Improving Ease and Speed of Use of Mobile Augmentative and Alternative Communication Systems through the use of Natural Language Processing and Natural Language Generation Techniques

Thesis submitted in fulfilment of the requirements for the degree of Doctor of Philosophy

Gennaro Imperatore BSc(Hons), MSc (Dist)

Department of Computer and Information Sciences

University of Strathclyde

Glasgow

April 2016

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: **(Gennaro Imperatore)**

Date:

Acknowledgments

First of all I would like to thank Dr Mark Dunlop for his great supervision and support. The biggest thank you should go to Donna for agreeing to meet with me and making me think of the initial idea which sparked this project

I would also like to thank EPSRC for funding this research

A great thank you to the people at SpeakAbility Edinburgh for helping me in the initial phases of this research and to the wonderful people at the "Tap and Talk" group at the University of Dundee for helping me during the design and the evaluation of the application. A special thanks goes to Rolf and Amy for the extensive feedback they provided and to the members of the British Aphasiology Society for giving me great advice to start my research.

Finally I would like to thank my Gran for helping me with proofreading and supporting me during this work.

Conferences and Publications

Conferences and Workshops Attended

- CHI 2014 (Paris)
- Technology Health and Wellbeing Workshop University of Strathclyde
- Technology Health and Wellbeing Workshop Glasgow Caledonian University
- British Aphasiology Society Research Update Meeting 2014 University of Newcastle
- Technology Health and Wellbeing University of Glasgow
- SICSA Conference on Speech and Language University of Edinburgh
- Technology Health and Wellbeing University of Strathclyde
- ASSETS 2015 (Lisbon, Portugal)

Publications

- "An Investigation into Appropriation of Portable Smart Devices by Users with Aphasia", Gennaro Imperatore and Mark Dunlop Proceedings of Assets 2015 (Posters), ACM Press, Portugal, October 2015.

Index

Chapter I: Background.....1

1.1 Research Motivation.....1

1.2 Introduction.....2

1.3 Rationale for Researching AAC.....2

1.4 Motivational Interview with Glasgow Based Speech and Language Therapist.....6

1.5 Methodology.....6

1.6 Related Work.....7

1.6.1 Conversation and Utterance Based Research in AAC.....7

1.6.2 Topic Shifting, Semantic Networks and Hyperlinks.....8

1.6.3 Narrative Based AAC and Pragmatics Modelling AAC.....9

1.6.4 Other Work.....10

1.7 How to Improve AAC.....10

1.8 Thesis Statement.....12

1.9 Research Approach.....12

1.10 Hypotheses.....13

1.11 Research Questions.....13

1.12 Summary.....14

1.13 Chapter Conclusions.....14

Chapter II: Background.....15

2.1 Introduction.....15

2.2 Augmentative and Alternative Communication Systems.....16

2.2.1 Overview of AAC Apps Available Today.....19

2.2.1.1 DynaVox Devices and Software.....20

2.2.1.2 GridPlayer.....20

2.2.1.3 Proloquo2Go.....22

2.2.1.4 RocketKeys.....23

2.2.1.5 OneVoice.....24

2.2.1.6 EasySpeak.....25

2.2.1.7 TalkRocket Go 26
2.2.1.8 Assistive Chat 27
2.2.1.9 Verbally 28
2.3 Natural Language Processing 29
2.3.1 Character Recognition and Tokenisation 31
2.3.2 Morphological Analysis 32
2.3.3 Lexical Analysis and Part of Speech Tagging 32
2.3.4 Syntactic Analysis 33
2.3.5 Semantic Analysis 33
2.4 Word Association Measures 34
2.4.1 Measures Based on Co–Occurrence Frequency 35
2.4.1.1 Mutual Information and Pointwise Mutual Information (PMI) 36
2.4.1.2 Significance Based Alternatives to PMI 37
2.4.2 Example Comparison of PMI, PMIsig and SCI Using Pre-Processor Discussed in Chapter V 40
2.4.3 Measures Based on Distance Between Words 41
2.4.4 Measures Based on Syntactic Relations 41
2.4.4.1 The Classic Lesk Algorithm 41
2.4.4.2 The Extended Gloss Algorithm 42
2.4.4.3 WordNet vs. PMIsig and Others 42
2.4.5 Latent Semantic Indexing/Analysis (LSI/LSA) 42
2.5 Introduction to the Companion Algorithm 44
2.6 Natural Language Generation 45
2.6.1 Generative Grammars 46
2.6.2 A Practical Example of NLG: Functional Unification Grammars and FUF/SURGE 47
2.6.3 SimpleNLG and Justification for its Use as Realiser 50
2.6.4 Language Models (N-Grams) 50
2.6.5 Semantic Grams 52
2.7 Inclusion of NLP and NLG in AAC in the Work of Others 53
2.8 Discussion 53
2.9 Key Lessons 54

2.10 Summary 54
Chapter III: Thesis Overview 56
3.1 Introduction 56
3.2 Thesis Overview 56
3.2.1 Dictum App Evaluation Overview 58
3.3 Thesis Structure 59
3.3.1 Chapter IV: Corpus Selection 59
3.3.2 Chapter V: Language Pre-Processor Development 60
3.3.3 Chapter VI : Capturing User Requirements in Cases of Users with Complex Communication Needs 62
3.3.4 Chapter VII AAC Application Design and Development 63
3.3.5 Chapter VIII: Natural Language Generation 64
Outcomes of Chapter: 64
3.3.6 Chapter IX: Evaluation of the AAC Application 65
3.3.7 Chapter X: Future Work 65
Chapter IV: Corpus Selection 66
4.1 Introduction 66
4.2 Corpora 67
4.2.1 The Brown Corpus 67
4.2.2 The British National Corpus 68
4.2.3 DBPedia 69
4.2.4 WordNet 70
4.2.4.1 Relationships Between Nouns in WordNet 70
4.2.4.2 Relationships Between Verbs in WordNet 71
4.2.5 Word Association Norms or Human Association Datasets 71
4.3 Web Based Corpora 73
4.4 Justification on the Use of Wikipedia Over Other Web Corpora 74
4.5 Using Wikipedia in Practice 75
4.6 The Use of Tagged Corpora as Reference for Document Processing 78
4.7 Corpus Size Reduction and Filtering Via the Use of Naïve Bayes Classification 78
4.7.1 Theoretical Background of Naïve Bayes Classification 78

4.7.2 Document Classification Using NBC in Weka 79
4.8 Evaluation of the Naïve Bayes Classifier 81
4.9 Using Wikipedia Classification to Filter Articles 82
4.10 User Evaluation of Best Subset of Wikipedia to Use for AAC App 85
4.11 Results of Crowdsourcing Based Evaluation 87
4.11.1 Unfiltered Article Results 87
4.11.2 Filtered Categories Articles Results 89
4.12 Summary 93
4.13 Key Lessons 93
Chapter V: NLP Pre-Processor 94
5.1 Introduction 94
5.2 NLP Pre-Processor 96
5.2.1 Input 97
5.2.2 Tokenisation, Stemming and Part of Speech Filtering 97
5.2.3 Vocabularies Generation 99
5.2.4 Co-Occurrence Counting 100
5.2.5 PMIsig Calculation and Semantic Map Generation 100
5.2.6 Verb Map 101
5.3 Accessing and Using WordNet in NLP Software 102
5.4 Ranking the Words in Topic Level Vocabularies 103
5.4.1 Future work: Merging the Rankings 104
5.5 Implemented Features Not Used in the Work 105
5.5.1 LSA 105
5.5.2 Word Association Norms Integration 105
5.6 Recapitulation of Pipeline 105
5.7 Summary 105
Chapter VI: Requirements Capturing in Cases of Complex Communication Needs 107
6.1 Introduction 107
6.2 Aphasia 108
6.3 Challenges of Working with Participants with Aphasia 108
6.4 The Technology Appropriation Model 109

6.5 Methods 110
6.5.1 Participants 110
6.5.2 Meetings 111
6.5.3 Choosing TApM 111
6.6 Results 112
6.6.1 Identifying the Technology 112
6.6.2 Analysis of Level 1 User Behaviour: First contact with Technology 112
6.6.3 Analysis of Level 2 User Behaviour: The Start of the Appropriation Process 113
6.6.4 Analysis of Level 3 User Behaviour: Analysing Appropriation in Everyday Life 113
6.6.4.1 The use of notes 114
6.6.4.2 Combination of Text to Speech and Speech to Text 114
6.6.4.3 Picture based communication 114
6.6.4.4 Dictation 115
6.6.4.5 Supported Independence 115
6.6.4.6 A special use of YouTube 115
6.6.4.7 Capturing AAC Requirements Using Level 2-3 Behaviour 115
6.7 Discussion 117
6.8 Summary 118
6.9 Key Lessons 118
Chapter VII: Dictum AAC Application Design and Development 119
7.1 Introduction 119
7.1.1 Discussion on Design Decisions Based on the Works of Others 119
7.2 Dictum Development 122
7.2.1 Version 1 123
7.2.2 Meeting with Potential Users and Speech Professional at the British Aphasiology Society Research Update Meeting 2014 125
7.2.3 Version 2 125
7.2.4 Version 3 (No Natural Language Generation) 129
7.2.5 Version 3 (Inclusion of Natural Language Generation) 130
7.3 Summary 132

Chapter: VIII Natural Language Generation 133
8.1 Introduction 133
8.2 Natural Language Generation Server 133
8.2.1 The Compansion Algorithm in Detail 134
8.2.1.1 Introduction to the Compansion Algorithm 135
8.2.1.2 Word Categories and Semantic Taxonomies 136
8.2.1.3 Semantic Parsing 136
8.2.2 Dictionary Translation and Sentence Realisation 138
8.3 NLG Server 140
8.3.1 Comparison of the Algorithm Developed as Part of this Work and Compansion 140
8.3.2 Formal Language 144
8.4 WordNet Server 146
8.5 Theoretical Implementation of Additional Functionalities of the Natural Language Generation Server 148
8.5.1 Improving the Grammatical Correctness of Sentences Using other NLG techniques 149
8.5.1.1 Using sentence likelihood to fill syntactic gaps left by the current NLG server 150
8.5.1.2 Using Syntactic Parse Trees to Fill Syntactic Gaps Left by the Current NLG server: A Theoretical Investigation 150
8.5.1.3 Other Uses of WordNet to Improve Sentence Quality 151
8.5.1.4 Limitations of WordNet and possible solutions 152
8.6 Full NLG Pipeline 152
8.7 Summary 153
Chapter IX: Evaluation of AAC Application 154
9.1 Introduction 154
9.2 Qualitative Evaluation 155
9.2.1 Follow Up Meeting with Potential Users of the *Tap and Talk Group* at the University of Dundee 155
9.2.2 Feedback from Participants with Complex Communication Needs 155
9.2.3 Feedback from Speech and Language Therapists Present at the Meeting 156

9.3 Quantitative Evaluation 157
9.3.1 Overview of GridPlayer 157
9.3.2 Overview of Verbally 163
9.3.3 Evaluation of Dictum using the Keystroke Saving Model 165
9.3.3.1 The Keystroke Saving Model 165
9.3.3.2 Choosing Sentences 165
9.3.3.3 Measurement of Keystroke Saving using Food Category Sentences 167
9.3.3.4 Measurement of Keystroke Saving using Music Category Sentences 173
9.3.3.5 Measurement of Keystroke Saving using Sport Category Sentences 178
9.3.3.6 Measurement of Keystroke Saving using Transport Category Sentences 181
9.3.3.7 Results 184
9.3.4 Discussion on Strength and Weaknesses of the AAC Application Developed in this Work Based on the Results of Keystroke Saving Evaluation 189
9.3.5 Introduction to User Modelling Evaluation 190
9.3.6 Evaluation of Dictum using Fitts- Hyman law 193
9.3.6.1 Setting up the evaluation and calculating *a* and *b* values 195
9.3.6.2 Simulation Overview and Discussion on Biasing the Random Generator 198
9.3.6.3 Results 200
9.3.6.4 Introduction of Swipe Movements into Results 212
9.4 Final Discussion 214
Chapter X: Future Work 216
10.1 Introduction 216
10.2 Pre-processor 216
10.3 AAC Application 217
Chapter XI: Final Discussion 219
11.1 Discussion of Overall Results Given Proposed Hypotheses and Research Questions 219
11.1.1 Discussion on Hypotheses 219

11.1.2 Discussion on Research Questions 220
11.2 Final Discussion 221
References 224
Appendix 1

Abstract

Communication is recognised as a human right by the United Nations. Currently there are millions of people who for a variety of reasons cannot communicate comfortably. For example, in the UK alone there are 250,000 people who as a result of a stroke are now unable to communicate and are affected by a condition known as Aphasia. These people are said to have Complex Communication Needs. With the proliferation of smart devices like tablets and Smartphone, people with Complex Communication Needs are discovering the assistive potential of these devices to aid them either in the recuperation of their communicative abilities or to assist them in their daily lives. These systems are called Alternative and Augmentative Communication Systems.

However current AAC systems suffer from the fact that they are cumbersome to use and users require a long time to form sentences, with the result that they cannot confidently communicate and therefore are left isolated and frustrated. Even though much work has been done in the area these systems are remain slow and communication is not effective.

This thesis investigates whether the inclusion of Natural Language Processing and Natural Language Generation techniques into Augmentative and Alternative Communication systems on mobile devices can improve the ease and speed of use for users with Complex Communication Needs by implementing "Dictum" an AAC app which makes use of NLG/NLP techniques.

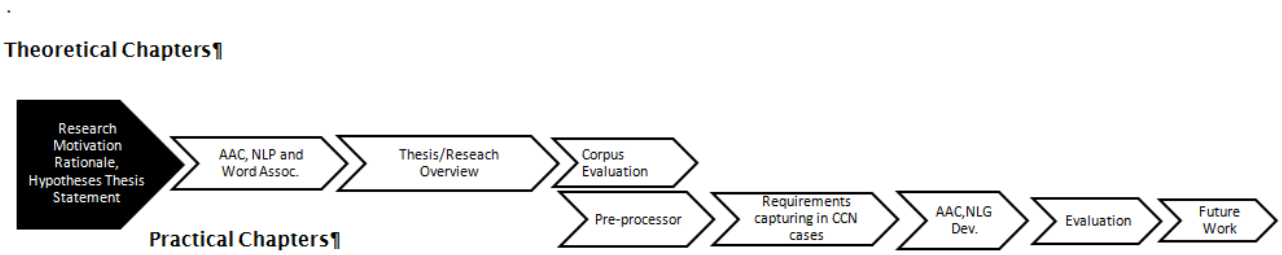
The work followed the approach of Action Research in which the target users help the investigator by identifying the problem, sanctioning the research and evaluating the results. Therefore, users were actively involved in the design of the application from the very start and gave feedback after each iteration leading to the final application.

This work has found that the inclusion of NLP and NLG in AAC does indeed improve ease and speed of use when compared to popular apps available today.

Dictum improves speed by doing two things: reducing the set space of words by providing words that are relevant to the last word inserted by using a Semantic Network of nouns and allowing the user to build sentences by requiring selection of

key words only and delegating the responsibility of sentence formation to the application itself. In addition, during the course of this work, an effective mechanism of capturing requirements for users with Complex Communication Needs discovered by looking at how users adapt the functionality of their devices.

The app was evaluated both quantitatively, by computing keystrokes savings and evaluating the interface using well-established HCI laws, and qualitatively by asking for the feedback of potential users and Speech and Language Therapists, following the practice of Action Research to involve those touched by the problem .



Chapter I: Background

1.1 Research Motivation

Communication is recognised as a human right by the United Nations[1] Currently millions of people worldwide have some trouble communicating[2]. The problem is so vast and the amount of causes of communication issues is so great that even finding exact statistics on the matter is difficult. The causes of speech and language issues are many and they can be either congenital or acquired later in life. With the proliferation of smart devices, users with special communicative needs are discovering the potential of these devices as assistive tools (Bradshaw, 2013; Flores et al., 2012; McNaughton and Light, 2013). Typically there are two categories of smart device applications for users who have special needs: Applications that assist the user in daily life and therapeutic applications. Additionally, special needs users sometimes adapt non assistive applications to suit their needs (Dix, 2007). Particularly interesting was the case of the game "fruit ninja" used by stroke affected players as a therapy tool (Fecht, 2014). The objective of the game is to slice fruit and vegetables by executing swiping motions on the surface of the screen, appearing at random locations on the screen of their mobile devices.

The research presented in this thesis focuses on how to improve current mobile communication systems for people with communication needs. These systems are classified as electronic Augmented (sometimes Augmentative) and Alternative Communication (AAC).

[1]http://staging.communicationmatters.org.uk/page/evidence-base
[2]http://www.nidcd.nih.gov/health/statistics/pages/vsl.aspx

1.2 Introduction

This Chapter will introduce the idea behind the research and justify via literature why mobile Augmentative and Alternative Communication (AAC) should be researched. It will be shown that the problem has not yet been solved. The problems highlighted by the AAC research community match those found by a Speech Therapist who was interviewed at the start of the research to gain insight into electronic AAC researchers.

The Chapter will conclude by arguing that a possible approach to improve the communication speed of mobile AAC apps is to merge the fields of Natural Language Processing, AAC and Text Entry. Linking these fields is the approach that is used in this work.

1.3 Rationale for Researching AAC

The aim of this paragraph is to discuss the reasons why it is important to research AAC. AAC is recognised as an official way of communication for those with speech issues by the United Nations (United Nations, 2006)

In 2013 the AAC-Rehabilitation and Research Centre (AAC-RERC) compiled a White Paper entitled "Mobile Devices and Communication Apps" in which they discuss the shift in the AAC world that the emergence of smart devices has caused. They state that "The proliferation of inexpensive mobile technology has dramatically changed the landscape for people with Complex Communication Needs" (CCN). The White Paper starts with a brief historical note which describes electronic AAC from their genesis, when AAC devices were displays where the user selected the desired option using eye tracking devices, moving to the 90s with the Personal Computer shift and moving on to when these devices eventually became portable, to today where more and more apps are being produced every day. Currently the majority of AAC apps are developed for iOS devices.

The White Paper lists the following advantages the shift to the mobile smart devices has had on AAC:

1. Increased social acceptance
2. A wide amount of apps available
3. Affordability as in the past some people with CCN were forced to ask for funding to buy AAC devices.

4. Non AAC apps can be useful to individuals with CCN
5. Inclusion of those who have only lost speech temporarily
6. The possibility for the user to have free trials and to decide whether to buy the app or not
7. Gentler learning curves

However, the White Paper also lists disadvantages of mobile AAC

1. Exclusion of those with cognitive impairments.
2. Lack of research, especially in cases where the AAC apps also serve to improve the language skills of the user
3. Lack of technical support
4. Attempting to cater for a large variety of symptoms or for varying degrees of the same symptoms is not an effective approach

The authors of the White Paper state that there will always be a need for specialized technology for users who have increased motor difficulties as part of their condition. The emergence of Smartphone and the likes has narrowed the access gap for those with severe motor impairments.

The White Paper concludes that there is little or no focus on research when developing commercial AAC applications and as a consequence there is little research available demonstrating the efficacy of the apps available on the market.

For these reasons there are some urgent questions that need to be addressed by researchers. These questions span many fields such as sociology, Speech and Language Therapy and Software Design/Engineering. For this work we are only interested in the latter. The authors propose the following research question: **"What Design Specifications should be included for those whose communication needs are poorly met by current AAC tech?"**

Traditionally the adoption of AAC by a patient (also referred to as AAC intervention) was made following an assessment by Speech and Language Professionals; now it is the patients themselves who download apps without consulting professionals and this can result in AAC apps which do not meet or only partially meet the needs of users. As McNaughton and Light do, AAC-RERC warn that AAC adoption should focus on the needs of the user and not on the capabilities of the device.

They conclude their work by proposing the following scenarios for researchers to focus on:

- The device meets most of the user's communication needs
- The device meets some of the user's needs and is used in conjunction with other devices. For example a secondary device is used when going out
- The device is used primarily for training and improving the user's language skills
- The user possesses a smart device but does not use it for verbal communication
- The user is unable to access the device for a variety of reasons, or he has a device but does not use it

Even with these problems the authors do acknowledge that the benefits of the inclusion of AAC into mobile devices outweigh the disadvantages: traditional AAC systems were "too heavy, too costly and sometimes too difficult to use".

The work by Bradshaw (Bradshaw, 2013) discusses the use of Apple devices as AAC devices. It concentrates on Proloquo2go and states that some of the barriers that were present with older AAC systems such as high costs and low acceptance by users are eliminated with iPods and iPads.

The paper concludes that there is still a lot to do and that users need individual assessments to determine whether AAC systems would be effective for them.

Advantages of the use of mobile technology as communicative tools are also highlighted by McBride (McBride, 2011) and by McNaughton and Light (McNaughton and Light, 2013). They agree with the previous paper in that mobile devices increase affordability and social acceptance. McNaughton and Light are particularly positive about the use of online resources such as blogs and technical forums also increases the diffusion and discussion of the latest AAC research developments, and it involves family members and patients into the process. McNaughton and Light however warn against the risk of losing the main focus and getting distracted by the device thus not providing effective AAC support. Also the full potential of current devices has to be exploited by AAC applications, not only the standard press and speak functionality. Things like cameras and sensors should be included.

McNaughton and Light's work concludes by stating that AAC apps development should follow an evidence based procedure and that applications should meet the individual needs of users, including factors such as linguistic competence. These apps should also consider the skills and knowledge of communication partners (the person or persons talking to the user).

Whilst Bradshaw, McBride, McNaughton and Light highlight the advantages of smart devices as communicative tools, Murphy et al. (Murphy et al., 1996) list some disadvantages of older AAC systems which have still not been resolved namely: lack of appropriate vocabularies and complicated systems which require training. However, one issue which noted by Murphy and colleagues which is being resolved by the emergence of apps is the lack of availability of AAC systems. Smith and Connolly (Smith and Connolly, 2008) identify issues with specialist electronic AAC devices by interviewing adults with experience using these devices. The problems listed by these users were: devices difficult to use without training, low reliability of device, limited vocabulary and concern about the communication partner's reactions about the slowness to form sentences and the lack of words.

Pennington et al. (Pennington et al., 2007) try to improve AAC intervention by focusing on the needs of the user. They conclude that the adoption of AAC is not always successful because AAC systems must be tailored to users whilst current trend is "one size fits all"

Clarke et al. (Clarke et al., 2001) discuss the views of young adults who used AAC before the emergence of Smartphone. They focus on three points: 1) operational difficulties, 2) self image 3) Perceived interaction benefit. There were no positive feelings towards operational difficulties in particular that electronic AAC systems were difficult to operate especially when looking for words. With respect to self image some users resented being the only ones using the device whilst others felt that their device improved their image by becoming their voice. Finally some users felt that AAC devices were not beneficial to interaction whilst others felt that the device improved their interaction with others especially because it "made people listen"

1.4 Motivational Interview with Glasgow Based Speech and Language Therapist

To get a better idea of current issues with AAC an interview was conducted with a Speech Therapist employed by NHS Greater Glasgow and Clyde. The interview lasted forty minutes and it focused on the use of mobile devices by patients who developed aphasia after their stroke and now use mobile devices to assist them communicating. It emerged that all patients used iPads and an app called GridPlayer. It is only outpatients who are allowed to use these devices due to information security protocols in force in hospitals. This means that users come into contact with mobile devices sometime after their admission. The priority of hospitals is to make the person self sufficient again before focusing on speech and communication. The main conclusion that was drawn from the conversation was that communicating via these applications is slow. The Speech Therapist made an example of people taking several minutes to form a sentence, with the consequence that the interlocutor lost interest or interrupted the speaker in an effort to help.

1.5 Methodology

Action Research (AR) is a critical enquiry process that involves diagnosing a problem situation, planning action steps and implementing and evaluating outcomes (Elden and Chisholm, 1993). AR is concerned with solving real world problems. AR is conducted through active participation of people in some way involved in a problem. Those involved help selecting the process sanctioning and evaluating the research. An example of AR in the work of others is the work of Grigis and Lazzari (Grigis and Lazzari, 2013). Grigis and Lazzari employed AR to design an AAC system with the help of potential users. The research focused in particular on rearranging the vocabulary for faster access to words.

1.6 Related Work

AAC Researchers feel there is still work to be done to improve speed and ease of use of electronic AAC systems. What follows will be a literature based discussion on what approaches are available to improve electronic AAC systems.

The review by Alm and Arnott (Arnott and Alm, 2013) provides an excellent starting point to discuss AAC improvement. It gives an overview of the different approaches to improving AAC from early attempts to the current day.

The main problem identified by all works cited by Arnott and Alm is that low rates of communication diminish user experience typical conversation rates are measured at 150-250 wpm. .Typical conversation rates of those who use AAC is 15 wpm or less; to date the most successful way to improve speed in AAC has been to include whole sentences in apps combined with phatic expressions. Phatic expressions are expressions used for general interaction such as greetings and farewells (e.g. hello, goodbye, how are you?)[3].

Alm and Arnott base their discussion on works that attempt to model conversation to improve AAC. Works discussed by them can be categorized in the following manner:

1. Modelling of conversation
2. Utterance based communication (where utterance means phrase)
3. Narrative based
4. Using NLP and NLG

1.6.1 Conversation and Utterance Based Research in AAC

Pre stored sentences or Utterances are currently the most effective and preferred way to improve speed of AAC apps (Todman et al., 2008a). AAC applications that use Utterances as input are called whole Utterance based application whereas AAC applications where the user has to form sentences word by word are called word based AAC (Todman et al., 2008a)

One of the most popular whole Utterance applications is CHAT (Todman, 2000) CHAT (Conversation Helped by Automatic Chat) utterances were preloaded into

[3]http://www.oxforddictionaries.com/definition/english/phatic

CHAT. A high number of sentences was included in the application to allow randomization. The system was designed for the following stages of conversation: greetings, small talk, closing and openings and farewells. The application also allows for backchannelling. Backchannels are short words that give feedback to the person to whom we are speaking by indicating that we have acknowledged what has been said (e.g. yeah, okay) Backchannels also give some indication on how we are feeling about the conversation.

Users of CHAT state that they had a more natural experience when using it compared to other apps as conversational speed was increased to 54 wpm.

Utterance based communication research also investigated storing phrases based on topic and type of phrases (e.g. joke, information, etc)

STANDUP (Arnott and Alm, 2013) for example was developed specifically for enabling users to make jokes.

Others have focused on storytelling with the goal of personalising by allowing the user to decide pace and rhythm of stories.

1.6.2 Topic Shifting, Semantic Networks and Hyperlinks

Hyperlinks were also proposed to navigate in AAC (Alm et al., 1990) although Alm states that generating appropriate hyperlinks for each user is an unfeasible task. The work explores the use of hypertext to improve AAC. The approach taken by Alm et al. differs from the one used in this work because Alm and his colleagues are interested in navigating whole sentences whilst the goal of this work is to navigate between key words as input to a NLG system .

Topic shifting is implemented in the TALK app (Arnott and Alm, 2013).

TALK design is based on the premise that social interaction is based on free flowing conversation. And that is why most AAC researchers are critical of pre-stored sentences.

The main focus of TALK is thus the social aspect of the conversation.

The main features of TALK are

- Opening and closing a conversation
- Back channelling
- Personalization
- Storing of pre-determined sentences.

Alm and Arnott report that Semantic Networks have been successfully used in the early 90s to navigate between sets of sentences and stories. Each sentence and story was given a specific semantic label so that the user could navigate between them.

SymbolPath (Wiegand and Patel, 2012a, 2012b) by Patel and Wiegand is an overlay python module for existing AAC applications that attempts to enable users to input words by tracing paths with their fingers on the screen much like Swype on mobile keyboard. SymbolPath arranges words to facilitate subject – verb- object construction, however words do not need to be in the correct order to form sentences. To achieve this, the application employs NLP/NLG techniques to generate all possible potential sentences given potential role of words and different order of words in the sentences.

1.6.3 Narrative Based AAC and Pragmatics Modelling AAC

Waller (Waller, 2006) gives an overview of how research has attempted to improve AAC system by focusing on narrative. Waller distinguishes communication into two categories: *transactional conversation* and *interactional conversation*. The former involves expressing needs, the latter involves for example telling stories, telling jokes or sharing experiences. Typically AAC apps focus on transactional conversation. Researchers in narrative AAC feel that improving interactional conversation in AAC systems can greatly improve communication for users, as it would enable them to tell stories about themselves and their daily experiences, thus improving the social aspect of communication for these users.

An example of narrative based AAC is the work by Black and others (Black et al., 2012, 2010) in which they attempt to support personal narrative for children with speech problems. Works like these put great emphasis on social aspects of conversation and are based on the argument that AAC research should be based on social aspects of conversation Modelling AAC apps around social aspects of conversation is also the focus of works by Todman and Alm (Todman and Alm, 2003) and McCoy et al. (McCoy et al., 2010). In practice they attempt to introduce

social aspects of conversation, formally referred to as *pragmatics*, into AAC sentence formation so that the user can answer appropriately in every social situations, as there are different ways of saying the same thing depending on who we are speaking to and where we are. In these approaches it is crucial that the app picks up what the other person is saying or at least pick up contextual information via sensors.

Todman and Alm (Todman and Alm, 2003) while discussing pragmatics approaches find that increasing speed of conversation alone within AAC would greatly increase social experience for users even without focusing on Narrative or Pragmatics aspects of conversation.

1.6.4 Other Work

Some works like Friend (Biswas and Samanta, 2008) also try to cater for users who have physical impairments as well as their speech impairments. The idea behind Friend is to provide a range of readymade interfaces for the user to choose from so that the application can both cater for the need of users with different disabilities and also adapt to users who have degenerative conditions so that they can use the same application throughout their lives.

1.7 How to Improve AAC

All works cited so far agree that the most strongly felt problem by users of electronic AAC systems is the slowness. This especially highlighted in studies by Todman and others (Todman, 2000; Todman et al., 2008b) which found that the faster the output from the app the more enjoyable it is for both parties in the conversation.

Todman summarises this in this statement**: Users are willing to give up on expressiveness in favour of speed.**

There is agreement among AAC researchers that the most effective approaches for improving speed and ease of use of electronic AAC systems: **storing pre-cached sentences and automatically generating sentences.** According to Todman, research should focus on devices capable of generating a reasonable approximation of natural conversation in favour of speed. Todman and others propose that sentence generation and pre-stored sentences should not be opposite but should be merged in the design of new apps. Again Todman summarises this in a sentence:

The main rule for AAC is that AAC should be appropriate and fast rather than ideal and slow.

Having the user slowly constructing the sentence from scratch is also not ideal. This is because, whilst constructing the sentence from scratch theoretically enables free flowing conversation, long pauses during sentences forming are not suited for conversation.

Johansen and Hansen (Johansen and Hansen, 2006) highlight that the goal of the AAC and Text Entry is the same:"**enabling the user to achieve the maximum input with minimum effort".** Therefore their study finds that linking the fields of Text Entry and AAC would be beneficial to both fields. The authors attempt to achieve the following in their work:

1. To find an alternative to QWERTY keyboard which requires less effort for the user to produce words and sentences. The authors state that since the English language has a high rate redundancy and has low entropy, QWERTY keyboard is highly ineffective. It has been calculated in work cited by the two authors that English has an entropy of one bit per character whereas each character press on a QWERTY keyboard requires 6.2 bits of information. In simple terms entropy is the average bits produced for each letter in the English language.
2. Faster production of conversational text. By this Johansen and Hansen mean that Natural Language Processing Techniques should be exploited to produce output that closely resembles human conversation, much like the work discussed earlier in this Chapter.
3. Focus on speed rather than accuracy. Text Entry Systems that rely on single character inputs are error prone; to mitigate this, systems today and those like the ones proposed by Johansen and Hansen attempt to tackle or mitigate this by automatic error correction. It will be shown that the work proposed in this thesis attempts to produce an error free system that does not rely on single character input, but rather on whole words input.
4. Focus on multimodal interaction for users who have motor disabilities coupled with their CCNs

The study by Johansen and Hansen focuses on users with ALS. They propose a system that can adapt to the degenerative nature of the condition. They conclude that the goal of AAC and Text Entry is one and the same (i.e. how to input text with the least effort and the most user friendly manner).

This overview of literature has shown that integrating Natural Language Processing can be a way to improve ease of use and speed of electronic Augmentative and Alternative Communication Systems. The work of others suggests also that NLP should be used in conjunction with current Text Entry techniques since researchers in both fields are working towards the same goal: maximum output with minimum effort.

It can also be concluded that the popular approach when applying NLP is that of updating the words dynamically based on previous input, as the review by Alm and Arnott (Arnott and Alm, 2013) found.

Having reached these conclusions the thesis statement and hypotheses will now be given.

1.8 Thesis Statement

The aim of this thesis is to research whether the integration of Natural Language Processing, Natural Language Generation and Text Entry can improve speed and ease of sentence formation for users with Complex Communication Needs who require electronic Augmentative and Alternative Communication software to communicate with others.

1.9 Research Approach

Given that the main problem of AAC system is slowness of input and that improving speed of communication would also improve social aspects of communications for users of AAC, and given that the work of others has found that applying NLP, NLG and Text Entry to AAC and also found that the use of Semantic Networks has been previously successful in improving speed and ease of use of electronic AAC, the research in this thesis will have the following approach:

1) Develop a Grid based AAC Application similar to GridPlayer (as discussed with SLT in Glasgow)
2) Research and use Natural Language Processing algorithms to create a network of related words and reduce search space to use within the developed grid based app. In practice when inserting the words required to form the sentence, words in AAC application should change dynamically offering related words based on the last word inserted
3) Research and use Natural Language Generation techniques and Text Entry techniques to enable users of the AAC application to form sentences using only key words regardless of the order they are inserted in, theoretically increasing speed and ease of use of the application

1.10 Hypotheses

H1. The integration of Natural Language Processing and Natural Language Generation with Augmentative and Alternative Communication will reduce the number of keys required to form sentences compared with competitor apps leading to the potential improvement in speed of sentence formation.

H2. The integration of Natural Language Processing and Natural Language Generation with mobile Augmentative and Alternative Communication systems will reduce screen navigation thereby improving ease of sentence formation by avoiding mental overload and stress on users.

1.11 Research Questions

Q1. Will the integration of Natural Language Processing and Natural Language Generation improve speed of sentence formation in mobile AAC systems by reducing the number of keys required to form sentences?

Q2. Will the integration of Natural Language Processing and Natural Language Generation improve ease of sentence formation in mobile AAC systems by reducing stress and avoid mental overload?

Q3. Can a system that excludes keyboard and relies on whole words input work as an effective input mechanism in mobile AAC applications?

Q4. Can we discover requirements through the use of User Centred Design in cases where the users have Complex Communication Needs?

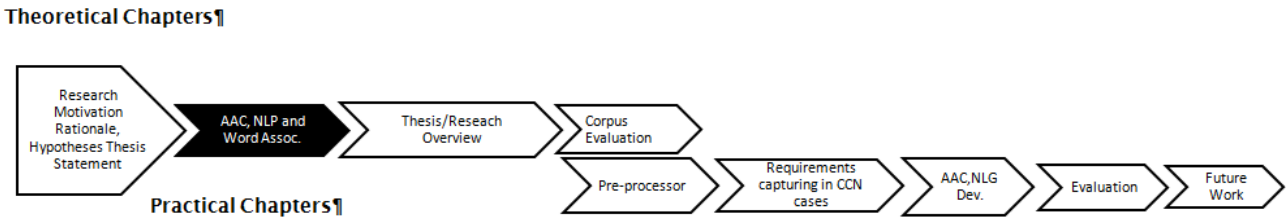
1.12 Summary

This Chapter has shown that there is great potential for AAC to assist those with Complex Communication Needs. The work of others has shown that there is lack of research in the area which is needed both to give the best solution for the user's conditions and to improve electronic systems themselves, which still suffer from being slow and do not exploit the full potential of the devices they are being made for.

A way to exploit this potential is to employ Natural Language Processing and Natural Language Generation techniques alongside with text entry techniques which some argue are one and the same and should be considered as one field.

1.13 Chapter Conclusions

1. AAC apps can help a wide variety of people with special needs commonly referred to as people with Complex Communication Needs.
2. It is the goal of this work to attempt to increase the speed and ease of use of AAC applications
3. Increasing the speed of AAC applications will also improve social aspects of conversation for users even if the design of the application does not focus on pragmatics
4. It is the goal of this work to develop a grid based electronic AAC system to test the given hypotheses by the integration of NLP,NLG and Text Entry with the application itself
5. The use of a Semantic Network of words to help the user navigate through the AAC application vocabulary could increase sentence formation speed
6. The use of NLG to help with sentence formation could increase sentence formation speed



Chapter II: Background

2.1 Introduction

The first Chapter justified why it is important to research AAC and highlighted the current limitations of AAC systems. Chapter I concluded that the main problem to resolve within electronic AAC systems is the speed of sentence formation which would not only improve communication but also social aspects for users.

Thus, Chapter I set out the main goal of this thesis: improving the ease and speed of current AAC applications. The work of others has shown that a promising approach to achieve this is the use of Natural Language Processing, Natural Language Generation and Advanced Text Entry techniques.

This Chapter will be a purely theoretical Chapter which will discuss in detail all theory required to understand the Natural Language Processing, Natural Language Generation and Text Entry techniques employed in this work to attempt to improve current AAC systems.

Firstly this Chapter will give a formal definition and taxonomy of AAC systems. It will then discuss the theory exploited in this work starting with a discussion on NLP with particular emphasis on Word Association Measures, as they were used to generate key files for the AAC application also developed as part of this work. The Chapter will conclude with a discussion on Natural Language Generation and introduce the reader to the Compansion algorithm which will be used as a blueprint to create an algorithm to enable users to generate sentences providing only key words.

2.2 Augmentative and Alternative Communication Systems

According to the International Encyclopaedia of Rehabilitation ("Augmentative and Alternative Communication | International Encyclopedia of Rehabilitation," 2010) AAC can so be defined as "the effort to study and when necessary compensate for temporary or permanent impairments […] of persons with severe disorders of speech and language production or comprehension".

AAC can be used to:

1. Augment the speech of individuals with disfluent or unintelligible speech
2. An alternative method of speech for individuals who lack speech or have insufficient amount of speech for effective communication

The term for those who require AAC is "people with Complex Communication Needs"

People with CCN include those with:

- Autism
- Cerebral palsy
- Apraxia – A motor disorder in which someone has difficulty with planning to produce speech
- Those with acquired conditions: Acquired Brain Injury, Neurological Disorders, victims of Stroke or Spinal Cord Injury, other progressive conditions.

There are two types of AAC:

Unaided AAC: manual signs and gestures

Aided AAC: requires supplemental materials. Electronic AAC such as the one developed in this work is classed as aided AAC.

"AAC includes any method of communication that supplements (augments) or replaces (provides an alternative to) the usual methods of speech and/or writing where these are impaired or insufficient to meet the individual's needs."("Augmentative and Alternative Communication | International Encyclopedia of Rehabilitation," 2010) are used by any category of user that has difficulties communicating. Literature refers to these users as "Users with Complex Communication Needs" (CCN).

AAC systems are comprised of four components (Murray and Goldbart, 2009):

> The *mode* describes the system used and the method used to convey the message. (e.g. AAC app, AAC book).
>
> The *means* describes the way in which the user accesses the mode. (e.g. pressing on keys, pointing etc.); there are many cases in which the means is indirect if the user needs the use of switches or other aids.
>
> The *representational system* describes the way we represent concepts that the user can communicate. (e.g. words or phrases).
>
> *Interaction strategies* describe the strategies that user adopt when using AAC systems (e.g. pre-storing sentences, saving names of pets and family members).

AAC mean **any** form of alternative communication, thus, mobile AAC is only a fraction of AAC. The most popular form of AAC are cardboard booklets (image below) given to patients in hospitals when they have trouble speaking after a stroke (Murray and Goldbart, 2009).

Bradshaw (Bradshaw, 2013) provides the following taxonomy of AAC applications

> *Text to speech*: apps that translate written documents to speech
>
> *Grid based systems*: apps that display words depicted by pictures, arranged in grid.
>
> *Word predictors*: apps that predict words based on keys typed by user.
>
> *Phrases*: apps that contained pre-set and/or customisable phrases for the user to select when needed.
>
> *Eye pointing*: these apps are designed to be used for people who use eye pointing as their primary input method.
>
> *Photo story*: apps that offer the possibility to take photos and include them in a slide show to tell a story.
>
> *Picture Exchange Communication Systems (PECS)*: these allow users to exchange pictures as a means of communication.

Diagram 1 provides a pictorial representation of all categories of AAC applications. The diagram was designed by researchers at the University of Edinburgh. The next paragraph will give an overview of commercial AAC apps available today on digital markets. All of the apps discussed and Dictum: the app developed as part of this thesis can be framed on the upper and centre right side of Diagram 1.

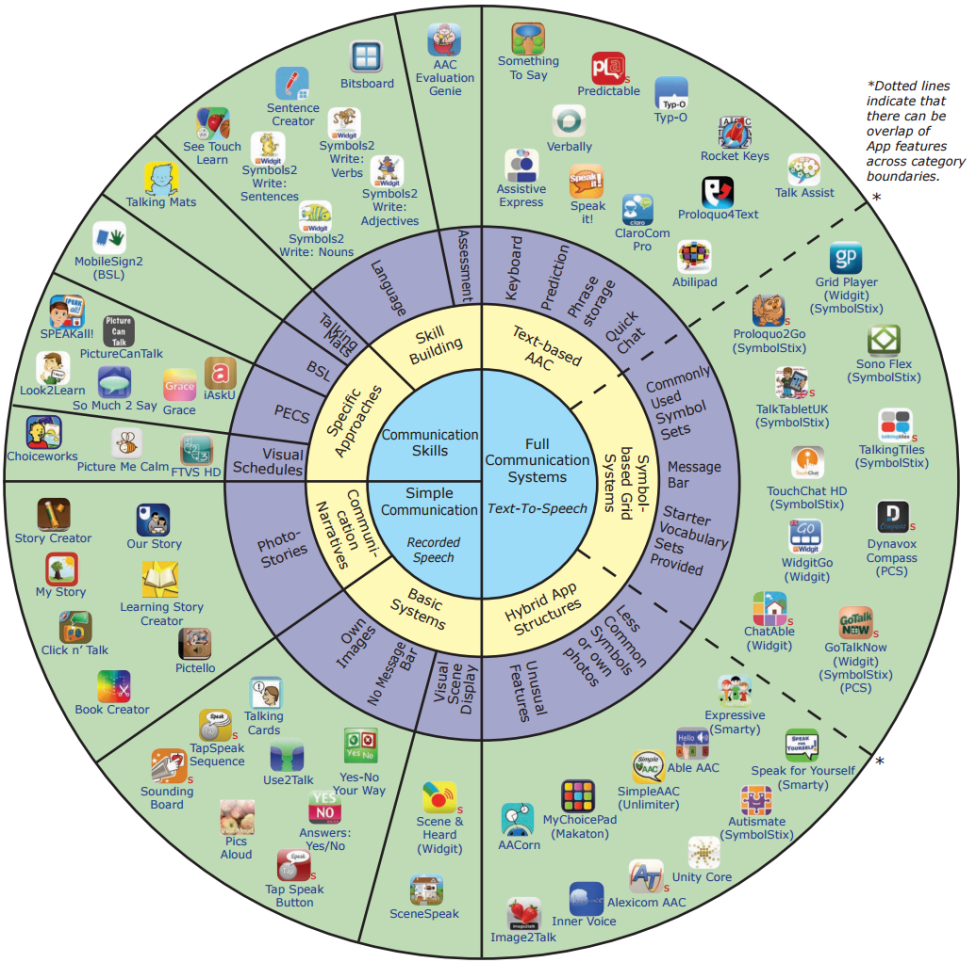


Diagram 1 A Diagram of The Most Popular AAC Apps by the University of Edinburgh[4]

[4]http://www.callscotland.org.uk/Common-Assets/ckfinder/userfiles/files/iPad-Apps-for-Complex-Communication-Support-Needs.pdf

Figure 1 AAC Cardboard book[5]

2.2.1 Overview of AAC Apps Available Today

This paragraph will give an overview of popular commercial AAC applications which were mentioned by both potential users and SLT professionals and are most recurring in electronic AAC review sites[6]. available today. These will be framed using Bradshaw's taxonomy of AAC applications.

[5]https://upload.wikimedia.org/wikipedia/commons/c/c3/Sample_page_from_AAC_communication_book.png
[6]http://appsforaac.net/

2.2.1.1 DynaVox Devices and Software

Image 1 Photograph of a DynaVox Device

DynaVox is a company that specialises in construction of Speech Generating Devices and software. DynaVox devices are structured using a hierarchical set of pages that contain different vocabulary for different contexts or situations. DynaVox's particular system of organising these pages is called 'InterAACT'.[7] DynaVox also focuses on catering for the needs of those who require alternative input mechanisms, in particular eye tracking and switches. In origin DynaVox produced bespoke devices to go with their software, however today with the emergence of smart devices, DynaVox has made their software installable on smart devices.

2.2.1.2 GridPlayer

GridPlayer is a grid based AAC system containing single words and phrases that was developed only for iOS devices. It comes both free and with a charge depending on the amount of customisation the user wants to perform. The app is organised into grid sets If the user wants to customise his grid sets he can buy an online account where customisations are stored. This app was mentioned during the initial meeting with the Speech and Language Therapist. The SLT stated that this app is popular among patients as it offers a wide number of words and phrases free.

The basic GridPlayer setting has four grid sets (shown in image below)

[7] http://www.speechbubble.org.uk/vocabulary/interaact

- Symbol Talker a
- Symbol Talker b
- Talking photos
    - Picture based communication
- Text talker phrasebook
    - A set of phatic expressions

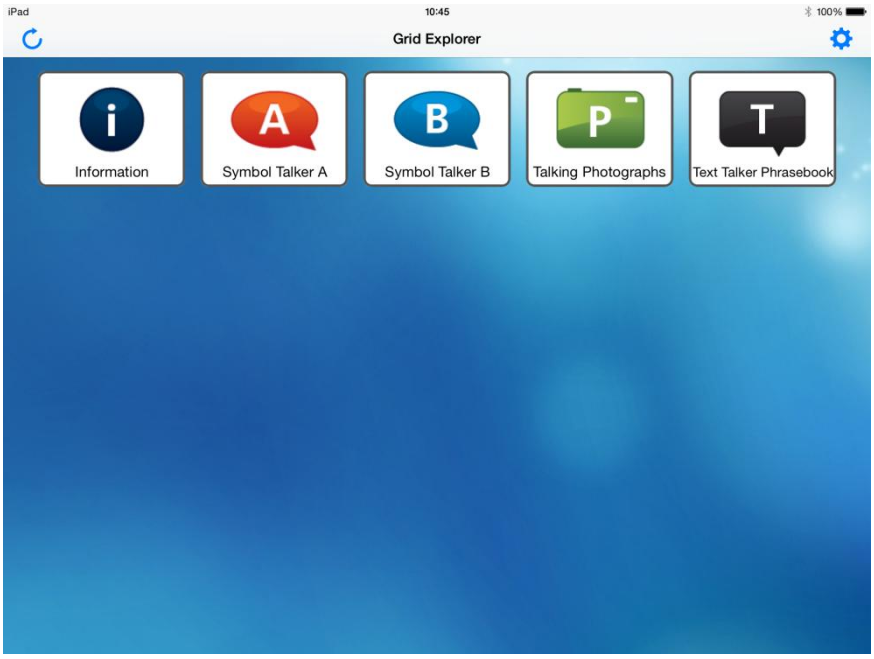


Image 2 GridPlayer App Main Screen

It gives the user the option to choose between male and female voice and includes basic text prediction (most likely next word). The text prediction is the standard one used by the iPhone.

2.2.1.3 Proloquo2Go

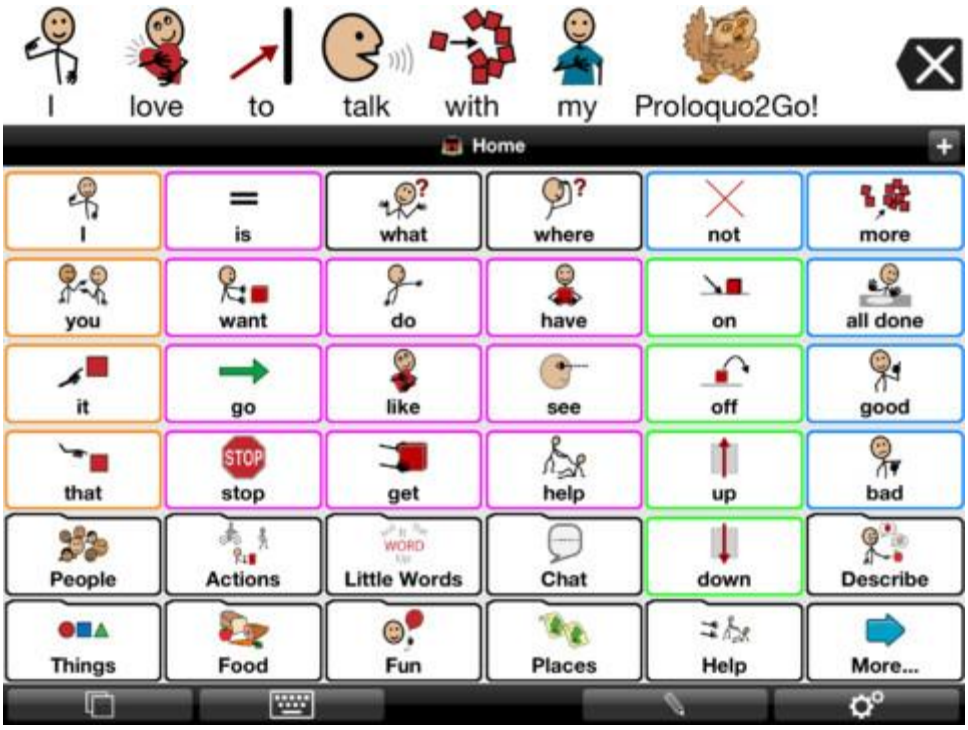


Image 3 Screenshot of Proloquo2Go[8]

Proloquo2Go costs 190 $ (as of 2014) and is a grid based system, which uses phrases and text prediction. The app contains 3 vocabularies for different levels. The grid size can be customised and has 23 different grid sizes (from 9 to 64 elements). Users can also add personal photos The vocabularies can be customised for multiple users. Popups appear for selections to reduce navigation. The app contains basic bigram text prediction and the ability for users to store previous sentences. Finally, the app can be used with alternative input mechanisms for those with physical impairments.

[8]http://www.friendshipcircle.org/apps/wp-content/uploads/2012/12/proloquo2go_5.jpg

2.2.1.4 RocketKeys

Shown in the image below; RocketKeys is a customisable talking keyboard. The app lets users build customised keyboards by choosing the exact keys, size, layout, colours, prediction, and voices.

Image 4 Screenshot of RocketKeys[9]

RocketKeys features basic bigram prediction which is updated daily via Twitter feed analysis. This also enables the user to talk about current trends and news. RocketKeys' prediction engine also corrects misspelled words. Users can talk with four different voices (old/young male/female) of which the speed, pitch and tone can be regulated. The app speaks on three different occasions, when talking, when the user taps a button or to prompt the user. Different voices can be used for each of these scenarios. The interface assists the user with a cursor to show touch points.

[9] http://a2.mzstatic.com/us/r30/Purple6/v4/3e/86/54/3e86546a-41a8-6a6e-cd6a-dcd6a38890f9/screen480x480.jpeg

2.2.1.5 OneVoice

Image 5 Screenshot of OneVoice App[10]

OneVoice focuses on children and comes with a pre-populated icon-based vocabulary. Like the other apps it relies on customisation to improve speed and to personalise the vocabulary to each user. Users can also customise the layout of the interface via drag and drop. Finally, OneVoice can speak using two male and two female voices.

[10]http://www.blogcdn.com//media/2012/05/onevoice52212.jpg

2.2.1.6 EasySpeak

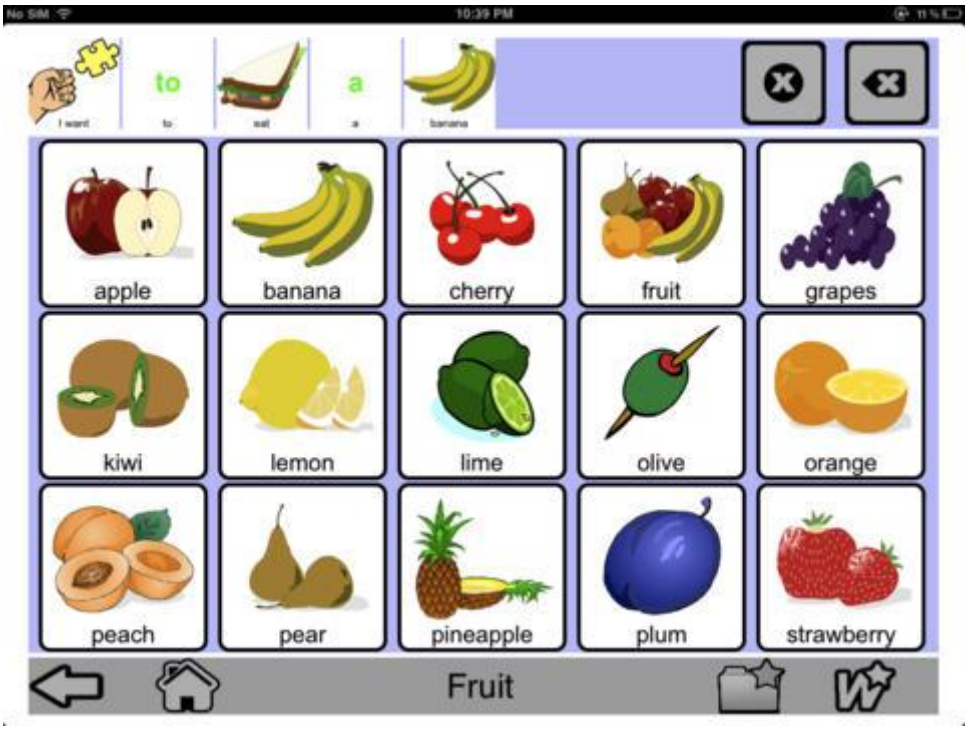


Image 6 Screenshot of EasySpeak[11]

EasySpeak contains 800 symbols. Like all AAC apps users can build sentences by selecting a combination of pictures. Users can customise by adding their own symbols with personalised sentences. Users can also organise the grid to suit their needs.

[11]http://a1.mzstatic.com/us/r30/Purple/v4/21/d3/e5/21d3e593-2f57-4220-c7b9-cb6e26f8094a/screen480x480.jpeg

2.2.1.7 TalkRocket Go

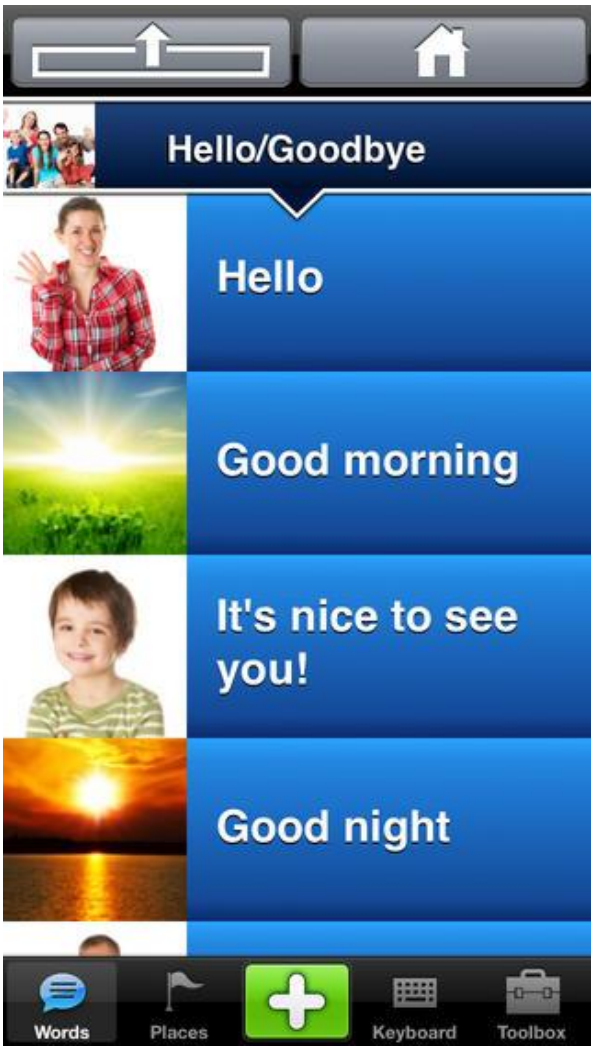


Image 7 Screenshot of TalkRocket Go[12]

TalkRocket Go is an icon-based app that enables users to communicate using pre-stored sentences. It also exploits location awareness to contextualise sentences. The interface and the sentences can be customised via web browser. TalkRocket Go is one of the few apps also available for Android.

[12]http://www.get-app-and-go.com/images/talkrocket.jpeg

2.2.1.8 Assistive Chat

Image 8 Assistive Chat Screenshot[13]

Assistive Chat is a keyboard-based app with bigram text prediction that expands with new words like standard predictive keyboards on Smartphone. Words can be put in a favourites list for quicker access. Like other AAC apps listed so far voices can be customised and Assistive Chat also enables the user to regulate speed, pitch and tone of voices.

[13]http://www.iautism.info/en/2011/02/26/review-assistive-chat/

2.2.1.9 Verbally

Image 9 Screenshot of Verbally[14]

Verbally, like Assistive Chat is a keyboard based app that contains fifty essential words and phatic expressions, such as greetings, basic requests and farewells. As users type, the app attempts to predict the next likely word. The keyboard can be configured to be either QWERTY or ABCD. Like other apps, users can select between male and female voices. Verbally also has the option to speak after each word is selected or to speak when the user presses the "speak" button. To avoid mistakes the app only registers input when the user lifts a finger off the screen. As an additional communication mechanism users can also select a chime sound to alert others.

