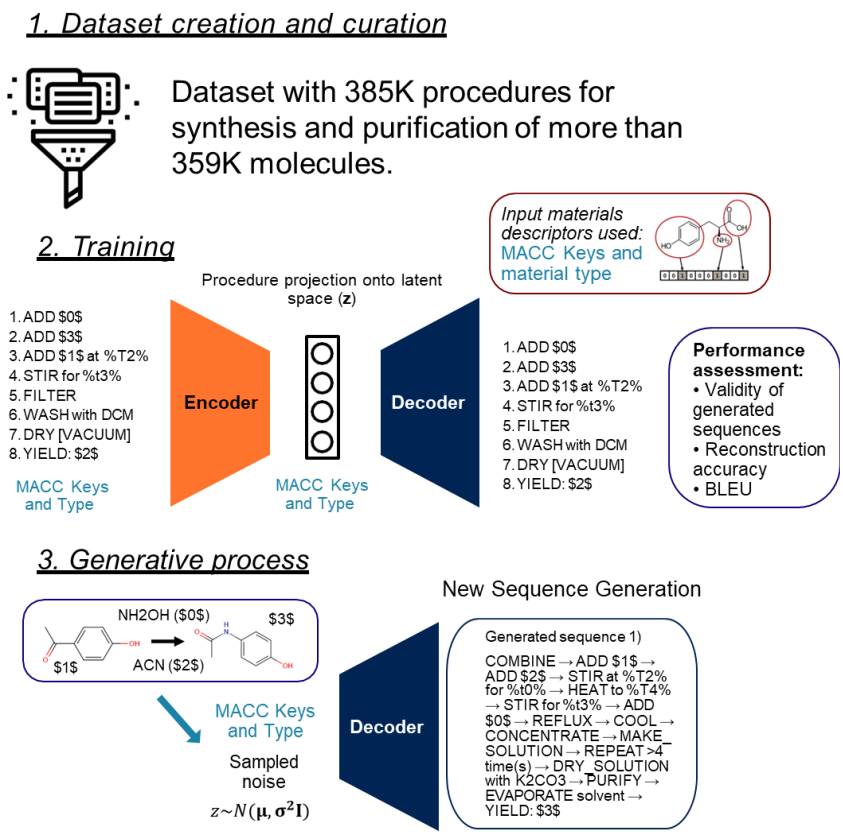


Figure 6-1. Steps for generative model development for manufacturing routes generation for primary domain.

6.2 Methods

6.2.1 Dataset development

In the previous chapter, ML models were developed to identify sections about manufacturing and then extract keywords for primary and secondary processing. Thus, sections focused on synthesis and purification were selected and named entities were extracted using the NER model. In this regard, sections that mentioned keywords related to dosage forms and excipients were discarded. In the same manner, records without operations were also excluded. On the other hand, procedures whose target could be associated with an InChI identifier were considered for further analyses. This resulted in a dataset with 716,707 records in 30,854 patents, where each represented the steps for a single reaction or purification of a unique target molecule. With this initial dataset, terminology was then standardised as indicated in Chapter 3, subsection Dataset

Cleaning and Curation. Nonetheless, specific details for the harmonisation of conditions, materials, and operations are provided below.

Firstly, materials classified as solvents and reactants were processed by removing stop-words. The terms that only consisted of anions were removed from the dataset based on a list, given that the information on the molecule entity was assumed to be incomplete or ambiguous. Then, InChI identifier was searched for each material using Pipeline Pilot software or pubchempy module in python. To tackle spelling mistakes of the unidentified materials, rapidfuzz module was used to determine the matching between the remaining unidentified terms and a dictionary built from the entities with InChI. This was only applied to the most common substances. For the standardisation of intermediates, conditions, and operations, it proceeded individually for each type of entity as indicated in Chapter 3 to generate clusters of entities. Then, through the most representative words of each cluster (closest words to the centroid), a label was assigned. For conditions, each cluster depicted the type of conditions such as temperature or time. These were further processed by separating magnitudes and units to later convert all to the same units. This was applied mostly for temperature and time, as these were the most prevalent conditions.

On the other side, the clusters of operations or intermediates gathered synonyms of the same action or type of substance, respectively. In the particular case of operations, the assignation was verified manually by revising all the terms. At this point, new assignations or types of actions were added since some terms found did not fit accurately with the initially allocated definition of the clusters. After this first validation, a second adjustment was carried out by considering the related materials and/or conditions. For instance, if a term had an ambiguous type of action, like ‘increase’, and was associated with a condition related to temperature, the type of action was reallocated to ‘heat’, or if the operation corresponded to a ‘dry’ action associated with a material such as ‘magnesium sulphate’, the operation was reassigned into ‘dry solution’ class. As a reference for the harmonisation of actions, the concepts and definitions proposed by [95,103] were adapted to this work. A list of the standardised terms used can be found in Appendix A. Then, the resulting dataset was validated by drawing a random sample of 100 procedures and manually comparing the observed information to the source. Recall, Precision, and F1-

score were estimated using this information. Finally, additional data was excluded according to the criteria described in Table 6-1 to be used in generative model training.

Table 6-1. Exclusion criteria of manufacturing procedures from the dataset for model training. [a] Duplicates corresponded to patents that reported the same molecules and procedures but with a different scope.

Exclusion criteria	Number of records discarded
1. Duplicate procedures and target [a]	231,958
2. Presence of ambiguous or irrelevant operation types such as “follow another procedure”, “secondary manufacturing”, “shape”, “decolourise”, “set”, “reduce”, “increase”, “return”, “replace”, “proceed”, and “others”	14,897
3. Number of steps lower than 2 and higher than 28 (excluding “no altering” or “no action” operations) b	35,592
4. More than 11 materials (target + solvents + reactants)	1,241
5. Finishing with operations related to “add”, “make solution” or starting with “remove”	6,537
6. Likely associated with multiple reactions	29,840
7. Less than 3 types of operations based on classes defined in Appendix A Table A1, errors in the order of operations, or multiple repeating steps	11,349
Final number of records	385,293

Ideally, it is expected that a manufacturing sequence provides a minimum amount of information useful to execute the process. Therefore, brief sequences with only one reported step (3) or with ambiguous terms (2) were excluded. Similarly, it was desirable that a procedure reported different or diverse types of operations that described, as accurately as possible, how materials were manipulated and/or the different transformations that these underwent to yield the target molecule. In this manner, the following five operation categories were defined founded on the terms observed in the final dataset: material handling, material mixing, parameter setting, chemical transformation, and separation methods. Thus, sequences with operations falling into at least three of these categories were maintained for model development (7). Finally, it was expected that a chain of operations maintained a logical order or consistency in terms of repetitions, transformations, or typical associated actions. An example of this criterion was that sequences with three or more consecutive operations being the same were discarded. Addition criteria, such as (3) (> 28 steps) and (4) aimed at optimising

computational cost and removing atypical sequences, were taken into account. Finally, the resulting dataset was randomly split into 80%, 10% and 10% for training, development, and validation, respectively. The final dataset can be found in https://zenodo.org/records/13377654.

6.2.2 Output representation

Manufacturing sequences were represented using two approaches as can be observed in. Table 6-2. Representation A only consists of a list of operations found in the procedure. This representation was employed in exploratory experiments. On the other hand, representation B was based on the implementation of action prediction using transformers by Vaucher et al [43]. Apart from operations, the sequence includes placeholders to associate materials and target molecules with the operations through an index within $$. Similarly, conditions of temperature, time, and pH were also included in this representation through a class within %% symbol. The number within %% indicates a range of the specified condition. The equivalency of classes and ranges is described in Appendix C. Finally, a YIELD token is used to indicate the target molecule and the process end. Compared to A, this representation provides more information about the process and is more readable. B was also used to train models and compared to Vaucher et al work.

Table 6-2. Examples of manufacturing sequences representations. $$: material; %% condition; T: temperature; t: time; if no letter is included within %%, it refers to pH.

Patent No. US8859535B2	**Output Representation A**	**Output Representation B***
Intermediate 178.2 (4-Bromophenyl)-(2-oxabicyclo[2.2.2]oct-4-yl)methanone To a solution of ***(4-Bromo-phenyl)-(2-oxa-bicyclo[2.2.2]oct-4-yl)methanol*** (224 mg, 0.821 mmol) in ***DCM*** (13 ml) was added a total of ***manganese dioxide*** (1.414 g, 16.264 mmol) at RT and the reaction mixture was stirred for 20 hrs. The suspension was filtered through a pad of hyflo and the residue was washed with ***DCM***, dried in vacuo to yield the title intermediate (220 mg, 0.745 mmol, 91%) as a colourless solid.	1. ADD 2. STIR 3. FILTER 4. WASH 5. DRY [VACUUM]	1. ADD 0 2. ADD 3 3. ADD 1 at %T2% 4. STIR for %t3% 5. FILTER 6. WASH with DCM 7. DRY [VACUUM] 8. YIELD: 2

6.2.3 Input representation

A key characteristic of the generative models was the capability of customising the generated sequence according to a target molecule or material properties. To ensure this, models were trained in a supervised manner by inputting information on either target or precursors + target. However, various options can be employed to represent molecules in ML. For instance, molecular descriptors have been historically used. In the same manner, different types of fingerprints have shown good results for distinct machine-learning applications [192,193]. Additionally, deep learning has enabled the usage of discrete representations such a SMILES and molecular graphs for properties prediction as well as molecule generation [193,194]. As can be seen, there are multiple alternatives to represent molecules. Thus, the first step of model design was to define which of these may be the most appropriate for manufacturing sequence generation.

In order to assess what representation could potentially provide better performance on generative modelling, their capability of sequence prediction was studied. Thus, several deep neural networks were trained using different types of representations. The evaluated representations are described in Table 6-3. Initially, as an exploratory stage, the predictability of representation A was carried out by modifying different settings of the models. Subsequently, to compare to other models available, deep neural networks to predict representation B based on Transformers proposed by Vaucher et al representation B were adapted.

Table 6-3. Assessed molecular representations for generative modelling.

Representation	Description
Molecular descriptors	Includes 1D and 2D descriptors. 1D groups atoms counts and physicochemical properties such as LogP, polar surface area, etc. 2D refers to molecule fragments and connectivity indices[195].
Simplified molecular-input line-entry system (SMILES)	Consists of sequence of ASCII characters assigned following rules based on molecular graphs and connectivity[196].
Molecular ACCess System (MACCS) Keys	Binary code of 166 bits where each value represents the presence or absence of a molecule substructure[197].

Min-hashed Fingerprint (MHFP-2048)	Sequence of integers from 0 to 2047. Each integer decodes a substructure of a molecule. Proposed by Capecchi et at in 2020[192].
SVAE	VAE to generate SMILES was pre-trained and the latent space obtained was used as input in the predictive models, the architecture of this model was adapted from Dollar et al[127]

6.2.3.1 Prediction of actions

Going more into detail about the architecture employed in the prediction, it was expected the different types of networks may yield different performances. In this manner, if only one type of architecture were used, it could be argued that the network employed may not have been able to use the information provided by the representations efficiently to produce good results. To study the effect of the inputs more accurately, two different types of neural networks were employed, one based on recurrent neural networks (RNN) and another using Transformers. The hyperparameters applied to each of these neural networks are described in Table 6-4. Subsequently, models to predict sequences were trained using as outputs only operations and varying target molecule representation.

When SMILES was used as a representation and fed directly into the model, an additional module was included. This module consisted of an embedding layer, which received SMILES characters, with a dimension of 300; 3 stacked GRU layers with 300 hidden units; an additive attention layer; a global max pooling; and finally, a dense layer with 128 units. Both, RNN and transformer networks, were trained using Adam optimizer. The learning rate was set in 5e-4 for RNN, while a learning rate scheduler was used for transformers where the learning rate started from 0 and increased linearly every step for 15 epochs to 1e-4. The training was initially programmed to continue for 150 epochs. BLEU score was monitored for a random sample drawn from the test set in each epoch. However, this was stopped early if there was no improvement in the performance metric for 30 consecutive epochs. Teacher forcing was employed to ease learning. Subsequently, additional transformers and RNN networks were trained using as inputs randomly generated vectors as control. Categorical cross entropy was used as a loss function.

Table 6-4. Sequence prediction networks. Architecture taken and adapted from the decoder proposed by Dollar et al[127]. [a] transformer blocks were implemented as proposed by Vaswani et al[112].

Layer type	Activation	Hyperparameters
RNN		
Input		Max sequence length: 30
Embedding		Embedding dim: 30
Dense	Relu	Units: 128
GRU (Layer 1)	Tanh	Units: 128
GRU (Layer 2)	Tanh	Units: 128
GRU (Layer 3)	Tanh	Units: 128
Dense	Softmax	Units: No. Operations
Transformers		
Input		Max sequence length: 30
Embedding		Embedding dim: 300
Transformer encoder [a]	Relu	No. blocks: 1 Heads: 1 Key dim: 512 Feed forward: 300
Transformer decoder [a]	Relu	No. Blocks: 1 *Self-attention* Heads: 1 Key dim: 512 *Cross attention* Heads: 1 Key dim: 512 Feed forward: 300
Dense	Softmax	Units: No. Operations

With the descriptors that provided the highest BLEU, a transformers network was then trained using as inputs target + precursors. Materials not related to operations of separation or purification were chosen as precursors. Thus, the inputs of each procedure were represented as a matrix with dimensions (number of materials, representation dimension). Transformers-based model was chosen to model inputs with this structure as this layer is permutation invariant. Therefore, the order of materials in the matrix representation does not affect the result.

6.2.3.2 Prediction of actions associated with materials and conditions

On the other hand, the assessment of the prediction of actions along with materials and conditions was limited to molecular descriptors and MACC Keys. The other representations were not included at this stage due to computational cost and performance. This is discussed in Results and Discussion section. In this case, the

modelling was centred on Transformers-based architectures and the settings are specified in Table 6-5.

Table 6-5. Sequence prediction networks for representation B. Adapted from Vaucher et al [43]

Layer type	Activation	Hyperparameters
Transformers		
Input materials		Max number of materials: 11
Input sequence		Max sequence length: 81
Layer normalisation		
Dense	Identity	Units: 256
Positional embeddings for materials		Embedding dim: 256
Dropout		Dropout rate: 0.4
Embeddings for operations		Embedding dim: 256
Transformer encoder	Relu	No. blocks: 4 Heads: 8 Key dim: 256 Feed forward: 2048 Attention dropout: 0.3 Dropout rate: 0.4
Transformer decoder	Relu	No. Blocks: 4 *Self-attention* Heads: 8 Key dim: 256 *Cross attention* Heads: 8 Key dim: 256 Feed forward: 2048 Attention dropout: 0.3 Dropout rate: 0.4
Dense	Softmax	Units: No. Operations

Compared to Vaucher's model, the main difference with the implementation in this work was related to how material information was processed. In the original model, precursors and targets are expressed as reaction SMILES. Since that information was not available in our dataset, materials were expressed as a set of features with dimensions (number of materials, number of features). Within the features, in addition to the descriptors of each material, a 3-component vector was introduced where each value corresponded to 0 or 1 depending on the material role in the process. Namely, 1 was assigned if a given material was classified into TARGET, REACTANT or SOLVENT, otherwise, the value

corresponded to 0. Additionally, contrary to the previous models for representation A, positional embeddings were used to associate operations and materials through placeholders. These embeddings were summed to each set of features after being passed through a linear layer. Dropout was also applied at this point to avoid overdependence on a group of features. In this manner, the output was processed by the Transformer blocks. Another variation of the model was also tested using 4 heads, 2 blocks, and 1024 feed-forward dimensions. The two neural networks were trained for 100 epochs with a batch size of 128. Nonetheless, early stopping was also applied as indicated in the previous section.

6.2.4 Modelling

For sequence modelling, different types of autoencoders (AE) were explored. As discussed in previous chapters, AE provides a framework to reduce dimensionality into a continuous latent space. This representation in turn can be helpful to represent discrete data such as manufacturing sequences. There exist various classes of AE depending on latent space probability distribution, loss function, and training strategy employed for parameter learning. Considering their relevance in generative modelling, these include variational autoencoders (VAE), adversarial autoencoders (AAE), and adversarially regularised autoencoders (ARAE). The training of any of these types of AE enables the assignment of a probability distribution to latent space either due to an assumption or determined by another neural network. Both, AAE and ARAE, follow a similar training scheme to generative adversarial networks (GAN). As for VAE, several variants can be found whose differences are centred on loss function and architecture. Finally, all the models trained were conditioned to ensure the generated manufacturing routes could be customised according to predefined inputs.

As discussed previously, two ways of expressing sequences were also assessed. Firstly, apart from being used to study the predictive power of different inputs, representation A was also used to investigate preliminarily the effect of distinct types of layers, hyperparameters, and models on the performance metrics chosen. This information was used to refine the model and assess the advantages or disadvantages of these initial settings. In addition, it was used as a starting point to implement and compare to representation B. Vaucher et al employed this later way of expressing sequences to

develop a predictive model using another dataset [43]. Thus, their implementation was also adopted for our dataset and used as a reference. Subsequently, generative models were also applied with additional modifications. More details of all the models trained are provided below.

6.2.4.1 Architecture design using output representation A

6.2.4.1.1 Model exploration

To find out the best-performing model to generate sequences based on representation A, variations of conditional VAE, AAE and ARAE were tested. Initially, the representation that gave a better prediction accuracy in previous experiments was used as a condition. In the case of AAE, deterministic and stochastic encoders were also tried, while deterministic encoder was only applied for ARAE as in the original proposal [119]. On the other hand, while the architecture of the decoders was fixed for all VAE, AAE, and ARAE, the encoder was changed depending on whether the condition was injected in this module. Additional components were also included depending on the models. Of the three evaluated models, ARAE possessed the highest complexity since included a discriminator and a generator network [119], while AAE only included a discriminator. In conditional ARAE, a regressor or classifier network can also be included depending on the formulation as proposed by Zhao et al [119]. This component predicts the condition given the latent vector and its inclusion promotes the independence between latent space and target features. The layers and their established hyperparameters for all the distinct modules are listed below.

- **Decoder.** RNN: an embedding layer with 50 dimensions; a normalization layer and a feed-forward network with 128 units that processed condition; 3 stacked GRU layers with 256 units, GRU initial states corresponded to concatenated latent vector and processed conditions passed through a dense layer with 256 units; a layer normalization and an output dense layer the yielded the logits for each step. All probabilities were decoded into operation through greedy search.
- **Encoder.** RNN: an embedding layer with 50 dimensions; a normalization layer; 3 stacked bidirectional GRU layers with 128 units; a layer normalization, a global max pooling 1D layer and an output layer varied depending on whether the encoder was deterministic or stochastic. In the deterministic case, the output layer consisted of a single dense layer with 128 units, while the stochastic layer had 2

dense layers with 128 units each that generated a mean-vector and a log var-vector. In addition, condition information was only included in VAE's encoder and was processed using a residual network with dimension 128 for both intermediate and final layers. Then, the results were normalised to be concatenated with the embedded operations and finally processed by the recurrent layers.

- **Discriminator.** A layer normalization layer, 2 dense layers with 128 units and leaky ReLu as activation function, and 1 dense layer to generate logits. Variations were attempted with concatenated conditions and latent representation.
- **Generator.** Similar to the discriminator replacing the output layer with a dense layer with 128 units and changing the activation function by ReLu in the intermediate layers. This module only was used for ARAE.
- **Predictor.** Similar to the discriminator replacing the output layer with a dense layer with 128 units and changing the activation function by ReLu in the intermediate layers.

Regarding training, a description of the process for VAE and AAE can be found in Algorithm 6-1 and Algorithm 6-2. Specifically in VAE, as suggested by Bowman et al[123], KL divergence weight (beta) was increased in every epoch. This approach is also known as KL-annealing. For this work β was adjusted linearly starting from 0 until a maximum of 0.05 for 60 epochs. After that, the optimization process was allowed to continue until reaching 100 epochs. Conditional ARAE parameters, on the other hand, were learnt as suggested by Zhao et al [119]. Nonetheless, RMSProp was used as optimiser for all the modules varying the learning rate. For the autoencoder, encoder, and predictor update steps, a learning rate of 1e-3 was set. Meanwhile, 2e-6 and 1-e5 were applied for the discriminator and the generator, respectively.

Algorithm 6-1. VAE training scheme

$\mathbf{z}$: latent space, $\mathbf{x}$: manufacturing sequence, and $\mathbf{y}$: target or target + precursors, $\beta = 0$
Initialise encoder (q) and decoder (p) weights ϕ and ψ randomly.
For each epoch:
For each training step:
1. Calculate $\boldsymbol{\mu}_{\mathbf{z}}$, and $\log \boldsymbol{\sigma}_{\mathbf{z}}^{\mathbf{2}}$

$$\boldsymbol{\mu}_{\boldsymbol{z}}, \log \boldsymbol{\sigma}_{\boldsymbol{z}}^{\boldsymbol{2}} = q_{\phi}(\boldsymbol{x}, \boldsymbol{y})$$

2. Sample $\mathbf{z}$ using reparameterisation trick with $\boldsymbol{\epsilon} \sim N(\mathbf{0}, \mathbf{I})$.

$$\boldsymbol{z} = \boldsymbol{\mu}_{\boldsymbol{z}} + \boldsymbol{\sigma}_{\boldsymbol{z}} \odot \boldsymbol{\epsilon}$$

3. Reconstruct $\mathbf{x}$.

$$\tilde{\boldsymbol{x}} = p_\psi(\boldsymbol{z}, \boldsymbol{y})$$

4. Determine loss and update parameters ϕ and ψ using Adam with learning rate 1e-4.

$$\mathcal{L} = \mathbb{E}_{\mathbf{z}\sim q(\mathbf{z}|\mathbf{x},\mathbf{y})}[\log p(\mathbf{x}|\mathbf{y},\mathbf{z})] - \beta \cdot D_{KL}[q(\mathbf{z}|\mathbf{x},\mathbf{y})||p(\mathbf{z}|\mathbf{y})]$$

Update β.
If epoch < 60:

$$\beta = 0.05 * \frac{epoch}{60}$$

Else: $\beta = 0.05$

Algorithm 6-2. AAE training scheme. Taken and adapted from Arjovsky et al and Makhzani et al[70,120]. [a] Discriminator with and without condition (y) was also attempted.

$\mathbf{z}$: latent space, $\mathbf{x}$: manufacturing sequence, and $\mathbf{y}$: target or target + precursors, $\lambda = 10$, $n_{disc} = 5$.
Initialise encoder (q), decoder (p), and discriminator (f) weights ϕ, ψ, and ω randomly.
For each training step:

1. ***Train encoder and decoder:***
   Compute $\tilde{\mathbf{z}} = q_\phi(\boldsymbol{x})$ if q is deterministic, otherwise:

$$\boldsymbol{\mu_z}, \log \boldsymbol{\sigma_z^2} = q_\phi(\boldsymbol{x})$$

$$\tilde{\boldsymbol{z}} = \boldsymbol{\mu_z} + \boldsymbol{\sigma_z} \odot \boldsymbol{\epsilon}, \text{ with } \boldsymbol{\epsilon} \sim N(\mathbf{0}, \mathbf{I})$$

   Compute $\tilde{\mathbf{x}} = p_\psi(\tilde{\boldsymbol{z}}, \mathbf{y})$
   Calculate loss $\mathcal{L}_{AE} = \mathbb{E}_{\mathbf{z}\sim q(\mathbf{z}|\mathbf{x},\mathbf{y})}[\log p(\mathbf{x}|\mathbf{y},\mathbf{z})]$
   Update ϕ and ψ using RMSProp with learning rate 1e-3.
2. ***Train discriminator*** [a]
   Compute $\tilde{\mathbf{z}}$ as in (1) and draw a random sample $\mathrm{z}\sim N(\mathbf{0}, \mathbf{I})$.
   Compute $\hat{\mathbf{z}} = \boldsymbol{\epsilon}\mathbf{z} + (\mathbf{1} - \boldsymbol{\epsilon})\, \tilde{\mathbf{z}}$ with $\boldsymbol{\epsilon}\sim U(0,1)$.
   Calculate loss:

$$\mathcal{L}_D = \mathbb{E}_{\tilde{\mathbf{z}}\sim q(\mathrm{z}|\mathbf{x})}[f_w(\tilde{\mathbf{z}})] - \mathbb{E}_{\mathbf{z}\sim \mathbb{P}_z}[f_w(\mathbf{z})] + \lambda \cdot \mathbb{E}_{\hat{z}\sim \mathbb{P}_{\hat{z}}}\left[\left(\left|\left|\nabla_{\hat{z}} f_w(\hat{\mathbf{z}})\right|\right|_2 - 1\right)^2\right]$$

   Update ω using RMSProp with learning rate 2e-6.
   Repeat n_{disc} times.
3. ***Train encoder/generator***
   Compute $\tilde{\mathbf{z}}$ as in (1)
   Calculate loss $\mathcal{L}_E = -\mathbb{E}_{\tilde{\mathbf{z}}\sim q(\mathbf{z}|\mathbf{x})}[f_w(\tilde{\mathbf{z}})]$
   Update ϕ using RMSProp with learning rate 1e-5.

All the models were initially set to be updated for 100 epochs. However, early-stopping was applied to avoid overfitting and reduce training times. As a consequence, the learning process was stopped after 20 epochs without improvement, starting to count after a warmup period of 10 epochs. As a stopping metric, geometric mean among various performance indicators was used. These metrics included uniqueness, validity, condition permutation importance, and reconstruction accuracy. To measure reconstruction

accuracy, BLEU score was used, which is described in Chapter 3. The other metrics employed in preliminary tests are defined in Table 6-6. Performance metrics were monitored in every epoch.

Table 6-6. Initial metrics used for assessing sequence generation performance. [a.] Validity criteria for representation B was limited to referencing all the materials in the output sequence and the correct identification of target molecule.

Metric	Definition	Calculation
Uniqueness	Proportion of unique sequences in N generated samples	$\frac{\#\ Unique\ sequences}{\#\ Generated\ samples}$
Validity [a]	Proportion of sequences that meet rules established in code found in https://zenodo.org/records/13377654	$\frac{\#\ Valid\ sequences}{\#\ Generated\ samples}$
Diversity	Average of the type of operations found in the generated sequences based on 5 types predefined. For more details see https://zenodo.org/records/13377654	For each sequence: $\frac{\#\ Types\ of\ operation\ in\ sequence}{5}$
Novelty	Proportion of generated sequences found in the training set	$\frac{\#\ sequence\ in\ training\ set}{\#\ generated\ sequences}$
Condition Permutation Importance or features importance (FI)	Mean decrease in reconstruction accuracy (BLEU) after permutating the condition. Based on *grouped feature importance* proposed by Au Q et al[198].	$BLEU - \sum_{i=1}^{n} \frac{BLEU_{\text{permutated-cond}}}{n}$

6.2.4.1.2 Effect of network type

At this point, all the models assessed were based on RNN. To assess the effect of other types of layers, CVAE was trained, adjusting the encoder and decoder architectures to deploy transformers. Transformer blocks and embeddings were implemented as in Vaswani's work [112]. Token and positional embeddings to process operations were the same for both modules. The encoder consisted of two stacked transformer encoder blocks that processed information on target molecular representation. In parallel, two transformer decoder blocks received embedded sequences of operations. These blocks, first, executed self-attention over operations and then performed cross-attention between output and target descriptors previously operated. The result of this layer was then pooled by a pooling by multihead attention layer (PMA). Finally, additional linear layers were incorporated to obtain the latent representation. PMA was based on Lee et al proposal[199].

The decoder had a similar architecture to the encoder. Nonetheless, latent representation was concatenated with the embedded operations. Subsequently, this result was processed by a linear layer. The output was then operated by the transformer decoder blocks in an autoregressive manner. Thus, a causal mask was required to ensure this behaviour. Finally, an output layer estimated operations probability. In this manner, operations, at each timestep, were sampled using greedy search. The details about the hyperparameters employed can be found in Table 6-7. As optimiser, Adam was used with a learning rate that was increased linearly every timestep until reaching 1e-4 for 15 epochs.

Table 6-7. Encoder and decoder architecture using Transformers layers. [a.] If encoder is deterministic only one linear layer was used and μ_z and $\log \sigma_z^2$ were excluded. [b.] Transformer decoder block was only included in CVAE. The other model did not process information on materials.

Layer type	**Activation**	**Hyperparameters**
Input sequence		Max sequence length: 30
Embedding		Embedding dim: 128
Encoder [b]		
Transformer encoder block (conditional)	Relu	No. blocks: 2 Heads: 4 Key dim: 128 Feed forward: 512
Transformer decoder block	Relu	No. Blocks: 2 *Self-attention* Heads: 4 Key dim: 128 *Cross attention* Heads: 4 Key dim: 128 Feed forward: 512
PMA	Relu	Heads: 4 Key dim: 128
Dense (μ_z)[a]	Identity	Units: 128
Dense ($\log \sigma_z^2$)[a]	Identity	Units: 128
Decoder		
Dense	Relu	Units: 128
Transformer encoder block (conditional)	Relu	No. blocks: 2 Heads: 4 Key dim: 128 Feed forward: 512
Transformer decoder block	Relu	No. Blocks: 2 *Self-attention* Heads: 4 Key dim: 128 *Cross attention*

		Heads: 4 Key dim: 128 Feed forward: 512
Dense	Softmax	Units: No. Operations

6.2.4.1.3 Effect of including precursors

In addition, the effect of precursors' inclusion along with the target molecule was assessed. This was done by using the best-performing model between CVAE and AAE based on transformers. RNN models were avoided since the precursors' order affects the results, whereas transformer has been used in problems where permutation invariance is required [199]. Moreover, a modification in the architecture was also introduced where the weights of Transformers encoder blocks in both, the decoder and encoder, were shared. This was motivated by the increase in computational cost due to the change in input structure.

6.2.4.2 Architecture design using target and precursors.

6.2.4.2.1 Selection of best model

To include targets and precursors, the layers employed were only centred on transformers as in the section Prediction of actions associated with materials and conditions. In this way, a material encoder module was introduced. This module processed materials data to be fed into either the encoder or decoder. In the same manner, the encoder and decoder followed a similar architecture. Nonetheless, the effect of hyperparameters such as the number of transformers blocks and β parameter values during training were explored. These experiments used CVAE as a main framework. For KL-annealing, β was increased linearly until the maximum value for 20 epochs. The models were trained for 100 epochs. Hyperparameters for each module are defined in Table 6-8.

Table 6-8. Material encoder architecture using Transformers layers. [a.] If the encoder is deterministic, only one linear layer was used, whereby μ_z and $\log \sigma_z^2$ were excluded. [b.] Transformer decoder block was only included in CVAE. The other model did not process information on materials.

Layer type	**Activation**	**Hyperparameters**
Input sequence		Max sequence length: 81
Embeddings for operations		Embedding dim: 256
Material encoder		
Input materials		Max number of materials: 11

Layer normalisation		
Dense	Identity	Units: 256
Positional embeddings for materials		Embedding dim: 256
Dropout		Dropout rate: 0.1
Transformer encoder	Relu	No. blocks: 2 Heads: 8 Key dim: 256 Feed forward: 2048 Attention dropout: 0.1 Dropout rate: 0.1
Encoder		
Transformer encoder block (conditional)	Relu	No. blocks: 2 Heads: 8 Key dim: 256 Feed forward: 2048 Attention dropout: 0.1 Dropout rate: 0.1
Transformer decoder block	Relu	No. Blocks: 2 *Self-attention* Heads: 8 Key dim: 256 *Cross attention* Heads: 8 Key dim: 256 Feed forward: 2048 Attention dropout: 0.1 Dropout rate: 0.1
PMA	Relu	Heads: 8 Key dim: 256
Dense (μ_z)	Identity	Units: 128
Dense ($\log \sigma_z^2$)	Identity	Units: 128
Decoder		
Dense	Relu	Units: 128
Transformer encoder block (conditional)	Relu	No. blocks: 2 Heads: 8 Key dim: 256 Feed forward: 2048 Attention dropout: 0.1 Dropout rate: 0.1
Transformer decoder block	Relu	No. Blocks: 2 *Self-attention* Heads: 8 Key dim: 256 *Cross attention* Heads: 8 Key dim: 256

		Feed forward: 2048 Attention dropout: 0.1 Dropout rate: 0.1
Dense	Softmax	Units: No. Operations

In contrast to the previous section, not all generative approaches were employed. At this stage, exploration was focused on CVAE and AAE. The exclusion of ARAE was primarily due to the complexity that this implied to model target + precursors. More details are given in the section Results and discussion. Additionally, a variation of CVAE was also evaluated, which is known as Invariant CVAE. Thus, the layers employed in the encoder were varied depending on the model. While CVAE included both Transformer encoder and decoder blocks as conditional information, that was not the case for other models where only transformer encoder blocks were used. For instance, contrary to CVAE, the encoder of the invariant version did not include information on the condition. As a result, the loss function was changed as proposed by Moyer et al [200]. Finally, an AAE was trained with the best architecture for the encoder and decoder as indicated in Algorithm 6-2.

6.2.4.2.2 Effect of β control strategy, encoder and decoder architecture

Another modification introduced to VAE was regarding how β parameter was updated. To have greater control over the loss, dynamic calculation of β on every step was carried out as in Control VAE [201]. In this variant, Shao proposed the usage of a proportional-integral (PI) controller, that can help to prevent posterior collapse and promote disentanglement of latent variables. The training algorithm with these modifications can be seen in Algorithm 6-3. For PI control, the target KL was initially set at 35. Subsequently, with the best model, the effect of this parameter was assessed.

Algorithm 6-3. CVAE and invariant CVAE training scheme with PI controller. [a.] Noam scheduler was run as suggested for transformers by Vaswani [112]. [b.] In Invariant CVAE, λ was also updated at each time step, being two times the value of β. Loss function was calculated according to

$\mathbf{z}$: latent space, $\mathbf{x}$: manufacturing sequence, and $\mathbf{y}$: target + precursors, C: target D_{KL}, t: timestep, $k_p = 0.01$, $k_i = 0.0001$, $I(0) = 0.0$, $\beta(0) = 0.0$.
Initialise encoder (q) and decoder (p) weights ϕ and ψ randomly.
For each epoch:
For each training step:
1. Calculate $\boldsymbol{\mu}_\mathbf{z}$, and $\log \boldsymbol{\sigma}_\mathbf{z}^2$

$$\boldsymbol{\mu}_\mathbf{z}, \log \boldsymbol{\sigma}_\mathbf{z}^2 = q_\phi(\mathbf{x}, \mathbf{y})$$

If Invariant CVAE:

$$\boldsymbol{\mu}_{\mathbf{z}}, \log \boldsymbol{\sigma}_{\mathbf{z}}^{2} = q_{\phi}(\mathbf{x})$$

2. Sample $\mathbf{z}$ using reparameterisation trick with $\boldsymbol{\epsilon} \sim N(\mathbf{0}, \mathbf{I})$.

$$\mathbf{z} = \boldsymbol{\mu}_{\mathbf{z}} + \boldsymbol{\sigma}_{\mathbf{z}} \odot \boldsymbol{\epsilon}$$

3. Reconstruct $\mathbf{x}$.

$$\tilde{\mathbf{x}} = p_{\psi}(\mathbf{z}, \mathbf{y})$$

4. Determine loss and update parameters ϕ and ψ using Adam with learning rate set according a noam scheduler[a] with factor 0.2 and warmup steps 16000.

$$\mathcal{L} = \mathbb{E}_{\mathbf{z} \sim q(\mathbf{z}|\mathbf{x},\mathbf{y})}[\log p(\mathbf{x}|\mathbf{y},\mathbf{z})] - \beta \cdot D_{KL}[q(\mathbf{z}|\mathbf{x},\mathbf{y})||p(\mathbf{z}|\mathbf{y})]$$

If Invariant CVAE [b]:

$$\mathcal{L} = (1+\lambda) \cdot \mathbb{E}_{\mathbf{z} \sim q(\mathbf{z}|\mathbf{x},\mathbf{y})}[\log p(\mathbf{x}|\mathbf{y},\mathbf{z})] - \lambda \cdot D_{KL}[q(z|x)||q(z)] - \beta \cdot D_{KL}[q(\mathbf{z}|\mathbf{x})||p(\mathbf{z})]$$

5. Update β.

If PI controller and $t > 0$:

$$e(t) = C - D_{KL}(t)$$

$$P(t) = \frac{k_p}{1 + \exp e(t)}$$

If $0.0 < \beta(t-1) < 1.0$: $I(t) = I(t-1) - k_i \cdot e(t)$

$\beta(t) = I(t) + P(t)$

If $\beta(t) > 1.0$: $\beta(t) = 1.0$

If $\beta(t) < 0.0$: $\beta(t) = 0.0$

$\lambda = 2 \cdot \beta(t)$

Else: β updated as in Algorithm 6-1 with β_{max} equals 0.05 and 0.01 for CVAE, and Invariant CVAE, respectively. λ was set at 0.02, being updated as β.

In addition, some final modifications to the architecture were studied. The encoder architecture in the architecture shown in Table 6-7 was compared to others only based on transformer encoder block. The pooling method was also assessed in the encoder. PMA was compared to the approach used in the BERT language model [202]. In BERT, the pooling method is typically applied to the output representations of the [CLS] token, which is a special token added to the input sequence. This [CLS] token is used as a kind of aggregate representation of the entire input sequence for downstream tasks. Finally, two approaches were explored for injecting the latent space into the decoder. The first method consisted of passing the latent space through a linear layer, applying layer normalization, and then summing it with token embeddings. The second approach entailed concatenating the latent space with the condition, and this combined representation is processed by the transformer encoder.

6.2.5 External Validation

The two best approaches were chosen to proceed with external validation via evaluation from experts. The experts were academics with PhD in areas related to chemistry and experience in the design of processes for primary manufacturing. The four participants counted with years of experience in the design and development of pharmaceutical processes in different stages of the primary domain. Some of them centred on purification and others in synthesis. The survey consisted of comparing actual and generated procedures of manufacturing for a given set of materials. In total, 50 sets of materials and target were drawn from the test with their respective procedure. Later, two manufacturing sequence were generated using the two best models. In this manner a total of one hundred questions were obtained. Then, these were split into 4 surveys of 25 questions each. This is illustrated in Table 6-9 and the surveys can be found in https://pinkie-pint-zoologist-408j.onrender.com/ or a sample of the question can be seen Appendix D.

Table 6-9. Survey design

Survey	Model	Questions	Target
1	Invariant CVAE	25	Set 1
2	CVAE	25	
3	Invariant CVAE	25	Set 2
4	CVAE	25	

6.3 Results and discussion

6.3.1 Data description

The final dataset comprises 385,293 records describing synthesis and/or purification processes for 359,912 molecules. These include substances that can be used as synthesis intermediates or active pharmaceutical ingredients. The data was obtained from 23,342 patents. Table 6-10 shows the efficiency of the extraction of keywords for each type. The overall precision and recall were 96.2% and 92.7%, respectively. The overall accuracy measured as F1-Score was 94.3%, which is comparable to other works having as a reference Kononova et al dataset of inorganic reactions with an accuracy of 93.0% [166]. To highlight, the recovery of materials from text, without considering TARGET, was the lowest with around 88%, followed by conditions, with a close value, and finally,

operations being approximately 99%. Thus, the main limitation in the data collection methodology concerned materials. The following sections will present an overview of trends and relationships of the types of entities extracted.

Table 6-10. Extraction performance metrics per entity type for primary manufacturing dataset.

	Precision	Recall	F1-Score	Total entities
OPERATIONS	97.4%	99.0%	98.2%	838
CONDITIONS	95.9%	90.4%	93.1%	386
MATERIALS	95.1%	88.2%	91.5%	1138
TARGET	100.0%	100.0%	100.0%	100

6.3.1.1 Materials

Compared to the other entities, a greater portion of the materials could not be extracted, being missed nearly 1 in every 10. Although this result can be comparable to the relative loss of conditions information, this may have a greater impact in absolute terms as material-related terms are more common than conditions. In total, 4,439,136 materials and 1,497,611 conditions were identified. This also implied that some key reactants may not be considered in the modelling thereby affecting models' performance if these were to be used as inputs. Another limitation of the dataset, in terms of materials extraction, was the cross-reference. It was noticed that, in some documents, reagents were referenced using acronyms or a specific notation defined by the authors. This hindered their identification to calculate molecular descriptors or fingerprints. Regarding the obtained data, the extraction process prioritised precision since this ensured that the extracted data was faithful to the original content. This can be seen in the results as precision was higher than recall, reaching values greater than 95% for all the types of entities. In this manner, this facilitated that most reactants and solvents detected could be associated with an InChI identifier to be standardised.

Within the three types of materials defined, the most frequent were terms related to intermediates representing 37.9%. These include generic words used to refer to a material or combinations. Although they do not provide specific information on the composition, they can still be useful to understand how materials are being handled during a process. This was the main motivation to extract and maintain these words in the dataset for modelling, as the combination of intermediates and operations can provide

a more comprehensive representation of a process description. This will be discussed in more detail in the modelling section. Regarding solvents, these included pure or combinations of solvents. This type represented 31.0% of materials, being distributed in 1,411 different substances. On the other hand, 31.1% of materials were reactants where greater diversity was observed compared to solvents with 218,776 substances. On the other hand, it could also be observed that the most frequent substances presented some relationships with some operations.

Figure 6-2 shows the most common materials in each category. To highlight, aqueous solvents (water and brine) are widely used, but their application is particularly recurrent in operations such as extraction and washing. In the former, it seems to be more associated with processes involving partition. Conversely, organic solvents were predominant in operations more related to reaction or purification steps such as crystallisation or chromatography. As for other reagents, there is no remarkable dominance of any materials except for some sulphate salts, acids, and bases. Sulphate salts of sodium and magnesium have a specific utility as desiccant agents, mainly used after a reaction occurs. HCl and NaOH seem to be consistently used in either reaction or purification steps, there is not a particular trend in their use. Organic bases such as DIPEA and triethylamine were usually associated with synthesis steps. Finally, other inorganic salts such as sodium bicarbonate and ammonium chloride were predominant in washing and quenching operations, respectively. Although it cannot be generalised for all substances intervening in a process, the examples discussed previously suggest materials can be linked to some operations. Thus, this fact can have two implications in terms of manufacturing path generation. Firstly, if materials are known. they could be used to predict, at least partially, a sequence of operations. A second option is that, for certain operations, some materials could be preset, therefore giving a more detailed description of a process. Nonetheless, the practical applicability of the first option would require prior knowledge of all materials involved in the process of a target.

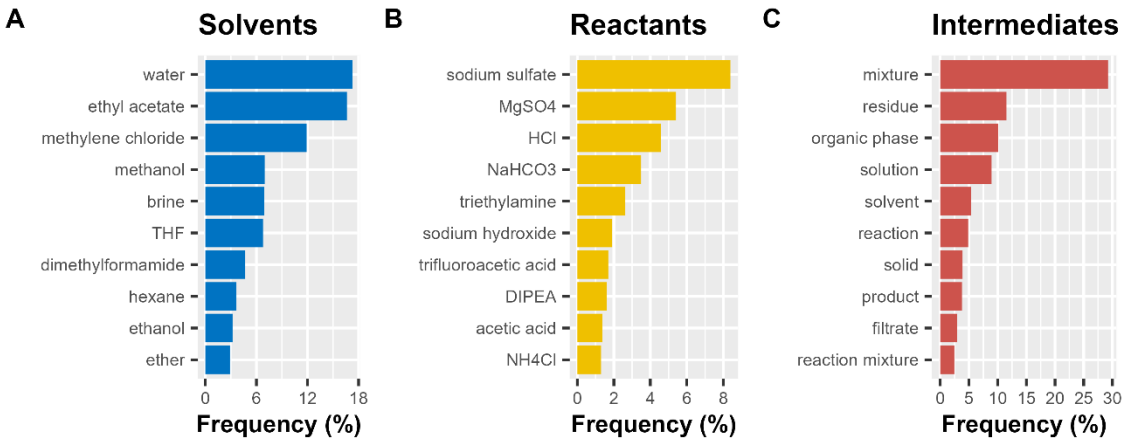


Figure 6-2 Top 10 most common substances in the different categories of materials. THF: tetrahydrofuran, DIPEA: N-ethyl-N-isopropylpropan-2-amine, Brine: typically, a saturated solution of sodium chloride in water.

6.3.1.2 Target Molecules

In terms of target molecules, although all of them were not exclusively APIs, they showed similar properties compared to typical drug substances. Specific properties such as partition coefficient, solubility, and molecular weight displayed a high degree of overlapping with approved drug substances by FDA, as illustrated in Figure 6-3. This figure contrasts the distribution of these properties between the training set and ZINC15 database [203]. In addition, the similarity in terms of drug-likeness could be observed through QED (quantitative estimated drug-likeness) score. This metric ponders 8 different properties typically used to describe the potential of drug candidates such as the mentioned molecular weight or LogP as well as other attributes such as the number of rotatable bonds, polar surface area, among others [204]. On average, QED scores were 0.5447 (SE = 0.2130) for the training set and 0.5462 (SE = 0.2192) for ZINC15 database molecules, respectively.

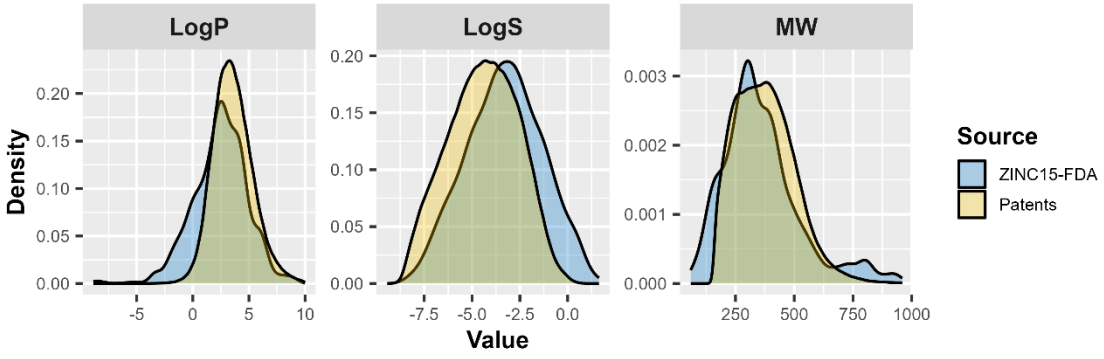


Figure 6-3. Comparison of properties distributions between target molecules in the training set (n = 359,912) and approved drugs by FDA recorded in ZINC15 database (n = 1,615) [203]. LogP, log-partition coefficient calculated using RDkit library [205]; LogS, log-molar solubility calculated using SolTranNet [206]; MW: molecular weight.

This alignment in chemical space confirms that the processes obtained are applied to typical substances found in the pharmaceutical industry. The generalisation of ML models depends on the dataset's scope, and the observed properties help to establish the type of compounds the models can provide reasonable outputs. For instance, 95% of target molecules in the training set have a molecular weight lower than 594.05 Da, thereby indicating that any model developed with these data may not generate plausible sequences of operations for compounds with high molecular weight, such as proteins.

6.3.1.3 Conditions

Conditions for approximately 34% of operations were extracted to provide a total of 1,497,611 data points. The conditions obtained corresponded to temperature, pH, time, pressure, type of atmosphere and number of times an operation is repeated, namely, repetitions. Although initially other types of data were obtained such as chromatographic conditions, these were not included in the final dataset. In the training set, temperature, time, and pressure were the most frequently reported, comprising 37.7%, 34.0%, and 16.0% of all data points, respectively. On the other hand, pH, the type of atmosphere and the number of repetitions summed to 12.4%.

In conditions such as type of atmosphere, temperature, and pressure, there was a clear predominance for certain values. This is shown in Figure 6-4 for temperature and type of atmosphere. In temperature, a trend towards processes executed under ambient temperature or values between 10 and 40°C was identified, representing 51.6%. Reasons such as costs and practical considerations may have contributed to this trend to a certain extent. The most common type of atmosphere corresponded to argon and nitrogen. These were seen to be used mainly in the early stages of a process to reduce environmental interferences or remove a particular component from the reaction mixture by purging. Regarding pressure, this parameter was mostly reported in qualitative terms such as reduced or vacuum pressure, with nearly 92.0% of this parameter records. These two values were usually associated with operations related to the concentration or removal of solvents. On the other hand, for the remaining 8.0%, the ranges of values presented a significant fluctuating varying from approximately vacuum to 4.1 MPa. Some

of these were associated with reactions such as hydrogenation, while others could be associated with operations such as filtration.

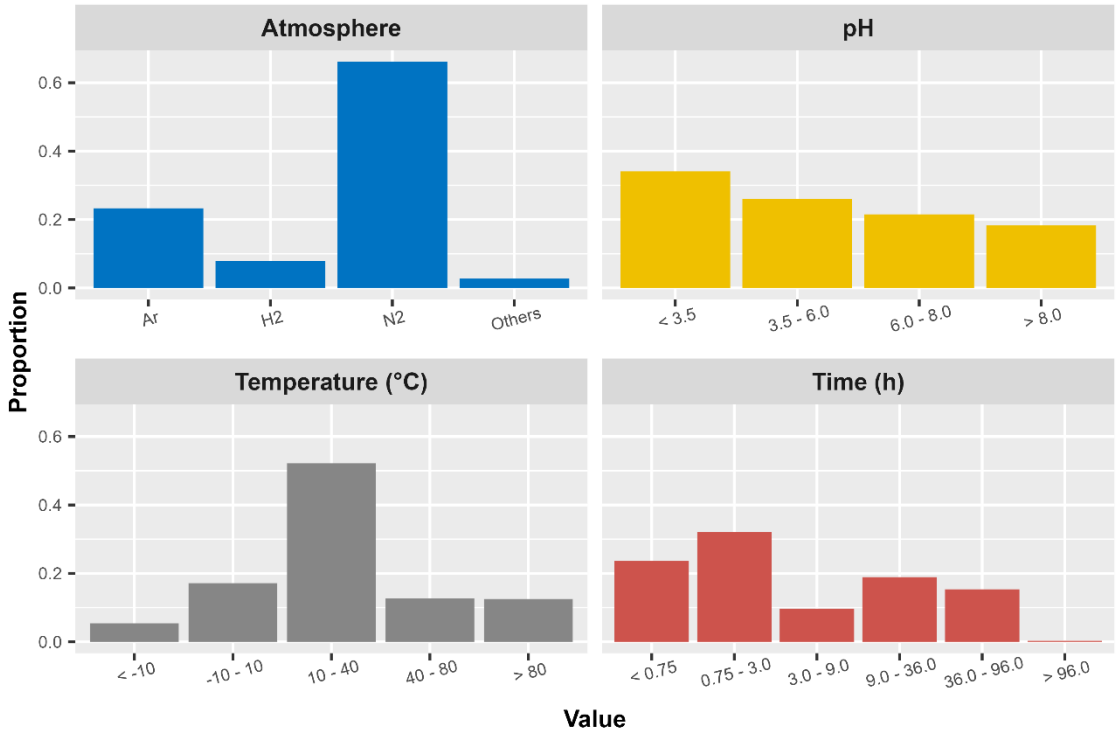


Figure 6-4 Distribution of conditions reported in the training set. Ar: Argon, H2: Hydrogen, N2: Nitrogen.

By contrast, pH and time exhibit a more even distribution based on the predefined categories as can be seen in Figure 6-4. It is also possible to observe that processes moderately to strongly acid seem to be more recurrent but without a remarkable difference concerning the other categories. In this regard, it is important to bear in mind that there were not many reports of pH in the operations obtained, representing 1.41% of conditions. As for the time, a bimodal behaviour was observed where the cutoff point was around 9 h. Thus, it is possible to see the majority of processes tend to last or proceed in intervals between 0.75 and 3 h. Conversely, it is also possible to find long-duration processes that mostly last between 9 and 36 hours. Finally, the number of repetitions was reported for operations such as washing and extraction where they are usually carried out between 2 and 4 times. This scenario was the most frequent for this type of condition representing 94.0% of records.

Thus far, some trends and patterns in the conditions obtained and their relationship with some operations have been shown. For instance, it was observed conditions such as repetitions or pressure tended to be mentioned along with certain operations. Similarly, although the relations between other conditions and operations were not evident, it could be seen that, for instance, temperature, type of atmosphere and time were found more often in the early stages of the process. In this manner, this indicates that these conditions can be more associated with reaction conditions. In turn, reaction conditions are defined by precursors and targets. In this manner, it should be possible to estimate conditions based on material properties in the same way as it is intended for operations. This idea has been exploited in other works, for instance, Karpovich et al trained a CVAE to predict temperature and time using both precursors and target information for inorganic reactions[190]. In the same manner, Vaucher et al developed several transformer-based architectures to predict both operations and conditions using reaction SMILES [43]. Having considered the literature and the available information, although this thesis scope originally intended to generate operations only, the ability to also generate conditions was explored in the assessed models. This is discussed in the next sections.

6.3.1.4 Operations

The training set contains 3,256,196 operations distributed in 20,192 terms. Nearly 90% of processes employed between 3 and 15 steps, with an average of 8.4 operations per process. After standardising the terminology, 60 standardised actions were identified. The top 10 most frequent operations and types are illustrated in Figure 6-5. The majority of operations corresponded to separation methods as shown in Figure 6-5B. This category included terms related to separation and purification steps such as “purify” (mostly related to chromatography), “dry”, “filter”, “crystallise”, etc. Next, it is possible to find terms related to material manipulation like “add” and mixing like “stir”. Other types of terms associated with a chemical transformation, such as “react” or “quench”, or indicating the adjustment of conditions such as “heat”, “cool”, and “adjust pH”, were less frequent. There was a small number of terms that did not fall into any of the previously mentioned classes. These consisted of operations indicating a physical transformation such as “condense”.

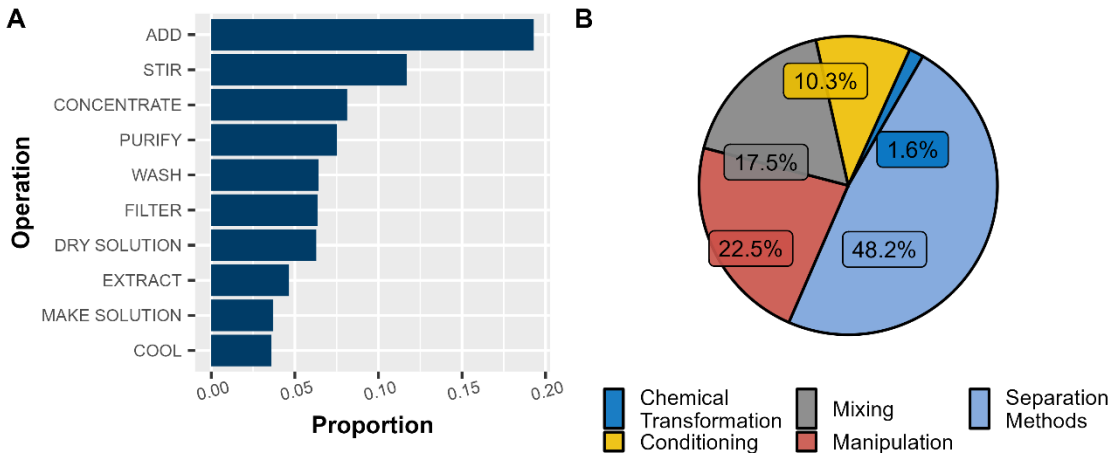


Figure 6-5 Most frequent operations by (A) standardised actions and (B) type of operation. Purify refers to chromatography. Dry Solution consists of the removal of water from a mixture usually through a desiccant agent. Make Solution refers to the combination of two or more materials to form a solution, a suspension, or other type of mixture.

Concerning the frequency of individual actions, most of the operations referred to terms more focused on the manipulation of materials or adjustment of process parameters. Terms such as ‘ADD’, ‘STIR’ or ‘MAKE SOLUTION’ were extensively used to describe processes. However, these actions can be considered generic since they can be used to describe either synthesis or purification steps, although they were more commonly found in the description of reaction steps. On the other hand, actions which describe more accurately an operation being performed were seen in purification steps. For instance, it was found chromatography (PURIFY) was the most recurrent followed by extraction with solvents (EXTRACT). The term concentration (CONCENTRATE) was also employed to indicate the removal of solvent, although the method applied to achieve this is not always specified. All these terms were characterised to be mainly found in the later stages of a process. An additional noteworthy aspect is the inclusion, within the training set, of processes exclusively focused on material purification. These processes differed from the others by starting with operations related to separation methods and the absence of precursors. To summarise, the frequency of certain operations and the order of appearance reveal discernible patterns indicating there is a logical sequence that operations tend to follow. A generative model is expected to recognise these patterns and reproduce the order based on inputs.

6.3.1.5 Other sources of bias

Given the training set covers patents from 1976 to 2021, changes in manufacturing procedures are anticipated, considering new technologies and trends. The first attribute of manufacturing sequences observed was the number of steps. Spearman correlation between the average sequence length and year was -0.25. Although this correlation is relatively weak, it suggests a subtle trend towards reducing the number of steps required to manufacture a product.

Another aspect analysed was the type of operations present. As discussed previously, the majority of operations were related to separation methods. In addition, these operations were more specific in terms of the type of technique employed compared to others, more generic. To explore change in this aspect, the evolution of the most common techniques over time was reviewed. Figure 6-6 exhibits a general increase in the patents utilising the different methods of separation, which include techniques like crystallisation or chromatography (PURIFY). However, this pattern is affected by the number of publications per year. Considering that the number of granted patents throughout the last decades has increased significantly, a rise in the mentions of the operations is expected. To account for this, the data were corrected by considering the evolution of the number of patents per year, as depicted in Figure 6-7. This correction revealed diverse patterns; for instance, mentions related to methods such as crystallisation and distillation have decreased, whereas solvent extraction and chromatography exhibited the opposite behaviour.

The relationships found suggest a temporal effect on manufacturing procedures. Given the number of patents in recent years is larger, models developed may favour the learning of the most recent procedures from the latest publications. Although this may be a desirable behaviour as it would generate sequences resembling more up-to-date approaches, these would also increase the likelihood of finding certain operations.

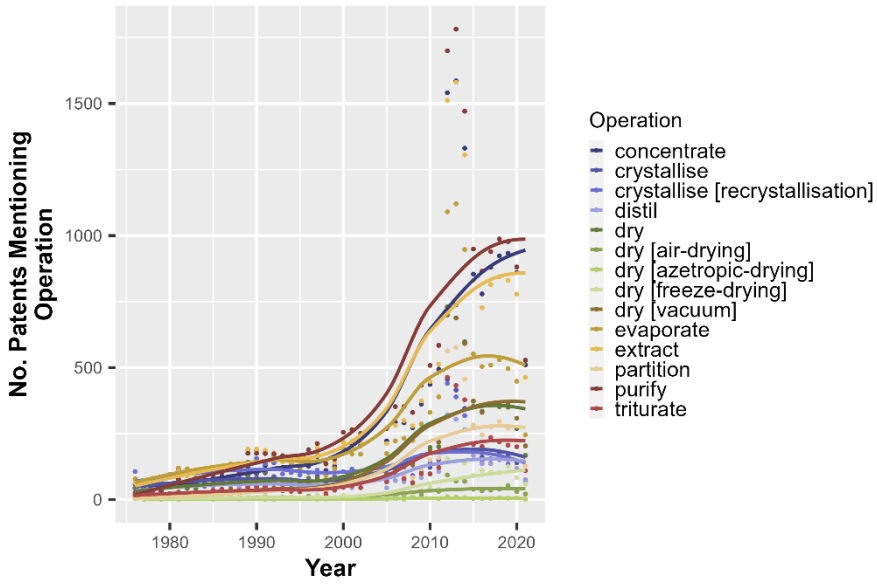


Figure 6-6 Evolution across time of the number of patents mentioning the most common separation methods. RECRYSTALLISATION refers to crystallisation from a solid material, while CRYSTALLISE starts from a solution. DRY indicates that the method of drying is unspecified.

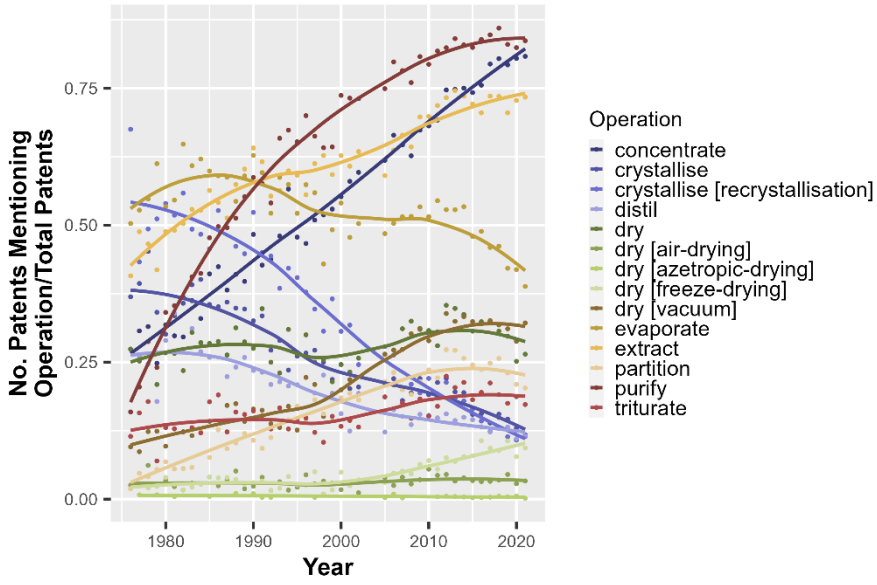


Figure 6-7. Evolution across time of the number of patents mentioning the most common separation methods with respect to the total number of patents.

Finally, the yield was examined as an additional factor to assess the scale of the processes under consideration. Figure 6-8 illustrates the distribution of the amount of

target molecule obtained by a process, revealing that 90% of the procedures yielded between 14 mg and 20 g of final product. This observation points out that the majority of procedures extracted from patents are conducted at the laboratory scale. While this focus on smaller scales provides valuable insights into early-stage developments, it may impose limitations on the applicability of the extracted knowledge to larger-scale industrial processes, which are more commonly found in marketed products. Nevertheless, despite the scale-related constraints, the dataset continues to offer valuable insights into the selection of operations for process design. Understanding the prevalent use of specific techniques and methodologies at the laboratory scale provides a foundation for future research aimed at closing the gap between laboratory-scale processes and their translation into larger-scale industrial applications.

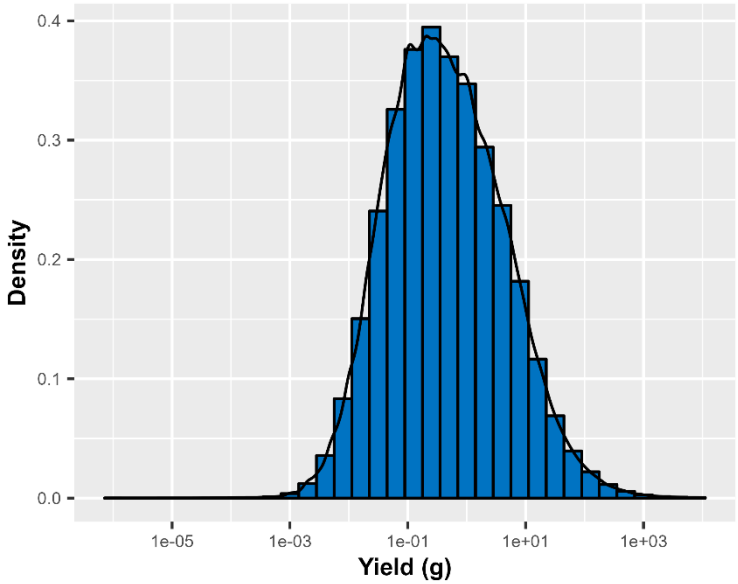


Figure 6-8. Distribution of the amount of target molecule obtained in the respective process in log-scale (n = 274,956)

6.3.2 Generative modelling

6.3.2.1 Selection of molecular representations

The comparison of sequence prediction accuracy, expressed as a BLEU score, for the different molecular representations of the target can be seen in Figure 6-9. All the representations provided an improvement in accuracy compared to control. The control consisted of a baseline for the worst case established by training a model with random

representations. These were generated by drawing a 128-dimensional random vector from a normal distribution for each target molecule. In the same way, each set of features presented differences in BLEU. In addition, the value of each representation was also affected by the network type. This was observed as the rank of the best-performing features varied depending on the architecture. In this manner, it is suggested that an architecture can exploit better certain features than the other. Nonetheless, when the network effect is assessed, there seems not to be a superior architecture.

On average, RNN and Transformers had a relatively close accuracy with values of 22.6% and 22.4%, respectively. On the other hand, by contrasting the best and worst descriptors in each model, the difference in Transformers was 1.89%, while 1.87% was obtained for RNN. Thus, the effect of the representation is greater than the architecture individually, as the network type has a difference of 0.02%. This indicates that the selection of a feature type can be more relevant than the network type. Nonetheless, in turn, the performance of a set of features is also dependent on the model. As a result, there is not a set of features that is consistently superior and the representation selection for a model may not be generalised for several applications.

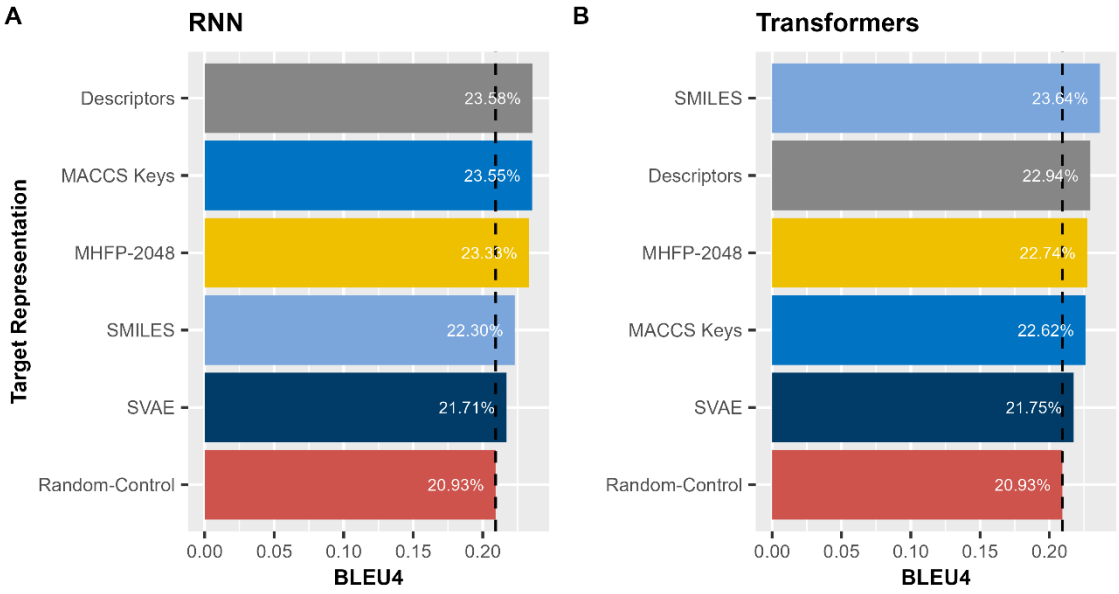


Figure 6-9 Comparison of sequence prediction accuracy in terms of BLEU scores resulting from distinct molecular representations of a target substance.

Regarding the performance of individual representation, SVAE had the lowest accuracy in both neural networks. an SVAE was pre-trained with SMILES to generate latent vectors

that can be used as molecular fingerprints. These types of representations have been used to predict properties such as partition coefficient and solubility in other works [207,208]. Pre-trained VAE for this purpose can be coupled with a predictor network during training, which can help to optimise latent space for property prediction. In our work, pretraining did not include such network, by which the VAE possibly centred on learning features for the generation of molecules. This might have led to poor performance compared to the other metrics. Performance using SVAE could be improved by further parameter fine-tuning for sequence prediction. This idea can also be supported by the results of SMILES representation.

SMILES performed well in transformers, in opposition to RNN. To make use of SMILES, an additional network was trained along with the other components of the models. This component consisted of 3 stacked GRUs and a bottleneck network composed of several 1D-CNN and an attention layer as suggested in the VAE encoder proposed by Dollar[127]. In this manner, the additional component corresponded to SVAE's encoder network. Thus, the fact that SVAE and the use of SMILES had mixed results indicates SMILES is more sensitive to architectural changes, by which a careful selection of hyperparameters is needed to ensure this representation produces good results for a given task. On the other hand, numerical features seemed to be more robust.

These representations gave variations in performance lower than 1% in both networks, lower than SMILES. Particularly, molecular descriptors were ranked 1 and 2 in accuracy for RNN and transformers, respectively. Additionally, in terms of implementation, it also offers another advantage compared to fingerprints such as MHFP-2048. While the molecular descriptors comprised 208 values, MHFP has 2048. As a result, the former may be less computationally expensive to train deep neural networks. This advantage can also be applied to MACC Keys, which has 166 bits. Thus, molecular descriptors were chosen for further experiments to predict sequences of operations.

Subsequently, the target and materials represented by the molecular descriptors were used together to train a transformers model for sequence generation. A large improvement was observed, resulting in a BLEU score of 31.2%. The change signified an increase of 10.3% with respect to control vs 2.01% obtained using only information on the target molecule. This points out that the target molecule, in isolation, cannot ensure an optimal performance. Instead, reactants can provide better guidance for operation

selection. This result can be expected since a target molecule can be synthetised and purified through various routes. On the other hand, when a synthesis route is established with information on materials and conditions, there are fewer freedom degrees for the selection. However, in practice, while a candidate molecule can be known, that is not necessarily the case for the synthesis route. This initially motivated to assess the performance using only the target. Nonetheless, given the importance of the inclusion of reactants in the process, it was decided to focus the modelling on conditioning manufacturing routes for target and reactants. As for the requirement of prior knowledge of reactants, reaction planners could be used to have an educated guess about feasible synthesis routes.

The inclusion of precursors also motivated the usage of a different output representation beyond the simple sequence of actions. Thus, based on Vaucher's work, representation B was evaluated. Output B relates each operation with its respective materials, providing a more informative and readable result. In addition, this representation also enables the incorporation of process conditions into the route. The authors also developed several models to predict operations based on transformers using as inputs reaction SMILES. To assess the effect of the architectural changes, Vaucher's transformers (model B) was adapted and retrained using our training set and varying the inputs. The other architecture (model A) differed mainly in the number of blocks and heads used, which was more similar to the transformers model used in the previous tests. MHFP and SMILES were excluded in these analyses. The former model could not be trained due to memory limitations. The latter, on the other hand, was trained but the network was unstable and further tuning was required by which these results were not included. Thus, the assessment was carried out with MACC Keys and molecular descriptors.

Table 6-11 reports the percentage of valid sequences and prediction accuracy in terms of BLEU and the percentages of predicted sequences with 100%, 90%, 75% and 50% matching with respect to reference. To remark, the validity criteria defined by Vaucher et al differ from the one used for this work. In their definition, the syntactical correctness is verified by checking whether sequences can be converted to actions without errors and ensuring that all the materials in the reaction are mentioned in the resulting sequence [43]. In our work, we verify that the second part of the criteria is met. Another key difference is the definition of actions. While Vaucher et al characterised manufacturing sequences in

24 actions, this work utilised 60. These facts mean that the models presented in this section cannot be directly compared to Vaucher's. Thus, the results of their model were recorded in Table 6-11 with informative purposes to provide intuition about the magnitude of these metrics.

Table 6-11. Performance metrics for prediction of actions associated with materials and conditions (Output B, n = 1000). [a.] Results from reference model developed by Vaucher et al[43]. Architecture B corresponded to an adaptation of Vaucher's Transformers trained on our dataset with different representations. Architecture A is a modified version of B with half transformer blocks and heads. [b] It refers to the quantiles of accuracies obtained by sequence matching evaluated using Levenshtein similarity. For instance, 3.6% of the sequences had 100% matching between actual and predicted in Vaucher's Transformer. [c] it only considers architecture A and B.

Architecture	Representation	Validity	BLEU	Accuracy [b]			
				100%	90%	75%	50%
Transformer [a]	Reaction SMILES	99.7%	54.7%	3.6%	10.1%	24.7%	68.7%
B	Molecular Descriptors	78.8%	52.5%	2.2%	5.0%	18.3%	72.4%
B	MACC Keys	**80.0%**	52.4%	**2.4%**	**5.1%**	**18.6%**	**73.1%**
A	Molecular Descriptors	78.4%	52.0%	1.2%	2.8%	14.0%	69.5%
A	MACC Keys	79.8%	**52.8%**	1.1%	2.8%	14.6%	71.0%
	Mean [c]	79.3%	52.4%	1.7%	3.9%	16.4%	71.5%
	CV [c]	1.0%	0.6%	38.9%	33.1%	14.7%	2.2%

Prediction accuracy was more dependent on the architecture. In this case, accuracy was measured using several metrics since BLEU seems not to be sufficient to conclude about the difference between architectures or representations. This could be evidenced as BLEU varied less than the other accuracy metrics. In this regard, 100% Matching can reflect more clearly the difference among the assessed factors. Specifically speaking on the architecture effect, B provided a greater reconstruction, being, nearly twice that achieved by A on average. Alternatively, BLEU scores for architecture A and B were 52.4% and 52.5%, respectively. In this manner, although BLEU would reach a similar conclusion to using the other metrics, the margin was smaller. Therefore, additional metrics strengthen the conclusion about the architecture effect. Similarly, the differences in models were also reflected in the validity as model B was also superior to A with 79.4% and 79.1%. Nonetheless, the variability in this metric seems to be mainly explained by the representation.

Overall, MACC Keys tended to generate a higher proportion of valid sequences. The effect magnitude was 1.3%, favouring MACC Keys over molecular descriptors. Regarding

prediction accuracy, MACC Keys was superior in model B for all the metrics, while the same applied to only 3 out of 5 metrics. In this manner, this result also suggests that input feature performance varies with the model architecture, being aligned with the initial results about representation effect. It also can be noticed that MACC Keys, in this case, were better than molecular descriptors. This contrasts with previous results where target information was the only input, and the output comprised actions. This also confirms that performance changes with output expression. Consequently, this would make the selection of optimal descriptors case based.

To summarise, the effect of different types of inputs and architectures on sequence prediction expressed in two ways was studied. Regarding inputs, information on targets and precursors provided a better performance for sequence prediction. On the other hand, the way these materials are represented for modelling can affect either prediction validity or accuracy. In addition, the results for a chosen representation can vary with model settings indicating that, from a performance perspective, the optimal features should be determined case by case. Nonetheless, the effect seems not to be substantial as several types of features had similar performances. In this manner, the selection can also be guided by other criteria such as interpretability or computational cost. Thus, the following representations were selected for generative modelling: molecular descriptors for generating action only and MACC Keys for action associated with materials and conditions.

As expected, the architecture also affects performance. Some aspects to highlight include RNN was slightly better than Transformers according to initial tests, although none was superior. In this manner, RNN and transformers were further compared for different generative models. Having said this, performance cannot be the only aspect to consider at the moment to select the best model. For instance, Transformers were chosen in cases where a set of materials are used as inputs. The use of transformers over RNN was driven by the ability of this architecture to handle data that is not necessarily dependent on the order, i.e., permutation invariance. Finally, other aspects of architecture such as network depth seem to be relevant in modelling. This was shown in the comparison of models A and B previously discussed. The effect of these factors is analysed for generative modelling in the next sections.

6.3.2.2 Models for sequences of actions

In the first instance, it was shown that the different architectures of generative models have the capability of generating sequences of operations. Examples are illustrated in Table 6-12. However, the faithfulness of the sequences and the degree to which target information guides the generation vary depending on the model. As for the similarity between generated and real chains of operations, by inspection, it was seen that some approaches tended to exhibit particular patterns. For instance, certain operations could be repeated for several consecutive steps contrary to real procedures. Likewise, operations typically expected at early stages of a process, such as addition or reaction, sometimes appeared in the end of the sequence. In consequence, the presence of these patterns may indicate that the explored models did not capture efficiently certain aspects such as the logical order of the operations.

Table 6-12. Examples of randomly generated sequences by different models.

Model	Generated Sequence
RNN AAE + stochastic encoder + conditioned discriminator	add → add → stir → add → stir → adjust pH → extract → wash → dry solution → dry solid → dry solution → purify → add → dry solution → filter → purify
	stir → add → stir → stand → add → stir
RNN ARAE	add → add → stir → make solution → heat → add → filter → combine → stir → add → filter → wash → dry solid [air-drying] → make solution → filter → wash → filter → concentrate → purify
	add → stir → heat → filter → adsorb → purify
RNN CVAE	treat → add → stir → cool → add → stir → quench → stir → separate → adjust pH [decrease] → extract → wash → dry solution → evaporate → separate → purify
	add → stir → add → stir → add → wash → dry solution → concentrate → purify → purify
Trans CVAE	make solution → add → irradiate → concentrate → dry solid [vacuum]
	add → heat → concentrate → add → wash → add → filter → wash → dry solution → filter → concentrate → concentrate

In addition, certain approaches were more prone to ignore the target. This is displayed in Figure 6-10B. Here, the effect of target descriptors on the reconstruction was measured by permutating features, while fixing the latent representation. The majority of models exhibited subtle or no changes in their accuracy. Consequently, these cases would point

out that the generated operations are not associated with the specific target. On the other hand, under the same criteria, CVAE models seemed to exploit this information to a greater extent. Interestingly, another aspect to highlight was the fact that RNN-based architectures were inferior in the usage of targets compared to Transformers.