The work of others cited in Chapter I has concluded that a way to improve AAC systems speed and ease of use can be to integrate Natural Language Processing techniques to AAC systems. Now that AAC systems have been discussed, the next paragraph will discuss Natural Language Processing with an emphasis on techniques used to measure word associations. As explained in Chapter I one of the approaches this work will take is to create a Semantic Network of words to limit search space of words for users thus easing the insertion process.

For the AAC app developed in this thesis it was decided to develop an Android OS app. Although the majority of commercial apps has been developed for iOS it, Android has been used for AAC research in the work of others. For example in their

[14] https://itunes.apple.com/us/app/verbally/id418671377?mt=8

work Grigis and Lazzari (Grigis and Lazzari, 2013) chose to use Android OS apps due to its diffusion and the increased chance of finding free apps on the Google Play market. Benefits of using Android OS for AAC are also noted by Higginbotham and Jacobs (Higginbotham and Jacobs, 2011)

2.3 Natural Language Processing

Natural Language Processing (NLP) is an umbrella term for a field of computer science that studies every aspect of human language, from information extraction using speech and text to the generation of new text based on some input (Chowdhury, 2003; Jurafsky and Martin, 2000; Manning and Schütze, 1999).

Liddy's review of NLP (Liddy, 2001) lists some goals of NLP research today: information extraction used to collect key information from a large text collection; question answering to give meaningful answers to a user's questions, summarisation and translation of documents. Finally one of the goals of future NLP systems: interactive dialogue systems that are capable of having conversations with users.

There are different approaches to NLP (Liddy, 2001) :

Symbolic: symbolic approaches try to analyse language by using human knowledge. Examples of symbolic approaches are Semantic Networks generated by humans. Semantic Networks are networks that connect words by their association either of meaning or association that words have because they make us think of other words.

Statistical: statistical NLP approaches employ mathematical techniques to analyse language (Liddy, 2001; Manning and Schütze, 1999). Statistical NLP approaches require the use of large corpora. A popular example of statistical NLP is Hidden Markov Models HMM. HMM are finite state machines that rely on probabilities to transition between states. HMM are used in Text Entry and Speech Recognition to determine the most likely next word.

Connectionist: connectionist approaches apply machine learning to NLP by using techniques such as neural networks to train computers to analyse language (Wermter et al., 1995). Wermter and his colleagues make the example of a neural network that was successfully trained to distinguish between grammatically correct and incorrect sentences.

Hybrid :Hybrid NLP techniques employ a combination of these approaches.

In this work, NLP will be required to find related sentences to build a Semantic Network, in order to make the search for words easier for the user. For the purposes of the work discussed here the Statistical Approach is the most suited.(Liddy, 2001). The statistical NLP techniques are also key in the steps required to generate text from compressed input. Text generation will be discussed theoretically at the end of this Chapter, and practically in Chapter IX. In Chapter IX, it could be argued that an hybrid approach is used as the system employs WordNet which is a knowledge based source.

Input in statistical NLP systems can be either in spoken form or in written form (Jurafsky and Martin, 2000). For the purposes of this work a collection of text will be used as input for the NLP system which will be developed as part of this work. As mentioned, Statistical NLP tasks require large collections of texts, (corpora) to analyse or produce language. . The next paragraphs will give an overview of the typical steps required for NLP processes.

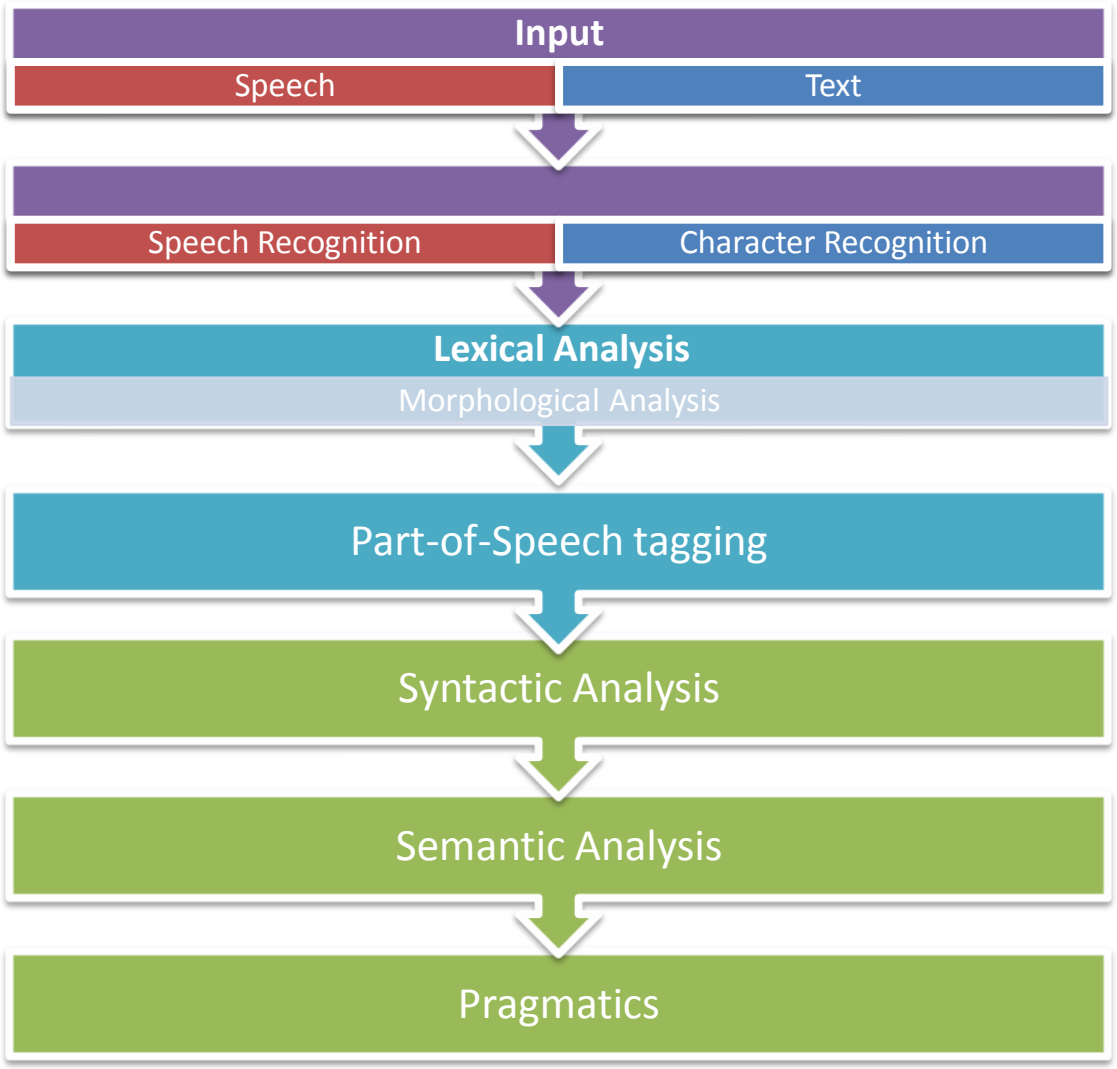


Figure 2 NLP System Architecture

The diagram above, adapted from (Manning and Schütze, 1999) is the most recurring in NLP reference texts like (Chowdhury, 2003; Jurafsky and Martin, 2000). Some texts add another layer for language generation based on input. This layer is present in systems that generate a dynamic response based on the words spoken by users.

2.3.1 Character Recognition and Tokenisation

The first part of any NLP task is called tokenisation and it involves breaking up the input into meaningful chunks (Jurafsky and Martin, 2000). The simplest scenario is when text files are used as input. If text input is physical documents then character recognition is also required, today this step is seldom required as the majority of corpora have been digitalised. Typically Regexes are used to perform tokenisation to break up the input into words and to remove unnecessary symbols such as

numbers and punctuation marks(Jurafsky and Martin, 2000) The task of tokenisation is harder in scenarios where tagged input is used (e.g. HTML pages or XML documents), in this cases an ad-hoc parser may be required to collect the data.

2.3.2 Morphological Analysis

The next step of the pipeline is Morphological analysis and Part of Speech tagging (Jurafsky and Martin, 2000; Liddy, 2001). Morphological Analysis is the step of breaking the word into morphemes. Morphemes are the smallest units of words. For instance for the word *breaking* its morphemes would be *break* and *ing*. NLP uses morphemes to gather information about individual words to contribute to understand their meaning within the sentence (Liddy, 2001). Liddy makes the practical example of the suffix *ed* in verbs. This suffix in verbs always means that an action took place in the past.

2.3.3 Lexical Analysis and Part of Speech Tagging

Lexical Analysis deals with individual words. Like morphological analysis, lexical analysis can contribute to interpret the meaning of words within sentences (Liddy, 2001). At this level many techniques exist to analyse each word. An example technique is Part of Speech tagging. Part of Speech tagging entails determining the correct Part of Speech for words in a sentence and assigning a tag to these (Jurafsky and Martin, 2000). The image below shows an example of POS tagging. POS abbreviations in the images mean the following

PRO = pronoun

V = verb

DET = determiner

PREP = preposition

N = noun

Figure 3 Example of Part of Speech Tagging

Part of Speech tagging is necessary to disambiguate in cases where words have multiple parts of speech, for example are both verbs and nouns. Part of Speech

tagging can be performed manually, to generate error-free datasets that can be used as training for automatic POS tagging algorithms, or automatically by a previously trained computer (Liddy, 2001).

2.3.4 Syntactic Analysis

Syntactic Analysis is used to find the grammatical role of a word or phrase within a sentence (Liddy, 2001);. for example to find whether a word in the sentence is a noun phrase or a verb phrase .The process of Syntactic Analysis requires the use of a parser and a grammar. There exist a variety of parsers and grammars depending on the language that is being parsed (Jurafsky and Martin, 2000). The output of this process is shown in the image below:

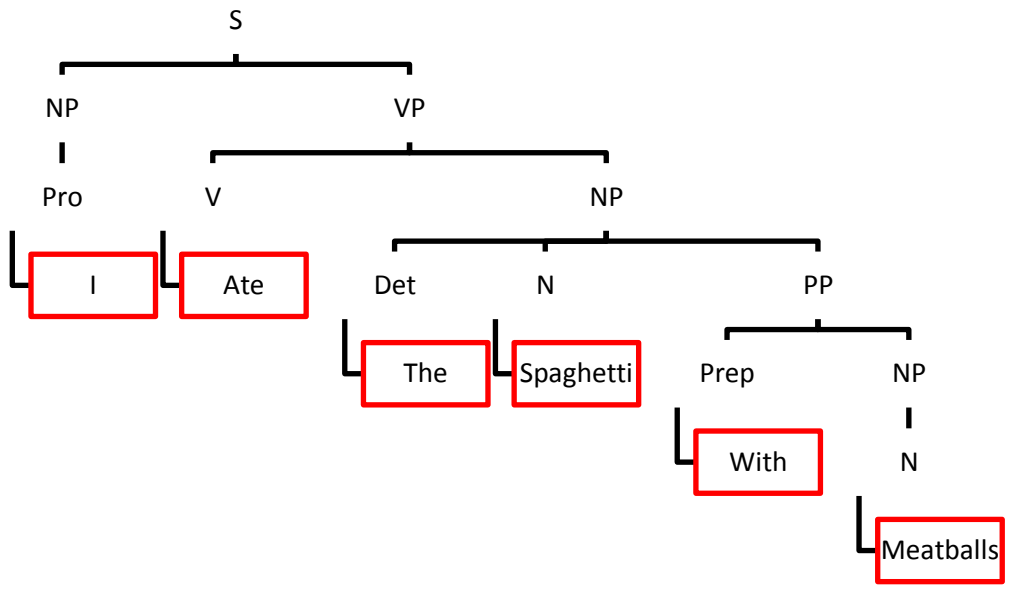


Figure 4 Example of Syntactic Parse Tree from (Jurafsky and Martin, 2000)

2.3.5 Semantic Analysis

Semantic Analysis is concerned with finding the meaning of a word within a sentence or text (Jurafsky and Martin, 2000) by examining interactions between words in sentences. One popular technique for Semantic Analysis is Word Sense Disambiguation (WSD) where a system attempts to assign the correct meaning to the sentence. for example the statistical approach to WSD looks at the frequency of

each sense of each word in a sense occurring in a corpus to assign a probability of each word having a particular meaning.

Word Sense Disambiguation works using another popular NLP technique called Semantic Role Labelling (SRL). The aim of semantic role labelling is to appropriately detect and assign semantic roles for each word in a sentence.

Below are some examples of semantic roles:

ACTOR

ENVIRONMENT

THEME

LOCATION

An application can make use of semantic roles in two ways : WSD , or exploiting word knowledge (in conjunction with other steps) to generate a sentences. Semantic Role Labelling is the central part of an algorithm called Compansion. Compansion will be discussed later in this Chapter.

2.4 Word Association Measures

One of the aims of this thesis is to find a way to present the user with words that are related to the last word selected. It was said that to achieve this, Statistical NLP techniques are most suited. The discussion that follows is based on the work of others and it will examine the array of measures for word relatedness to show how one was selected to achieve the intended aim.

Calculations to find related words are referred to as Word Similarity Measures (WSM) or Word Associations Measures (WAM) (Washtell and Markert, 2009), (Chaudhari et al., 2011).

Word associations are of two kinds : Paradigmatic Associations and Syntagmatic Associations. Rapp (Rapp, 2002) defines Paradigmatic Associations as words that are similar to each other or words that are linguistically related to each other such as meronyms , antonyms and synonyms; and Syntagmatic associations as words that invoke (make us think of) another word but cannot be said to be formally similar.

Washtell and Markert classify WAMs in three types:

- Measures based on co-occurrence frequency
- Measures based on distance between words
- Measures based on syntactic relations

Damani (Damani, 2013) classifies measures in a different fashion from Washtell:

Knowledge based: based on thesauri, Semantic Networks and other knowledge sources.

Distribution based: assessing similarity by comparing the distribution of nearby words.

Co-occurrence based : based on counting occurrence of words appearing together

2.4.1 Measures Based on Co–Occurrence Frequency

Co occurrence is defined as the appearance of a pair of words in text (Manning and Schütze, 1999). Co-occurrence can be measured at document level (the number of times words co-occur in a document, summed up over the corpus) or at corpus level (the number of documents in which the co-occurrence occurs).

Co-occurrence based measures are the standard for finding Syntagmatic Associations The simplest and most frequent in literature is Pointwise Mutual Information (discussed later); others include the t-score, the z-score, Chi Squared (χ^2) shown in Table. Chi-squared and T-test suffer from the fact that they assume normal distribution of words (Dunning, 1993) and therefore fail because distribution of words in English is not normal. Words in fact are distributed in Zipfian fashion (Zipf, 1935). Moore (Moore, 2004) explains Zipfian distribution and shows that for a 500.000 word corpus 60.5% of words occur 5 times or less and 32.8% of words occur only once. Moore uses this example to criticise the log likelihood ratio as he finds it gives high values to rare occurrences of words.

LLR	$\sum_{\substack{x' \in \{x, \neg x\} \\ y' \in \{y, \neg y\}}} p(x', y') \log \frac{p(x', y')}{p(x')p(y')}$
Pearson's χ^2 squared test	$\sum_{\substack{x' \in \{x, \neg x\} \\ y' \in \{y, \neg y\}}} \frac{\left(\hat{f}(x', y') - E\hat{f}(x', y')\right)^2}{E\hat{f}(x', y')}$
PMI	$\log \frac{p(x, y)}{p(x)p(y)}$
T-test	$\frac{\left(\hat{f}(x', y') - E\hat{f}(x', y')\right)^2}{\sqrt{\hat{f}(x', y')(1 - \frac{\hat{f}(x', y')}{N})}}$

N	Total number of tokens in the corpus
$f(x), f(y)$	Unigram frequencies of x,y in the corpus
$p(x), p(y)$	$f(x)/N$, $f(x)/N$
$\hat{f}(x', y')$	Span constrained (x,y) word pair frequency in corpus
$\hat{p}(x, y)$	$\hat{f}(x', y')/N$

Table 1 Examples of Co-Occurrence Based Measures from (Chaudhari et al., 2011)

Co-occurrence measures are computed by using word windows. A window between words is formally defined as the direction-independent distance between the occurrences of the two words.

More simply windows are the number of words before and after a word for which we are trying to find related words. Target Words are in practice the lemmas in a document.

2.4.1.1 Mutual Information and Pointwise Mutual Information (PMI)

$$\mathrm{PMI} = \frac{\mathrm{p(x, y)}}{\mathrm{p(x)} * \mathrm{p(y)}}$$

Equation 1 Pointwise Mutual Information

The reference work by Manning and Schütze (Manning and Schütze, 1999) defines mutual information as "an information theoretically motivated measure for discovering interesting collocations. PMI was first introduced by Church and Hanks in (Church and Hanks, 1990). PMI was created to create an objective measurement to find word associations that closely resembled word associations generated by humans. These associations are called Word Association Norms. (WANs). Typically these are generated by asking a person the first words that come to their mind when they hear a cue word .However, because they are generated by humans, according to Church and Hanks WANs have the limitation of being

subjective. More simply if 100 people are asked to give associated words to a specific word it is more than likely that they will come up with 100 different or slightly differing answers. Therefore the creation of PMI was an attempt to find an objective measurement of word associations.

Manning and Schütze formally define PMI as "**The amount of information provided by the occurrence of the event represented by y, about the occurrence of the event represented by x [where x is the first word and y is the second word]"**

Manning and Schütze also explain the limitation of PMI. The measure does not work very well with low frequency events (words which occur rarely). For sparse words PMI tends to spike. To explain this limitation they give the example of extreme cases: the case of perfect dependence, when two words always occur together, and the case of perfect independence when the occurrence of one word does not give any information about the other word. It can be said that PMI is a good measure for independence but not for dependent words because the final score depends on the frequency of terms (low frequencies produce a low denominator and therefore a high result).

2.4.1.2 Significance Based Alternatives to PMI

To overcome the limitation of PMI several works introduce new measures that use statistical significance to determine whether the co-occurrence of two words is merely random or if it is significant. The next paragraphs will discuss the most prevalent in literature.

2.4.1.2.1 CSR and cPMId

Damani in (Damani, 2013) introduces a new word association measure called cPMId. cPMId is based on the concept of Co-occurrence Significance Ratio (CSR) introduced by Chaudhari in previous work (Chaudhari et al., 2011). CSR and cPMId classify significance in two different ways: corpus level significance, determined by counting co-occurrence frequencies across a whole corpus. And document level significance, determined by counting co-occurrence of words within a single document.

CSR determines document level significance of a span constrained word pair by comparing the frequency of the word pair in the document and the expected frequency of the same word pair in the document when the words are shuffled in random order. If the frequency of the pair within the document is greater than the

expected one in the random document the frequency is considered to be significant and not the result of chance. CSR determines corpus level significance by calculating the ratio of actual co-occurrence across the corpus and expected co-occurrence across the corpus and determining whether it is the result of chance.

$$CSR = \frac{Z}{E(Z) + \sqrt{K} + \sqrt{{ln\delta}/{(-2)}}}$$

Equation 2 Equation for CSR

CSR works as follows :given a word pair

K is the number of documents in the corpus containing at least one occurrence of the word pair.

Z is the number of documents out of K that contain at least one significant co-occurrence of the word pair.

The word pair is said to be significant at corpus level if Z is more than or equal to the expected number of significant documents.

In later work Damani (Damani, 2013) investigates whether the inclusion of document level significance and corpus level significance does indeed improve PMI and how much each significance contributes to improving the measure. He finds that document level significance and corpus level significance alone do not contribute to improving PMI but it is their combination that achieves the improvement. He introduces a new measure that performs better than both CSR and PMI called cPMId

$$cPMId = \frac{d(x,y)}{d(x) * \frac{d(y)}{D} + \sqrt{d(x)} * \sqrt{\frac{\ln \delta}{(-2.0)}}}$$

Equation 3 Equation for cPMId

Where $d(x,y)$ is the total number of documents in which a span constrained co-occurrence of a word pair occurs.

$d(x)\ and\ d(y)$ is the total of documents containing respectively the word x or the word y.

$D$is the total number of documents in the corpus.

δ is a parameter between 0 and 1 which is tweaked to improve results.

Washtell and Markert (Washtell and Markert, 2009) also introduced PMIsig and SCI which will be discussed now.

2.4.1.2.2 PMIsig

To introduce statistical significance into PMI Washtell and Markert (Washtell and Markert, 2009) introduce PMIsig. PMIsig statistical significance is represented by the equation under the square root that is multiplied with the standard PMI equation. Washtell and Markert explain this by saying that statistical significance is proportional to the square root of the sample size. To calculate statistical significance, given a co-occurrence of two words Washtell and Markert assume the sample size to be the frequency (or probability) of the word that occurs less often and take the square root of that value to be the significance of the occurrence.

$$\text{PMIsig} = \frac{p(x,y)}{p(x) * p(y)} * \sqrt{\min(p(x), p(y)}$$

Equation 4 PMI Significance

Table 2 shows the output of PMI, PMISig and SCI using the NLP pre-processor discussed in Chapter IV. It was decided to employ PMIsig for this work as the words it outputted were more suited.

2.4.1.2.3 SCI

Washtell and Markert also create new measure by multiplying the conditional probability (the probability of observing the second word, given the first) by PMI. The new measure obtained is called Semi Conditional Information. SCI was introduced because all of the measures discussed so far including PMIsig produce symmetrical results. For example the associations united states and states united have the same value. But this does not reflect the human perception of word associations. So SCI was created to produce asymmetrical results that more closely matched human ones.

Equation for SCI is shown below:

$$\text{SCI} = \frac{p(x,y)}{p(x)\sqrt{p(y)}}$$

Equation 5 Semi Conditional Information

Damani and Ghonge (Damani and Ghonge, 2013) are critical of Watshell and SCI and introduce new measures, which according to the data presented perfom better

than SCI and PMIsig. They argue that introducing statistical significance is the right approach to overcoming the limitations of PMI but that so far significance has been applied incorrectly. They attempt to resolve this by introducing revised measures

2.4.2 Example Comparison of PMI, PMIsig and SCI Using Pre-Processor Discussed in Chapter V

As part of the practical work of this thesis an NLP pre-processor was develop to generate the files required for the AAC also developed as part of this thesis

The table below was generated using the pre-processor The data contained in this table were generated by asking the pre-processor to find and rank words related to the word *car*. The table compares PMI, PMIsig and SCI.

PMI	PMIsig	SCI
car motorail	car race	car race
car chaparral	car vehicle	car pit
car undertray	car track	car driver
car hoistway	car road	car lap
car downforce	car start	car downforce
car under body	car train	car vehicle
car flatcars	car speed	car rear
car aiguilles	car series	car passenger
car speedwalk	car service	car pace
car tracy's	car control	car diffuser
car back stretch	car passenger	car track
car diffuser	car design	car accident
car jackman	car driver	car train
car fleming's	car run	car truck
car understeer	car pass	car wheel
car soapbox	car air	car cars
car bond's	car tran sport	car indy
car coachbuilder	car move	car automobile
car carmaker	car year	car cable
car corvette	car people	car safety
car talladega	car system	car Andretti
car sportscar	car bus	car crash
car motorail	car lead	car driving
car ceaparra	car will	car motorail

Table 2 Comparing PMI, PMIsig and SCI

2.4.3 Measures Based on Distance Between Words

Distance is the number of tokens separating an occurrence of one word and the nearest occurrence of another word.

Co-Dispersion (CD) is yet another measure introduced by Washtell and Markert in (Washtell and Markert, 2009). It is defined as the ratio of the mean observed distance between two words to the expected distance between them, where the expected distance is derived from the frequency of the most frequent word type.

$$CD = \frac{\frac{1}{\max(p(x),p(y))}}{M(dist_{xy1} \dots dist_{xyn})}$$

Equation 6 Co-Dispersion

2.4.4 Measures Based on Syntactic Relations

Measures based on syntactic relations, measure word association by looking at grammatical relations between words (Washtell and Markert, 2009). The most popular way of carrying out these types of measurements is to use word networks such as WordNet. WordNet is an online thesaurus that links sets of synonyms by syntactic relations such as meronymity (a noun being a constituent part of another noun: window is a meronym of house) and antonymity (a noun being the opposite of another noun) (Miller, 1990). WordNet will be examined in detail in the Corpus Selection Chapter. For the purposes of understanding the measurements discussed in the next chapters it is sufficient to know that WordNet like a dictionary, contains definitions for each word, if words have multiple meanings, WordNet list all of its meanings (referred to as senses) and contains a definition for each meaning. The measurements work by comparing the words in the definitions of words being compared.

2.4.4.1 The Classic Lesk Algorithm

The adapted version of the Lesk algorithm for WordNet is a simple count of overlapping words between the dictionary definitions of two word senses in a sysnset. More simply given two words the algorithm measures similarity by counting the number of matching words between their definitions (stopwords are removed from each definition) .The algorithm compares senses for each word and the two which have the highest number of matching words in the definition are set to be the most similar.

2.4.4.2 The Extended Gloss Algorithm

The extended gloss algorithm (Pedersen et al., 2004) works like the classic algorithm with the difference that definitions of all hypernyms, and hyponyms for each sense are concatenated to the main definition of the sense. The similarity measurement is then performed on the resultant string of the concatenation process.

2.4.4.3 WordNet vs. PMIsig and Others

Even though WordNet can also be used to calculate word similarity like PMIsig and the others, using WordNet for similarity measurement has an inherent limitation: the classification is dependent on WordNet. The classification can only be performed between target words and words present in WordNet. This means that there is a potential Out Of Vocabulary problem. PMI has the advantage of being corpus independent as classification is performed among pairs in the corpus.

2.4.5 Latent Semantic Indexing/Analysis (LSI/LSA)

Latent Semantic Indexing (or Analysis depending on the context in which it is applied) will be discussed in the paragraph even though it was not used in the project as it is more suited to find syntagmatic word associations(Deerwester et al., 1990) . LSI was introduced by Deerwester in (Deerwester et al., 1990).

The first step to apply LSA is to convert the corpus into what is called vector space model. This is common practice in Information Retrieval techniques such as LSA (Jurafsky and Martin, 2000). Converting a corpus into vector space model means structuring data contained into the corpus as a matrix

Terms	**Documents**								
	c1	c2	c3	c4	c5	m1	m2	m3	m4
human	1	0	0	1	0	0	0	0	0
interface	1	0	1	0	0	0	0	0	0
computer	1	1	0	0	0	0	0	0	0
user	0	1	1	0	1	0	0	0	0
system	0	1	1	2	0	0	0	0	0
response	0	1	0	0	1	0	0	0	0
time	0	1	0	0	1	0	0	0	0
EPS	0	0	1	1	0	0	0	0	0
survey	0	1	0	0	0	0	0	0	1
trees	0	0	0	0	0	1	1	1	0
graph	0	0	0	0	0	0	1	1	1
minors	0	0	0	0	0	0	0	1	1

Figure 5 Term Document Matrix from (Deerwester et al., 1990)

LSI is used to improve the quality of search results when retrieving information (Deerwester et al., 1990). LSI was created to return search results which are closer to what a human would deem relevant and not only return matches based on character similarity between the query and the documents in an index or database. For example if someone does not remember the name of a book correctly and inserts words with a similar meaning to the correct title, LSI should be able to return the correct result. Deerwester created LSI to find syntagmatic associations like synonyms to produce better search results. Rapp (Rapp, 2002) also finds that LSI is suited to find syntagmatic word associations.

The second step of LSI is to break the term document matrix by using a mathematical technique called Singular Value Decomposition (SVD) . SVD returns three separate matrices by convention referred to as T S and D. The full mathematical details of SVD will not be given here. After obtaining T, S and D, an operation called rank approximation is carried out, rank approximation consists in reducing the size of the S matrix . The rank (k) represents the number of values that are kept from the S matrix. Multiplying reduced S by T and D returns a new matrix which is used for similarity measurements.

By performing mathematical operations on the new matrix three measures that can be obtained:

- How similar two terms are

  By computing the dot product of the two words vector (the rows of the matrix)
- How similar two documents are

  By computing the dot product of the two document vector (the columns of the matrix)
- How related is a particular term to a particular document

  By comparing single values in cells.

Co-occurrence based measures were preferred over LSA as LSA is most suited to find syntagmatic associations such as synonymy and polysemy (words that have multiple meanings). This is stated in (Landauer et al., 1998) and (Islam and Inkpen, 2006); for the purposes of this work we are interested in paradigmatic associations.

2.5 Introduction to the Compansion Algorithm

The Compansion Algorithm was created by Demasco and McCoy in 1992 (Demasco and McCoy, 1992). They propose a different kind of keyboard that uses words instead of single letters as input. Words are lemmatised and the goal is to reduce input speed and sentence formation responsibility on the part of the user. Demasco and McCoy propose that this system greatly benefits users with severe physical impairments and users with linguistic impairments who still retain the ability to read and choose the keywords of a sentence. They state that Compansion can be used as "a conversational aid in settings where grammatical output is desired ", which is precisely the goal of this work. Put simply this algorithm works by having the user insert keywords of a sentence and having the application form potential sentences and finally having the user select the desired sentence.

Compansion works in three stages shown in figure below

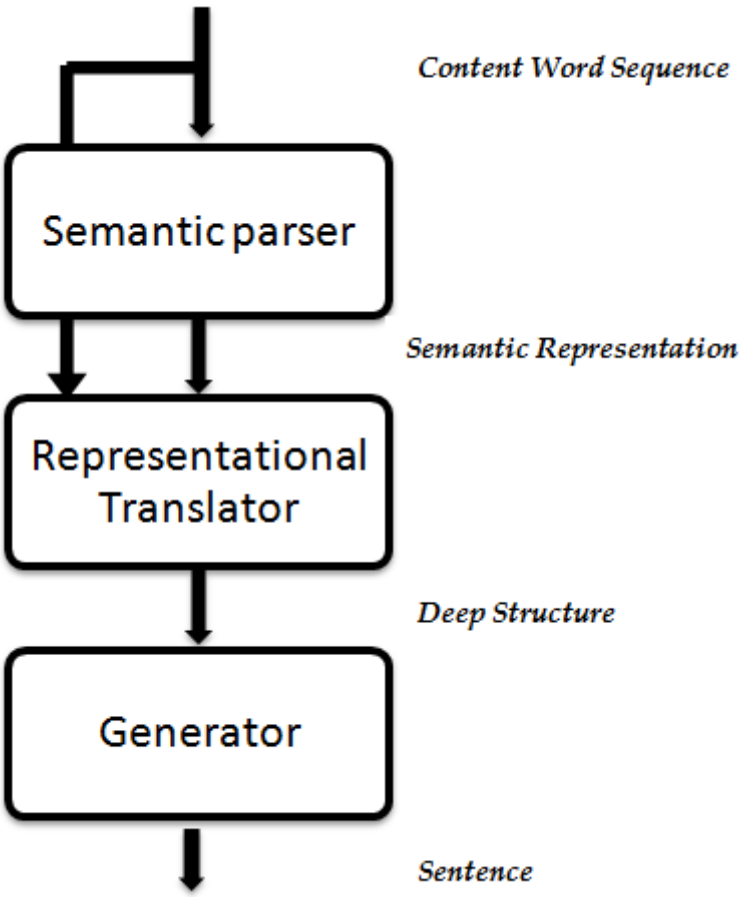


Figure 6 High Level Overview of the Compansion System (Demasco and McCoy, 1992)

Lemmas are passed to the semantic parser to assign roles to each word, each role is translated into a sub-sentence and finally each sub-sentence is merged into a sentence.

Demasco and McCoy give the example of the input JOHN EAT APPLE. JOHN is given the role of ACTOR, EAT is given the role of ACTION and APPLE is given the role of THEME, given this input Compansion returns John eats the apple.

This was a short overview of the Compansion Algorithm. Compansion will be examined in more detail in the section concerning the Natural Language Generator developed as part of this work so as to more easily justify design choices taken when developing the generator based on the Compansion Algorithm.

2.6 Natural Language Generation

Natural Language Generation (NLG) is a subfield of Computational Linguistics and Natural Language Processing that focuses on computer systems which are capable of generating human language like output. NLG employs a computationally friendly model of human language and tries to map that model to human language. One might think that NLG systems are only used to talk to the user or simulate dialogue between the user and a computer system or device, however NLG is also concerned with presenting data to user in a human friendly manner (e.g. weather reports, sensor data) and generating documents automatically.

Although NLG systems can differ typically they perform the following three tasks:

Content Selection: There will be instances where a system has to choose from a wide variety of input. At this stage the system determines the best input to use and pass on to the realiser.

Surface Realisation: during this phase the system selects the appropriate words and arranges them in the correct order.

Production: during this phase the system presents the generated text to users.

There is no definite agreement among NLG experts about what input to NLG systems should be. Usually NLG systems tend to be tailored around the device or larger system they are part of.

2.6.1 Generative Grammars

A common technique used in NLG is to use Context Free Grammars to generate sentences. Context Free Grammars (CFGs) are the most widely used mathematical structure to model natural languages. (Jurafsky and Martin, 2000). A CFG is made of two things: a set of rules (also called productions), and a lexicon.

The rules determine ways in which symbols of the language can be expressed together and the lexicon: is the collection of words and symbols which makes up the language.

There are two sets of symbols in CFGs: Terminal Symbols and Non-terminal Symbols. Terminal Symbols correspond to words in the language. Non terminal symbols express grammatical groups of words in language such as Noun phrases (NP), Nouns (N), Verbs Phrases (VP) and Verbs.

CFGs can be used both to parse and generate sentences . For this reason CFGs are also called generative grammars.

For example the rules shown below could be used to generate simple sentences. The arrow in CFG syntax means that symbols on the left can be rewritten as symbols on the right.

$$
\begin{aligned}
NP &\rightarrow Det\ Nominal \\
NP &\rightarrow ProperNoun \\
Nominal &\rightarrow Noun \mid Nominal\ Noun
\end{aligned}
$$

Figure 7 Example of Context Free Grammar

$$
\begin{aligned}
Det &\rightarrow a \\
Det &\rightarrow the \\
Noun &\rightarrow flight
\end{aligned}
$$

Figure 8 Example of Context Free Grammar

The above rules can be used to generate the sentence :*a flight*. The sentence is formed using the rule that a Noun Phrase can be composed by a determiner and a noun, the determiner is the word "a" and the noun is the word "flight".

As mentioned CFGs can be used to parse the structure of sentences, if the sentence *a flight* were to be parsed this would the result.

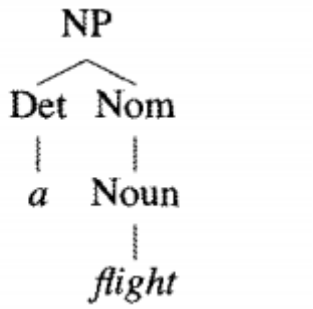


Figure 9 Context Free Grammar for Noun Phrase Represented as Tree

It is common to represent CFG rules derivations as trees like the one shown above. Usually CFGs have a start symbol conventionally set as S.

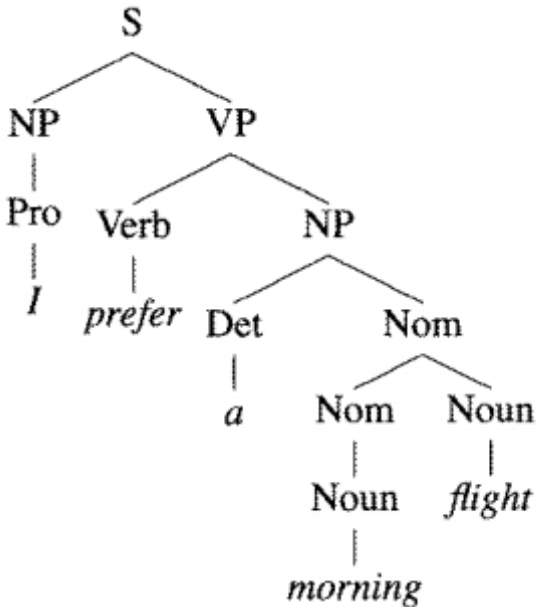


Figure 10 Context Free Grammar for Sentence Represented as Tree

2.6.2 A Practical Example of NLG: Functional Unification Grammars and FUF/SURGE

Using CFGs as generator is inefficient. (Bird et al., 2009),(Jurafsky and Martin, 2000). Traditional categories of context-free grammar are symbols and a new symbol is required for every new grammatical case. To capture every possible scenario in the English language with its many exceptions, the number of symbols required becomes very large .For this reason a more efficient way of representing grammar was invented. These are called Functional Unification Grammars (FUG) FUG was introduced by Martin Kay in (Kay, 1984). FUG overcomes the limitation of representing language using CFG, by introducing data structures similar to dictionaries that contain attributes and values called Functional Descriptions (FD)

Figure 11 and Figure 12 show an example of an FD; FDs can also contain other Functional Descriptions.

$$\begin{bmatrix} \text{CAT} & \text{NP} \\ \text{NUM} & \text{SG} \\ \text{PERS} & 3 \\ \text{CASE} & \text{NOM} \end{bmatrix}$$

Figure 11 Example of Feature Description

$$\begin{bmatrix} \text{CAT} & \text{NP} & \\ \text{AGR} & \begin{bmatrix} \text{NUM} & \text{SG} \\ \text{PERS} & 3 \\ \text{CASE} & \text{NOM} \end{bmatrix} \end{bmatrix}$$

Figure 12 Example of Feature Structure Encapsulation

The advantage of using FDs over CFGs can be made apparent with the example of the grammatical rule known as agreement. In simple terms for a sentence to be grammatically correct verb and subject have to agree, for instance if the subject is singular then the verb must also be singular. To enforce the agreement rule CFG requires the introduction of new symbols. Jurafsky and Martin make the example of auxiliary questions, where the auxiliary has to agree with the verb for the question to be grammatically correct. Figure 12 shows an example of a feature containing another features as attribute.

$$S \rightarrow 3sgAux\ 3sgNP\ VP$$
$$S \rightarrow Non3sgAux\ Non3sgNP\ VP$$

Figure 13 CFG Containing New Rules for Enforcing Agreement

$$3sgAux \rightarrow does \mid has \mid can \mid \ldots$$
$$Non3sgAux \rightarrow do \mid have \mid can \mid \ldots$$

Figure 14 CFG Adding new Rules for Questions

$$3SgNP \rightarrow Det\ SgNominal$$
$$Non3SgNP \rightarrow Det\ PlNominal$$
$$SgNominal \rightarrow SgNoun$$
$$PlNominal \rightarrow PlNoun$$
$$SgNoun \rightarrow flight \mid fare \mid dollar \mid reservation \mid \ldots$$
$$PlNoun \rightarrow flights \mid fares \mid dollars \mid reservations \mid \ldots$$

Figure 15 CFG Rules for Enforcing Agreement

To enforce this in CFGs a rule for every case is required, Jurafsky and Martin state that just for the simple case above of yes/no auxiliary questions, the number of symbols required to enforce agreement doubles. In the example above two new symbols are required one for the third person singular auxiliary and one for the non third person singular auxiliary. New symbols are also required for third person and non third person singular noun phrases and so on.

FDs on the other hand make this process much simpler by exploiting two characteristics of FD. Firstly the fact that FD values can be FDs themselves, as shown in the example below, where the value for the agreement attribute is a FD containing the number and the person., and secondly the property known as ... shown in Figure 17 where an attribute can be used as value for other attributes within the sentence by using a label .

$$\begin{bmatrix} \text{CAT} & NP \\ \text{AGREEMENT} & \begin{bmatrix} \text{NUMBER} & sg \\ \text{PERSON} & 3rd \end{bmatrix} \end{bmatrix}$$

Figure 16 Enforcing Agreement with Feature Descriptions

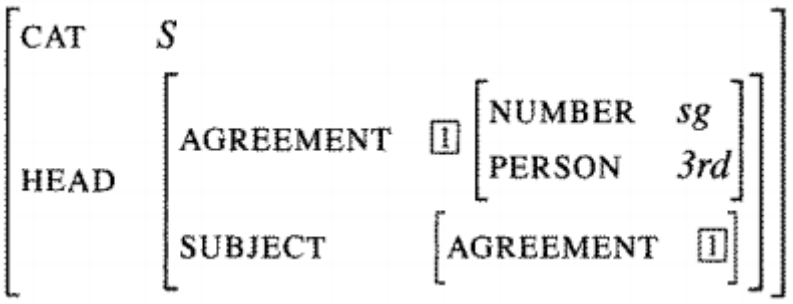


Figure 17 Enforcing Agreement with FDs

2.6.3 SimpleNLG and Justification for its Use as Realiser

SimpleNLG is a Java realiser implemented by Gatt and Reiter (Gatt and Reiter, 2009). The goal of SimpleNLG is to provide programmers, linguists and those who require sentence generation in general to achieve fast sentence generation. According to Gatt and Reiter the task of language generation can be subdivided into two tasks: Tactical generation, and generation of sentence itself. Tactical generation involves making linguistic choices given the input. In practice it means that given some input of words multiple valid English sentences could be generated. SimpleNLG performs no tactical generation and it is for the developer to specify the role of each input within the sentence. SimpleNLG provides a straightforward API that does not require the developer to learn complex formalisms. A basic understanding of English grammar is sufficient to use this realiser.

It was decided to employ SimpleNLG as realisation engine for this work, firstly because of the fact that it is a Java Library and so it could be seamlessly integrated to the AAC application. In contrast FUF/Surge and other FUG based engine work using LISP formalisms as semantic input which need to be mastered appropriately for them to be effective. SimpleNLG gives finer control to non expert developers who require to generate sentences in their applications. Secondly the kind of sentences to be realised as part of the work (simple statements, requests, and questions) are appropriately handled by SimpleNLG. Thirdly sentence generation itself is computationally faster when compared to formalism based generator such as FUF/SURGE, this enables real time sentence generation as required in this thesis. Finally SimpleNLG has been successfully used in the work of others for example Black et al. in (Black et al., 2012).

2.6.4 Language Models (N-Grams)

N-grams are sequences of text. The number of words contained in the sequence is determined by the N (Jurafsky and Martin, 2000). For example, a sequence of two

words is referred to as bi gram and a sequence of three words is called tri gram. N-grams can also be referred to as Language Models (LM) They can be used for two main purposes: they can predict **the likelihood of a word given some history of previous words which is equal to N-1,** or**, they can predict the likelihood of a whole sentence.** N-grams predict the next word as a sequence of words using some previous history of words. The N stands for the number of words minus one that are used to compute the prediction.

Jurafsky and Manning make the following example to explain how n-gram word prediction works: consider the sentence "The water is so transparent that" if one wanted to compute the probability of the next word being "the" one could use the following formula

$$\frac{Count\ (the\ water\ is\ so\ transparent\ that\ the)}{Count(the\ water\ is\ so\ transparent\ that)}$$

But even if one were to use the web as a corpus finding all of the occurrences of these two sentences and counting them is unfeasible even by modern standards.

For this reason n-gram models predict the next word by using an approximation with a few preceding words. This is assumption is called the Markov Assumption.

Jurafsky and Martin (Jurafsky and Martin, 2000) defines the Markov assumption as "**the product for each word of the conditional probability of that word given some prefix of the last few words.**"

The weakness of Markov Assumption is that it decontextualises words from sentences so there are some cases where the prediction does not match the intended sentence of the user. Because n-gram frequency is the main decider of the prediction outcome. As a practical example: there is a programmer who is using text prediction to write about the cause of problems with his latest algorithm and wants to write the sentence *the problem with my algorithm was a stray pointer*. It is more than likely that if a bi-gram or even a tri-gram model were used the prediction algorithm would produce "cat" or "dog", as the keyword in the sentence is "algorithm" and the bi-gram "stray pointer" is (or can be) common when talking about algorithms.

The most common N-Grams available today are, bigrams (predict using the previous word), trigrams (predict using the two previous words) and quadrigrams (predict using the three previous words).

The building block of statistical N-Gram Language Models is the probability chain rule (Jurafsky and Martin, 2000).

The chain rule is obtained by rewriting the formula for conditional probability:

$P(A|B) = \frac{P(A,B) equivalent\ to\ P(B|A)}{P(B)}$, where P(A,B) (also referred to as represents the probability of the co-occurrence of *A and b* and P(B) is the probability of B appearing alone.

$$P(Sentence) = P(w_1)P(w_2|w_1)P(w_3|w_1,w_2) \dots P(w_n|w_1^{n-1})$$

To calculate the conditional probability of a word coming next the formula for Maximum Likelihood Estimation:

$$P(w_2|\, w_1) = \frac{Count\ (w_1 w_2)}{Count\ (w_1)}$$

2.6.5 Semantic Grams

Wiegand and Patel (Wiegand and Patel, 2012b) propose a new method for text prediction in AAC using Semantic Grams. Semantic Grams are collections of words that appear together in text. For example, the span constrained word pairs used in Word Association Measures discussed earlier in this Chapter could be considered semantic grams, albeit only Wiegand and Patel and the work they cite refer to them as semantic grams. The idea behind Wiegand and Patel's work is to implement text prediction in cases where input is unordered. The work compares four different algorithms that take a set of unordered words as input and return a list of candidate next words ranked from the most likely to the least likely. Two of the algorithms use N-Grams, and the other two use semantic grams. Both the N-Gram algorithms and the Semantic N-Grams algorithms compare two approaches to word prediction: word prediction using conditional probability and word prediction using joint probability.

Whilst the goal and approach of Wiegand and Patel's work would be suitable for the purposes of this thesis, the author themselves recognise that the work is at the early stages and more work is required to expand the work beyond n-grams and semantic grams of size 2. For example, the algorithms require a set of candidate words to select from. In ideal cases the candidate words would be selected from the whole corpus, however to reduce computational requirements of the algorithms, the authors selected only the most frequent co-occurring words that appeared as n-

grams or semantic grams with the input words. The algorithm was not evaluated using the whole corpus as candidate words.

2.7 Inclusion of NLP and NLG in AAC in the Work of Others

Most of the work cited so far makes use of NLP and NLG (sometimes both). SymbolPath (Wiegand and Patel, 2012a) uses Natural Language Generation to generate simple sentences with a subject-verb-object structure.

The works by Black and others (Black et al., 2012, 2010) use data gathered from sensors to generate sentences about the user's daily experiences. Their system is targeted at children with Complex Communication Needs and attempts to use sensor data to enable children to recount their day at school. The system used a collection of RFID tags in different location in school and placed on different objects. RFID cards were also given to school staff so that the system could detect when an interaction occurred between the child and the member of staff. Children could also ask the interlocutor if the conversation could be recorded so that it could be included in the story generated by the system. The system used the collected data to generate stories for the child to choose from.

2.8 Discussion

Chapter I showed that there is need of research in electronic AAC, and that a promising avenue of research is the integration of NLP, NLG and Text Entry to improve ease and speed of use of these systems. This Chapter has discussed in detail the theoretical concepts that will be used in the practical parts of this work. As mentioned at the end of Chapter 1, among the word association measurements that were discussed it was decided to focus on co-occurrence based one, firstly because they are best suited to find paradigmatic associations and secondly because even though there are claims that some distance based measures could perform better, co-occurrence based approaches are well established in literature. Significance based approaches have been shown to overcome the inherent limitation of classical PMI. It has been shown that there are several alternatives of significance based co-occurrence Word Association Measures from which to choose but PMIsig was shown to perform better with the chosen corpus, even though discussion on corpus choice is the topic of a subsequent Chapter. PMIsig is useful to reduce user search space and give users a better choice of words, but to improve ease of use NLG is

also required to generate whole sentences using only key input. This approach is based on an algorithm which was also discussed in this Chapter called Compansion.

2.9 Key Lessons

From the theoretical based discussion presented in this Chapter we can conclude the following :

1. Among NLP approaches, statistical NLP can help with the proposed goal of this work, especially finding word associations. A discussion based on the work of others has shown that the semantic relatedness measure most suited to the needs of this work is PMIsig.
2. PMIsig will be used to produce key input for an algorithm that takes compressed text as input and generates full sentences. This algorithm will be similar to Compansion and expanded in Chapter VIII
3. The algorithm can be used to generate sentences with the user only having to insert key words.

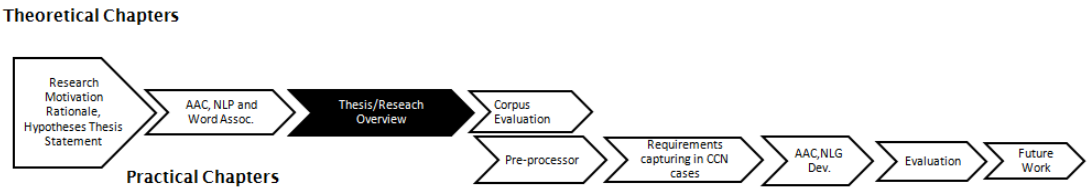
2.10 Summary

The goal of this work is to improve current AAC system by the integration of Natural Language Processing, Natural Language Generation and Text Entry Techniques.

Two ways are proposed in this work to improve electronic AAC ease and speed. The first is to reduce word search time for the user by dynamically changing the input, presenting the user with words related to the last word inserted. The second is to delegate the responsibility of sentence formation to the AAC application, by using an NLG algorithm that accepts key words as input and returns fully formed grammatically correct sentences.

This Chapter has given the reader the theoretical background needed to implement this. The Chapter first started with a formal definition of AAC, a categorisation of users who benefit from AAC and a list of the most popular AAC applications currently available. An overview of Natural Language Processing followed to introduce the terms of the field and techniques which will be used to produce the data required for the AAC application to work. A literature review of works on word relatedness justified the choice of PMIsig over the alternatives. A short explanation of semantic role labelling was given in the NLP paragraph as it is

required to understand the Compansion algorithm. Compansion is what the Natural Language Generation part of this work is based on. This Chapter only provided a short overview of Compansion. Fine detail will be given in the Chapter regarding the AAC app development as the Compansion algorithm will be compared with the ad hoc algorithm developed during this work.



Chapter III: Thesis Overview

3.1 Introduction

Previous chapters have given rationale as to why it is important to improve electronic Augmentative and Alternative Communication applications. To achieve this, the work of others suggests that sentence formation speed within AAC applications can be augmented via the use Natural Language Processing, Natural Language Generation and Advanced Text Prediction techniques. The aim of this Chapter is to give the reader a clear overview of how the integration of NLP/NLG and Text Entry with AAC was researched, approached practically and evaluated so as to make subsequent chapters clearer.

Chapter III will start off with a high level view of the various stages this research was conducted in.

The Chapter will conclude with an overview of the content of each Chapter, with clear emphasis on research goals and Chapter outcomes.

3.2 Thesis Overview

The following hypotheses and research questions were proposed in Chapter I

H1. The integration of Natural Language Processing and Natural Language Generation with Augmentative and Alternative Communication will reduce the number of keys required to form sentences compared with competitor apps leading to the potential improvement in speed of sentence formation.

H2. The integration of Natural Language Processing and Natural Language Generation with mobile Augmentative and Alternative Communication systems will reduce screen navigation thereby improving ease of sentence formation by avoiding mental overload and stress on users.

Q1. Will the integration of Natural Language Processing and Natural Language Generation improve speed of sentence formation in mobile AAC systems by reducing the number of keys required to form sentences?

Q2. Will the integration of Natural Language Processing and Natural Language Generation improve ease of sentence formation in mobile AAC systems by reducing stress and avoid mental overload?

Q3. Can a system that excludes keyboard and relies on whole words input work as an effective input mechanism in mobile AAC applications?

Q4. Can we discover requirements through the use of User Centred Design in cases where the users have Complex Communication Needs?

To prove these hypotheses and answer these research questions, research carried out as part of this thesis was conducted in three stages:

1) The first stage focused on the implementation of the theoretical concepts discussed in the previous chapters. It was argued with the aid of the work of others, that to improve text entry ease and speed within AAC systems two things would be required: first a way for users to more easily navigate among words in the application via a Semantic Network generated by ranking word co-occurrence with word similarity measurements, and second: delegating sentence formation to the application itself by using an algorithm similar to Demasco and McCoy's "Compansion", in practice having the user only insert key words of a sentence to obtain a fully formed and grammatically correct result . This stage of the research focused on finding the most suited corpus to generate the files required for Dictum. In practice the files were a collection of nouns and verbs and Semantic Network which enabled users to navigate among related words. The files were generated using a pre–processor also developed during this stage and which was responsible for analysing and for the output of the required files.

2) The second stage focused on researching the best design and implementation for an AAC application that makes use of data generated in the previous stage of research. The design phase was conducted with the help of potential users and speech professionals. The interface of the app was based on current applications available to buy on app markets. Requirements were also captured by applying an HCI model called "Technology Appropriation Model". This model has never been applied

before to cases of users with Complex Communication Needs. Results of applying the observation of Appropriation for requirements gathering were published at the ASSETS conference of 2015.

3) The third stage focused on the implementation and research of the Natural Language Generation part of this thesis. While the first part provided input for users the aim of the first stage was to find a way to delegate the generation of grammatically correct sentences to the application. The NLG algorithm follows the idea of "Compansion". Users have to insert the key nouns and verbs and the application tries to provide sentences. The NLG server was only partially implemented to generate basic sentences. The NLG algorithm was then expanded theoretically to cater for more complex communication scenarios.

3.2.1 Dictum App Evaluation Overview

The evaluation of the Dictum was conducted in two parts. The first part of the evaluation was qualitative and it involved going back to the University of Dundee "Tap and Talk "group to ask potential users and professionals feedback on the application. The second part of the evaluation was quantitative and it involved comparing the application developed as part of this work with AAC apps available on the market which were chosen as they were familiar to users and professionals met while undertaking this work. The comparison between Dictum and competitor apps was carried out by selecting a set of test sentences which contained words present in all applications being compared. The number of taps and screens required to form sentences were compared to verify whether the hypotheses were proven. This approach led to a discussion on other aspects of the application such as scenarios where the NLG model could be improved, aspects that other applications address which were not examined and ways to improve the interface. The final part of the evaluation consisted in applying an adapted version of Hick Hyman Law to measure the average time required by a user to find a word in the Semantic Network by simulating difference grades of performance of the prediction.

3.3 Thesis Structure

3.3.1 Chapter IV: Corpus Selection

Outcomes of Chapter:

1) Investigation of corpora available today.
2) Investigation of the structure of Wikipedia and on different ways to obtain from it and analyse it
3) Investigation on how to conduct NLP research on crowdsourcing platforms

Research aim(s):

1) To identify the best corpus or corpora to use for the generation of the data required for the AAC application

Chapter IV will discuss the corpus selection process. This Chapter will compare the available corpora and attempt to find the one best suited to produce the files required for the AAC application also developed as part of this thesis. Chapter II has explained how any NLP task requires a corpus. However, depending on the task that is required a suitable corpus has to be selected. The aim of the Chapter is to find it. As stated in the thesis statement the overall goal of this thesis is to apply NLP techniques to AAC to measure whether the integration of these techniques improves text entry speed and ease of use for the user.

Chapter IV will start off with an overview of the most used corpora in NLP today. It will be argued that for the purposes of this thesis the best corpus to use is Wikipedia. The process of analysing Wikipedia will be discussed through various iterations. First it was unclear whether to use the whole of Wikipedia. After an extended investigation it emerged that using a subset would be more computationally feasible and more manageable. Four categories of conversation were selected to be the most representative of common conversation: food, music, sports and transport. First a Naïve Bayes Classifier was trained to classify articles, but due to precision errors commonly associated with computational classification an alternative avenue was explored: Wikipedia's own classification was used to gather the most relevant articles.

Representative categories of Wikipedia were chosen for each of the four conversational topics. Articles were then downloaded in hierarchical fashion starting from the representative category of each topic to all of the subcategories.

Articles were downloaded down to a depth of three in the hierarchy. For example for the conversational topic food, the category at the top of the tree was "Food and Drink", the first tier was all subcategories of "Food and Drink" and the subsequent tiers were subcategories of the subcategories.

It will also be discussed how the British National Corpus' tagged data was used to filter out all Parts of Speech but nouns, and to identify the verbs most commonly associated with each noun.

Chapter III will conclude with a discussion on the selection of the best subset of Wikipedia articles The subset was selected by conducting a crowdsourced experiment was set up where users classified the first hundred words from several combination of articles.

3.3.2 Chapter V: Language Pre-Processor Development

Outcomes of Chapter:

1) Implementation of fully working pre-processor for the analysis of Wikipedia article collections and XML files of the British National Corpus
2) Generation of vocabularies and word networks

Research aim(s):

1) To research the best way to rank and filter data for the AAC application

Chapter V will examine how NLP theory explained in Chapter II was implemented in software to generate the data required for the AAC application to work. The research aim of this Chapter is how to best present and/or filter the data that is passed to the AAC Application . Data were generated via a pre-processor that took in Wikipedia articles as input and generated vocabulary files and a Semantic Network, to enable users to input nouns and verbs and to navigate through a network of related nouns.

The pre-processor uses all theoretical concepts discussed in Chapter II to generate the following files:

- Vocabulary of all words in the corpus
- One vocabulary file per each conversational category
- A Semantic Network of related nouns generated using PMIsig with a window of 10
- A verb-noun network which contains the most used verbs for each noun

The vocabulary data is presented in the form of verbs and noun to select in order for the application to form sentences.

The Chapter will conclude with an analysis of the best approach to use to rank words presented to users.

3.3.3 Chapter VI : Capturing User Requirements in Cases of Users with Complex Communication Needs

Chapter outcomes:

1) A set of requirements for the AAC application
2) Poster accepted at ASSETS 2015

Research aim(s):

1) To capture user requirements by applying the Technology Appropriation Model. This model has never been applied to cases of users with Complex Communication Needs

Chapter VI will apply a model called "Technology Appropriation Model" (TApM) to identify the best requirements to include in the AAC application. During the development of Dictum, as each version was completed potential users were interviewed to give feedback on how to improve the subsequent iterations of the application. However because users had Complex Communication Needs it was difficult for them to express what features they would have liked. To infer what their requirements were it was decided to analyse the conversational data collected during the interviews and to apply TApM. TApM analyses how users appropriate technology to suit their needs. The word Appropriation in this context assumes a meaning that is different from the English sense of the word. To Appropriate means to adapt a technology or a piece of software to suit one's needs and to use a piece of technology in ways that designers of the technology themselves did not foresee.

By applying TApM it was possible to capture a list of requirements to improve current AAC systems. The application of TApM to cases of CCN therefore represents a contribution to research and the results of this Chapter were published as poster at the 2015 ASSETS conference.

3.3.4 Chapter VII AAC Application Design and Development

Outcomes of Chapter:

1) Development of Android OS AAC application

Chapter VII will discuss the development of Dictum: the AAC application central to this thesis. It will describe development specifics and will justify each design decision. Several iterations of the software were developed. Each iteration was shown to potential users to obtain feedback on the subsequent iteration. The last version of the app was designed to connect to a NLG server to receive fully formed sentences as response. The Chapter will compare the Compansion algorithm with the NLG algorithm developed in the thesis.

3.3.5 Chapter VIII: Natural Language Generation

Outcomes of Chapter:

1) Development of ad-hoc NLG algorithm based on Compansion
2) Investigation into NLG APIs
3) Investigation of WordNet APIs and their integration into NLG engine
4) Investigation of Microsoft N-Gram web service and its integration into NLG engine

Research aim(s):

1) To research the best way to design a Natural Language Generation algorithm that generates full grammatically correct sentences using only nouns and verbs as input.

Chapter VIII will discuss the design and development of the NLG algorithm to which the AAC application connects to, and receives fully formed sentences back from. The Chapter will start with a detailed description of the Compansion algorithm as Compansion was used as a blueprint for the NLG algorithm. The main difference is that Compansion remains a theoretical algorithm, whereas the algorithm developed in this work, is practical and was made to integrate seamlessly with the AAC application.

Due to the timescale, the NLG algorithm was developed to form basic sentences and was designed to be scalable. It will be shown that even though the implemented NLG does not cater for many conversational scenarios, the way in which it was designed lends itself to expansion and modification.

The Chapter will conclude with a discussion on how to fill syntactic gaps left by the implemented NLG. The N-gram model will be explained as it is the central part of today's text entry systems. Because the implemented system already provides all of the words required to form the sentence, rather than rely on the next most likely word like text entry systems do, sentence likelihood will be measured to form correct sentences. The example of finding the correct determiner before nouns will be used to explain this.

3.3.6 Chapter IX: Evaluation of the AAC Application

Outcomes of Chapter:

1. Quantitative Evaluation of AAC application
2. Qualitative Evaluation of AAC application (based on feedback given by users and professionals).
3. Discussion on strength and weaknesses of the AAC application.
4. Discussion on possible improvements.

Research aim(s):

1. Quantitatively evaluate the AAC application.
2. Qualitatively evaluate the AAC application.
3. Prove/disprove Hypotheses given in this Chapter
4. Answer proposed research question.

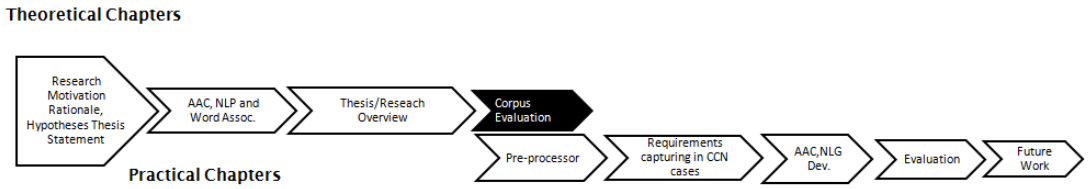
This Chapter will discuss the final evaluation of the AAC application. The evaluation was conducted both by comparing Dictum (the AAC application developed as part of this work) with commercial apps familiar to users encountered during the course of this work, and by showing the prototype to potential users by visiting the Dundee group that helped with the requirements capturing discussed in Chapter VI. This discussion will include feedback from both users and Speech and Language professionals. The Chapter will show that integrating NLP/NLG techniques in AAC application does indeed increase speed and ease of use over current AAC application. There are scenarios where the proposed model could be improved, the Chapter will include a balanced discussion, possible solutions and future direction which will lead to the Future Work Chapter.

3.3.7 Chapter X: Future Work

Outcomes of Chapter:

1. Identifying future avenues of research that were not pursued due to timescale

This Chapter will discuss unanswered questions that arose during this thesis and how to address them



Chapter IV: Corpus Selection

4.1 Introduction

Chapter II has given an introduction to NLP techniques. It was argued that for the purposes of this project statistical NLP was best suited. All statistical NLP tasks require the use of a corpus. This Chapter will discuss the process used to select the corpus used for all the NLP needed for semantic prediction by the AAC application and the pre-processor designed and developed to provide the semantic map file used by the AAC application.

Firstly, the Chapter will examine the choice of corpora currently available. Justifications will be given for why Wikipedia was favoured instead of these corpora. Once the choice was made to use Wikipedia it was unclear whether the whole of it would be used or only a fraction of it. To provide a wide vocabulary of words for users and to cater for the widest amount of scenarios initially it was decided to use the whole of Wikipedia by downloading dump files of the latest articles. As the investigation progressed it became clear that some sort of article filtering would be required, and Naïve Bayes classification was used to classify articles. To increase precision it was decided to use Wikipedia's internal classification but since categorising Wikipedia dumps proved to be difficult it was decided to download articles based on the category pages of Wikipedia. The two classification methods will be compared to show that category based classification is more effective to form the final file.

4.2 Corpora

As was explained in Chapter III, any Natural Language Processing task requires the use of a Corpus.

Corpora can be classified in different ways: for example Wojtinnek et al (Wojtinnek et al., 2012) divide corpora into two categories, conversational corpora and non conversational corpora, and further subdivide the categories into plain text corpora, Category based Corpora (Wikipedia) and Network based corpora (WordNet).

Aston (Aston and Barnard, Lou, 1998) gives a more detailed classification of corpora:

> *Geographical corpora*: based on the location where the language is spoken. The BNC is an example of this as it is a representation of English spoken in Britain.
>
> *Spoken language corpora*: Corpora which contain transcription of spoken conversations.
>
> *Mixed corpora:* Corpora which contain both spoken and written corpora.
>
> *Historical corpora:* Corpora which contain historical texts and are used to study language changes over long periods of time.
>
> *Child/ learner corpora*: corpora based on children or literature aimed at learners of language.
>
> *Genre specific*: corpora based on a specific topic or genre in literary fiction
>
> *Multilingual corpora*: corpora which contain content in more than one language.

The next paragraph will give a short overview of what corpora are available today for those who research language and Natural Language Processing.

4.2.1 The Brown Corpus

The Brown Corpus (Kucera, 1967) was the first annotated corpus. Its compilation started in the 60s and its content is continually updated. Other annotated corpora like the Lancaster-Oslo-Bergen Corpus (LOB) (Johansson et al., 1978) are based on the Brown Corpus. The LOB Corpus contains 500 samples of English language distributed across 15 genres. Corpora today, however are several times larger than the Brown Corpus.

4.2.2 The British National Corpus

The British National Corpus project started in the '90s. It consists of a collection of documents that should represent contemporary British English (Aston and Barnard, Lou, 1998). It is composed of both written and transcribed spoken English although written English represents 90% percent of the corpus.

The BNC contains tagged data (Leech, G. Garside, R. and Bryant, M, 1994). The image below shows an excerpt of the XML version of the British National Corpus. Information about single words is contained in the "w" tag. The "w" tag contains several attributes with grammatical information on the word. Part of Speech information is contained in the "pos" attribute. "hw" contains the stemmed word, the information contained in these tags was used in the pre-processor to remove unnecessary words. The pre-processor is the topic of Chapter V. The BNC was tagged using an automatic system called CLAWS (Leech, G. Garside, R. and Bryant, M, 1994)./

```
<p>
  - <s n="2">
    - <hi rend="bo">
        <w pos="SUBST" hw="aids" c5="NN1">AIDS </w>
        <c c5="PUL">(</c>
        <w pos="VERB" hw="acquire" c5="VVN-AJ0">Acquired </w>
        <w pos="ADJ" hw="immune" c5="AJ0">Immune </w>
        <w pos="SUBST" hw="deficiency" c5="NN1">Deficiency </w:
        <w pos="SUBST" hw="syndrome" c5="NN1">Syndrome</w>
        <c c5="PUR">)</c>
      </hi>
      <w pos="VERB" hw="be" c5="VBZ">is </w>
      <w pos="ART" hw="a" c5="AT0">a </w>
      <w pos="SUBST" hw="condition" c5="NN1">condition </w>
      <w pos="VERB" hw="cause" c5="VVN">caused </w>
      <w pos="PREP" hw="by" c5="PRP">by </w>
      <w pos="ART" hw="a" c5="AT0">a </w>
      <w pos="SUBST" hw="virus" c5="NN1">virus </w>
      <w pos="VERB" hw="call" c5="VVN">called </w>
      <w pos="SUBST" hw="hiv" c5="NP0">HIV </w>
      <c c5="PUL">(</c>
      <w pos="ADJ" hw="human" c5="AJ0-NN1">Human </w>
      <w pos="SUBST" hw="immuno" c5="NN1">Immuno </w>
      <w pos="SUBST" hw="deficiency" c5="NN1">Deficiency </w>
      <w pos="SUBST" hw="virus" c5="NN1">Virus</w>
      <c c5="PUR">)</c>
      <c c5="PUN">.</c>
    </s>
  - <s n="3">
      <w pos="ADJ" hw="this" c5="DT0">This </w>
      <w pos="SUBST" hw="virus" c5="NN1">virus </w>
      <w pos="VERB" hw="affect" c5="VVZ">affects </w>
      <w pos="ART" hw="the" c5="AT0">the </w>
      <w pos="SUBST" hw="body" c5="NN1">body</w>
      <w pos="UNC" hw="'s" c5="POS">'s </w>
      <w pos="SUBST" hw="defence" c5="NN1">defence </w>
      <w pos="SUBST" hw="system" c5="NN1">system </w>
    - <mw c5="CJS">
        <w pos="ADV" hw="so" c5="AV0">so </w>
        <w pos="CONJ" hw="that" c5="CJT">that </w>
      </mw>
      <w pos="PRON" hw="it" c5="PNP">it </w>
      <w pos="VERB" hw="can" c5="VM0">can</w>
      <w pos="ADV" hw="not" c5="XX0">not </w>
      <w pos="VERB" hw="fight" c5="VVI">fight </w>
      <w pos="SUBST" hw="infection" c5="NN1">infection</w>
      <c c5="PUN">.</c>
    </s>
</p>
```

Image 10 an excerpt of the British National Corpus

4.2.3 DBPedia

DBPedia is the relational database version of Wikipedia (Bizer et al., 2009). It lets user perform complex queries on articles.

DBPedia is mentioned here for completeness, however for the purposes of this work the standard article classification was sufficient to adequately gather articles for processing.

4.2.4 WordNet

WordNet was born as an attempt to build a dictionary that could exploit the capabilities of computers (Miller, 1990) (Fellbaum, 1998). Standard dictionaries order words alphabetically and this is not always the most efficient method to find words. WordNet was built around the structure of the English language using the help of linguists and psychologists. The WordNet project started in 1985.

WordNet divides words into nouns, verbs, adjectives, and function words, although function words are not included in the database. Unlike standard dictionaries, words in WordNet can be included more than once if they have the potential to be different Parts Of Speech depending on context.

WordNet differentiates between word forms and word meanings. Word form is how a word is written or spoken. Word meaning is formally intended as "lexicalised concepts that [word] forms can be used to express. ".

Each word in WordNet represented as a list of its synonyms called Synsets. Synsets can be thought of as the nodes of the WordNet network. Synsets are connected via the following relations:

4.2.4.1 Relationships Between Nouns in WordNet

The taxonomy shown below was taken from Wikipedia[15] and shows all Synset relations for both verbs and nouns:

hypernyms: B is a hypernym of A if every A is a (kind of) B (canine is a hypernym of dog)

hyponyms: B is a hyponym of A if every B is a (kind of) A (dog is a hyponym of canine)

coordinate terms: B is a coordinate term of A if *A and b* share a hypernym (wolf is a coordinate term of dog, and dog is a coordinate term of wolf)

meronym: B is a meronym of A if B is a part of A(window is a meronym of building)

[15]https://en.Wikipedia.org/Wiki/WordNet

holonym: B is a holonym of A if A is a part of if B(building is a holonym of window)

4.2.4.2 Relationships Between Verbs in WordNet

hypernym: the verb B is a hypernym of the verb A if the activity A is a (kind of) B (to perceive is an hypernym of to listen)

troponym: the verb B is a troponym of the verb A if the activity B is doing A in some manner (to lisp is a troponym of to talk)

entailment: the verb B is entailed by A if by doing A you must be doing B (to sleep is entailed by to snore)

coordinate terms: those verbs sharing a common hypernym (to lisp and to yell

Diagram 2 shows a visual representation of Synsets relationships in WordNet

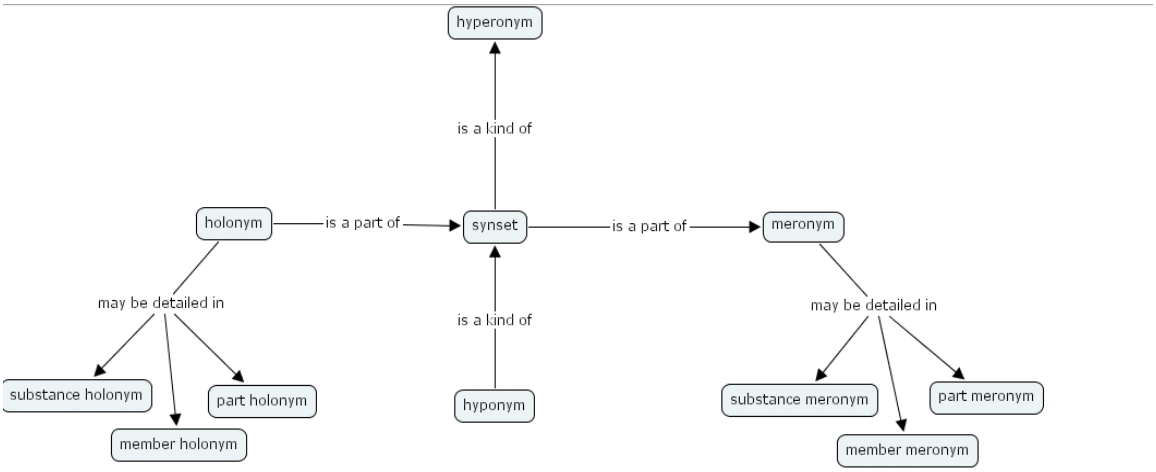


Diagram 2 WordNet Synset Relationships[16]

4.2.5 Word Association Norms or Human Association Datasets

Another source of information for word relatedness is Word Association Norms. Word Association Norms are human compiled list of cue and responses. They are compiled by psycholinguists by asking a set of participants to state the first thing that comes to their mind when they hear a word. Usually these lists are used to evaluate relatedness

[16]http://cmapspublic.ihmc.us/rid=1119191792718_1287770555_278/WordNet%20Synset%20details.cmap?rid=1119191792718_1287770555_278&partName=htmljpeg

measures by calculating the Spearman Rank Correlation Coefficient (SRC) of the two lists (Wojtinnek et al., 2012). The SRC is a measure of statistical dependence between two variables. (Yule and Kendall, 1968) To calculate Spearman Correlation between two sets of data, the two sets are ranked, then the following equation is used to obtain the coefficient

$$\rho = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_i (x_i - \bar{x})^2 \, (y_i - \bar{y})^2}}$$

Where x represents element from the first list and y represents elements from the second list, $\bar{x}$ and $\bar{y}$ represent the mean ranking for each list

Equation 7 Spearman Rank Correlation Coefficient[17]

According to Washtell and Markert (Washtell and Markert, 2009) the most popular Word Association Datasets are:

- The Kent Dataset (Kent and Rosanoff, 1910)
- The Minnesota Dataset (Russell et al., 1954)
- The Edinburgh Associative Thesaurus (Kiss et al., 1973)
- The Florida Dataset (Nelson and McEvoy, 1998)

Figure 18 (below) shows an excerpt of the Edinburgh Associative Thesaurus (EAT). The first column contains the target word presented to the participants and the second column contains what participants answered upon hearing the target word. The important data to analyse word associations is contained in the 4th, 5th, 6th and 7th column of the EAT dataset. The 4th column indicates how many people were presented with the target word. The 5th column indicates how many people out of those who were presented with the target word responded with the cue word in the second column. The 6th column indicates the forward association strength calculated by dividing the number of people who gave a particular response given a target by the total number of people who were presented with that target word. The 7th column contains the backward association strength which is calculated in the same way as the forward association strength by reversing the cue and target word.

The other datasets mentioned above contain the same data formatted in a different way.

[17] https://statistics.laerd.com/statistical-guides/spearmans-rank-order-correlation-statistical-guide-2.php

```
FOOD, EAT, YES, 180, 73, .406, .409, .0100, .0099, 9, 2, 4.0, 1,
18, 147, 5.84, , N, 1.69, .63, .203, 1, 11, 61, 4.75, , N, 1.20,
.70, .181, 1
FOOD, DRINK, YES, 180, 9, .050, 0.000, .0650, .0200, 2, 2, 2.0,
0, 18, 147, 5.84, , N, 1.69, .63, .203, 1, 20, 82, 4.95, G, V,
2.58, .90, .117, 1
FOOD, HUNGER, YES, 180, 7, .039, .364, .0070, .0337, 1, 2, 2.0,
1, 18, 147, 5.84, , N, 1.69, .63, .203, 1, 15, 17, 4.16, , N,
2.46, .31, .030, 1
FOOD, HUNGRY, YES, 180, 6, .033, .274, .0010, .0123, 2, 2, 2.0,
1, 18, 147, 5.84, , N, 1.69, .63, .203, 1, 12, 23, 4.24, , AJ,
1.50, .58, .159, 1
FOOD, FAT, YES, 180, 4, .022, 0.000, .0250, 0.0000, 1, 2, 0.0, 0,
18, 147, 5.84, , N, 1.69, .63, .203, 1, 16, 60, 4.80, , N, 1.47,
.60, .229, 1
FOOD, GOOD, YES, 180, 4, .022, 0.000, .0030, 0.0000, 2, 2, 0.0,
0, 18, 147, 5.84, , N, 1.69, .63, .203, 1, 8, 807, 2.53, , AJ,
.75, .75, .586, 1
FOOD, HEALTH, YES, 180, 4, .022, .049, 0.0000, .0029, 0, 2, 1.0,
1, 18, 147, 5.84, , N, 1.69, .63, .203, 1, 21, 105, 3.54, , N,
1.71, .24, .004, 1
FOOD, FRUIT, YES, 180, 3, .017, 0.000, .0020, .0134, 1, 2, 2.0,
1, 18, 147, 5.84, , N, 1.69, .63, .203, 1, 17, 35, 6.00, G, N,
1.47, .41, .103, 1
FOOD, MEAL, YES, 180, 3, .017, .488, .0080, .0263, 2, 2, 2.0, 0,
18, 147, 5.84, , N, 1.69, .63, .203, 1, 6, 30, 5.98, T, N, 2.50,
.83, .033, 1
FOOD, STEAK, YES, 180, 3, .017, .056, 0.0000, .0060, 0, 2, 2.0,
2, 18, 147, 5.84, , N, 1.69, .63, .203, 1, 19, 10, 6.47, , N,
2.20, .33, .065, 1
FOOD, WATER, YES, 180, 3, .017, 0.000, .0080, .0083, 3, 2, 1.0,
0, 18, 147, 5.84, , N, 1.69, .63, .203, 1, 18, 442, 6.27, , N,
2.00, .94, .146, 1
FOOD, APPLE, YES, 180, 2, .011, .020, .0040, .0079, 2, 2, 2.0, 1,
18, 147, 5.84, , N, 1.69, .63, .203, 1, 17, 9, 7.00, , V, 1.82,
.77, .090, 1
FOOD, LUNCH, YES, 180, 2, .011, .205, .0020, .0354, 2, 2, 3.0, 1,
18, 147, 5.84, , N, 1.69, .63, .203, 1, 17, 33, 5.46, , N, 1.88,
.59, .053, 1
FOOD, MOUTH, YES, 180, 2, .011, .022, 0.0000, .0156, 0, 2, 2.0,
2, 18, 147, 5.84, , N, 1.69, .63, .203, 1, 19, 103, 5.47, , N,
1.58, .47, .098, 1
FOOD, NOURISH, NO, 180, 2, .011, ¥, ¥, ¥, ¥, ¥, ¥, ¥, 18, 147,
5.84, , N, 1.69, ¥, ¥, ¥, ¥, ¥, ¥, , , ¥, ¥, ¥, 0
FOOD, NOURISHMENT, NO, 180, 2, .011, ¥, ¥, ¥, ¥, ¥, ¥, ¥, 18,
147, 5.84, , N, 1.69, ¥, ¥, ¥, ¥, ¥, ¥, , , ¥, ¥, ¥, 0
FOOD, POISON, YES, 180, 2, .011, 0.000, 0.0000, .0002, 0, 2, 1.0,
1, 18, 147, 5.84, , N, 1.69, .63, .203, 1, 18, 10, 5.65, , N,
.75, .19, .022, 1
FOOD, SUBSTANCE, YES, 180, 2, .011, .027, 0.0000, .0003, 0, 2,
1.0, 1, 18, 147, 5.84, , N, 1.69, .63, .203, 1, 18, 33, ¥, G, N,
```

Figure 18 Example Dataset

4.3 Web Based Corpora

A new kind of corpus has emerged in recent years that adds a new category to Aston's classification given at the start of this Chapter: Web Corpora (Volk, 2002). There is however, a distinction that must be made between the goals of Corpus Linguists and the goals of this work. The main goal of linguists is to understand more about the language and analyse its structure. The goal of this work is to exploit language structure to identify related words and generate meaningful sentences. As stated

previously, the output of this process is directly presented to users. Therefore it is crucial to identify a suitable corpus that contains words that a person would use in conversation whilst not being conversational. The web contains an infinite amount of information so it is simply not sufficient to just download random texts, there has to be preparation and careful selection of data (Kehoe and Renouf, 2002). Kehoe and Renouf and Volk (Volk, 2002) show that some pre-processing is required to identify suitable web pages. A solution to this is to use Web references websites such as Wikipedia. Wikipedia like reference texts have the advantage of containing the topic of the article in the title and articles are categories using human friendly terminology created by editors and readers themselves. In corpora such as the Brown and the BNC, documents are and sorted using alphanumerical systems, each document of the corpus has to be queried or read to understand what the topic of each document is.

Using web reference texts instead of prose or journalistic sources has also the advantage of making the search for documents which are related to a specific subject easier thus, in theory, limiting the amount of unrelated words, there will still be a number of unrelated words even in specified sources but they can be reduced by using appropriate techniques. This problem was encountered in this work and it was dealt with by using crowdsourced human judgement, discussed at the end of this Chapter.

4.4 Justification on the Use of Wikipedia Over Other Web Corpora

There are also other reasons why web reference texts are more suited as corpora: The most important reason is that the task of measuring Semantic Similarity requires a large corpora. For example Wettler and Rapp (Wettler and Rapp, 1993) used the corpora listed in this Chapter, in conjunction with other corpora, to form a corpora of 33 million words to measure word associations. However the number of words is not the main measure to consider. As repeatedly stated, one of the goals of this research is to improve a communication tool that must cater for the needs of the user in a variety of situations that, ideally, are not even foreseen during development. Therefore the main measure when selecting the corpus, must be the variety of topics and how difficult it is to determine if documents are about a specific topic. For the reasons given above

Wikipedia is an ideal candidate as it contains virtually every topic possible, its size also ensures word vocabulary variety.

Wikipedia has been successfully used to build Semantic Networks in the work of others. For example the work by Wojtinnek and Pulman (Wojtinnek et al., 2012) has shown that using Wikipedia as a corpus for building Semantic Networks is more suited than using either uncategorised plain text corpora or network based corpora such as WordNet.

Other work by Damani, Chedda and Chaudari (Damani et al., 2012) has shown that using Wikipedia as a corpus with PMI ,SCI and PMIsig provides satisfactory results. Whilst the work aims to compare web based corpora to Wikipedia when using word similarity measurements, it showed that PMI and its significance based alternatives work well with Wikipedia. Further it showed that **Wikipedia performs far better as a corpus for word similarity measurement when measuring the window of co-occurrence in the same document, rather than measuring it as the number of documents that contain the word pair (as in cPMId)**, that is a further reason why the work by Damani and Ghonge and the work by Chaudari Damani and Laxman (Chaudhari, Damani, and Laxman 2011) was not used in this work. Finally Damani, Chedda and Chaudari themselves show that **appropriate window sizes to use with Wikipedia are between 5 and 10 (direction independent).**

Using Wikipedia offers flexibility as specific topics can be targeted having in some cases the same amount of words as the Brown Corpus and the LOB for a single topic. Wikipedia can be obtained either by downloading the latest dumps in XML format or by downloading the web pages directly with different mechanisms.

4.5 Using Wikipedia in Practice

The downside of downloading Wikipedia is its very large size. A basic dump of all of Wikipedia is 9 GB[18] of compressed text that generates a 40GB single XML file containing several million pages of content when uncompressed.

Using Wikipedia dumps is a two-step process:

The first step is to download the latest dump, which can be downloaded as a single file or by downloading 25 separate files. For our purposes it was decided to download 5 files (3 million words).

[18] https://dumps.Wikimedia.org/

The second step is to convert the XML file into text and remove references and citations. This was done by developing a java application with the use of an open source library called Bliki which is freely available for non-profit purposes. This library generates a text file for each Wikipedia article in the XML file.

The content of the page is contained within the <text> tag. However text is enclosed within other tags used for citations and links to other pages.

Example:

*{{about|**the domestic dog**|related species known as "dogs"|Canidae|other uses|Dog (disambiguation)|}} {{pp-semi-indef}}{{pp-move-indef}} {{Taxobox | name = Domestic dog | fossil_range = {{Fossil range|0.033|0}}<small>[[Pleistocene]] –[[Recent]]</small> | status = DOM | image = YellowLabradorLooking new.jpg <!-- Please do not change lead image without discussion --> | image_width = 260px | image_caption = Yellow [[Labrador Retriever]], the most registered breed of 2009 with the [[American Kennel Club|AKC]] | regnum = [[Animal]]ia | phylum = [[Chordate|Chordata]] | classis = [[Mammal]]ia | ordo = [[Carnivora]] | familia = [[Canidae]] | genus = ''[[Canis]]'' | species = ''[[Gray Wolf|C. lupus]]'' | subspecies = '''''C. l. familiaris'''''<ref name = "MSW3 Canis lupus familiaris">{{cite web|url=http://www.bucknell.edu/MSW3/browse.asp?id=14000752|title=Mammal Species of the World – Browse: Canis lupus familiaris|publisher=Bucknell.edu |year=2005 |accessdate=12 March 2012}}</ref> | trinomial = ''Canis lupus familiaris''<ref name="MSW3 Lupus"/> | synonyms = {{collapsible list|bullets = true |title=<small>Species synonymy</small> |''aegyptius'' Linnaeus, 1758 |''alco'' C. E. H. Smith, 1839 |''americanus'' Gmelin, 1792 |''anglicus'' Gmelin, 1792 |''antarcticus'' Gmelin, 1792 |''aprinus'' Gmelin, 1792 |''aquaticus'' Linnaeus, 1758 |''aquatilis'' Gmelin, 1792 |''avicularis'' Gmelin, 1792 |''borealis'' C. E. H. Smith, 1839 |''brevipilis'' Gmelin, 1792 |''cursorius'' Gmelin, 1792 |''domesticus'' Linnaeus, 1758 |''extrarius'' Gmelin, 1792 |''ferus'' C. E. H. Smith, 1839 |''fricator'' Gmelin, 1792 |''fricatrix'' Linnaeus, 1758 |''fuillus'' Gmelin, 1792 |''gallicus'' Gmelin, 1792 |''glaucus'' C. E. H. Smith, 1839 |''graius'' Linnaeus, 1758 |''grajus'' Gmelin, 1792 |''hagenbecki'' Krumbiegel, 1950 |''haitensis'' C. E. H. Smith, 1839 |''hibernicus'' Gmelin, 1792 |''hirsutus'' Gmelin, 1792 |''hybridus'' Gmelin, 1792 |''islandicus'' Gmelin, 1792 |''italicus'' Gmelin, 1792 |''laniarius'' Gmelin, 1792 |''leoninus'' Gmelin, 1792 |''leporarius'' C. E. H. Smith, 1839 |''major'' Gmelin, 1792 |''mastinus'' Linnaeus, 1758 |''melitacus'' Gmelin, 1792 |''melitaeus'' Linnaeus, 1758 |''minor'' Gmelin, 1792 |''molossus'' Gmelin, 1792 |''mustelinus'' Linnaeus, 1758 |''obesus'' Gmelin, 1792 |''orientalis'' Gmelin, 1792 |''pacificus'' C. E. H. Smith, 1839 |''plancus'' Gmelin, 1792 |''pomeranus'' Gmelin, 1792 |''sagaces'' C. E. H. Smith, 1839 |''sanguinarius'' C. E. H. Smith, 1839 |''sagax'' Linnaeus, 1758 |''scoticus'' Gmelin, 1792 |''sibiricus'' Gmelin, 1792 |''suillus'' C. E. H. Smith, 1839 |''terraenovae'' C. E. H. Smith, 1839 |''terrarius'' C. E. H. Smith, 1839 |''turcicus'' Gmelin, 1792 |''urcani'' C. E. H. Smith, 1839 |''variegatus'' Gmelin, 1792 |''venaticus'' Gmelin, 1792 |''vertegus'' Gmelin, 1792 }} }} The domestic '''dog''' (''Canis lupus familiaris'')<ref name="MSW3 Lupus">{{cite web|url=http://www.bucknell.edu/MSW3/browse.asp?id=14000738 |title=Mammal Species of the World – Browse: lupus |publisher=Bucknell.edu |accessdate=10 August 2010}}</ref><ref name="ADW"/> is a [[gray wolf subspecies|subspecies of]] the [[gray wolf]] (''Canis lupus''), a member of the [[Canidae]] family of the [[mammal]]ian order [[Carnivora]]. The term "domestic dog" is generally used for both [[domesticated]] and [[feral]] varieties. The dog was the first domesticated animal<ref name=altai-dog-dna/> and has been the most widely kept [[working dog|working]], [[hunting]], and [[pet]] animal in human history. The word "dog" can also refer to the male of a canine species,<ref>{{cite web |url=http://dictionary.reference.com/browse/dog |title=Dog |publisher=[[Dictionary.com]] }}</ref> as opposed to the word "bitch" which refers to the female of the species. [[MtDNA]] evidence shows an evolutionary split between the modern dog's lineage and the modern wolf's lineage around 100,000 years ago.{{Citation needed|reason=citations in Wikipedia articles contradict this|date=January 2014}} A recent study of "well-preserved remains of a dog-like canid from the Razboinichya Cave" in the Altai Mountains of southern Siberia concluded that a particular instance of early wolf domestication approximately 33,000 years ago did not result in modern dog lineages, possibly because of climate disruption during the [[Last Glacial Maximum]].<ref name=altai-dog-dna>{{cite journal |doi=10.1371/journal.pone.0057754 |title=Ancient DNA Analysis Affirms the Canid from Altai as a Primitive Dog |year=2013 |editor1-last=Hofreiter |editor1-first=Michael |last1=Druzhkova |first1=Anna S. |last2=Thalmann*

|first2=Olaf |last3=Trifonov |first3=Vladimir A. |last4=Leonard |first4=Jennifer A. |last5=Vorobieva |first5=Nadezhda V. |last6=Ovodov |first6=Nikolai D. |last7=Graphodatsky |first7=Alexander S. |last8=Wayne |first8=Robert K. |journal=PLoS ONE |volume=8 |issue=3 |pages=e57754 |pmid=23483925 |pmc=3590291}}</ref> The authors postulate that at least several such incipient events have occurred. A study of fossil dogs and wolves in Belgium, Ukraine, and Russia tentatively dates domestication from 14,000 years ago to more than 31,700 years ago.<ref>Germonpréa, Mietje; Sablin, Mikhail V.; Stevens, =Rhiannon E.; Hedges, Robert E.M.; Hofreiter, Michael; Stiller, Mathias; Després, Viviane R. (February 2009). ''Fossil dogs and wolves from Palaeolithic sites in Belgium, the Ukraine and Russia: osteometry, ancient DNA and stable isotopes''. Journal of Archaeological Science 36 (2): 473–90. doi:10.1016/j.jas.2008.09.033</ref> Another recent study has found support for claims of dog domestication between 14,000 and 16,000 years ago, with a range between 9,000 and 34,000 years ago, depending on mutation rate assumptions

The open source library Bliki[19] was used to remove tags and convert the content of the dumps to plain text. Bliki is freely available to use and it is mainly used for converting Wikipedia dumps content to HTML .The open source community has implemented a wrapper to convert the HTML output produced by Bliki to text.

```
The domestic dog (Canis lupus familiaris)  is a subspecies of the gray wolf
(Canis lupus), a member of the Canidae family of the mammalian order
Carnivora.
The term "domestic dog" is generally used for both domesticated and feral
varieties.
The dog was the first domesticated animal and has been the most widely kept
working, hunting, and pet animal in human history.
The word "dog" can also refer to the male of a canine species,  as opposed to
the word "bitch" which refers to the female of the species.
MtDNA evidence shows an evolutionary split between the modern dog's lineage
and the modern wolf's lineage around 100,000 years ago.
A recent study of "well-preserved remains of a dog-like canid from the
Razboinichya Cave" in the Altai Mountains of southern Siberia concluded that
a particular instance of early wolf domestication approximately 33,000 years
ago did not result in modern dog lineages, possibly because of climate
disruption during the Last Glacial Maximum.
The authors postulate that at least several such incipient events have
occurred.
A study of fossil dogs and wolves in Belgium, Ukraine, and Russia tentatively
dates domestication from 14,000 years ago to more than 31,700 years ago.
Most breeds of dogs are at most a few hundred years old, having been
artificially selected for particular morphologies and behaviors by people for
specific functional roles.
Through this selective breeding, the dog has developed into hundreds of
varied breeds, and shows more behavioral and morphological variation than any
other land mammal.
It is common for most breeds to shed this coat.
```

[19]https://code.google.com/p/gwtWiki/downloads/detail?name=bliki.3.0.19.zip&can=2&q=

The classification was included in the application to work as soon as a text file was generated to classify it and place it into the appropriate folder. These folders are later used during the language pre-processing. Later it was decided to download the articles directly from the web pages, by using Wikipedia categorisation as a reference. Downloading articles proved to be faster, and it gave greater control over the content of the articles.

4.6 The Use of Tagged Corpora as Reference for Document Processing

During pre-processing phase, stemming required and later it was also discovered that POS filtering would be also be required It was decided to exploit the information contained in the BNC to perform the tasks of stemming and POS filtering by querying specific tags in the XML. As will be explained later only nouns were kept in the final file generated by the pre-processor. Justification for this will be given in the appropriate section.

Part of Speech filtering was also performed by looking up BNC tags and checking whether each word was a noun. If the word is not a noun the pre-processor discards it. This decision was made to shift the burden of sentence realisation on the application. Wikipedia is not a conversational corpus so if all parts of speech are kept the resulting dictionaries would contain adjectives and verbs which are not normally present in simple conversation such as those carried out by our target users.

4.7 Corpus Size Reduction and Filtering Via the Use of Naïve Bayes Classification

4.7.1 Theoretical Background of Naïve Bayes Classification

In cases like the ones at hand, where we have a set of documents and we have to determine to which category they belong, a popular method is to employ Naïve Bayes Classification (NBC) (Jurafsky and Martin, 2000). NBC is a machine learning technique as it works by training a computer to classify documents.

To carry out NBC one starts off with a set of documents D, where each document d belongs to the D set and a fixed set of classes C where each class c belongs to the set C. The goal of NBC is to determine the class of each document. In practical terms to

determine to which category each document belongs. Machine Learning algorithms are either supervised or unsupervised. The difference lies in whether the learning process requires the help of a human supervisor. NBC is a supervised learning algorithm as it requires a human to select documents for training. Documents have to be representative of each class and have to be labelled to enable the classifier to recognise documents of a particular class. The equation below shows how to compute the probability of a document belonging to a particular class ($P(c|d)$ is equivalent to probability of class c given a document d). In the "equation" the alpha symbols the alpha symbol means "proportional to".

$$P(c|d) \propto P(c) \prod_{1 \leq k \leq n_d} P(t_k|c)$$

Equation 8 Naïve Bayes Classification

Formally Jurafsky and Martin explain this equation as follows : the most likely class is the class that maximises the product of the prior probability of the class and the probability of the of each term (word) in the document belonging to a class".

To compute this in practice, as mentioned training documents are required, each of the training documents is treated as a set of terms, (each term is referred to as $t_k$in the equation) .This representation is referred to as "bag of words" because each document becomes a collection of terms and their frequency regardless of their order in the document. These terms become representative of the class they belong to. When the classifier encounters new documents the presence of these terms increases the likelihood of them belonging to a particular class. After having selected the training set, one has to compute the probability of each term occurring in a particular class by calculating the product of the probability of each term belonging to a class. (i.e. $P(t_k|c)$) and the probability of the class (how likely the class is to occur at all), simply by counting the number of documents in each particular class and dividing them by number of classes.

Once the classifier is trained it uses the values calculated during training to classify documents.

4.7.2 Document Classification Using NBC in Weka

In this Chapter the word class will mean the category to which a specific document belongs. Not to be confused with a class as intended in Software Engineering.

At the early stages of this work when still using Wikipedia dumps, even though only a small percentage of Wikipedia was downloaded, the process still generated a large amount of documents, therefore some form of document filtering was required. It was decided to filter document based on topics, and to discard those documents that did not discuss the chosen topics. Therefore a Naïve Bayes classifier was created and trained to classify text files, using the open source software Weka(Hall et al., 2009). The process of text classification with Naïve Bayes in general and with Weka is the following[20]:

1) Stem, lowercase and remove stopwords.
2) Remove words that appear below a certain frequency.
3) Turn the document into a vector of word, each element of the vector is the frequency of the word, or the TFIDF of the word.
4) Train the classifier
5) Evaluate the classifier

TF*IDF is an information retrieval measure for a corpus calculated for each word by multiplying the number of times a word appears in each document and its inverse document frequency , which is calculated by dividing the total number of documents in the corpus by the number of documents which contain the word. It is expressed by the formula

$$\mathrm{idf}_t = \log \frac{N}{\mathrm{df}_t}.$$

Term frequency can be further calculated in three ways, Boolean, logarithmical and numerical. We used numerical i.e. using a raw count of term frequency.

The classifier was trained using four initial categories

- Food and drink
- Music
- Sports
- Transport

These topics were taken from a paper version of AAC which is given to patients in the hospital when they show symptoms of aphasia after being admitted (drawn in figure

[20]https://weka.Wikispaces.com/Programmatic+Use

below for illustration). It was given to the author by the Speech Therapist with whom the initial interview was conducted.

Figure 19 Conversational Categories of AAC book

4.8 Evaluation of the Naïve Bayes Classifier

As is standard when evaluating classifiers, a small subset of the dataset was removed and reserved for evaluation. Testing worked by placing the test documents into appropriate folders so that they would be labelled to evaluate the accuracy of classification.

For each class, 35 Wikipedia articles were used for training and 10 for evaluation.

Evaluation of the classifier using a testing set produced 97% accuracy.

The main objective of this process was to integrate the classifier inside the Java application that parses the Wikipedia dump files, so that Wiki articles could be classified as soon as they are parsed. Weka provides libraries so that classifiers can be integrated into applications.

Because the classifier always tries to classify documents within the four categories, it initially produced incorrect results when classifying documents about topics other than the four it was trained for. To avoid this instead of letting the classifier classify the documents by auto selecting the highest probability, the application looks at the

probability for each class and looks at the value. The application classifies a document as "other" if the probability of belonging to every class is below 0.6.

Feature reduction was also applied to the classifier to exclude noise from the classifier vocabulary. The process in WEKA is called attribute selection. According to literature, the most widely used attribute selector when performing Naïve Bayes is information gain with a threshold of 0 or above, information gain is calculated as the difference between the entropy of the class and the entropy of the class given the attribute.

Further testing led to the increase of the probability threshold for classification. The threshold was increased from 0.6 to 0.85.

4.9 Using Wikipedia Classification to Filter Articles

Due to the long times required and the difficulties in categorising articles when using dumps, it was eventually decided to build a corpus by exploiting Wikipedia's own categorisation using the same categories that were used for the Naïve Bayes classifier.

Wikipedia contains a special page which lets users navigate through its categories. The following super categories were chosen for each topic.

Food and Drink - "Foods"

Music – "Music "

Sports – "Sports "

Transport - "Transport

A master category was selected for each topic. It was decided to download articles hierarchically down to a depth of 3 tiers from the master category. In practice articles were obtained by implementing a web crawler by using a Web API called JSoup[21]. Articles were collected by recursively scanning the subcategories starting from the four master categories .as a new subcategory was encountered all articles for that category were downloaded. The crawler sent HTTP requests to the Wikipedia Category pages and queried for the name of the sub categories and the name of the articles by querying the XHTML attributes of the relevant nodes.

[21] **jsoup**.org/

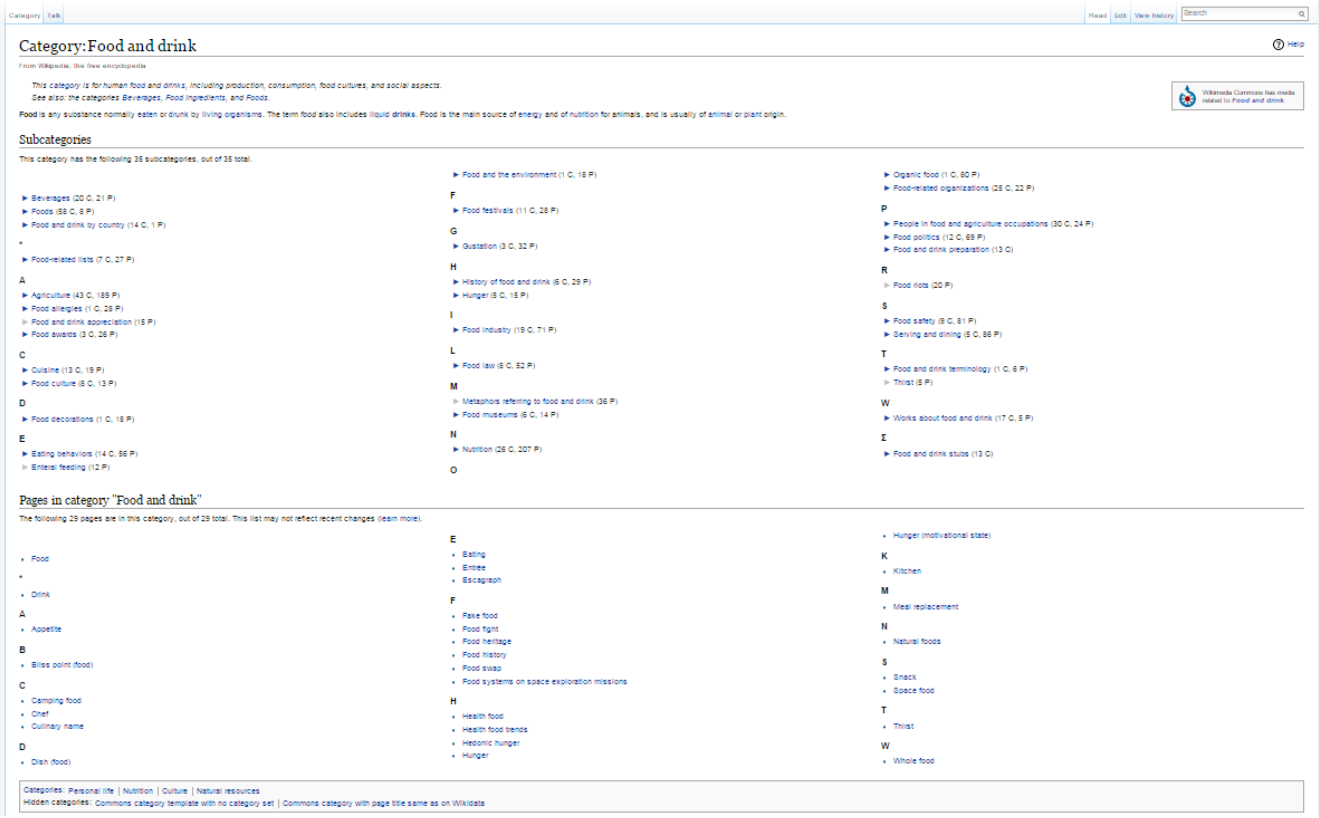


Figure 20An example Category Page from Wikipedia. Used by the Web Crawler to Gather Relevant Articles

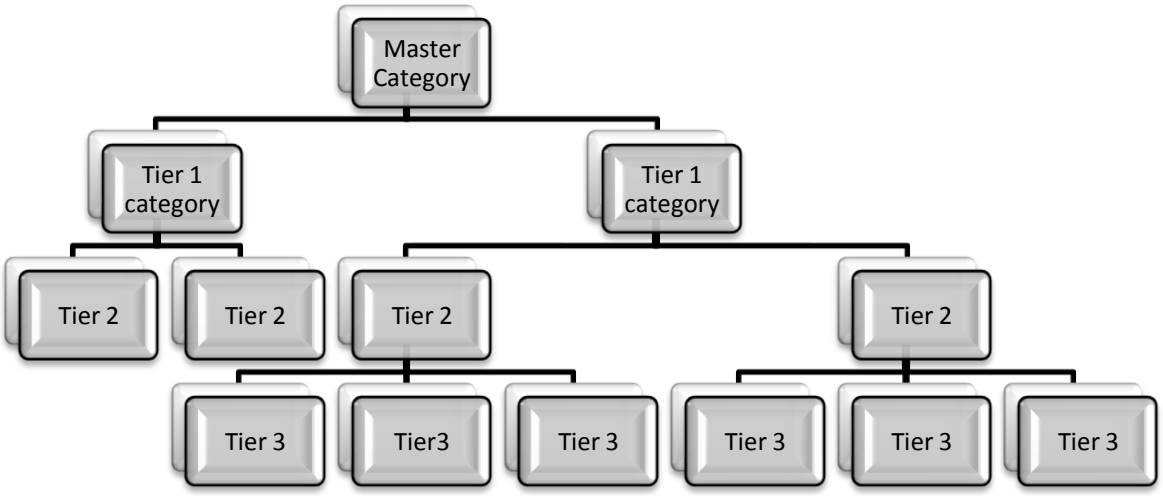


Figure 21 An ideal Representation of Tiers to Understand How the Crawler Works. (Not Representative of Wikipedia).

Wikipedia is not structured as a standard hierarchy (as represented in the figure above) so it can happen that some sub categories have subcategories in common with a previous tier. To eliminate the effects of this duplicate articles were eliminated when grouping the articles. In practice when the crawling process is finished articles are grouped into one folder that is then passed to the pre-processor. When files are copied to the folder duplicate articles are eliminated.

Collection of articles is used by the pre-processor to produce topic vocabularies.

In total the process of Wikipedia classification produced 24 different topic vocabularies which all could potentially be used as input for the pre-processor.

Twelve vocabularies were produced by using all of the categories at tier/depth 1 and the other twelve were produced by filtering out categories at depth 1 as shown in the table.

Categories shown in bold are categories that were kept, others were discarded to produce filtered categories vocabularies.

4.10 User Evaluation of Best Subset of Wikipedia to Use for AAC App

To evaluate which set of articles produced the best vocabularies to use for the AAC application it was decided to use crowdsourcing.
Crowdsourcing means posting a work for participants (called workers) to carry out (which takes the name of job). The most popular and first website of this kind is Amazon Mechanical Turk. However for fiscal reasons taxpayers outside the UK cannot posts job. The best alternative is another website called CrowdFlower. CrowdFlower is not as powerful as Amazon MT but ways can be found to achieve the intended result.

The job was designed and posted as follows. A subset of 100 words was taken from each topic vocabulary. The four subset were merged into one spreadsheet and the order of the words was randomised. The task for each worker was to classify each word in the spreadsheet.

It was decided to use a representative subset of the corpus for two reasons: stressfulness of the task and costs. It would have been unfeasible to ask participants to scan through all words of all vocabularies. In total there were two sets (all categories and handpicked categories) of four vocabularies for each depth, for a total of three depths. On average each vocabulary contained 5000 words at depth 1 with the count doubling at each depth. Therefore including all words would have been not only a tedious task for users but also a costly one.

Workers classified words by using radiobuttons. Under each word there were 5 radiobuttons:

- Food
- Music
- Sports
- Transport
- Other

If a worker could not comfortably classify a word as one of the four vocabulary topics then he/she selected “other.” The image below shows questions as they were presented to users.

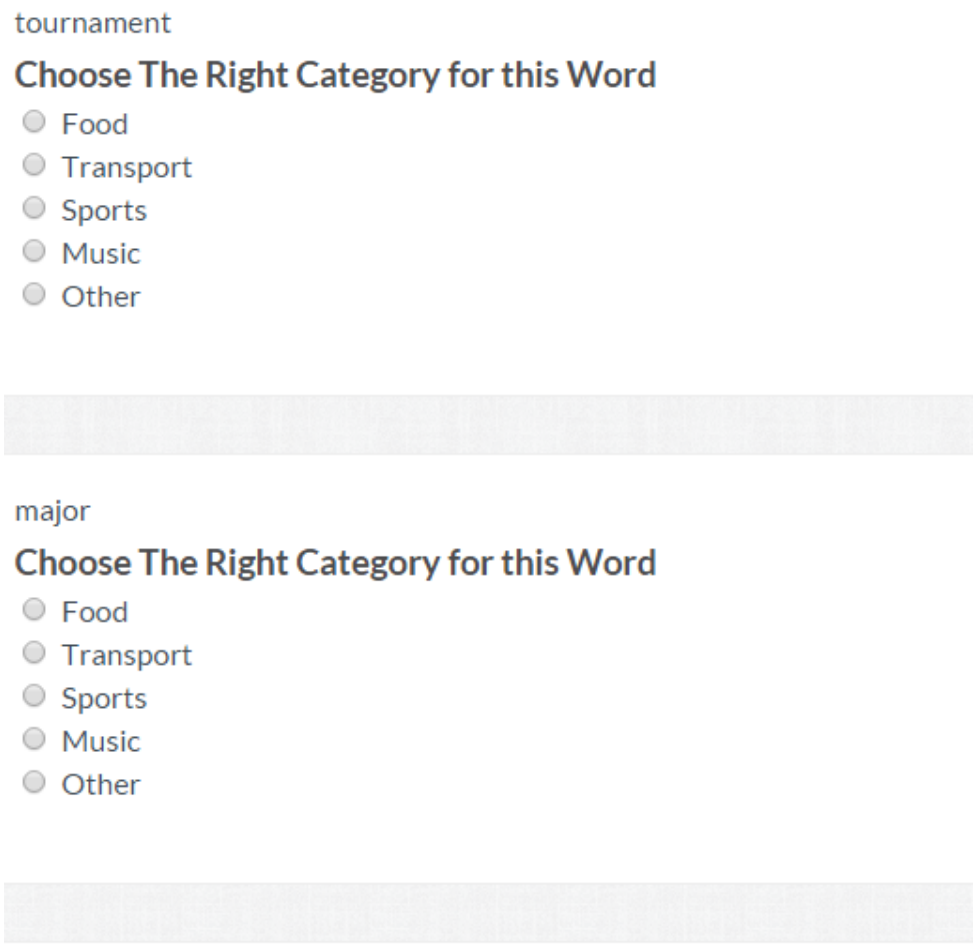


Image 11 Job as Presented to Workers

Once a job is designed, the next step is to decide how many judgments per row are required for the job to be considered complete and how many rows of the spreadsheet will appear on each page. Since there were a total of 400 words it was decided that each page would contain 40 rows splitting the data over 10 pages.

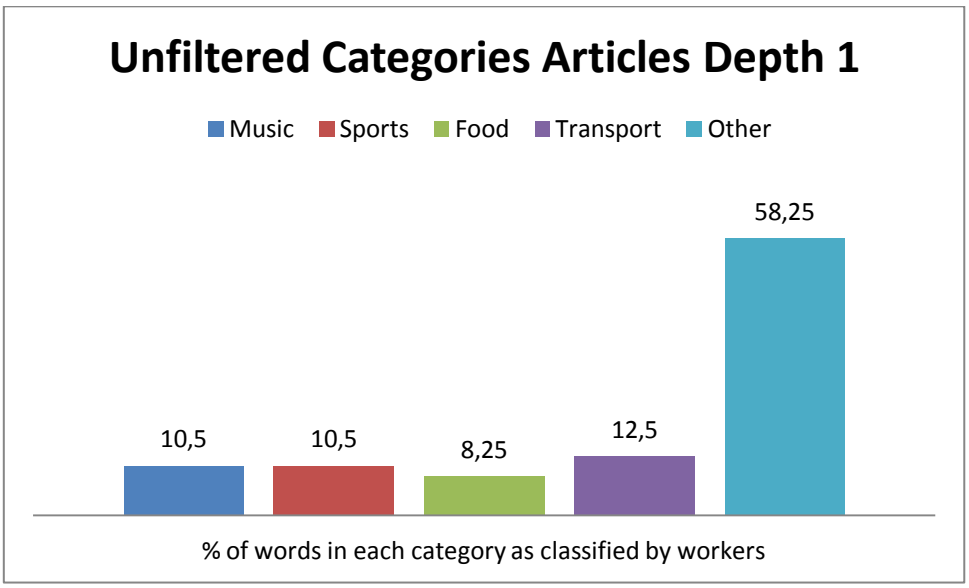
In total 6 jobs were posted on CrowdFlower. Each job contained a depth of Wikipedia. Three jobs were posted for Unfiltered categories and three for Filtered categories.