Resuming the slight superiority of CVAE in terms of target association with sequences, a possible explanation could be related to the regularisation applied by KL divergence in the loss function. This particularity has been exploited by different variants of VAE to disentangle latent space in other domains [209,210]. This characteristic promotes the independence between latent features and makes them as informative as possible [209]. In this manner, the model learns latent features without redundant information. Since conditional data is provided, the model would then try to separate conditional data, that corresponds to the target, from other sources of variation, avoiding overlapping between information. As a result, the variation explained by target descriptors is not incorporated into the latent representation, allowing the decoder to utilise the target more explicitly.

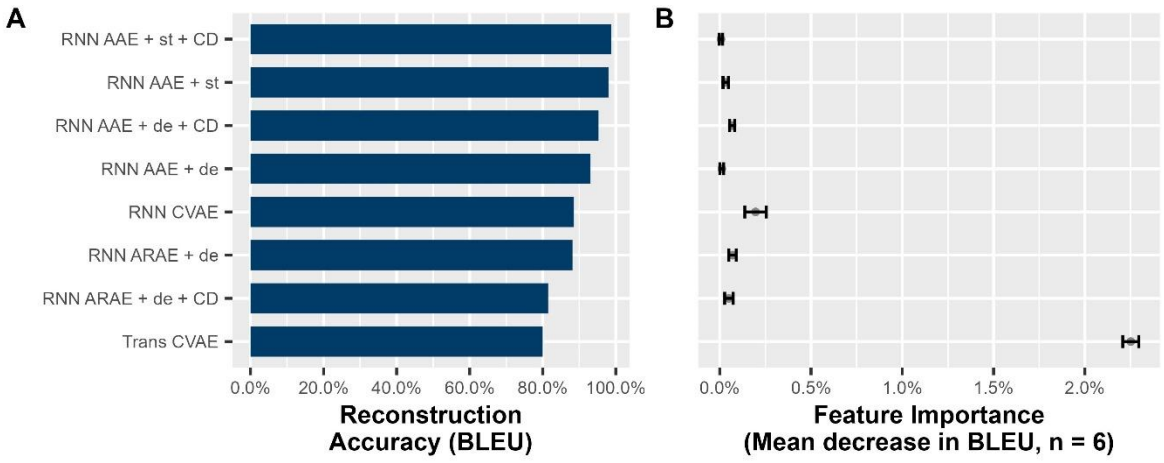


Figure 6-10 Reconstruction Accuracy (A) and condition importance (B) for different generative models for output A generation using target molecular descriptors. CD: condition included in discriminator; st: stochastic encoder; de: deterministic encoder.

Unlike CVAE, models based on adversarial training exhibit superior performance in reconstruction, as shown in Figure 6-10A. This result aligns with findings in fields such as text generation, where AAE or ARAE are generally better at recreating examples [211]. Additionally, the observations are aligned with the trade-off between reconstruction and disentanglement established for VAE. This was formalised through Higgins et al with β-

VAE, where the weight of KL-divergence (β) in VAE's objective function controls the balance between the two properties mentioned previously [125]. Usually, higher β values result in greater degree of disentanglement. However, it is worth noting that excessively high values may lead VAE to collapse in sequence modelling, whereby β must be selected thoroughly [126]. Thus, the obtained results may confirm that the trends observed in other domains also apply to the generation of manufacturing sequences.

Thus far, none of the models demonstrated superior performance in all assessed aspects. The ability to incorporate conditional information is critical to ensure that generated sequences are customised for specific targets. Simultaneously, achieving a good reconstruction is necessary to generate realistic manufacturing sequences. While AAE and ARAE were the best for the latter aspect, the performance in incorporating conditional information was lacking. Additionally, as typical GAN-based approaches, these models face challenges during the training process, including training stability issues due to the loss function [63]. In this manner, strategies like monitoring combined with early stopping are required to identify the optimal parameters and prevent collapse. The inclusion of additional components, such as discriminator or generator networks, adds complexity, making models more sensitive to hyperparameters. Under these circumstances, despite CVAE's lower reconstruction, it is still sufficient to ensure good-quality sequences. Furthermore, CVAE provides a more robust framework that allows better customisation. Therefore, for manufacturing sequence generation, CVAE seems to be a more convenient option.

6.3.2.3 Effect of inputs and pooling on the generation of actions

Having defined CVAE as a reference framework, the effects of input and network architecture were studied. The performance metrics of these variations are reported in Table 6-13. The first aspect to remark is that the inclusion of reactants as inputs did not enhance reconstruction accuracy, but it had a greater importance compared to target only. This is consistent with prediction results discussed previously, where materials improved predictability. In addition, this behaviour also allowed to confirm that reactants possess a greater impact in the selection of operations compared to the target alone. As discussed in previous sections, while there might exist multiple paths to manufacture a target molecule, the options can be bounded when reactants are included. These are required to be handled in particular manners. Additionally, reactants can be associated

with the profile of impurities after reaction, which in turn may hint which techniques are feasible or more appropriate for target isolation and purification. Thus, the improvement in feature importance might indicate that the model is learning that type of association between operations and combinations of materials. In consequence, the inclusion of reactants provides additional information which bounds the selection of operations, helping to a more customised sequence generation.

Table 6-13. Performance metrics for variations in CVAE architecture. [1.] Percentage of unique sequences obtained from new generated sequences; [2.] Percentage of generated sequence found in the training set. *Pooling by Multihead Attention, implemented as proposed by Lee et al [199].

Network type	Input	Pooling Method	BLEU	FI	Uniqueness[1]	Novelty[2]
RNN	Target	NA	88.6%	0.2%	95.0%	3.2%
Transformers	Target	Max Pooling	**92.3%**	0.0%	89.4%	4.1%
Transformers	Target	PMA*	80.0%	2.3%	98.1%	1.5%
Transformers	Target + Reactants	Max Pooling	91.5%	17.1%	96.6%	2.9%
Transformers	Target + Reactants	PMA	80.1%	**17.2%**	**99.6%**	**0.7%**

Regarding network architecture, there was a special emphasis on the pooling mechanism. This component is part of the encoder, and its role is to aggregate unit operations in a single n-dimensional vector. During the encoding, unit operations are embedded so that each has a numerical representation. These vectors are further operated by an attention mechanism along with conditional information. Finally, this results in a series of vectors representing a sequence. Nonetheless, to project sequence representation onto a latent space, it is necessary to reduce the dimensionality across operations, whereby pooling is crucial. There were two mechanisms tested which included max pooling and pooling by multihead attention (PMA). Preliminary tests indicated that this had a strong impact on performance, and it was verified in the results illustrated above.

As observed, each mechanism captures and condenses information differently, thereby affecting the model's ability to generate diverse and meaningful sequences of operations. To remark, the trade-off between reconstruction and feature importance appears to be once more present. Max pooling may be ignoring conditional information to some extent, being more explicit when target descriptors were the only input. On the other hand,

although PMA utilised more conditions in both cases, the reconstruction was penalised. Another interesting aspect to mention involves uniqueness and novelty. These metrics can act as indicators of the diversity of sequences generated by a model and the risk of overfitting. Models employing max pooling generated less diverse sequences. This may suggest that the models are not effectively learning a latent structure of the sequences and their association with materials, instead max pooling promotes that models memorise certain patterns.

Concluding this part, the inputs and pooling mechanism influence the performance of CVAE for manufacturing process generation. Inputs, including reactants and target descriptors, provide context for generating meaningful operation sequences. It is worth mentioning that, in practice, it is challenging to have information on reactants whereby this limitation requires the usage of complementary tools to find appropriate candidate materials for target manufacturing. On the other hand, the choice between max pooling and PMA contributes to balance reconstruction accuracy and the incorporation of conditional information. Although PMA offered a lower reconstruction, it leveraged material data to guide selection and produce more diverse sequences.

6.3.2.4 Models for Generation of actions, conditions, and association of actions with materials

The previous experiments have demonstrated the capacity of generative models to produce sequences of operations and link an output sequence with a set of materials, to a certain extent. Furthermore, both, target, and reactants, have proven to be more beneficial for sequence customisation than target alone. Finally, among the evaluated frameworks, CVAE emerged as the most convenient for this task. Subsequently, certain architectural characteristics were defined, with transformers and PMA networks being more promising in generating more diverse and input-conditioned manufacturing sequences. However, at this point, the sequences generated consisted solely of operations without expressing their relationship with materials. While this information can help provide initial insights into what actions and techniques execute for target fabrication, the generation of additional information - such as conditions or association between materials and operations - can yield a more comprehensive output. For instance, the final sequence could also provide information on addition order.

Accordingly, CVAE was then adapted to generate chains of operations represented as output B. Previously, it was hypothesised that CVAE's loss function favours latent feature disentanglement, which in turn supports conditional generation. In this way, Invariant CVAE, a variant of CVAE that encourages disentangled latent features, was also examined. Figure 6-11 displays the evolution of performance metrics such as validity, reconstruction, and feature importance throughout network training for both CVAE and its invariant counterpart. Overall, CVAE provided the best results in terms of all the metrics, except for validity, compared to the invariant version, regardless of the number of layers. In general, a high value for this hyperparameter did not exhibit a substantial benefit in either of the variants. The most notorious effect could be observed in CVAE for feature importance, where the difference between the deepest and shallowest network was approximately 5%. Since the model depth did not exhibit a strong effect, the number of layers was set at 2 for further experiments to study other modifications in the architecture.

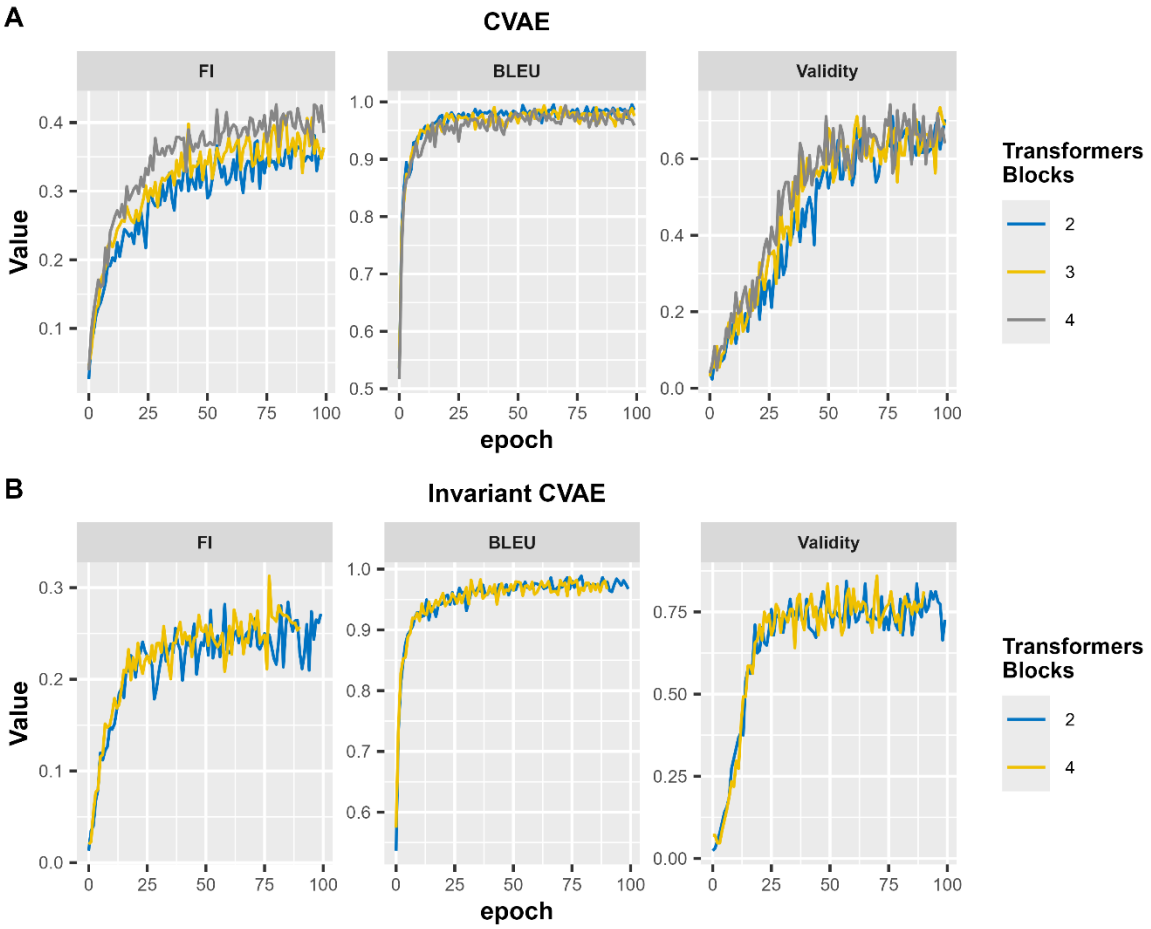


Figure 6-11 Learning curves of feature importance (FI), BLEU and generated sequence validity for A) CVAE ($\beta = 0.05$) and B) Invariant CVAE ($\beta = 0.01$, $\lambda = 0.02$) trained with different number of layers.

Thus, additional modifications were assessed using a simple CVAE model as a reference. Firstly, different network arrangements for the encoder were tested. One was based on transformers encoder which mimics language model BERT (bidirectional encoder representation from transformers), and the other used Vaswani's original proposal [112,202]. Aligned with previous results, a different approach to represent the entire sequence in a single vector was attempted based on the BERT model. This simply involved taking the [START] token representation to be used for mean and log-var estimation. Finally, two methods of processing the latent vector by the decoder were also evaluated. The results of these changes are summarised in Table 6-14. On average, Vaswani's transformers and PMA favoured all the metrics, showing values above the average for most cases where these networks were employed. Conversely, the injection method had mixed results. While concatenating the latent representation with the

condition resulted in a higher proportion of valid sequences, the reconstruction and condition importance were negatively affected. However, when the latent was injected using embeddings with the best encoder and pooling method, the validity was still higher than the average, offering the greatest values for reconstruction and feature importance. To remark, although these settings worked best for simple CVAE, the invariant CVAE does not require conditioning for encoding, thereby BERT-based architecture was chosen to be more appropriate in that case.

Table 6-14 Performance metrics for architecture modification in CVAE model in the encoder, pooling method, and latent space injection trained using PI for β dynamic adjustment with KL target = 35.

Network type	Pooling Method	Injection	Validity	BLEU	FI	Accuracy 100%
BERT	[START]	embeddings	71.1%	94.3%	41.2%	72.4%
BERT	[START]	condition	**74.6%**	94.1%	43.0%	76.3%
BERT	PMA	embeddings	73.2%	94.6%	43.7%	77.7%
BERT	PMA	condition	74.0%	95.1%	43.1%	77.7%
Vaswani	[START]	embeddings	73.8%	95.2%	44.9%	79.8%
Vaswani	[START]	condition	74.4%	95.0%	43.1%	77.0%
Vaswani	PMA	embeddings	74.3%	**95.3%**	**45.3%**	**80.6%**
Vaswani	PMA	condition	74.8%	95.1%	42.9%	78.1%
Mean			73.8%	94.8%	43.4%	77.5%
Std			1.2%	0.4%	1.3%	2.5%

Finally, a model based on the best architecture obtained at this point was retrained using AAE framework. In previous results, AAE did not perform better than CVAE using RNN and generating sequences of actions without additional information. In the end, under different settings, AAE still performed worse than VAE models for all the metrics. BLEU, FI, and validity for the test set were 89.5%, 17.3%, and 45.4%, respectively. Alternatively, there were several attempts at training ARAE using the current architecture, but they were not successful. It was noticed that the model's loss function had an unstable behaviour, which led to a rapid collapse. However, this does not discard that AAE or ARAE can be employed for the current task but suggests that the selection of an appropriate architecture is more critical. In this manner, these results allow to confirm that CVAE was the best generative model for manufacturing sequence generation.

6.3.2.5 *Effect of KL-divergence weight (β)*

Preliminary experiments indicated that KL-divergence weight (β) in the loss function was a key parameter for CVAE training. High values of this parameter yielded posterior collapse, resulting in sequences with visible errors in operations order or low diversity, as well as a lack of usage of information on materials. In the same manner, when the value was low, the model converged into a simple autoencoder, exhibiting similar issues related to diversity and conditioning. This behaviour was consistent with literature for domains like text generation and drug discovery employing VAE [125,201]. Considering this, some authors have proposed frameworks such as Dynamic VAE, which implements proportional-integral (PI) control to dynamically tune β during training [212]. This implementation, as found by Huajie et al., has achieved a good balance between disentanglement and reconstruction in computer vision [212]. Thus, this approach was implemented and compared to traditional KL-annealing.

Figure 6-12 illustrates the effect of the β-controller on different metrics during training, and the overall performance on the test set is reported in Table 6-15. While this parameter remained stable after reaching a maximum with KL-annealing, PI made β decrease slowly. This effect was also reflected on KL-loss, where the divergence had the opposite behaviour, revolving around 35 throughout all the training steps. Thus, it can be seen how PI controller offered a more explicit control over the loss function, as KL divergence value becomes a hyperparameter. As for the other metrics, the consequence of selecting low values was displayed. Here, the proportion of valid sequences and feature importance increased progressively as β did. Interestingly, the reconstruction showed the inverse trend. The patterns exhibited for low $\beta = 0.05$ and $\beta = 0.11$ results from the model ignoring KL-regulation and focusing on reconstruction. Consequently, in these cases, the VAE would lose the probabilistic and generative properties.

Regarding the models with PI controller and $\beta = 0.22$, these two provided similar learning curves for reconstruction, feature importance, and validity. It is worth noting that the maximum value for both models was close. This shows that both strategies can provide similar results as long as β is appropriate. Nonetheless, for hyperparameter exploration, PI controller guarantees there is no posterior collapse. Alternatively, KL-annealing requires the adjustment of values such as rate, schedule function, and cycles [213]. As a result, PI can be more useful to explore β given a target KL since these hyperparameters

are not required. However, the determination of this target value is perhaps the main limitation of using this approach. To tackle this, some researchers have suggested that target KL divergence can be related to the training set size (N) as defined in the following expression: $target\ KL \geq \log_2 N$ [212]. In our case, this value was approximately 18, while, empirically, a target KL of 35 gave the best results. In this way, although it might not be accurate, it can still provide a good initial guess for testing. Therefore, while PI might not always be superior to KL-annealing in terms of performance, the reduced number of hyperparameters required can be more advantageous for model design and optimization processes, allowing for more efficient experimentation and potentially faster convergence.

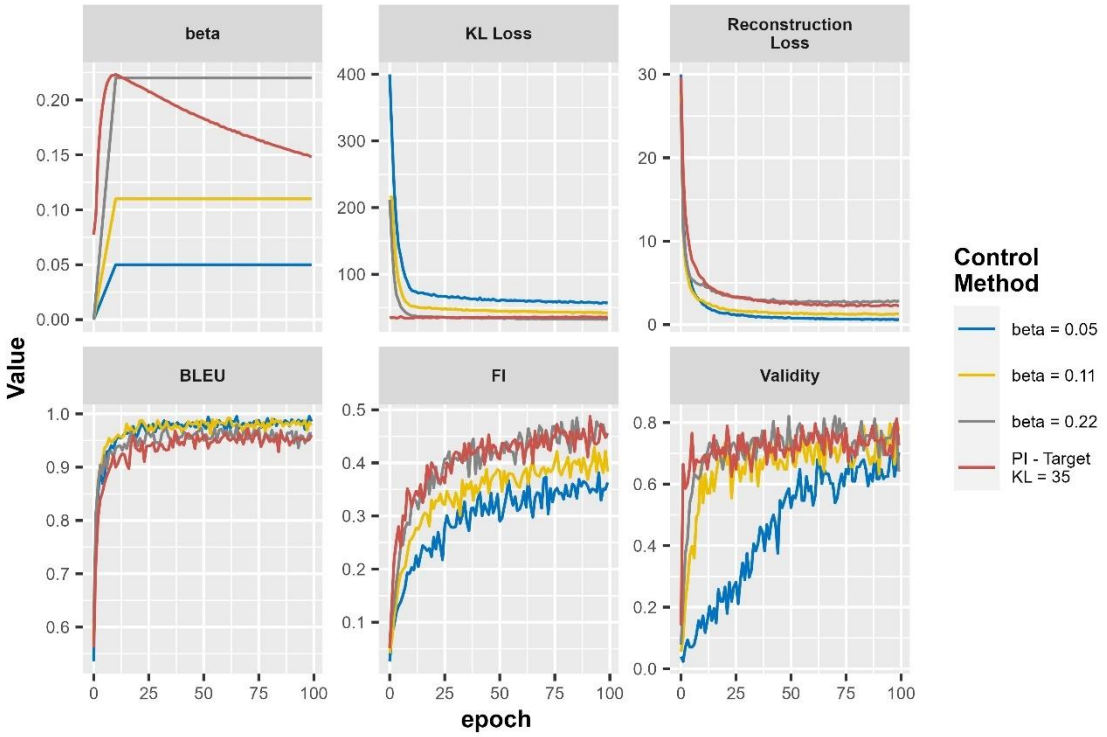


Figure 6-12. Effect of β-control method on CVAE performance during the training process calculated on development set.

Table 6-15. Performance metrics for CVAE trained under different β-control strategies.

β-controller	Validity	BLEU	FI	Accuracy 100%	Geometric Mean
PI - KL target = 35	74.3%	95.3%	45.3%	80.6%	71.3%
KL annealing - $\beta = 0.11$	73.4%	**98.1%**	40.4%	**89.9%**	71.5%
KL annealing - $\beta = 0.22$	**75.2%**	95.6%	**46.1%**	79.2%	**71.6%**

On the other hand, in the invariant model, λ was also found to be important. When this was not annealed in the same way as β, the model was more prone to collapse. In this manner, this hyperparameter also required a careful assessment to ensure the best possible balance between all the performance metrics. Thus, Invariant CVAE was retrained using PI varying KL target to explore its effect on performance. The results can be seen in Table 6-16. To highlight, PI approach with KL-target of 35 yielded the best results for Invariant and simple CVAE, providing a better balance between the metrics. By comparing both β-controllers, it is evident KL-annealing prioritised reconstruction optimisation over the validity and feature importance contrasting to PI. Nevertheless, this could have occurred due to β and λ values. Regarding the effect of KL target, a reduction in this hyperparameter reduces reconstruction accuracy. Simultaneously, it increases validity and feature importance, having a greater effect on the latter metric. A substantial effect was not observed when the relation between β and λ was changed from 1:1 to 2:1.

Table 6-16. Effect of KL target on performance metric for Invariant CVAE.

KL target	Lambda	Validity	BLEU	FI	100% Accuracy	Geometric Mean
25	$\lambda = \beta$	77.7%	90.7%	**33.6%**	55.2%	60.3%
30	$\lambda = \beta$	77.3%	93.3%	32.8%	65.0%	62.6%
30	$\lambda = 2\beta$	**77.9%**	93.0%	32.4%	64.4%	62.4%
35	$\lambda = \beta$	**77.9%**	95.6%	31.1%	**75.8%**	**64.7%**
KL annealing	$\beta = 0.01$, $\lambda = 0.02$	76.2%	**97.5%**	24.5%	85.7%	62.8%

To sum up, the findings highlight the critical role of KL-divergence weight (β) and λ, in training CVAEs effectively. Through empirical analysis and comparison between different β controllers, it was shown the importance of KL divergence in the loss function to achieve optimal model performance, balancing reconstruction accuracy, validity, and feature

importance. The benefits and drawbacks of KL annealing and PI have been discussed. While no major differences in performance were seen when optimal values were used, PI seems to offer greater versatility independently of the VAE variant. This approach primarily requires a single hyperparameter, which has a relation to the training set size, useful to obtain a starting point for experimentation. However, even after assessing KL weights, CVAE still outperforms Invariant CVAE, with the exception of validity. In terms of reconstruction, the main difference was seen in the accuracy quantile, as BLEU results were rather close. Finally, both models mainly differed in feature importance, with a 14.1% variation, suggesting that CVAE makes greater use of conditional information. This result may be related to the encoder architecture and loss function. Regarding the latter, even though the relationship between lambda and beta did not seem the change the results substantially, it is possible that a different set of values can help to improve the models. Exploring different combinations of lambda and beta values may offer opportunities to further enhance model performance in future research.

6.3.3 Conditional generation of manufacturing sequences

From the modelling process, two architectures based on the VAE framework – CVAE and Invariant CVAE – succeeded in generating plausible manufacturing sequences. Apart from producing simple chains of operations, these approaches also achieved to output comprehensive procedures which link materials and operations, as well as provide initial values for certain conditions. In this regard, these models can be comparable to predictive approaches, such as Transformers-based models developed by Vaucher et al [43]. An illustration of a generated procedure for a given target substance and set of materials can be seen in Table 6-17. Nonetheless, it is worth noting that there are still several challenges and limitations for their application which will be further extended throughout this section.

Table 6-17. Example generated vs actual manufacturing sequence. CVAE and Invariant CVAE were trained using PI with target KL = 35.

No	Item	Sequence
1	Materials	copper(II) acetate (REACTANT); 4-(trifluoromethoxy)phenylboronic acid (REACTANT); 2-oxo-1,2-dihydropyridine-3-carbaldehyde (REACTANT); 2-oxo-1-(4-(trifluoromethoxy)phenyl)-1,2-dihydropyridine-3-carbaldehyde (TARGET)

	Ground Truth	1: ADD 2-oxo-1,2-dihydropyridine-3-carbaldehyde, 2: ADD 4-(trifluoromethoxy)phenylboronic acid, 3: ADD copper(II) acetate, 4: ADD, 5: IRRADIATE at > 80°C for 20 min, 6: COOL to 25°C, 7: CONCENTRATE, 8: PURIFY by chromatography, 9: YIELD: 2-oxo-1-(4-(trifluoromethoxy)phenyl)-1,2-dihydropyridine-3-carbaldehyde
	CVAE	1: ADD, 2: ADD 4-(trifluoromethoxy)phenylboronic acid, 3: ADD 2-oxo-1,2-dihydropyridine-3-carbaldehyde, 4: ADD copper(II) acetate, 5: STIR at > 80°C for 20 min, 6: MICROWAVE, 7: COOL to 25°C for 2 h, 8: QUENCH, 9: WASH with brine, 10: DRY SOLUTION, 11: FILTER, 12: EVAPORATE solvent, 13: PURIFY by chromatography, 14: ELUTE, 15: YIELD: 2-oxo-1-(4-(trifluoromethoxy)phenyl)-1,2-dihydropyridine-3-carbaldehyde
	Invariant CVAE	1: ADD copper(II) acetate, 2: ADD 2-oxo-1,2-dihydropyridine-3-carbaldehyde, 3: ADD 4-(trifluoromethoxy)phenylboronic acid, 4: STIR at 25°C for 48 h, 5: PARTITION, 6: SEPARATE organic phase, 7: DRY SOLUTION, 8: EVAPORATE solvent, 9: YIELD: 2-oxo-1-(4-(trifluoromethoxy)phenyl)-1,2-dihydropyridine-3-carbaldehyde
2	Materials	6-chloro-1,7-naphthyridin-4-yl trifluoromethanesulfonate (REACTANT); piperidin-3-yl-carbamic acid tert-butyl ester (REACTANT); DIPEA (REACTANT); DCM (SOLVENT); (s)-tert-butyl 1-(6-chloro-1,7-naphthyridin-4-yl)piperidin-3-ylcarbamate (TARGET)
	Ground Truth	1: ADD 6-chloro-1,7-naphthyridin-4-yl trifluoromethanesulfonate, 2: ADD piperidin-3-yl-carbamic acid tert-butyl ester, 3: ADD DIPEA, 4: ADD DCM, 5: STIR at 25°C for 6 days, 6: PARTITION with ethyl acetate, 7: SEPARATE, 8: WASH organic phase, 9: DRY SOLUTION with MgSO4, 10: CONCENTRATE, 11: PURIFY by chromatography, 12: YIELD: (s)-tert-butyl 1-(6-chloro-1,7-naphthyridin-4-yl)piperidin-3-ylcarbamate
	CVAE	1: ADD 6-chloro-1,7-naphthyridin-4-yl trifluoromethanesulfonate, 2: ADD piperidin-3-yl-carbamic acid tert-butyl ester, 3: ADD DIPEA, 4: ADD DCM at 0°C for 20 min, 5: HEAT to 25°C for 48 h, 6: PURIFY by chromatography, 7: YIELD: (s)-tert-butyl 1-(6-chloro-1,7-naphthyridin-4-yl)piperidin-3-ylcarbamate
	Invariant CVAE	1: ADD DCM, 2: ADD 6-chloro-1,7-naphthyridin-4-yl trifluoromethanesulfonate, 3: ADD DIPEA, 4: ADD piperidin-3-yl-carbamic acid tert-butyl ester at 25°C, 5: STIR for 5 h, 6: COOL to 25°C, 7: ADD, 8: EXTRACT with ethyl acetate, 9: WASH extracts, 10: DRY SOLUTION with MgSO4, 11: REMOVE solvent, 12: YIELD: (s)-tert-butyl 1-(6-chloro-1,7-naphthyridin-4-yl)piperidin-3-ylcarbamate

As generative models, CVAE and Invariant CVAE can assist process design to explore manufacturing space and find multiple routes for producing a molecule. Compared to predictive approaches, at the generation stage, the decoder receives not only materials but also requires latent features, which represent the manufacturing procedure and can act as procedure descriptors. For the trained models, latent features consisted of 128

variables with a standard normal distribution. In this manner, upon feeding the decoder with randomly sampled variables from this distribution and information on materials, manufacturing procedures can be generated. In turn, the random nature of the sampling process makes sequence generation stochastic. Consequently, while predictive approaches tend to associate inputs with a single and most likely output, generative models can potentially provide a wider variety of procedures. It is worth mentioning that a variety of predictions can be imposed in predictive models through different sampling techniques such as beam search or random search, for instance, in deep learning models. However, the trend to output a particular sequence associated with an input is still expected.

Resuming the diversity of sequences in generative models, this aspect was reflected in the uniqueness of sequences created. Of 1000 procedures generated for 5 molecules of the test set with their respective reactants, 985 and 922 variants were obtained, on average, employing CVAE and Invariant CVAE, respectively. Additionally, 77.9% and 81.5% of these sequences were valid for CVAE and its counterpart, under the criteria that these reported all the materials involved. The invariant version had a greater sparsity in the number of sequences ranging from 772 to 1000, which suggests that the diversity of outputs may have a stronger dependence on conditional information for this model. Aligned with exploration capabilities, these values confirm the multiplicity of options that can be produced. Nevertheless, this property also entails the practical challenge of ranking the most appropriate or convenient procedures efficiently to guide selection.

In this regard, rule and model-based approaches could complement the generative models. In the first case, sequences could be filtered according to predefined patterns indicating the presence or absence of certain operations or conditions. Alternatively, additional models can be coupled with the generative networks to estimate a property that characterises the fitness of the output. This idea has been implemented in different domains such as drug discovery and chemical reaction generation to carry out a target search and ensure the validity of the outputs. More specifically, in drug discovery, neural networks, which make use of latent information, have been applied by Gomez-Bombarelli et al to predict molecular properties such as partition coefficient or synthesisability [208]. In the same manner, for chemical reactions, R. Tempke and T. Musho utilised semi-empirical techniques to determine Gibbs free energy of new generated reactions to

establish thermodynamical favourability [60]. Motivated by these examples, a neural network was trained to predict manufacturing process yield.

Yield was tested since it is a widely employed performance metric for processes and was obtained in the consulted patents, thereby being accessible for model training. In our models, yield was predicted using latent representation obtained either from CVAE or Invariant CVAE, and MACC Keys of the materials. More details about the architecture and training can be found in https://zenodo.org/records/13377654. Yield-BERT was used as a reference model since it was trained using datasets built on patents, as in our case [188]. The results are summarised in Table 6-18. The overall performance was low, although it is comparable to the reference. It has been argued that data from patents can be noisy due to errors intrinsic to the source or found in the extraction, hindering model learning [188]. Nonetheless, under a similar dataset split scheme, there was a slight improvement using the latent representation. This might indicate that the information on the process could benefit yield predictability. Another aspect to highlight, it has been proposed that smoothing techniques can improve results as shown in the table [188]. In addition, while Yield-BERT was trained in different datasets as each represents a different distribution of yield [188], that was not considered in our model training. Finally, although neither CVAE nor Invariant CVAE latent space generated accurate predictions, they might still be employed for tasks that do not require high accuracy. Specifically, further research could focus on the discrimination between low and high-yield reactions and the confirmation of the effect of the inclusion of process descriptors. In addition, as reported in other works, data preprocessing and stratification might enhance prediction performance.

Table 6-18. Determination coefficient (R^2) for process yield prediction for different models. Yield-BERT's authors trained that model using datasets: Gram and subgram. The former compile values of yield obtained when the final amount of product was in the order of grams. Subgram uses data with amounts in the order of miligrams.

Model	Dataset	Dataset split	R^2
Invariant CVAE	This work	Random	0.276
CVAE	This work	Random	0.250
Yield-BERT[188]	Subgram	Random	0.195
	Gram	Random	0.117
	Subgram	Random (smoothed)	0.388
	Gram	Random (smoothed)	0.277

6.3.4 Expert assessment

The results of the survey are summarised in Table 6-19. In general, both models were able to produce outputs comparable to the ground truth. A model was successful if the generated sequence was at least as good as the actual sequence, i.e., if the expert chose both or the generated sequence. The average success rates for CVAE and Invariant CVAE were 40% and 32%, respectively, indicating that CVAE can generate a higher proportion of plausible sequences compared to the invariant version. Nonetheless, when performance was compared under different sets of materials, it was found that invariant CVAE seemed to be less sensitive to changes in the inputs. This was evident as CVAE yielded success rates of 12% and 68% for two different sets, whereas invariant CVAE produced values of 28% and 36%, showing lower variability.

Table 6-19. Results survey.

Survey	Model	Target set	Actual	Both	Generated	None
1	Inv CVAE	Set 1	6	5	2	12
2	CVAE	Set 1	7	0	3	15
3	Inv CVAE	Set 2	7	4	5	9
4	CVAE	Set 2	7	15	2	1
All			27	24	12	37

When questioned about the motivation for their choices, some experts indicated that they found inconsistencies in addition order of reactants, missing information, or illogical order of certain operations. In this manner, this feedback provided initial insights into potential errors in the generated sequences. A deeper analysis on the characteristics of the chosen sequences led to conclude that, firstly, a lack of information on reactants often resulted in none of the sequences being selected. When none of the sequences were selected, the average number of materials reported was 1.9, the lowest compared to the other cases. On the other hand, when the actual or generated sequences were chosen, the average number of materials was 2.7 and 3.3, respectively. This suggests that the lack of information on critical materials negatively impacts the quality and interpretability of the generated manufacturing procedures. A limitation in the dataset was the loss of information on materials. Thus, this suggests that improving the completeness of the dataset in terms of the materials involved in each reaction can enhance model performance. This could be achieved by improving the methodology employed to extract

information or by applying complementary tools to complete missing reactants or detect inconsistencies [214].

Another aspect revised was the number of steps of the sequences. It was seen when the generated procedures were chosen, these had more steps compared to the actual sequence in 66.7% of the cases. This might indicate that experts may favour one over the other due to the inclusion of more details. To mitigate this potential bias, future assessments could compare the actual sequence with multiple generated sequences or avoid comparisons and evaluate sequences individually.

Finally, expert opinion allows the evaluation of the plausibility of the generated sequences by contrasting them to actual sequences. While both models assessed in the survey demonstrated to generate suitable procedures, this did not occur in the majority of the cases, which makes models to require further refinements to be applicable in practice. Apart from this, the survey also highlighted characteristics that affect the quality of the procedures generated. In this regard, patterns related to addition order of reactants, missing information, and degree of detail in the procedures were found to be indicative of suitability of a generated sequence to be executed. Considering, these aspects could be of help in future refinement of the models or in designing rules that filter out invalid sequences.

6.4 Summary

Deep generative models were successfully trained to generate manufacturing sequences for primary manufacturing. This was achieved by employing a training set consisting of more than 385K manufacturing procedures obtained from patents between 1976 and 2021. It is worth remarking that the dataset contained mainly procedures at lab-scale using batch mode. Nearly 90% of records reported producing between 14 mg and 20 g of target product. In terms of conditions, the majority of processes were executed at room temperature. Other aspects such as pH, atmosphere, and time were more diverse. As for operations, purification via chromatography was found to be widely applied in the collected procedures. Consequently, this operation was frequently found in the generated sequences. However, through manual revision, it was found that other types of purification techniques are also mentioned. Regarding materials, a wide diversity of substances was extracted. Although substances such as water or sodium sulphate were

recurrently used in different operations, these did not represent a vast difference with respect to other materials. Finally, more than 350K target molecules, which included active ingredients and synthesis intermediates, were obtained. The distribution of relevant properties in the pharmaceutical context, such as solubility, molecular weight, and partition coefficient were comparable to FDA list of active ingredients. In this manner, the type of substances typically included in the training set can be categorised as small molecules (< 500) of pharmaceutical interest.

With regards to the models, VAEs emerged as the best models for the generation of manufacturing sequences. Compared to other autoencoders such as AAE and ARAE, these showed more stability during training and superior values in performance metrics such as reconstruction accuracy and condition importance in the reconstruction. Subsequently, the effect of different architectural aspects on performance metrics was evaluated, being the most relevant pooling mechanism in the encoder and the method to control the weight of KL divergence in the loss function.

Finally, two variants of VAE, CVAE and Invariant CVAE, offered the best results based on automatic performance evaluation. Both approaches were transformers networks conditioned on a set of materials of a chemical reaction represented by their respective MACC Keys. These models output a sequence of operations which also refers to certain conditions – temperature, pH, atmosphere, and time – and the materials intervening in certain operations. This representation describes better the process with regard to the simple sequence of actions. Nonetheless, it still faces challenges to be a complete and sufficiently detailed description of a process due to the lack of conditions or materials reported in a substantial number of procedures. These limitations are explained by the extraction process and ambiguity exhibited by patents.