To evaluate the quality of the six different corpora, statistics were calculated based on answers to which the majority of users agreed on. The next paragraph will discuss the results of the evaluation in detail. The goal of the evaluation was to find the corpus which contained the least words which the majority of workers agreed should be classified as "Other"

4.11 Results of Crowdsourcing Based Evaluation

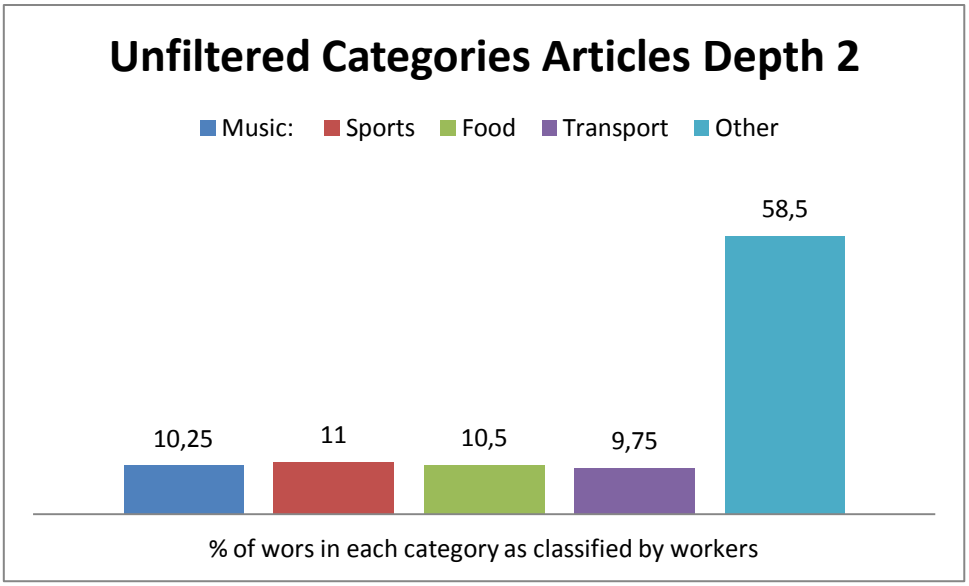
4.11.1 Unfiltered Article Results

Topics	Raw count	% of words classified by users
Music	42	10,5
Sports	42	10,5
Food	33	8,25
Transport	50	12,5
Other Depth 1	233	58,25



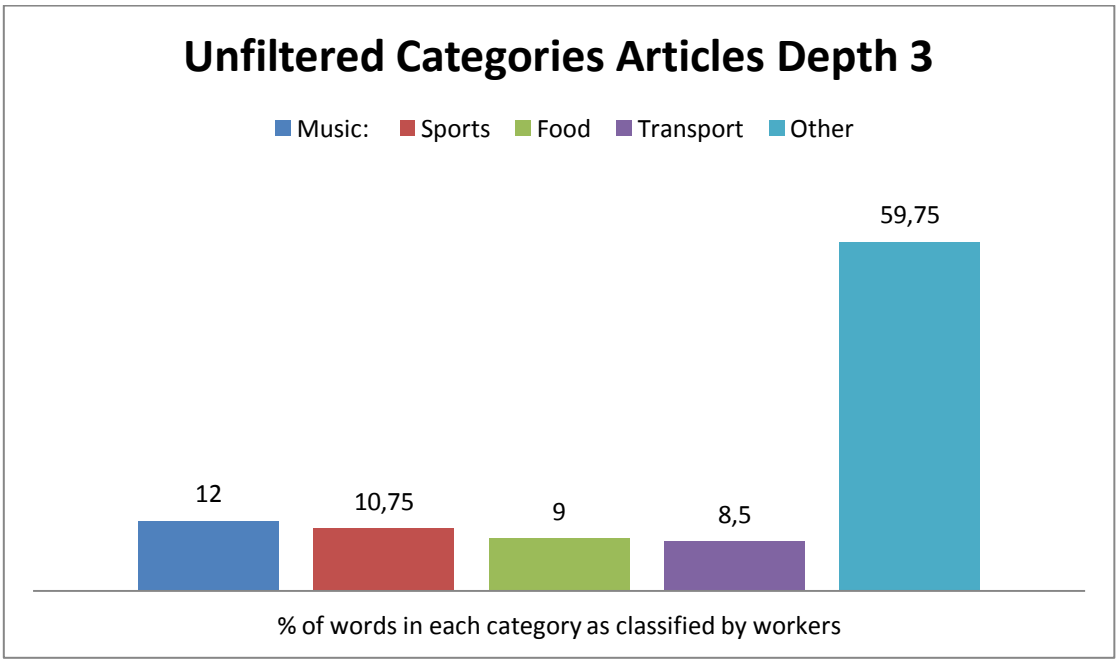
Graph 1 Classification of Words in Unfiltered Articles at Depth 1

	Nr	% of words classified by users
Music:	41	10,25
Sports	44	11
Food	42	10,5
Transport	39	9,75
Other Depth 2	234	58,5

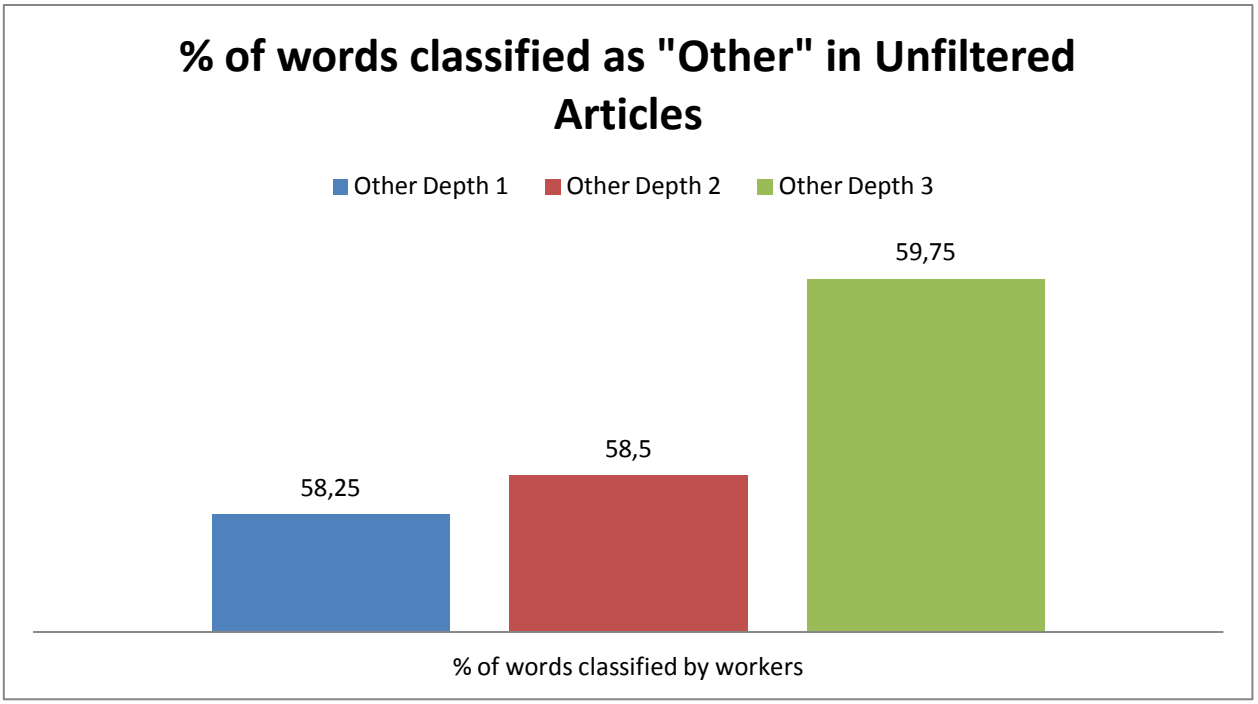


Graph 2 Classification of Words in Unfiltered Articles at Depth 2

Music:	48	12
Sports	43	10,75
Food	36	9
Transport	34	8,5
Other Depth 3	239	59,75



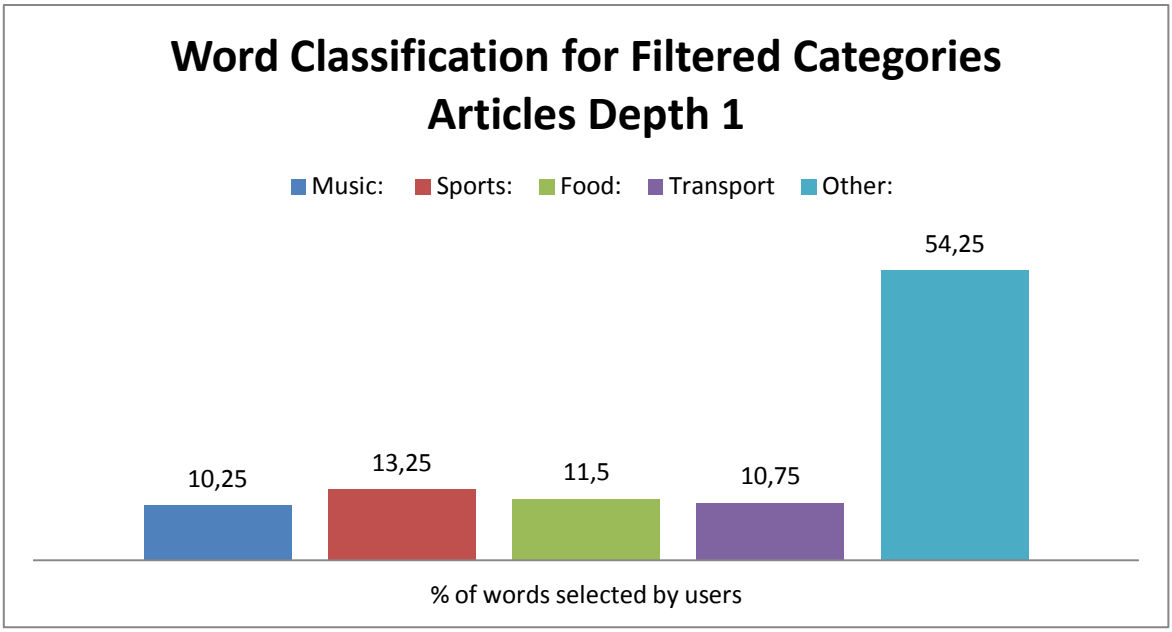
Graph 3 Classification of Words in Unfiltered Articles at Depth 3



Graph 4 Percentage of Words Classified as "Other" in Unfiltered Articles

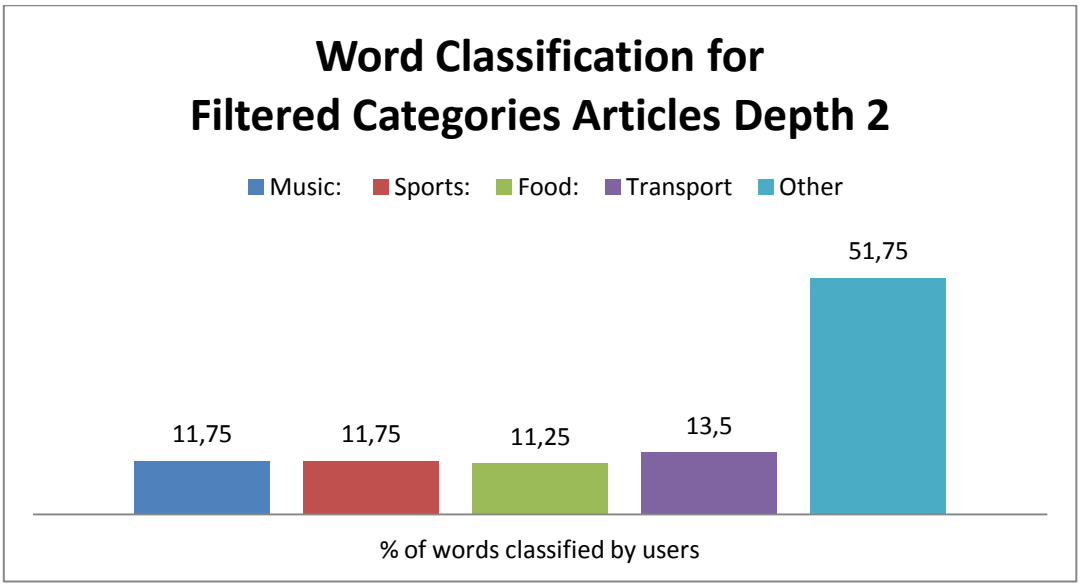
4.11.2Filtered Categories Articles Results

Depth 1	Raw count of words classified by users	% of words classified by workers
Music:	41	10,25
Sports:	53	13,25
Food:	46	11,5
Transport	43	10,75
Other Depth 1	217	54,25



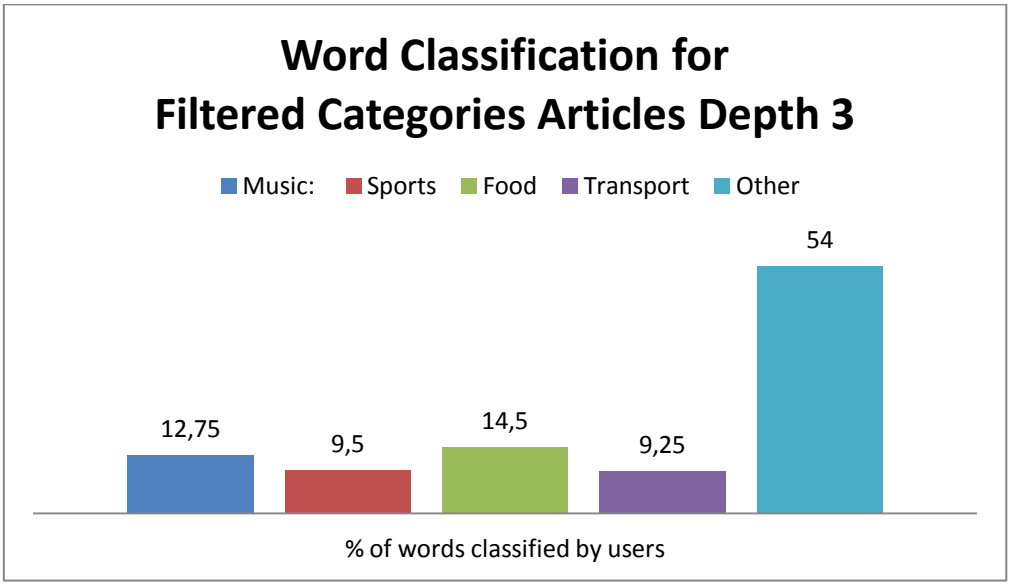
Graph 5 Classification of words in Filtered articles at Depth 1

Depth 2	Raw counts of words classified by users	% of words classified by Workers
Music:	47	11,75
Sports:	47	11,75
Food:	45	11,25
Transport	54	13,5
Other Depth 2	207	51,75

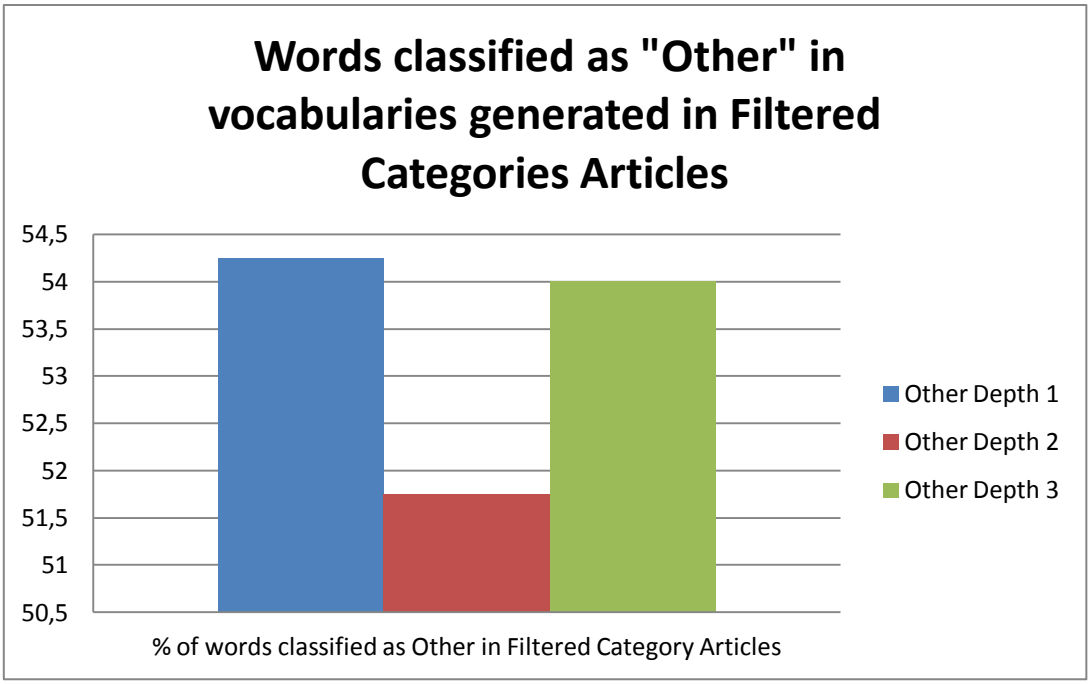


Graph 6 Classification of Words in Filtered Articles at Depth 2

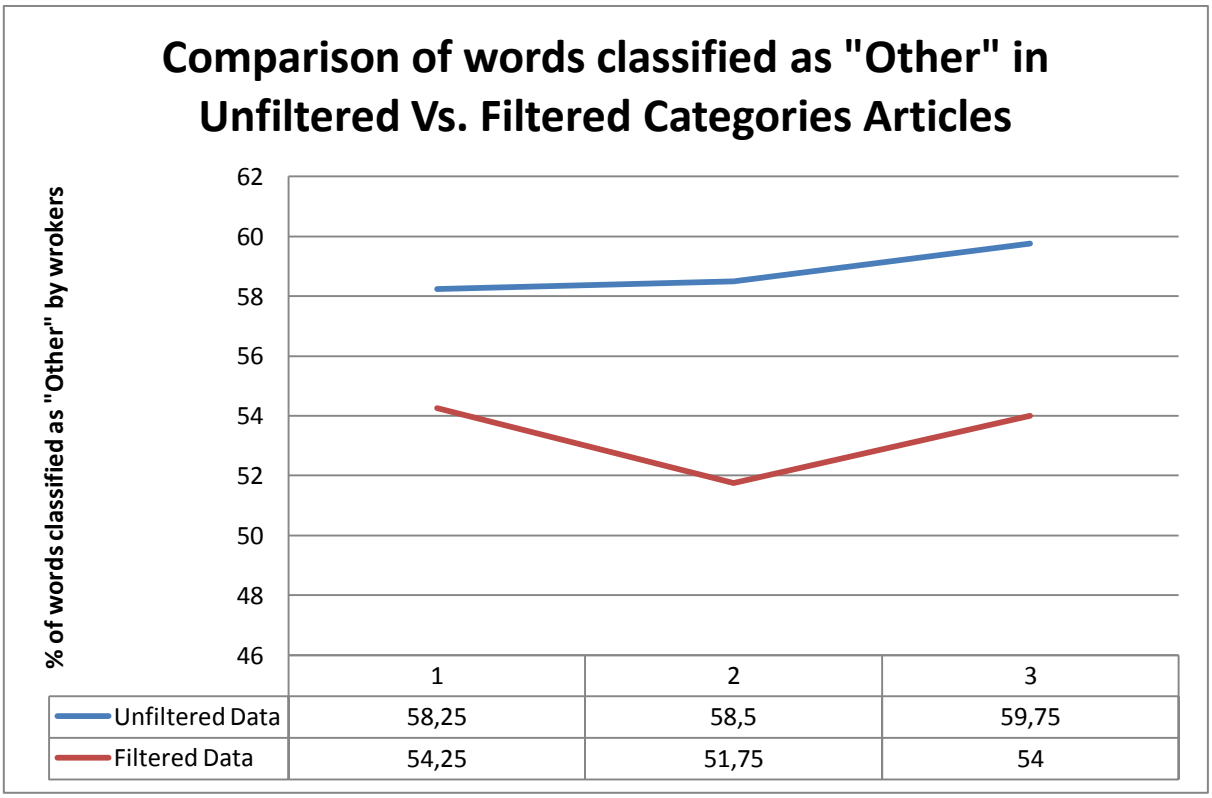
Depth 3	Raw count of words classified by users	% of words classified by workers
Music:	51	12,75
Sports	38	9,5
Food	58	14,5
Transport	37	9,25
Other Depth 3	216	54



Graph 7 Classification of Words in Filtered Articles at Depth 3



Graph 8 Percentage of Words Classified as "Other" in Articles From Filtered Categories



Graph 9 Comparison of Filtered and Unfiltered Articles

The evaluation found that the collection of articles which produced the best vocabulary was the one formed by Filtered Categories at depth 2. Filtered categories performed best overall. Unfiltered and filtered articles at depth 3 performed worse for the reason that the topics discussed in the articles steered away from the original

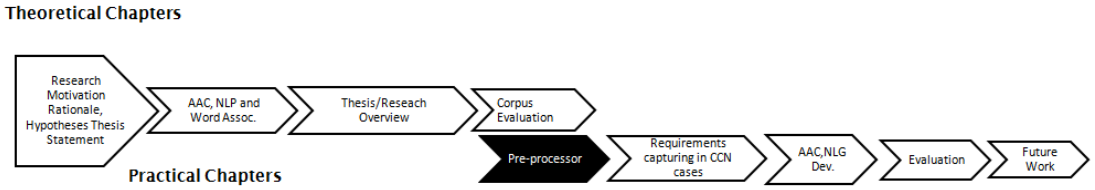
topic thereby increasing the amount of irrelevant words. Depth 2 of Filtered categories performed best because articles contained at that level discussed the original topics in more detail, increased the amount of relevant words and reduced the amount of irrelevant words. Depth 2 unfiltered also had the advantage of being balanced.

4.12 Summary

This Chapter examined the process used to choose and process the corpus used to generate the words used by the AAC application. It started with an examination of the most used corpora in NLP today. Justifications were given for why these were discarded in favour of Wikipedia. The Chapter concluded with a discussion on the results of an evaluation conducted via crowdsourcing to identify the best combination of Wikipedia Articles to use as a corpus.

4.13 Key Lessons

1. There exist a wide variety of Corpora to analyse and work with language
2. The needs of this work differ from those of Corpus Linguists as the results of processing have to be presented to users in conversational scenarios.
3. The current trend is to use the web as a corpus, however the web requires some processing to identify suitable pages. Reference texts are suitable because the topic of the document is easily identifiable,
4. Because the AAC application requires words about specific topics relevant web articles have to be collected. The best way to achieve this is to use reference web pages like Wikipedia
5. Wikipedia is best suited for this work not only because of point given above but also because it has been used successfully in previous work to build Semantic Networks and works well with the window based co-occurrence measures discussed in Chapter 2



Chapter V: NLP Pre-Processor

5.1 Introduction

Chapter V will discuss the practical implementation of the theory explained in Chapter II. The pre-processor is responsible for generating the lookup files used by the AAC application to provide the words required to form sentences

Chapter IV has shown how Wikipedia was most suited as a corpus for the intended goal of this work. Chapter IV discussed how different article collections were evaluated to find the best one. The corpus was chosen via a crowdsourced evaluation and was then used as input for the pre-processor that will be discussed here

This Chapter will start off the basic steps NLP steps that were implemented as part of the pre-processor. This will lead to a discussion on the implementation of PMIsig used to generate the semantic map used by the AAC application to enable the user to navigate and select related words.

Chapter V will conclude with a discussion on the use of WordNet, the British National Corpus and TFIDF for filtering and vocabulary ranking tasks.

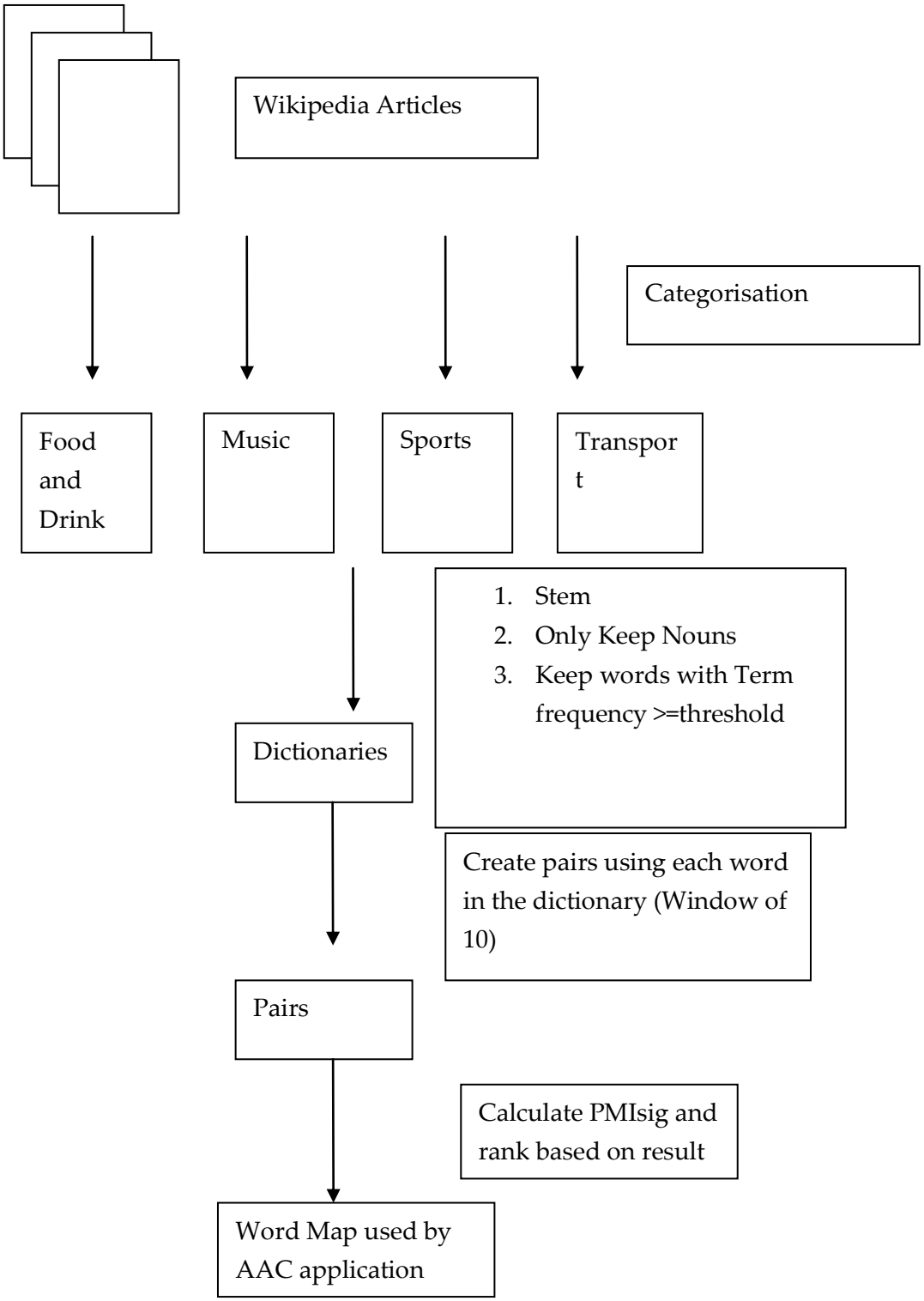


Diagram 3 The pre-processor pipeline

5.2 NLP Pre-Processor

This Chapter will discuss the steps of the pre-processor in detail. The main goal of the pre-processor is to generate the files which constitute the input (i.e. nouns and verbs) available to the user to form sentences in Dictum. With the exception of some steps (which can be carried out separately) the pre-processor works by a series of sequential steps.

The steps involved in the pre-processor are the following:

1) Stemming the Wikipedia articles
2) Tokenising and removing all POS except nouns from the articles
3) Counting word frequencies of the input and generating the vocabularies
4) Counting word co-occurrence
5) Calculating Word Association Measures
6) Generating the Word Map/Semantic Network used by Dictum

The pre-processor also includes the following functionalities which are not part of the main pipeline but do require the counting part of the pipeline for them to work:

- Sort the vocabulary files based on TFIDF
- Counting the noun-verb pair frequency
- Counting the noun-adjective frequency
- Merge the generated Semantic Network map with Word Association Norms files.

At the end of the pipeline the pre-processor generates the following files

- Four folder level vocabularies (A vocabulary for each topic)
- A master vocabulary (a file which merges the four vocabularies)
- The four vocabularies sorted by TFIDF (used by the application)
- The Semantic Network (a map which links each word in the main vocabulary by relatedness)
- A verb-noun network (a file which links each noun in the vocabulary with its most commonly associated verbs)
- An adjective-noun network (**same as previous but with adjectives – unused in the pre - processor**)

Pre-processor algorithms were designed to work both with text and XML files in the format of the British National Corpus (BNC) as it was decided to use both Wikipedia and BNC. Wikepdia as the main corpus and BNC as secondary corpus. The BNC is stored in XML files The XML parsing functionality implemented in the pre-processor was used to filter words based on their Part of Speech and for stemming.

5.2.1 Input

The input to the pre-processor are Wikipedia articles organised in folders named after the four topics: Food, Music, Sports and Transport. Before the articles can be utilised stemming is required. The next paragraph will discuss how stemming was implemented and how by exploiting tagged data contained in the British National Corpus a high standard result was achieved when compared to the classic Snowball stemmer.

5.2.2 Tokenisation, Stemming and Part of Speech Filtering

The first step of the pre-processor is tokenization (finding words in text). Tokenisation was implemented using the standard regex (Regular Expression) for finding words. "**[A-Za-z]+'?[a-z]**" . Note that the expression also caters for contractions and possessives. Words also need to be lowercased to avoid duplicates.

The first step of any NLP system is stemming, sometimes referred to as lemmatisation. As explained in the theoretical Chapter stemming is the process of reverting all words to their root form. This means that all words become grouped into their respective word class.

The de facto standard algorithm used for stemming is the porter algorithm (or its variant snowball which produces very similar results), the porter stemmer works by a set of hardcoded rules that sometimes do not produce grammatically correct words as the algorithm does not take into account exceptions. The example that is always made to show the limitations of the porter algorithm is that of the word “ponies”. To stem this word the porter algorithm uses the rule of removing the suffix which makes the word plural (i.e. es) thus producing the result poni, which is not grammatically correct. This result is acceptable in an informational retrieval scenarios where stemming is used for indexing purposes and words don’t have to be grammatically correct.

However, the non-words that it produces were unsuitable for the purposes of this research, as the result of the stemming process, is directly presented to the user by the AAC application.

As mentioned in the previous Chapter, as an alternative to the Porter stemmer the British National Corpus was used as stemmer. This was possible because the BNC contains tagged data for each word in the corpus. Information includes Part of Speech, syntactic role and lemma.

Using lemma data a table was compiled containing each word in the corpus alongside its respective lemma. The stemming method worked by looking up the table and returning the lemma. If the word was not present in the table, the method returned the original word.

anarch is polit philosophi that advoc stateless societi often defin as self govern voluntari institut anarch hold the state to be undesir unnecessari or harm slevin carl anarch the concis oxford dictionari of polit ed iain mclean and alistair mcmillan oxford univers press while anti statism is central some argu as subtl and anti dogmat philosophi anarch draw on mani current of thought and strategi anarch doe not offer fix bodi of doctrin from singl particular world view instead flux and flow as philosophi anarch as mass social movement has regular endur fluctuat in popular the termanarch is compound word compos from the word anarchi and the suffix ism the first known use of this word was in anarchi at merriam

Text Snippet 1 Sample Text Stemmed Using the Snowball Algorithm

anarchism is political philosophy that advocate stateless society often defined as help govern voluntary institution anarchism hold the state to be undesirable unnecessary or harmful slevin carl anarchism the concise oxford dictionary of politics ed iain mclean and alistair mcmillan oxford university press while government statism is central some argue as subtle and government dogmatic philosophy anarchism draw on many current of think and strategy anarchism does not offer fix body of doctrine from single particular world view instead flux and flow as philosophy anarchism as mass social movement has regularly endure fluctuation in popularity the term anarchism is compound word compose from the word anarchy and the suffix ism the first know use of this word was in anarchy at merriam

Text Snippet 2 Sample Text Stemmed Using the Tagged Data of the British National Corpus

The two text snippets above the difference between the use of the snowball stemmer and the use of the BNC tags for stemming. It is clear that using tags contained in the BNC produced better results, by reducing the amount of non-words to 0 and leaving the names of people and places unmodified.

BNC was also used to generate a table of all nouns contained in it. This table was compared against Wikipedia articles to eliminate all words which cannot assume in any case the role of nouns therefore keeping all nouns.

The POS filtering method generates a new file for each Wiki article containing lemmatised nouns. These files are placed in a new folder for each category named "Stemming …" where … stands for the respective category.

5.2.3 Vocabularies Generation

The next step in the pre-processing is to count term frequency of words. Term frequencies are placed into vocabulary files. The pre-processor produces one vocabulary file for each category plus a global vocabulary which is ranked based on corpus level frequency (the frequency at which word appears over the all collection).

Each line of the vocabulary file contains the word and the corpus level frequency or the folder\topic level frequency. The reason for having two types of vocabularies is that the corpus level frequency vocabulary is used to calculate PMIsig, and to perform topic level TFIDF to rank words based on their relatedness with their respective topic, this will be explained in a dedicated paragraph. The topic vocabularies are used to count the words in the articles for each topic and to delete those which are below a certain threshold, the remaining words above the threshold are used to generate the corpus level vocabulary.

```
food 5844
can 1492
may 1210
product 1166
make 1158
animal 916
new 754
year 743
taste 729
produce 718
time 714
process 704
people 699
world 688
united 687
```

Figure 22 folder level vocabulary

5.2.4 Co-Occurrence Counting

After the vocabularies have been generated the next step is to produce the target pairs used to calculate how much the words in these pairs are related. In the theoretical Chapter it was discussed how co-occurrence based similarity measures were favoured over distance based ones and how among these, PMIsig was favoured over other Mutual Information based relatedness measures. Co-occurrence based measures are also referred to as Window Based Measures as they require a window in which to identify co-occurring terms. Chapter II showed that PMIsig performs best when the window size is set at 10. In practice setting the window size at 10 means that for a given target word, co-occurring words are located 10 words or less (if the word is among the first ten of a document) before the target word and 10 words or less (if the word is among the last ten of a document) after the target word. Each pair's frequency is then counted and the pairs which occur below a certain threshold (set at the same value as the vocabulary threshold) are eliminated from the list.

```
states,united 660
united,states 660
york,new 308
new,york 308
american,actor 306
actor,american 306
application,ser 277
ser,application 277
player,american 265
american,player 265
singer,american 254
american,singer 254
continuation,application 231
```

Figure 23 Co-Occurence Frequencies Generated by the Pre-Processor

5.2.5 PMIsig Calculation and Semantic Map Generation

When all co-occurrences have been calculated the pre-processor calculates PMIsig for all pairs. The formula for PMIsig has been explained in Chapter II. For each first word in the pair, the algorithm creates a new line in the Word Map file containing the word itself and a list containing all words related to that word. Related words in the list are ranked from the most related to the least related where relatedness is

measured as the value returned by the PMIsig calculation. When a new word is found a new line in the file is created. Thus the file format generated is the following.

Word->relatedword1……..relatedwordn

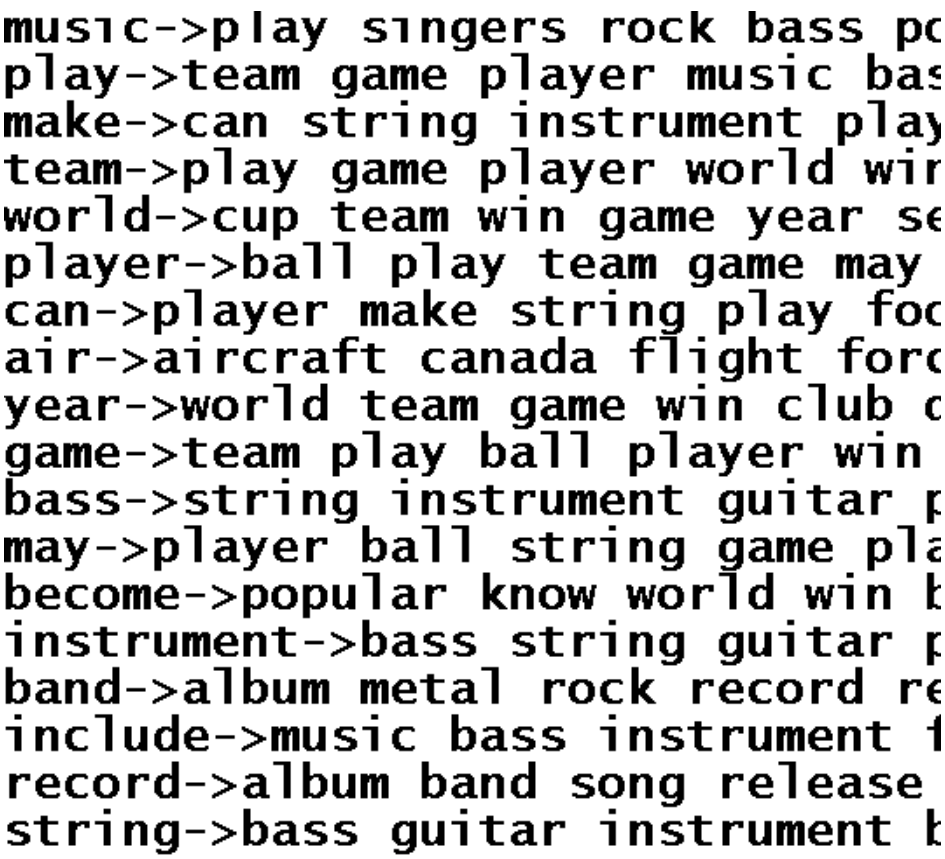


Figure 24 The Network of Related Words Generated by the Pre-Processor

5.2.6 Verb Map

To be usable the AAC application also requires a way for the user to insert verbs, as all POS but nouns were removed during pre-processing. For this purpose the pre-processor generates a separate Word Map that contains all words in the corpus level dictionary coupled with the verbs that are most likely used with each noun. The list of most likely used verbs is compiled once again by exploiting tagged data contained in the BNC. To find a relevant verb the algorithm scans the XML BNC files to find all occurrences of the target noun, once it locates the noun it scans the sentence of which the noun is part of from right to left until it finds a verb. All the verb-noun occurrences are counted. After the counting is finished a Word Map is created in the same fashion as the semantic Word Map.

The result of this algorithm is the following

Noun-> verb1……verb1

```
drink,coffee 95
present,problem 93
make,coffee 86
buy,food 86
make,history 86
show,evidence 85
provide,food 84
eat,meat 83
write,song 78
produce,evidence 7
save,energy 76
say,britain 76
get,problem 75
see,woman 75
drink.alcohol 69
```

Figure 25 Noun-Verb Co-Occurences Generated by the Pre-Processor

5.3 Accessing and Using WordNet in NLP Software

The purpose and structure of WordNet was explained in Chapter II. The purpose of the discussion that follows is to explore how to access WordNet data programmatically. There are two ways to access the WordNet data:

The first way is to download an offline copy of WordNet and using freely available APIs to access the data.

The second way is to download WordNet data in SQL format and building appropriate infrastructure in the application to access the data.

The first alternative was initially considered since an open-source library was found that integrated with the pre-processor since it also was developed in C#. The library was developed by Troy Simpson[22] . Troy Simpson also developed an algorithm measuring semantic similarity between words. This algorithm was integrated in the pre-processor to further investigate the capabilities of the library.

[22] http://www.codeproject.com/Articles/11835/WordNet-based-semantic-similarity-measurement

Further investigation led to choosing the PERL library WordNet::Similarity (Pedersen et al., 2004). The library was chosen instead of Troy Simpson's as WordNet::Similarity contains a variety of WordNet based similarity measures that have been developed from the early nineties to 2003 all the results of peer reviewed publications. Developers of the library themselves were authors of the main algorithm of this library (Banerjee and Pedersen, 2003). The algorithm is an improvement on the classic Lesk similarity measurement for WordNet named after its inventor: Michael Lesk.

This choice in favour of WordNet::Similarity was also made because WordNet::Similarity offers greater flexibility and control. It lets the developer choose which similarity algorithm to employ and to manually check which word senses score the highest similarity. Using the PERL alternative also proved advantageous during the AAC development. WordNet::Similarity relies on the WordNet::QueryData library which lets developer query WordNet from PERL. In the Chapter dealing with the Natural Language Generation part of this work it will be explained how QueryData was used to expand the selection of available words when generating sentences by exploiting hypernyms and hyponyms. The algorithms implemented in WordNet::Similarity were explained in the previous Chapter when discussing Word Association Measures based on Syntactic association between words.

5.4 Ranking the Words in Topic Level Vocabularies

As users select a category in Dictum they have to be presented with the most representative words for that category. Thus words need to be properly ranked. Investigation into the problem found that there were three possible ways in which words could be ranked. The first is raw frequency (the frequency at which the word appears at topic-folder level), the second is TFIDF (at topic folder level), the third is WordNet ranking. The second and third ranking methods will now be explained.

TFIDF was mentioned in the previous Chapter when explaining Wikipedia naïve Bayes classification. TFIDF is an information retrieval measure that is typically used to find keywords in document to perform document indexing (Jurafsky and Martin, 2000). In NLP scenarios TFIDF is used to find eliminate stopwords and therefore determine which words in the document are the most relevant to the topic discussed in it. TFIDF works on the principle that if words are frequent throughout

the collection they are probably words that are common to the language and are not important for any topic discussed in a particular document. Usually when calculating TFIDF, term frequency is counted at document level and the calculation returns the key words for a specific document. However for the purpose of finding the most relevant words for each topic, instead of using term frequency at document level, term frequency at topic level was used, which had been calculated during the vocabulary generation.

5.4.1 Future work: Merging the Rankings

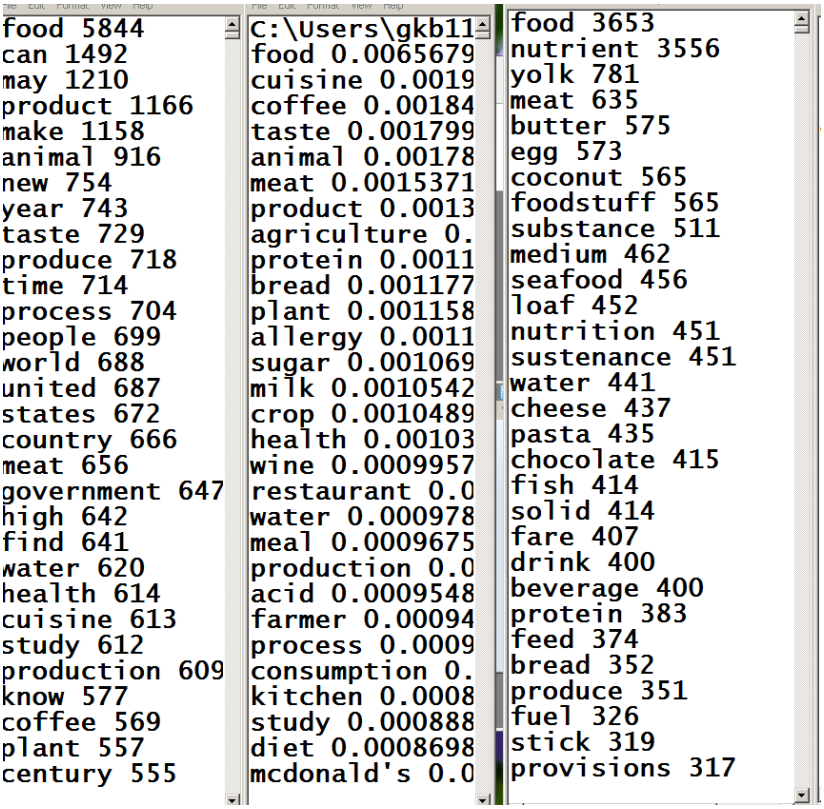


Figure 26 Different Ways of Ranking Vocabularies

The screenshot above shows the differences between the outputs of the rankings mechanisms discussed so far. Respectively from left to right the image shows, Ranking from raw frequency at folder level, ranking from TFIDF at folder level and finally ranking from adapted Lesk algorithm using WordNet

Further investigation is required into how to merge these rankings. Due to timescale constraints they were not investigated but only theorised.

Possible ways to merge the rankings could include:

- Averaging the different rankings and creating a merged list, this would require a suitable way to resolve conflicts among identical rankings.
- Finding a way to merge the ranking scales of each word and sort them by the resulting unit of measure
- Using crowdsourcing to rank words

5.5 Implemented Features Not Used in the Work

5.5.1 LSA

The pre-processor also contains methods to perform Latent Semantic Analysis. Latent Semantic Analysis was not investigated further in the work, because as explained in Chapter 3 it is more suited to finding paradigmatic word associations.

5.5.2 Word Association Norms Integration

Functionality was also included in the pre-processor to integrate Word Association Norms datasets into the Semantic Network generated by computing PMIsig. But because Word Association Norms contain a small amount of words (of which most are verbs) and would cover only a fraction of the words contained in the topic vocabularies, it was decided not to integrate WANs into the Semantic Network.

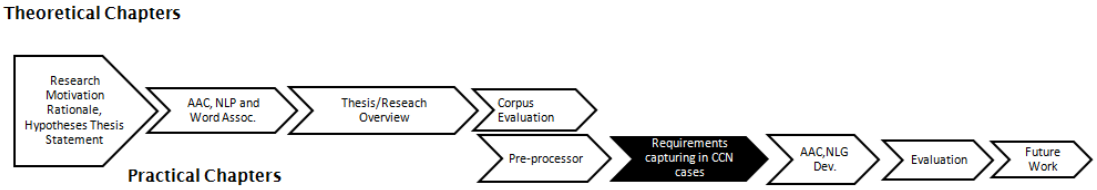
5.6 Recapitulation of Pipeline

1) Stemming the input
2) Tokenising and removing all POS except nouns from the input
3) Counting word frequencies of the input and generating the vocabularies
4) Counting word co-occurrence
5) Calculating Word Association Measures
6) Generating all files used by Dictum

5.7 Summary

This Chapter has discussed the implementation of the theoretical concepts discussed in the theoretical Chapter and the practical use of Wikipedia to generate the files required by the AAC application which is the core of this work. Each step of the pre-processor was explained in detail and justification was given for all steps of the algorithm. The Chapter also discussed the integration of the British National Corpus and WordNet alongside Wikipedia to perform the additional tasks of Part of Speech

filtering and vocabulary ranking based on relatedness with each of the four main topics of the vocabularies generated. Different ways of ranking the vocabularies were also discussed. The Chapter concluded with a short discussion on the features of the pre-processor that were implemented but were not used in the final work.



Chapter VI: Requirements Capturing in Cases of Complex Communication Needs

6.1 Introduction

Previous chapters have discussed the theory required to implement NLP/NLG techniques required to improve ease and speed of AAC for users with Complex Communication Needs. The rest of this thesis will discuss the implementation of Dictum: The AAC app developed as part of this work.

This Chapter will explore the requirements analysis technique that was used to capture requirements for Dictum. Since the aim of this work is not to develop a full system but to answer the question of whether applying NLP to AAC improves AAC, the requirements gathered in this Chapter were not used in the prototype developed. However valuable lessons were learned from the requirements analysis and the work discussed in this Chapter is a contribution to Human Computer Interaction in case where users cannot comfortably express their requirements.

After meetings with two aphasia help groups the conversational data gathered during the meeting was analysed by using a model called the Technology Appropriation Model (TApM).

Firstly this Chapter will discuss the challenges faced by researchers when working with persons with Aphasia, it will then discuss TApM in detail, which will lead to a description of the meetings where data was gathered. The Chapter will conclude with a discussion on the results of the application of the model and of the contribution to which the application of this model lead

6.2 Aphasia

Currently 250.000 people in the UK[1] and 1.000.000 people in the US[2] have aphasia. Aphasia is defined as a "communication disability caused by damage to the language centres of the brain. It can come as the result of a stroke, head injury, brain tumour or other neurological illness" Aphasia is erroneously thought as being only an impediment in speech but it brings a variety of other symptoms (Galliers et al., 2012) such as: difficulty with writing and reading and the inability to cope with numbers. People with aphasia can have additional difficulties such as physical impairments and vision problems.

6.3 Challenges of Working with Participants with Aphasia

The work by Galliers et al. (Galliers et al., 2012) explores challenges faced by researchers when working with participants with aphasia. The work offers a useful insight into the symptoms of aphasia

People with aphasia have difficulty in understanding[23] abstract concepts, for example some find it difficult to recognise an object through a drawing and find it much easier with photographs. Often the users find the use of an Avatar instead of live videos, distracting and uncomfortable, users also struggle to answer speculative questions, such as being asked to imagine situations where the person would have had access to the tool used as soon as they suffered the stroke.

Information retention and retrieval is another problem that those with aphasia face, word recall problems are universal in people with aphasia; they often try to compensate by making mimicking gestures. These recall problems sometimes translate to other kinds of information such as names of pets and members of family.

People with aphasia can also have difficulty with arithmetical skills and can lack an intuitive understanding of numbers. They also often have difficulties in coping with actions that have to be performed simultaneously, or as chains of linked actions. Finally, some people can be affected by attention deficits and have physical impairments in addition to other effects of their stroke.

---

23 https://www.floridahospital.com/aphasia/statistics

The other main type of aphasia apps is therapeutic apps. For example "Naming Therappy"[24] which presents users with images of objects on screen and the objective is for the user to recall what the object is called.

6.4 The Technology Appropriation Model

The Technology Appropriation Model was introduced by Carrol et al. in 2001 (Carroll et al., 2001). The abbreviation TApM will be used so it is not confused with TAM usually reserved for the Technology Acceptance Model. Salovaara (Salovaara, 2009) highlights that these models are primarily used in sociological contexts and little work has been done in other areas such as software/hardware accessibility. In an effort to bridge the gap this Chapter will apply TApM to an HCI context.

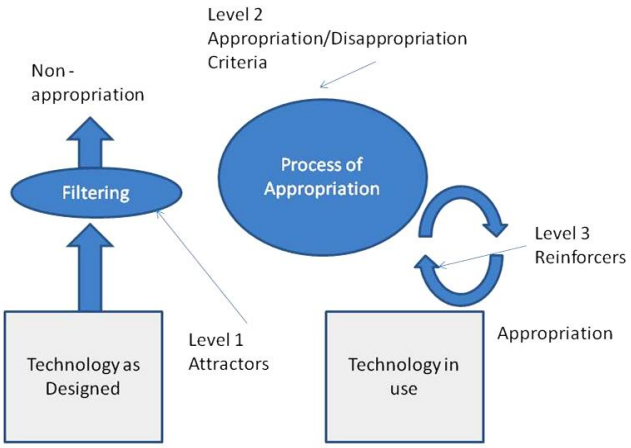


Figure 27 The Technology Appropriation Model, adapted from (Carroll et al., 2001)

As shown in Figure 1, the Technology Appropriation Model consists of three main stages:

- Technology-as-designed component
  - The technology before being appropriated by the user
- Process of Appropriation
  - The process of appropriating the technology to suit user needs.
- Technology-in-use
  - The user actually starts to use the technology adapting it to his needs.

Carroll and others later expanded the model in subsequent work (Carroll et al., 2007)

[24] Naming TherAppy.*Tactus Therapy Solutions*. http://tactustherapy.com/apps/naming/

- Level 1 – at this level the user is unfamiliar with the technology and has his first contact with it. For example seeing it in a shop or being introduced to it by friends or family.
- Filtering phase – during this phase the user decides if the technology interests him, albeit with a superficial or incorrect understanding of the technology.
- Level 2 - Appropriation process: during this phase the user acquires a deeper understanding of the technology through frequent use.
- Level 3 – long term use: this level refers to the use of a technology through time. The technology is properly appropriated by the user and is integrated into everyday life.

TApM has also the advantage that it focuses strongly on the use of technology from the perspective of the user. Works like those of McNaughton and Light (McNaughton and Light, 2013) advise that researchers working with assistive technology should never lose focus of the users the technology is trying to assist.

Usually when applying TApM user interviews focus on intended use of technology (Carroll et al., 2001)(Carroll et al., 2002). In contrast, the interviews conducted as part of this work were primarily focused on capturing requirements for the AAC application. The following sections discuss our methods and results from this process.

6.5 Methods

6.5.1 Participants

The data analysed in this work were collected during User Centred Design sessions with 10 (2 in Edinburgh and 8 in Dundee) users with aphasia conducted with the help of Aphasia Help Groups in Country. Two groups were contacted: one was Speakability Edinburgh and the other was the Dundee NHS Tayside research group at the University of Dundee. The Dundee group is focused on using iPads as an aid and focuses on getting together as a group and sharing knowledge of new apps downloaded or new functionality that was unknown to some members of the group. Both groups meet monthly. Participants were people of both sexes and from different age groups (ranging from late thirties to late seventies). None of the participants had physical impairments which precluded or reduced their ability to use their tablet or Smartphone. Participants had varying degrees of communication

difficulties, ranging from the very severe (only being able to speak short sentences and requiring time to search for the appropriate word) to those who had nearly regained their full ability to speak.

At the Dundee meeting three speech professionals were also present. One of the professionals was responsible for organising the meeting between us and the group. This professional also helped with introductions and gave me advice on how to interact with users who had severe communication difficulties.

6.5.2 Meetings

The meetings were held in a roundtable fashion with each participant asked to share his or her experience with smart devices. The length of discussions with each participant varied depending on his/her communicative abilities and the amount of experience he/she had with their device. Meetings with users in both cities lasted for 2 hours divided among the number of users present.

Firstly, users were asked to share their personal experiences with mobile devices and to discuss in which ways (if any) smart devices help them in their daily lives. It was during this phase of the discussion that Appropriation emerged. The second part of the conversation was focused on demonstrating the prototype of the Version 2 of the Augmented and Alternative Communication application.

Despite the popularity of applications such as Proloquo2Go (Sennott and Bowker, 2009)), such applications were unknown to all participants. Those who used mobile AAC preferred GridPlayer or others that are less known, but the majority of users did not use specific AAC apps and preferred using customised notes prepared either by themselves or by a member of their family or Speech Therapist . This indicates that Appropriation can be triggered by the user not knowing of the existence of apps that implement behaviours for which they created Appropriation methods.

6.5.3 Choosing TApM

The decision to apply TApM was taken retrospectively whilst analysing the notes taken during conversations with users. Our discussions with users were focused on requirements capturing for improving current AAC systems on mobile devices. Usually requirements capturing sessions are conducted by having the user focus on how he will use the device in the future, but in line with the findings of Galliers et

al. (Galliers et al., 2012), that aphasia affected people have trouble with abstract concepts, it happened that the conversation always steered towards current use. The only instance where users focused on the future was when they remembered aphasia causing them a negative experience and were afraid it could happen again. After the meetings, whilst analysing the notes taken during the conversations it was found that rather than having specific requirements, as usually happens in standard requirements capturing, data were composed of users discussions on how they adapted their apps and how they would have liked these adaptations to be transposed to the AAC app. This meant that in such cases (where the user cannot comfortably express his requirements) the standard requirements capturing process is ill-suited. From this it was understood that these "current experience based" data are exactly the kind of data used in models like TApM.

The next section will present the results of the analysis of the discussions in terms of the Technology Appropriation Model. The terms user and participant will be used interchangeably

6.6 Results

This section will present the results of this work by analysing the data collected by conversing with users by applying TApM in the following way: first the target technology that was Appropriated will be identified then Level 1 2 and 3 behaviour will be analysed The section will conclude with a list of requirements for an AAC application gathered as a result of this analysis.

6.6.1 Identifying the Technology

Before applying the model the target Technology must be identified. Our target technology will be portable smart devices (tablets and Smartphone) as all of our users had iPads and one user also used his Smartphone to communicate.

6.6.2 Analysis of Level 1 User Behaviour: First contact with Technology

In level 1 the user will come into contact with the technology for the first time and will decide whether to adopt the technology or no, a phase referred to as *filtering* within the model. Most filtering is due to the user's age and previous experience with technology If we look at the Edinburgh group, the participants owned iPads but did not use them frequently because "they couldn't make it work" so their

Appropriation was less marked than users from Dundee. In contrast Dundee participants were more open to the technology mainly because their group was specifically set up to explore these devices. If they struggled they could ask members of the group to help them. They also enjoyed the fact that they could show others what they had learnt since the last meeting. We did not directly ask our users to describe their first contact with the technology but for members of Dundee it emerged during conversation that the technology was given to them as a gift by members of their family. For most members of Dundee their first contact occurred when they joined the support group.

6.6.3 Analysis of Level 2 User Behaviour: The Start of the Appropriation Process

In level 2 of the model users start to customize their devices by installing additional apps and inventing additional uses for the apps already present on the device. To study this level the researcher must identify user motives for Appropriation to begin; these motives are called Appropriation Criteria. For example, Carrol in work (Carroll et al., 2001) uses word coding to identify Appropriation Criteria. In the instance discussed here word coding, cannot comfortably be used because data were conversational in nature. Instead questions phrased in the first person from the perspective of a person with aphasia were used. Analysing meeting notes four main questions that drive the appropriation filtering decisions (in addition to some less common issues) were identified:

A. Will it help me talk about myself with others?
B. Will it help me remember words?
C. Will it help me understand written documents?
D. Will it help me when I'm writing to friends and family?
E. Other

6.6.4 Analysis of Level 3 User Behaviour: Analysing Appropriation in Everyday Life

In level 3 TApM looks at how Appropriation caters for the needs of everyday life. In this section we look at six example appropriations for commonly used mobile apps that were identified in this analysis

6.6.4.1 The use of notes

All of the participants made use of notes either as a reminder or to store snippets of text as an aid when composing text within other applications such as Instant Messaging or e-Mails. Notes were also used as a form of storytelling to tell other people important life experiences or personal information. All of the participants had special notes reserved for presentational information (e.g. name, birthplace, etc.).

6.6.4.2 Combination of Text to Speech and Speech to Text

In some cases participants invented combinations of functionality to obtain what they wanted. Participants who had difficulty with reading physical documents, transferred the document to their device using the OCR application installed on the phone so that they could hear the text via Text To Speech.

6.6.4.3 Picture based communication

Participants who had difficulty with writing devised a shortcut by saving images of favourite topics to their device and then pasting them inside correspondence to make the recipient understand what they meant or to augment their poor descriptive skills when writing.

Another form of picture based communication was the use of slideshows with pictures and text to tell a story. To be set up, this sometimes requires the help of a communication partner or of a Speech Therapist depending on the abilities of the participant. In one case the slideshow had been prepared by the participant's wife and contained his life story with particular emphasis on how life changed after the stroke.

A third form of picture based communication was for the participant to show photos without text to communicate. This is a common form of communication as it does not require the help of a communication partner. It has been explored in several works such as the app PhotoTalk (Allen et al., 2007). PhotoTalk supports communication by providing a platform for participants to independently capture personally meaningful images and to then share them with their communication partners. The work of Levin et al. (Levin et al., 2007) also looks at picture based communication.

6.6.4.4 Dictation

Participants who had regained a good level of speech but still had some problems with remembering word spelling resorted to dictation to compose emails or to store difficult words into notes for later writing.

6.6.4.5 Supported Independence

One participant and his wife adapted iTunes to their needs. The participant is a jogging enthusiast but since the stroke that caused his aphasia his wife is worried when he is out jogging. The participant's wife uses the "Find my iPhone" [25]app to periodically check where her husband is while he is jogging thus giving them both peace of mind.

6.6.4.6 A special use of YouTube

One participant could not understand nor remember lyrics to his favourite songs after his stroke. He had problems understanding the lyrics to songs as he could not cope with both the music and the lyrics at the same time. He used the YouTube application as a sort of training mechanism to re-learn his favourite songs.
We can conclude from this that, in some cases Appropriation is triggered by the user not knowing what is available.

6.6.4.7 Capturing AAC Requirements Using Level 2-3 Behaviour

Table 3. User Appropriation Behaviour Coupled with Proposed Questions (Section 4.4)

	Notes	Combi	Pict. B. Com.	Dict.	iTunes	Youtube
QA	■		■	■		
QB						■
QC		■				
QD	■		■	■		
QE					■	■

Table 1 couples questions proposed in level 2 analysis with Appropriation behaviours listed in level 3 analysis. It will be used as the basis for a discussion that

[25]Find My iPhone. *App Store*. https://itunes.apple.com/us/app/find-my-iPhone/id376101648?mt=8

will show how Appropriation behaviour was applied to capture requirements for the AAC application. These apps are used by those who have difficulties communicating as a way to integrate or replace their communicative abilities. During the meetings participants asked questions like "Can I use my notes?", "Can I dictate to it?" In general users wanted to find all of the functions they had created via Appropriation collected into the AAC app. Therefore by looking at level 3 behaviour we have requirements for the AAC application

These requirements were inferred as follows:

The first step to infer requirements is to categorise users in their appropriating behaviour.

First we have to exclude the special cases where users appropriated iTunes and YouTube as they are unrelated to communication and therefore cannot be used to gather requirements for the AAC application.

We are thus left with:

1. Users who Appropriated Notes
2. Users who Appropriated Dictation
3. Users who Appropriated Text-To Speech and Speech to Text
4. Users who Appropriated Picture Based Communication

By looking at each of these four behaviours we can gather as many requirements.

From users in group 1: Use text contained in user notes to produce sentences for users to introduce themselves.

*From users in group 2:*Use text contained in user notes to collect sentences for different everyday situations.

From users in group 3:Include dictation for cases where user wants to store a word in the app and remembers how to pronounce it but not how it's spelt.

From users in group 4:Include pictures to integrate description of places, pets, and members of family.

In line with the findings of McNaughton and Light (McNaughton and Light, 2013) it was found that users are willing to use their devices to communicate in

public (as "everyone has a smart device") and would like their devices to offer them the right choice of words and sentences based on current context; for example helping them communicate when they are in the bank or in the supermarket.

6.7 Discussion

By examining Appropriation of software by users with aphasia the following contributions were made:

The foundations were built to fill the gap found by Salovaara (Salovaara, 2009) (TApM is mostly used in sociological studies), by showing that it can be used effectively in HCI/Software Engineering contexts.

By examining Appropriation of a particular piece of software or hardware by a special needs community it was showed that the requirements capturing process can be facilitated in cases where the target user has difficulty expressing his needs because of inherent difficulties that come with his condition, especially the difficulty of imagining abstract situations such as imagining himself using the application in the future. The use of TApM counters this by letting developers discuss potential requirements with users by focusing only on current use.

Current works in Appropriation focus on adapting design to make software or technology more appropriable by users; for example the work by Dix (Dix, 2007) where he proposes guidelines for software designers so that software lends itself to Appropriation. Dix uses two use-cases to prove the validity of his guidelines. It is my personal opinion that the reverse is also true. Appropriation should be examined to improve subsequent versions of software.

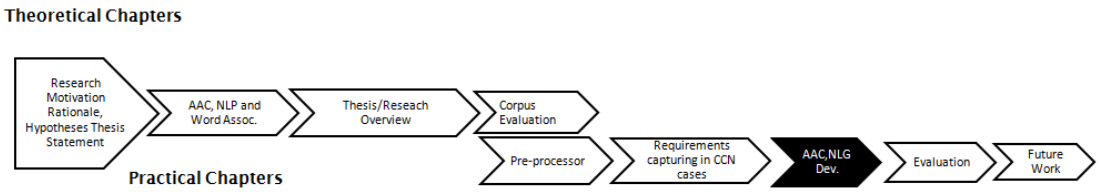
A limitation of this study was that TApM was applied retrospectively. In future it would be interesting to apply the model while conducting the discussions with users by structuring the conversation towards Appropriation and comparing the results to the results of this work. Another limitation was the small sample size of the population. However this was due to the difficulty of finding not only people with Aphasia, but people with Aphasia who used smart devices.

6.8 Summary

This Chapter looked at Appropriation of smart devices by users with aphasia. It first presented aphasia and how its symptoms affect users. It then presented the Technology Appropriation Model and analysed discussions with users with aphasia based on this model. It was showed that users with aphasia adapt smart devices and apps for a wide variety of uses; this is either caused by a real lack of functionality or by the user not knowing that apps are available. Data were analysed using the Model and presented user requirements gathered using the Model. It was found that that looking at Appropriation of smart devices and software in general by the special needs population can serve as a powerful tool for software design and maintenance. From this work it can also be concluded that whilst current research in Appropriation is focused on making software more appropriable, research should also focus on looking at Appropriation for better software design.

6.9 Key Lessons

1. It is difficult to capture requirements in cases of users with Complex Communication Needs
2. Appropriation is the way in which users adapt their device or the functionalities and apps of their device to suit their needs.
3. Appropriation is often the result of users not knowing what applications or functionalities are already available to them
4. Observing Appropriation behaviour of users with Complex Communication Needs can be an effective tool for capturing requirements.



Chapter VII: Dictum AAC Application Design and Development

7.1 Introduction

This Chapter will examine the design and development of Dictum: the AAC application central to this work. It will start off with the guidelines given in the work of others to design effective AAC systems. Chapter VII will then examine and justify the design of Dictum. The implementation of the application will be examined by exploring the various versions of the application leading to the final version which makes full use of the NLP techniques discussed so far and which exploits NLG techniques discussed in Chapter VIII.

7.1.1 Discussion on Design Decisions Based on the Works of Others

The problem of designing effective AAC is very important; bad design of AAC systems can lead to systems which are useless for users. To date very little work looks at how to design effective aided AAC (Light et al., 2008). The work just cited attempts to fill this gap by giving guidelines to design effective aided AAC. According to Light and her colleagues there are many variables which impact the effectiveness of aided AAC; among these the main ones are :

- Variables related to the user
- Variables related to tasks
- Variables related to the AAC system

All of these variables affect performance and in many cases prevent the user from communicating with ease and fluidity.

With respect to variables relating to the physical and cognitive condition of users, it was decided that the Dictum app should cater for users who have no motor difficulties, no visual impairments (or impairments overcome by the use of glasses), are regaining their ability to speak and have sufficient writing or reading skills which enable them to effectively use Dictum. This decision was taken for two reasons: The first reason is the research aim of this thesis is to investigate on the impact of the introduction of NLP/NLG on AAC systems primarily from the perspective of performance. The medical impact of the Dictum system should be investigated in future work. The second reason is that these target users were the majority encountered during meetings with potential users, so focusing on this user group is not unrealistic.

Variables related to communication tasks concern aspects such as the complexity of tasks which users have to perform in order to communicate using aided AAC. Light et al. give particular emphasis on the attention required of the user. In practice according to Light and her colleagues tasks in aided AAC systems should be designed such that they should not require excessive attention distracting the user from the conversation and the interlocutor.

The design of Dictum caters for communication tasks complexity by delegating the responsibility of complex sentence formation to the application and by reducing search space of words. The goal of this design is to increase fluidity and require as little concentration as possible on the part of the user.

The last group of variables which affect performance are those related to the AAC application itself. For Light the aspects to consider when dealing with application related variables are: the size of the display, the selection technique and the final output.

The work by Brandenburg et al. (Brandenburg et al., 2013) also gives guidelines for the effective design of AAC applications. Guidelines given by Brandenburg and colleagues are the following:

1) Text should be supplemented with pictures, symbols or spoken word.
2) Large text, simple font, lightly coloured background and lightly coloured text should be used.
3) Size of buttons should be maximised.
4) Tab navigation is recommended, and the navigation hierarchy should be kept at most at a three tier depth.
5) Apps should be overall simple and disturbing backgrounds should be kept to a minimum.

Bradshaw (Bradshaw, 2013) identifies factors which make the use of AAC effective for users:

- Ease of use
- Reliability
- Availability of technical support
- Voice and language of the device
- Involvement of user in choice of AAC
- Time generating message

Higginbotham and colleagues in (Higginbotham et al., 2007) look at access to AAC for people with Complex Communication Needs. They state that for AAC to be really accessible users should be able to: use and benefit from the device, converse with someone, obtain or retrieve information from the communication device and be able to help others who use the device.

The proposed design of Dictum meets all of these requirements but one. Dictum is intended for fast sentence formation but the user also obtains information from it. Higginbotham and the others include this in the requirements because their work looks at a wide variety of users and devices.

As Light et al. stress, one factor impacting access to AAC are users themselves. Higginbotham et al. state that to be really accessible the AAC device should fit

around the user's physical condition. It has been explained at the start of this thesis that users with CCN include a wide variety of people with differing needs. For the purposes of this work, as just mentioned, the design of Dictum focuses on users with no physical impairments, no visual impairments and whose CCN do not prevent them from using a grid based AAC device/app. This type of target user is realistic and is based upon potential users met during this work.

Light et al. also give a taxonomy of the various components of an AAC system. These will now be listed and for each one it will be explained how each component of the taxonomy was designed in Dictum.

Concepts: the vocabulary of the system.

In Dictum the vocabulary is represented by the most recurring words in the Wikipedia pages selected as corpus.

Representation: the symbols used to represent the vocabulary.

As is standard with electronic AAC the vocabulary is represented by a grid of words with pictorial cues.

Organisation: how the vocabulary is organised in the system.

The organisation of the vocabulary files used by Dictum was explained in Chapter V when discussing the output of the pre-processor.

Presentation: how the words are presented to the user.

Dictum presents the data in a grid as is standard with Grid Based AAC systems (as discussed in Chapter II).

The remainder of this chapter will discuss the various iterations of Dictum and how feedback from potential users helped each iteration.

7.2 Dictum Development

Dictum is a tablet app developed for Android OS. Even though the majority of AAC apps is developed for the iOS environment (Grigis and Lazzari, 2013), it was chosen to develop for Android due to the greater flexibility of this system, the absence of barriers to upload and test applications on physical devices and greater experience of the author developing in Android. The same motivation for choosing the Android development environment was given in the work of Grigis and Lazzari (Grigis and Lazzari, 2013). The emergence of Android OS for the development of

AAC applications is also noted by Higginbotham and Jacobs (Higginbotham and Jacobs, 2011).

As is standard with Android applications, Dictum was developed in Java. In total 3 versions of the application were developed in the course of this work. The first 2 versions were developed with the purpose of showing a semi-working prototype to users to improve on interface and to give users an idea of what the application would look like in the end.

7.2.1 Version 1

The first version of the Dictum prototype, shown below, used a simple interface with buttons and no pictorial cues. The main menu of the app showed the four categories of conversation from the Wikipedia articles used by the pre-processor

- Food
- Music
- Transport
- Sports

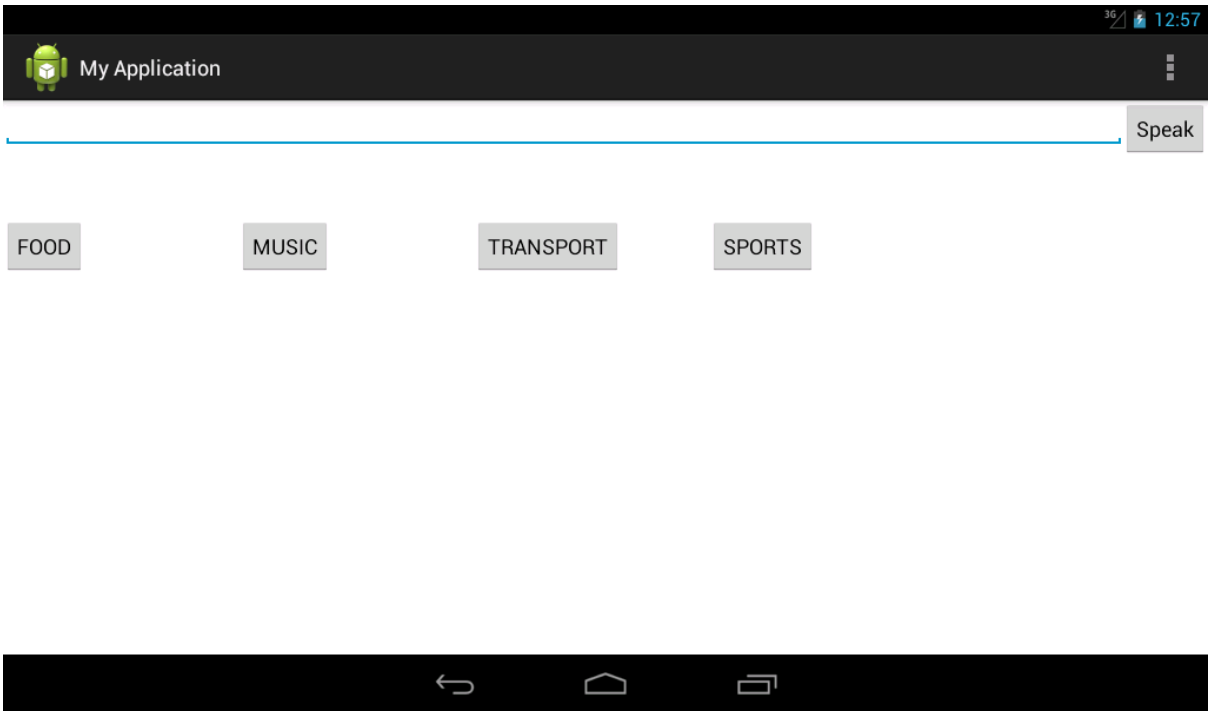


Figure 28 Main Screen of AAC Application

The choice of these four categories was explained in Chapter IV (Choosing the Corpus) and Chapter V (NLP Pre-Processor Development).

Once a category is chosen by the user, the application lists words for that category. Figure 29 shows words for the Transport category. When a category is first selected the semantic web map still does not come into play thus, words are ranked based on the ranking given to them by the pre-processor algorithm.

Dictum is designed as a state machine. The state machine states are the four categories plus a dynamic state. The dynamic state means that a word has been chosen so the application must dynamically change the choice of available words based on the Semantic Network. Thus, as each word is selected, the available selection of words changes based on semantic relatedness. Words are ranked based on semantic relatedness with last inserted word. It was explained in the pre-processor chapter that the structure of the Semantic Network is as follows: "word -> [relatedword 1..related word n]". In practice this means that when the application goes into dynamic state the text file containing the Semantic Network is read and the related words for the selected words are loaded into memory and presented to the user.

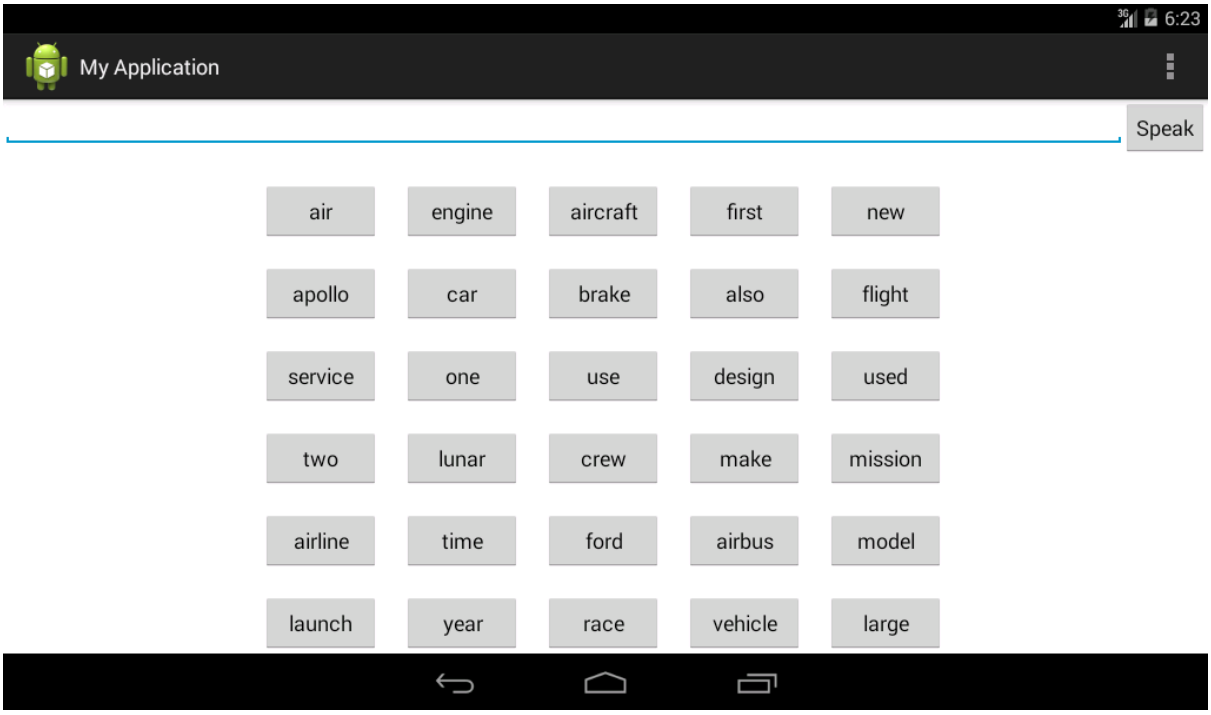


Figure 29 Words for the Transport Category

7.2.2 Meeting with Potential Users and Speech Professional at the British Aphasiology Society Research Update Meeting 2014

After version 2 was completed, a meeting was attended with potential users held by the British Aphasiology Society (BAS) called the Research Update Meeting (RUM) at the University of Newcastle. The Research Update Meeting is held every year by the BAS to explore the latest research in Speech and Language Therapy to improve the quality of life of those with aphasia. The RUMs have a different theme every year. The theme of 2014 was "Digital Technology and Aphasia".

A short presentation about this research was given during which version 1 of the app was presented to the audience to get some initial feedback on Dictum, on the avenues that the research should explore and future directions in general.

The audience was positive about the idea in general and confirmed that to their knowledge the idea was novel and that whilst some current applications do employ some degree of text prediction no current app uses the amount of Natural Language Processing proposed for this work .

In line with (Brandenburg et al., 2013) they suggested that buttons be bigger, contrast higher and images contain pictorial cues about the word they contain to help the user.

7.2.3 Version 2

Version 2 of Dictum (shown in Figure 3) featured UI improvements and new functionality. The state machine at the core of the app did not change. UI improvements included higher contrast achieved by changing background and button colour. The button colour was set to dark grey and text in buttons was set to white. The background colour was set to orange. The main changes regarded the introduction of new functionality to make it actually possible for the user to form sentences.

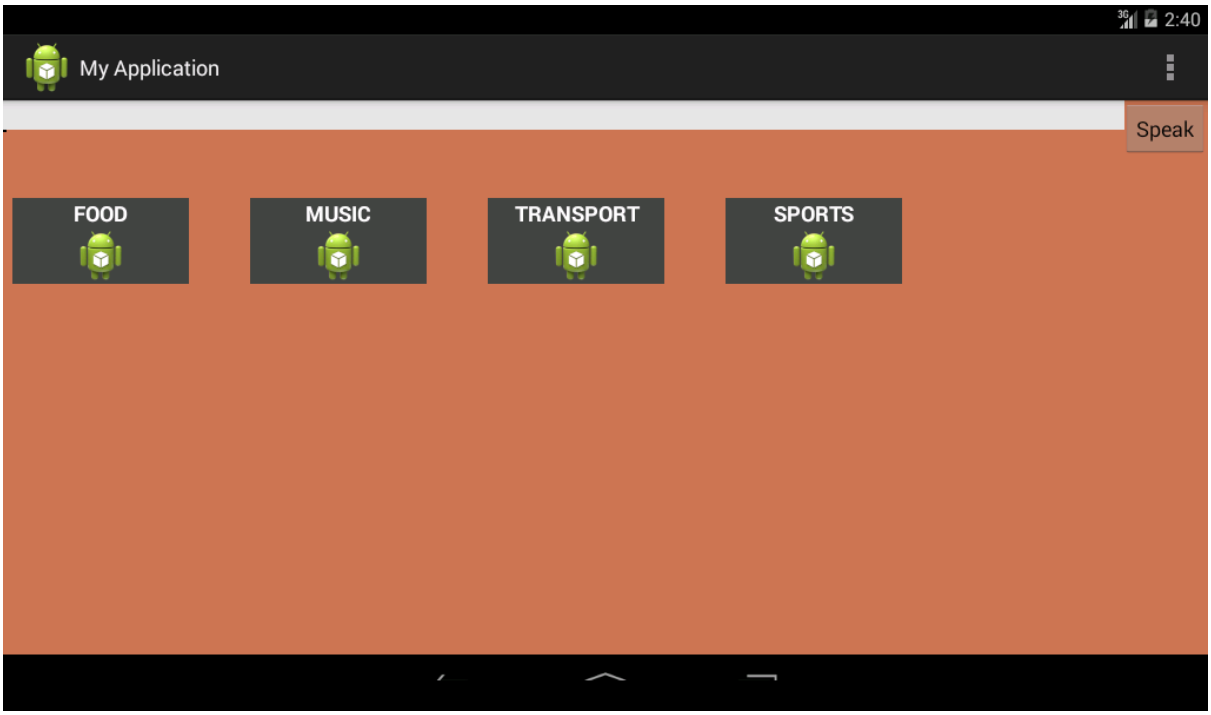


Figure 30 The Four Categories with Updated UI

During the pre-processing phase all words are lemmatised. But for the user to form meaningful sentences suffixes for nouns are required. A first attempt at reintroducing suffixes on nouns was made in version 2 by hard coding them into the application. Shown below, suffixes were inserted by user by tapping on a word inside the edit text and selecting a suffix from the menu shown in the figure below.

Figure 31 First Attempt at Suffix Insertion

All stopwords and all Parts of Speech (POS), except for nouns, were removed during the pre-processing phase. Removing stopwords is the first step in Corpus filtering when calculating PMI. However, stopwords, like suffixes, are required to form meaningful sentences. As a first attempt, stopwords were reintroduced by adding a listview on the right hand side containing alphabetically sorted stopwords. This version also used a Word Map that contained both verbs and nouns.

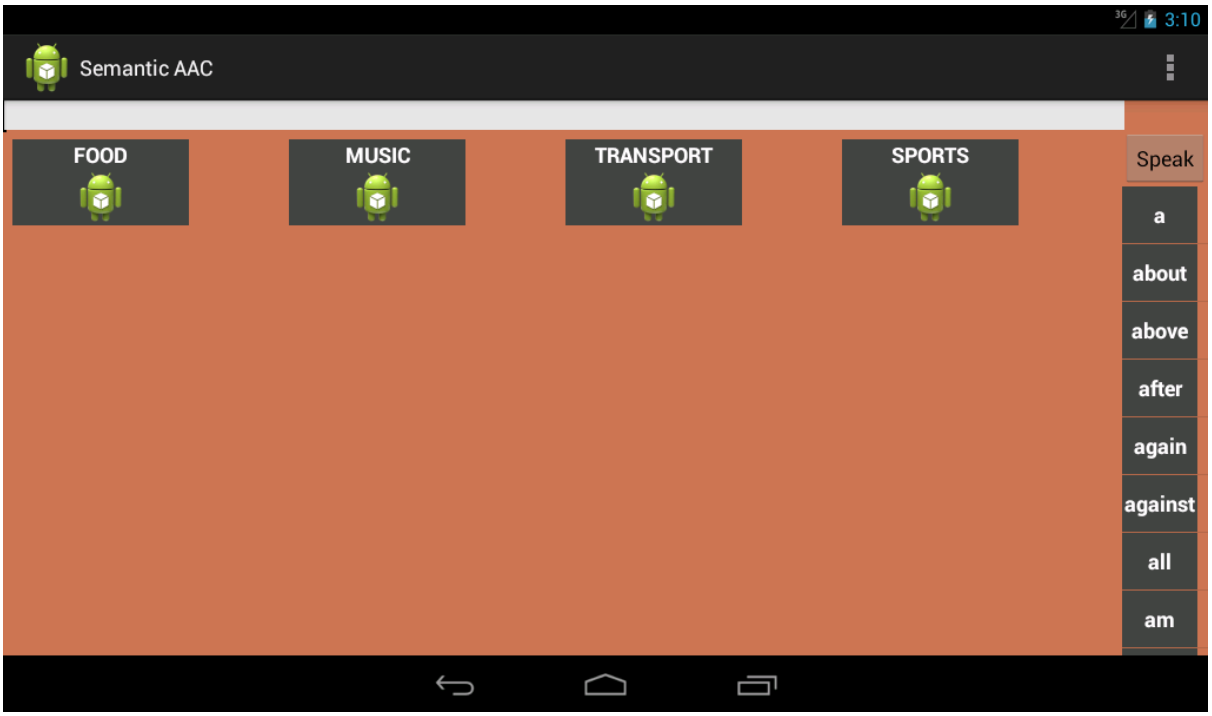


Figure 32 First Attempt at Reintroduction of Stopwords.

Version 2 also introduced pictorial cues for categories in line with AAC development guidelines (figure below).

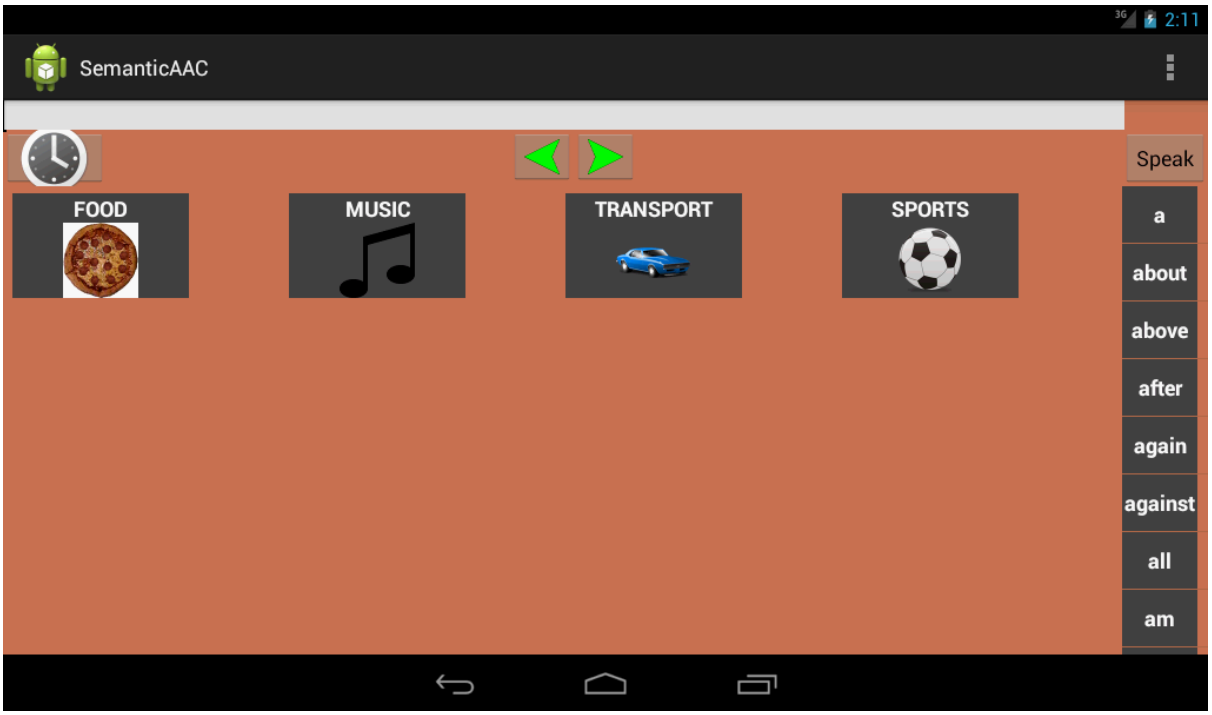


Figure 33 Introduction of Pictorial Cues and New Features

New features introduced in Version 2 were the ability to navigate a history using the green arrows at the top of the screen, words related to time (e.g. months, days of the week), accessed by the user using the button with the picture of a clock.

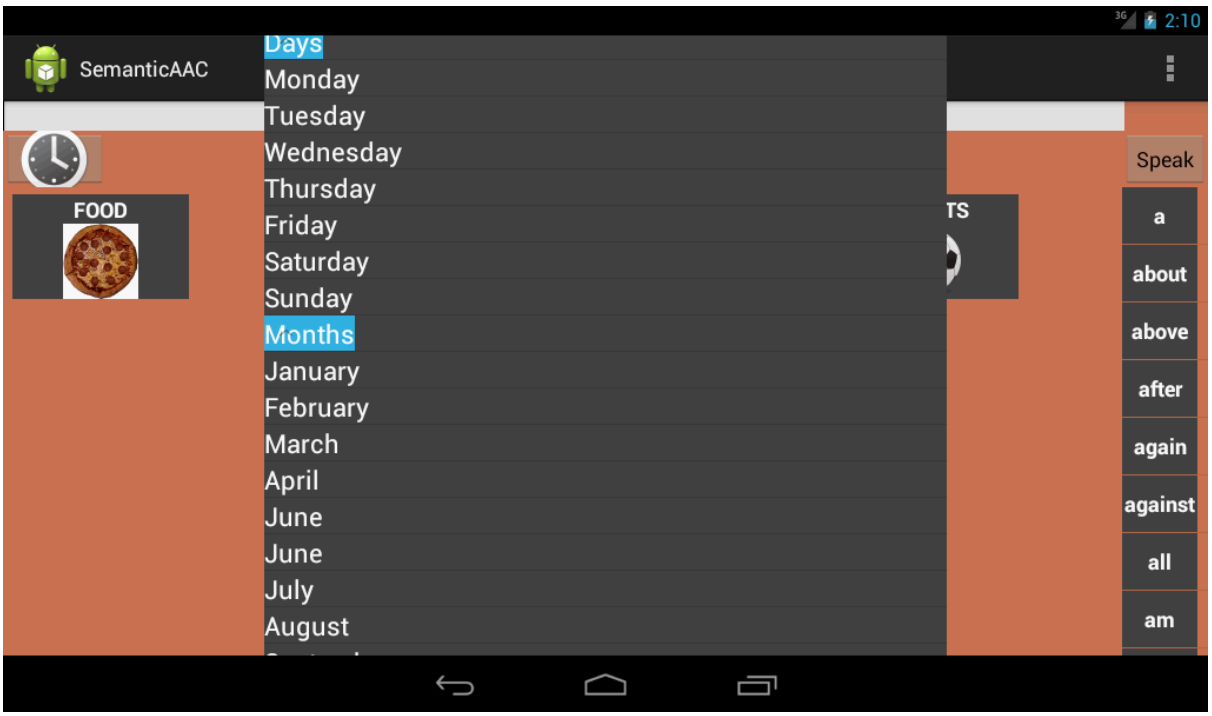


Figure 34 Words to Express Time

7.2.4 Version 3 (No Natural Language Generation)

Version 3 is the first version to feature text prediction via the use of the OpenAdaptTxt prediction engine[26].in an effort to facilitate sentence generation. Suggestions from the engine are shown in the list on the right hand side. The suggestion engine listens for changes inside the edit bar, and whenever the event is triggered suggestions are updated, the switch button is used to switch places of the last two words for cases where the adjective ends up before the noun.

The Word Map used by this version was also modified (as discussed in the pre-processor Chapter) to include only nouns. The insertion of verbs was delegated to a second Word Map that appears on the right hand side of the screen and contains verbs that are most frequently associated with the selected noun. Connecting elements and suffixes for verbs and nouns were delegated to next word prediction generated by the OpenAdapTxt engine. The use of OpenAdaptTxt (OATXT) was abandoned in favour of using other techniques for two main reasons: the first is that OATXT requires training to provide relevant predictions, the second is that thanks to the way Dictum is structured most of the generation is already provided by the NLG Server and it was concluded that OATXT could only be used to fill syntactic gaps or to improve the grammatical correctness of generated sentences. OATXT was therefore taken out of the system and the tasks of filling syntactic gaps and improving sentence correctness were delegated to the Microsoft N-Gram Web service. The specifics of this will be discussed in Chapter VIII.

[26] http://openadaptxt.sourceforge.net/

7.2.5 Version 3 (Inclusion of Natural Language Generation)

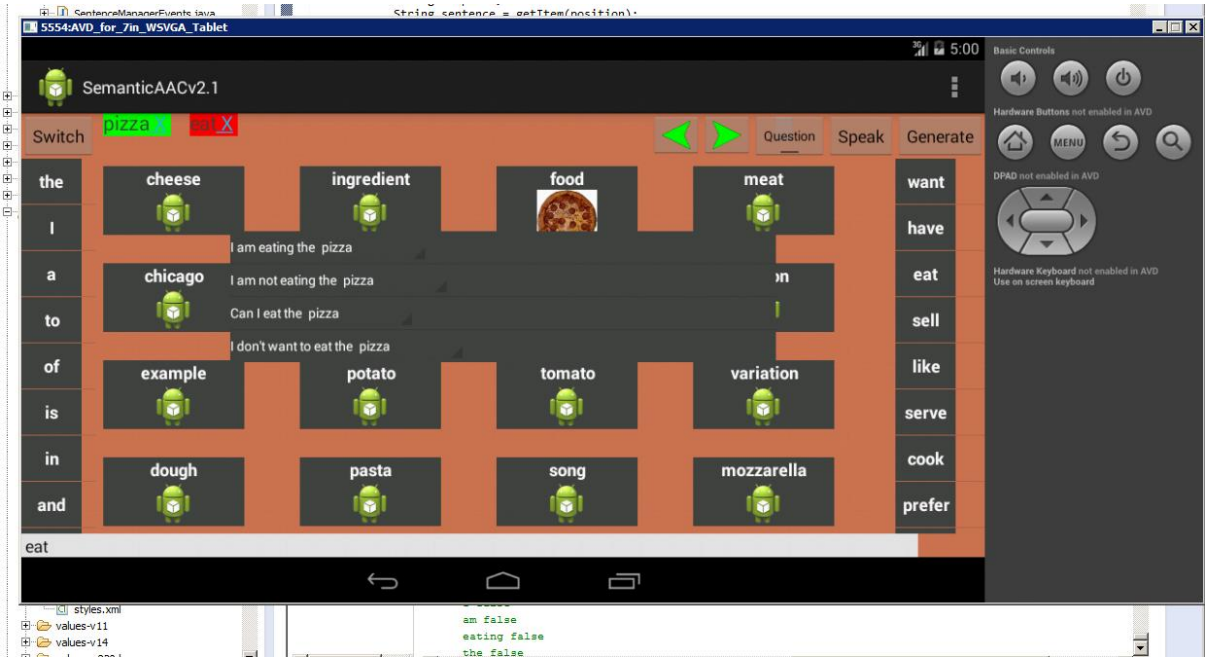


Figure 35 Screenshot of Version 3 of the App

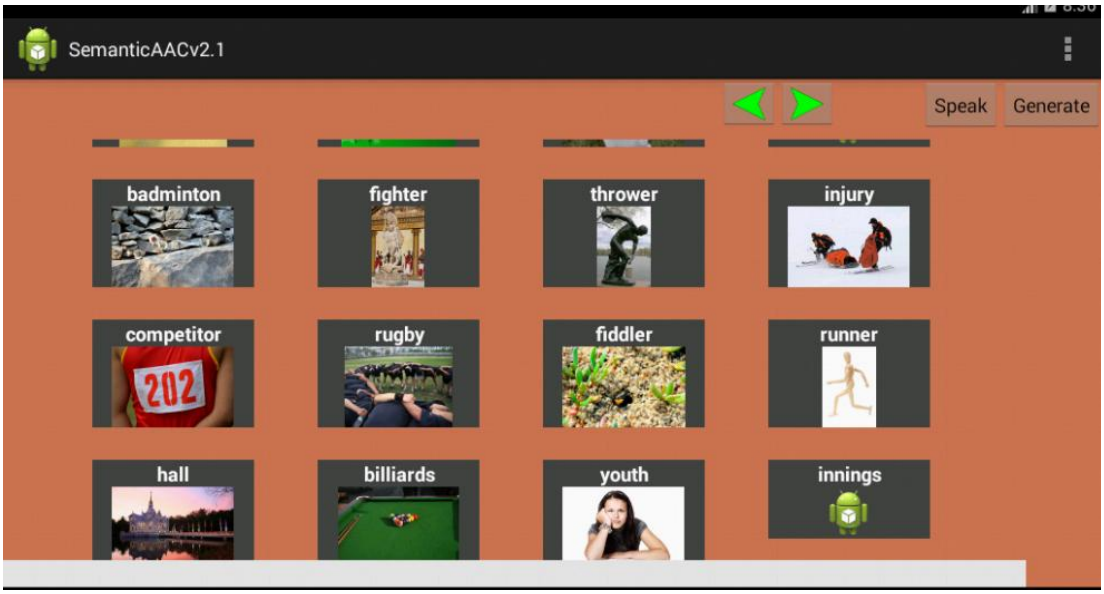


Figure 36 Screenshot of Dictum with Pictorial Cues

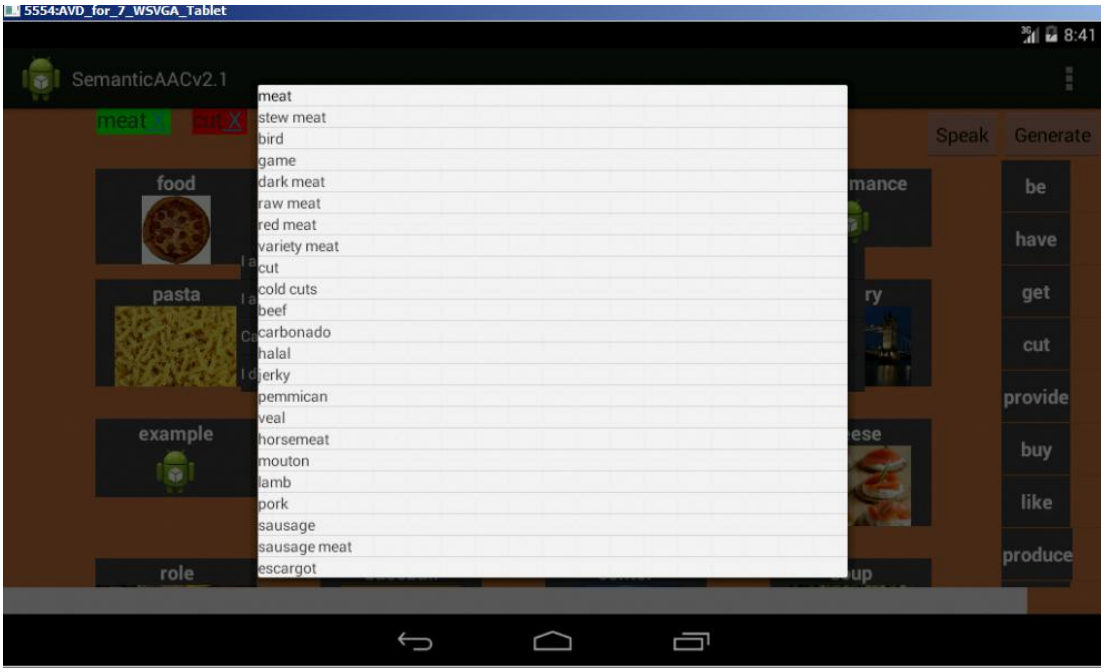


Figure 37 Screenshot of Dictum presenting Hyponyms to Offer more Choice to Users

Version 3 with NLG is the final version of the Dictum. Several changes made to this version became new features. The interface was changed to handle the insertion of verbs and nouns which are displayed on the upper part of the screen. This is shown in the screenshot above. As a first attempt nouns were given a green background and verbs were given a red one. This was later changed. The edit text containing the final sentence was moved to the bottom of the screen.

The justification for these changes is to shift the burden of responsibility for sentence generation to the application rather than the user. Whilst the integration of the text prediction engine does greatly improve the word insertion speed, sentence formation directly using the prediction still requires a relatively high number of taps and some time looking for the correct word. In line with the guidelines given by Light and others (Light et al., 2008) delegating the responsibility of sentence formation to Dictum increases fluidity of communication and does not distract the user from his interlocutor.

Version 3 was modified to integrate the Natural Language Generation engine in the application. Due to hardware limitations NLG is performed on a remote server. The application connects to the remote server via SSH tunnelling when the user clicks on the "Generate" button.

As will be explained in the next Chapter the interface and the algorithm were developed so that there would be no incorrect input from the user. For example,

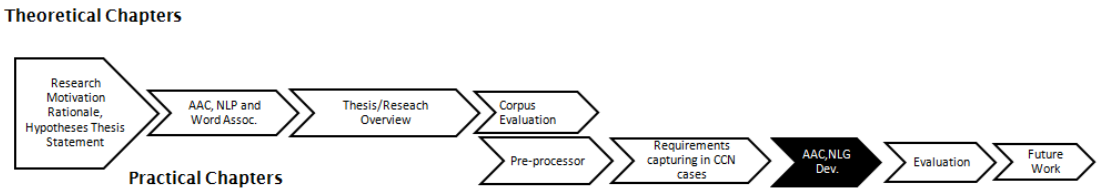
even if a user decided to delete a noun and leave verbs the application would still accept the input as valid and attempt to generate sensible sentences with what is available.

The text prediction part was delegated to the server and was removed from the application. This was done in line with the main goal of delegating as much of the sentence formation responsibility as possible to the app.

The next Chapter will discuss the NLG server which the app connects to in detail.

7.3 Summary

This Chapter looked at the implementation of Dictum: the AAC application central to this work. The app was developed in Android. The development of the application followed an iterative process in line with the Action Research methodology followed as part of this project. Each version of the application was shown to potential users who gave feedback which was included in the subsequent iteration. In total three versions of the application were developed; the third version of the application included the ability to connect to an NLG server to give input from the user to the server and receive grammatically correct sentences back.



Chapter: VIII Natural Language Generation

8.1 Introduction

The purpose of this Chapter is to describe the Natural Language Generation algorithm used and developed as part of this work to delegate the responsibility of sentence formation to the AAC application. Firstly this Chapter will discuss the high level design of the NLG system by comparing it to the Compansion algorithm which was used as a blueprint. Secondly this Chapter will discuss the implementation details of the NLG system and justify why the system requires the use of a server. The Chapter will conclude with a discussion on the unimplemented aspects of the NLG, with particular focus on their feasibility and how their design was created in such a way as to be easily integrated into a full scale system.

8.2 Natural Language Generation Server

This paragraph will discuss the technical details of the Natural Language Generation server. The way that natural language generation server was designed is largely based on the Compansion algorithm. However the Compansion algorithm is mostly theoretical, even though there has been an attempt by McCoy and Pennington to make it practical (McCoy and Pennington, 1998). The algorithm developed in this work is in many aspects different..

To justify why certain decisions were made the Compansion algorithm will now be explained in fine detail. After that the differences between the algorithm presented here and the Compansion algorithm will also be discussed. Even though this is not a theoretical Chapter the discussion is made here as it is relevant to the subsequent technical discussion.

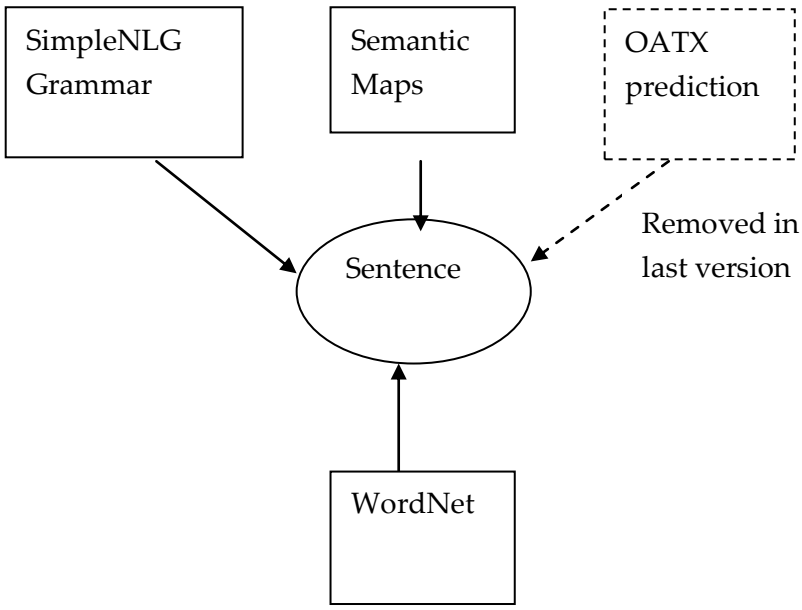


Diagram 4 The Architecture of the of the NLG Server

8.2.1 The Compansion Algorithm in Detail

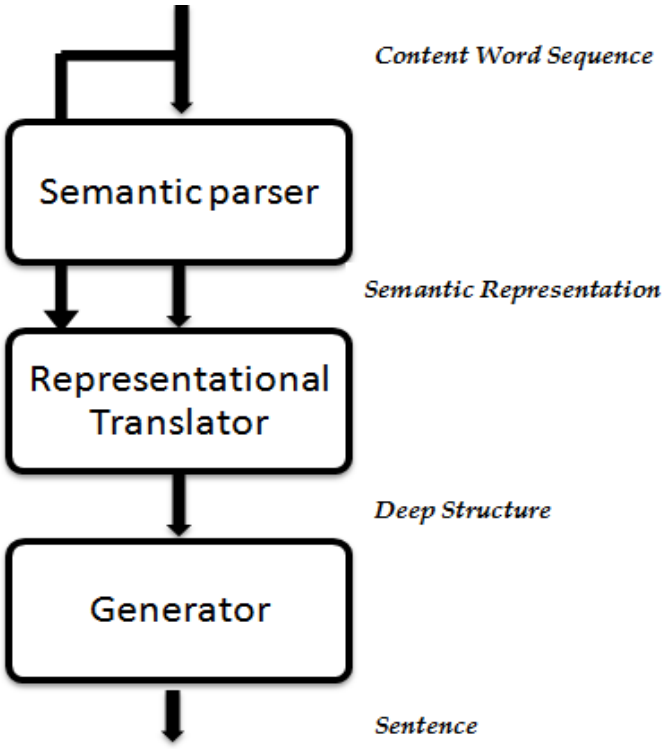


Diagram 5 High Level Representation of the Compansion Algorithm

8.2.1.1 Introduction to the Compansion Algorithm

Chapter II (Theoretical Background) gave a brief overview of the Compansion algorithm. It showed the figure above and explained the main stages of this algorithm. Compansion will now be explained in fine detail so that it can be compared with the modified version developed as part of this work.

The input of the algorithm is lemmatised words which can be objects (animate or inanimate, so all nouns) verbs, or adjectives. For the sake of explaining the algorithm clearly we will assume these three constitute the only type of input possible for Compansion.

The semantic parser takes the input given and tries to assign a possible role that each word will have in potential sentences. In this scenario each part of a sentence is called a sub frame, the generated sentence will be called frame. In frame representation nouns are said to have a role with respect to the verb.

It will be seen later that the Formal Language developed in my work nouns also play a role with respect to verbs. This relation is clearly visible in the formal representation of input which gets sent to the server from the AAC application.

Nouns can typically have three main roles in a sentence: they can be ACTORs (the ones which perform an action), they can be LOCATION (a place where the action takes place) or they can be THEMEs (the object or person being acted upon).

The first obstacle to the success of algorithm like Compansion is that semantic parsing usually requires a full input sentence for semantic parsing to be effective. This is because the first step of the semantic labelling process is to perform syntactic parsing (determine the syntactic role of each word within the sentence. A brief explanation of syntactic parsing was given in Chapter 2) then, based on a set of grammatical rules to find the most appropriate semantic role.

8.2.1.2 Word Categories and Semantic Taxonomies

To overcome the limited input Demasco and McCoy rely on hardcoded semantic information associated with every word available in the system. The main categories of the Compansion system are:

- Verbs
- Objects (which include any noun)
- Adjectives
- Adverbs

Each word in the system can belong to multiple categories. The most intuitive example is that of words that can be both nouns and verbs.

Each of these categories is broken down into a semantic taxonomy

- Verbs
    - Verbal
    - Relational
    - Mental
    - Material
- Objects
    - Physical
        - Animate
        - Inanimate
    - Abstract

Compansion also includes further subdivisions but for the purposes of this discussion mentioning these will be enough. There are also semantic taxonomies for adverbs and adjectives. But since the AAC app does not currently include adjectives and adverbs as possible input from the user they have been omitted from the above list.

8.2.1.3 Semantic Parsing

Once Compansion checks for all possible roles for all words in the input in the pre-stored taxonomy it generates a sub frame for each word inserted. A subframe consists of the word itself alongside a list of possible roles that the word could have within a sentence.

Demasco and McCoy use the input JOHN STUDY as an example. Two subframes would be generated given this input. One for JOHN, with its taxonomy classing it as an animate object and two for study: one classing it as a location and the other as verb.

The goal of the semantic parsing (the first step of Compansion) is to generate a frame that contains the semantic role of each word in the input and some additional data about the sentence that can help the other parts of the pipeline, such as the type of sentence and the tense of the main verb. The diagram below shows the output of the semantic parser for the input JOHN STUDY WEATHER UNIVERSITY. This input is used by Demasco and McCoy to explain the fine details of the pipeline. Each of these word is called a subframe.

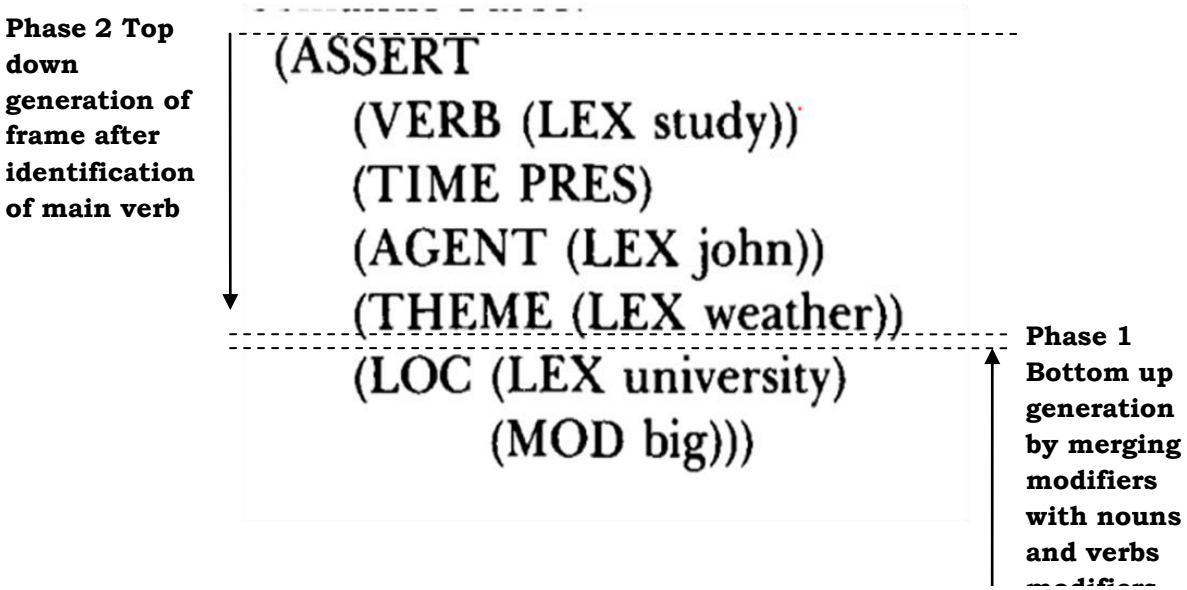


Diagram 6 Output of the Semantic Parser of the Compansion Algorithm

Each subframe in the input can have three possible roles:

1) It is the main verb of the sentence
2) It is a noun phrase
3) It is a modifier of either a verb or a noun

Phase 1 of the semantic parsing deals with the third possibility first: it tries to merge modifiers (adjectives and adverbs) with the noun they modify. Once modifiers have been identified and coupled with the word they modify the two subframes (modifier and modified) get merged into a single frame and are considered dealt

with by the parser, exploiting the rule that particular adverbs and adjectives affect particular nouns and verbs. For the example used by Demasco and McCoy (JOHN STUDY WEATHER BIG UNIVERSITY) UNIVERSITY and BIG will be merged.

Once all modifiers have been dealt with, the parser tries to identify the main verb of the sentence. Identifying the main verb helps to assign the semantic role for the rest of the sentence because it dictates much of the structure of the sentence. If for instance the main verb of a sentence is GO there won't be a THEME role but there will definitely be TO LOC role and possibly a FROM LOC role.

Each verb in the taxonomy contains skeleton frames that list all possible roles for roles and links to that verb (e.g. the verb STUDY will have an AGENT that performs the THEME and a THEME role).

If there are multiple verbs in the input frames all verbs are scanned until one of the verbs semantically fits with the other frames. Back to the example there are two possible verbs WEATHER (to endure) and STUDY. WEATHER is discarded as main verb because JOHN can be an actor but there is no valid THEME to perform the WEATHERing action. STUDY therefore becomes the main verb. JOHN becomes the ACTOR because it is the only role it can assume. UNIVERSITY and WEATHER could both be THEMES but since UNIVERSITY can be both a THEME and a LOCATION, the theme becomes WEATHER, and the UNIVERSITY assumes the role of LOCATION.

8.2.2 Dictionary Translation and Sentence Realisation

The result of semantic parsing is passed on to the Dictionary Translator. This part is responsible for translating the output of the semantic parser. In practice the translator determines how each piece of the semantic structure can be realised (turned into a sentence); the Dictionary Translator is also responsible for ensuring that the original order of input is maintained in the final sentence. The semantic parser and the Dictionary Translator are kept separate because the same sentence can be expressed in different ways depending on the context where the sentence is being presented.

The result of the Dictionary Translator is passed on to the Sentence Generator which ultimately generates the final English sentence. As discussed in Chapter II there are many approaches to sentence generation. The inventors of the Compansion system opted for the use of Formal Unification Grammar (FUG). As explained in Chapter II

FUGs and their variants describe sentences as sets of pairs consisting of an attribute and a value. This attribute value description of language is formally referred to as Functional Description (FD). This formalism is particularly suited to represent human language and grammar in a computationally friendly manner. The attribute describes the nature of the value (e.g. category of word, whether a word is plural or singular, verb tense and so on). Attributes in FUGs are usually either syntactic, semantic, or functional. To work, FUGs require two things, a lexicon (a set of words with grammatical information. (e.g. what parts of speech it can assume, how it is written in singular and plural, how the word is conjugated if it is a verb, etc.), and the grammatical rules of the language. Both the lexicon and the grammar are described using the FD formalism. According to Demasco and McCoy one can think of the FD describing as a set of holes where the input to the Compansion system, also described using Functional Description fits to generate sentences. Merging input and grammar is called unification. The image below is taken borrowed from the work of Demasco and McCoy and shows the unified Functional Descriptions which if passed on to their generator would generate "The apple is eaten by John"

```
((CAT S)
 (FOCUS GOAL)
 (PROT
   ((NNP ((N === JOHN)))
    (PROPER YES)))
 (VERB
   ((VVP ((V === EAT)))
 (GOAL
   ((NNP
    ((N === APPLE))))))))
```

Functional Description 1: The Functional Description Which Would Generate the Sentence "The Apple is Eaten by John"

In the Functional Description above, the main attributes are CAT and FOCUS; CAT denotes the category of the Functional Description. An S at the root of the Functional Description denotes that we are formalising a sentence. GOAL as FOCUS denotes that the focus of the sentence is a particular goal. In practice this means that the goal is moved to a prominent position in the sentence, and that instead of saying John eats the apple, the focus shifts on the apple being eaten by John. The rest of the FD specifies that the protagonist (PROT) is the proper noun JOHN and the action is EAT, and the GOAL is APPLE. This FD of the sentence is unified with the FD of the

English language to generate the full FD for the sentence. The final FD is passed on to the generator which takes care of generating the proper English sentence.

8.3 NLG Server

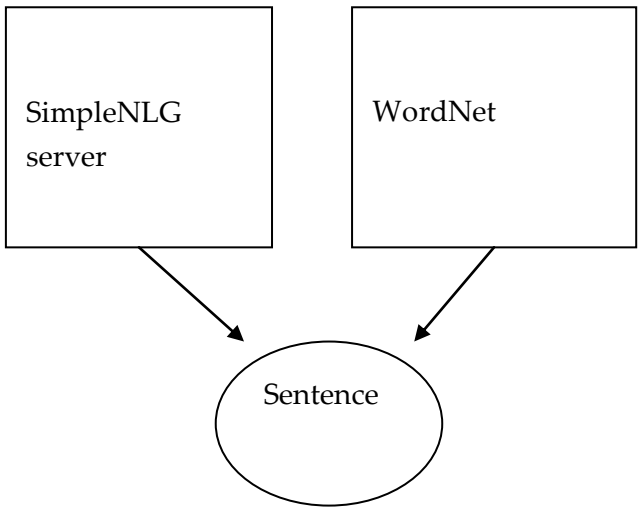


Diagram 7 The Current NLG System

The central part of the NLG system is the Java SimpleNLG library (Gatt and Reiter, 2009). To work, this library requires a lexicon which contains a list of words representative of the English language correlated with additional grammatical information such as for example conjugations and plural forms. This lexicon generates a heavy memory demand as it is loaded in memory for fast retrieval. For this reason it was decided to delegate the Natural Language Generation responsibility to a dedicated server. In practice the prototype sends a POST request containing a formal string to a desktop PC via SSH which executes a JAR file and returns a list of possible sentences with the given input.

8.3.1 Comparison of the Algorithm Developed as Part of this Work and Compansion

The Compansion algorithm was explained in detail because it was used as a blueprint to develop a similar algorithm to generate text from compressed input that would work in practice and integrate seamlessly with the AAC application.

This paragraph will explain the Natural Language Generation server developed for this work in detail by comparing it with the Compansion algorithm.

The algorithm required for the Language Generation has the following pre-requites:

1. There is no incorrect input
2. The order of the words does not matter
3. There are no limits to the number of nouns and verbs
4. Nouns cannot be subjects and do not perform actions

and the following constraint:

1. User has access to verbs only after selecting noun

The input to the algorithm is words selected by users on the AAC application of two types:

- Nouns
- Verbs

The nouns are stored in the vocabulary files discussed in the pre-processor chapter. They are selected by tapping on words on the grid at the centre of the screen. As explained once a noun is selected, the list of available nouns changes based on nouns that are related to the last word selected so a user can input as many nouns and verbs as he/she wishes. When a noun is selected a list of related verbs appears on the right hand side of the screen. The process of generating the verb list was discussed in the pre-processor Chapter (Chapter V).

Compansion stores information by storing the word only once and then adding multiple roles associated for that word. In contrast the algorithm developed as part of this work stores the words in multiple files. Whilst this approach is costly in data storage it has the advantage of not requiring semantic parsing at all because even if a word could potentially assume the role of verb and noun, there is no ambiguity as there are only nouns on the grid and only verbs on the buttons of the right hand side of the screen. The chosen storing method also integrates seamlessly with the AAC interface.

Even today the Compansion algorithm remains a theoretical one which has not been implemented. In contrast the NLG algorithm discussed here was implemented and integrated into the prototype, albeit it was not realised fully due to the time

constraints and due to the fact that the aim of this work is not to produce a fully functional system but rather to prove or disprove the given hypotheses.

The main drawback of the Compansion algorithm is that it requires semantic parsing at runtime. The algorithm developed as part of this work bypasses the need for semantic parsing as the role for each word of the input can be inferred from the message sent to the NLG server without ambiguity. This is because if a user selects a word from the sentence it will always be a noun, if the user selects a word from the right hand side list it will always be a verb.