Qualitatively, the models generated logical sequences with a high proportion of valid examples (>70%). The reconstruction accuracy, which can serve as an indicator of how faithful the generated samples can be to the original data, was higher than 95% using BLEU score for both models. Interestingly, when the condition (set of materials) was permutated, changes in the reconstruction of approximately 45% and 31% were observed for CVAE and its invariant counterpart, respectively. This suggests that materials may have a more important role in the generation process in CVAE than in the invariant, which could point out that the former model can have a better customisation capability.

However, this aspect cannot be measured directly by these metrics as well as the experimental validity by which manual assessment is required. On the other hand, Both VAEs exhibited their potential to generate diverse manufacturing procedures. For instance, more than 90% different variants were obtained in 1000 procedures generated for 5 different molecules. However, the excess of options, without a method of prioritisation, can hinder the practical utility of the model. In this manner, alternatives such as process yield estimation for each process were discussed. Aligned with other limitations of the models in practical scenarios, apart from the lack of completeness in the process descriptions and the certainty in experimental validity, it is necessary to ensure the safety of the process and prior knowledge of the materials to be used. The last item can be addressed through complementary models for the predictions of chemical reactions and conditions, for instance, the methods cited in this chapter.

To complement this, a survey was carried out to assess the plausibility of the generated sequences. The survey results indicated that both the CVAE and Invariant CVAE models were capable of generating sequences similar to the actual ones, with success rates of 40% and 32%, respectively. While CVAE produced a higher proportion of plausible sequences, Invariant CVAE demonstrated more consistency across different material sets, with less variability in its success rates. Expert feedback revealed that issues like inconsistencies in the order of reactant additions, missing information, and illogical steps were common errors in the generated sequences. Further analysis suggested that sequences with fewer materials were often rejected, emphasising the importance of a more complete dataset. Additionally, sequences with more steps were frequently preferred, indicating a potential bias. These insights point to the need for further refinement of both models, with future efforts focusing on resolving these issues to improve their practical usability.

Chapter 7. Deep Generative Models for Secondary Manufacturing Process Design

7.1 Introduction

Thus far, deep generative models (DGMs) have proven to be capable to generate manufacturing sequences with the advantages and drawbacks discussed previously for primary processing. Interestingly, this domain also comprises most of the efforts in the development of DL applications for assisting process design. That could be seen in the diversity of datasets and models available for tasks related to synthesis of active ingredients. Secondary processing, on the other hand, seems to be a field with many opportunities for the development of tools to aid the design of manufacturing processes of dosage forms.

Compared to synthesis and purification, whose process revolves around a target molecule, secondary processing must consider both the active pharmaceutical ingredient (API) and the vehicle to deliver this substance to a patient, or dosage form. The selection of the form obeys therapeutical, pharmacokinetic, physicochemical criteria related to an API as well as patient requirements. Once a dosage form is defined, this can be associated with certain techniques and types of materials, which are well-defined, for its fabrication. This can be illustrated using tablets as reference, which corresponds perhaps to the most studied delivery system.

Tablets hold a big portion of the market share which can be attributed to advantages in stability, administration, and cost. In terms of manufacturing, they are usually produced employing three techniques: direct compression, dry granulation, and wet granulation. It is worth mentioning that these techniques correspond to the conventional approaches for tablet preparation, however, there exists variations and alternatives. Each of these techniques has a sequence of unit operations established that can be considered as a template that can be adjusted depending on each case. Therefore, the early stages of process design typically involve the selection of the most suitable method for a given API. Consequently, several frameworks have been proposed to support a systematic selection.

In this regard, approaches such as manufacturing classification system (MCS) can be highlighted [215]. MCS makes use of API mechanical properties such as compressibility, density, load, among others to determine a manufacturability index [215]. This value can then be associated with one of the conventional processes mentioned previously [215]. As

this case, depending on the dosage form, secondary manufacturing has strongly relied on mechanistic models, empirical rules, and developer's expertise to guide the selection of operations, excipients, and/or conditions [216]. However, to the best of our knowledge, there are no tools that integrate deep learning methodologies, particularly deep generative models (DGMs), to generate chains of unit operations tailored for the manufacturing of dosage forms based on specific API properties.

Exploring the use of DGMs for this purpose offers a promising objective due to their capacity to handle complex data and generate novel solutions. DGMs could automate the generation of plausible manufacturing sequences that are optimised for various APIs. By incorporating API into the model, it would be possible to predict the most suitable manufacturing processes, including the selection operating conditions. This approach could support the early stages of process design, reduce development time, and enhance the adaptability of manufacturing processes to new APIs.

Thus, this chapter explores the applicability of DGMs for the generation of sequences of unit operations for secondary processing. This was achieved in an analogous manner as with primary manufacturing as can described in Figure 7-1. Firstly, a dataset for secondary manufacturing is built for various dosage forms. The characteristics of the training set are discussed throughout this work. Then, a DGM is trained using this data. The model employed consist of a conditional variational autoencoder (CVAE), which has proven, in this work, to provide good results in the generation of procedures in primary domain. Then, the chapter concludes with a discussion about the capacity and limitations of the approaches employed.

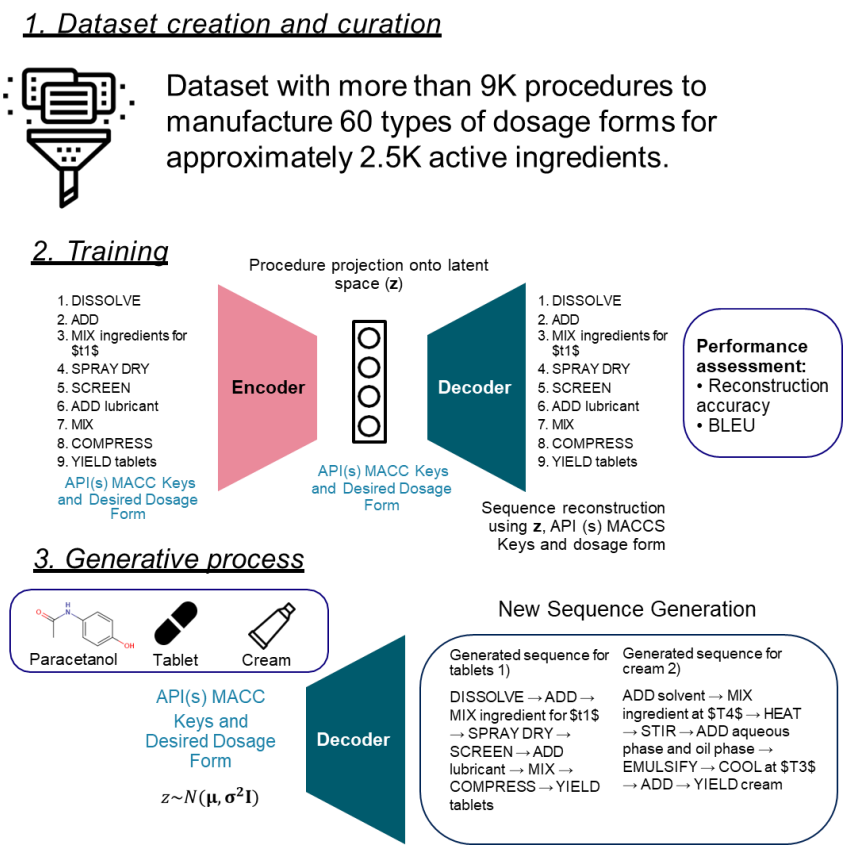


Figure 7-1. Steps for generative modelling for secondary manufacturing route generation.

7.2 Methods

7.2.1 Dataset development

After selecting candidate sections containing manufacturing procedures and extracting their information as proceeded in previous chapter, procedures mentioning dosage forms and active pharmaceutical ingredients (API) were chosen, thereby obtaining more than 89,012 records. Subsequently, information such as dosage forms, API, operations, and conditions were standardised. In the case of API, standardisation was achieved by matching the obtained names with synonyms from different sources, which included Pipeline Pilot® software and databases such as pubchem and ChEMBL[98,100]. In this way, all the APIs were expressed as InChI in the dataset. Once API were processed and duplicates removed, records with no more than 3 active substances were kept.

Similarly, dosage forms, intermediates, and operations were standardised using a semi-automated approach. As in primary processing, terms were preprocessed by removing stop-words and lower casing. Next, these were clustered using K-Means. Terms were represented through principal components analysis (PCA), employing the principal components that explained at least 80% of variance of the respective embeddings for each term. The numerical representation for the terms corresponded to fasttext embeddings pretrained for named entity recognition as indicated in Chapter 4. All the groups were then manually revised to assign a label and validate their correspondence with the terms. Thus, a total of 100 actions were defined to describe operations, while intermediate products were grouped into more than 190 categories. These intermediates referred to generic terms used to refer to states, transformations, or materials formed during manufacturing process. Finally, more than 20 classes were used to describe dosage forms.

Subsequently, data were filtered to exclude ambiguous, incomplete, irrelevant or atypical records. In this manner, procedures referring to synthesis or biological processes or containing ambiguous operations were discarded. Likewise, data without certain diversity of operations were not included. In this context, the diversity of operation was established according to the different types of operations with respect to the total number of operations. Thus, for instance, for a process with 10 steps, if there were fewer than 4 different types of operations, this was then excluded. This rule was applied to ensure the procedures were as informative as possible. Additionally, records with 3 or fewer operations were also discarded. The analysis was limited to non-sterile dosage forms that had more than 50 data points. Table 7-1 details the exclusion criteria and the number of procedures excluded. Finally, 80% of data was drawn randomly for model training and the remaining was halved for development and validation. The final dataset can be find in https://zenodo.org/records/13377701.

Table 7-1. Exclusion criteria of manufacturing procedures for secondary manufacturing dataset.

Exclusion criteria	Number of records discarded
8. Sequence with less than 3 or more than 24 steps	26,447
9. Undefined dosage form or procedures with unconventional dosage forms (less than 50 data points)	16,891

10. API without InChI identified or undefined	16,208
11. Products reporting more than 3 APIs	311
12. Procedures containing ambiguous operations, or related to primary manufacturing or biological processes	3,986
13. Sterile dosage forms	1,977
14. Diversity	1,219
15. Sequences starting with ADD	6,197
16. Duplicates	6,560
Final number of records	9,216

7.2.2 Input and Output representation

Compared to primary manufacturing where the inputs consisted of the set of materials (reactants and solvent) and target molecule, in this case, two inputs were required: the desired dosage form and the respective APIs, given their importance of both elements in the definition of the process. For the modelling, the number of dosage forms was limited to the most frequent, by which the atypical records were discarded as described in the previous section. On the other hand, active ingredients were represented using Molecular ACCess System (MACCS) Keys. Ideally, the inclusion of excipients is desirable as these also contribute to operation selection. Nonetheless, due to constraints on the data about excipients, this information was not considered in neither inputs nor outputs. This will be discussed in more detail in results section. Regarding output representation, this included the sequence of standardised actions with information on intermediates and conditions such as temperature, time, and pH, as exemplified in Table 7-2. The conditions were expressed through a class within $$ symbol, where each class indicated a range of values. The ranges defined for this work can be found in Appendix C. Finally, a YIELD token was added along with the final dosage form to indicate the end of the process.

Table 7-2. Output representation for secondary processing. $$: condition; T: temperature; t: time; if no letter is included within $$, it refers to pH.

Patent No. US5985323	**Output representati**on
EXAMPLE 4 In a stainless steel Hobart mixer were placed 3298 grams of acetaminophen and 495 grams of microcrystalline cellulose coprocessed with low viscosity sodium alginate, 85:15, prepared as described in	1. MIX ingredients for $t1$ 2. ADD solvent 3. ADD 4. GRANULATE 5. DRY 6. PLACE at $T4$ for $t4$

Example 1. These materials were blended for *five minutes* before 1250 grams of deionized water was slowly added until all powder was incorporated into granular particles. This granulation required 10 minutes after which the granules were dried on a paper-lined tray which was placed in a *60° C*. oven for *4 hours* and then allowed to remain in the oven *overnight*. The dried granular material was passed through a # 16 US standard mesh screen and then was placed in a Patterson-Kelley twin shell blender. To this blender were added 178 grams of croscarmellose sodium and 989.5 grams of Avicel® PH 102. These ingredients were mixed for 10 minutes after which 39.5 grams of magnesium stearate which had been passed through a #45 US standard mesh screen was added to the blender. Mixing was continued for an additional 5 minutes. This formulation was compressed on a B2 tablet press using tooling to prepare caplet-shaped tablets.	7. STAND for $t4$ 8. SCREEN 9. MIX 10. ADD 11. MIX ingredients for $t2$ 12. SCREEN 13. ADD 14. MIX 15. COMPRESS 16. YIELD tablets

7.2.3 Generative modelling

In the previous chapter, several approaches for generative modelling were discussed, mostly centred on generative adversarial networks (GAN) and variational autoencoders (VAE). In the end, the best performing models for primary manufacturing were VAE-based architectures. Consequently, for secondary manufacturing, the model providing best results for primary domain based on automated metrics were then adapted and tested in this chapter. Particularly, CVAE using PI controller comprising three modules - input encoder, sequence encoder and decoder employing Transformers layers - were trained, where the latter component was autoregressive. The main difference with respect to the models used in primary manufacturing was the material encoder, or in this context denominated input encoder. That module was fed with information on both the desired dosage form and active ingredients. The specifics of the architecture are described in Table 7-3.

Table 7-3. Proposed CVAE architecture based on Transformers for secondary manufacturing sequence generation.

Layer type	Activation	Hyperparameters

Input sequence		Max sequence length: 81
Embeddings for operations		Embedding dim: 256
Input encoder		
Input API		Max number of materials: 3
Input Dosage Forms		Number of categories: 21
Layer normalisation		
Dense	Identity	Units: 256
Embeddings for dosage form		Embedding dim: 256
Dropout		Dropout rate: 0.1
Transformer encoder	Relu	No. blocks: 2 Heads: 8 Key dim: 256 Feed forward: 2048 Attention dropout: 0.1 Dropout rate: 0.1
Encoder		
Transformer encoder block (conditional)	Relu	No. blocks: 2 Heads: 8 Key dim: 256 Feed forward: 2048 Attention dropout: 0.1 Dropout rate: 0.1
Transformer decoder block	Relu	No. Blocks: 2 *Self-attention* Heads: 8 Key dim: 256 *Cross attention* Heads: 8 Key dim: 256 Feed forward: 2048 Attention dropout: 0.1 Dropout rate: 0.1
PMA	Relu	Heads: 8 Key dim: 256
Dense (μ_z)	Identity	Units: 128
Dense ($\log \sigma_z^2$)	Identity	Units: 128
Decoder		
Dense	Relu	Units: 128
Transformer encoder block (conditional)	Relu	No. blocks: 2 Heads: 8 Key dim: 256 Feed forward: 2048 Attention dropout: 0.1 Dropout rate: 0.1
Transformer decoder block	Relu	No. Blocks: 2 *Self-attention*

		Heads: 8 Key dim: 256 *Cross attention* Heads: 8 Key dim: 256 Feed forward: 2048 Attention dropout: 0.1 Dropout rate: 0.1
Dense	Softmax	Units: No. Operations

As for the training scheme, model parameters were learnt as indicated in Algorithm 7-1. Adam using Noam scheduler was used as optimiser. Weights were updated for 100 epochs while monitoring performance metrics. Considering that the dataset size for secondary domain was smaller than the used in the primary processing, making models more prone to overfitting, hyperparameters such as batch size, and dropout rate were varied to choose the best combination. Dropout has been widely used to prevent overfitting and improve generalisability in neural networks [184]. Batch size, on the other hand, defines the number of updates that the model undergoes during training; it has been suggested that a smaller batch size may favour performance in architecture like Transformers, when dealing with small dataset [217]. Thus, batch sizes of 32 and 64 and dropout rates of 0.0 and 0.2 were assessed. Additionally, target KL (C) has been reported to be related to data samples (N) as indicated in this relationship $C \geq \log_2 N$ [212]. In this manner, values of 30 and 40 were also evaluated. Finally, another strategy applied in pretrained language models, such as BERT, or other autoencoders is the application of token dropout [202,218]. In this approach, input sequences of operations are corrupted through several techniques; for this work, deletion of a random step was applied. This can serve as a form of data augmentation, which could help to mitigate issues caused by smaller datasets. Deletion probabilities of 0.0 and 0.05 were experimented.

Algorithm 7-1. Training scheme CVAE with PI controller.

$\mathbf{z}$: latent space, $\mathbf{x}$: manufacturing sequence, and $\mathbf{y_1}$: API represented with MACC Keys, $\mathbf{y_2}$: dosage form, C: target D_{KL}, t: timestep, $k_p = 0.01$, $k_i = 0.0001$, $I(0) = 0.0$, $\beta(0) = 0.0$.
Initialise encoder (q) and decoder (p) weights ϕ and ψ randomly.
For each epoch:
    For each training step:
1. Calculate $\boldsymbol{\mu_z}$, and $\log \boldsymbol{\sigma_z^2}$

$$\boldsymbol{\mu}_{\mathbf{z}}, \log \boldsymbol{\sigma}_{\mathbf{z}}^{\mathbf{2}} = q_{\phi}(\mathbf{x}, \mathbf{y_1}, \mathbf{y_2})$$

2. Sample $\mathbf{z}$ using reparameterisation trick with $\boldsymbol{\epsilon} \sim N(\mathbf{0}, \mathbf{I})$.

$$\mathbf{z} = \boldsymbol{\mu}_{\mathbf{z}} + \boldsymbol{\sigma}_{\mathbf{z}} \odot \boldsymbol{\epsilon}$$

3. Reconstruct $\mathbf{x}$.

$$\tilde{\mathbf{x}} = p_{\psi}(\mathbf{z}, \mathbf{y_1}, \mathbf{y_2})$$

4. Determine loss and update parameters ϕ and ψ using Adam with learning rate set according a noam scheduler[a] with factor 0.2 and warmup steps 16000.

$$\mathcal{L} = \mathbb{E}_{\mathbf{z} \sim q(\mathbf{z}|\mathbf{x}, \mathbf{y})}[\log p(\mathbf{x}|\mathbf{y}, \mathbf{z})] - \beta \cdot D_{KL}[q(\mathbf{z}|\mathbf{x}, \mathbf{y})||p(\mathbf{z}|\mathbf{y})]$$

5. Update β with PI controller

   If $t > 0$:

   $e(t) = C - D_{KL}(t)$

   $P(t) = \dfrac{k_p}{1 + \exp e(t)}$

   If $0.0 < \beta(t-1) < 1.0$: $I(t) = I(t-1) - k_i \cdot e(t)$

   $\beta(t) = I(t) + P(t)$

   If $\beta(t) > 1.0$: $\beta(t) = 1.0$

   If $\beta(t) < 0.0$: $\beta(t) = 0.0$

Regarding the performance metrics employed, these are summarized in Table 7-4. The evaluation focused on measuring overfitting, generated samples' faithfulness, and the effect of the dosage form and API on generation. In sequence generation, the recurrent generation of the same sequences can indicate overfitting, which can be observed through uniqueness. The reconstruction accuracy, measured by the BLEU score and Levenshtein similarity, can be used to evaluate how similar the generated sequences are to the actual samples. Finally, the effect of conditions was analysed through feature permutation (FI) for both dosage form and active ingredients, to assess the impact of these factors on the sequence's customizability.

Table 7-4. Automatic performance metrics employed for generation of manufacturing sequences for secondary domain.

Metric	Definition	Calculation
BLEU score		See Chapter 3 Methods
Levenshtein similarity		See Chapter 3 Methods
Uniqueness	Proportion of unique sequences in N generated samples	$\dfrac{\#\ Unique\ sequences}{\#\ Generated\ samples}$

Condition Permutation Importance or features importance (FI)	Mean decrease in reconstruction accuracy (BLEU) after permutating the condition. Based on *grouped feature importance* proposed by Au Q et al [198].	$BLEU - \sum_{i=1}^{n} \frac{BLEU_{\text{permutated-cond}}}{n}$

7.3 Results and discussion

7.3.1 Data description

A total of 9,216 experimental procedures were collected for 21 distinct types of dosage forms and 2,614 active pharmaceutical ingredients. These data were extracted from 4,933 patents. Compared to the primary manufacturing dataset, the secondary dataset contains much fewer number of observations. This can be seen as the primary dataset has approximately 40 times the secondary data. Consequently, generative model adaptation and training may be more challenging since this condition makes deep neural networks more prone to overfitting.

Another aspect to highlight is the number of materials obtained. 107,562 inactive ingredients were identified, where 82.8% corresponded to other excipients and the remaining were categorised as solvents. Thus, on average, around 11 inactive substances intervened in each process. Figure 7-2 shows the top 10 most common substance found. It can be seen a clear dominance of water and alcohols in the use of solvents, which is expected considering the requirements of dosage forms and most of these solvents are considered safe for human consumption. Concerning the other materials, there is a predominance of compounds used widely to produce solid dosage forms, having a wide variety of functionalities within a formulation such as lubricants, fillers, disintegrants, etc. This is consistent with the most common dosage forms extracted which will be discussed later in this chapter.

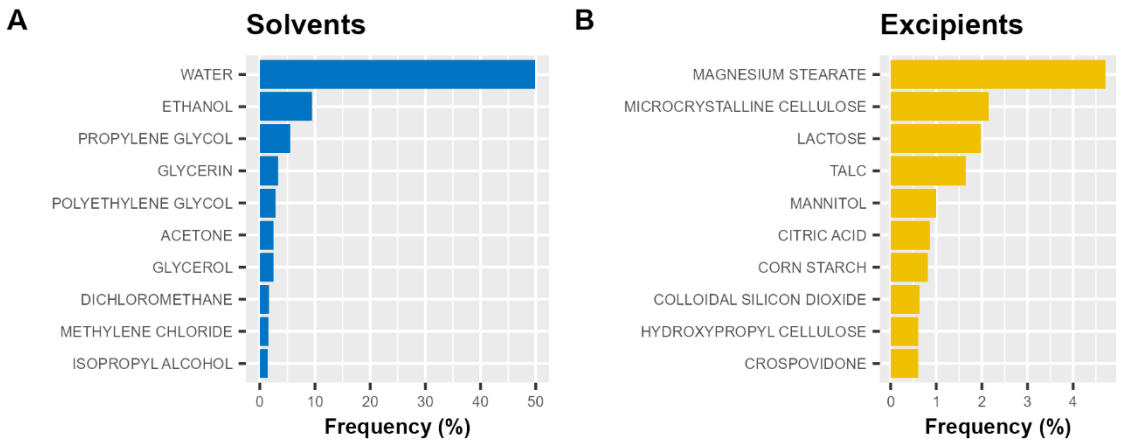


Figure 7-2 Top 10 most mentioned substances for the secondary manufacturing dataset.

Nonetheless, although many of these substances play a key role in product manufacturing, they were not considered for generative modelling mainly due to two reasons. Firstly, it is important to note that most of these excipients were reported using common denominations. While this information allows for the chemical characterization of the substances to a certain extent, it may not suffice to differentiate the relevant properties that define a chain of unit operations. For instance, microcrystalline cellulose was frequently referenced in multiple formulations. However, bulk properties, which are key to ensuring product manufacturability and efficacy, such as particle size distribution or shape, were rarely reported. Similarly, data such as grade or quality, which may provide hints about these relevant attributes, are usually missing. As a result, material information can be considered incomplete owing to source constraints.

On the other hand, although material data was incomplete, its inclusion in output representation could still provide a more informative and readable manufacturing sequence, as observed in the modelling of primary processing. However, this requires associating every material with the respective operation in which it intervened. However, this was only achieved for 2.0% of the operations, in the case of excipients, and 11.5% operations involving solvents. In contrast, a large number of operations in primary manufacturing could be linked with some material.

This contrast indicates that the methodology applied for data extraction did not work effectively for secondary manufacturing. This behaviour may be result from differences in how texts in both domains are structured. For instance, materials are directly described

in the procedures for synthesis and/or purification. However, inactive ingredients are often listed and then described with different denominations in the procedures for dosage form preparations. This structure limits the capability of the methodology to associate terms since it relies on constructing dependency trees, which is only possible if all the terms are located in the same sentence. Therefore, to leverage material information more effectively, future works should revolve around developing methods that allow the association between terms regardless of their location in the text and creating databases that provide more detailed and complete data on excipients and their properties. Considering these constrains, it was decided not to include materials such as excipients or solvents in either inputs or outputs.

7.3.1.1 Active ingredients

Of 2,614 active ingredients extracted, 356 corresponded to FDA-approved molecules. The remaining can be chemical entities that were approved at some point, new substances with the potential to be commercialised, or chemical variants of compounds currently being marketed like salts. Notably, approximately 2,178 chemical entities have been reported by the FDA for multiple drug products as of June 2024 [219]. By comparing the distribution of properties such as solubility, partition coefficient, and molecular weight of the set of APIs obtained, it was shown that there is a high degree of matching with FDA-approved active ingredients, as illustrated in Figure 7-3. Therefore, despite the lower availability of substances for training models in comparison to primary processing, the active ingredients in the dataset are comparable in number and chemical space to those on the market. This suggests that the dataset can be representative of the expected chemical space of drug products.

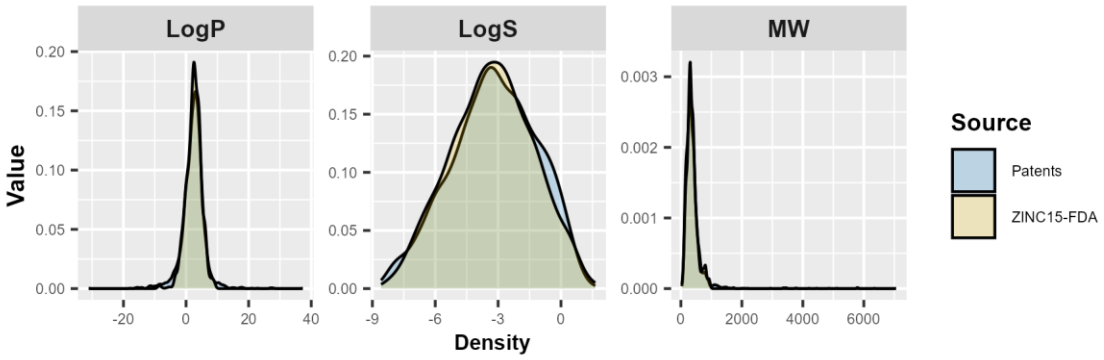


Figure 7-3 Comparison of properties distributions between secondary manufacturing dataset molecules in (n = 2,178) and approved drugs by FDA recorded in ZINC15 database (n = 1,615)[203]. LogP, log-partition coefficient,t calculated using RDkit library 31; LogS, log-molar solubility calculated using SolTranNet [206]; MW: molecular weight.

Another aspect to consider is that a drug product can contain multiple APIs. In the patents dataset, 81.4% of dosage forms only have a single active ingredient. The remaining products contain 2 or 3 APIs, representing 15.4% and 3.26% of formulations, respectively. On the other hand, FDA products presented the following distribution: 87.4% corresponded to single API products, while 9.8% and 1.6% of products have 2 and 3 active substances, in turn. Additionally, there was 1.2% of forms with 4 or more APIs. In the dataset built, the formulations with this number of active ingredients were excluded since their rarity could hinder model's ability to generalise well. Thus, it was opted to focus on the most common formulations, which are more likely to be useful in practice. In this manner, the obtained dataset, apart from being comparable in terms of the types of chemical substances, can closely represent the distribution of the most recurrent active ingredients per dosage form in real situations.

7.3.1.2 Dosage forms

Figure 7-4 shows the most frequent dosage forms found in the dataset. 21 different types of forms were identified. Solids, such as tablets, capsules, and granules, were predominant, reflecting real-world situations. As a reference of this, approximately 50% of products approved by FDA are tablets [219]. However, although tablets were dominant in the dataset, the percentage obtained did not surpass 35%.

Multiphase dosage forms, such as emulsions, creams, gels, among others, were more prevalent compared to FDA-approved products. Each of these forms did not represent more than 1% [219]. Similarly, non-conventional drug delivery systems such as

microparticles, microspheres, and nanoparticles were also present in the dataset. These systems, in particular, can be administered through various routes such as oral, topical, parenteral, or nasal [220]. This category is not explicitly found in FDA listings, as these systems are often considered intermediates used to form other products like tablets, capsules, injectables, etc [220]. However, due to their well-defined manufacturing operations, it is more convenient to categorise them separately in the generative modelling [220].

Overall, divergences between the distribution of dosage forms in the market and the dataset obtained are evident. These differences can be observed in the overrepresentation of forms which are not as widely utilised as solid forms. This may be due to the source, as patents tend to include a higher variety of products to ensure the legal protection has a wider scope. Another discrepancy was related to the number of forms included in the final version of the dataset; while FDA reported more than 60 different types of forms, this work was limited to 21, focusing on the most relevant products in practical situations. Nonetheless, the diversity of products obtained can be advantageous for a generative model, as it may enhance the learning process of different manufacturing routes and alterative for delivery systems whose information is not as abundant.

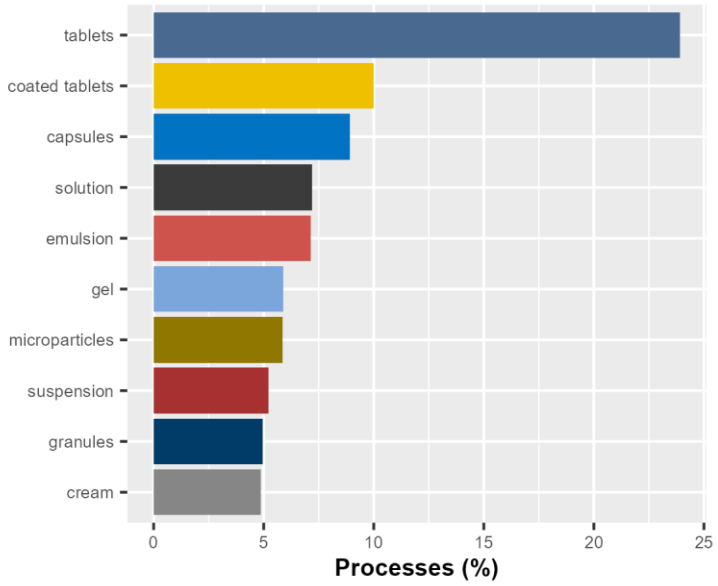


Figure 7-4 Top 10 most frequent dosage forms found in the extracted patents.

Finally, it is worth noting that sterile dosage forms were not included in the dataset, even though they represented less than 10% of the obtained data. For reference, injectables, a common sterile dosage form, account for nearly 20% of FDA-approved products [219]. The requirement for sterility adds complexity to the manufacturing process, necessitating specific operations and conditions to guarantee this property. Therefore, future work could aim to obtain more data and include sterility in the generative modelling process.

7.3.1.3 Conditions

20,271 operative conditions were extracted, obtaining data for nearly 19.6% of the identified operations. Among all types of conditions, the most frequently reported were temperature and time, representing 62% of data. Other types of conditions were more generic, such as stirring rate, which were not comparable as these depend on the type of equipment employed. Others were more dosage form specific, namely, compression force for tablets. However, each of these categories did not surpass 6% of data.

Figure 7-5 summarises the distribution of values for pH, temperature and time, which were the most widespread over different types of operations. As for pH, it can be seen the values tended to be concentrated around pH between 4.5 and 7.0. Additionally, the majority of processes with pH reported were set at acid ranges. Temperature and time, on the other hand, have a more evenly distribution. Most operations seemed to have been performed under temperatures between 25°C and 50°C. However, it is possible to highlight that actions such as ‘DRYING’ and ‘DISSOLUTION’ are typically executed at temperatures greater than 50°C. Finally, most of the operations reporting time included ‘DRYING’ and mixing related operations. Interestingly, mixing times tended to be shorter, being lower than 20 minutes. Conversely, drying processes seemed to take longer times, reporting in many cases values greater than 120 minutes.

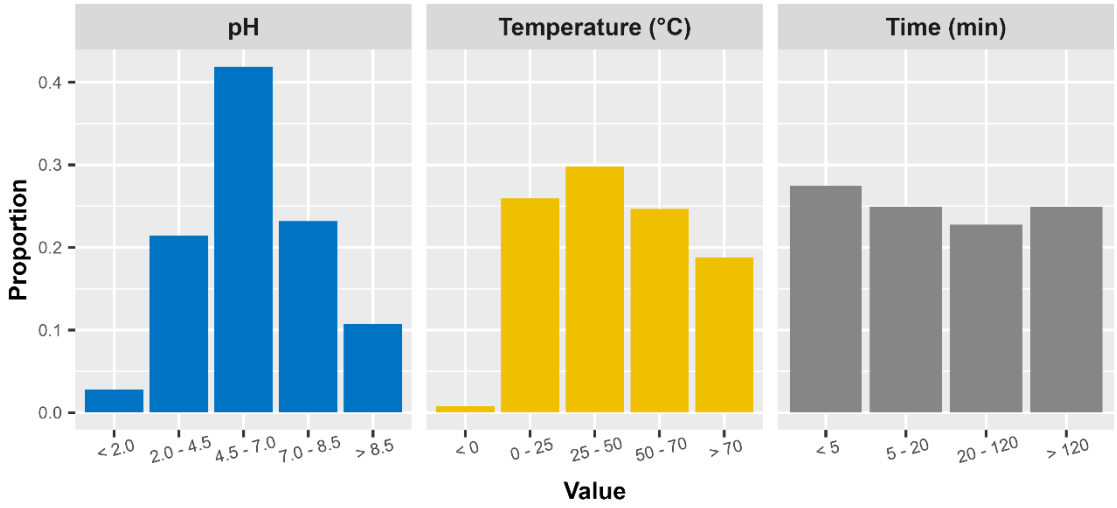


Figure 7-5. Distribution of conditions pH, temperature, and time obtained.

The conditions found were limited to certain types of operations. The most reported conditions were related to actions such as ‘HEAT’, ‘COOL’, ‘DRY’ and mixing related operations ‘STIR’, ‘MIX’, etc. Additionally, as noted previously, certain operations presented certain trends with respect to particular values. Thus, it is clear the dataset showed a clear relationship between conditions and certain operations. As a results, this leads that the trained generative models generate conditions only for these operations. This fact marks a limitation in the dataset considering that in real situation additional conditions for more types of operations can be expected.

7.3.1.4 Operations

In total, the dataset contains 77,227 operations, having an average of 8 operations per process. Figure 7-6 shows the top 10 most mentioned operations for all the processes and for different types of dosage forms. The global distribution is more similar to tablets as this dosage form is the most frequent in the dataset. Furthermore. It can be observed the different dosage forms exhibited different distribution of unit operations. Likewise, the presence of operations characteristics for each dosage form was evident. For instance, in the case of tablets, action such as ‘COMPRESS’, ‘GRANULATE’, and ‘DRY’, which are usual in a wet granulation process for tablet manufacturing. In the same manner, specific actions for the preparation of emulsion and solution can also be seen. This is consistent with the fact that each dosage form can be characterised by different types of methods and technology for their manufacturing process. This also confirms that the

dosage form is a strong conditional for process selection, in a high level, and more detailed particularities of the process are defined by API properties.

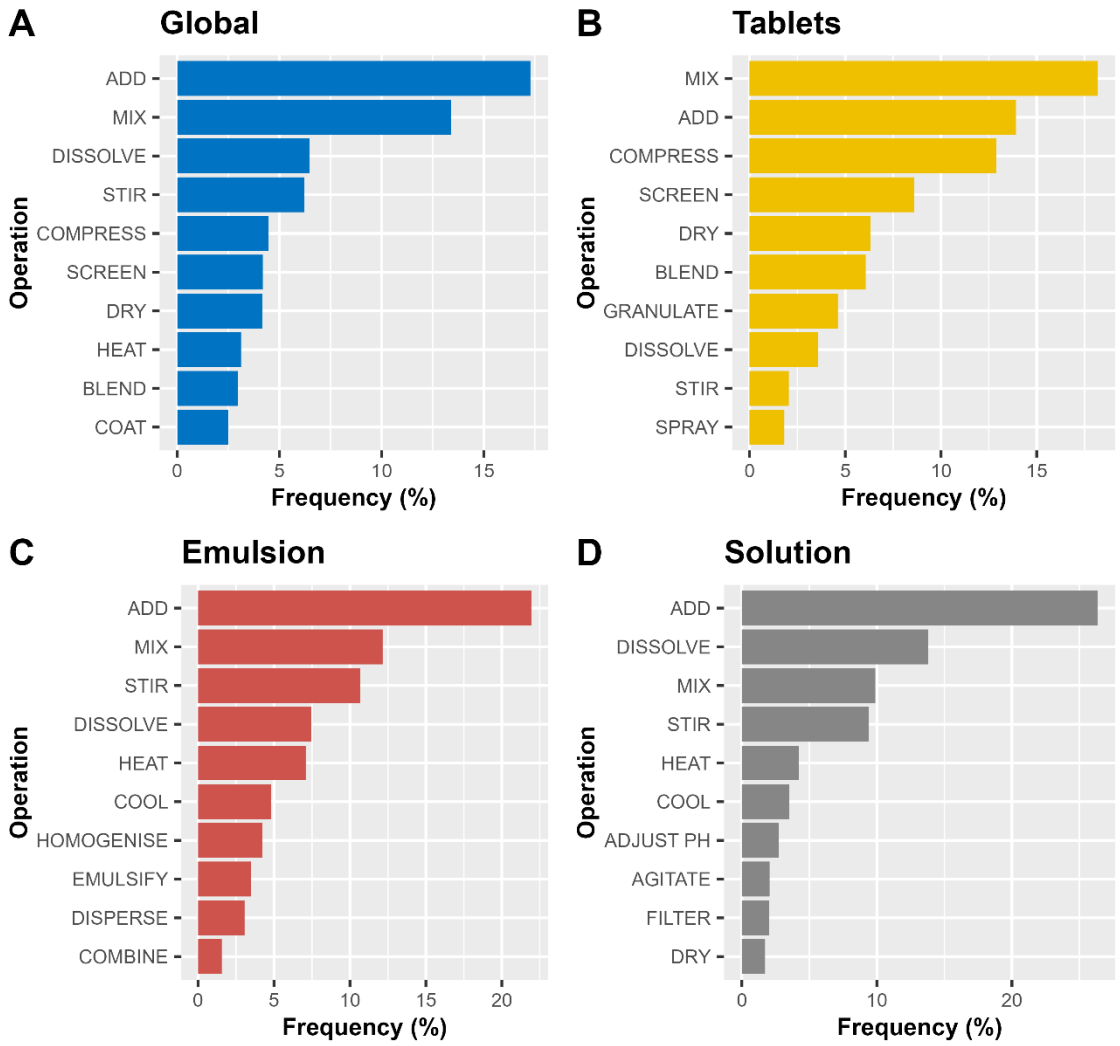


Figure 7-6 Top 10 most frequent operations found for the entire dataset (A) and some common dosage forms such as tablets (B), emulsion (C), and solutions (D).

7.3.1.5 Dataset limitations

Firstly, as with primary processing, the majority of records came from the last decade as illustrated in Figure 7-7A. This is clearly associated with that the number of granted patents has increased significantly and constantly since there is available data. As a result, the most recent procedures will have a higher weight for the modelling. Another interesting pattern was related to the evolution of the number of operations per process.

When comparing the number of operations vs time, a spearman correlation of 0.18. Although it is a weak correlation, this seems to suggest that the number of steps of a procedure is increasing across time, on average. This was the opposite behaviour to what it was observed in primary manufacturing where more recent inventions may favour efficacy over complexity. On the other hand, in secondary manufacturing, this pattern may be related to the development of dosage forms which are more complex and require more steps to achieve the desired quality.

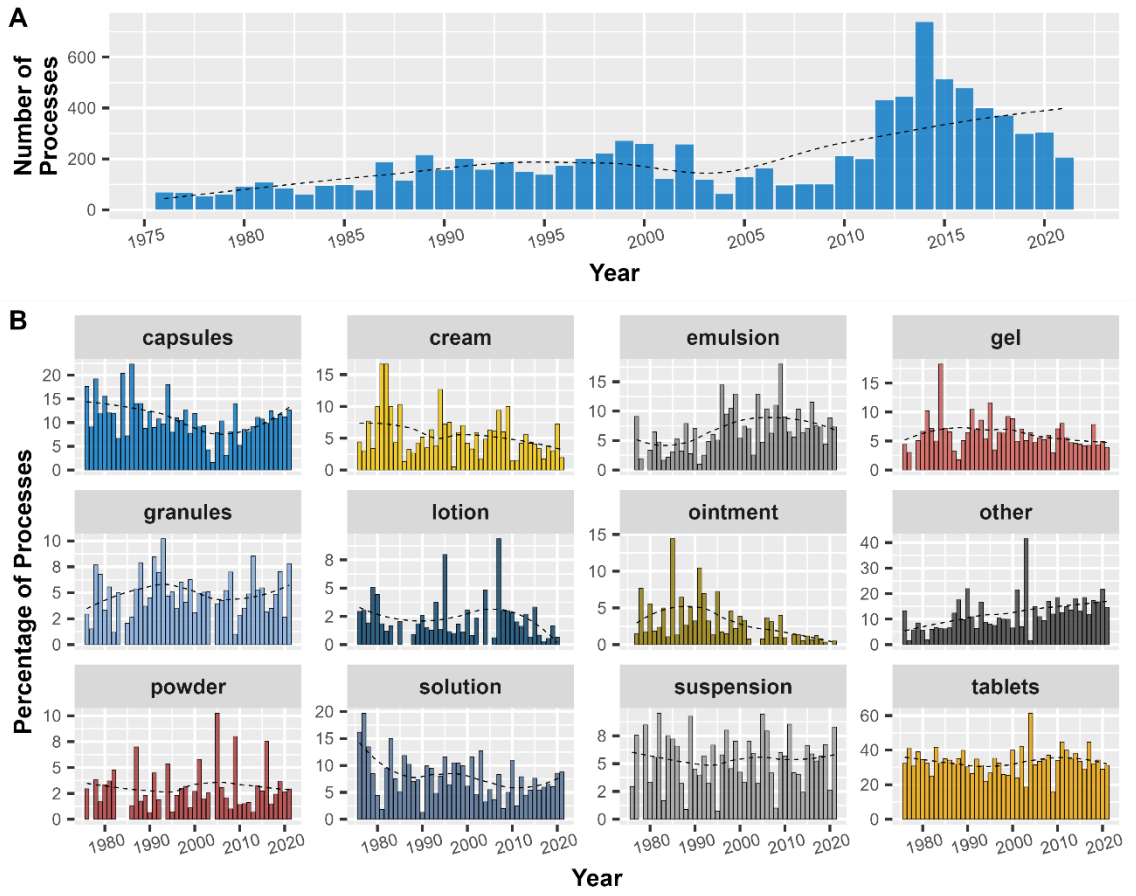


Figure 7-7 Evolution of the number of records obtained across time (A) and breakdown by form (B). Other refers to non-conventional dosage forms such as microspheres, microparticles, nanoparticles, among others.

Additionally, for secondary processing, dosage forms have been demonstrated to be a strong determinant of the manufacturing process by which their evolution across time was explored. The contribution of the difference dosage forms to all the records per year, depicted in Figure 7-7B, there are mixed results. For instance, solid dosage forms such

as tablets, powders, and granules together tended to maintain a constant contribution across time. On the other hand, semisolids like creams and ointments have been declining gradually, while the proportion of special delivery systems (others) such as microspheres, nanoparticles, among others seems to be increasing. As for the behaviour of semisolids, a review of formulations approved between 2016 and 2022 suggests that these were based on traditional approaches and industry seems to be conservative in the development of novel procedures and formulations [221]. Although this is statement is not conclusive due to the lack of complete information on technology employed [221], this may imply that the procedures employed to manufacture semisolid forms are well established and consolidated, thereby leading to fewer inventions in this area. Alternatively, another explanation is a change in research focus. For instance, delivery systems based on multiparticles are becoming more relevant. These systems offer greater control in API release, more versatility in the administration route, and enhanced product stability [220]. This also explains the increasing patents related to these products.

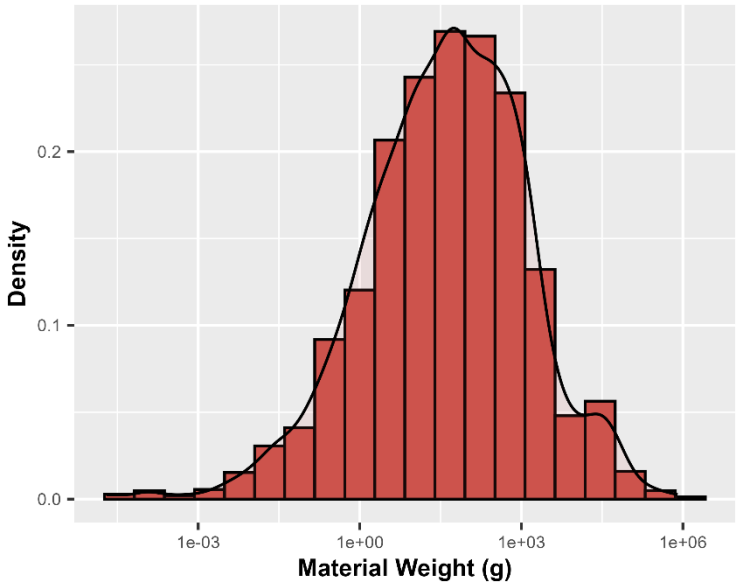


Figure 7-8 Distribution of material weighted for secondary manufacturing in log-scale. Median weight = 49.9 g (N = 2,576)

Another aspect revised was the scale of the obtained procedures. This was achieved by analysing partial information on the amounts weighed of materials extracted from 2,594 procedures. The distribution of the weights can be seen in Figure 7-8. It was found that 90% of amounts weighted between 0.1 g and 10 kg of material. Thus, the amounts of

product manufactured by these procedures can fall into lab and pilot scale. This data help to establish the scope of the processes studied in terms of scale. Nonetheless, the values obtained suggest a wider range compared to primary processing in this work.

7.3.2 Generative modelling

7.3.2.1 Selection of best hyperparameters

In this chapter, a CVAE with PI controller employed in primary processing was adapted for secondary processing to evaluate its capability in generating sequences for this mode of manufacturing. During the adaptation, dataset size and characteristics of the output representation, which included operations and conditions, were considered. Thus, hyperparameters related to model architecture, such as number of layers and units, were maintained with respect to primary manufacturing. This was done to ensure the model has the capacity to learn patterns with similar complexity. On the other hand, to address potential overfitting caused by the smaller dataset, the effect of hyperparameters such as batch size, dropout, token dropout, and target KL was assessed. The results of automatic metrics can be seen in Table 7-5.

Table 7-5. Performance metrics assessed over the test set for CVAE using different values of dropout rate (A), token dropout (B), batch size (C), and KL target (D).

A	B	C	D	BLEU	FI_{df}	FI_{API}	Accuracy 100%	gMean
0.2	0	32	30	**59.2%**	10.2%	17.2%	**22.1%**	31.9%
0.2	0.05	32	30	59.1%	13.7%	18.5%	20.0%	35.0%
0.2	0	32	15	50.8%	11.6%	17.5%	11.5%	31.9%
0	0	32	30	56.4%	11.6%	18.4%	17.1%	33.1%
0.2	0.05	32	15	51.5%	**16.2%**	**20.2%**	8.8%	**35.9%**
0.2	0.05	64	15	49.5%	12.2%	17.3%	8.7%	31.9%
0	0.05	64	30	55.1%	13.2%	18.9%	13.9%	34.2%
0	0.05	32	15	46.9%	12.6%	18.2%	6.2%	32.2%
0.2	0	64	15	49.9%	12.7%	17.3%	11.3%	32.4%
0.2	0	64	30	58.6%	12.6%	19.0%	18.1%	34.4%
0	0	32	15	48.2%	14.2%	18.3%	10.2%	33.4%
0.2	0.05	64	30	57.4%	11.2%	16.4%	14.6%	32.1%
0	0	64	30	55.4%	13.6%	19.3%	16.2%	34.7%

0	0.05	64	15	46.4%	13.3%	17.5%	7.5%	32.2%
0	0.05	32	30	53.3%	14.3%	18.8%	10.2%	34.6%
0	0	64	15	49.3%	14.6%	18.7%	8.6%	34.1%
		Average		*52.9%*	*13.0%*	*18.2%*	*12.8%*	*33.4%*
			CV	*8.4%*	*11.4%*	*5.3%*	*37.2%*	*4.0%*

Variations in all the metrics were observed indicating an effect caused by the changes performed. Interestingly, an inverse relationship between feature importance of either dosage form or API and reconstruction accuracy was identified. This relationship was also detected for primary manufacturing, which indicates that the trade-off between condition importance and reconstruction is maintained. This relationship also explains that the geometric mean has the lowest variability as one of the metrics can compensate the others. Another noteworthy aspect was that API features seem to be more important than dosage form as these generate a greater change in BLEU. This might point out that high level processes are conditioned by dosage form, while specific details are determined by API properties. Additionally, a strong association between FI_{API} and FI_{df} was observed, with a Pearson correlation of 0.76, which suggests that both metrics are being affected in a similar way by the hyperparameters. However, the coefficient of variation of FI_{df} is approximately twice as high as that of FI_{API}, indicating that the former may be more sensitive to certain changes in model settings.

The effect of the hyperparameters on BLEU and FI_{df} are shown in Figure 7-9. As can be seen in normal plots (Figure 7-9B and C), the most important hyperparameters for reconstruction corresponded to target KL (D) and dropout rate (A). Batch size (C) and token dropout (B), and all the interactions did not seem to have a substantial impact on this metric compared to factors A and D. By observing the magnitude of the main effects displayed in Figure 7-10A and B, the highest levels for dropout ratio and target KL yield a higher BLEU. Although batch size and token dropout were not as important, the effects were negative, thereby suggesting that a low batch size and the absence of token deletion favour reconstruction accuracy. On the other hand, the factor with the highest effect for FI_{df} was an interaction between batch size (C) and token dropout (B), illustrated in Figure 7-10C. The combination of hyperparameter that increased FI_{df} was a batch size of 32 and a token dropout of 0.05. The other factors seemed to have a lower effect where

both, dropout and target KL were negative. This result is consistent with the inverse relationship found between reconstruction and feature importance.

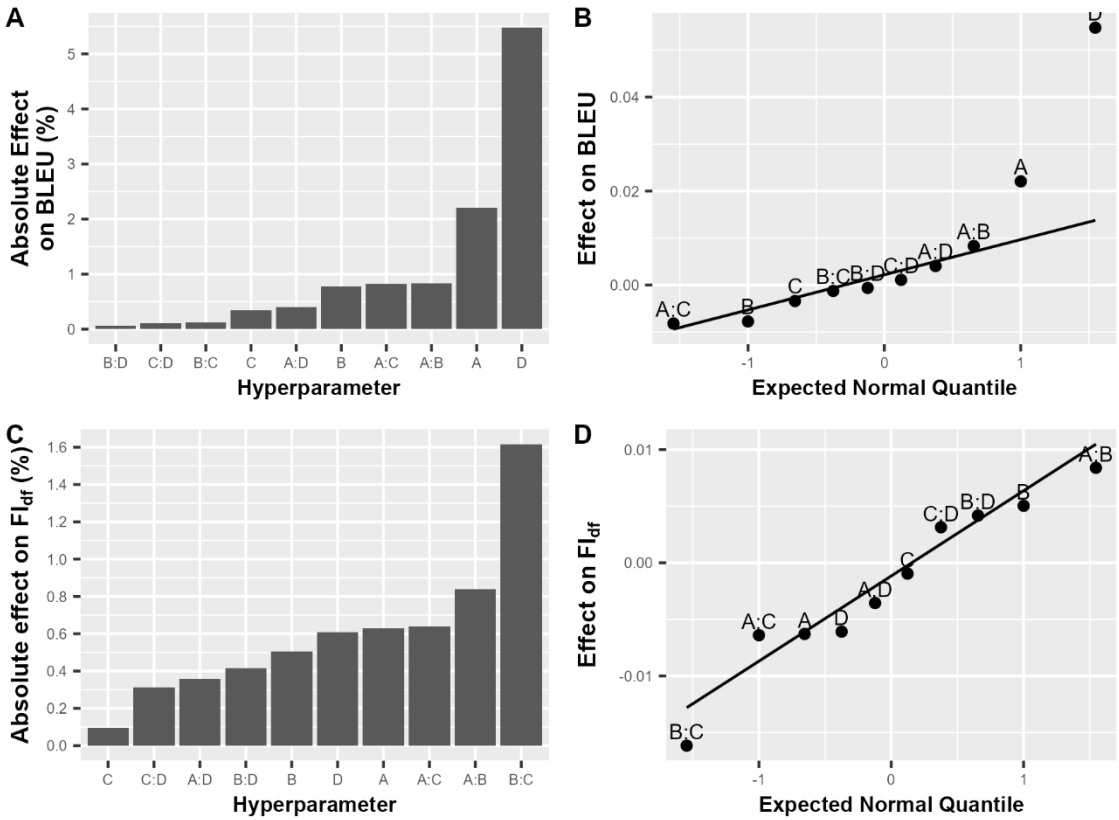


Figure 7-9 Effects of hyperparameters on reconstruction (BLEU) and dosage form importance (FI_{df}). Plots A and C represent the absolute effect for BLEU and FI_{df}, respectively. Plots B and D correspoind to the normal plots of the effects for BLEU and Fi_{df}, respectively. The hyperparameters codes are dropout rate (A), token dropout (B), batch size (C), and KL target (D).

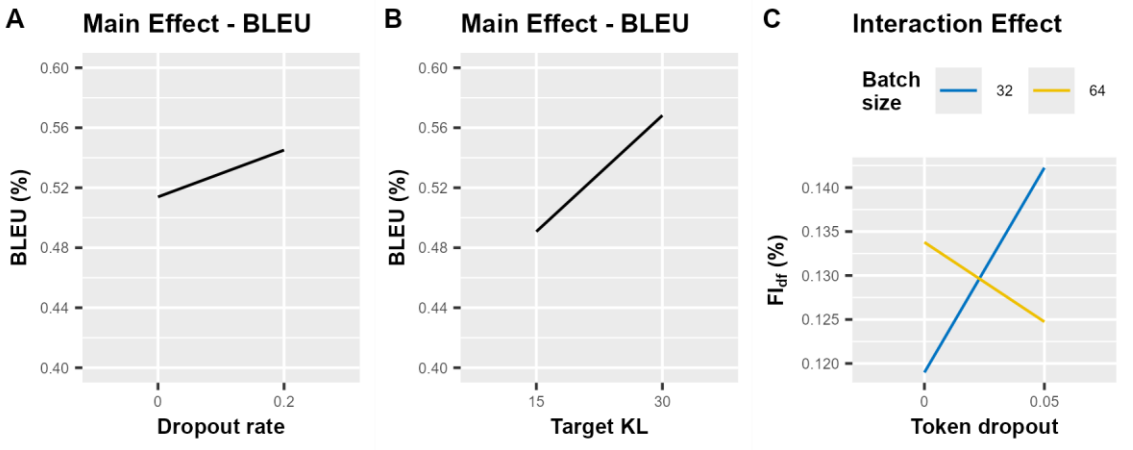


Figure 7-10. Most important effect for reconstruction (A – dropout rate and B – Target KL) and dosage form importance (C – Interaction Token dropout + Batch size)

When comparing the effect of reconstruction accuracy and dosage form importance, mixed results were obtained. For instance, while a high dropout rate, a high target KL, and absence of token dropout yielded a greater reconstruction, the opposite behaviour happened with feature importance. Nonetheless, the only hyperparameter which has a similar behaviour in both metrics was batch size, whereby the lowest value was more convenient for model performance. In this manner, a batch size of 32 was chosen. It is also worth mentioning that the effect of batch size was remarkable, widening the range of the levels assessed might help to evidence a clearer difference. The remaining factors were set considering the following rationale. Despite the fact token dropout may affect negatively the reconstruction, this effect was not too strong. Conversely, it was more important for feature importance, by which token dropout was considered necessary. The same was applied for reconstruction, where dropout rate and target KL were more important for this metric, while these did not have a substantial influence on feature importance. Thus, the model with dropout rate of 0.2 and target KL of 30 was chosen as a reference.

7.3.2.2 Generation of manufacturing sequences

The selected model offered a reasonable balance between the metrics assessed. The reconstruction, and feature importance for both API and dosage form were 59.1%, 18.5%, and 13.7%, respectively. All these values were superior to the average of all the models trained. Some examples of generated sequences are displayed in Table 7-6. The

generation process carried out by the decoder, which is fed with APIs expressed as MACC Keys, a selected dosage form, and a 128-dimensional vector drawn from a normal distribution. For the model, the selection of dosage form is limited to 21 options. The model demonstrated to be capable of generating sequences with certain degree of customisation. This can be observed as the output contained operations which are characteristic of the input dosage form. Similarly, the model was also able to assign conditions to certain operations. On the other hand, different molecules produced different procedures, which can be an indicator of the diversity of the sequences generated. However, this can also be result from the stochastic nature of the algorithm.

Table 7-6. Examples of generated sequence for different APIs and dosage forms.

API/dosage form	Output Sequence
Naproxen	
Tablets	DISSOLVE ; ADD ; MIX ingredient for $t1$; SPRAY DRY ; SCREEN ; ADD lubricant ; MIX ; COMPRESS ; YIELD tablets
Cream	ADD solvent ; MIX ingredient at $T4$; HEAT ; STIR ; ADD aqueous phase and oil phase ; EMULSIFY ; COOL at $T3$; ADD ; YIELD cream
Solution	ADD ; DISSOLVE ; STIR ; ADD solution at $T3$; DISSOLVE ; STIR ; ADD ; YIELD solution
Irbesartan	
Tablets	MIX ; ADD blend and mixture ; MIX ; COMPRESS ; FILL ; SEAL ; ADD ; YIELD tablets
Cream	HEAT at $T4$; MELT ; ADD aqueous solution and ingredient ; EMULSIFY ; YIELD cream
Solution	ADD aqueous dispersion and solvent ; DISSOLVE ; ADD ; STIR ; YIELD solution

Qualitatively, it could be observed, in some cases, there were inconsistencies in certain procedures. For instance, operations related to tables such as ‘COMPRESS’ were present in procedures for solutions or other types of forms. Additionally, uncommon patterns were sometimes generated. Specifically, operations typically found at the beginning or middle of the process were included at the end of the sequence. Future work could develop a validity metric which enable the detection or the quantification of this types of inconsistencies. This could be incorporated during the training process so that a more appropriate optimisation can be performed. In addition, this metric could potentially help to detect and clean defective sequences in the dataset.

Another drawback of the model is the lack of readability of the output. This was mentioned previously in method section. As can be seen in the example, the simple sequence of operations did not provide enough information to proceed to a next scenario which would be the execution in a lab environment. The lack of excipients in the sequences mark a great difficulty in the interpretation of the procedures as it is not known what is being added or mixed, for example. Due to this limitation, the model was not further validated through experts' revision as it happens with primary manufacturing. In order to improve this, improvements in the extraction of data are necessary. In addition, the development of descriptors for excipients useful for machine learning. Although materials can be represented chemically through several methods such as SMILES, InChI, etc. In secondary processing, a single chemical entity can have variants with different bulk properties which are more relevant in the development of dosage forms.

Finally, the proposed model demonstrated several advantages, including the ability to generate diverse and customised sequences of unit operations tailored to specific APIs and dosage forms. Despite the disadvantages mentioned previously, this model could provide high level information on the potential technique or methodology to be applied to manufacture a particular pharmaceutical product. For instance, in tablets context, it could indicate whether a dry processing or wet granulation could be feasible or convenient by exploring multiple scenarios created by the model.

7.4 Summary

Deep generative modelling could be applied to generate sequence of manufacturing for secondary manufacturing. Using a dataset with more than 9 thousand procedures built on patents, a CVAE using PI controller was trained. Regarding the training set, this contains 9,216 experimental procedures covering lab and pilot scale, including 21 types of dosage forms and 2,614 active pharmaceutical ingredients, derived from 4,933 patents. APIs in the dataset closely match FDA-approved molecules in terms of properties such as solubility, partition coefficient, and molecular weight, suggesting the dataset's representativeness of the chemical space of drug products. The dataset primarily consists of single-API dosage forms, with the remaining containing two or three APIs, aligning with the distribution seen in FDA-approved products.

The dataset is significantly smaller than the primary manufacturing dataset, which presents challenges for training generative models due to the risk of overfitting. The dataset also includes 107,562 inactive ingredients, mostly excipients and solvents, but their incomplete and inconsistent reporting limits their usefulness for generative modelling. However, the dataset is limited by its focus on commonly reported conditions and a lack of complete information on newer dosage forms and sterile products. Future work should aim to enhance data extraction techniques and expand the dataset to include more detailed and comprehensive material and condition information.

With the obtained, various models were trained maintaining a similar architecture to handle similar complexity patterns to that used for primary processing. Hyperparameters such as batch size, dropout, token dropout, and KL target were adjusted to prevent overfitting due to the smaller dataset. The evaluation revealed that changes in hyperparameters impacted performance metrics, with dropout rate and KL target being significant for reconstruction accuracy (BLEU), while the interaction between batch size and token dropout was crucial for dosage form importance (FI_{df}). Based on these results, hyperparameters were chosen for the final model, balancing reconstruction and feature importance.

The model showed customisation capability, generating sequences with operations characteristic of the input dosage form and assigning conditions to certain operations. However, some inconsistencies were noted, such as inappropriate operations for specific dosage forms and uncommon patterns in the sequences. Additionally, the lack of excipients in the output sequences limited their readability and practical application. Future improvements could include developing a validity metric for sequence consistency, enhancing data extraction for excipients, and exploring fingerprints for API and excipients that represent bulk properties. Additionally, the model could benefit from architecture variations to inject conditional information and assessing different types of regularisation techniques. These points could help to improve performance. Despite these drawbacks, the model could offer potential insights into suitable manufacturing techniques for specific pharmaceutical products.

Chapter 8. Conclusions

8.1 Conclusions and future work

Throughout this thesis, deep generative models (DGMs) with the capacity to generate chains of unit operations were successfully trained for both primary and secondary manufacturing. The suitability of different types of models, widely used in multiple fields such as drug discovery, were explored which included variational autoencoders (VAE) and generative adversarial networks (GAN). Particularly, for the latter, a variant known as adversarial autoencoders (AAE) was evaluated. In the end, conditional variants were employed to customise output sequences making them suitable for a target molecule and intervening reactants, in the case of primary manufacturing, or active ingredients and dosage form, for secondary processing.

In the first instance, for the primary domain, this work demonstrates the potential of DGMs to create procedures not limited to only a set of unit operations, but also including details such as some operative conditions and/or materials addition order. Interestingly, the generation process makes use of properties of the target substance and reactants involved, providing some level of customisation of the output sequence. In this manner, the model provides informative and readable outputs which could assist to establishing proper experimental settings in early stages of design and development of processes. Additionally, the ability of DGMs to produce diverse manufacturing sequences could allow the exploration of novel routes, which in turn, could help to search for alternative pathways more appropriate for user's needs.

Subsequently, the best approach obtained for primary processing was adapted for the secondary domain. Here, DGMs were able to output manufacturing procedures with information on unit operations and conditions by feeding the model with information on active ingredients and the desired dosage form. In the end, the models for both modes of manufacturing were successfully trained for the intended task. Nonetheless, several challenges and opportunities for improvement for its applicability in the design of manufacturing processes in real-world emerged during the development process.

Overall, this research represents a step forward in the use of AI, specifically, DGMs in pharmaceutical development and a contribution to advance in industry digital transformation. DGMs enable rapid prototyping of manufacturing routes, which can reduce the time to develop new pharmaceutical processes. In addition, this approach

enhances flexibility as allows to generate and explore alternative procedures that can adapt better to new requirements such as changes in materials or regulations, reducing time-consuming experiments. DGMs in turn can help identify manufacturing routes that are more resource-efficient, potentially decreasing costs and lowering the environmental impact of pharmaceutical production. To encourage wider adoption, DGMs must undergo rigorous validation against experimental data and existing experimental processes to gain user trust. Additionally, developing interfaces and standards that allow integration with available digital infrastructure will be essential to ensure that DGMs can be implemented across the pharmaceutical industry.

The design and development of DGMs in this work can be broken down into several critical steps: 1) data collection, 2) creation of tools to structure a suitable training set, and 3) training of generative models. Looking back to the areas for improvement in each step, and all the techniques employed throughout this thesis, the selection of hyperparameters is an area to explore. This task was found to be challenging due to the amount and the varied effect they have over performance. Although attempts to explore the effect of several hyperparameters in different models were carried out, the exploration of different algorithms of optimisation such as random search or Bayesian-based approaches could aid in the identification of better combinations of hyperparameters to improve overall performance. Finally, in each of these steps, different outcomes were achieved that led to the final models. The conclusions drawn at each stage of this research are summarised below.

Data collection (Chapter 4)

This first part collected patents and analysed their relevancy for pharmaceutical manufacturing. The main source of this information corresponded to United States Patents and Trademark Office (USPTO). The documents searched covered a period between 1976 and 2021. Using several search strategies, more than 3 million patents with potential information were obtained. To establish their relevancy, firstly, patents with cooperative patent classification (CPC) related to pharmaceutical inventions were selected. This information was not reported for all the patents by which a deep learning model was trained to aid in the identification of relevant documents based on abstract. The DL approach achieved a recall and a precision of 82.9% and 88.7%, respectively.

After filtering, a small portion of all the collected patents remained with 208.664 patents, less than 10%.

Using topic modelling over these documents, the content was revised. Based on key words generated by non-negative matrix factorisation (NMF), 5 main topics were identified whose distribution in the patents was as follows: Biological Products (19.2%), Therapeutical Applications (34.1%), Small Molecules (4.7%), Pharmaceutical Compositions (34.0%), and Manufacturing (4.7%). The main topics shed light on the content of the patents and showed their alignment with pharmaceutics. In this manner, for instance, information on secondary manufacturing could be found in patents belonging to Pharmaceutical Compositions; on the other hand, primary processing data, such as synthesis or purification methods, was more related to topics like small molecules or biologics. Finally, trends along the time were established showing, in general, a significant increasing in the number of granted patents over the years, leading to most patents being granted in recent years. The former trend might generate some bias towards recent files. Regarding substances class, most documents had content associated with biological products followed by small molecules.

Overall, a corpus useful for the development of a dataset and different types of models was developed. This was limited to patents whereby the dataset may inherit the typical characteristics of this types of documents such as ambiguity and incompleteness in the information retrieved. Additionally, as could be seen, the portion of relevant documents with respect to search results was rather small, indicating that a substantial amount of information is lost, and more documents are necessary to have bigger dataset relevant for developing models. Future work could explore alternative such as including other sources of patents such as those from European Patent Office (EPO) or other types of documents such as scientific papers. Additionally, substantial efforts have been done by research community in the developments of datasets in the primary domain, finding commercial databases like Reaxys® and Pistachio [222,223222,223]. On the other hand, for secondary processing, to the best of our knowledge, there is not an available corpus or dataset that gathers information on manufacturing methods in a standardised manner, comparable to the primary processing. Thus, this domain could provide a more interesting focus for future developments.

Creation of Tools to Structure a Suitable Training Set (Chapter 5)

This chapter describes the design and development of models necessary to extract the specific information from the obtained patents such as operations, conditions, and materials. The framework proposed for the obtention of information consisted of three steps: section filtration, named entity recognition (NER), and standardisation. The first two steps were the subject of study in this chapter. To filter relevant sections, Latent Dirichlet Allocation (LDA) combined with kMeans was employed. Both were unsupervised approaches whose roles comprised the generation of a numerical representation for each section with LDA to be grouped via kMeans clustering. Additionally, this approach also allowed the summarisation into interpretable keywords to define each cluster topic. The interpretability and the agreement of assigned topics with the actual section was validated manually by revising a sample of sections, resulting in a degree of agreement of 81.7%, in the worst case. Subsequently, for section filtration, topics related to manufacturing were selected. The agreement between relevant and irrelevant assigned by the method and manual review was also assessed through Cohen's kappa, obtaining 91.1% which can be considered a good level of agreement. This approach attained to select patents sections related to pharmaceutical manufacturing for either primary or secondary domain.

Next, a NER model was trained to identify key terms from the relevant sections. This model consisted of a bidirectional long-short term memory (BiLSTM) neural work coupled with conditional random field (CRF) layer. Fasftex algorithm was employed as word embeddings. This method is a supervised approach that was trained on a dataset encompassing more than 7K sentences manually labelled. The overall performance for the NER model, measured using F1-score micro average, was 84.2%. An error analysis showed that errors were occasional, and the model mostly failed at omitting certain entities or confusing types of materials.

The approaches used in this work are characterised by being traditional and well-stablished models which are convenient for relatively small training sets. With more recent developments in natural language processing (NLP), Transformers architecture, which can be said is the main driver of large language models (LLM), have become the state of the art (SOTA) in this field [113]. While the demand for data is high for pretraining architecture of this type, strategies such fine tuning have allowed to reduce the dataset size [113]. Having said this, future work focused on increase accuracy of NER models could fine tune LLM or pre-trained models in similar domains to explore whether there is an

improvement. Additionally, it has been found that domain specific models can perform better. Thus, the development of separate methods for primary and secondary processing might provide better alternative. All these recommendations have been centred on enhancing NER performance. As for topic modelling, LLM have also provided another method to represent not only words, but also blocks of text. LDA, employed for topic modelling, represents text founded on co-occurrence of words whereby there is no direct notion of meaning as it happened with word embeddings. Thus, using deep learning-based approaches to represent section may capture better similarities and differences between texts [224]. Finally, unsupervised modelling was used for section filtering. A change of focus by using supervised approach would be more reliable to validate, although it would imply manual labelling of a sufficiently large number of sections.

Deep Generative Models for Primary Manufacturing (Chapter 6)

Using the framework stablished previously, a training set for primary manufacturing was built. Information on unit operations, conditions, and materials could be extracted in most of the cases. However, limitation in the extraction of materials was the main concerned to ensure the completeness of the dataset. In the end, the dataset contained more than 385K procedures for synthesis and purification of either active ingredients or synthesis intermediates. More than 350K target substances were identified whose chemical space was comparable to FDA active ingredients in terms of properties such as solubility, partition coefficient, and molecular weight. In this regard, most target substances corresponded to small molecules (< 500 Da). In addition, approximately 90% of processes yielded between 14 mg and 20 g of target product, thereby showing that the procedures obtained were mostly at lab-scale. All this could help to establish the expected scope of the generative models, limiting the generated sequences to laboratory procedures for small molecules of pharmaceutical interest.

Subsequently, different types of autoencoders were trained using the training set. All of these were designed to generate sequences conditioned by either only target molecule or the set including target molecule and materials. VAE offered the best results, as indicated by metrics such as reconstruction accuracy and condition importance. AAE and ARAE exhibited greater instability during training and a poorer performance to generate manufacturing sequences. Additionally, variants of VAE such as CVAE and invariant

CVAE were also tested, which offered the best results upon assessing several performance metrics. At this level, apart from reconstruction accuracy and condition importance, a validity criterion for the sequence was also defined. Both autoencoders were based on transformer blocks and demonstrated to be capable of generating manufacturing sequences. The output procedures consisted of chains of unit operations which included details related to addition order and conditions such as time, pH, temperature. The models were conditioned by the set of materials and target substance represented using MACC Keys.

Specifically speaking about the generation process and the quality of the outputs, the models generated logical sequences with a high proportion of valid examples (>70%). The reconstruction accuracy, an indicator of faithfulness, was higher than 95% using BLEU score for both models. On the other hand, variations of approximately 45% and 31% in BLEU score were observed for CVAE and its invariant counterpart by permutating conditional features (set of materials), respectively. This suggests that the generation process is more sensitive to materials and target molecule in CVAE than in the invariant version. Finally, the plausibility of the generated sequences was evaluated through a survey.

In this survey with 100 samples, experts compared real and generated procedures and choose which option was the most suitable to perform in a lab environment. Experts could response the generated, the actual sequence, both, or neither of the options, although the actual and the generated were randomised so that the study was blinded. The model was considered successful if the expert selected both or the generated sequence. On average, the success rate for both models 32% and 40% for invariant CVAE and simple CVAE, respectively. This shows that both models can generate plausible manufacturing procedures, nonetheless, there still room for improvement to make it usable in real life situations. Additionally, while CVAE had a better success rate, the results were lest consistent. A higher variability was obtained in CVAE under different sets of materials. While CVAE yielded a success rate of 12% and 68% for both sets, the invariant counterpart had 28% and 36%. In the end, this behaviour could be explained by the importance that condition had in each model.

Another aspect to highlight is the potential the capacity of generative diverse procedures given the stochastic nature of the models. While this can be useful to explore alternative

routes for product manufacturing, the excess of options, without prioritisation, can be an additional obstacle for model applicability. Thus, process yield estimation was tested as a method of prioritisation, using encoder output as inputs having R^2 lower than 0.30. While this is comparable to other results using patents as main source of data, it might not be sufficiently reliable [188].

Finally, there were several challenges identified to make models more applicable in practical scenarios. Sequence prioritisation and safety concerns should be included. Future approaches to tackle these aspects could explore the usage of complementary methods to predict yield or other parameters appropriate to prioritise processes such as impurities. Similarly, alternatives should be sought to assess safety aspects as a part of the validity criteria of a process. For instance, LLM could be employed to assess whether incompatibilities among materials, and conditions or mixtures which can cause undesirable reactions. All these in order to have a fully automated generator of sequences. In addition, limitations were observed due to missing values for conditions and materials in the dataset, apart from improving methods of extraction as discussed or implementing the models employed in commercial datasets such as Reaxys® or Pinochio to establish a benchmark with respect other approaches for the generation of sequences based on sequence-to-sequence models [43]. Interestingly, in other domain, generative models have been used to style transfer or topic transfer, an idea that could be explored is a method transfer approach so that a known method for a set of materials and target can be adapted for a different set [225]. An additional aspect to explore is the visualisation and interpretability of the latent space. VAE are also known for their applications in dimensionality reduction and clustering. A further exploration of the latent space could shed lights in the types of relationships and patterns that the models is capturing.

Deep Generative Models for Secondary Manufacturing (Chapter 7)

For secondary manufacturing, a dataset was built that contained 9,216 experimental procedures covering lab and pilot scale. It also included 21 types of dosage forms and 2,614 active pharmaceutical ingredients. In this case, APIs chemical properties closely resemble those belonging to FDA-approved molecules in terms of properties such as solubility, partition coefficient, and molecular weight, thereby indicating the dataset can be representative of the chemical space of typical drug products. Furthermore, the dataset also comprised non-sterile drug products with up to 3 APIs. Similarly, information

on unit operations was extracted but with missing data of conditions and materials. In contrast to primary manufacturing dataset, it was not possible to associate materials with the respective operation where they intervene in most cases. In this manner, there were limitations in the interpretability of the output sequences compared to the primary domain.

Then, a CVAE was trained conditioned by both API(s) and dosage form. The API(s) were represented using MACC Keys as with the materials and target molecules in primary manufacturing. The architecture employed was adapted from the best model obtained for primary manufacturing. Given the dataset size was smaller than primary manufacturing dataset, hyperparameters such as batch size, dropout, token dropout, and KL target were adjusted to prevent overfitting. Hyperparameters were chosen for the final model, balancing reconstruction accuracy and feature importance. In the end, CVAE exhibited customisation capability, generating sequences with operations according to the indicated dosage form and assigning conditions to certain operations. The best model had a reconstruction accuracy of 59.1% as BLEU score. In terms of feature importance, API(s) seemed to be more relevant with a variation of 18.5% vs 13.7% variation for the dosage form. Lastly, manual revision of some generated sequences showed inconsistencies. For instance, inappropriate operations for specific dosage forms and uncommon patterns in the sequences. Additionally, missing materials in the output limited readability. However, despite these drawbacks, to the best of our knowledge, contrary to primary manufacturing where approaches to generate procedures have been proposed in recent years, this was not the case for secondary processing. Thus, this study gave insights into the applicability of generative models for assisting secondary manufacturing process design.

Model improvements for this task should point in several directions. Firstly, increase the dataset size and enhance quality of the available data could help to build a more readable and practical representation for manufacturing processes. As mentioned previously, development of methodologies for the extraction of data specific for this domain or the creation of a standardised database which enable to share and harmonise different types of processes for the different dosage forms could facilitate the design and training of models to aid the design of processes for drug products. Secondly, the inclusion of information on excipients, as a conditional, can be key to ensure greater customisation of the output sequences and also a greater interpretability if, for example, bulk properties

of these are incorporated into the modelling. This latter aspect can be relevant as this type of properties usually determine the feasibility of different methods of manufacturing.