The algorithm also limits the grammatical permutations that each combination of words could generate. The Compansion algorithm relies on the order of words to infer what is meant. The algorithm for the AAC has to be able to generate sentences consistently regardless of the order of the words, as it is possible for a person with CCN to mix up the order of words. To achieve this, the NLG server generates only certain kinds of sentences. In Compansion it is also possible for inanimate objects to be the subject of sentences (e.g. the apple is being eaten by John as opposed to John is eating the apple). In the NLG server discussed here the subject is always the first person. It is likely that an expansion to this system would be required for a fully functional system, for example to cater for cases where the user talks about someone else. But it is my opinion that enabling sentences in the passive form would introduce ambiguity and increase algorithmic complexity unnecessarily.

The Compansion algorithm also relies on the generation of parse trees to decide on the outcome. It was discussed that a parse tree is generated which in turn gets translated into possible sentences based on lexicon. To avoid this and speed up the process it was decided to employ the SimpleNLG library to perform the sentence realisation.

The diagram below shows a high level view of the sentence generation process. When a user has completed noun selection he can press the “Generate” button to send data to the server. The next paragraph will discuss the rules of the Formal Language.

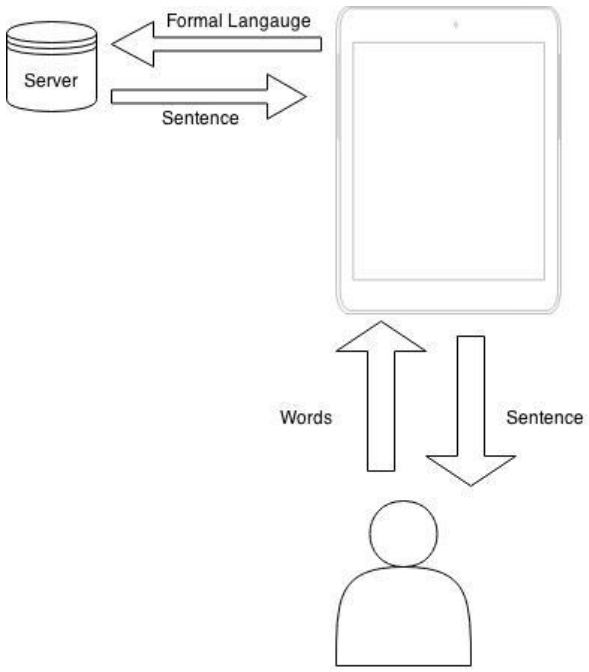


Diagram 8 Communication Between App and NLG Server

As a bridge between the application (acting as client) and the server a Formal Language was developed to send a sort of metadata to the server. Using a Formal Language is common and most NLG engines available do so[27] (Elhadad and Robin, 1996). The reason it was opted to develop a personalised Formal Language instead using one that was available was that these Formal Language are developed for very complex linguistic tasks such as Machine Translation (Elhadad and Robin, 1996; Kay, 1984) and due to the restricted timescale available it was decided that it would be more feasible just to create an ad hoc Formal Language. Another reason for this was that the Formal Languages available are written in LISP or other specifications which need to be mastered before using the engines That is another barrier to their use as it takes some time to get comfortable with the system[28], and ad hoc Formal Language can also be adapted to the way in which input is given by the user.

The next paragraph will discuss the Formal Language. Not all of the features of the Formal Language are used by the application but they were included in the

[27]http://www.nlg-Wiki.org/systems/
http://www.fb10.uni-bremen.de/anglistik/langpro/webspace/jb/info-pages/nlg/ATG01/node19.html[28]

discussion to show that the language is scalable and it can be easily adapted to future versions of the application.

8.3.2 Formal Language

This paragraph will give a detailed explanation of the Formal Language used by the client and the server to generate sentences. The client will generate a Formal Language sentence given the input from the application, and the server will translate that into sentences.

These are the symbols used by the Formal Language:

- [] denote Start - End of word relationship. A relationship is intended as a link between nouns and verbs and nouns and adjectives.
- () word group, words of the same kind are grouped here. Words in this group are affected in the same way. Word group can be groups of nouns or groups of verbs.
- # denotes Part of Speech , v,n respectively verb noun.
- - denotes that noun(s) affect(s) verb(s)

Diagram 9 High Level View of the NLG System Developed in this Work

The symbol # was chosen because it is used by the WordNet library when querying for different word senses. This way the application is facilitated when using data from WordNet.

A dash is used to indicate a relationship between parts of speech. Typical usage scenarios are listed in Table below

Action	**Formal Language Output**
User selects a noun	[(Word#n#0)]
User selects a verb after selecting a noun	[(Word#n#0) –(verb#v#0)]
User selects multiple nouns	[(Word1#n#0; Word2#n#0)]
User selects multiple verbs related to 1 or	[(Word#n#0Word2#n#0)-(verb#v#0;verb2#v#0)]

many nouns

User creates separate relationships [(Word#n#0) –(verb#v#0)] [(Word#n#0) –(verb#v#0)]

The table below shows example outputs from the Formal Language.

Formal Language	**Result**
[(car#n#0) –(drive#v#0)]	I want to drive the car
	I do not want to drive the car
	I am driving the car
	I am not driving the car
	Can I drive the car?
[(car#n#0)]	I want to drive the car
	I do not want to drive the car
[(pizza#n#0; bread#n#0)]	I want the pizza and the bread
	I do not want the pizza and the bread

[(Pizza#n#0bread#n#0)-(buy#v#0;eat#v#0)]	I want to buy and eat the pizza and the bread I do not want to buy and eat the pizza and the bread
[(Piiza#n#0) –(eat#v#0)] [(water#n#0) –(drink#v#0)]	I want to eat the pizza. I want to drink the water. I do not want to eat the pizza. I do not want to drink the water

8.4 WordNet Server

WordNet was examined in Chapter IV. The information contained in it is also used by the NLG server to expand on the available vocabulary without placing a further load on the application. By exploiting the hyponym relations between sysnset WordNet expands the available vocabulary considerably. The principle is that verbs that affect a root word also affect its hyponyms. For example if a dog barks a poodle also barks much like the principle of inheritance in object oriented programming.

Figure 38 shows the "WordNet Search Engine[29]". This website can be used to query WordNet. The figure shows the result of querying the search engine for the word *meat* and then looking at all of its hyponyms. Hyponyms were used to expand the application's vocabulary.

[29]http://WordNetweb.princeton.edu/perl/webwn

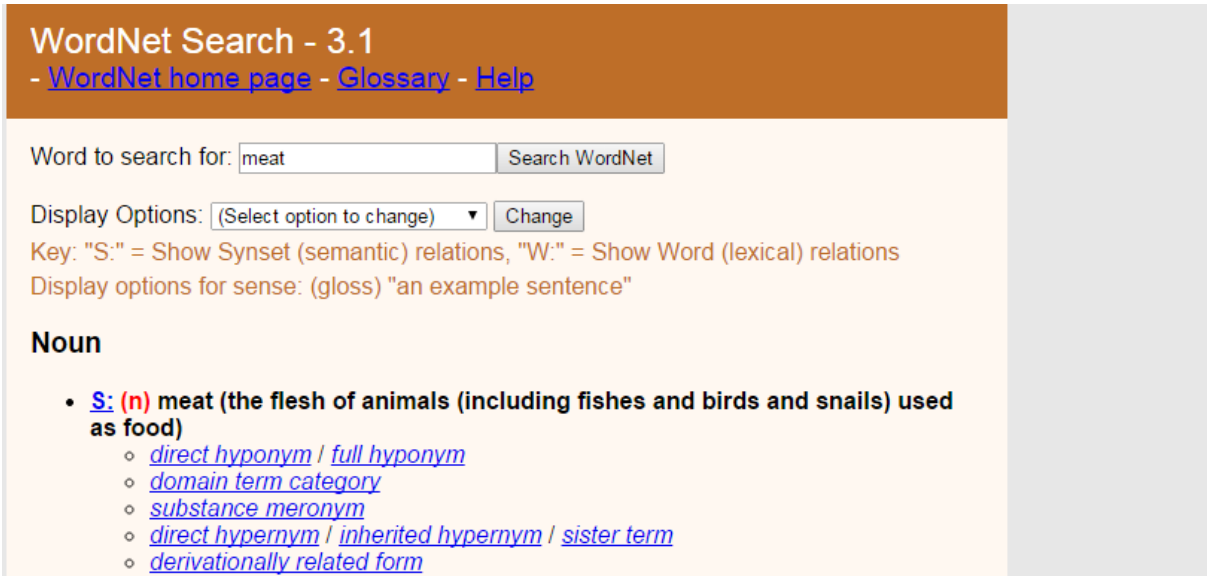


Figure 38 The WordNet Web Search Engine

- S: (n) **meat** (the flesh of animals (including fishes and birds and snails) used as food)
  - ***direct hyponym*** / *full hyponym*
    - S: (n) stew meat (tough meat that needs stewing to be edible)
    - S: (n) bird, fowl (the flesh of a bird or fowl (wild or domestic) used as food)
    - S: (n) game (the flesh of wild animals that is used for food)
    - S: (n) dark meat (the flesh of the legs of fowl used as food)
    - S: (n) raw meat (uncooked meat)
    - S: (n) red meat (meat that is dark in color before cooking (as beef, venison, lamb, mutton))
    - S: (n) variety meat, organs (edible viscera of a butchered animal)
    - S: (n) cut, cut of meat (a piece of meat that has been cut from an animal carcass)
    - S: (n) cold cuts (sliced assorted cold meats)
    - S: (n) beef, boeuf (meat from an adult domestic bovine)
    - S: (n) carbonado (a piece of meat (or fish) that has been scored and broiled)
    - S: (n) halal ((Islam) meat from animals that have been slaughtered in the prescribed way according to the shariah)
    - S: (n) jerky, jerked meat, jerk (meat (especially beef) cut in strips and dried in the sun)

8.5 Theoretical Implementation of Additional Functionalities of the Natural Language Generation Server

So far this Chapter has explained the NLG to the extent in which it was implemented in software. This paragraph will discuss the design of the fully implemented system from a theoretical model that can generate grammatically correct full sentences.

This paragraph will start off with an expansion to the Formal Language. Firstly the Formal Language will be expanded to cater for as many communicative situations as possible.

Secondly it will discuss how an investigation was conducted into finding a way to fill the syntactic gaps left by the implemented system. It was my initial intention to include text prediction into the system but since most of the system already generates full sentences there is little need to predict the next word but rather to correct grammatical assumptions such as finding the correct determiner before a noun.

Using Adjectives

An adjective is specified by concatenating an = symbol to the noun string

Formal Language	Result
[(car#n=adjective)]	Car "Adjective"
[(word#n =red,big)]	Big and Red Car

A noun can have multiple adjectives

Specifying different subjects or talking about activities with other people

Formal Language	Result
{John}-[(pizza#n)-(eat#v)]	John and I are eating some pizza
[(pizza#n)-(eat#v)]-{John}	I am eating some pizza with John

Being in a location, going to a location, coming from a location

Negatives and expressing the inability to do something:

Negations are made by adding a !symbol before a verb or a noun

The inability to perform an action is express by a !!before a verb.

Formal Language	Result
[(car#n; !bus#n)-(drive#v)]	*I am driving the car and not the bus*
	I can drive the car but not the bus
	I want to drive the car but not the bus
[(car#n)-(!!drive#v)]	*I can't drive the car*

8.5.1 Improving the Grammatical Correctness of Sentences Using other NLG techniques

The implemented system at present has two main components: the "SimpleNLG" engine and the data generated by querying hyponyms to expand the available vocabulary. SimpleNLG is responsible for taking care of the grammatical structure of sentences and it was previously discussed how this is done by comparing the algorithm of this work with the Compansion Algorithm of Demasco and McCoy. At present however the system makes some assumptions to make some sensible sentences which are not necessarily grammatically correct. For example the system assumes the determiner the before the noun, this is not always correct because the user might be asking for a quantity of something and even if that quantity of something has not been specified by the user it might still not be syntactically correct to use "the" in some cases.

8.5.1.1 Using sentence likelihood to fill syntactic gaps left by the current NLG server

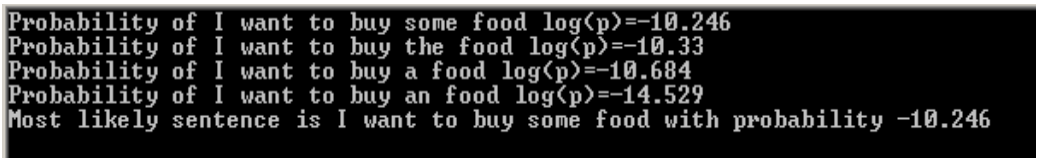


Image 12 Sentence Likelihood Calculated for Test Sentences via Microsoft N-Gram Service

The feasibility of using sentence likelihood to fill the gaps left by the implemented NLG, was investigated by using the web service offered by Microsoft at the Microsoft N-Gram[30] website. A small C# project was created that sends requests to the server to measure the likelihood of some test sentences.

To use the service, the developer requests a user token and as long as he/she doesn't send concurrent requests to the server using the same token he/she is free to send as many requests as required.

The web server lets the developer choose up to a 4-gram language model. To implement this small study a 4-gram model was used, as the higher the number of grams the more precise the statistical prediction is.

The scale used by the web service and seen in the screenshot is in logarithm to avoid numbers cancelling out. Because calculations are dealing with such small numbers if the raw probability is used, the final result becomes smaller and smaller and eventually cancels out. Using logarithm saves space, avoids cancelling out problems and the various probability logs can be summed to calculate the final probability.

This investigation showed that using N-Grams to estimate sentence likelihood is an effective method of assigning the correct determiner to nouns.

8.5.1.2 Using Syntactic Parse Trees to Fill Syntactic Gaps Left by the Current NLG server: A Theoretical Investigation

This approach was investigated for completeness however due to the timescale and to the algorithmical complexities required to speed up the comparison of the trees it was not investigated in practice and not used in this work.

[30] research.microsoft.com/en-us/collaboration/focus/cs/web-n-gram.aspx

As a way to improve the grammatical correctness of sentences generated by system the Treebank database was explored. The tree bank system is a collection of parse trees which can be queried using a specified SQL like language show below.

The idea behind this investigation was to query the database to look for sentence structures which were similar to the ones generated by the system in an effort to improve the final result. As with the n gram example the objective was to find the correct the determiner, given that the system always gives "the" by default. To achieve this, the queries searched for all sentences that contained a particular noun preceded by a determiner, the determiner which preceded the noun most frequently was deemed to be the correct one.

However this approach was discarded in favour of using the Microsoft Web server approach discussed in the following paragraph stop this was because the Microsoft Web server contains more words and querying is a less time-consuming process, as it is sufficient to calculate the statistical likelihood of a sentence with each determiner.

8.5.1.3 Other Uses of WordNet to Improve Sentence Quality

So far only hyponyms have been used as a way to expand the vocabulary. Other uses could include:

- The use of antonyms to improve negations:

  For example if a person wants to communicate that he does not want to do something, the opposite could be offered as an alternative in the form of a positive statement.
- The use of meronyms and holonyms to improve sentence correctness

  If there are multiple nouns in a potential sentence the algorithm should check if there is a meronym/holonym relation between a pair of words and exploit that to form better sentences.

A small program was developed using the PERL WordNet:Query library to assess the feasibility of programmatically analysing relations between words. The algorithm, given a set of 2 or more nouns checks whether there is meronymity between them For each pair of nouns the method queries all meronyms of the first

word and checks if the second word is present in the resulting list. Output from the algorithm is shown below.

```
comparing words bread and flour
 flour is meronym of  bread
comparing words bread and soup
comparing words flour and bread
comparing words flour and soup
comparing words soup and bread
comparing words soup and flour
flour
```

Figure 39 Output from WordNet Meronymity Check Algorithm

8.5.1.4 Limitations of WordNet and possible solutions

WordNet lexicon is limited and sometimes even if words are present in the lexicon, relationships that one would expect are not present. This happens either because there isn't a strict relationship between the words in the grammatical sense but one is generally assumed (i.e. the case of sugar and doughnut or pizza and mozzarella) or because it is altogether missing from the dataset.

This does not mean that this method is unfeasible. There are solutions that have been developed to find meronyms with unseen datasets. For example (Girju et al., 2006) successfully implemented a learning algorithm to train a computer to detect meronymity between words.

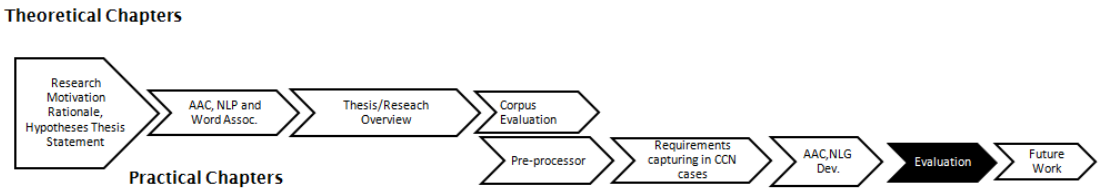
8.6 Full NLG Pipeline

1) Scan WordNet for hyponyms of nouns inserted to present the user with sub types of words.
2) If there are multiple nouns check if there is a relationship between them.
    a. If the words are meronyms /holonyms modify the potential sentence to show a relationship between the nouns.
3) Use the Realiser to generate a temporary sentence with "the" determiners as placeholders
4) Pass the sentence to the Microsoft N-Gram determiner finder to assign determiners correctly.
5) Pass the sentence to SimpleNLG to realise the sentence.
6) Output to user.

8.7 Summary

The purpose of this Chapter was to discuss the implementation and design of the NLG server which communicates with the application developed as part of this work to generate sentences. The Chapter started off with a discussion on the compunction algorithm which was used as a blueprint for the algorithm used by the server. The main difference between the Compansion algorithm and the algorithm discussed in this Chapter is that the algorithm developed here is practical whereas compunction remains mainly theoretical. Some other aspects of the Compansion algorithm where also modified whether because there were limiting or to make the system fit better with the application. The Chapter then continued to explain the practical implementation of the NLG server explaining each component. In the interest of time some aspects of the system remained unimplemented however they were discussed here and it was shown that they were thoroughly investigated and feasible and they could be easily integrated in a fully functional system.

.



Chapter IX: Evaluation of AAC Application

9.1 Introduction

This Chapter will discuss the evaluation of the AAC application. The evaluation was conducted both qualitatively and quantitatively. In line with the Action Research methodology chosen for this project, qualitative data were provided by potential users. The users were the same members of the Dundee based group which gave feedback on the initial version of the app and provided data on which Appropriation analysis was performed (discussed in Chapter VI). At this meeting both users and speech and language professionals were present; professionals also provided impressions and feedback on the application.

Quantitative data were gathered first by comparing Keystroke Saving percentages of the application developed in this work against two popular apps available on the iOS market which were mentioned by both users and professionals during the course of this research, and second by comparing the interfaces of the three applications using a recent HCI user performance prediction model which combines two well established HCI laws: Fitts' Law and Hick Hyman Law.

9.2 Qualitative Evaluation

9.2.1 Follow Up Meeting with Potential Users of the *Tap and Talk Group* at the University of Dundee

To evaluate the final version of the application another meeting was organised with the "Tap and Talk" group at the University of Dundee. A total of 6 people with aphasia was present (two were the same who were present at the first meeting), a Computer Scientist and two Speech Therapists were also present at the meeting.

9.2.2 Feedback from Participants with Complex Communication Needs

The meeting started with the author of this work showing a video on the app, listing all of its functionalities and showing how to form sentences. After the video was shown there were some questions about the app from the professionals. The majority of participants said that they found the video difficult to follow and would have preferred slides with images of different actions being performed (i.e. the finger on the button) alongside verbal explanation.

Dictum was then shown to either pairs or single participants, depending on their availability as they were taking part in other activities also scheduled as part of the meeting. Reactions about the app were mixed. The first pair was composed of one man with severe aphasia and one of the Speech Therapists. The Speech Therapist helped the man understand by making examples relating to other activities they had undertaken together. The man was unsure what the difference was between applications he had installed on his mobile phone and iPad and the app being developed in this thesis. It was explained to him that it would work in a similar fashion to the ones he had but the objective was to increase speed and ease. He answered that he is more comfortable using the set of sentences made by his wife. He asked whether the app would only work to generate sentences that would be spoken by the app or if the generated sentences could be used in written communications, or interact with other apps, and if for example the app could be used to order takeaway by sending a text message.

One woman was enthusiastic about the app and said that she liked the idea of

words changing based on the previous word. She also liked the idea that she could generate whole sentences using even a single word. She had no advice about what to add or take out from the application.

The next woman had no questions about the app and no feedback to give. Whilst she understood the purpose of the app and what was being asked of her it was unclear whether she understood what she could ask and if she could formulate opinions about the app.

One woman was particularly helpful and constantly came back to give further feedback about the app. She had a slight difficulty with remembering words but she understood everything and had developed the habit of taking extensive notes on her thoughts and impressions about the app. Her comments were detailed and even technical; in particular she asked that navigation history should also be represented graphically on the screen as it is sometimes hard to keep track of previous actions on the app. She asked that words should interact with each other beyond simple verb noun relationships as the prototype does. In practice if there is a relationship between each noun this should be exploited by the application. This comment was particularly useful because it matches the design of the theoretical NLG system that was developed as part of the thesis, where the relationship between nouns is checked against WordNet to form more appropriate sentences. The user should also be able to enforce this relationship by connecting words manually. This participant also suggested the app should listen to speech to give appropriate selection of words. Her final suggestion was that a guide be included in the app to show its use.

9.2.3 Feedback from Speech and Language Therapists Present at the Meeting

Following the meeting at Dundee University, the video shown to potential users and professionals was also shown to other professionals by one SLT who was present at the meeting. The SLTs met to watch the video and had a brainstorming session to discuss feedback and impressions of the app.

Professionals particularly liked the idea of Semantic Networking and judged it to be a powerful tool to improve communication. They suggested that the app show the sentence selected by the user more clearly. The group also noted some potential issues for those with visual impairments; they suggested a setting for a higher visual contrast. Some also suggested the app be more visually appealing, however they

did not provide further suggestions on how to achieve this. The group of professionals suggested the re-introduction of a keyboard to slightly modify the structure of generated sentences. In my opinion, this issue should be further investigated in future work as it has the potential to be a bottleneck and could defeat the purpose of delegating sentence generation to the app. Another suggestion was that the app could also convey messages before the sentence generation is requested. The SLTs did not give any examples but presumably what they thought of is a scenario where the user selects the words they want and they can make the app speak to them to save time. The group asked what happens if the word the user is looking for is not in the app vocabulary. Currently, Dictum does not cater for this scenario. The group's final suggestion was to investigate the potential of this application as a therapeutic tool, however they did not provide any details as to how this investigation should be conducted.

9.3 Quantitative Evaluation

9.3.1 Overview of GridPlayer

This paragraph will give a small overview of the GridPlayer app, When the user first starts the app he is presented with a choice of different screens, the main screens are Symbol Talker A and symbol Talker B. Symbol Talker A is used to form basic sentences and has a reduced vocabulary, the number of elements on the screen is also lower than Symbol Talker B. An image of the categories for Symbol Talker A is shown below. This overview and the subsequent evaluation will primarily use Symbol Talker B as it offers a more complete vocabulary and it was used during the evaluation of Dictum.

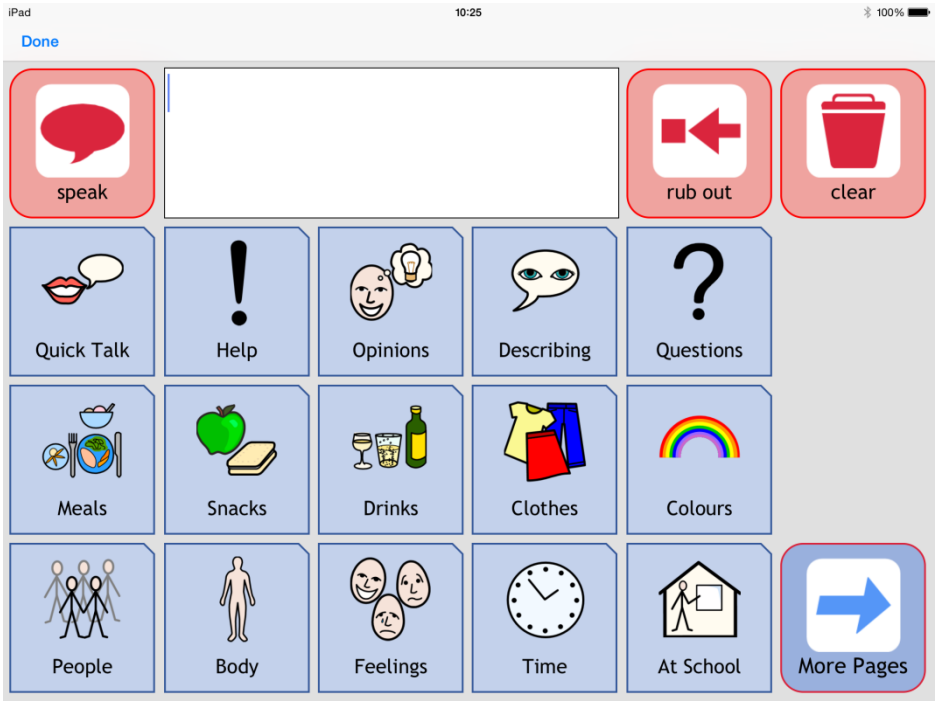


Image 13 Main Page for Sym Talker A

The image below shows the words and phrases given to the user when he/she selects the category "Meals", in Symbol Talker A.

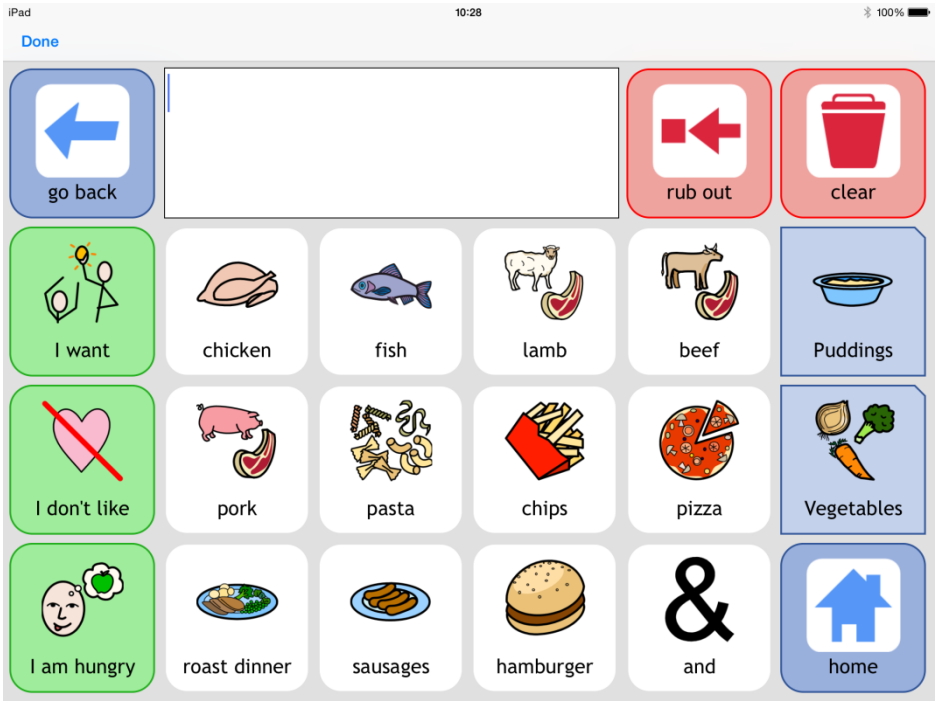


Image 14 The Meals Page of Symbol Talker A

Image 15 shows the main screen of Symbol Talker B. Symbol Talker B has a bigger grid and offers a larger vocabulary.

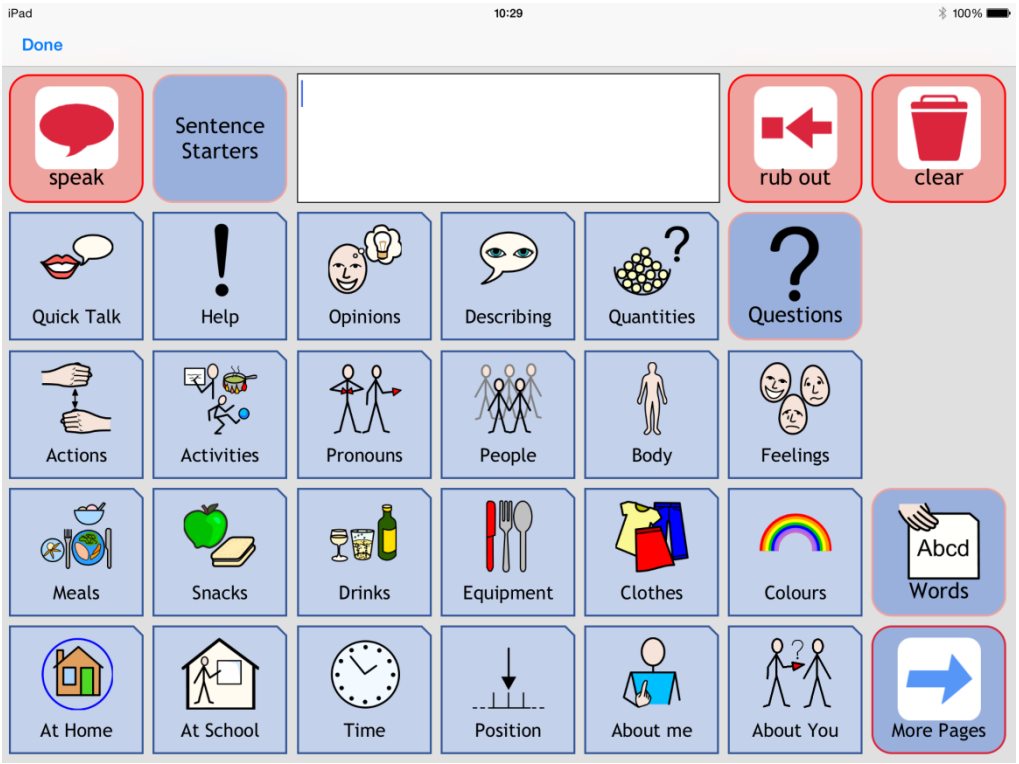


Image 15 Page 1 of Symbol Talker B Menu

Image 16 shows the selection of words offered when the user selects "Sentence Starters" from the main page. This page is shown as it is required to form request sentences like those generated by Dictum. For example requests for objects or requests to perform some activity would be generated by going through this screen.

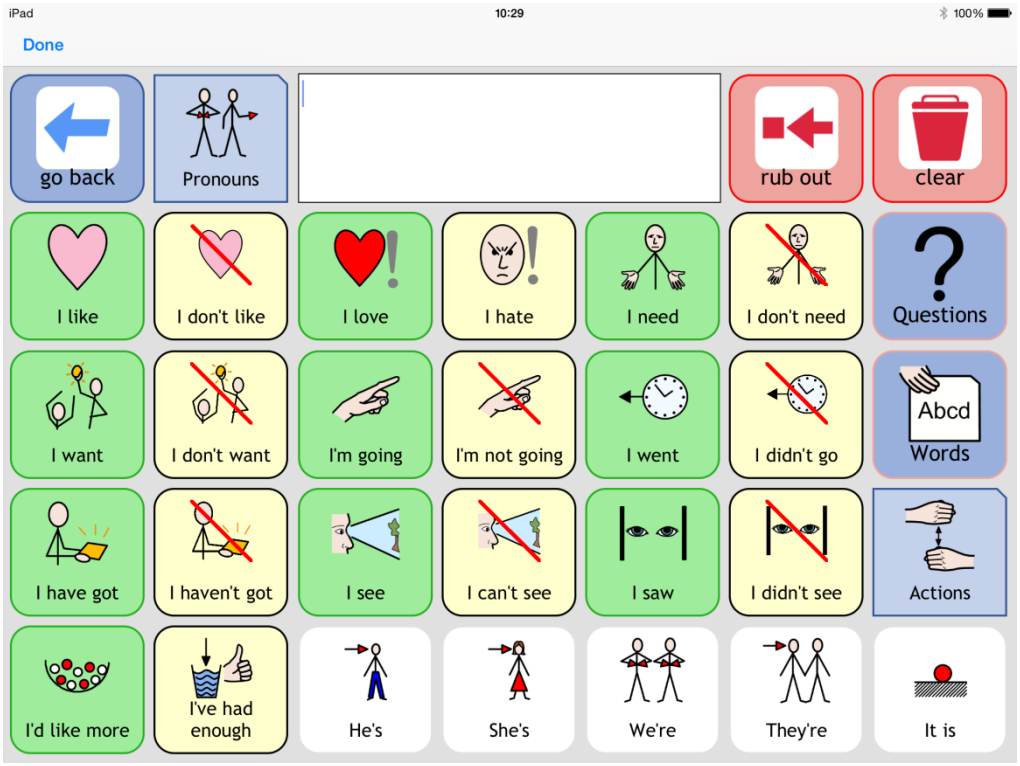


Image 16 The Sentence Starters Page

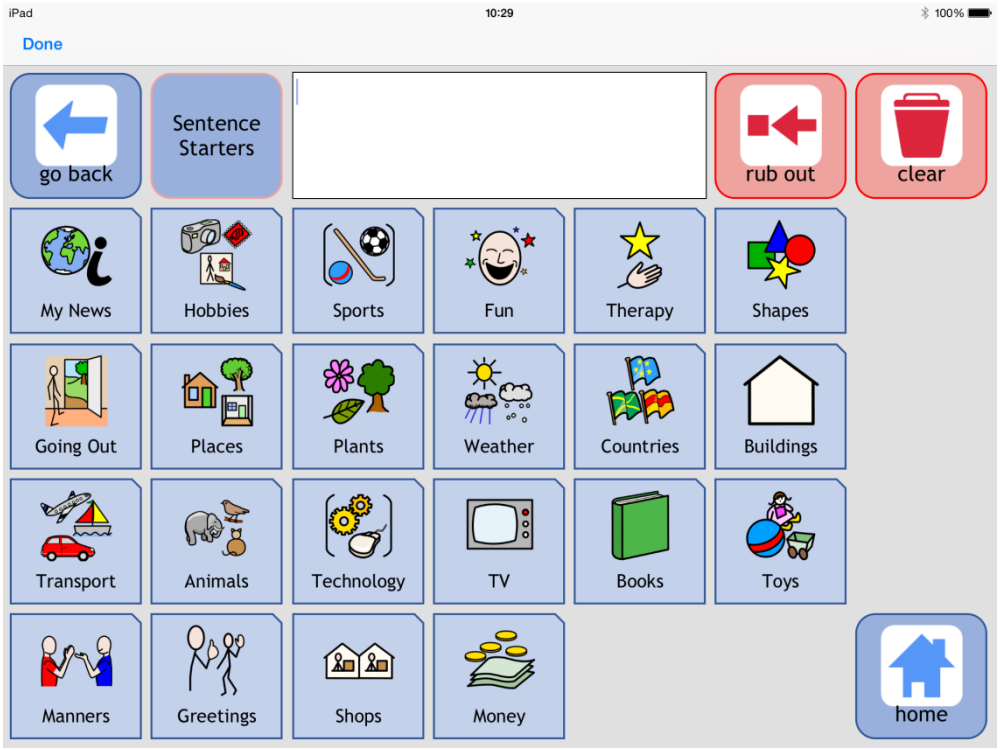


Image 17 The Second Page of the Sym Talker B Main Menu

Image 17 shows the second page of the Symbol Talker B menu. As mentioned previously in this thesis this is one of the bottlenecks of grid based AAC apps. The

user requires some time to find the appropriate category and the desired word within that category.

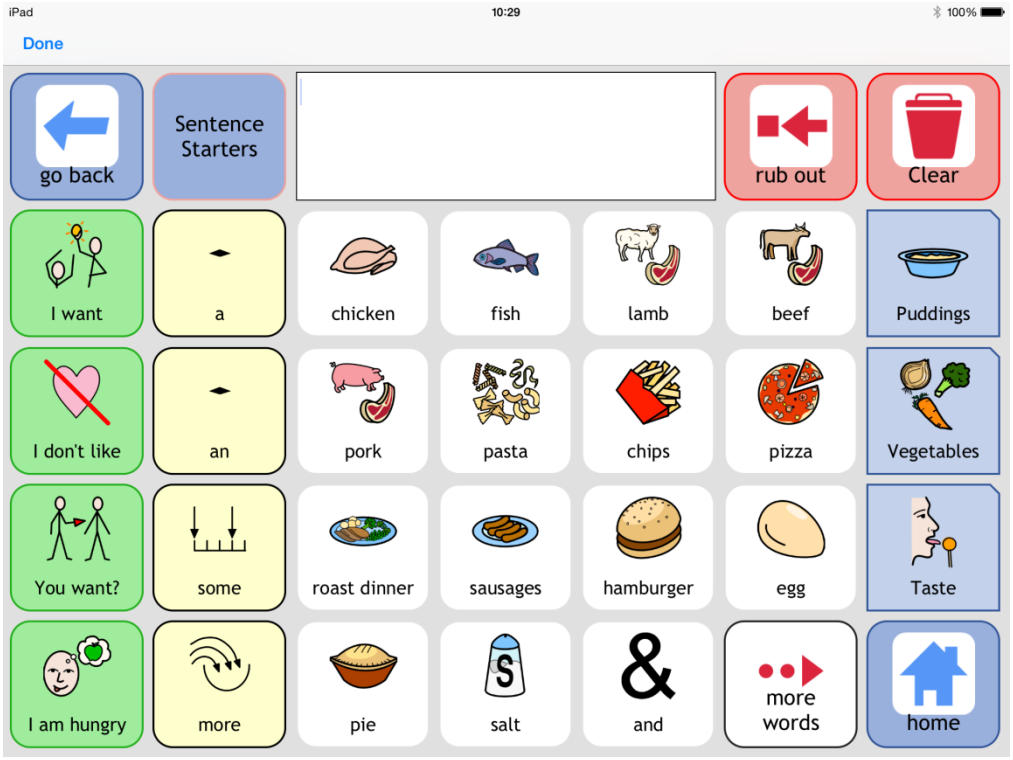


Image 18 The "Meals" Page of Symbol Talker B

Image 18 and Image 19 show the words available in GridPlayer when the user selects the category "Meals" in Symbol Talker B.

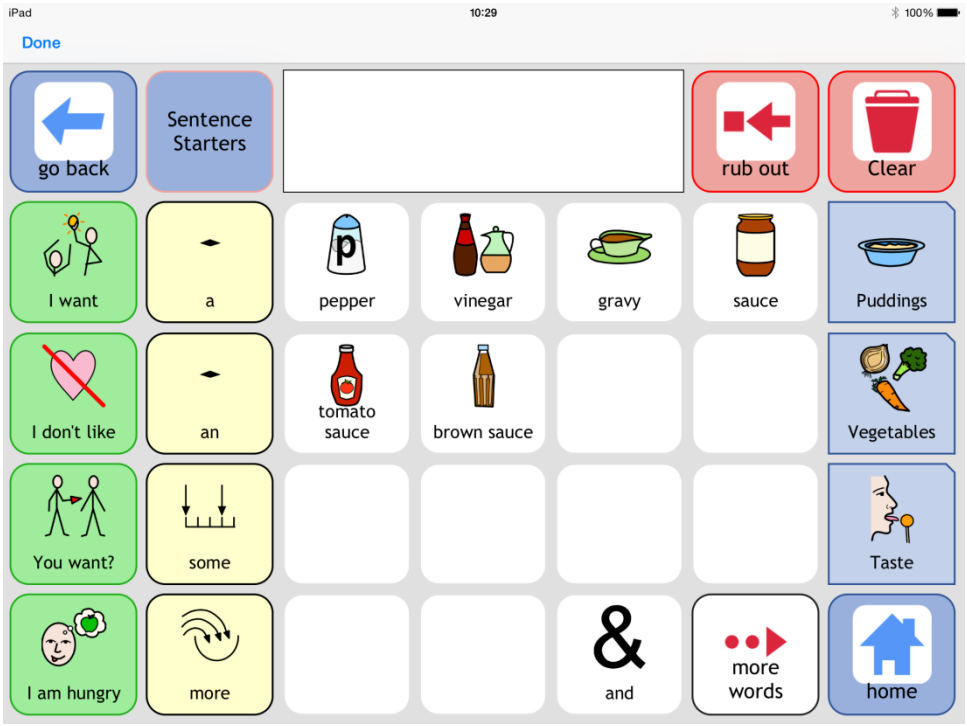


Image 19 The Second Page of the "Meals" Section for Symbol Talker B.

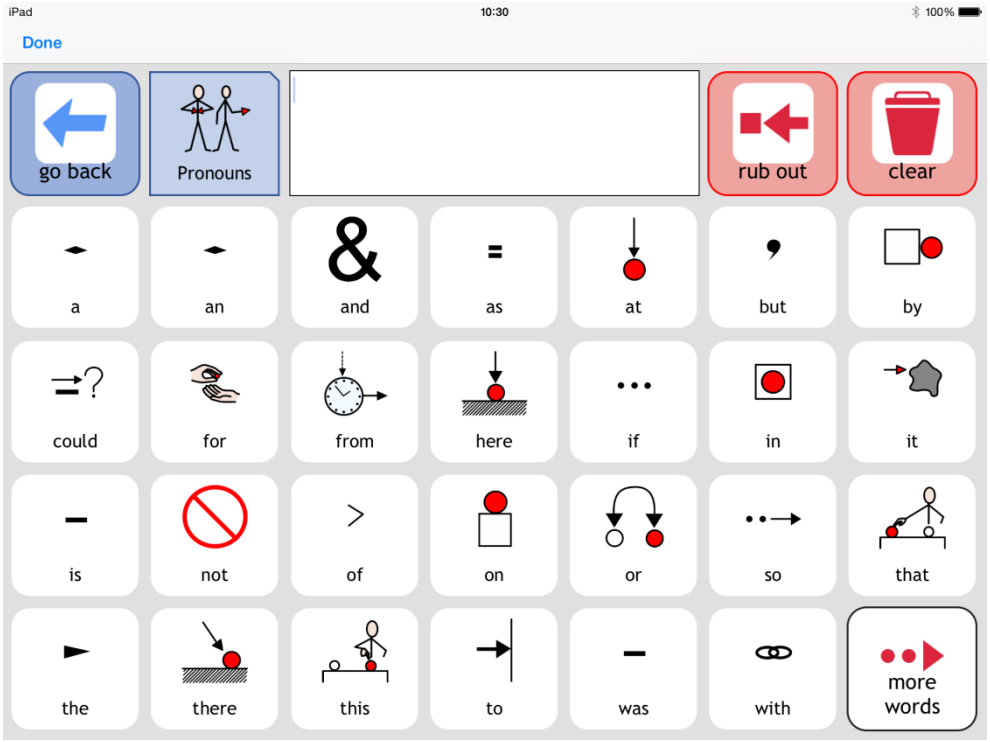


Image 20 The "Words" Page

The "Words" page shows articles and prepositions.

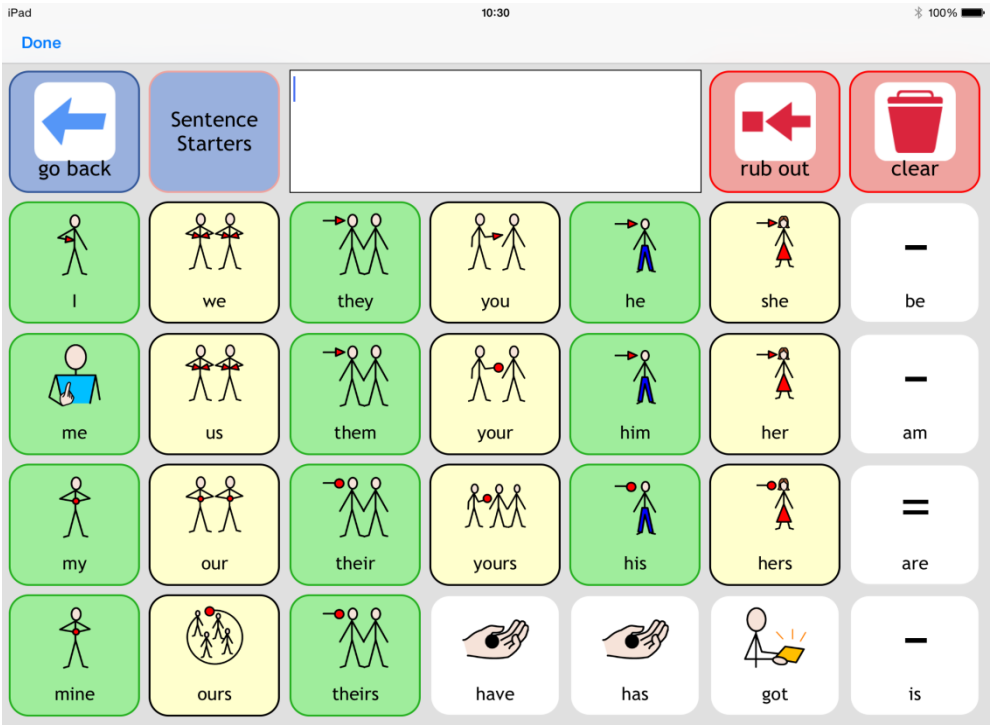


Image 21 The "Pronouns" Page

9.3.2 Overview of Verbally

The interface of Verbally and its interaction with users are different from those of GridPlayer, words are not organised hierarchically but rather there is one screen where the user can select two tabs. The first tab shows words and a keyboard and the second tab contains useful phrases. Unlike GridPlayer, Verbally uses a keyboard to enable the user to insert sentences, the keyboard works in the same way as a standard Android OS keyboard: when the user starts to type he/she is presented with predictions. Verbally also provides bigram predictions in an effort to save time. Unlike Grid Player and Dictum, there is no risk of a word not being present in the vocabulary. Image 22 shows the interface of Verbally and a selection of words the user is presented with. The most common ones are articles and pronouns.

Image 22 The Verbally Interface when the "Words" Tab is Selected

Image 23 The Verbally Interface when the"Phrases" Section is Selected

9.3.3 Evaluation of Dictum using the Keystroke Saving Model

9.3.3.1 The Keystroke Saving Model

The first part of the evaluation was conducted by calculating the Keystroke Saving (Trnka and McCoy, 2008) of Dictum against two competitor apps: Verbally and GridPlayer. Verbally and GridPlayer were chosen because they were known and used by Speech Professionals and potential users encountered during the course of this work. Equation 9 shows how Keystroke Saving is calculated. AAC apps that use prediction mechanisms are frequently evaluated by comparing the number of keys required by applications being compared (Higginbotham et al., 2009) (Trnka et al., 2007).

For each topic in Dictum, a set of sentences was fashioned to compare the number of keys required to form sentences in each application. The number of keys required is shown in the diagram below. For each sentence there are three kinds of symbols in the diagram: the red squares are for functional buttons such as the button to speak or the button to generate sentences, the blue squares indicate the selection of words, light blue in the case of single keystrokes or dark blue in the case of whole words- The arrows indicate the flow of sentence formation, red arrows indicate that the screen has changed, blue arrows indicate that the screen has not changed.

$$KS = \frac{keys_{normal} - keys_{prediction}}{keys_{normal}} x100$$

Equation 9 Equation for Keystroke Saving

9.3.3.2 Choosing Sentences

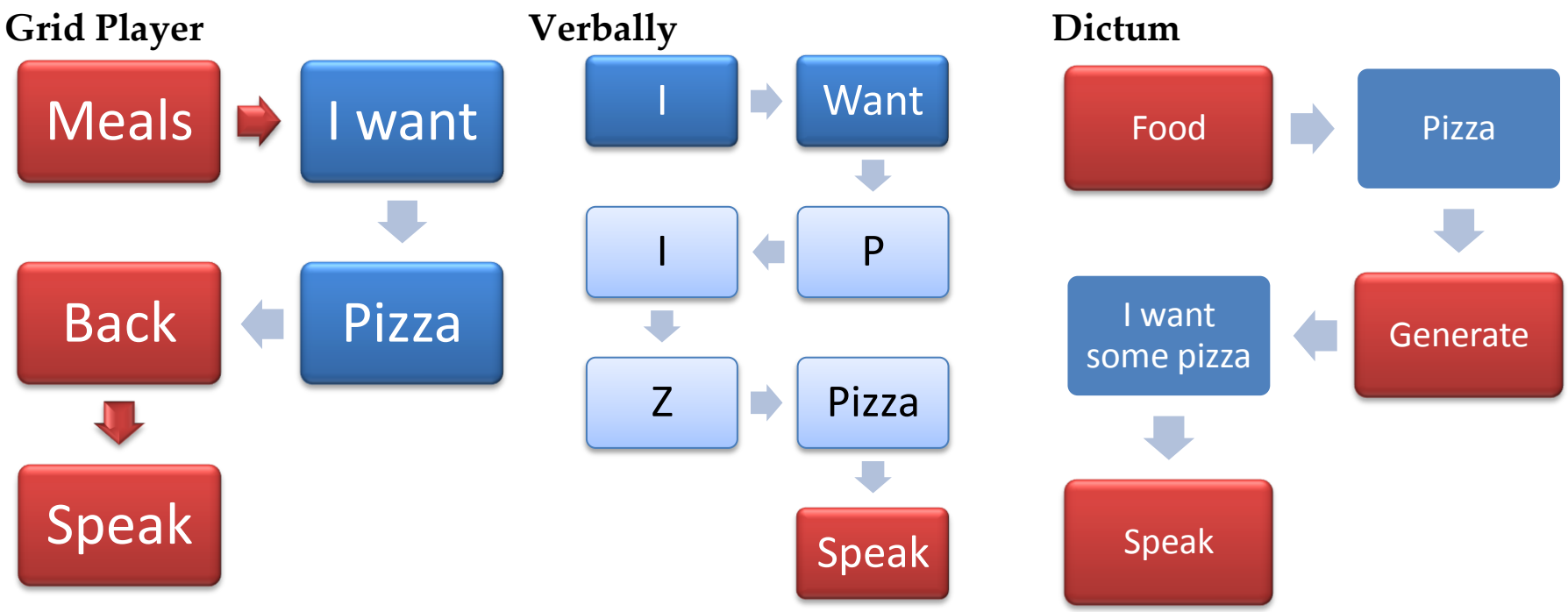
There exists no standard dataset for the evaluation of AAC application, and there is no agreement on how to evaluate AAC applications (Higginbotham et al., 2009). Most works measure Words Per Minute compared to other apps. Kristensson and Vertanen (Kristensson and Vertanen, 2012) cite an AAC dataset; most of the sentences in the dataset are related to health issues and physical conditions. This is presumably because the AAC dataset includes sentences taken from cardboard AAC booklets which are given to patients after having had a stroke and having lost the ability to speak.

The sentences used for evaluation were formed around vocabularies present in the three applications. Given that AAC applications are mostly used by users for expressing needs, for making requests and for forming simple sentences. The

evaluation sentences were formed by imagining common communication scenarios related to the four categories of Dictum (i.e. Food, Music, Sports, and Transport). The sentences were formed mainly by keeping in mind that sentences formed using AAC are usually short. Evaluation sentences also include information relating to when the request and action take place and with whom the user wants to take part in the activity.