Finally, applied to both domain and considering recent developments in deep generative models and deep learning, normalising flows and diffusion models could become alternatives to CVAE. Although these models have not been utilised for the output format used in this work, other representation different to sequences could also be employed such as graphs to represent procedures. For instance, in drug discovery, the use of molecular graphs with diffusion models have become more common [56]. Despite the fact that the usage of graphs is more computationally expensive, it seems that this representation can improve performance. This has been shown in tasks such as solubility prediction using graph neural networks [193]. Going into topics more specific to deep learning, the building blocks of most neural architectures has traditionally been the multilayer perceptron (MLP). Nonetheless, a recent development to replace MLP is Kolmogorov Arnold Networks (KAN) [226]. This type of networks has been claimed to outperform MLP [226]. The application and incorporation of these new architectures might improve generative modelling performance and generalisability, in this manner more research in this field is necessary to confirm the impact of these new approaches on manufacturing sequence generation.

References

1. Destro, F. & Barolo, M. A review on the modernization of pharmaceutical development and manufacturing – Trends, perspectives, and the role of mathematical modeling. *Int J Pharm* **620**, 121715 (2022).
2. Aitken, M. Understanding the pharmaceutical value chain. *Pharmaceuticals Policy and Law* **18**, 55–66 (2016).
3. Am Ende, D. J. & Am Ende, M. T. Chemical engineering in the pharmaceutical industry. *Chemical Engineering in the Pharmaceutical Industry* 1–644 (2019) doi:10.1002/9781119600800.
4. Taylor, D. The pharmaceutical industry and the future of drug development. *Issues in Environmental Science and Technology* vols 2016-Janua 1–33 Preprint at https://doi.org/10.1039/9781782622345-00001 (2016).
5. Shah, N. Pharmaceutical supply chains: key issues and strategies for optimisation. *Comput Chem Eng* **28**, 929–941 (2004).
6. Schuhmacher, A., Hinder, M., von Stegmann und Stein, A., Hartl, D. & Gassmann, O. Analysis of pharma R&D productivity – a new perspective needed. *Drug Discov Today* **28**, 103726 (2023).
7. de la Torre, B. G. & Albericio, F. The Pharmaceutical Industry in 2022: An Analysis of FDA Drug Approvals from the Perspective of Molecules. *Molecules 2023, Vol. 28, Page 1038* **28**, 1038 (2023).
8. Seeds of change Measuring the return from pharmaceutical innovation 2020 Contents.
9. Berndt, E. R., Nass, D., Kleinrock, M. & Aitken, M. Decline In Economic Returns From New Drugs Raises Questions About Sustaining Innovations. *https://doi.org/10.1377/hlthaff.2014.1029* **34**, 245–252 (2017).
10. Ward, D. J., Martino, O. I., Simpson, S. & Stevens, A. J. Decline in new drug launches: myth or reality? Retrospective observational study using 30 years of data from the UK. *BMJ Open* **3**, (2013).

11. Paul, S. M. *et al.* How to improve R&D productivity: the pharmaceutical industry's grand challenge. *Nature Reviews Drug Discovery 2010 9:3* **9**, 203–214 (2010).

12. Aitken, M. Understanding the pharmaceutical value chain. *Pharmaceuticals Policy and Law* **18**, 55–66 (2016).

13. Sarkis, M., Bernardi, A., Shah, N. & Papathanasiou, M. M. Manufacturing and Distribution. (2021).

14. Pollak, Peter; Badrot, Andrew; Dach, R. API Manufacturing: Facts And Fiction - Contract Pharma. https://www.contractpharma.com/issues/ 2012-01/view_features/api-manufacturing-facts-and-fiction/.

15. Arden, N. S. *et al.* Industry 4.0 for pharmaceutical manufacturing: Preparing for the smart factories of the future. *Int J Pharm* **602**, 120554 (2021).

16. Zagalo, D. M., Sousa, J. & Simões, S. Quality by design (QbD) approach in marketing authorization procedures of Non-Biological Complex Drugs: A critical evaluation. *European Journal of Pharmaceutics and Biopharmaceutics* **178**, 1–24 (2022).

17. Grangeia, H. B., Silva, C., Simões, S. P. & Reis, M. S. Quality by design in pharmaceutical manufacturing: A systematic review of current status, challenges and future perspectives. *European Journal of Pharmaceutics and Biopharmaceutics* **147**, 19–37 (2020).

18. Bian, Y. & Xie, X.-Q. *Generative Chemistry: Drug Discovery with Deep Learning Generative Models*.

19. Yang, Y. *et al.* Deep learning for in vitro prediction of pharmaceutical formulations. doi:10.1016/j.apsb.2018.09.010.

20. Ruthotto, L. & Haber, E. An introduction to deep generative modeling. *GAMM Mitteilungen* **44**, (2021).

21. Aggarwal, A., Mittal, M. & Battineni, G. Generative adversarial network: An overview of theory and applications. *International Journal of Information Management Data Insights* **1**, 100004 (2021).

22. Meyers, J., Fabian, B. & Brown, N. De novo molecular design and generative models. *Drug Discov Today* (2021) doi:10.1016/J.DRUDIS.2021.05.019.

23. Woodley, J. M. Innovative process development and production concepts for small-molecule API manufacturing. *Computer Aided Chemical Engineering* **41**, 67–84 (2018).

24. Porta, R., Benaglia, M. & Puglisi, A. Flow Chemistry: Recent Developments in the Synthesis of Pharmaceutical Products. *Org Process Res Dev* **20**, 2–25 (2015).

25. Adler, C. *et al.* Process Development for Active Pharmaceutical Ingredients Following a Developmental Cascade. *Chimia (Aarau)* **60**, 523–529 (2006).

26. Federsel, H.-J. Chemical Process Research and Development in the 21st Century: Challenges, Strategies, and Solutions from a Pharmaceutical Industry Perspective. *Acc Chem Res* **42**, 671–680 (2009).

27. ICH. INTERNATIONAL CONFERENCE ON HARMONISATION OF TECHNICAL REQUIREMENTS FOR REGISTRATION OF PHARMACEUTICALS FOR HUMAN USE PHARMACEUTICAL DEVELOPMENT Q8(R2). (2009).

28. ICH. Guidance for Industry Container Closure Systems for Packaging Human Drugs and Biologics CHEMISTRY, MANUFACTURING, AND CONTROLS DOCUMENTATION. **20857**, 301–827 (1999).

29. Mittal, B. Formulation Development. *How to Develop Robust Solid Oral Dosage Forms from Conception to Post-Approval* 39–67 (2017) doi:10.1016/b978-0-12-804731-6.00003-0.

30. Gad, S. C. *Pharmaceutical Manufacturing Handbook: Production and Processes*. *Pharmaceutical Manufacturing Handbook: Production and Processes* (2007). doi:10.1002/9780470259818.

31. Osakwe, O. *Pharmaceutical Formulation and Manufacturing Development: Strategies and Issues*. *Social Aspects of Drug Discovery, Development and Commercialization* (Elsevier Inc., 2016). doi:10.1016/b978-0-12-802220-7.00008-9.

32. Mittal, B. Pharmaceutical Unit Operations. in *How to Develop Robust Solid Oral Dosage Forms from Conception to Post-Approval* 69–95 (2017). doi:10.1016/b978-0-12-804731-6.00004-2.

33. Mittal, B. Process Scale-up, Tech-Transfer, and Optimization. *How to Develop Robust Solid Oral Dosage Forms from Conception to Post-Approval* 137–153 (2017) doi:10.1016/b978-0-12-804731-6.00007-8.

34. Arden, N. S. *et al.* Industry 4.0 for pharmaceutical manufacturing: Preparing for the smart factories of the future. *Int J Pharm* **602**, 120554 (2021).

35. FDA. *Using Artificial Intelligence & Machine Learning in the Development of Drug and Biological Products*. https://www.imdrf.org/documents/machine-learning-enabled-medical- (2023).

36. Ema. Committee for Medicinal Products for Human Use (CHMP) 3 Committee for Medicinal Products for Veterinary Use (CVMP) 4 Reflection paper on the use of Artificial Intelligence (AI) in 5 the medicinal product lifecycle 6 Draft 7 Draft agreed by Committee for Medicinal Products for Human Use (CHMP) Methodology Working Party. (2023).

37. Liu, Q. *et al.* Landscape Analysis of the Application of Artificial Intelligence and Machine Learning in Regulatory Submissions for Drug Development From 2016 to 2021. *Clin Pharmacol Ther* **113**, 771–774 (2023).

38. Murphy, K. P. *Machine Learning : A Probabilistic Perspective*. 1067 (2012).

39. Li, Q.-Y. *et al.* Machine Learning: A New Approach for Dose Individualization. *Clin Pharmacol Ther* (2023) doi:10.1002/CPT.3049.

40. Cavasotto, C. N. & Scardino, V. Machine Learning Toxicity Prediction: Latest Advances by Toxicity End Point. *ACS Omega* **7**, 47536–47546 (2022).

41. Nagy, B. *et al.* Application of Artificial Neural Networks in the Process Analytical Technology of Pharmaceutical Manufacturing-a Review. *AAPS J* **24**, 3 (2022).

42. Ishida, S., Terayama, K., Kojima, R., Takasu, K. & Okuno, Y. AI-Driven Synthetic Route Design Incorporated with Retrosynthesis Knowledge. *Cite This: J. Chem. Inf. Model* **2022**, 1357–1367 (2022).

43. Vaucher, A. C. *et al.* Inferring experimental procedures from text-based representations of chemical reactions. *Nat Commun* **12**, 2573 (2021).

44. Kim, E. J., Kim, J. H., Kim, M. S., Jeong, S. H. & Choi, D. H. Process Analytical Technology Tools for Monitoring Pharmaceutical Unit Operations: A Control Strategy for Continuous Process Verification. *Pharmaceutics 2021, Vol. 13, Page 919* **13**, 919 (2021).

45. Cacciarelli, D. & Kulahci, M. A novel fault detection and diagnosis approach based on orthogonal autoencoders. *Comput Chem Eng* **163**, 107853 (2022).

46. Forbes, T. P., Gillen, J. G., Souna, A. J. & Lawrence, J. Unsupervised Pharmaceutical Polymorph Identification and Multicomponent Particle Mapping of ToF-SIMS Data by Non-Negative Matrix Factorization. *Anal Chem* **94**, 16443–16450 (2022).

47. Harshvardhan, G., Gourisaria, M. K., Pandey, M. & Rautaray, S. S. A comprehensive survey and analysis of generative models in machine learning. *Comput Sci Rev* **38**, 100285 (2020).

48. Samanta, B. *et al. NEVAE: A Deep Generative Model for Molecular Graphs *. Journal of Machine Learning Research* vol. 21 (2020).

49. Joo, S., Kim, M. S., Yang, J. & Park, J. Generative Model for Proposing Drug Candidates Satisfying Anticancer Properties Using a Conditional Variational Autoencoder. *ACS Omega* **5**, 18642–18650 (2020).

50. Lin, E., Lin, C.-H. & Lane, H.-Y. Relevant Applications of Generative Adversarial Networks in Drug Design and Discovery: Molecular De Novo Design, Dimensionality Reduction, and De Novo Peptide and Protein Design. *Molecules* **25**, (2020).

51. Tomczak, J. M. Why Deep Generative Modeling? *Deep Generative Modeling* 1–12 (2022) doi:10.1007/978-3-030-93158-2_1.

52. Openai, I. G. NIPS 2016 Tutorial: Generative Adversarial Networks.

53. Bian, Y. & Xie, X.-Q. Generative chemistry: drug discovery with deep learning generative models. (2020).

54. Bilodeau, C., Jin, W., Jaakkola, T., Barzilay, R. & Jensen, K. F. Generative models for molecular discovery: Recent advances and challenges. *WIREs Computational Molecular Science* **12**, (2022).

55. Zang, C. & Wang, F. MoFlow: An Invertible Flow Model for Generating Molecular Graphs. in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* vol. 10 617–626 (ACM, New York, NY, USA, 2020).

56. Xu, M., Powers, A., Dror, R., Ermon, S. & Leskovec, J. Geometric Latent Diffusion Models for 3D Molecule Generation. (2023).

57. Grisoni, F., Moret, M., Lingwood, R. & Schneider, G. Bidirectional Molecule Generation with Recurrent Neural Networks. *J Chem Inf Model* **60**, 1175–1183 (2020).

58. Noh, J. *et al.* Inverse Design of Solid-State Materials via a Continuous Representation. (2019) doi:10.1016/j.matt.2019.08.017.

59. Kim, S., Noh, J., Ho Gu, G., Aspuru-Guzik, A. & Jung, Y. Generative Adversarial Networks for Crystal Structure Prediction. **22**, 3 (2023).

60. Tempke, R. & Musho, T. Autonomous design of new chemical reactions using a variational autoencoder. *Communications Chemistry 2022 5:1* **5**, 1–10 (2022).

61. Li, S., Wang, X., Wu, Y., Duan, H. & Tang, L. Generation of novel Diels–Alder reactions using a generative adversarial network. *RSC Adv* **12**, 33801–33807 (2022).

62. Bort, W. *et al.* Discovery of novel chemical reactions by deep generative recurrent neural network. *Scientific Reports 2021 11:1* **11**, 1–15 (2021).

63. Bond-Taylor, S., Leach, A., Long, Y. & Willcocks, C. G. Deep Generative Modelling: A Comparative Review of VAEs, GANs, Normalizing Flows, Energy-Based and Autoregressive Models. (2021) doi:10.1109/TPAMI.2021.3116668.

64. Ian Goodfellow, Yoshua Bengio, and A. C. Deep Learning. *Genet Program Evolvable Mach* **19**, 305–307 (2017).

65. Kingma Google, D. P., Welling, M. & -Delft, B. An Introduction to Variational Autoencoders. *Foundations and Trends R in Machine Learning* **xx, No. xx**, 1–18 (2019).

66. Ruthotto, L. & Haber, E. An introduction to deep generative modeling. *GAMM Mitteilungen* **44**, (2021).

67. Goodfellow, I. *et al.* Generative Adversarial Networks. *Commun ACM* **63**, (2020).

68. Lin, E., Lin, C.-H. H. & Lane, H.-Y. Y. *Relevant Applications of Generative Adversarial Networks in Drug Design and Discovery: Molecular de Novo Design, Dimensionality Reduction, and de Novo Peptide and Protein Design*. *Molecules* vol. 25 3250 (MDPI AG, 2020).

69. Alom, M. Z. *et al.* A State-of-the-Art Survey on Deep Learning Theory and Architectures. *Electronics 2019, Vol. 8, Page 292* **8**, 292 (2019).

70. Arjovsky, M., Chintala, S. & Bottou, L. Wasserstein GAN.

71. Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V. & Courville, A. Improved Training of Wasserstein GANs Montreal Institute for Learning Algorithms.

72. Sarkar, D. *Text Analytics with Python*. *Text Analytics with Python* (2019). doi:10.1007/978-1-4842-4354-1.

73. Cardie, C. Empirical Methods in Information Extraction. *AI Mag* **18**, 65–65 (1997).

74. Balducci, B. & Marinova, D. Unstructured data in marketing. *J Acad Mark Sci* **46**, 557–590 (2018).

75. Churchill, R. & Singh, L. The Evolution of Topic Modeling. (2022) doi:10.1145/3507900.

76. Chauhan, U. & Shah, A. Topic Modeling Using Latent Dirichlet allocation. *ACM Computing Surveys (CSUR)* **54**, (2021).

77. Rüdiger, M., Antons, D., Joshi, A. M. & Salge, T. O. Topic modeling revisited: New evidence on algorithm performance and quality metrics. *PLoS One* **17**, e0266325 (2022).

78. Gillis, N. The Why and How of Nonnegative Matrix Factorization. (2014).

79. Davies, D. L. & Bouldin, D. W. A Cluster Separation Measure. *IEEE Trans Pattern Anal Mach Intell* **PAMI-1**, 224–227 (1979).

80. Blei, D. M., Ng, A. Y. & Edu, J. B. *Latent Dirichlet Allocation Michael I. Jordan*. *Journal of Machine Learning Research* vol. 3 (2003).

81. Bui, Q. V., Sayadi, K., Amor, S. ben & Bui, M. Combining Latent Dirichlet Allocation and K-Means for Documents Clustering: Effect of Probabilistic Based Distance Measures. in (eds. Nguyen, N. T., Tojo, S., Nguyen, L. M. & Trawiński, B.) vol. 10191 248–257 (Springer International Publishing, Cham, 2017).

82. Al-Garadi, M. A. *et al.* Text classification models for the automatic detection of nonmedical prescription medication use from social media. *BMC Med Inform Decis Mak* **21**, 1–13 (2021).

83. Yao, L., Mao, C. & Luo, Y. Clinical text classification with rule-based features and knowledge-guided convolutional neural networks. *BMC Med Inform Decis Mak* **19**, 71 (2019).

84. Tran, T. & Kavuluru, R. Supervised approaches to assign cooperative patent classification (CPC) codes to patents. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* **10682 LNAI**, 22–34 (2017).

85. Abdelgawad, L., Kluegl, P., Genc, E., Falkner, S. & Hutter, F. Optimizing Neural Networks for Patent Classification.

86. Kowsari, K. *et al.* Text classification algorithms: A survey. *Information (Switzerland)* **10**, (2019).

87. Martinčić-Ipšić, S., Miličić, T. & Todorovski, L. The Influence of Feature Representation of Text on the Performance of Document Classification. *Applied Sciences 2019, Vol. 9, Page 743* **9**, 743 (2019).

88. Mikolov, T., Chen, K., Corrado, G. & Dean, J. Efficient Estimation of Word Representations in Vector Space. *1st International Conference on Learning Representations, ICLR 2013 - Workshop Track Proceedings* (2013) doi:10.48550/arxiv.1301.3781.

89. Li, Q. *et al.* A Survey on Text Classification: From Traditional to Deep Learning. *ACM Trans Intell Syst Technol* **13**, 1–41 (2022).

90. Li, J., Sun, A., Han, J. & Li, C. *A Survey on Deep Learning for Named Entity Recognition*. http://neuroner.com/ (2020).

91. Perera, N., Dehmer, M. & Emmert-Streib, F. Named Entity Recognition and Relation Detection for Biomedical Information Extraction. *Frontiers in Cell and Developmental Biology* vol. 8 673 Preprint at https://doi.org/10.3389/fcell.2020.00673 (2020).

92. Zhang, Y. *et al.* Chemical named entity recognition in patents by domain knowledge and unsupervised feature learning. *Database* **2016**, 49 (2016).

93. Trewartha, A. *et al.* Quantifying the advantage of domain-specific pre-training on named entity recognition tasks in materials science. *Patterns* **3**, 100488 (2022).

94. He, J. *et al.* ChEMU 2020: Natural Language Processing Methods Are Effective for Information Extraction From Chemical Patents. *Front Res Metr Anal* **0**, 12 (2021).

95. Vaucher, A. C. *et al.* Automated extraction of chemical synthesis actions from experimental procedures. *Nature Communications 2020 11:1* **11**, 1–11 (2020).

96. Daniel, J. & Martin, J. H. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. (2020).

97. Honnibal, M. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. Preprint at https://spacy.io/ (2017).

98. Wishart, D. S. *et al.* DrugBank 5.0: a major update to the DrugBank database for 2018. *Nucleic Acids Res* **46**, D1074–D1082 (2018).

99. Kim, S. *et al.* PubChem 2023 update. *Nucleic Acids Res* **51**, D1373–D1380 (2023).

100. Mendez, D. *et al.* ChEMBL: towards direct deposition of bioassay data. *Nucleic Acids Res* **47**, D930–D940 (2019).

101. Heller, S. R., McNaught, A., Pletnev, I., Stein, S. & Tchekhovskoi, D. InChI, the IUPAC International Chemical Identifier. *J Cheminform* **7**, 1–34 (2015).

102. Peryea, T. *et al.* Global Substance Registration System: consistent scientific descriptions for substances related to health. *Nucleic Acids Res* **49**, D1179–D1185 (2021).

103. Wang, Z. *et al.* ULSA: unified language of synthesis actions for the representation of inorganic synthesis protocols. *Digital Discovery* **1**, 313–324 (2022).

104. Zeiler, M. D. & Fergus, R. Visualizing and Understanding Convolutional Networks. in 818–833 (2014). doi:10.1007/978-3-319-10590-1_53.

105. Bartlett, P. L., Montanari, A. & Rakhlin, A. *Deep Learning: A Statistical Viewpoint*. (2021).

106. Ruder, S. An overview of gradient descent optimization algorithms. (2016).

107. RNN, LSTM & GRU. http://dprogrammer.org/rnn-lstm-gru.

108. Elman, J. L. Finding structure in time. *Cogn Sci* **14**, 179–211 (1990).

109. Hochreiter, S. & Schmidhuber, J. Long Short-Term Memory. *Neural Comput* **9**, 1735–1780 (1997).

110. Chung, J., Gulcehre, C., Cho, K. & Bengio, Y. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. (2014).

111. Yang, S., Yu, X. & Zhou, Y. LSTM and GRU Neural Network Performance Comparison Study: Taking Yelp Review Dataset as an Example. in *2020 International Workshop on Electronic Communication and Artificial Intelligence (IWECAI)* 98–101 (IEEE, 2020). doi:10.1109/IWECAI50956.2020.00027.

112. Vaswani, A. *et al.* Attention Is All You Need. (2017).

113. Zhao, W. X. *et al.* A Survey of Large Language Models. (2023).

114. Sousa, T., Correia, J., Pereira, V. & Rocha, M. Generative Deep Learning for Targeted Compound Design. *J Chem Inf Model* **61**, 5343–5361 (2021).

115. Bilodeau, C., Jin, W., Jaakkola, T., Barzilay, R. & Jensen, K. F. Generative models for molecular discovery: Recent advances and challenges. *Wiley Interdiscip Rev Comput Mol Sci* **12**, e1608 (2022).

116. Gu, X., Cho, K., Ha, J.-W. & Kim, S. DialogWAE: Multimodal Response Generation with Conditional Wasserstein Auto-Encoder. (2018).

117. Nie, W., Narodytska, N. & Patel, A. B. RELGAN: RELATIONAL GENERATIVE ADVERSARIAL NETWORKS FOR TEXT GENERATION.

118. Li, P., Pei, Y. & Li, J. A comprehensive survey on design and application of autoencoder in deep learning. *Appl Soft Comput* **138**, 110176 (2023).

119. Zhao, J., Kim, Y., Zhang, K., Rush, A. M. & LeCun, Y. Adversarially Regularized Autoencoders. *35th International Conference on Machine Learning, ICML 2018* **13**, 9405–9420 (2017).

120. Makhzani, A., Shlens, J., Jaitly, N., Goodfellow, I. & Frey, B. Adversarial Autoencoders. (2015).

121. Ghojogh BGHOJOGH, B., Ghodsi ALIGHODSI, A., Karray KARRAY, F. & Crowley MCROWLEY, M. Generative Adversarial Networks and Adversarial Autoencoders: Tutorial and Survey.

122. Dai, B., Wang, Z. & Wipf, D. The Usual Suspects? Reassessing Blame for VAE Posterior Collapse. (2019).

123. Bowman, S. R. *et al.* Generating Sentences from a Continuous Space. (2015).

124. Yan, C., Wang, S., Yang, J., Xu, T. & Huang, J. Re-balancing Variational Autoencoder Loss for Molecule Sequence Generation. in *Proceedings of the 11th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics* vol. 20 1–7 (ACM, New York, NY, USA, 2020).

125. Higgins, I. *et al.* beta-vae: Learning basic visual concepts with a constrained variational framework. *International conference on learning representations* (2016).

126. Richards, R. J. & Groener, A. M. Conditional beta-VAE for De Novo Molecular Generation. (2022).

127. Dollar, O., Joshi, N., Beck, D. A. C. & Pfaendtner, J. Attention-based generative models for de novo molecular design. *Chem Sci* **12**, 8362–8372 (2021).

128. Papineni, K., Roukos, S., Ward, T. & Zhu, W.-J. BLEU. in *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics - ACL '02* 311 (Association for Computational Linguistics, Morristown, NJ, USA, 2001). doi:10.3115/1073083.1073135.

129. Berger, B., Waterman, M. S. & Yu, Y. W. Levenshtein Distance, Sequence Comparison and Biological Database Search. *IEEE Trans Inf Theory* **67**, 3287 (2021).

130. Buitinck, L. *et al.* API design for machine learning software: experiences from the scikit-learn project. (2013).

131. Řehůřek, R. & Sojka, P. Software framework for topic modelling with large corpora. (2010).

132. Abadi, M. *et al.* TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. (2016).

133. Swain, M. C. & Cole, J. M. ChemDataExtractor: A Toolkit for Automated Extraction of Chemical Information from the Scientific Literature. *J Chem Inf Model* **56**, 1894–1904 (2016).

134. Adnan, K., Akbar, R., Khor, S. W. & Ali, A. B. A. Role and Challenges of Unstructured Big Data in Healthcare. *Advances in Intelligent Systems and Computing* **1042**, 301–323 (2020).

135. Žagar, J. & Mihelič, J. Big data collection in pharmaceutical manufacturing and its use for product quality predictions. *Scientific Data 2022 9:1* **9**, 1–11 (2022).

136. Alemayehu, D. & Berger, M. L. Big Data: transforming drug development and health policy decision making. *Health Services and Outcomes Research Methodology 2016 16:3* **16**, 92–102 (2016).

137. Huang, C. C. & Lu, Z. Community challenges in biomedical text mining over 10 years: success, failure and the future. *Brief Bioinform* **17**, 132–144 (2016).

138. Hardin, S. Text and Data Mining Meets the Pharmaceutical Industry: Markus Bundschus Speaks. *Bulletin of the Association for Information Science and Technology* **43**, 42–44 (2017).

139. Ayvaz, S. *et al.* Toward a complete dataset of drug-drug interaction information from publicly available sources. *J Biomed Inform* **55**, 206 (2015).

140. Harpaz, R. *et al.* Text Mining for Adverse Drug Events: the Promise, Challenges, and State of the Art. *Drug safety : an international journal of medical toxicology and drug experience* **37**, 777 (2014).

141. Stefanowitsch, A. Corpus linguistics A guide to the methodology.

142. Habibi, M., Wiegandt, D. L., Schmedding, F. & Leser, U. Recognizing chemicals in patents: a comparative analysis. *J Cheminform* **8**, 1–15 (2016).

143. Wishart, D. S. *et al.* DrugBank 5.0: a major update to the DrugBank database for 2018. *Nucleic Acids Res* **46**, D1074–D1082 (2018).

144. Dosage Forms | FDA. https://www.fda.gov/industry/structured-product-labeling-resources/dosage-forms.

145. States Patent, U. & Office, T. MPEP - Chapter 0900 - Prior Art, Classification, and Search. (2019).

146. Sarkar, D. *Text Analytics with Python*. (Apress, Berkeley, CA, 2019). doi:10.1007/978-1-4842-4354-1.

147. Kononova, O. *et al.* Opportunities and challenges of text mining in materials research. *iScience* **24**, 102155 (2021).

148. Natural Language Processing. in *SAGE Research Methods Foundations* (SAGE Publications Ltd, 1 Oliver's Yard, 55 City Road, London EC1Y 1SP United Kingdom , 2020). doi:10.4135/9781526421036879118.

149. Lau, J. H. & Baldwin, T. An Empirical Evaluation of doc2vec with Practical Insights into Document Embedding Generation. 78–86 (2016).

150. Dynomant, E. *et al.* Doc2Vec on the PubMed corpus: study of a new approach to generate related articles. (2019).

151. Risch, J. & Krestel, R. Domain-specific word embeddings for patent classification. *Data Technologies and Applications* **53**, 108–122 (2019).

152. Bojanowski, P., Grave, E., Joulin, A. & Mikolov, T. Enriching Word Vectors with Subword Information. *Trans Assoc Comput Linguist* **5**, 135–146 (2016).

153. Le, Q. v. & Mikolov, T. Distributed Representations of Sentences and Documents. (2014).

154. Noble, W. S. What is a support vector machine? *Nature Biotechnology 2006 24:12* **24**, 1565–1567 (2006).

155. Mishu, S. Z. & Rafiuddin, S. M. Performance analysis of supervised machine learning algorithms for text classification. *19th International Conference on Computer and Information Technology, ICCIT 2016* 409–413 (2017) doi:10.1109/ICCITECHN.2016.7860233.

156. Bansal, Ashish. ADVANCED NATURAL LANGUAGE PROCESSING WITH TENSORFLOW 2 build real-world effective nlp... applications using ner, rnns, seq2seq models, tran. (2021).

157. Text classification with an RNN | TensorFlow. https://www.tensorflow.org/text/tutorials/text_classification_rnn.

158. Manning, C. D., Raghavan, P. & Schütze, H. Introduction to Information Retrieval. *Introduction to Information Retrieval* (2008) doi:10.1017/CBO9780511809071.

159. Saito, T. & Rehmsmeier, M. The Precision-Recall Plot Is More Informative than the ROC Plot When Evaluating Binary Classifiers on Imbalanced Datasets. *PLoS One* **10**, (2015).

160. Salman, S. & Liu, X. Overfitting Mechanism and Avoidance in Deep Neural Networks. (2019) doi:10.48550/arxiv.1901.06566.

161. Ouellette, L. L. How Many Patents Does It Take To Make a Drug? Follow-On Pharmaceutical Patents and University Licensing, 17 Mich. *Telecomm. Tech. L. Rev* **299**, (2010).

162. Jones, N. & Dean, A. B. Current patenting trends for biologics versus small molecules. *http://dx.doi.org/10.4155/ppa.12.34* **1**, 225–227 (2012).

163. Makurvet, F. D. Biologics vs. small molecules: Drug costs and patient access. *Med Drug Discov* **9**, 100075 (2021).

164. de la Torre, B. G. & Albericio, F. The Pharmaceutical Industry in 2021. An Analysis of FDA Drug Approvals from the Perspective of Molecules. *Molecules* **27**, (2022).

165. Olivetti, E. A. *et al.* Data-driven materials research enabled by natural language processing and information extraction. *Appl Phys Rev* **7**, 041317 (2020).

166. Kononova, O. *et al.* Text-mined dataset of inorganic materials synthesis recipes. *Sci Data* **6**, 1–11 (2019).

167. Lowe, D. M. Extraction of chemical structures and reactions from the literature. (2012) doi:10.17863/CAM.16293.

168. Huang, M. S. *et al.* Biomedical named entity recognition and linking datasets: survey and our recent development. *Brief Bioinform* **21**, 2219–2238 (2020).

169. Leaman, R., Wei, C. H. & Lu, Z. TmChem: A high performance approach for chemical named entity recognition and normalization. *J Cheminform* **7**, S3 (2015).

170. Luo, L. *et al.* Data and text mining An attention-based BiLSTM-CRF approach to document-level chemical named entity recognition. (2017) doi:10.1093/bioinformatics/btx761.

171. Giorgi, J. M. & Bader, G. D. Towards reliable named entity recognition in the biomedical domain. doi:10.1093/bioinformatics/xxxxxx.

172. Saad, F., Aras, H. & Hackl-Sommer, R. Improving named entity recognition for biomedical and patent data using bi-lstm deep neural network models. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* **12089 LNCS**, 25–36 (2020).

173. Introduction, A. & Retrieval, I. *Introduction to Information Retrieval*. *Choice Reviews Online* vol. 46 (2009).

174. Salgado, C. M., Azevedo, C., Proença, H. & Vieira, S. M. Noise versus outliers. *Secondary Analysis of Electronic Health Records* 163–183 (2016) doi:10.1007/978-3-319-43742-2_14/TABLES/5.

175. Sculley, D. Web-scale k-means clustering. in *Proceedings of the 19th international conference on World wide web - WWW '10* 1177 (ACM Press, New York, New York, USA, 2010). doi:10.1145/1772690.1772862.

176. Davies, D. L. & Bouldin, D. W. A Cluster Separation Measure. *IEEE Trans Pattern Anal Mach Intell* **PAMI-1**, 224–227 (1979).

177. Rousseeuw, P. J. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *J Comput Appl Math* **20**, 53–65 (1987).

178. Pedregosa, F. *et al.* Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* **12**, 2825–2830 (2012).

179. Yu, H., Inoue, K., Hara, K. & Urahama, K. A Robust K-Means for Document Clustering. *Journal of the Institute of Industrial Applications Engineers* **6**, 60–65 (2018).

180. Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K. & Dyer, C. *Neural Architectures for Named Entity Recognition*.

181. Luo, L. *et al.* An attention-based BiLSTM-CRF approach to document-level chemical named entity recognition. *Bioinformatics* **34**, 1381–1388 (2018).

182. Agrawal, A., Fu, W. & Menzies, T. What is wrong with topic modeling? And how to fix it using search-based software engineering. *Inf Softw Technol* **98**, 74–88 (2018).

183. McHugh, M. L. Interrater reliability: the kappa statistic. *Biochem Med (Zagreb)* **22**, 276 (2012).

184. Srivastava, N., Hinton, G., Krizhevsky, A. & Salakhutdinov, R. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research* **15**, 1929–1958 (2014).

185. Jacovi, A., Shalom, O. S. & Goldberg, Y. Understanding Convolutional Neural Networks for Text Classification. *EMNLP 2018 - 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP, Proceedings of the 1st Workshop* 56–65 (2018) doi:10.48550/arxiv.1809.08037.

186. Zhu, Y. *et al.* A Survey on Deep Graph Generation: Methods and Applications. (2022) doi:10.48550/arxiv.2203.06714.

187. Schwaller, P. *et al.* Molecular Transformer: A Model for Uncertainty-Calibrated Chemical Reaction Prediction. *ACS Cent Sci* **5**, 1572–1583 (2019).

188. Schwaller, P., Vaucher, A. C., Laino, T. & Reymond, J.-L. Prediction of chemical reaction yields using deep learning. *Mach Learn Sci Technol* **2**, 015016 (2021).

189. Gao, H. *et al.* Using Machine Learning To Predict Suitable Conditions for Organic Reactions. (2018) doi:10.1021/acscentsci.8b00357.

190. Karpovich, C., Jensen, Z., Venugopal, V. & Olivetti, E. Inorganic Synthesis Reaction Condition Prediction with Generative Machine Learning. (2021).

191. Kwon, Y., Kim, S., Choi, Y. S. & Kang, S. Generative Modeling to Predict Multiple Suitable Conditions for Chemical Reactions. *J Chem Inf Model* **62**, 5952–5960 (2022).

192. Capecchi, A., Probst, D. & Reymond, J. L. One molecular fingerprint to rule them all: Drugs, biomolecules, and the metabolome. *J Cheminform* **12**, 43 (2020).

193. Lee, S. *et al.* Novel Solubility Prediction Models: Molecular Fingerprints and Physicochemical Features vs Graph Convolutional Neural Networks. **7**, 12268–12277 (2022).

194. Lim, J., Ryu, S., Kim, J. W. & Kim, W. Y. Molecular generative model based on conditional variational autoencoder for de novo molecular design. *Journal of Cheminformatics 2018 10:1* **10**, 1–9 (2018).

195. Xue, L. & Bajorath, J. Molecular descriptors in chemoinformatics, computational combinatorial chemistry, and virtual screening. *Comb Chem High Throughput Screen* **3**, 363–372 (2000).

196. Lunnon, W. F., Brunvoll, J., Cyvin, S. J., Cyvin, B. N. & Balaban, A. T. SMILES, a Chemical Language and Information System: 1: Introduction to Methodology and Encoding Rules. *J Chem Inf Comput Sci* **28**, 31–36 (1988).

197. Durant, J. L., Leland, B. A., Henry, D. R. & Nourse, J. G. Reoptimization of MDL keys for use in drug discovery. *J Chem Inf Comput Sci* **42**, 1273–1280 (2002).

198. Au, Q., Herbinger, J., Stachl, C., Bischl, B. & Casalicchio, G. Grouped Feature Importance and Combined Features Effect Plot. (2021).

199. Lee, J. *et al.* Set Transformer: A Framework for Attention-based Permutation-Invariant Neural Networks. (2018).

200. Moyer, D., Gao, S., Brekelmans, R., Steeg, G. Ver & Galstyan, A. Invariant Representations without Adversarial Training. *Adv Neural Inf Process Syst* **2018-December**, 9084–9093 (2018).

201. Shao, H. *et al.* ControlVAE: Tuning, Analytical Properties, and Performance Analysis. *IEEE Trans Pattern Anal Mach Intell* (2020).

202. Devlin, J., Chang, M.-W., Lee, K. & Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference* **1**, 4171–4186 (2018).

203. Sterling, T. & Irwin, J. J. ZINC 15 – Ligand Discovery for Everyone. *J Chem Inf Model* **55**, 2324–2337 (2015).

204. Bickerton, G. R., Paolini, G. V, Besnard, J., Muresan, S. & Hopkins, A. L. Quantifying the chemical beauty of drugs. (2012) doi:10.1038/NCHEM.1243.

205. RDKit. https://www.rdkit.org/ doi:doi.org/10.5281/zenodo.591637.

206. Francoeur, P. G. & Koes, D. R. SolTranNet–A Machine Learning Tool for Fast Aqueous Solubility Prediction. *J Chem Inf Model* **61**, 2530–2536 (2021).

207. Tevosyan, A. *et al.* Improving VAE based molecular representations for compound property prediction. *J Cheminform* **14**, 1–14 (2022).

208. Gómez-Bombarelli, R. *et al.* Automatic Chemical Design Using a Data-Driven Continuous Representation of Molecules. *ACS Cent Sci* **4**, 268–276 (2018).

209. Mathieu, E., Rainforth, T., Siddharth, N. & Teh, Y. W. Disentangling Disentanglement in Variational Autoencoders.

210. Chen, R. T. Q., Li, X., Grosse, R. & Duvenaud, D. Isolating Sources of Disentanglement in VAEs. doi:https://doi.org/10.48550/arXiv.1802.04942.

211. Cífka, O., Severyn, A., Alfonseca, E. & Filippova, K. Eval all, trust a few, do wrong to none: Comparing sentence generation models. (2018).

212. Shao, H. *et al.* Rethinking Controllable Variational Autoencoders. in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* 19228–19237 (IEEE, 2022). doi:10.1109/CVPR52688.2022.01865.

213. Fu, H. *et al.* Cyclical Annealing Schedule: A Simple Approach to Mitigating KL Vanishing. (2019).

214. Chen, L. Y. & Li, Y. P. AutoTemplate: enhancing chemical reaction datasets for machine learning applications in organic chemistry. *J Cheminform* **16**, 1–16 (2024).

215. Leane, M., Pitt, K. & Reynolds, G. Pharmaceutical Development and Technology A proposal for a drug product Manufacturing Classification System (MCS) for oral solid dosage forms The Manufacturing Classification System (MCS) Working Group. (2014) doi:10.3109/10837450.2014.954728.

216. Troup, G. M. & Georgakis, C. Process systems engineering tools in the pharmaceutical industry. *Comput Chem Eng* **51**, 157–171 (2013).

217. Xu, P. *et al.* Optimizing Deeper Transformers on Small Datasets. *ACL-IJCNLP 2021 - 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, Proceedings of the Conference* 2089–2102 (2021) doi:10.18653/V1/2021.ACL-LONG.163.

218. Shen, T., Mueller, J., Barzilay, R. & Jaakkola, T. Educating Text Autoencoders: Latent Representation Guidance via Denoising. (2020).

219. Drugs@FDA: FDA-Approved Drugs. https://www.accessdata.fda.gov/scripts/cder/daf/index.cfm.

220. Lengyel, M., Kállai-Szabó, N., Antal, V., Laki, A. J. & Antal, I. Microparticles, Microspheres, and Microcapsules for Advanced Drug Delivery. *Scientia Pharmaceutica 2019, Vol. 87, Page 20* **87**, 20 (2019).

221. Herbig, M. E., Evers, D. H., Gorissen, S. & Köllmer, M. Rational Design of Topical Semi-Solid Dosage Forms-How Far Are We? *Pharmaceutics 2023, Vol. 15, Page 1822* **15**, 1822 (2023).

222. NextMove Software | Pistachio. https://www.nextmovesoftware.com/pistachio.html.

223. Reaxys | An expert-curated chemical database | Elsevier. https://www.elsevier.com/products/reaxys.

224. Zhao, H. *et al.* Topic Modelling Meets Deep Neural Networks: A Survey. (2021).

225. Jin, D., Jin, Z., Hu, Z., Vechtomova, O. & Mihalcea, R. Deep Learning for Text Style Transfer: A Survey. *Computational Linguistics* **48**, 155–205 (2022).

226. Liu, Z. *et al.* KAN: Kolmogorov-Arnold Networks. (2024).

Appendices

Appendix A: List of standardised operations

Table A1 List of standardised operations for primary manufacturing. Terminology based on Vaucher et al [1]. a. 'MAKE SOLUTION': dissolve and disperse a solid into a solvent to form either a solution or a suspension. b. 'DRY SOLUTION': addition of a desiccant agent to remove the remaining water in a solution. c. 'CONDITION': allow system to equilibrate. d. 'SPE': solid phase extraction. e. 'PURIFY': purification typically by chromatography.

Standardised Operation	Class	Standardised Operation	Class
MAKE SOLUTION [a]	material mixing	BUBBLE	others
ADD	material handling	DRY [AIR-DRYING]	separation methods
STIR	material mixing	ADJUST PH [NEUTRALISE]	setting parameters
COLLECT	material handling	NO ACTION	others
EVAPORATE	separation methods	NO ALTERING	others
WASH	separation methods	ADJUST PH	setting parameters
DRY SOLUTION [b]	separation methods	PARTITION	separation methods
TRITURATE	separation methods	ELUTE	material handling
REMOVE	material handling	IRRADIATE	setting parameters
DRY [FREEZE-DRYING]	separation methods	LOAD	material handling
REACT	chemical transformation	COMBINE	material mixing
PURIFY [e]	separation methods	QUENCH	chemical transformation
REFLUX	setting parameters	ABSORB	material handling
DISTIL	separation methods	DIGEST	chemical transformation
HEAT	setting parameters	SUBLIME	phase transition
COOL	setting parameters	DRY [AZETROPIC-DRYING]	separation methods
CONCENTRATE	separation methods	DEGAS	setting parameters
CRYSTALLISE [RECRYSTALLISATION]	separation methods	VACUUM	setting parameters
STAND	setting parameters	SOAK	material handling
TREAT	material mixing	SET TEMP	setting parameters
EXTRACT	separation methods	CENTRIFUGE	separation methods
PHASE SEPARATION	separation methods	CONDITION [c]	setting parameters
DRY [VACUUM]	separation methods	CHANGE	material handling

FILTER	separation methods	ADSORB	material handling
CRYSTALLISE	separation methods	PRESSURISE	setting parameters
ADJUST PH [INCREASE]	setting parameters	DESALT	separation methods
MELT	phase transition	SONICATE	material mixing
SEPARATE	separation methods	REPEAT	setting parameters
ADJUST PH [DECREASE]	setting parameters	MICROWAVE	setting parameters
DRY	separation methods	SPE [d]	separation methods

Table A2 List of standardised operations for secondary processing. a. 'SIZE': reduction in particle size without a defined method. b. 'HME': hot melt extrusion.

Standardised Operations		
ADD	MAINTAIN	RECONSTITUTE
STIR	IMMERSE	DIRECT COMPRESS
FILTER	CURE	PRECOMPRESS
WASH	GRIND	DISTIL
DRY	MILL	REFLUX
MIX	ALLOW STAND	HYDRATE
WEIGH	DISCARD	CRYSTALLISE
DISSOLVE	DISPENSE	IRRADIATE
COAT	SET TEMP	ADHERE
FILL	COMBINE	FLUID BED GRANULATE
APPLY	DISCHARGE	PURGE
SPRAY	LUBRICATE	DEPRESSURISE
DILUTE	SIZE [a]	ABSORB
EXTRUDE	CAST	VACUUM
SPHERONISE	MELT	ADSORB
SPRAY COAT	SPREAD	PULVERISE
AIR DRY	POLISH	HME [b]
ENTRAP	SONICATE	ENCASE
PLACE	ADJUST PH - ACID	LAMINATE
REMOVE	FILM COAT	CONDITION
PACK	DRY GRANULATE	CONDENSE
PUMP	SOLIDIFY	SHAPE
NO ACTION	ROLLER COMPACT	EXPOSE
EMULSIFY	MOISTEN	SWELL
EVAPORATE	VACUUM DRY	ADJUST PH - BASIC
COLLECT	FEED	BUBBLE
KEEP	AGITATE	RECIRCULATE
HEAT	PASS	PRESSURISE
NO ALTERING	INTRODUCE	PRECIPITATE

CENTRIFUGE	PELLETISE	ADJUST TONICITY
ADJUST VOLUME	SEAL	AGE
HOMOGENISE	ADJUST PH - NEUTRALISE	GEL
COOL	CUT	DYALISE
SCREEN	DEGAS	DEIONISE
TRANSFER	ADJUST PH	POLYMERISE
FLUIDISE	FREEZE	THICKEN
SUSPEND	LYOPHILISE	ASEPTIC FILL
STORE	MICRONISE	LEAVE
DISPERSE	AEROLISE	SPRAY CONGEAL
STAND	DRY BLEND	MELT GRANULATE
SEPARATE	SPRAY DRY	DESORB
RINSE	EXTRACT	PHASE SEPARATION
PORTION	DRILL	ELECTROSPIN
ENCAPSULATE	DISTRIBUTE	PHASE INVERSION
KNEAD	ALLOW	IMPREGNATE
GRANULATE	INDUCE COACERVATION	SUBLIME
BLEND	PURIFY	DESALT
COMPRESS	DECANT	SINTER
WET GRANULATE	CONCENTRATE	MICROWAVE
INJECT	VORTEX	

1. Vaucher, A. C. *et al.* Automated extraction of chemical synthesis actions from experimental procedures. *Nature Communications 2020 11:1* **11**, 1–11 (2020).

Appendix B: Topics keywords for LDA analysis

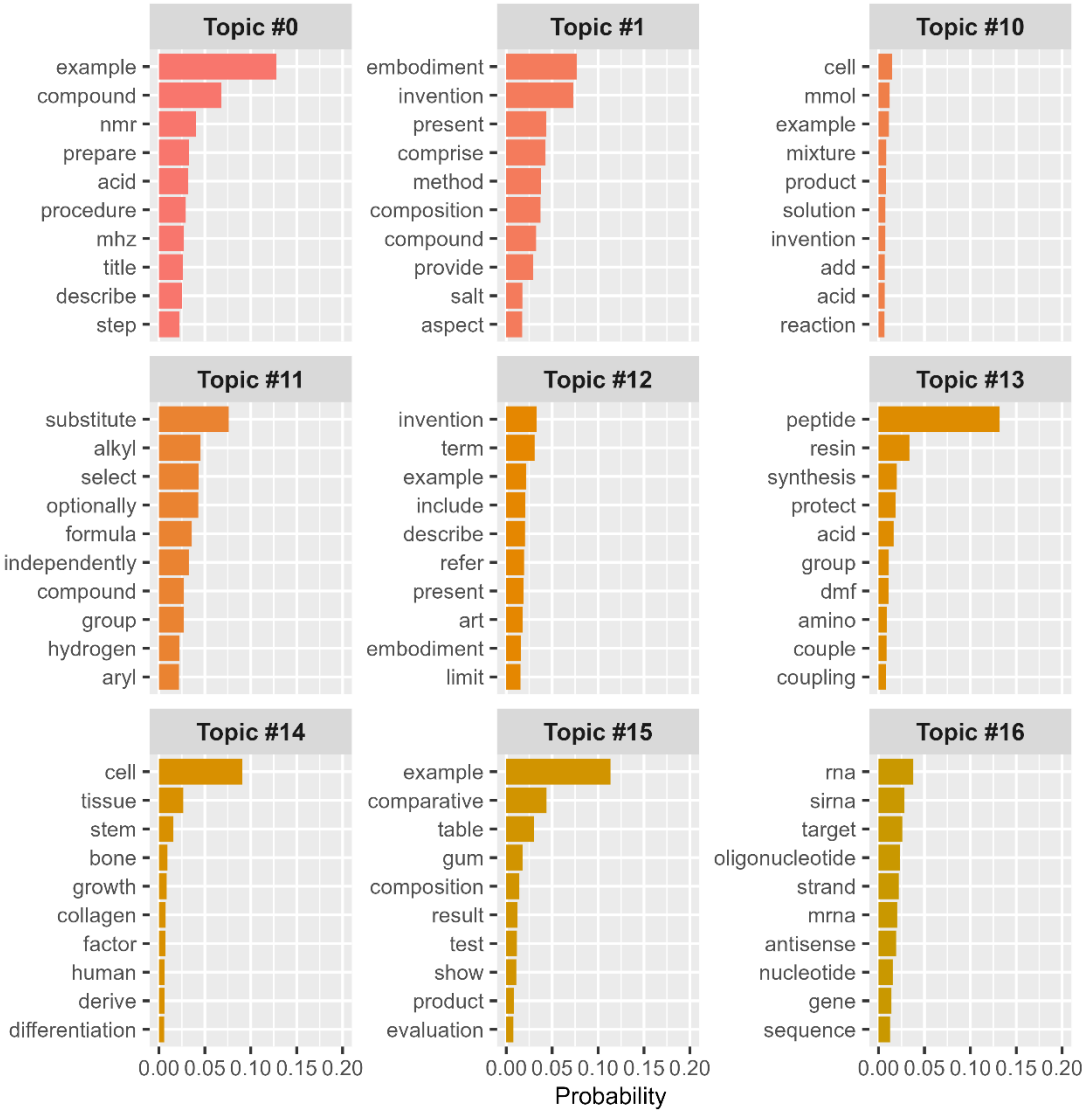


Figure A1. Keywords for LDA analysis with truncated data for topics 0 to 16.

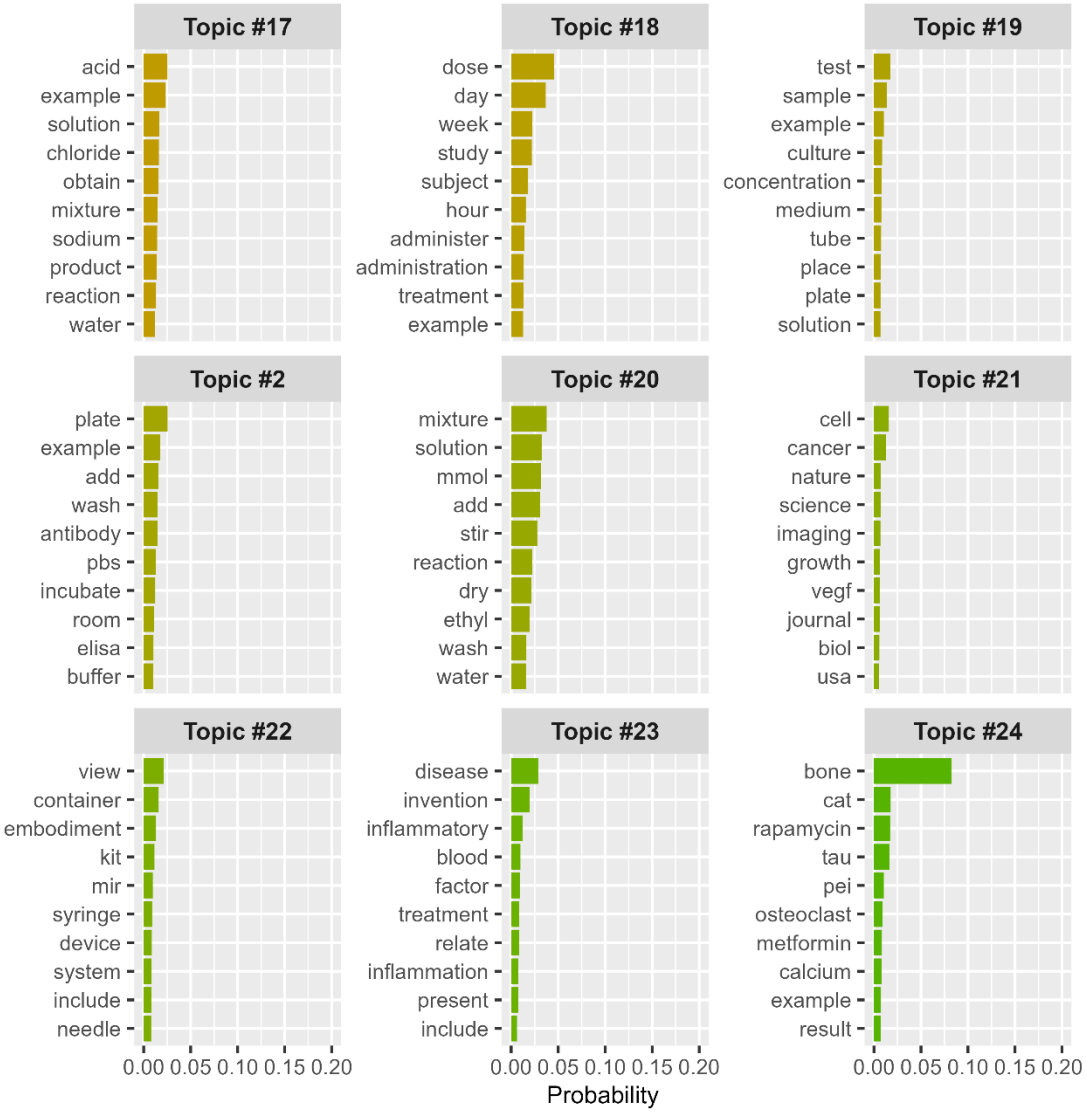


Figure A2. Keywords for LDA analysis with truncated data for topics 17 to 24.

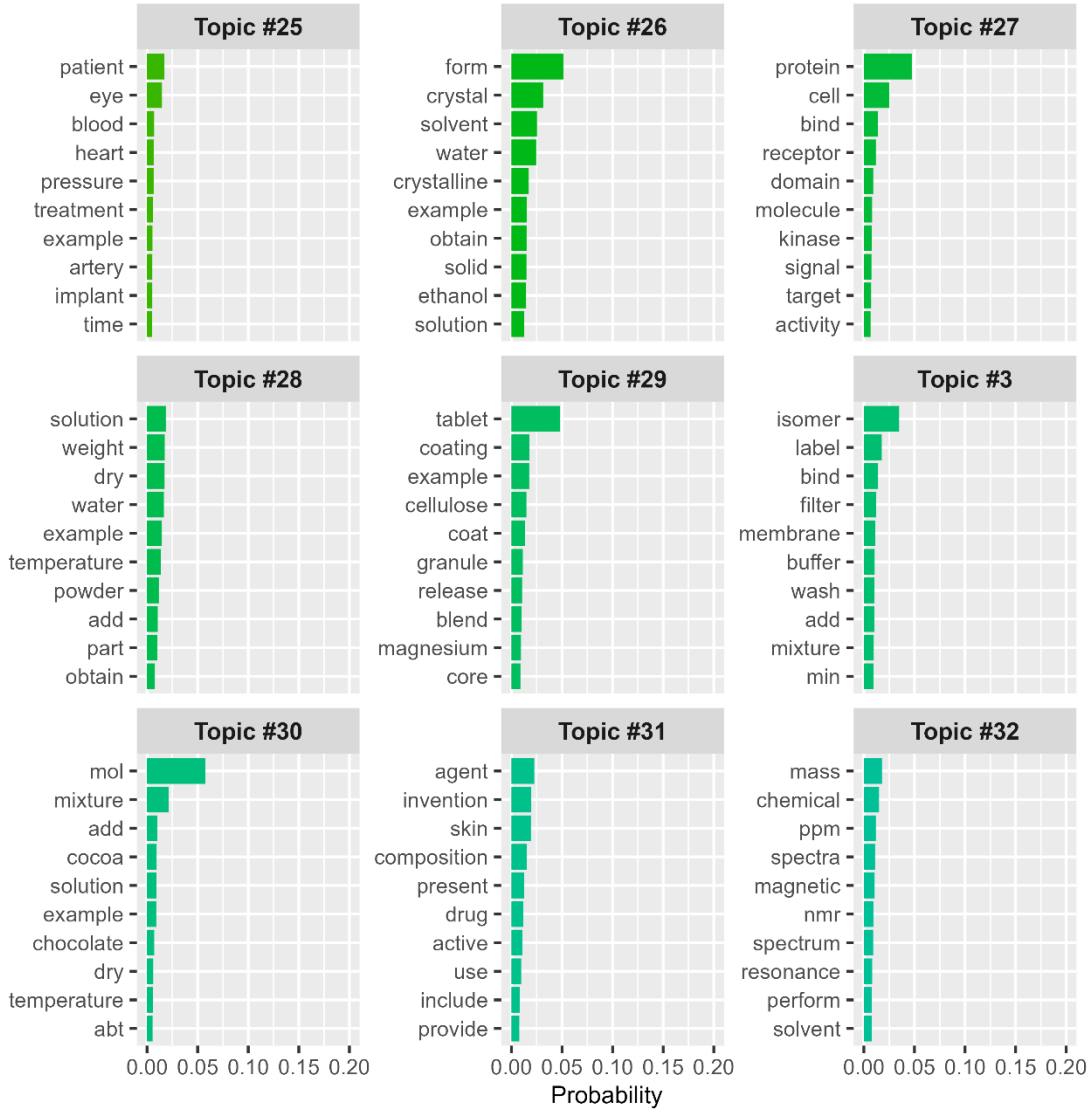


Figure A3. Keywords for LDA analysis with truncated data for topics 25 to 32.

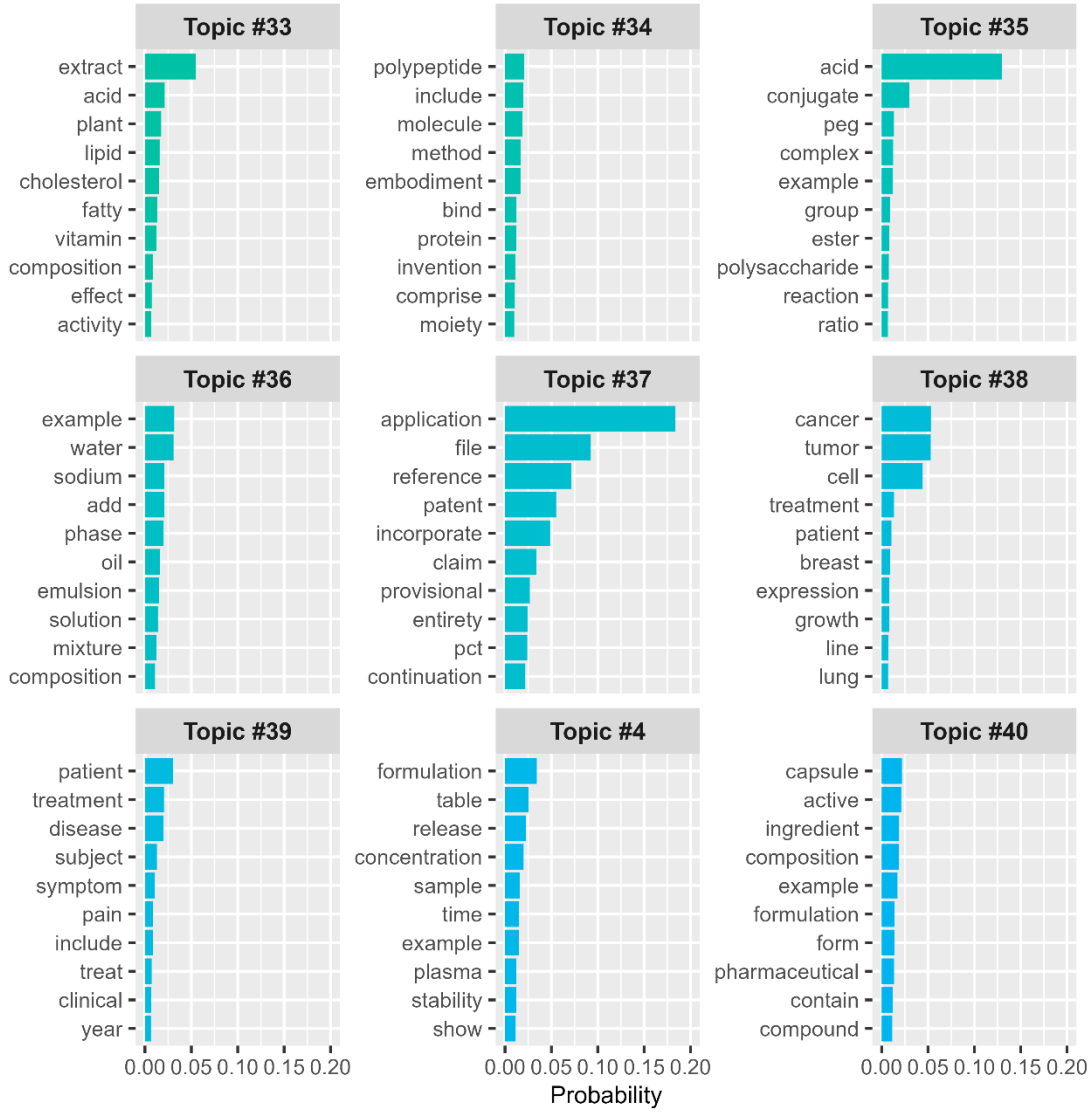


Figure A4. Keywords for LDA analysis with truncated data for topics 33 to 40.

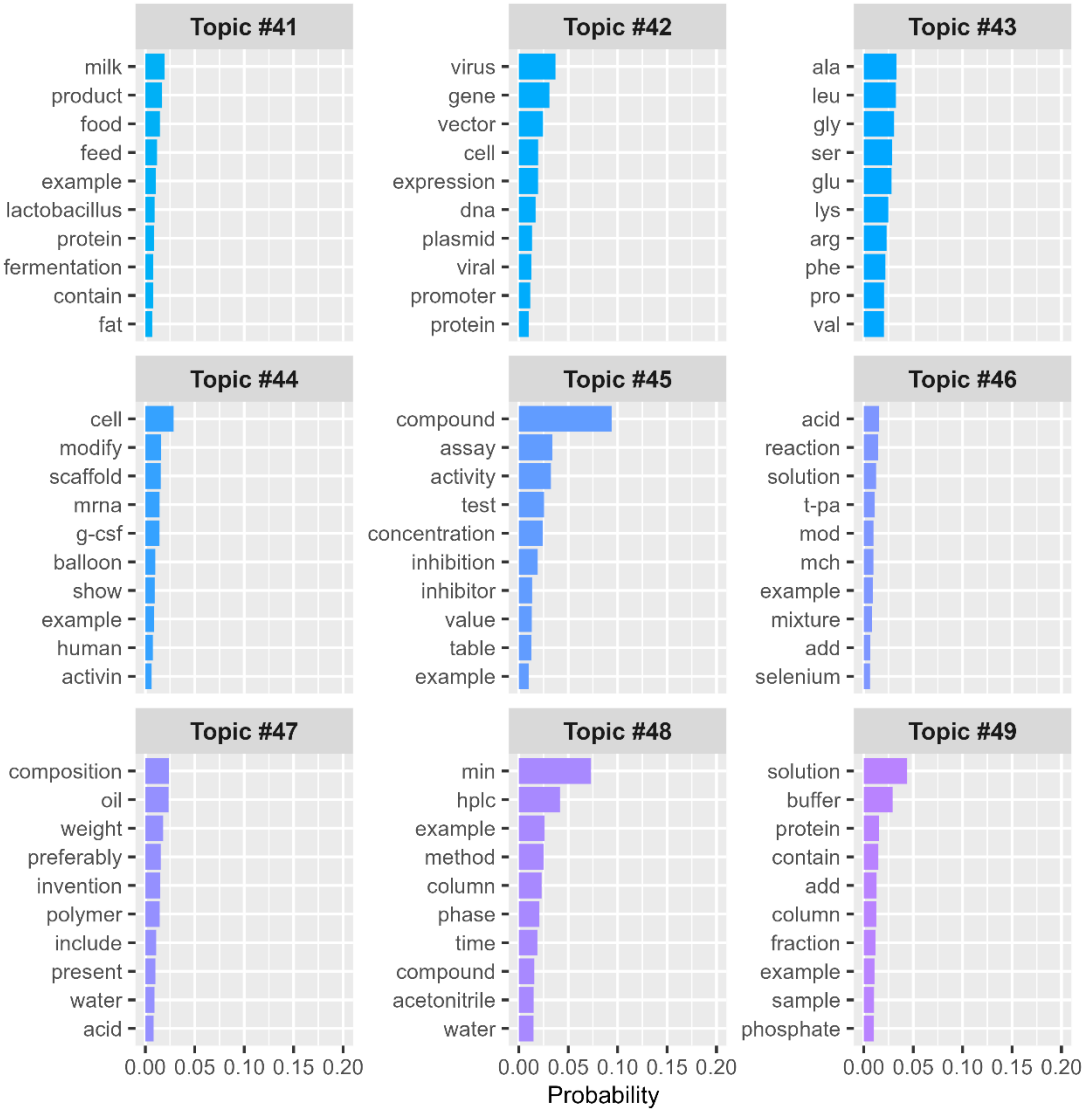


Figure A5. Keywords for LDA analysis with truncated data for topics 41 to 49.

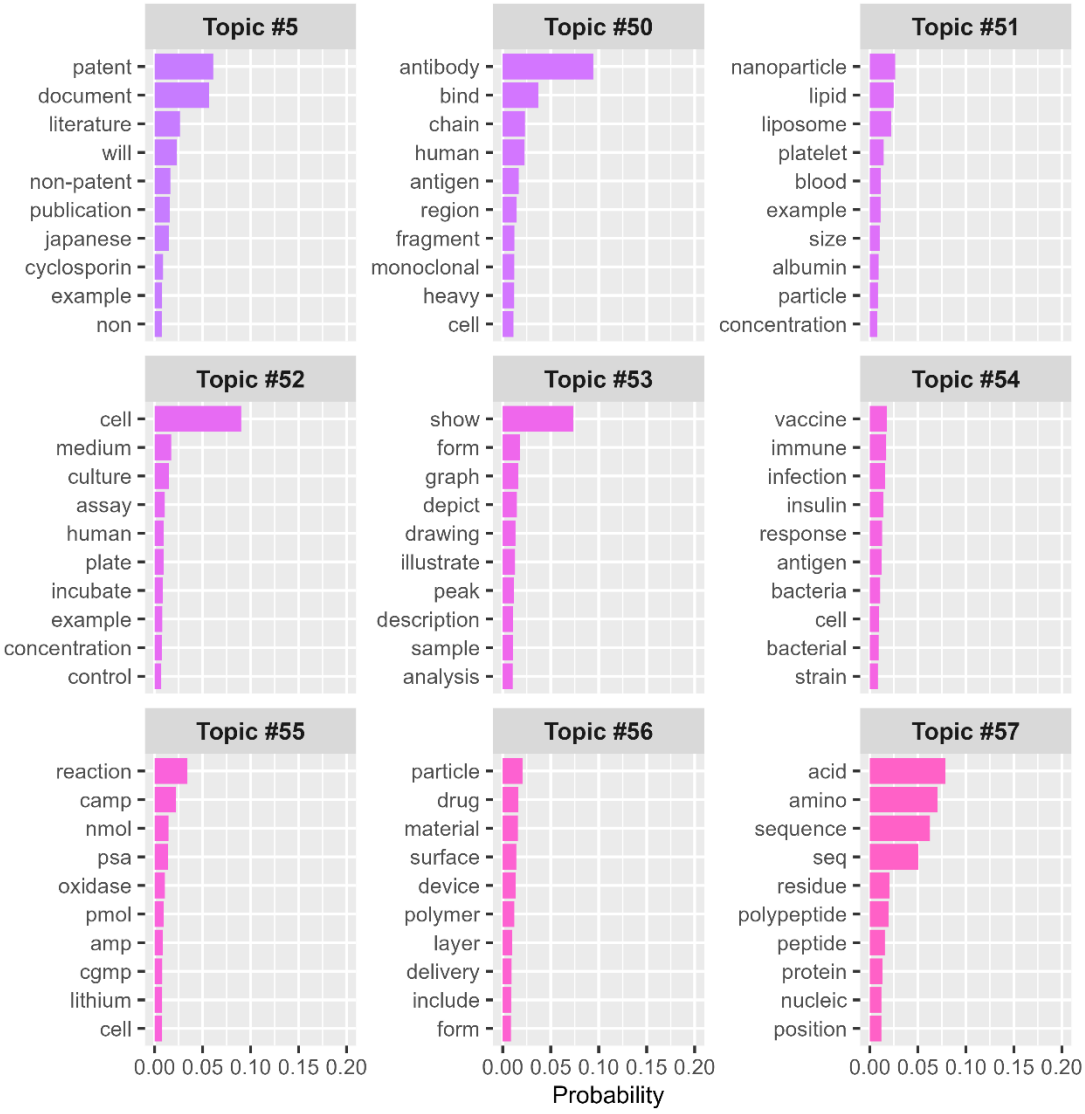


Figure A6. Keywords for LDA analysis with truncated data for topics 50 to 57.

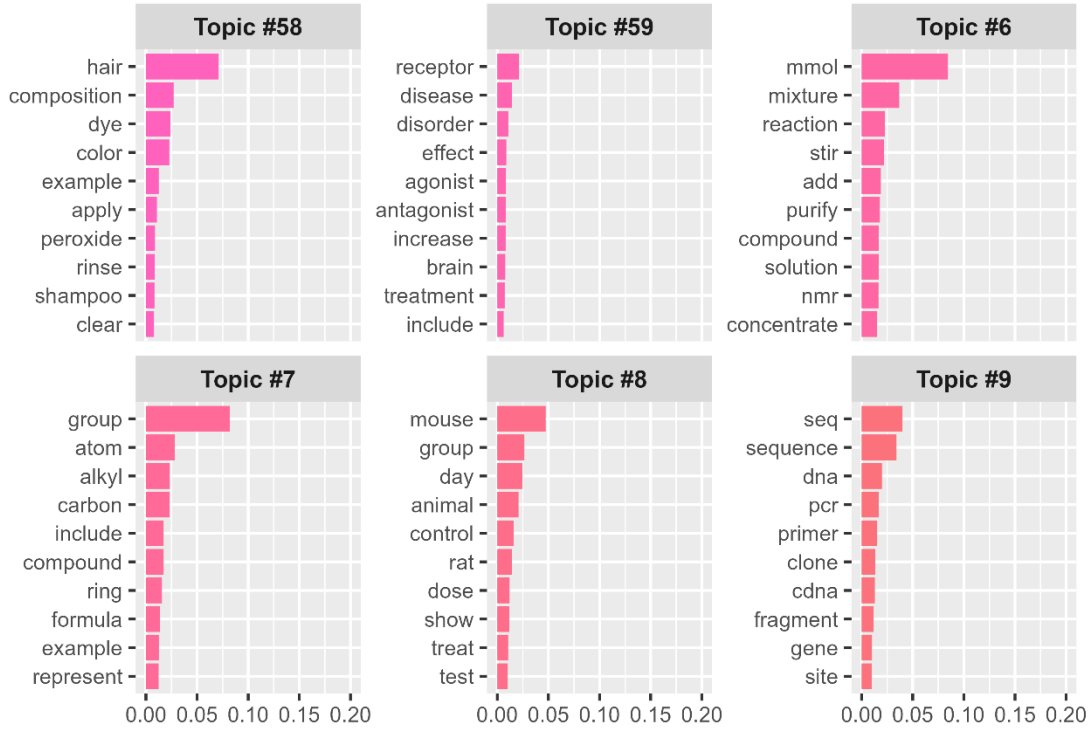


Figure A7. Keywords for LDA analysis with truncated data for topics 6 – 9, 58, and 59.

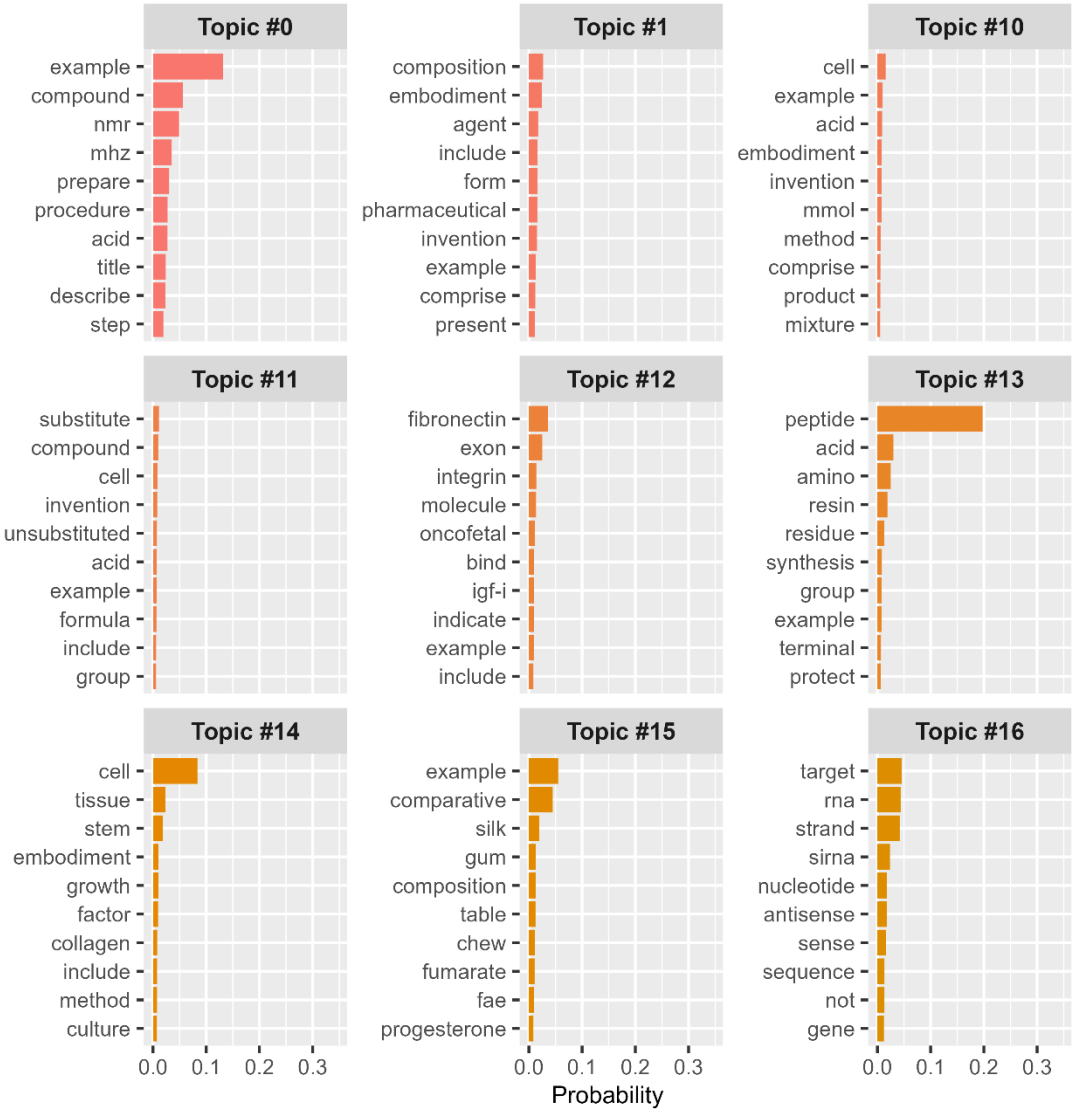


Figure A8 Keywords for LDA analysis without truncation for topics 0, 1, 10 - 16.