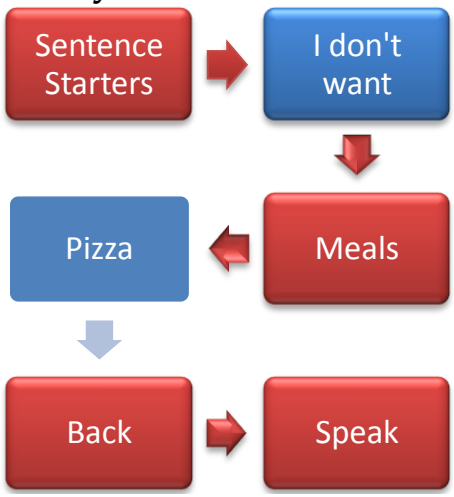
9.3.3.3 Measurement of Keystroke Saving using Food Category Sentences

1. I want pizza

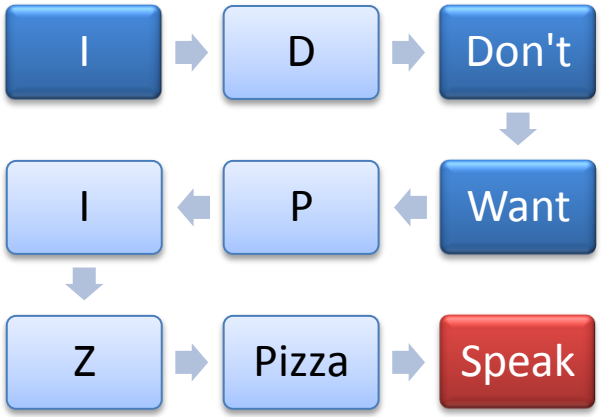


2. **I don't want pizza**

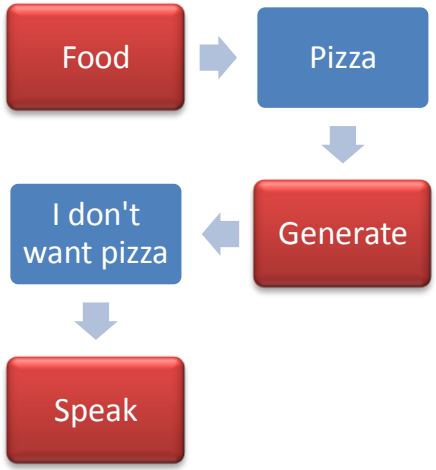
GridPlayer



Verbally

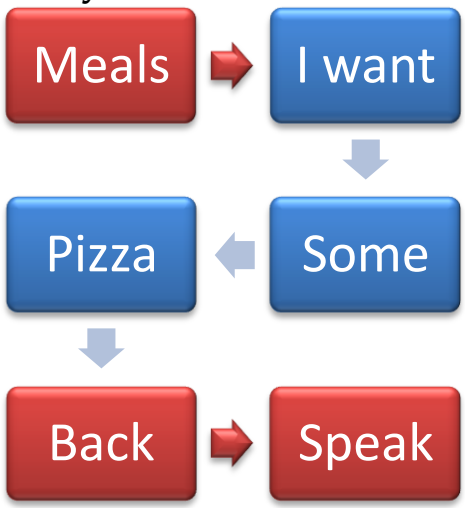


Dictum

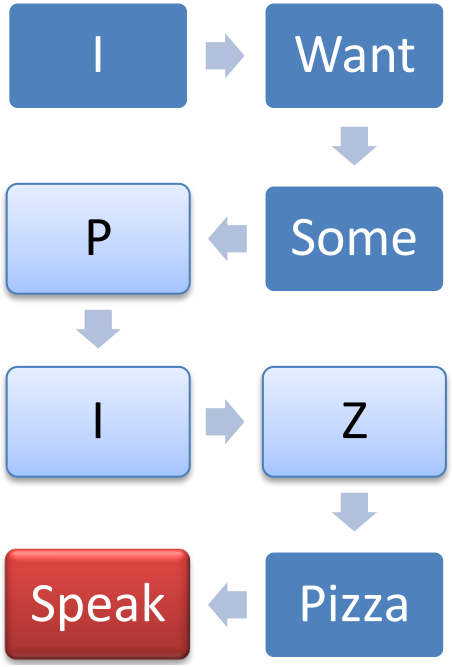


3. **I want some pizza**

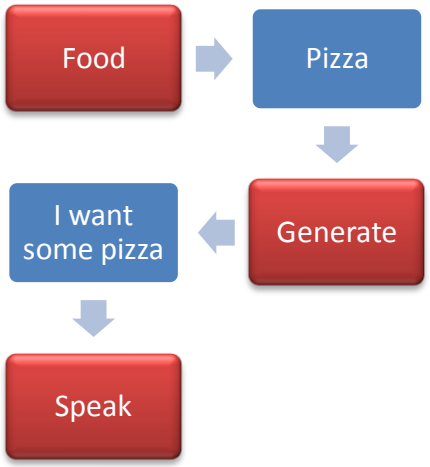
GridPlayer



Verbally



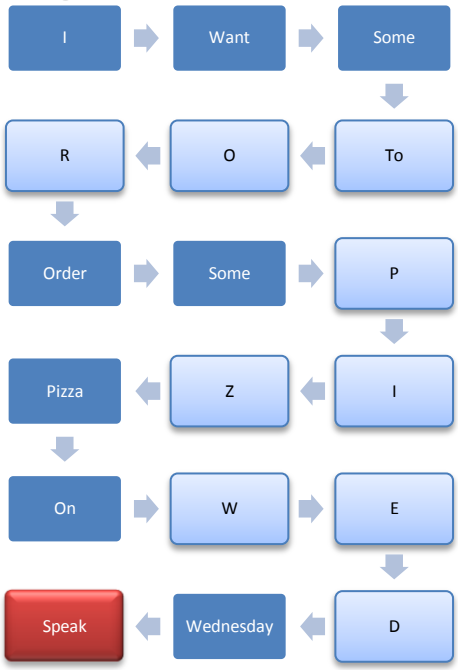
Dictum



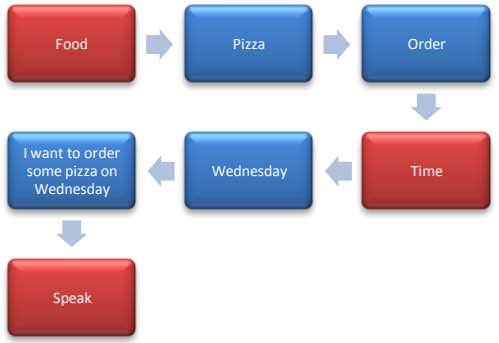
4. **I want to order some pizza on Wednesday**

GridPlayer

Verbally



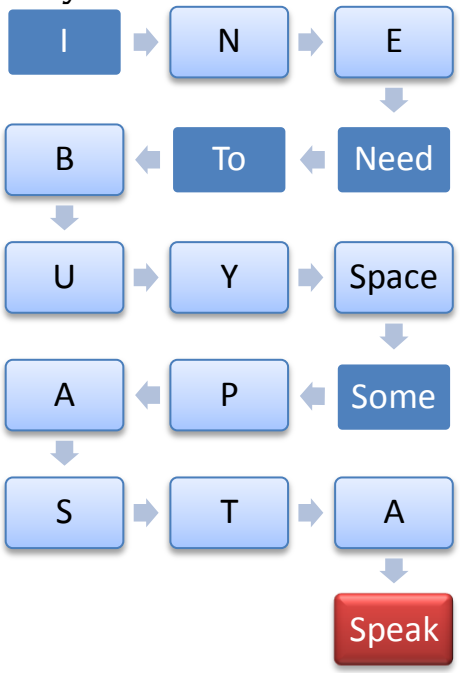
Dictum



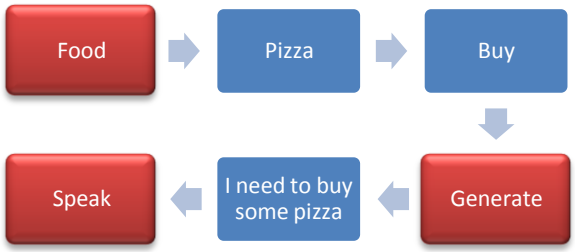
5. **I need to buy some pasta**

GridPlayer

Verbally



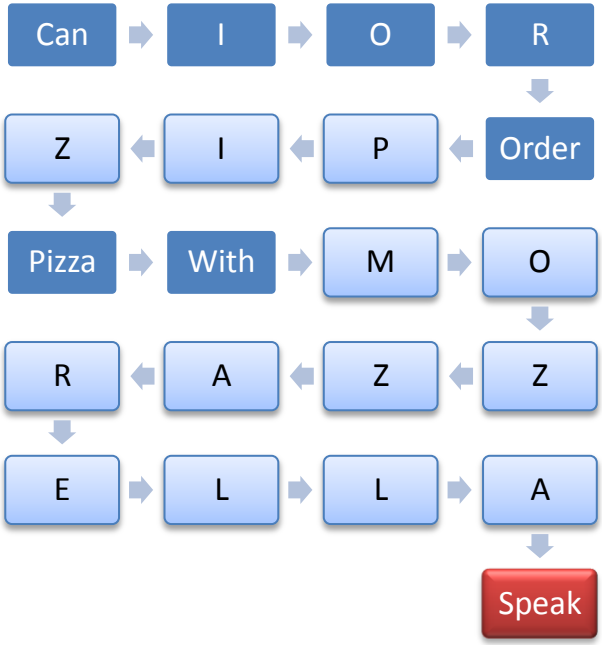
Dictum



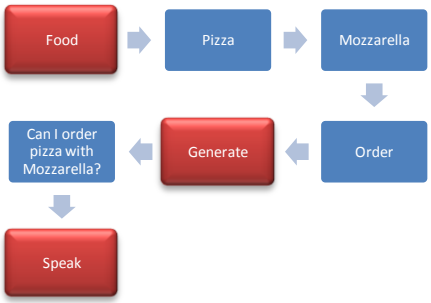
6. **Can I order pizza with mozzarella?**

GridPlayer

Verbally



Dictum

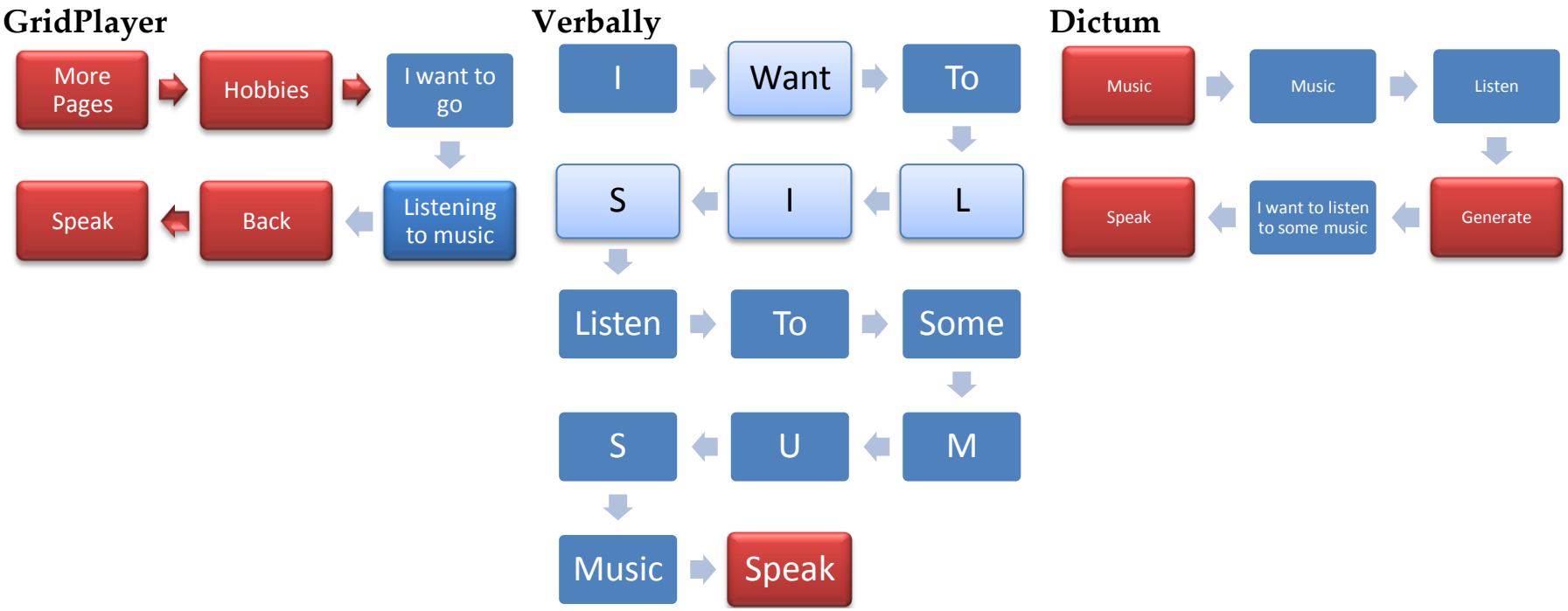


9.3.3.4 Measurement of Keystroke Saving using Music Category Sentences

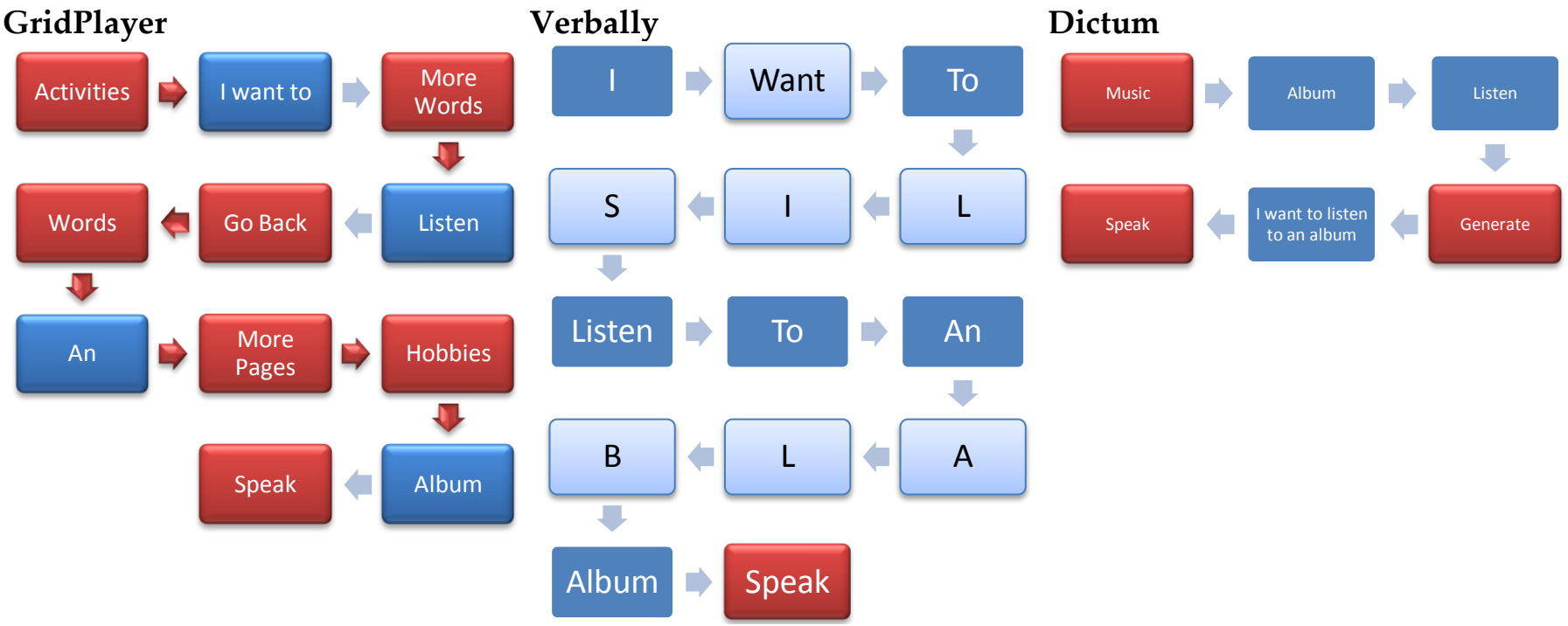
It is not possible to form music related sentences in GridPlayer, as most of the words required are not present in the vocabulary. However, because the "Hobbies" category contains words such as "iPod" and "Radio" it will be assumed that missing words are in the second page of the "Hobbies" category.

1. **I want to listen to some music**

   This sentence is a special case for GridPlayer as there is a button in the "Hobbies" category for the sentence "Listening to music"

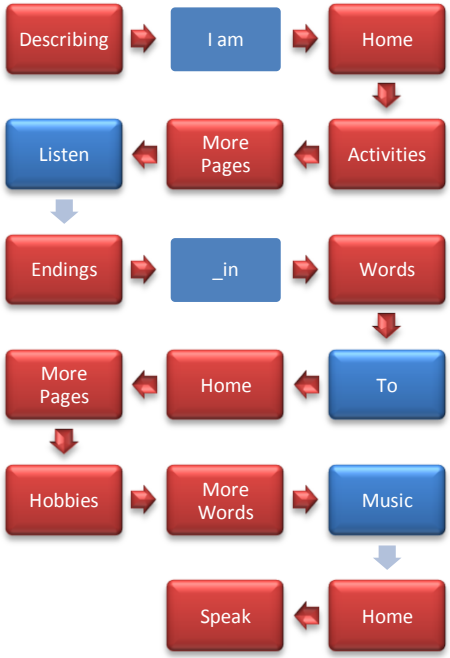


2. I want to listen to an album

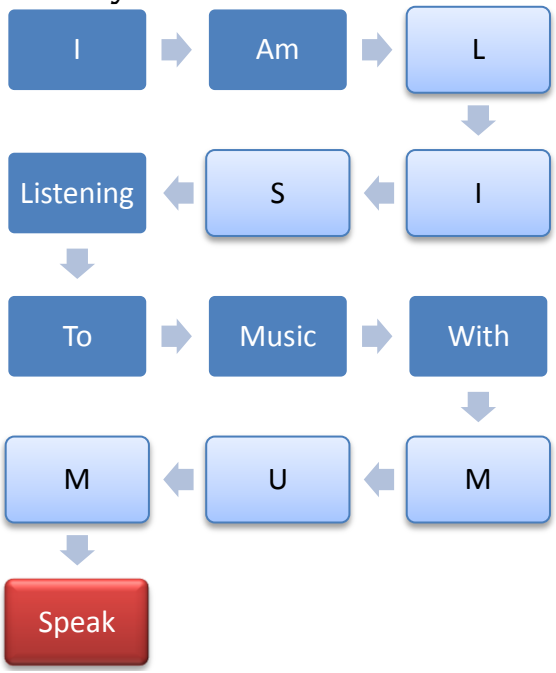


3. **I am listening to music with mum**

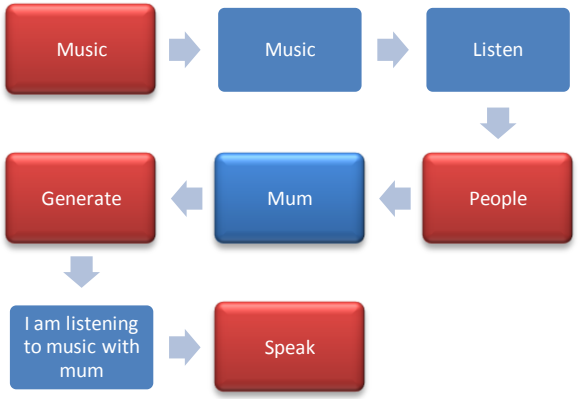
GridPlayer



Verbally

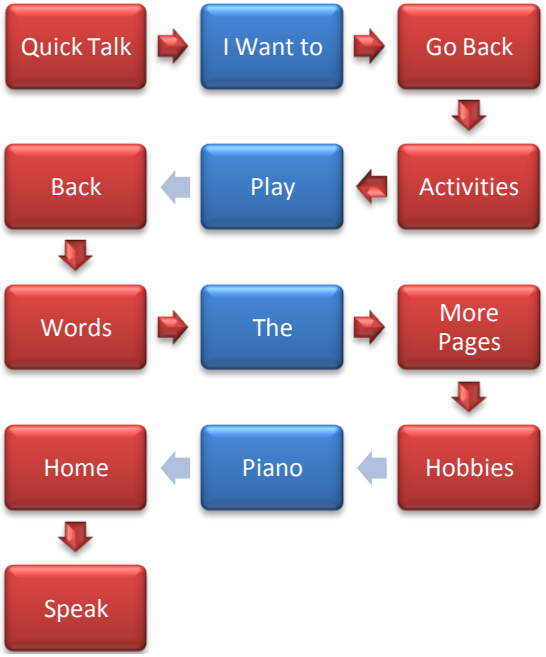


Dictum

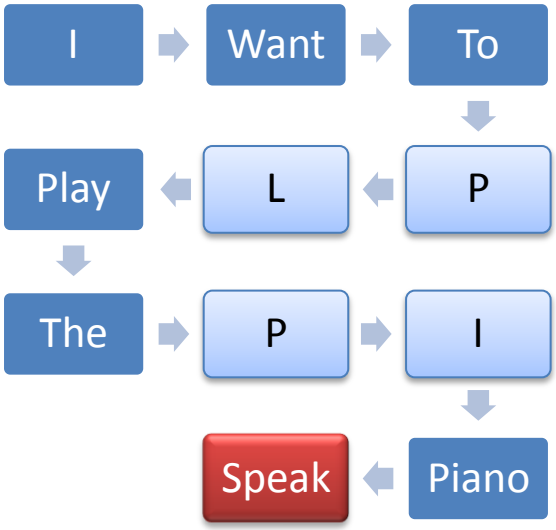


4. **I want to play the piano**

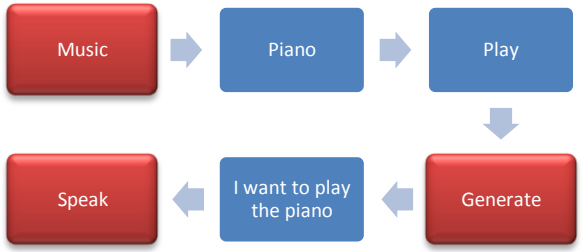
GridPlayer



Verbally



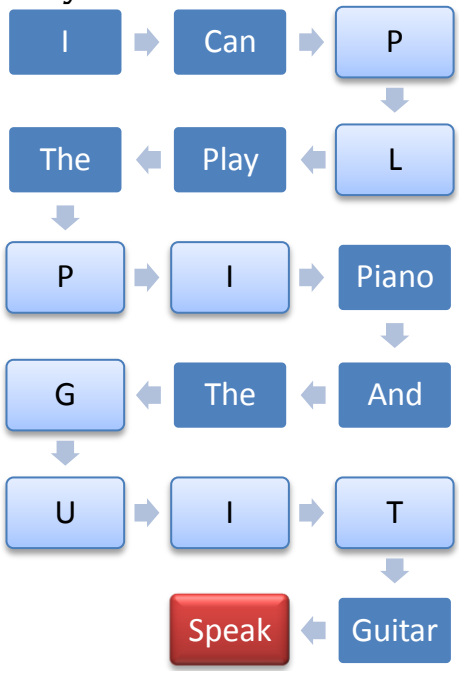
Dictum



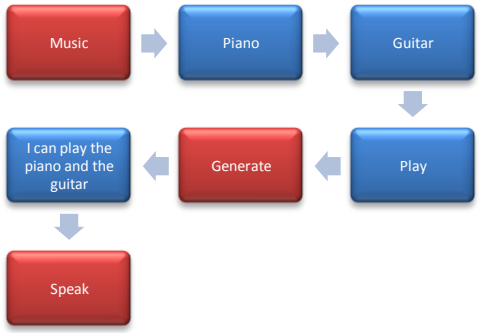
5. **I can play the piano and the guitar**

GridPlayer

Verbally

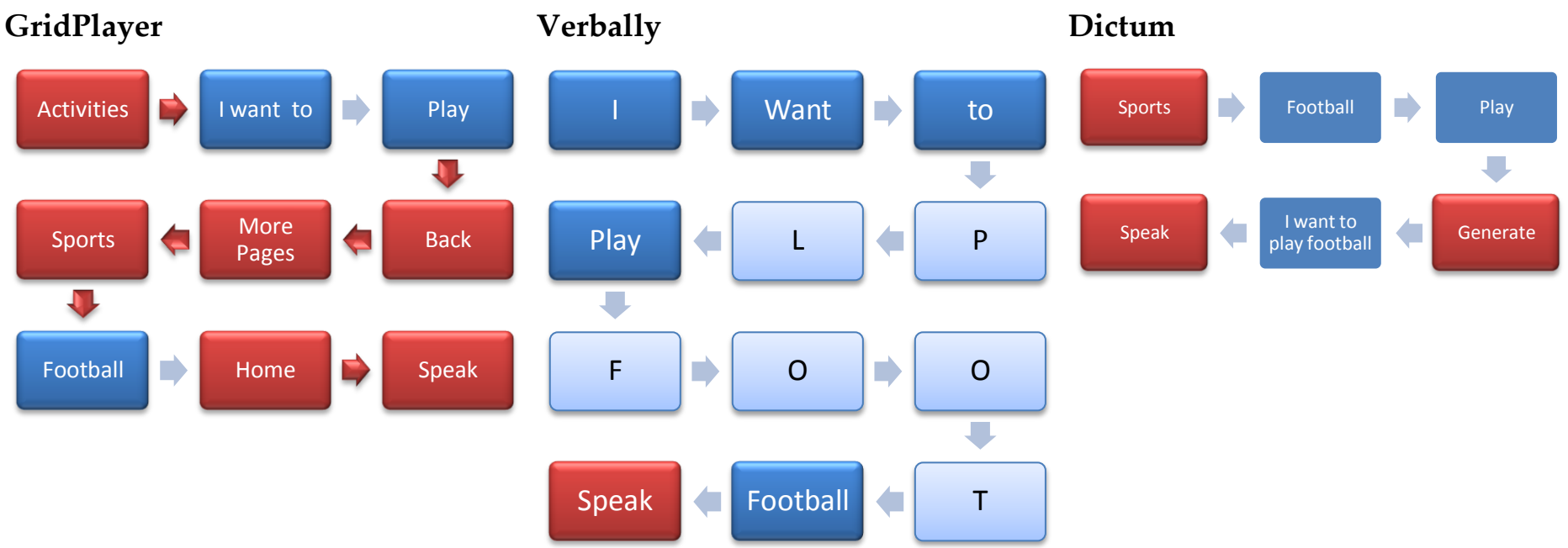


Dictum



9.3.3.5 Measurement of Keystroke Saving using Sport Category Sentences

1. I want to play football



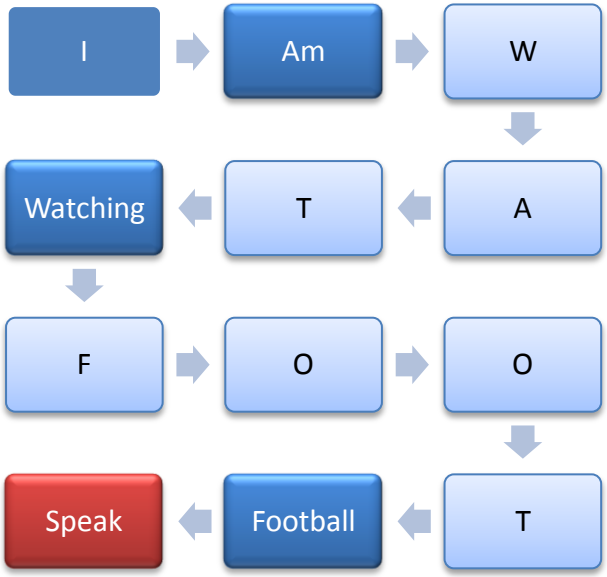
2. I am watching football

For this sentence GridPlayer gives the option to say: I watch football, to indicate that the person habitually watches football. There is no way for the person to say that they are currently watching football.

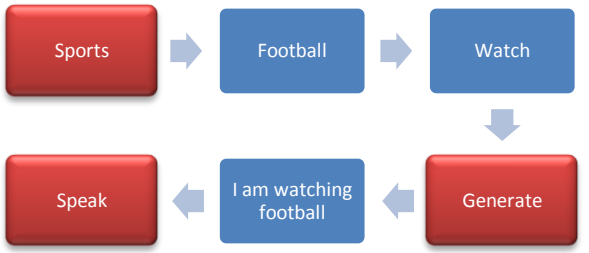
GridPlayer

NOT POSSIBLE IN APP

Verbally

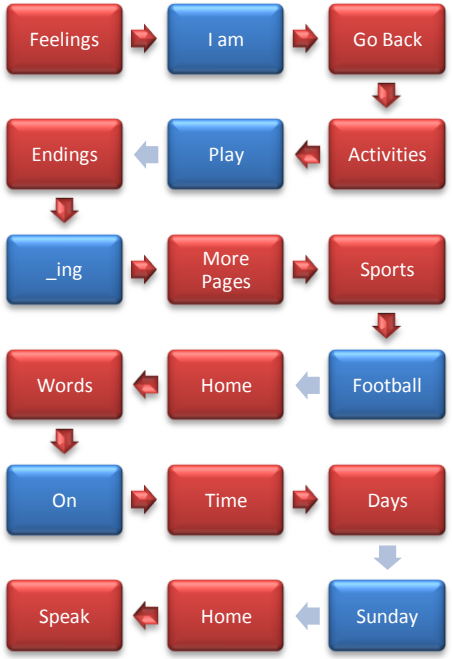


Dictum

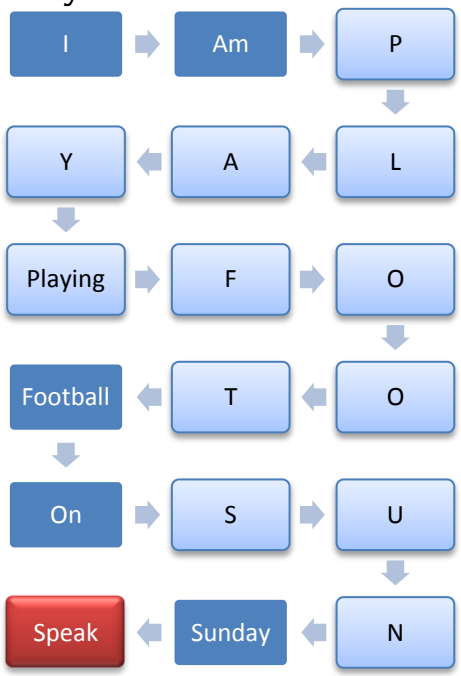


3. **I am playing football on Sunday**

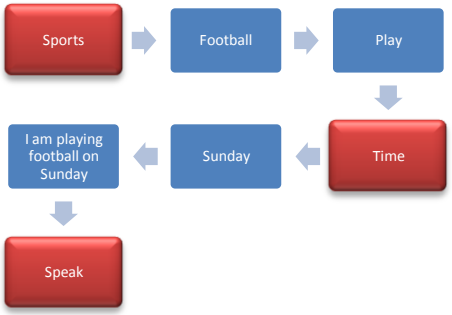
GridPlayer



Verbally

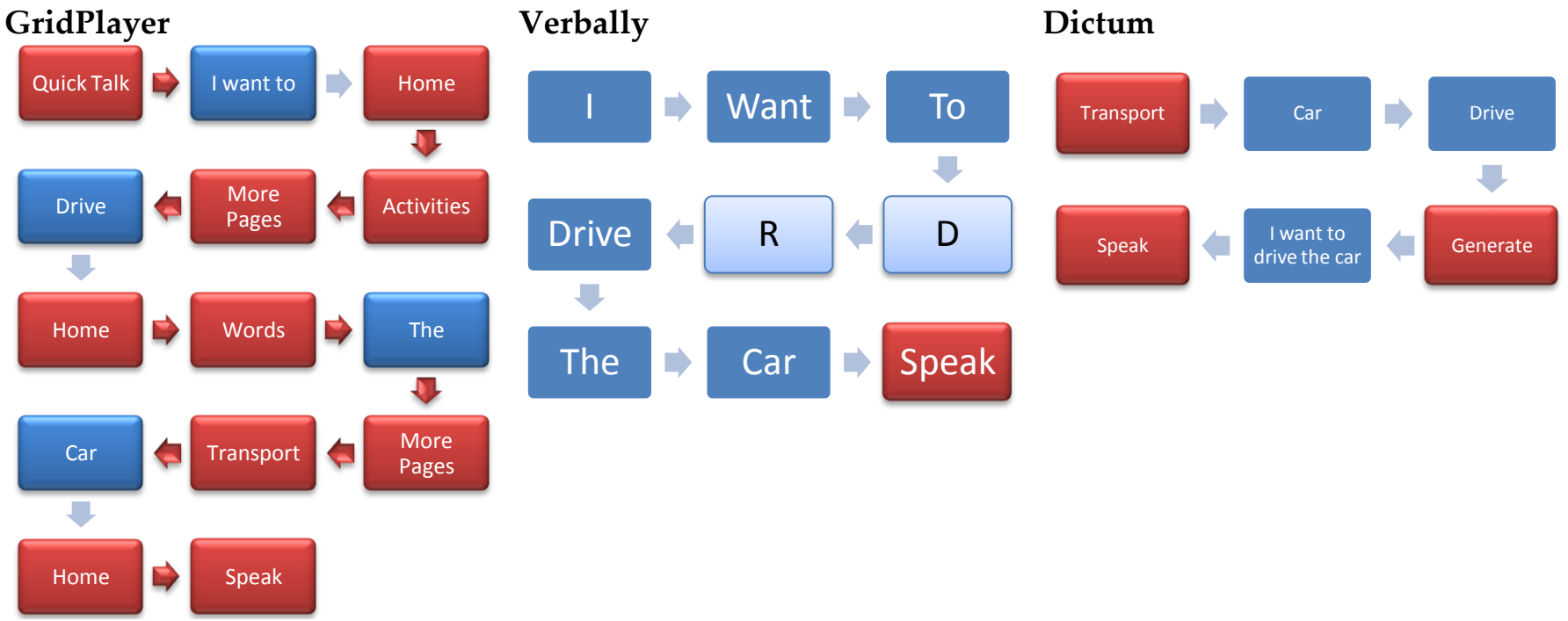


Dictum

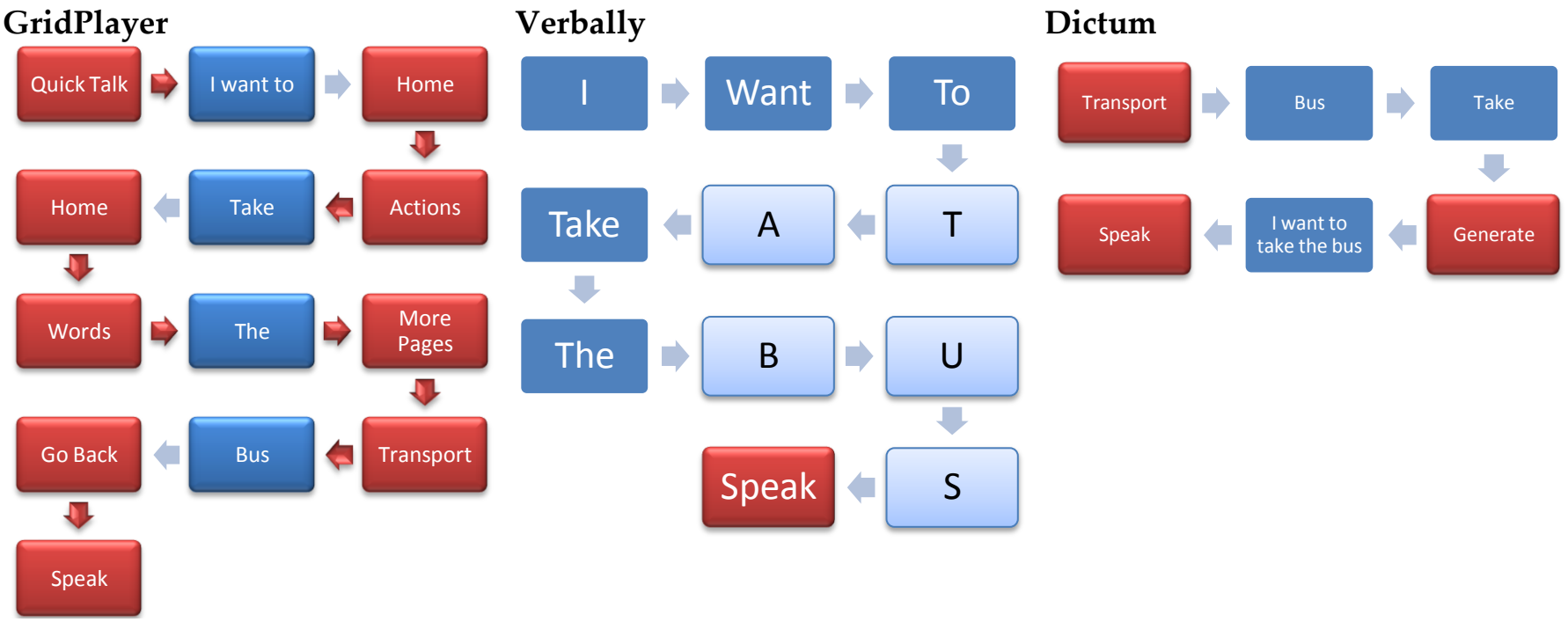


9.3.3.6 Measurement of Keystroke Saving using Transport Category Sentences

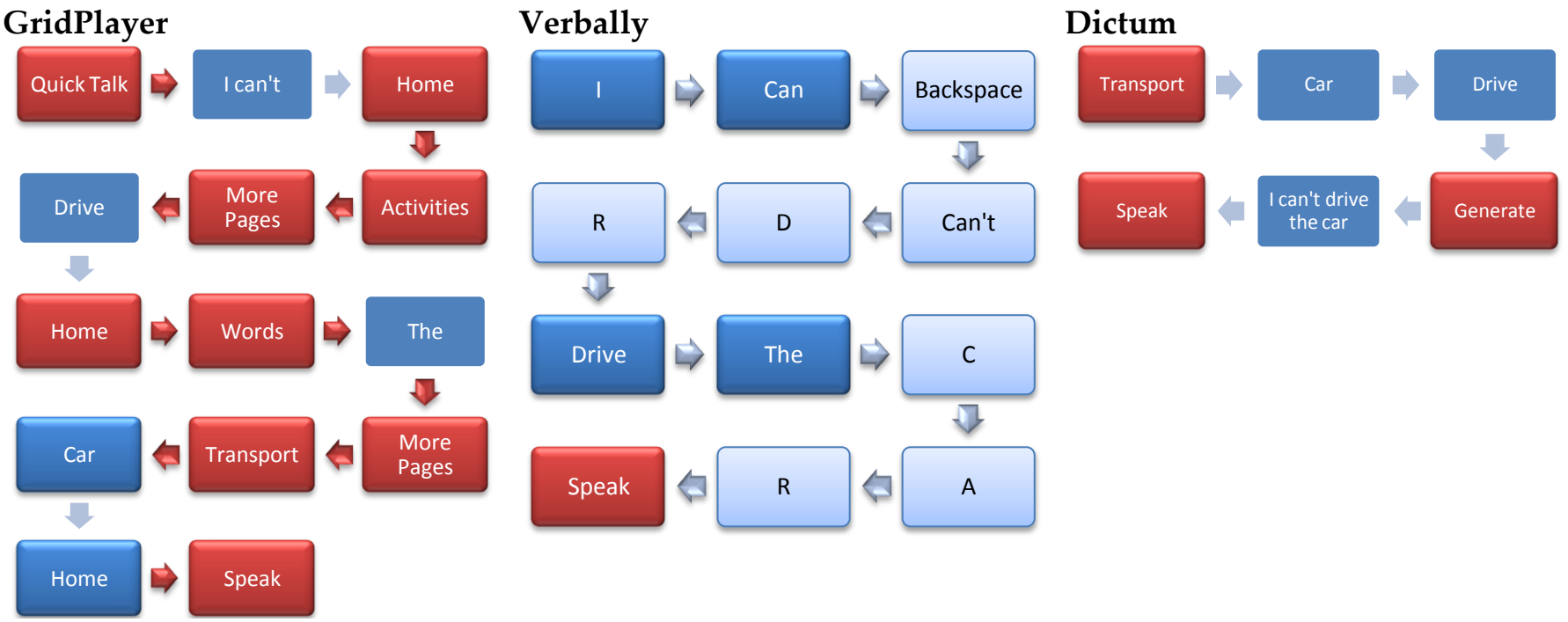
1. I want to drive the car



2. **I want to take the bus**



3. **I can't drive the car**



9.3.3.7 Results

Food Sentences - GridPlayer

Sentences	Number of taps	Number of screens	Dictum	KS %
I want pizza (3)	5	3	5	0%
I don't want pizza(4)	6	5	5	16%
I want some pizza(4)	6	2	5	16%
I want to order some pizza on Wednesday (8)	14	11	7	50%
I need to buy some pasta (6)	15	9	6	60%
Can I order Pizza with Mozzarella?(7[31])	22	16	7	68.1%

Table 4 Performance Comparison of Thesis App and GridPlayer for Food Sentences

Food Sentences – Verbally

Sentences	Number of taps	Number of screens	Dictum	KS%
I want pizza (3)	7	1	5	28.5%
I don't want pizza(4)	6	1	5	16%
I want some pizza(4)	8	1	5	37.5%
I want to order[32] some pizza on Wednesday (8)	18	1	7	61.1%
I need to buy some pasta(6)	16	1	6	62.5%
Can I order pizza with Mozzarella?(7)	21	1	7	66.6%

Table 5 Performance Comparison of Thesis App and Verbally for Food Sentences

[31] Question mark counts as a character, as to insert it in GridPlayer a special button needs to be pressed

[32]Assumed to be in More Pages of Activities Section.

Music Sentences - GridPlayer

Sentences	Number of taps	Number of screens	Dictum	KS %
I want to listen[33] to some Music (7)	6	4	6	0%
I want to listen to an Album (7)	11	8	6	40%
I am listening to music with mum(7)	17	14	8	52.9%
I want to play the piano (6)	13	9	6	53.8%
I can play the piano and the guitar (8)	20	13	7	65%

Table 6 Performance Comparison of Dictum and GridPlayer for Music Sentences

Music Sentences -Verbally

Sentences	Number of taps	Number of screens	Dictum	KS%
I want to listen to some Music (7)	14	1	6	57.1%
I want to listen[34] to an Album (7)	14	1	6	57.1%
I am listening to music with mum (7)	13	1	8	38.4%
I want to play the piano (6)	11	1	6	45.4%
I can play the piano and the guitar (8)	17	1	7	58.8%

Table 7 Performance Comparison of Dictum and Verbally for Music Sentences

[33] Assumed to be in" More Pages" of "Activities" section
[34] Assumed to be in "Activities" section of GridPlayer

Sports Sentences-GridPlayer

Sentences	Number of Taps	Number of Screens	Dictum	KS%
I want to play football (5)	9	7	6	33.3%
I am watching football	NOT POSSIBLE	NOT POSSIBLE	NOT POSSIBLE	NOT POSSIBLE
I am playing football on Sunday (6)	18	13	7	56.2%

Table 8 Performance Comparison of Dictum and GridPlayer for Sports Sentences

Sports Sentences-Verbally

Sentences	Number of Taps	Number of Screens	Dictum	KS%
I want to play football (5)	12	1	6	50%
I am watching football (4)	12	1	6	50%
I am playing football on Sunday (6)	18	1	7	61.1%

Table 9 Performance Comparison of Dictum and Verbally for Sports Sentences

Transport sentences-GridPlayer

Sentences	Number of Taps	Number of Screens	Dictum	KS %
I want to drive the car (6)	14	10	6	57.1%
I want to take the bus (6)	13	10	6	53,8%
I can't drive the car (5)	14	11	6	57.1%

Table 10 Performance Comparison of Dictum and GridPlayer for Transport Sentences

Transport sentences-Verbally

Sentences	Number of Taps	Number of Screens	Dictum	KS %
I want to drive the car (6)	9	1	6	33%
I want to take the bus (6)	11	1	6	45.4%
I can't drive the car (5)	12	1	6	50%

Table 11 Performance Comparison of Dictum and Verbally for Transport Sentences

Average Keystroke Saving % of Dictum compared to Verbally and GridPlayer

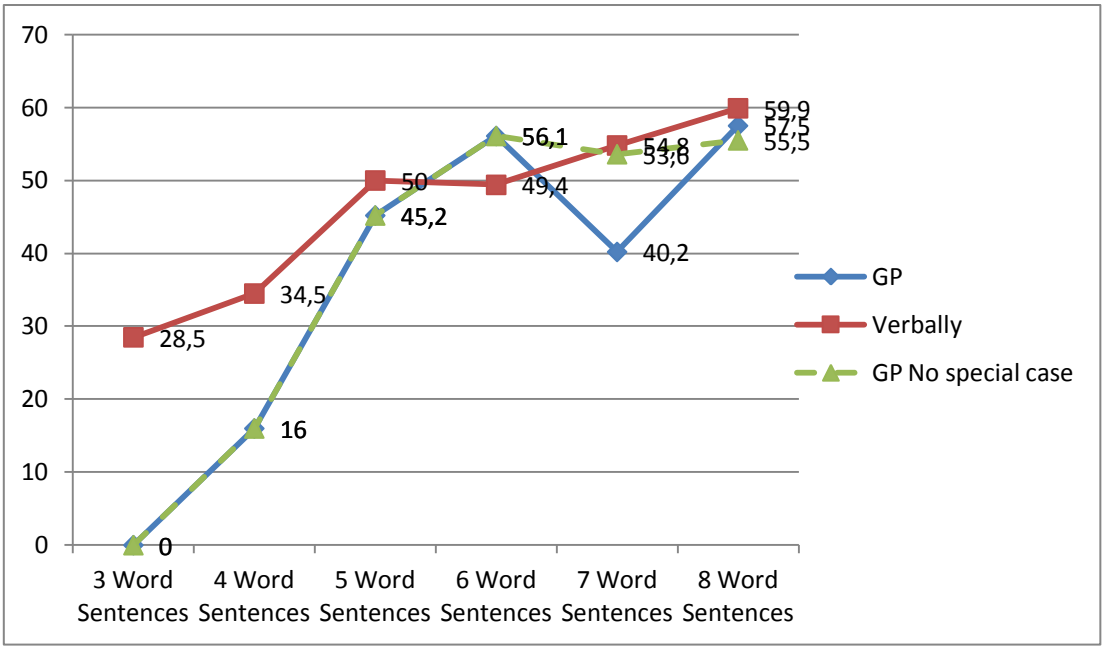
Sentence Length	AVG. Saving compared to GridPlayer	Avg. Saving compared to Verbally
3	0%	28.5%
4	16%	34.5%
5	45.2%	50%
6	56.1%	49.4%
7	40.2%	54.8%
8	57.5%	59.9%

Table 12 Average Keystroke Saving for GridPlayer and Verbally Compared to Dictum

The results of this evaluation were obtained by comparing Dictum against GridPlayer and Verbally, however results could be generalised to all Grid Based AAC apps available today. As mentioned in the theoretical Chapter, the most used means of obtaining higher speed on most Grid Based apps is to customise application to suit the user's needs. In most cases this has also the disadvantage of requiring assistance. Even the apps which contain some amount of text prediction are limited to bigram and do not offer contextual help to the user. The results of this evaluation show that by using a Semantic Network of related words and delegating the responsibility of sentence formation to the application it is possible to achieve higher speeds even without the need for the user to customise the app and involve communication partners in the process. In future work it would be interesting to investigate the impact of customisation on Dictum. It would also be interesting to investigate the impact of using whole utterances (buttons containing whole sentences) on the Dictum system.

One of the proposed research questions at the start of this thesis asked whether a text entry system like Dictum, which relies solely on whole words and has no keyboard input could effectively work for users. This evaluation has shown that whole word text entry is possible albeit there will be some cases where the user is unable to find the desired words in the vocabulary and other ways will have to be found to insert new words, and to adapt the Semantic Network in real time.

In sentences used here for evaluation, Dictum proved to be the fastest in most cases. In one case GridPlayer had one button to generate the sentence "listening to music" which was assumed to have the same meaning as "I am listening to music". The advantage which both Verbally and the thesis app had over GridPlayer was that sentence formation was possible on a single screen whereas the constant changing of screens in GridPlayer proves to be a considerable mental overload when the app is unfamiliar. Considerable effort on the part of the user is also required when he/she wants to form sentences in a different tense. The example of *I am watching football* is one, where the sentence is in present continuous; the user has to navigate through multiple screens to access the "Endings" tab in GridPlayer.



Graph 10 Key Stroke Saving

The graph shows the average saving of keystrokes using Dictum. As can be seen from the graph the advantage becomes more substantial as the number of words in the sentence increases, conversely for sentences that are shorter than three words there is no advantage and then as the sentence length increases so does the Keystroke Saving.

For completeness the dotted line on the graph shows the Keystroke Saving percentage excluding the case where GridPlayer had a single button to generate the same sentence.

For sentences greater than 8 words it can be assumed that the saving will stabilise at around 50%.

9.3.4 Discussion on Strength and Weaknesses of the AAC Application Developed in this Work Based on the Results of Keystroke Saving Evaluation

Using NLP/NLG techniques has the potential to improve text entry rates and ease of use of AAC mobile applications dramatically. However, communicative scenarios are so diverse that the current model can only cater for some cases and not others. The system performs greatly when the user expresses needs and where activities are driven by objects themselves instead of the activity.

For example in the case of food the system is very fast because there are common actions, mostly eat and want with every noun that has to do with food. The system is slower in cases where the sentence is driven by the verb (the action) instead of the object. For example, for the category "Transport", it is more likely to find different verbs associated with different means of transport. It can happen that it takes more time to find the desired verb after having selected a noun. It can also happen that a noun is found only after having selected a related noun. This issue could be solved by the inclusion of a search feature, where the user hints at the desired word and looks at it in the system. But this issue is recurrent throughout this Chapter because having the user hint at the desired word increases the number of keys required and could potentially outweigh the benefits of the current implementation. This requires further investigation in future work.

The prediction model performs well in presenting users with relevant verbs. There are some cases where the verbs make sense grammatically but not conversationally. This is mostly due to the fact that whilst the corpora that were used provide a good model of the English language and the metadata contained in them make it easier to classify them, this representation of language sometimes does not translate well to conversational scenarios.

There is also a need to include phatic expressions in the system and also to include pragmatics so that the user can use small talk and exchange pleasantries with his/her communication partner.

A further improvement of Dictum would be the inclusion of whole utterances, although the impact of such inclusion should be properly investigated.

There are some instances where GridPlayer performs better than Dictum. For example in cases where the user talks about numbers and quantities and some scenarios related to transport such as purchasing return tickets for buses or trains, or when the user talks about currency and basic arithmetic. In a future implementation of the application the user has to be given whole utterances to speed up the process and reduce the number of keystrokes.

9.3.5 Introduction to User Modelling Evaluation

The second part of the evaluation of Dictum was conducted via the use of user modelling techniques. User modelling evaluation techniques simulate and try to predict the behaviour of users when using a particular piece of software or device

(Koester and Levine, 1997). The pioneering work in user modelling is the book "The Psychology of Human Interaction" by Card, Moran and Newell (Card et al., 1983). In the book, they set out the foundations for many subsequent works, in particular the GOMS model and the Keystroke Level Model (KLM) (Card et al., 1980) which is the simplest form of GOMS model.

All of these techniques have the same underlying principles and goal: trying to model and/or predict user behaviour by measuring time taken to execute a task by users. This includes time to physically perform the steps required to carry out the task and mental time required to think about these steps (Card et al., 1980).

For example, the original KLM had five operators to calculate the time taken to execute a task :

- Time to press a key (K)
- Time to point to an object on screen (P)
- Time to move hands on keyboard (H)
- Time to mentally prepare for an action (M)
- Time for system response (R)

The total time to carry out a task is obtained by adding the single task times.

In the context of AAC, KLM can be seen in works by Koester and Levine (Koester and Levine, 1997) (Koester and Levine, 1998). In these works Koester and Levine apply KLM to verify whether execution times usually predicted for users of AAC actually reflect the true times taken by both able bodied users and users with different physical conditions.

GOMS and KLM are based on well established human behaviour laws: Fitts' Law and Hick-Hyman law.

Fitts' Law describes a user's movement time when acquiring or pointing to a visual target. Hick-Hyman law describes human decision time when selecting an element on screen based on the information content of elements on screen (Cockburn et al., 2007).

The formula for Fitts' Law is as follows:

$$MT = a + b * ID$$
$$ID = \log_2 \frac{A}{W} + 1$$

Where MT is Movement Time, A is amplitude of movement between an origin element on screen and a target element on screen, and W is the width of the target button. ID is referred to as Index of Difficulty.

Equation 10 Fitts' Law

Whereas the formula for Hick Hyman Law is as follows:

$$T = a + b * H$$
$$H = \log_2 \frac{1}{p}$$

Equation 11 Hick-Hyman Law

Where T is the time taken by an user to select an item on screen, H is the information content of an object. The information content is calculated as the inverse of the probability of a user selecting a particular object. If the objects on screen have equal probability of being selected then the equation becomes as follows and C equals the number of elements on screen .

$$T = a + b * \log_2 C$$

Equation 12 Hick Hyman Law with Objects of Equal Probability

Both Hicks Hyman and Fitts' contain a and b variables. These a and b variables are calculated by observing time taken by users performing the task: either movement or scanning (depending on the law). The variables are obtained by performing linear regression on the times taken to execute the tasks paired with either the dimension of buttons (Fitts') or the information content of buttons (Hick-Hyman).

The next paragraph will evaluate Dictum by applying a variation on a model proposed by Cockburn et al.(Cockburn et al., 2007) which merges Fitts' and Hick Hyman to predict the time taken by users to find items on different kinds of menus

To merge the two equations Cockburn and his colleagues create a new equation

$$T_i = T_{dsi} + T_{pi}$$

Equation 13 Fitts' Hyman Equation

Where T_i is the total time to select each object on screen.T_{dsi} is the decision search time (Hick Hyman Law) and T_{pi} (Fitts' Law) is item pointing time.

The T_{pi} part of the equation remains unvaried from Equation 10. The T_{dsi} part of the equation is modified from the original Hicks Hyman model to cater for the fact that users have different levels of expertise, so the new model interpolates between linear visual search time (time taken to find objects for beginners) and Hick Hyman decision time (time taken to find objects for experts).

$$T_{dsi} = (1 - e_i)T_{vsi} + e_i T_{hhi}$$
$$e_i = L * (l - \left(\frac{l}{t_i}\right))$$
$$0 \leq e_i \leq 1$$
$$0 \leq L \leq 1$$

Equation 14 Fitts Hyman Equation Interpolating between Novice and Expert Users

The T_{dsi}(decision search time) part of the equation is broken down into T_{vsi} (visual search time) and T_{hhi} Hick Hyman time. The variable e_i indicates the user's expertise, a value that ranges between 0 (complete novice) and 1(complete expert). It is modelled taking into account: t_i, the number of previous trials and L, the learnability of the interface which ranges from 0 (non-learnable interface) and 1 (learnable interface). Learnability depends on the average length that objects move on the screen proportional to half of the length of the menu.

The Hick Hyman model assumes that visual search time(T_{vsi}) is linear when the user is novice, and then becomes negligible when the user becomes expert.

$$T_{vsi} = b_{vs}n + a_{vs}$$

Equation 15 Visual Search Time in Fitts Hyman

Once a user is expert the time to find an object depends on the information conveyed by that object.

9.3.6 Evaluation of Dictum using Fitts- Hyman law

The purpose of this paragraph is to discuss the evaluation of Dictum by comparing it to Verbally and GridPlayer by using a variation on the Fitts – Hyman predictive model. As stated in the previous paragraph, the Fitts-Hyman predictive model was created by Cockburn, Gutwin and Greenberg in (Cockburn et al., 2007).

The model tries to predict the impact of different interface content on user execution time when selecting an object on screen.

For this thesis an adapted model was created to evaluate the interface of Dictum with different performance scenarios of the predictions offered by the Semantic

Network generated by the NLP pre-processor (Chapter V). Using simulations to predict different performances of an application has been used before in literature, for example simulating the impact of different interfaces for users with special needs has been done before in (Biswas and Robinson, 2010), where the authors simulate the impact of changes in the interface of an AAC application on users with different impairments.

The goal of the simulations executed for this evaluation was to emulate different performance scenarios of the Semantic Network which presents word predictions to users of Dictum.

The different performance scenarios simulated for this evaluation were the following:

The first scenario was chosen to simulate extremely poor performance of the Semantic Network

- 70% of times the user is able to find the desired word in the first third of the prediction results.

The second scenario simulates a slightly poor performance by the system

- 90% of times the user is able to find the desired word in the first third of the prediction results.

The third scenario simulates a realistic performance by the system

- 60% of times the user is able to find the desired words in the first tenth of the prediction results.

Finally the last three scenarios simulate a perfect performance of the Semantic Network.

- 90% of times the user is able to find the desired words in the first tenth of the prediction results.
- 95% of times the user is able to find the desired words in the first tenth of the prediction results.
- 99% of times the user is able to find the desired words in the first tenth of the prediction results.

The simulation simulates users navigating through the Semantic Network of Dictum. The simulation is executed by using a random generator which transverses the Semantic Network simulating a user selecting words. In practice the random generator generates a random number between 0 and 49. Each generated number represents a button press from the user. The range of the generator is made to match the minimum and maximum index of buttons in the grid containing nouns at the centre of the screen.

The random generator is biased to simulate the performance scenarios listed above. The model is modified from the Cockburn et al. model to account for the fact that the experiment is simulated and for the fact that buttons on screen are sorted by relatedness with last word inserted (which is akin to likelihood of selection).

Although testing with real users would be ideal, simulating the experiment offers greater flexibility to analyse differences in the performance of the Semantic Network. Additionally it is difficult to find suitable users with CCN and even if found, these experiments can be stressful for such individuals For the experiment to work, the time a non expert user fixates on an element of a list when seeking a particular element is required. Since not enough real users were available to participate in the experiment, the fixation time on screen was taken from the work of others who have empirically observed participants looking for objects in unfamiliar lists.

The differences between the Cockburn et al. model and the model used for the simulation will be explained in more detail in the next paragraph.

9.3.6.1 Setting up the evaluation and calculating *a* and *b* values

The first step in applying user modelling techniques is to calculate the *a* and *b* variables. Both Hick Hyman Law and Fitts' Law equations contain *a* and *b* variables. These are calculated by observing users performing a particular task and then computing linear regression on the measurements.

In the Cockburn et al. model which has just been discussed, a set of three *a* and *b* values is required:

1) *a* and *b* for movement time/pointing time, for the part regarding Fitts' Law

2) *a* and *b* for novice users who require linear time to find the desired object on screen

3) *a* and *b* for expert users who take Hick Hyman time to find an object.

For the purposes of this experiment only the first two sets of *a* and *b* was calculated as discussed below.

For the part of the equation regarding the time of movement (Fitts' Law), the calculations were based on previous work by El Batran and Dunlop (El Batran and Dunlop, 2014). The authors calculate *a* and *b* by measuring tap interaction time of users on a 7 inch screen, the same size as the device (Samsung Galaxy Tab 3) which was used to show the application to users and to evaluate it. El Batran and Dunlop find an *a* value of 52.12 and a *b* value of 14.62. However it was decided not to use these values in the final results since although the target users for this work are able bodied they are for the most part older adults who have slower movement times than the individuals used in El Batran and Dunlop's work.

For the part regarding the visual search time and decision time (Hick Hyman), the simulation discussed at the start of this chapter was used to simulate a user looking for buttons on screen.

The design of the experiment and the formula used differed from those discussed in the work of Cockburn et al. for two reasons. The first reason is that empirically deriving a and b requires the observation of a considerable amount of users performing a task and as discussed throughout this work it is very difficult to find users with CCN who are also familiar with electronic AAC and mobile devices. The second reason is that even in the case of non expert users, the formula for execution time must cater for the fact that buttons on screen are ranked based on Word Association Strength (WAS) with the last word selected.

This means that the simulation is concerned with novice users. Therefore only one set of a and b values was calculated representing novice users navigating a set of words sorted by Word Association Strength with the last word selected (the noun grid).

The original model was reformulated and the following formula was used for calculating time required to find desired word:

$$\mathrm{T} = a + bWAS$$
$$\mathrm{WAS} = nTargetButtons - buttonRanking$$

Equation 16 Formula Used in this Work to Represent Hick Hyman Time for Dictum

The formula looks at novice users using Dictum but instead of considering the impact of different numbers of buttons on screen (Equation 15), it looks at how ranking buttons based on semantic similarity impacts the scanning time, by simulating different scenarios where the desired words are located in different positions in the list.

Given that the task for each user is to search for a target button given an origin button the probability of the user clicking a button was assumed to be the strength of association with the origin button which is the reverse ranking of each button. Using ranking (Word Association Strength) instead of entropy (as per the original model) is valid because the ranking of buttons in Dictum is based on Mutual Information which like entropy is also based on information content (as explained in Chapter II).

In practice a simulation was created and executed which simulates users navigating through the Semantic Network of nouns. Times taken to locate buttons were based on works on neurolinguistics which have found that each person fixates on objects onscreen for a fixed amount of time that ranges between 180ms and 275ms when looking for an object onscreen (Rayner, 2009). This was used as time for a novice user. As mentioned before, it is assumed that a novice user takes linear time to find the desired button and reads each object onscreen before finding the one he is looking for. In practice the simulation algorithm randomly selects an initial time between 180ms and 270ms and multiplies it by the position of a randomly selected index which represents a button on the screen. For example if the generator returns button number four, four represents the fourth position on the screen and the initial time is multiplied by 4.The simulation simulates users having to read each object to find the desired word

The Cockburn et al. model assumes that each button has equal probability of being selected by the user. As mentioned, equal probability cannot be assumed for buttons of Dictum, as these are ranked using human like word associations. Equal probability of buttons being selected by novice users can be assumed for the interfaces of Verbally and GridPlayer, as the buttons are arranged in an order that is not related to the probability of selecting a particular word, rather, either words are arranged alphabetically (in the case of Verbally) or in an order the reasoning of which is not clear (GridPlayer).

9.3.6.2 Simulation Overview and Discussion on Biasing the Random Generator

The algorithm used for the simulation takes in the number of tasks and a Boolean variable as parameters. This variable determines whether the random generator should be biased.

The purpose of the algorithm is to use the adapted Hick Hyman model to calculate the average time taken by users to find a single word given different performances of the Semantic Network, where users find the majority of desired words in different sections of the screen. To achieve this, the random generator was biased to simulate different grades of performance of the Semantic Network. To balance the study, and to cater for apps such as Verbally and GridPlayer a scenario was also simulated where the random generator is unbiased.