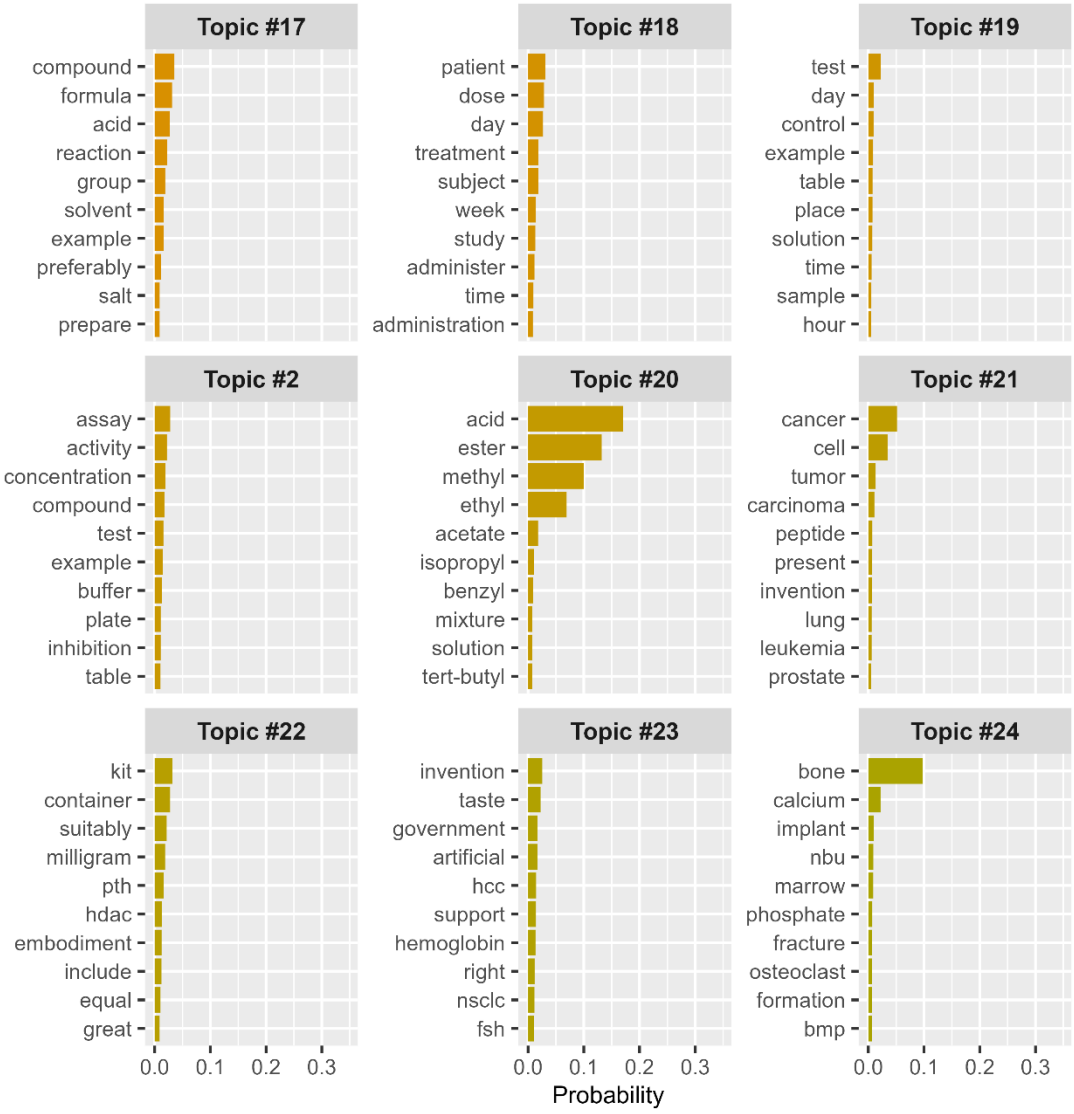


Figure A9 Keywords for LDA analysis without truncation for topics 17 - 24, and 2.

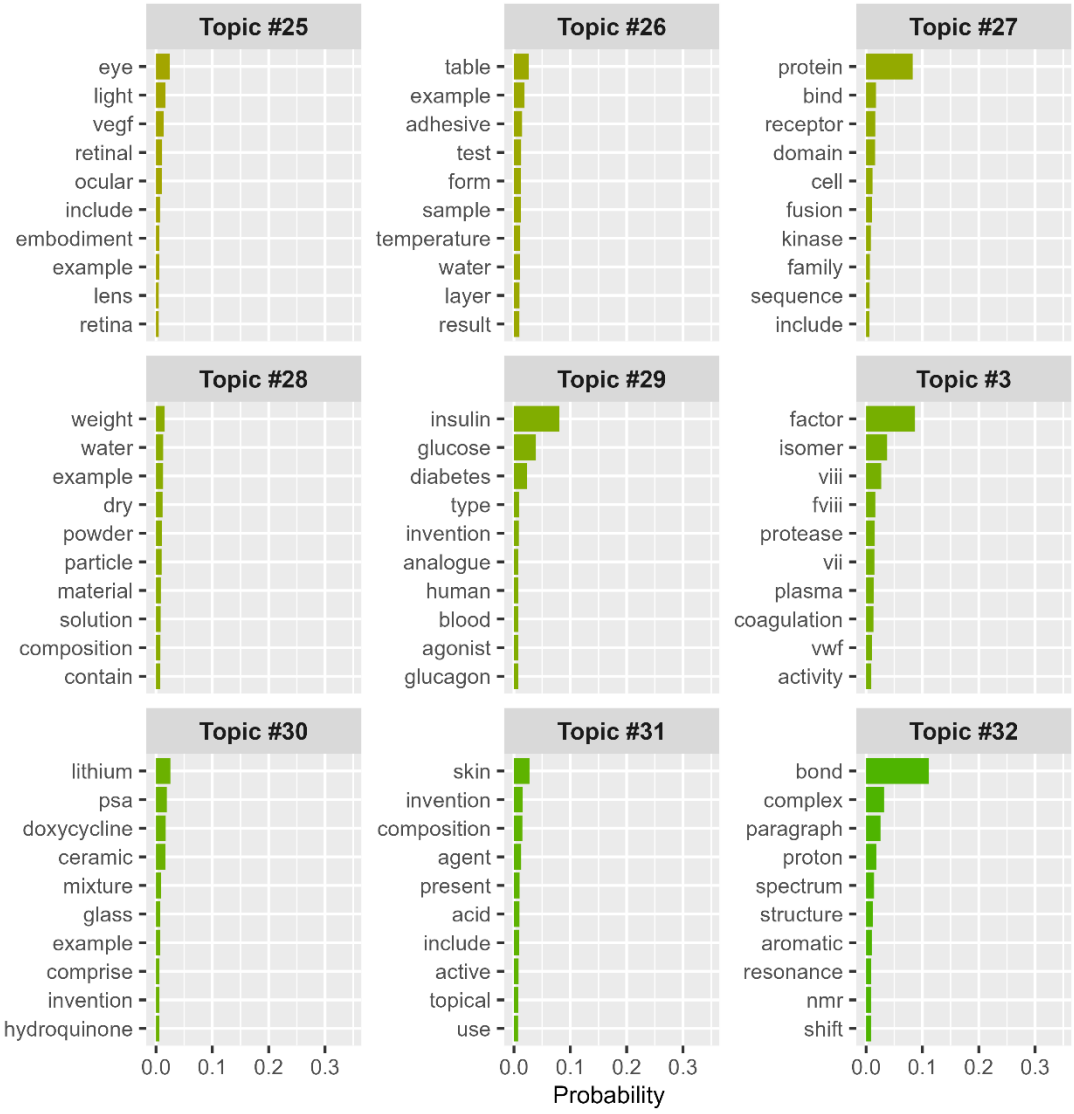


Figure A10 Keywords for LDA analysis without truncation for topics 25 - 32.

Figure A11 Keywords for LDA analysis without truncation for topics 33 - 40.

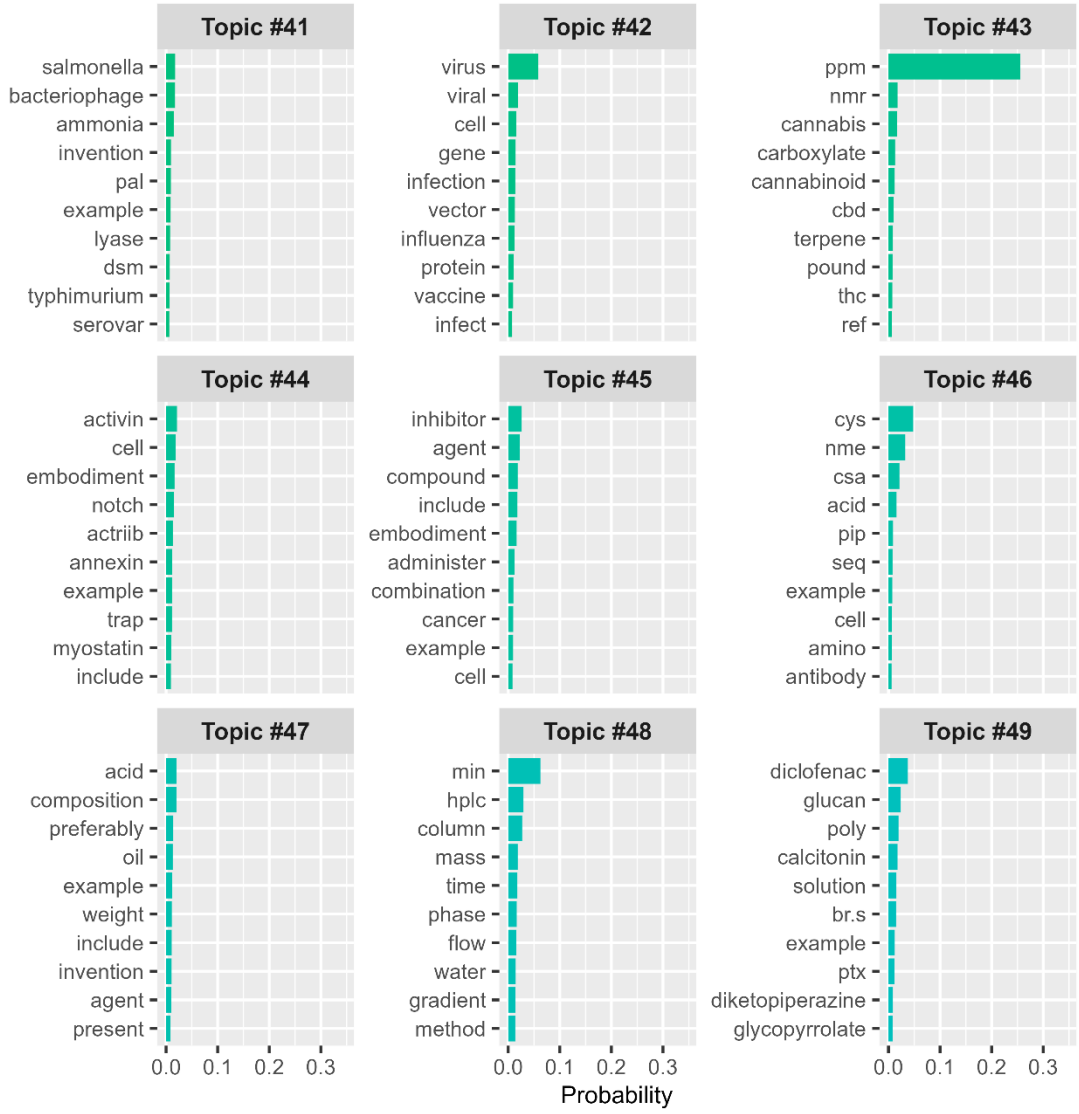


Figure A12 Keywords for LDA analysis without truncation for topics 41 - 49.

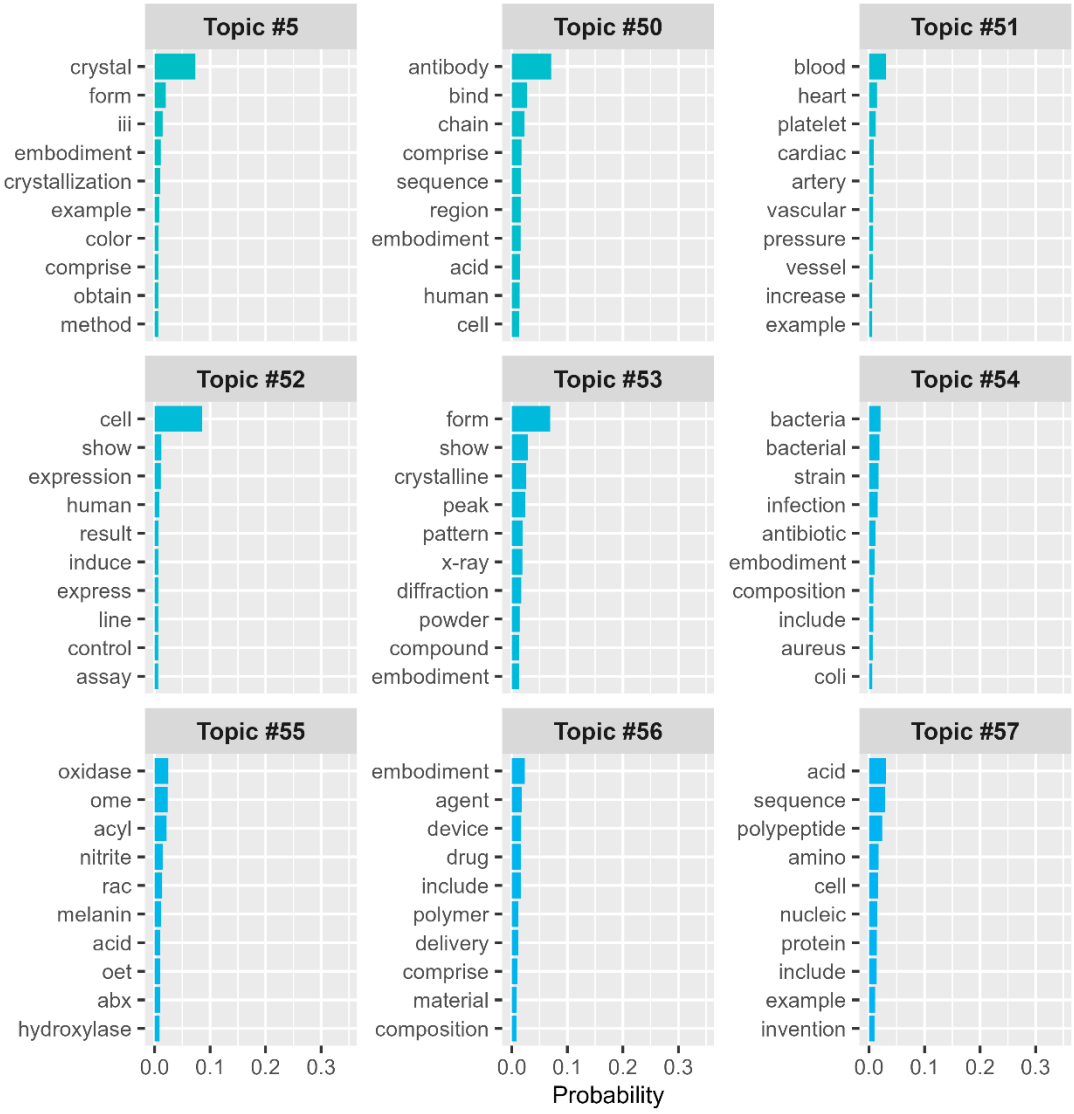


Figure A13 Keywords for LDA analysis without truncation for topics 50 – 57 and 5.

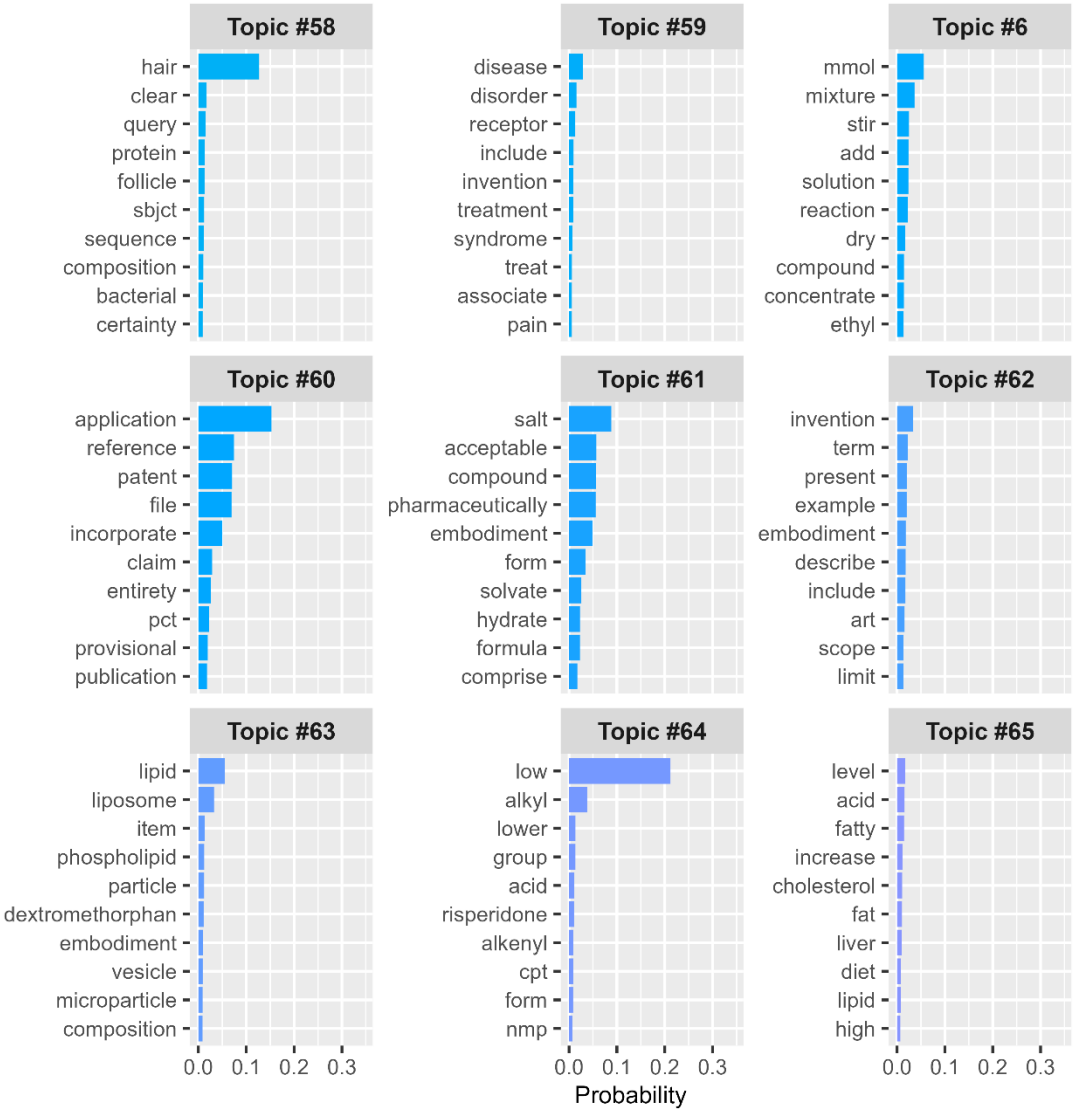


Figure A14 Keywords for LDA analysis without truncation for topics 58 – 65 and 6.

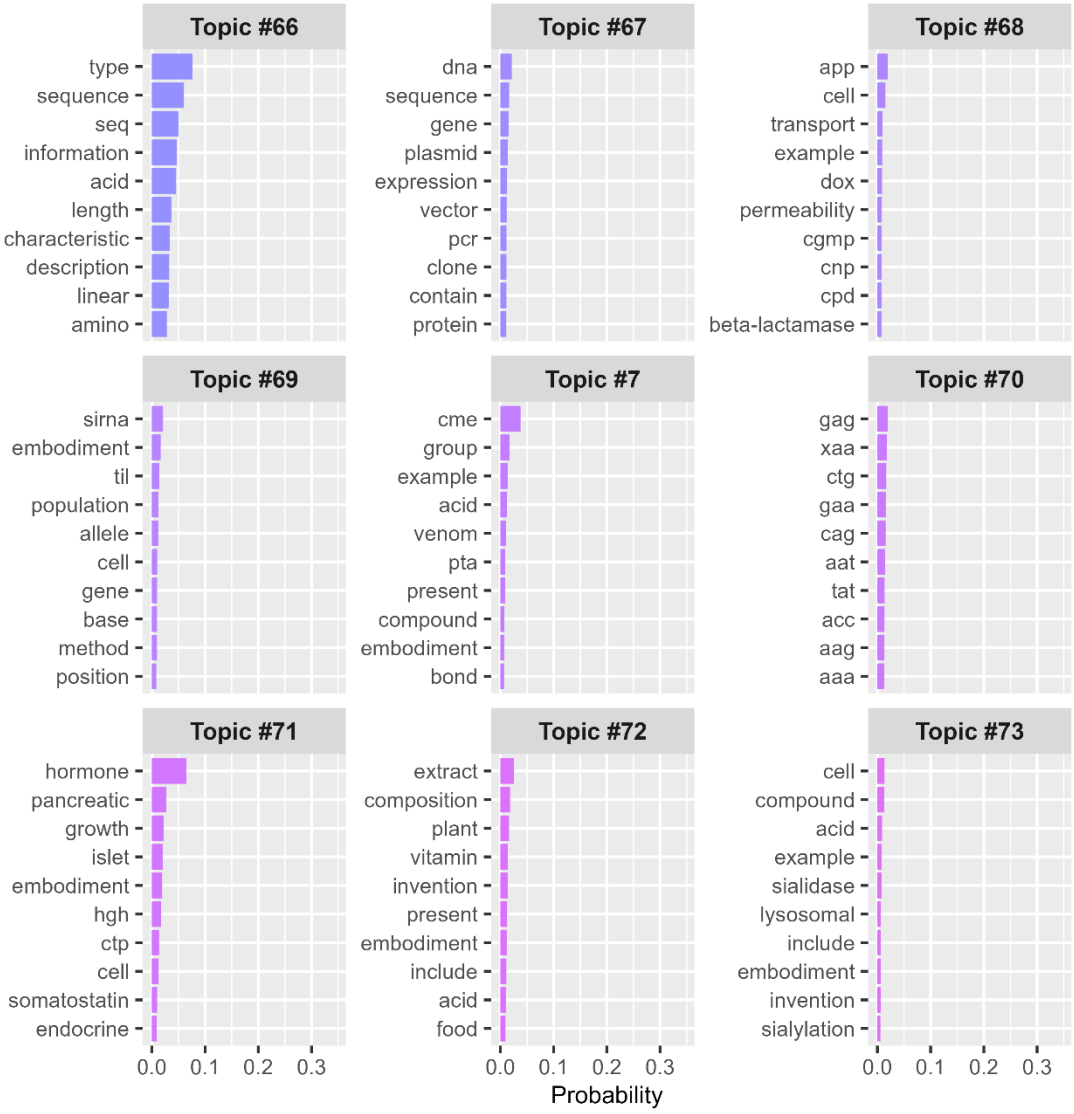


Figure A15 Keywords for LDA analysis without truncation for topics 66 – 73 and 7.

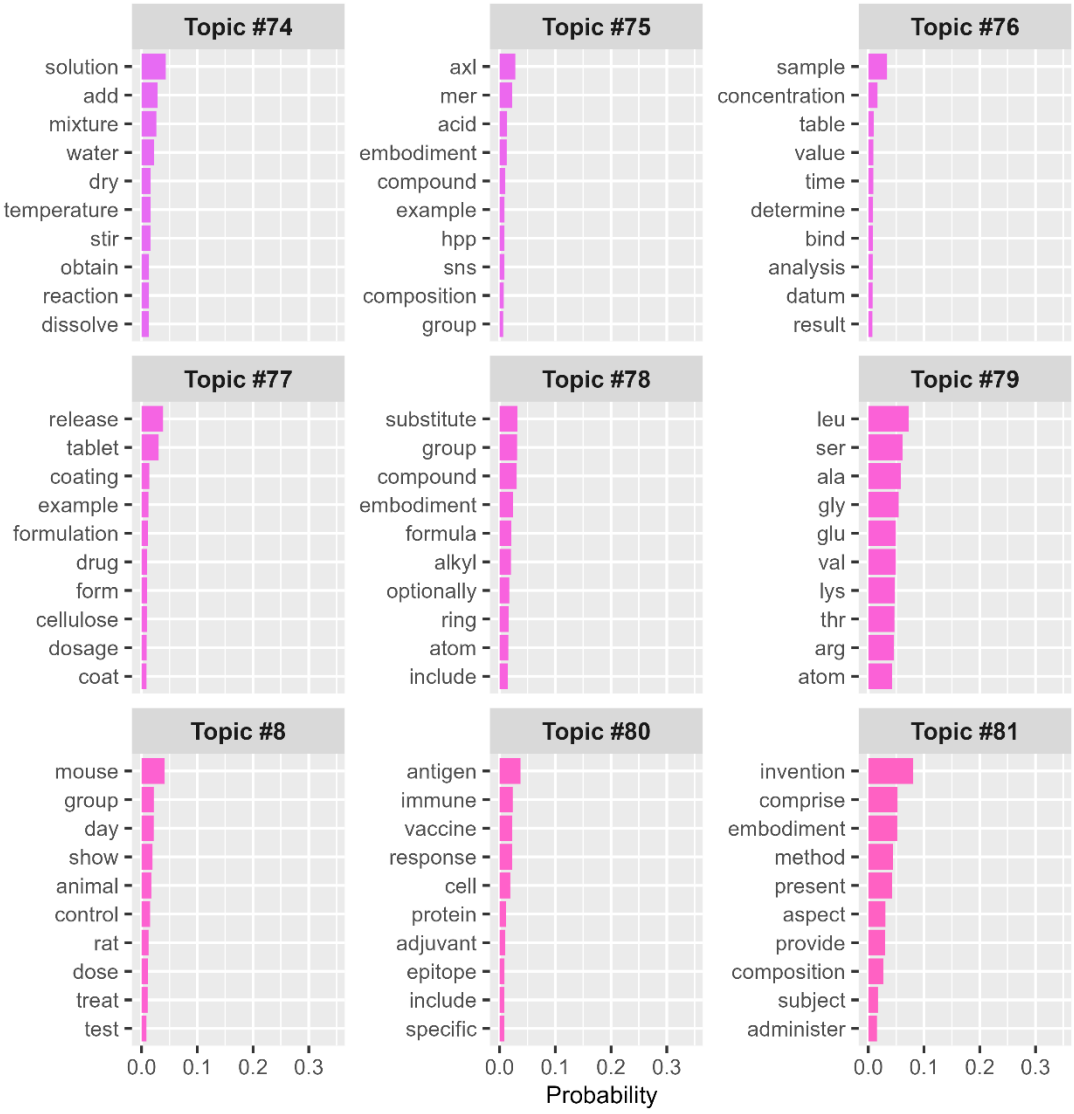


Figure A16 Keywords for LDA analysis without truncation for topics 74 – 81 and 8.

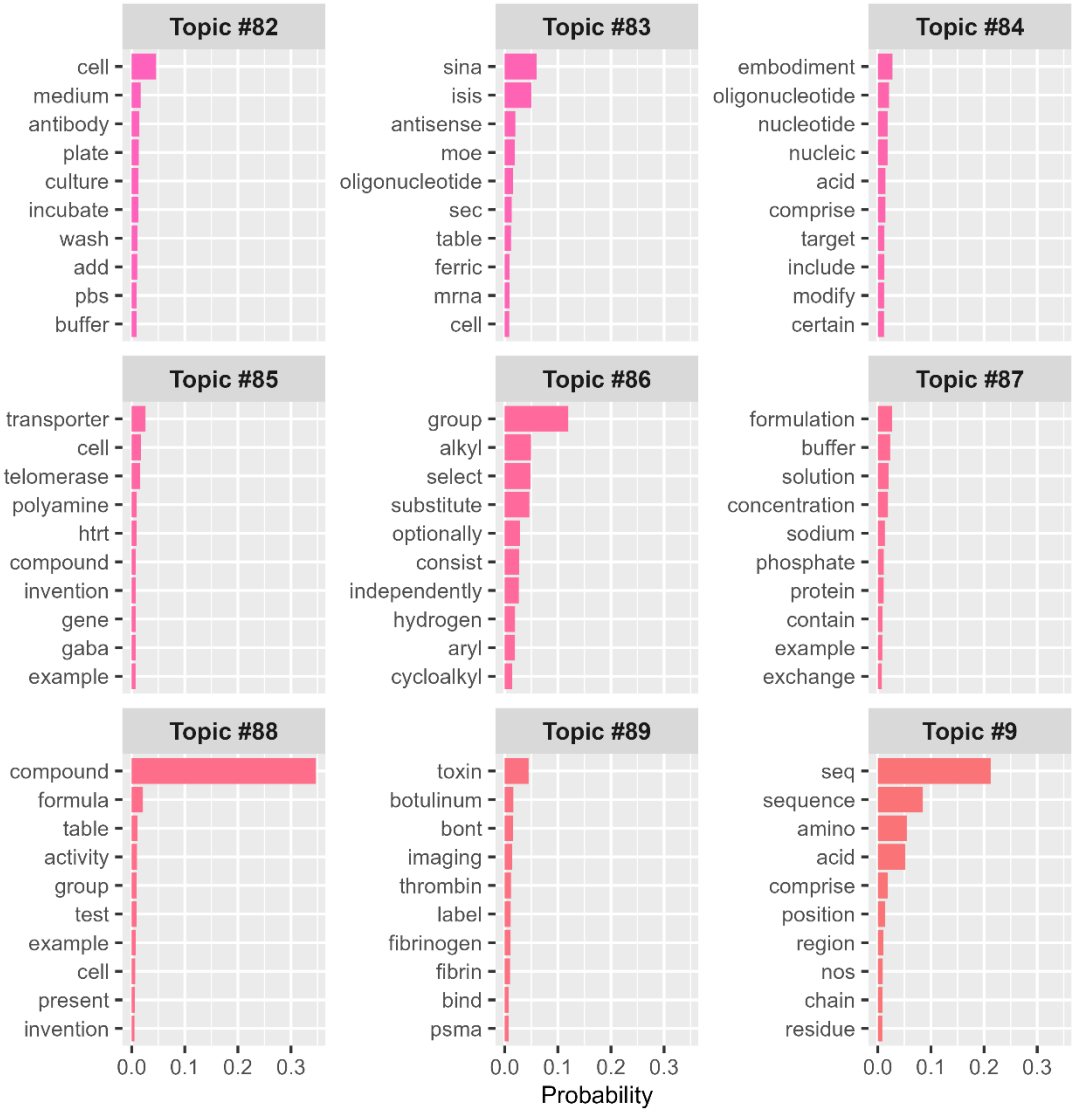


Figure A17 Keywords for LDA analysis without truncation for topics 82 – 89, and 9.

Appendix C: Ranges for clusters of conditions

Table A3 Ranges of temperature for primary manufacturing.

Group	Values (°C)
T0	< -10
T1	-10 – 10
T2	10 – 40
T3	40 – 80
T4	> 80

Table A4 Ranges of time for primary manufacturing.

Group	Values (h)
t0	< 0.75
t1	0.75 – 3.00
t2	3.00 – 9.00
t3	9.00 – 36.0
t4	36.0 – 96.0
t5	> 96.0

Table A5 Ranges of pH for primary manufacturing.

Group	Values
0	< 3.5
1	3.5 – 6.0
2	6.0 – 8.0
3	> 8.0

Table A6 Ranges of temperature for secondary manufacturing.

Group	Values (°C)
T0	< 0
T1	0 – 25
T2	25 – 50
T3	50 – 70
T4	> 70

Table A7 Ranges of time for secondary manufacturing.

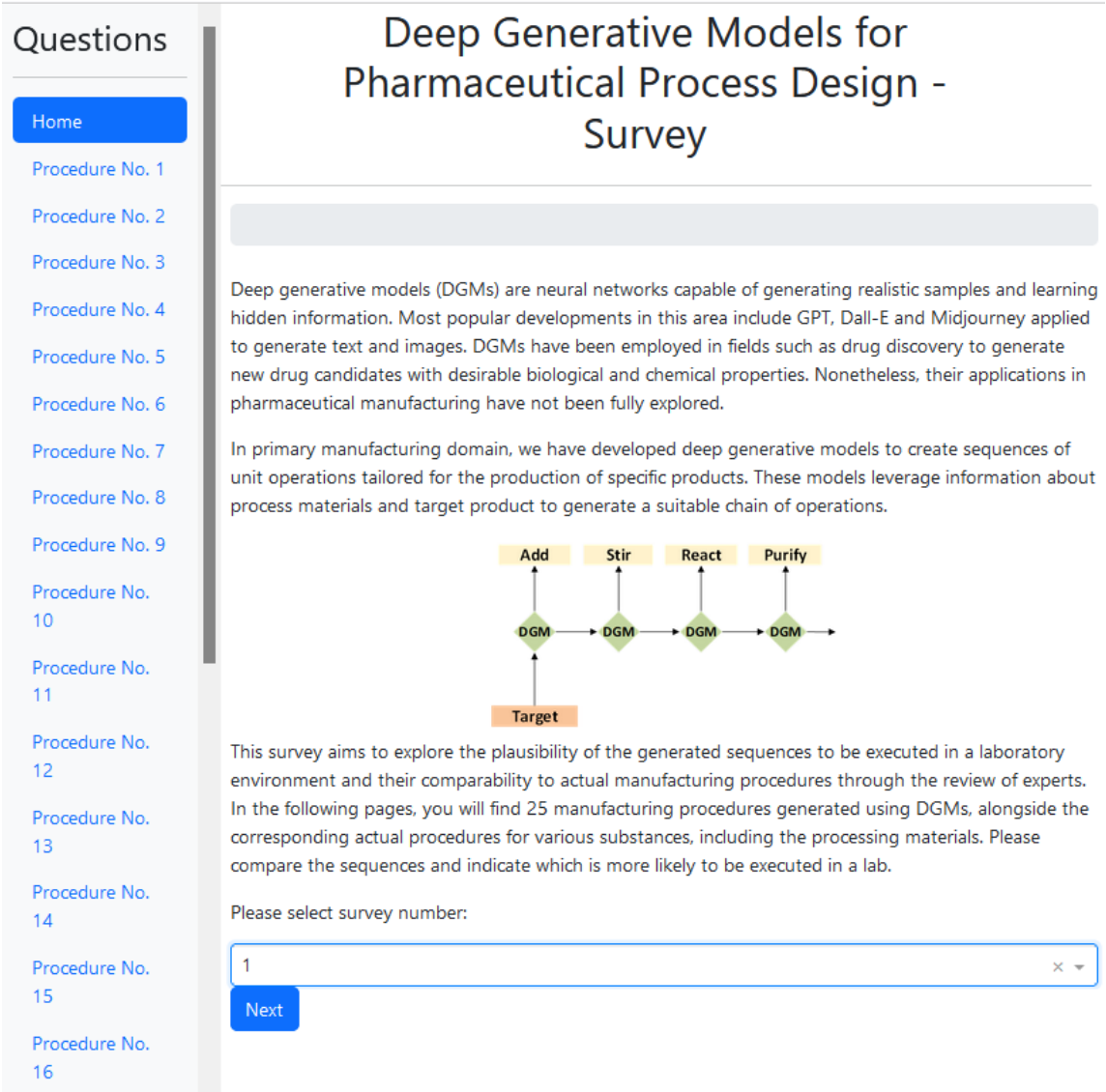
Group	Values (min)
t0	< 5
t1	5 – 20
t2	20 – 120
t3	> 120

Table A8 Ranges of pH for secondary manufacturing.

Group	Values (min)
0	< 2.0
1	2.0 – 4.5
2	4.5 – 7.0
3	7.0 – 8.5
4	> 8.5

Appendix D: Survey for assessment of the quality of generated procedures for primary manufacturing by CVAE and Invariant CVAE

Below can be seen examples of the questions asked to validate the performance of the generative models



Deep Generative Models for Pharmaceutical Process Design - Survey

Progre

Procedure No. 1

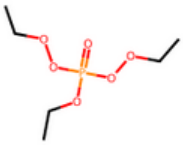
Consider the following list of reactants and two sequences, labeled 'A' and 'B,' for manufacturing a product ('TARGET'). Choose the sequence of operations that is consired more suitable for manufacturing the product based on the provided materials. If neither sequence is superior or if clarity is lacking, you may also select 'Both' or 'None'.

1. Materials:

	Role	Name
0	REACTANT	sodium hydrogen
1	SOLVENT	THF
2	SOLVENT	ethyl 2-diethoxy phosphate
3	SOLVENT	water
4	TARGET	ethyl (e)-3-(4,4-difluorocyclohexyl)acrylate

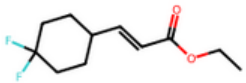
REACTANT (0)

SOLVENT (1)



SOLVENT (2)

SOLVENT (3)



2. Procedures:

	A	B
0	[START]	[START]
1	ADD THF	ADD sodium hydrogen
2	ADD ethyl 2-diethoxy phosphate	ADD ethyl 2-diethoxy phosphate
3	STIR at 65°C for 5 h	ADD THF at 0°C
4	ADD water	STIR at 0°C for 20 min
5	EXTRACT with ethyl acetate	ADD at 0°C
6	REPEAT 3 time(s)	STIR at 25°C for 2 h
7	WASH organic phase with brine	ADD water
8	REPEAT 2 time(s)	EXTRACT with ethyl acetate
9	DRY SOLUTION with Na2SO4	REPEAT 2 time(s)
10	REMOVE solvent	WASH organic phase with brine
11	PURIFY by chromatography	DRY SOLUTION with Na2SO4
12	YIELD: ethyl (e)-3-(4,4-difluorocyclohexyl)acrylate	FILTER
13	[END]	EVAPORATE
14		PURIFY by chromatography
15		YIELD: ethyl (e)-3-(4,4-difluorocyclohexyl)acrylate
16		[END]

Response:

- ○ A
- ◉ B
- ○ Both
- ○ None

Back Next

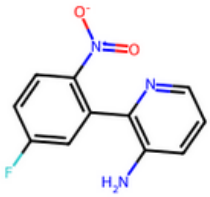
Procedure No. 16

Consider the following list of reactants and two sequences, labeled 'A' and 'B,' for manufacturing a product ('TARGET'). Choose the sequence of operations that is consired more suitable for manufacturing the product based on the provided materials. If neither sequence is superior or if clarity is lacking, you may also select 'Both' or 'None'.

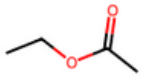
1. Materials:

	Role	Name
0	REACTANT	Pd / C
1	REACTANT	(5-fluoro-2-nitrophenyl)pyridin-3-ylamine
2	SOLVENT	ethyl acetate
3	TARGET	4-fluoro-n2-pyridin-3-yl-benzene-1,2-diamine

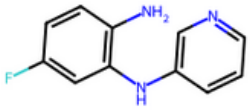
REACTANT (0)



REACTANT (1)



SOLVENT (2)



TARGET (3)

2. Procedures:

	A	B
0	[START]	[START]
1	ADD ethyl acetate	ADD (5-fluoro-2-nitrophenyl)pyridin-3-ylamine
2	ADD (5-fluoro-2-nitrophenyl)pyridin-3-ylamine	ADD ethyl acetate
3	ADD Pd / C	DEGAS
4	STIR at 25°C for 2 h	ADD Pd / C
5	FILTER	STIR at 25°C for 2 h
6	WASH with ethyl acetate	ADD
7	CONCENTRATE filtrate	STIR for 48 h
8	YIELD: 4-fluoro-n2-pyridin-3-yl-benzene-1,2-diamine	FILTER
9	[END]	CONCENTRATE
10		PURIFY by chromatography
11		YIELD: 4-fluoro-n2-pyridin-3-yl-benzene-1,2-diamine
12		[END]

Response:

- ○ A
- ○ B
- ◉ Both
- ○ None

Back Next