The formula below (Equation 17)[35] was used to bias the random generator towards a smaller or larger fraction of values. The formula takes in two parameters: the first is a threshold value (t) which represents the range of values in which the majority of the numbers generated should be. The second parameter (p) represents the percentage of numbers generated which should be within the bias.

$$Target\ Index = (Rand()^{\ln t / \ln p}) * Rand([0, n\ of\ target\ buttons - 1])$$
$$t = threshold$$
$$p = \%\ of\ numbers\ generated\ by\ the\ random\ generator\ to\ be\ in\ the\ bias$$
$$n = 50$$

Equation 17 Biasing Function for Random Generator

To better explain the simulation flow a high level overview of the steps comprising the simulation will be given here

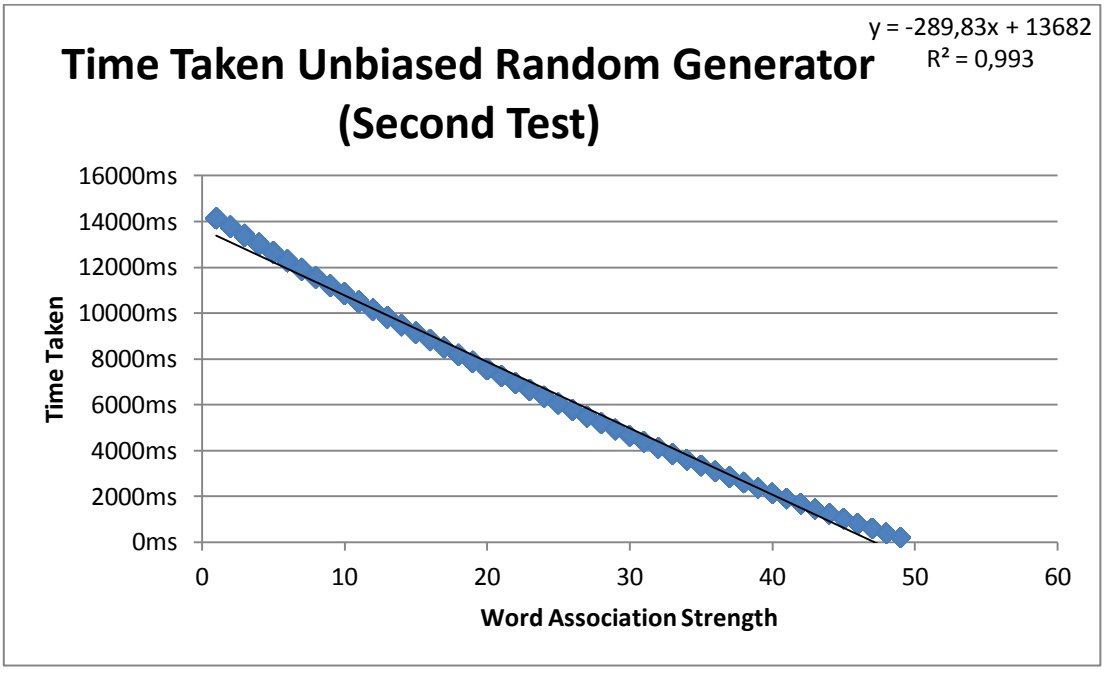
1. *First task?*
    - *y. Origin word = First word in Semantic Network*
    - *n. Origin word = previous target word*
2. *Generate random number*
    - *i. If biased use formula with appropriate parameters*
    - *ii. If unbiased use standard rand function*
3. *Target word = word with index equal to random generated number*
4. *Repeat until 5000 tasks*

[35]http://geekyisawesome.blogspot.co.uk/2010/07/biasing-uniform-random-fraction.html

9.3.6.3 Results

This paragraph will discuss the results of this evaluation by comparing the *a* and *b* values obtained by performing linear regression on this data. Each simulation was repeated twice to assess the effect of natural variability caused by the random variation. With the exception of the first two simulations, where the random generator is unbiased, when simulations are repeated, results are consistent and differences (if any) in *a* and *b* values are negligible (shown in Table 20).

The graphs below show the results of linear regressions for each simulation. The objective of these simulations was to measure the average time to find a single button under different performances of the Semantic Network. Each plot in the graphs represents a tap on the screen at different Word Association Strengths. Each graph contains 5000 plots. To help the reader, below each graph a table shows the tap distribution.



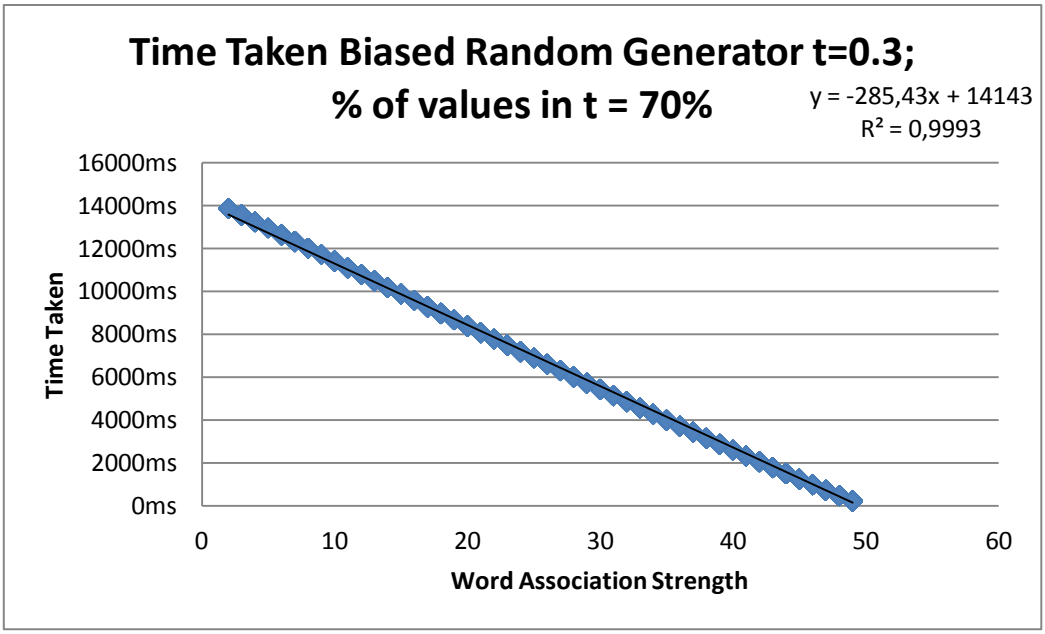
Graph 11 Second Test With Unbiased Random Generator

WAS	Taps							
49	97	1.94	31	96	1.92	12	108	2.16
48	100	2	30	97	1.94	11	99	1.98
47	99	1.98	29	95	1.9	10	96	1.92
46	103	2.06	28	99	1.98	9	95	1.9
45	108	2.16	27	96	1.92	8	93	1.86
44	110	2.2	26	98	1.96	7	97	1.94
43	109	2.18	25	95	1.9	6	97	1.94
42	114	2.28	24	96	1.92	5	92	1.84
41	111	2.22	23	99	1.98	4	100	2
40	119	2.38	22	106	2.12	3	102	2.04
39	107	2.14	21	104	2.08	2	94	1.88
38	106	2.12	20	103	2.06	1	97	1.94
37	107	2.14	19	112	2.24	0	0	0
36	105	2.1	18	113	2.26	Total	5000	100
35	91	1.82	17	107	2.14			
34	100	2	16	114	2.28			
33	97	1.94	15	112	2.24			
32	94	1.88	14	107	2.14			
			13	104	2.08			

Table 13 Tap Distribution Unbiased Random Generator

Graph 11 shows linear regression results obtained by running a simulation with unbiased random numbers. In practice this scenario could simulate other standard Grid Based apps where selection of words from a novice user is random as the words are arranged in an order that has nothing to do with their relatedness to their categorisation and is based on arbitrary decisions on the part of developers.

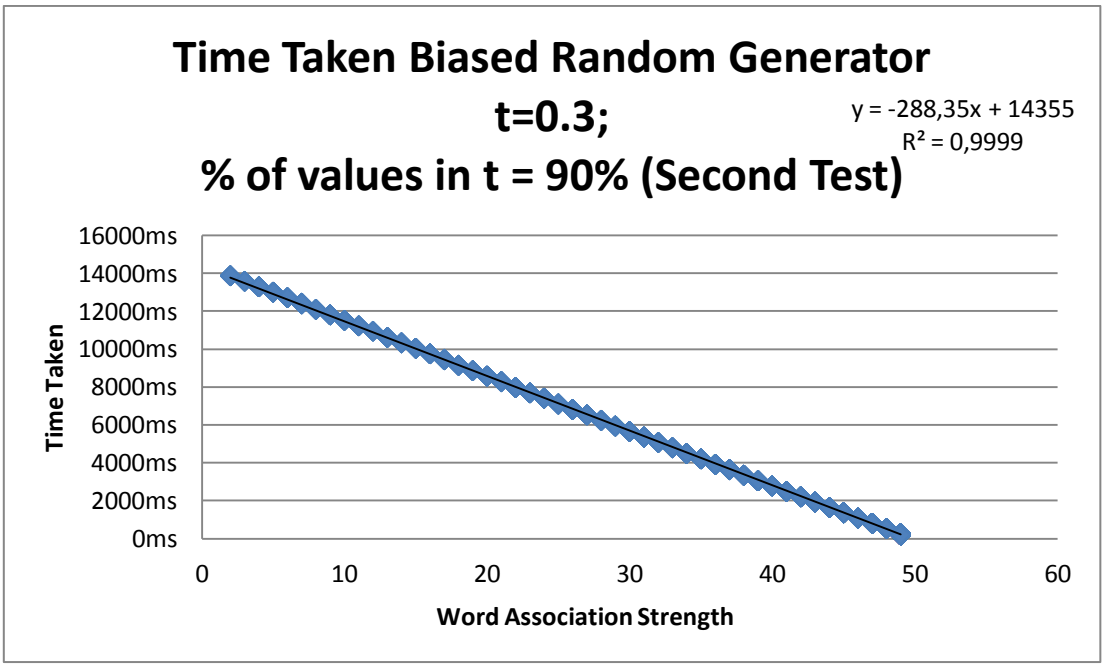
Since the average time obtained is different every time, there will be scenarios were users are faster to find the desired button even without prediction offered by Dictum. However as shown earlier by measuring Keystroke Saving, even if a user is able to locate a single word faster, the combination of various screens and the increased number of buttons required will mean that, in most cases, the user will still take more time than when using Dictum.



Graph 12 Time Taken for novice users with t =0.3 and % < t =70

WAS	N. Taps	% Taps	32	40	0.8	14	39	0.78
49	2029	40.58	31	60	1.2	13	16	0.32
48	383	7.66	30	44	0.88	12	28	0.56
47	239	4.78	29	52	1.04	11	38	0.76
46	203	4.06	28	55	1.1	10	38	0.76
45	127	2.54	27	31	0.62	9	5	0.1
44	134	2.68	26	53	1.06	8	35	0.7
43	106	2.12	25	22	0.44	7	39	0.78
42	95	1.9	24	52	1.04	6	24	0.48
41	83	1.66	23	47	0.94	5	11	0.22
40	72	1.44	22	21	0.42	4	29	0.58
39	82	1.64	21	45	0.9	3	34	0.68
38	70	1.4	20	41	0.82	2	36	0.72
37	58	1.16	19	15	0.3	1	0	0
36	74	1.48	18	47	0.94	0	0	0
35	43	0.86	17	41	0.82	Total	5000	100
34	54	1.08	16	11	0.22			
33	60	1.2	15	39	0.78			

Table 14 Tap Distribution for t = 0.3 and 70% in t

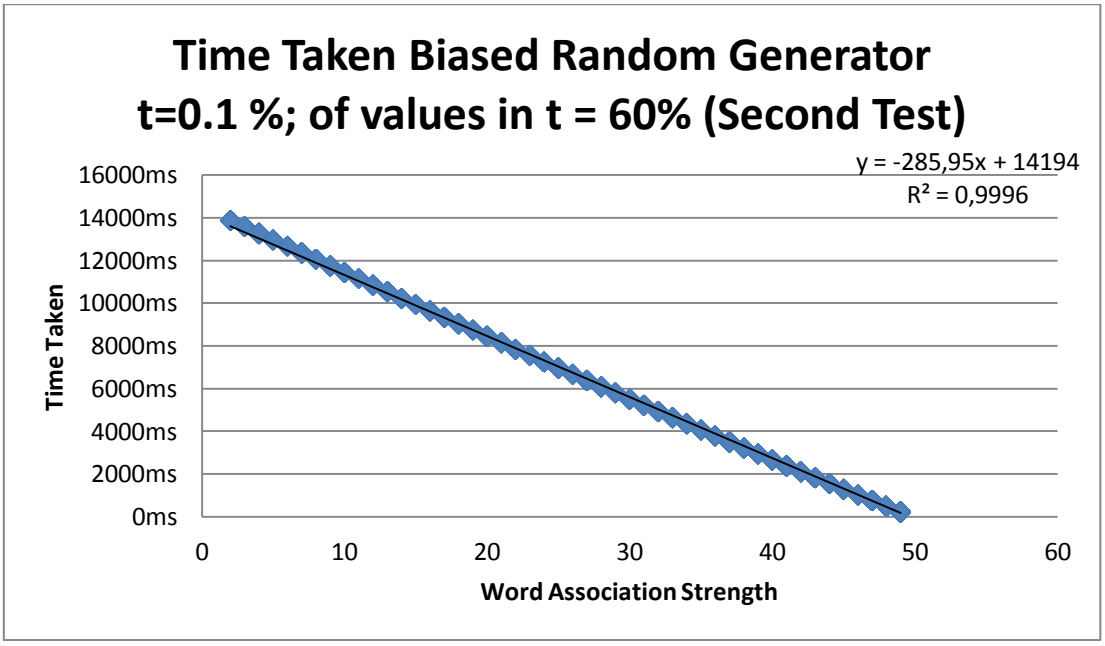


Graph 13 Time Taken for novice users with t =0.3 and % in t =90

WAS	Taps	%taps
49	3683	73.66
48	198	3.96
47	110	2.2
46	90	1.8
45	77	1.54
44	55	1.1
43	51	1.02
42	45	0.9
41	43	0.86
40	31	0.62
39	38	0.76
38	31	0.62
37	23	0.46
36	32	0.64
35	29	0.58
34	26	0.52
33	20	0.4
32	16	0.32
31	25	0.5
30	24	0.48
29	22	0.44
28	13	0.26
27	13	0.26
26	18	0.36
25	15	0.3
24	16	0.32
23	16	0.32
22	16	0.32
21	9	0.18
20	12	0.24
19	13	0.26
18	15	0.3
17	13	0.26
16	12	0.24
15	13	0.26
14	13	0.26
13	14	0.28
12	4	0.08
11	9	0.18
10	9	0.18
9	14	0.28
8	12	0.24
7	13	0.26
6	9	0.18
5	11	0.22
4	13	0.26
3	6	0.12
2	10	0.2
1	0	0
0	0	0
Total	5000	100

Table 15 Tap Distribution t =0.3 % 90

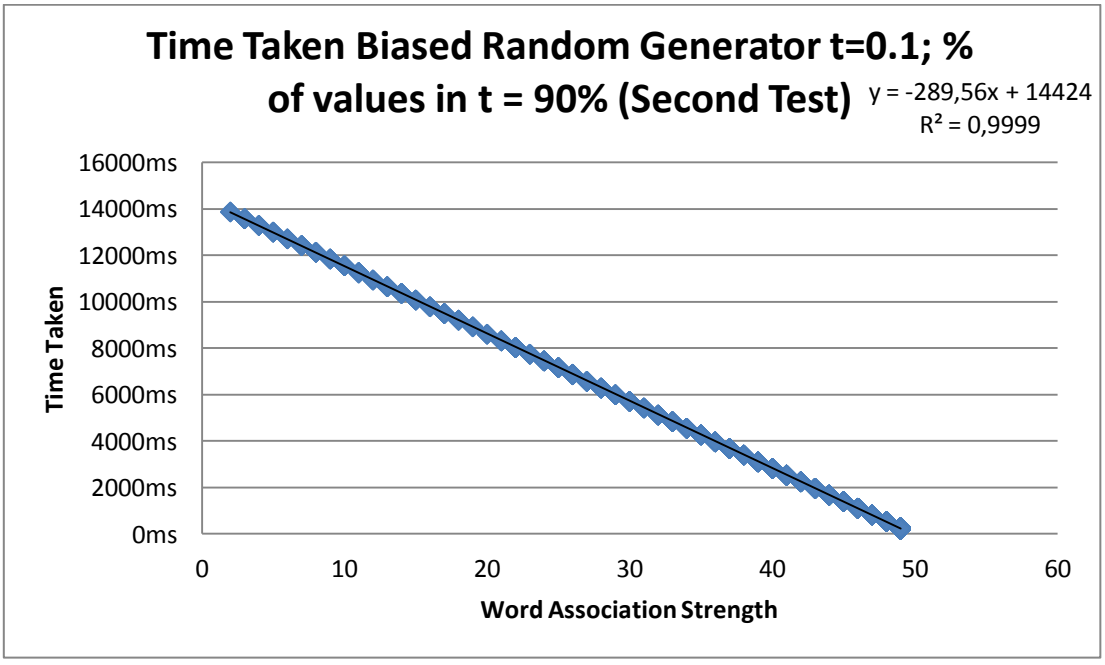
Graph 12 and Graph 13 simulate scenarios where different percentages of desired words are found by users of Dictum in the first third of the prediction results. Given that for each noun Dictum offers a set of 50 related words, the first third includes the first 17 related nouns.



Graph 14Time Taken for novice users with t =0.1 and % in t =60

WAS	Taps	% Taps
49	2466	49.32
48	333	6.66
47	212	4.24
46	145	2.9
45	143	2.86
44	112	2.24
43	95	1.9
42	84	1.68
41	74	1.48
40	84	1.68
39	59	1.18
38	62	1.24
37	68	1.36
36	41	0.82
35	61	1.22
34	35	0.7
33	51	1.02
32	49	0.98
31	32	0.64
30	52	1.04
29	19	0.38
28	50	1
27	41	0.82
26	21	0.42
25	41	0.82
24	35	0.7
23	23	0.46
22	33	0.66
21	34	0.68
20	37	0.74
19	10	0.2
18	30	0.6
17	33	0.66
16	25	0.5
15	14	0.28
14	30	0.6
13	26	0.52
12	30	0.6
11	3	0.06
10	32	0.64
9	22	0.44
8	29	0.58
7	23	0.46
6	6	0.12
5	23	0.46
4	26	0.52
3	24	0.48
2	22	0.44
1	0	0
0	0	0
	5000	100

Table 16 Tap Distribution t =0.1 %60

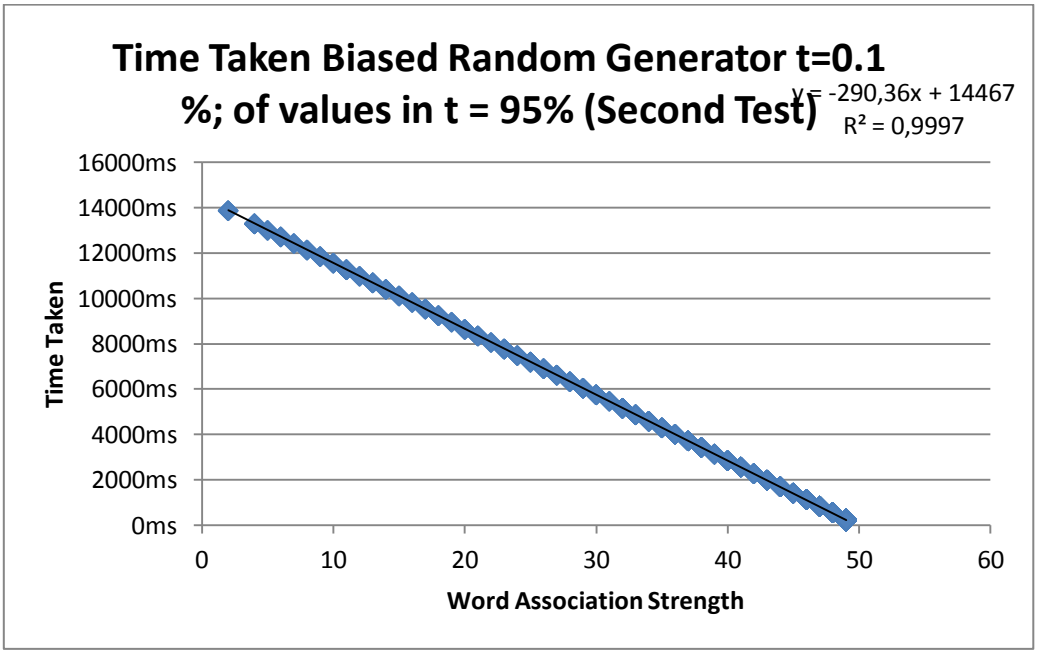


Graph 15Time Taken for novice users with t =0.1 and % in t =90

WAS	Taps	% Taps
49	4223	84.46
48	136	2.72
47	75	1.5
46	58	1.16
45	43	0.86
44	38	0.76
43	26	0.52
42	30	0.6
41	23	0.46
40	26	0.52
39	17	0.34
38	14	0.28
37	15	0.3
36	18	0.36
35	13	0.26
34	13	0.26
33	11	0.22
32	13	0.26
31	7	0.14
30	10	0.2
29	15	0.3
28	9	0.18
27	5	0.1

26	10	0.2	16	2	0.04	6	7	0.14
25	10	0.2	15	8	0.16	5	7	0.14
24	8	0.16	14	6	0.12	4	5	0.1
23	10	0.2	13	7	0.14	3	5	0.1
22	7	0.14	12	7	0.14	2	7	0.14
21	5	0.1	11	4	0.08	1	0	0
20	10	0.2	10	9	0.18	0	0	0
19	3	0.06	9	4	0.08		5000	100
18	7	0.14	8	2	0.04			
17	8	0.16	7	4	0.08			

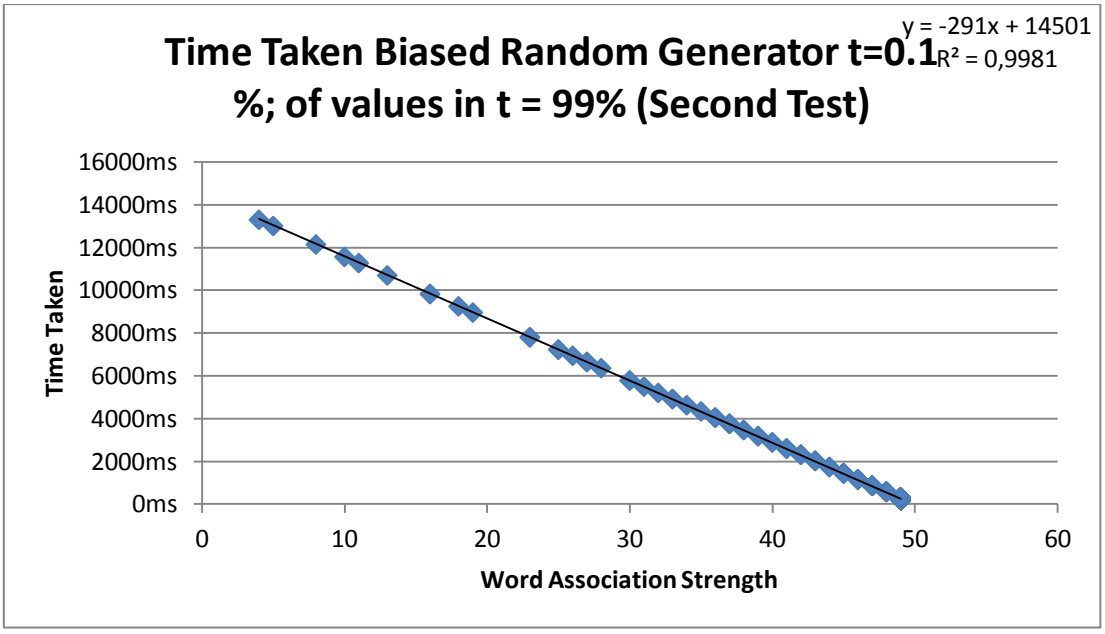
Table 17 Tap Distribution t= 0.1 % in t = 90



Graph 16 Time Taken for Novice users with t =0.1 and % in t =95

WAS	Taps	%Taps						
49	4579	91.58	32	6	0.12	14	3	0.06
48	70	1.4	31	5	0.1	13	5	0.1
47	44	0.88	30	6	0.12	12	2	0.04
46	30	0.6	29	3	0.06	11	4	0.08
45	24	0.48	28	5	0.1	10	3	0.06
44	19	0.38	27	9	0.18	9	4	0.08
43	17	0.34	26	6	0.12	8	2	0.04
42	15	0.3	25	5	0.1	7	3	0.06
41	11	0.22	24	2	0.04	6	1	0.02
40	10	0.2	23	4	0.08	5	5	0.1
39	13	0.26	22	4	0.08	4	1	0.02
38	8	0.16	21	2	0.04	3	2	0.04
37	9	0.18	20	4	0.08	2	6	0.12
36	7	0.14	19	3	0.06	1	0	0
35	9	0.18	18	5	0.1	0	0	0
34	8	0.16	17	3	0.06		5000	100
33	8	0.16	16	1	0.02			
			15	5	0.1			

Table 18 Tap Distribution t = 0.1 % in t = 95



Graph 17 Time Taken for Novice Users with t =0.1 and % in t =99

WAS	Taps	% Taps						
49	4920	98.4	42	5	0.1	34	1	0.02
48	14	0.28	41	2	0.04	33	0	0
47	9	0.18	40	2	0.04	32	2	0.04
46	6	0.12	39	2	0.04	31	2	0.04
45	4	0.08	38	1	0.02	30	1	0.02
44	4	0.08	37	1	0.02	29	0	0
43	3	0.06	36	2	0.04	28	2	0.04
			35	1	0.02	27	1	0.02

26	0	0	16	1	0.02	6	1	0.02
25	1	0.02	15	1	0.02	5	0	0
24	1	0.02	14	0	0	4	0	0
23	0	0	13	0	0	3	0	0
22	0	0	12	1	0.02	2	0	0
21	1	0.02	11	1	0.02	1	0	0
20	2	0.04	10	1	0.02	0	0	0
19	1	0.02	9	0	0		5000	100
18	1	0.02	8	1	0.02			
17	0	0	7	1	0.02			

Table 19 Tap Distribution t = 0.1 % in t = 99

The last graphs show linear regression results of data obtained by simulating users finding the desired word in the first 5 words of the prediction. As the percentage of desired words in the range increases the model predicts that the required time to find the word decreases, as would be expected.

A	B	Bias threshold	% < bias threshold
13682	-289.83	UNBIASED	UNBIASED
13706	-290.75	UNBIASED	UNBIASED
14130	-285.2	0.3	70%
14143	-285.43	0.3	70%(Second Test)
Δa =13	**Δb =0.23**		
14352	-288.3	0.3	90%
14335	-288.35	0.3	90% (Second Test)
Δa =17	**Δb =0.05**		
14194	-285.95	0.1	60%
14194	-285.95	0.1	60%(Second Test)
Δa =0	**Δb =0**		
14424	-289.57	0.1	90%
14424	-289.56	0.1	90%(Second Test)
Δa =0	**Δb =0.01**		
14467	-290.37	0.1	95%
14467	-290.36	0.1	95%(Second Test)
Δa =0	**Δb=0.01**		
14498	-290.94	0.1	99%
14501	-291	0.1	99% (Second Test)
Δa =3	**Δb =0.06**		

Table 20 Results of Hick Hyman Simulation

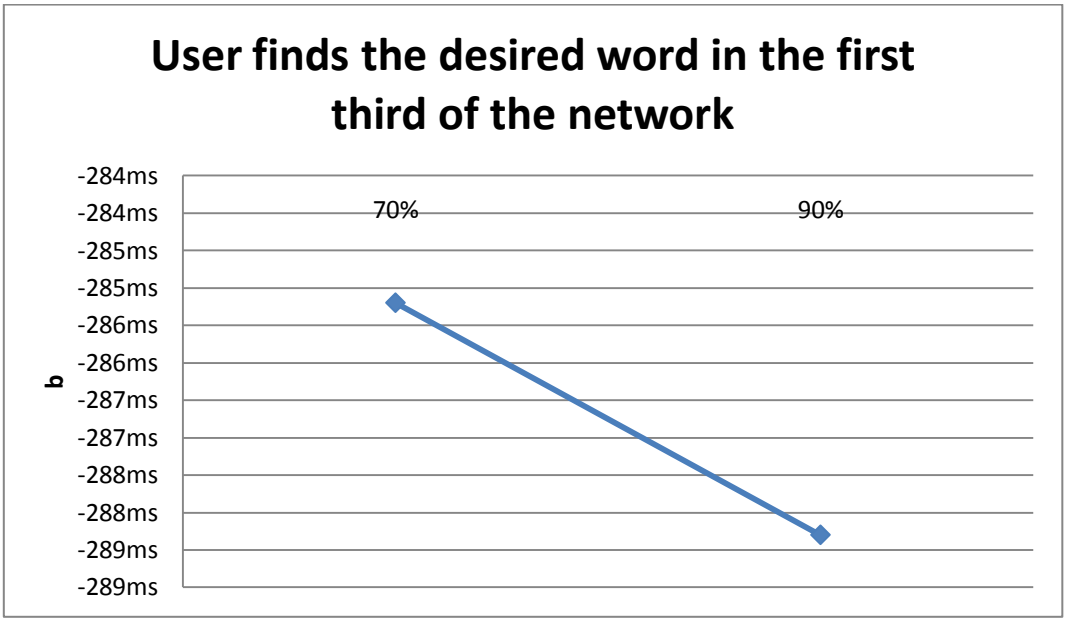
Semantic Network performance	Dictum performance over unbiased random generator (worst performance)
70% in first 1/3	+4.4ms
90% in first 1/3	+1.48ms
60% in first 1/10	+3.88ms
90%in first 1/10	+0.26ms
95%in first 1/10	-0.54ms
99%in first 1/10	-1.17ms

Table 21 Comparison Between Times to Find Single Button

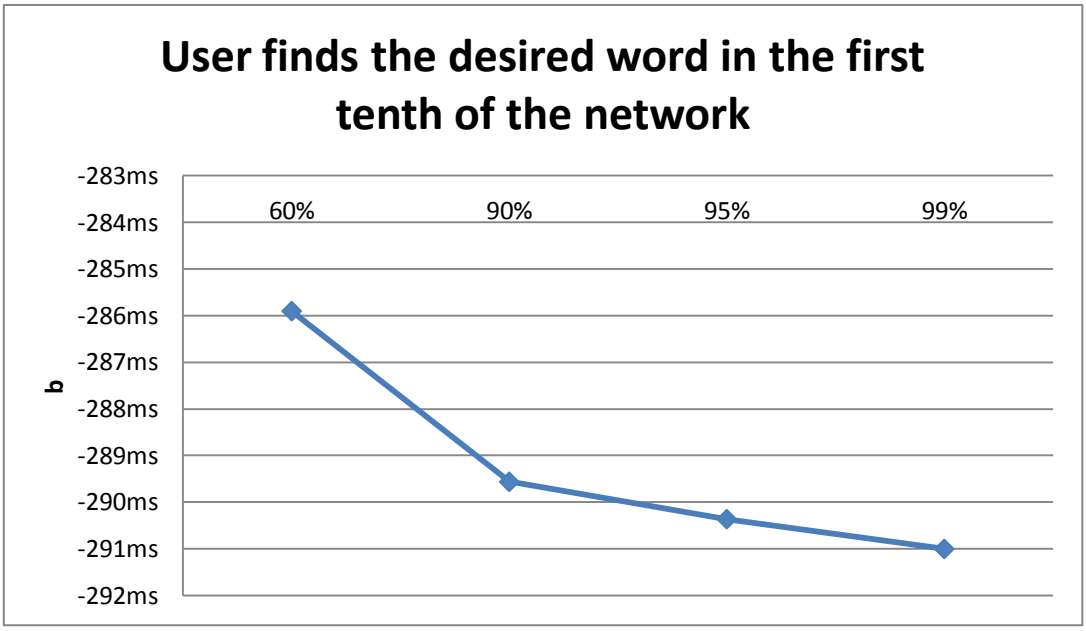
The results of the evaluation will now be recapitulated and discussed further.

The following scenarios were simulated:

- 70% of times the user is able to find the desired word in the first third of the prediction results.
- 90% of times the user is able to find the desired word in the first third of the prediction results.
- 60% of times the user is able to find the desired words in the first tenth of the prediction results.
- 90% of times the user is able to find the desired words in the first tenth of the prediction results.
- 95% of times the user is able to find the desired words in the first tenth of the prediction results.
- 99% of times the user is able to find the desired words in the first tenth of the prediction results.



Graph 18 b values for different performances of the Semantic Network when desired word is in first third of list



Graph 19 b Values for Different Performances when Desired Word is Found in First Tenth of List

Table 20 groups the results of the evaluation just discussed. Each simulation was run for 5000 times simulating a series of users performing 5000 button clicks and navigating through the Semantic Network, to measure the average time taken to find a noun.

The first two rows of the table show the results of the simulation using the unbiased random generator. The results show that the *a* and *b* values generated by the unbiased generator are different every time the test is run, as one would expect in apps where buttons are in no particular order, and the user is unfamiliar with the interface. The results could be assumed to mirror the ones obtained by users of apps such as GridPlayer and Verbally.

The simulations executed to generate *a* and *b* by biasing the random generator were run twice to verify whether the simulation generated consistent results even after multiple runs. The results remain consistent varying only slightly in some cases. The differences between runs for *a* and *b* (symbolised as a delta) are shown in Table 20 in a separate row after the rows containing each block of simulation pairs.

The results obtained by the simulations show that in ideal scenarios where the Semantic Network offers the desired word in the first five results, scanning times are constant varying only slightly. In these scenarios, time to find the desired button is reduced compared to other scenarios. This is graphically shown by Graph 18 and Graph 19 which show the trend in *b* values during simulations. The immediate advantage of using a Semantic Network is not apparent in these tests. The biggest

advantage of Dictum remains the Keystroke Saving. Another aspect which must be considered when looking at these results is that all necessary words are contained in a dynamic screen whereas apps like GridPlayer contain several different screens through which the user must navigate. The influence of these different screens over user performance cannot be easily simulated therefore it is the topic for future work.

9.3.6.4 Introduction of Swipe Movements into Results

The evaluation discussed so far took into account scanning time and Hick Hyman Law. To do so it calculated scanning time for the full set of 50 related nouns the user can choose from once he selects a noun. However, the Dictum interface contains maximum 12 noun buttons at any one time (Version 3 with images) in the grid. To improve and expand on the results of the simulation further calculations were performed on the data to account for the fact that every 12 buttons the user must perform a swipe movement to search for the desired related noun. The amount of time required for a swipe was taken from El Batran and Dunlop's work which was previously mentioned. For swipe tasks they find that users on a 7 inch screen device require an *a* and *b* time of respectively 9.62ms and 55.63ms. To incorporate swipe times into the graphs it was calculated how many swipe movements a user would require to find each button. This amount was then multiplied by the *b* and added to the original tap time. The data were again plotted and linear regression was performed. Since the plotted graphs look very similar to the ones just discussed, the results are presented without graphs in Table 22.

a	B	Original b	Bias
13879	-294.28	-290.75	Unbiased
14317	-289.15	-285.43	0.3 70%
14530	-291.97	-288.35	0.3 90%
14377	-289.78	-285.95	0.1 60%
14602	-293.23	-289.57	0.1 90%
14644	-293.98	-290.37	0.1 95%
14676	-294.59	-291	0.1 99%

Table 22 *a* and *b* Values for Search + Swipe Movement

The simulations were also run by adding both the *a* and *b* values to the total time and by examining the impact of different Indexes of Difficulty on the final results. Overall, the scanning times plus swipe times while differing in value, kept overall constant with values similar to those shown in Table 22. The inclusion of swipe times did not help explain why the unbiased generated times result in a lower average time. Even though the difference is very small this should be investigated

further with the help of real users. Factors such as mental overload, mistakes, mental preparation times were not easily includable in the simulation so it could be the case that with real users average scanning time differences between apps are more marked. The strong point of Dictum for both novice and expert users remains the high Keystroke Saving even if scanning times were the same in an experiment with real users. A study with real users would also examine scenarios where users were forming real sentences and were also using verbs to form sentences. As explained in Chapter VII verbs are located on the right hand side of the screen and are presented to users using a verb map created using the BNC. Verbs were not included in this study as its main purpose was to look at different performances of the Semantic Network of nouns.

9.4 Final Discussion

This evaluation has shown that the integration of NLP/NLG in AAC does improve speed and ease of use for users with CCN.

These results, and primarily the results of the Keystroke Saving comparison prove that Dictum is faster than competitor apps and that the inclusion of NLP and NLG by combining a Semantic Network and generating sentences using key words provided by the user does indeed improve ease and speed of use.

This is because even if the user modelling simulation showed only slight variations in scanning time, the high savings in keystrokes and the reduced mental overload due to having all words on one screen makes Dictum faster to use.

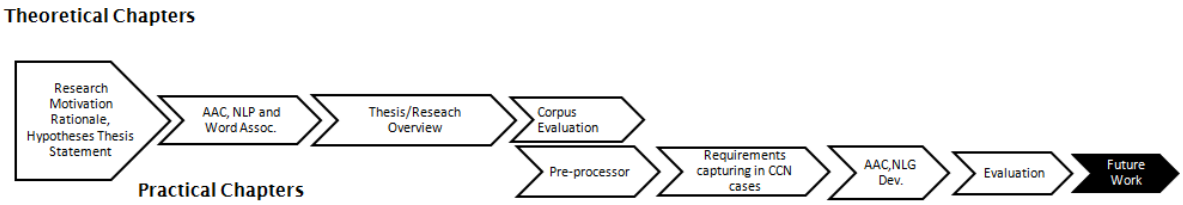
One limitation of the app which will have to be investigated in future is the fact that sometimes a noun is only accessible through selecting a related noun. This is because the app has no search features due to the absence of a keyboard. An investigation should verify whether the inclusion of a search feature in cases where nouns are not present in the main screen still saves strokes when compared to competitor apps.

In future work, if a sufficient number of users is found, a test should be run using the Cockburn et al. model and this model to compare the results against the simulation. The performance of expert users of Dictum should also be studied.

The evaluation has built upon the work of Trnka and McCoy (Trnka and McCoy, 2008) by applying the Keystroke Saving model to AAC and has built upon the work of Cockburn et al. (Cockburn et al., 2007) by adapting their predicting model to Dictum in a scenario where non expert users are presented with an interface where buttons have different probabilities of being selected.

A contribution has been made to finding an appropriate measure of performance for AAC, which is the topic of works such as (Hill and Romich, 2002) and (Smith et al., 2006), Hill and Romich propose a new formula based on Fitts' Law called Rate Index which is calculated by dividing the words per minute by the selection index (a variation on Fitts' Law). Smith et al. developed a real time AAC evaluation system which integrates with the AAC systems and calculates communication rates while the user is communicating with the AAC application.

This evaluation, similar to the work of Hill and Romich, used well established HCI laws to predict the impact of different grades of performance of the Semantic Network on the time taken to form sentences. The results of the evaluation were simulated due to the difficulty in finding suitable participants. Simulations are a well established practice in these cases; an example of this is the work of Biswas and Robinson (Biswas and Robinson, 2010). In their work they use a simulation to evaluate the impact of different interfaces on users with different disabilities.



Chapter X: Future Work

10.1 Introduction

This Chapter will discuss future work and questions that remained unanswered due to the limited timescale of this project.

10.2 Pre-processor

An improvement to the pre-processor could be the inclusion of second order co-occurrences to understand second order co-occurrences (Islam and Inkpen, 2006). Islam and Inkpen make the example of the words *police* and *car* if for example *mechanic* is related to *car* and *agent* is related to *police* then by second order *agent* and *mechanic* are also related.

Another aspect of the pre-processor which was only partially investigated was the inclusion of Word Association Norms. These were downloaded and examined, the choice was made not to include them in the final pre-processor because they were mostly verbs and there was a limited amount of words in the datasets. However including these in the pre-processor pipeline could have beneficial effects.

As was mentioned, there could be better ways to rank words in the vocabularies. The final work settled for using TFIDF at folder level but there were a variety of ranking mechanisms that could have been employed and there was also the possibility of merging some of these ranking mechanisms, this should also be a topic for further investigation.

10.3 AAC Application

An important aspect which must be investigated in future work is how to reintroduce the keyboard into the system. The keyboard is necessary for the user to introduce new words into the system and to slightly modify the structure of the generated sentence. This issue was raised during the discussion of the feedback given by the Speech and Language Therapists who were present at the Dundee meeting. The introduction of the keyboard has to be implemented in such a way as not to defeat the purpose of the Natural Language Generation system. The complexity of this matter is the reason why it is discussed as future work.

The introduction of new words is also a critical aspect of this system; at the moment the files used by the system are previously trained by the pre-processor and then transferred to the system. But since this does not let the user introduce new words into the system a way has to be found to train the system in real time especially when building the Semantic Network of related words. The system must also look for related verbs in the case of a new noun or related nouns in the case of a new verb.

The communication partner also plays a key role in AAC and in standard conversation so ideally a fully functional system must be able to respond to what the interlocutor is asking or saying, therefore the internal model of the system should be able to provide relevant words depending on the communication partner and the nature of the conversation. Context also plays a key role in the words the user might use so using information provided by the device such as location and time of the day. For example if the system should detect that the user is in the bank it could provide words that are related to banks, the same with supermarkets and newsagents.

To achieve this a new a statistical model should be designed which gives a higher probability to words based on context and the person the user is speaking to. This statistical model should give a greater weight to words that are frequently used by the user and should be able to store sentences that are frequently used by the user and present them without the need for the user to reform them from scratch. This feature was also requested by the potential users met during the course of this work.

Chapter VI (Capturing Requirements in cases of CCN) has shown that Appropriation analysis can be used to capture requirements in cases where users

cannot comfortably express requirements but it was also useful in identifying requirements which were not included in the final application. Future work should attempt to include some or all of these requirements.

WordNet has shown great potential as a source of additional information to help generate grammatically correct sentences, however during the course of this work only the relationship between nouns was investigated. However, WordNet also includes relationships between verbs and other verbs and between nouns and verbs. This should also be investigated further.

Alternative sources of linguistic information should also be sought and their inclusion investigated. At the moment the application makes use of the Microsoft N-gram Web service and WordNet, however as discussed in the relevant section there are some problems related to Out Of Vocabulary therefore the use of databases such as VerbNet and PropNet should also be investigated to include as many words in the system as possible and new kind of information to improve the quality of the generated sentences.

The use of parse trees databases should also be investigated further. At the moment it was only investigated with the purpose of finding the correct determiner, but given the vast amount of information contained in these datasets, it is possible that they could improve sentence generation in other ways which were not explored in this work.

The system should include all additions discussed in the final part of the NLG Chapter which for the moment where only discussed from a theoretical point of view.

Finally the user modelling evaluation discussed in Chapter IX should be carried out with real users with Complex Communication Needs to verify and build upon the results of the simulation. The results of the simulation indicated that scanning time for novice users when looking for nouns remained constant. Carrying out the same experiment with real users would either confirm or show that this is not the case and help to understand the perspective of novice users of Dictum. It is my opinion that the decreased mental load of Dictum compared to traditional grid based apps makes the Dictum app faster than competitors even with respect to scanning times, however the results of the evaluation did not reflect this.

Chapter XI: Final Discussion

11.1 Discussion of Overall Results Given Proposed Hypotheses and Research Questions

11.1.1 Discussion on Hypotheses

This work investigated whether the inclusion of Natural Language Processing and Natural Language Generation Techniques improved the ease and speed of Augmentative and Alternative Communication applications. The problem was outlined in Chapter I and rationale was given as to why there is a need to research in this area. At the end of Chapter I two main hypotheses were proposed:

H1. The integration of Natural Language Processing and Natural Language Generation with Augmentative and Alternative Communication will reduce the number of keys required to form sentences compared with competitor apps leading to the potential improvement in speed of sentence formation.

The integration of Natural Language Processing and Natural Language Generation with Augmentative and Alternative Communication Systems does indeed reduce the number of keys required to form sentences. The evaluation in Chapter IX has shown that there is a substantial Keystroke Saving when compared to traditional AAC systems, however the advantage is lost when sentences are shorter than 3 words. It has also been investigated by using well established HCI laws whether the use of the Semantic Network is faster than interfaces of competitor apps where words are arranged in no particular order. The simulation used showed that scanning times in Dictum and competitor apps are similar and that they only vary slightly with different performances of the Semantic Network. Because these experiments were simulated, it could be that these results do not reflect the results of a study with real users. Therefore based on the results presented in this work the strongest point of Dictum in terms of speed remains the considerable key stroke saving obtained by using NLG techniques.

H2 The integration of Natural Language Processing and Natural Language Generation with mobile Augmentative and Alternative Communication systems will reduce screen navigation thereby improving ease of sentence formation by avoiding mental overload and stress on users.

It has been shown that the integration of Natural Language Generation improves ease sentence formation. This has been primarily shown by the feedback which was given by both users and Speech and Language Professionals It emerged that the interface could be improved with further work, The feedback received by both users and professionals was positive. The most important fact to emerge from conversation with both users and professionals was that because Dictum is concentrated in one screen it reduces the chances of mental overload and in general, reduces stress for users, also because the burden of responsibility to form grammatically correct sentences lies with the Dictum rather than users. In future, those interviewed suggested that Dictum's interface could be improved by improving contrast and by catering for those with visual impairments.

11.1.2 Discussion on Research Questions

The following research questions were also proposed in Chapter I. This paragraph will discuss how this work answered the proposed research questions.

Q1. Will the integration of Natural Language Processing and Natural Language Generation improve speed of sentence formation in mobile AAC systems by reducing the number of keys required to form sentences?
Q2. Will the integration of Natural Language Processing and Natural Language Generation improve ease of sentence formation in mobile AAC systems by reducing stress and avoid mental overload?
Q3. Can a system that excludes keyboard and relies on whole words input work as an effective input mechanism in mobile AAC applications?
Q4. Can we discover requirements through the use of User Centred Design in cases where the users have Complex Communication Needs?

The results of the two evaluations have positively answered Q1 and Q2, in some cases Dictum achieved a Keystroke Saving higher than 50%. And in cases where the Semantic Network offers the desired word in the first results scanning time for users is slightly reduced. Whilst it is true that Dictum does not cater for all communicative scenarios, it is not a full system and Chapter VIII has outlined in detailed how a full system would be implemented. The improvement in ease of use was highlighted by users and professionals interviewed during the course of this work who confirmed that both stress and the risk of mental overload were reduced by the proposed design of Dictum. Key factors in the reduction of stress and risk of mental overload were the absence of screen navigation and the ability of Dictum to generate whole

sentences using only key words. This is confirmed both in literature(Light and McNaughton, 2012) and by the feedback given by users and professionals during conversation.

With respect to Q3 keystroke comparison between Dictum and competitor apps has shown that a system which doesn't use a keyboard can be very effective in many cases, however since communication is complex and there are many scenarios to cater for, in some cases a keyboard could be required. In whole words systems the user is constrained by the app's vocabulary. It can be concluded from the results of the evaluation that even if a system relies solely on whole words and manages to cater effectively for the majority of communicative scenarios, some form of letter input is still required, at least to insert new words or modify the app's internal vocabulary.

Q4 was also positively answered. It was shown in Chapter VII that observing user appropriation can be an effective tool to capture requirements in cases where users have Complex Communication Needs.

11.2 Final Discussion

This work investigated whether the inclusion of NLP and NLG techniques in Mobile AAC systems improves the ease and speed of the systems for use with Complex Communication Needs.

The work found that this is possible and a system was developed which was faster than the most popular free apps available on the market today. Aspects of the system were only developed theoretically but the results of the evaluation showed that the right approach was followed and that with further work even better results can be achieved. The main advantages brought by the inclusion of NLP and NLG in AAC were the reduced number of keys required to form sentences, achieved mainly by the use of NLG which enables the user to form sentences using only key words, and the reduced stress on users which was achieved by the reduced number of keys just mentioned and by the absence of screen navigation which causes stress for users. The direct link between screen navigation and stress (which can sometimes lead to mental overload) was confirmed in both literature (Light and McNaughton, 2012) and in discussion with users when designing and evaluating Dictum. Therefore it can be concluded that greater speed of use was achieved (by reducing keys) and greater ease of use was achieved (by reducing screen navigation and

number of keys required) thus positively answering the hypotheses and the first two research questions given at the start of this thesis. The results of the user modelling evaluation need to be further investigated with the help of real participant to quantify the real impact of the performance of the Semantic Network. However, it can be assumed that a high performance of the Semantic Network also contributes to reducing stress and helps the user focus on the conversation and the interlocutor rather than on the search for words.

This research was conducted extensively by looking at all avenues of investigation using the work of others as foundation, chose the most effective ones. Due to timescale limitations some avenues were left for future work. Even so the work managed to prove all given hypotheses and answer all given research questions exhaustively.

It is my opinion, that by continuing this work it would be possible to develop a fully functional system which could achieve high-speed and could cater for a greater variety of communication scenarios in which the Dictum app is expanded to handle more topics and exploits additional input such as the location where a conversation is taking place, the type of interlocutor and the relation he/she has with the user.

Finally, work contributed to several fields, namely AAC, NLP/NLG and HCI especially by finding an effective way of capturing requirements for user who cannot express themselves; namely the contributions of this work were:

- Applying Action Research methodology to the development of AAC
- Investigating the inclusion of NLP NLG in AAC
- Investigating the use Wikipedia as a corpus in NLP tasks.
- Applying crowdsourcing to the evaluation of corpora.
- Utilising HCI appropriation to capture requirements in cases of users with Complex Communication Needs.
- Implementing an algorithm similar to the Compansion system and overcoming its limitations and turning it into a real system and not only theoretical.
- Implementing a Formal Language to communicate between the NLG server and the application which was effective and easily scalable.
- Developing an NLG system that uses several sources of information including Word net and Microsoft N Gram Web Service.

References

Allen, M., McGrenere, J., Purves, B., 2007. The Design and Field Evaluation of PhotoTalk: A Digital Image Communication Application for People, in: Proceedings of the 9th International ACM SIGACCESS Conference on Computers and Accessibility, Assets '07. ACM, New York, NY, USA, pp. 187–194. doi:10.1145/1296843.1296876

Alm, N., Arnott, J., Newell, A.F., 1990. Hypertext as a Host for an Augmentative Communication System, in: Proceedings of the European Conference on the Advancement of Rehab Technology. Presented at the ECART, Maastricht, The Netherlands.

Arnott, J.L., Alm, N., 2013. Towards the improvement of Augmentative and Alternative Communication through the modelling of conversation. Comput. Speech Lang., Special Issue on Speech and Language Processing for Assistive Technology 27, 1194–1211. doi:10.1016/j.csl.2012.10.008

Aston, G., Barnard, Lou, 1998. The BNC handbook: Exploring the British National Corpus with SARA.

Augmentative and Alternative Communication | International Encyclopedia of Rehabilitation [WWW Document], 2010. URL http://cirrie.buffalo.edu/encyclopedia/en/article/50/ (accessed 1.15.15).

Banerjee, S., Pedersen, T., 2003. Extended Gloss Overlaps As a Measure of Semantic Relatedness, in: Proceedings of the 18th International Joint Conference on Artificial Intelligence, IJCAI'03. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 805–810.

Bird, S., Klein, E., Loper, E., 2009. Natural Language Processing with Python. O'Reilly Media.

Biswas, P., Robinson, P., 2010. Evaluating the Design of Inclusive Interfaces by Simulation, in: Proceedings of the 15th International Conference on Intelligent User Interfaces, IUI '10. ACM, New York, NY, USA, pp. 277–280. doi:10.1145/1719970.1720010

Biswas, P., Samanta, D., 2008. Friend: A Communication Aid for Persons With Disabilities. IEEE Trans. Neural Syst. Rehabil. Eng. 16, 205–209. doi:10.1109/TNSRE.2008.917305

Bizer, C., Lehmann, J., Kobilarov, G., Auer, S., Becker, C., Cyganiak, R., Hellmann, S., 2009. DBpedia - A crystallization point for the Web of Data. Web Semant. Sci. Serv. Agents World Wide Web, The Web of Data 7, 154–165. doi:10.1016/j.websem.2009.07.002

Black, R., Reddington, J., Reiter, E., Tintarev, N., Waller, A., 2010. Using NLG and Sensors to Support Personal Narrative for Children with Complex Communication Needs, in: Proceedings of the NAACL HLT 2010 Workshop on Speech and Language Processing for Assistive Technologies, SLPAT '10. Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 1–9.

Black, R., Waller, A., Turner, R., Reiter, E., 2012. Supporting Personal Narrative for Children with Complex Communication Needs. ACM Trans Comput-Hum Interact 19, 15:1–15:35. doi:10.1145/2240156.2240163

Bradshaw, J., 2013. The use of augmentative and alternative communication apps for the iPad, iPod and iPhone: an overview of recent developments. Tizard Learn. Disabil. Rev. 18, 31–37. doi:10.1108/13595471311295996

Brandenburg, C., Worrall, L., Rodriguez, A.D., Copland, D., 2013. Mobile computing technology and aphasia: An integrated review of accessibility and potential uses. Aphasiology 27, 444–461. doi:10.1080/02687038.2013.772293

Card, S.K., Moran, T.P., Newell, A., 1980. The Keystroke-level Model for User Performance Time with Interactive Systems. Commun ACM 23, 396–410. doi:10.1145/358886.358895

Card, S.K., Newell, A., Moran, T.P., 1983. The Psychology of Human-Computer Interaction. L. Erlbaum Associates Inc., Hillsdale, NJ, USA.

Carroll, J., Howard, S., Peck, J., Murphy, J., 2007. From Adoption to Use: the process of appropriating a mobile phone. Australas. J. Inf. Syst. 10. doi:10.3127/ajis.v10i2.151

Carroll, J., Howard, S., Vetere, F., Peck, J., Murphy, J., 2002. Just what do the youth of today want? Technology appropriation by young people, in: Proceedings of the 35th Annual Hawaii International Conference on System Sciences, 2002. HICSS. Presented at the Proceedings of the 35th Annual Hawaii International Conference on System Sciences, 2002. HICSS, pp. 1777–1785. doi:10.1109/HICSS.2002.994089

Carroll, J., Howard, S., Vetere, F., Peck, J., Murphy, J., 2001. Identity, Power And Fragmentation in Cyberspace: Technology Appropriation by Young People. ACIS 2001 Proc.

Chaudhari, D.L., Damani, O.P., Laxman, S., 2011. Lexical Co-occurrence, Statistical Significance, and Word Association, in: Proceedings of the Conference on Empirical Methods in Natural Language Processing, EMNLP '11. Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 1058–1068.

Chowdhury, G.G., 2003. Natural language processing. Annu. Rev. Inf. Sci. Technol. 37, 51–89. doi:10.1002/aris.1440370103

Church, K.W., Hanks, P., 1990. Word Association Norms, Mutual Information, and Lexicography. Comput Linguist 16, 22–29.

Clarke, M., McConachie, H., Price, K., Wood, P., 2001. Views of young people using augmentative and alternative communication systems. Int. J. Lang. Commun. Disord. R. Coll. Speech Lang. Ther. 36, 107–115.

Cockburn, A., Gutwin, C., Greenberg, S., 2007. A Predictive Model of Menu Performance, in: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '07. ACM, New York, NY, USA, pp. 627–636. doi:10.1145/1240624.1240723

Damani, O.P., 2013. Improving Pointwise Mutual Information (PMI) by Incorporating Significant Co-occurrence. ArXiv13070596 Cs.

Damani, O.P., Chedda, P., Chaudhari, D., 2012. Wikipedia is a Practical Alternative to the Web for measuring Co-occurrence based Word Association.

Damani, O.P., Ghonge, S., 2013. Appropriately Incorporating Statistical Significance in PMI., in: Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics, Seattle, Washington, USA, pp. 163–169.

Deerwester, S., Dumais, S.T., Furnas, G.W., Landauer, T.K., Harshman, R., 1990. Indexing by latent semantic analysis. J. Am. Soc. Inf. Sci. 41, 391–407.

Demasco, P.W., McCoy, K.F., 1992. Generating Text from Compressed Input: An Intelligent Interface for People with Severe Motor Impairments. Commun ACM 35, 68–78. doi:10.1145/129875.129881

Dix, A., 2007. Designing for Appropriation, in: Proceedings of the 21st British HCI Group Annual Conference on People and Computers: HCI...But Not As We Know It - Volume 2, BCS-HCI '07. British Computer Society, Swinton, UK, UK, pp. 27–30.

Dunning, T., 1993. Accurate Methods for the Statistics of Surprise and Coincidence. Comput Linguist 19, 61–74.

El Batran, K., Dunlop, M.D., 2014. Enhancing KLM (keystroke-level model) to fit touch screen mobile devices. ACM Press, pp. 283–286. doi:10.1145/2628363.2628385

Elden, M., Chisholm, R.F., 1993. Emerging Varieties of Action Research: Introduction to the Special Issue. Hum. Relat. 46, 121–142. doi:10.1177/001872679304600201

Elhadad, M., Robin, J., 1996. An Overview of SURGE: a Reusable Comprehensive Syntactic Realization Component.

Fecht, S., 2014. Playing Fruit Ninja Helps Stroke Victims [WWW Document]. Pop. Mech. URL http://www.popularmechanics.com/science/health/med-tech/playing-fruit-ninja-helps-stroke-victims-16332178 (accessed 1.31.15).

Fellbaum, C., 1998. A Semantic Network of English: The Mother of All WordNets. Comput. Humanit. 32, 209–220.

Flores, M., Musgrove, K., Renner, S., Hinton, V., Strozier, S., Franklin, S., Hil, D., 2012. A comparison of communication using the Apple iPad and a picture-based system. Augment. Altern. Commun. Baltim. Md 1985 28, 74–84. doi:10.3109/07434618.2011.644579

Galliers, J., Wilson, S., Roper, A., Cocks, N., Marshall, J., Muscroft, S., Pring, T., 2012. Words Are Not Enough: Empowering People with Aphasia in the Design Process, in: Proceedings of the 12th Participatory Design Conference: Research Papers - Volume 1, PDC '12. ACM, New York, NY, USA, pp. 51–60. doi:10.1145/2347635.2347643

Gatt, A., Reiter, E., 2009. SimpleNLG: A Realisation Engine for Practical Applications, in: Proceedings of the 12th European Workshop on Natural Language Generation, ENLG '09. Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 90–93.

Girju, R., Badulescu, A., Moldovan, D., 2006. Automatic Discovery of Part-Whole Relations. Comput Linguist 32, 83–135. doi:10.1162/coli.2006.32.1.83

Grigis, D., Lazzari, M., 2013. Augmentative and Alternative Communication on Tablet to Help Persons with Severe Disabilities, in: Proceedings of the Biannual Conference of the Italian Chapter of SIGCHI, CHItaly '13. ACM, New York, NY, USA, pp. 17:1–17:4. doi:10.1145/2499149.2499175

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H., 2009. The WEKA data mining software: an update. ACM SIGKDD Explor. Newsl. 11, 10–18.

Higginbotham, D.J., Bisantz, A.M., Sunm, M., Adams, K., Yik, F., 2009. The effect of context priming and task type on augmentative communication performance. Augment. Altern. Commun. Baltim. Md 1985 25, 19–31. doi:10.1080/07434610802131869

Higginbotham, D.J., Shane, H., Russell, S., Caves, K., 2007. Access to AAC: Present, past, and future. Augment. Altern. Commun. 23, 243–257. doi:10.1080/07434610701571058

Higginbotham, J., Jacobs, S., 2011. The Future of the Android Operating System for Augmentative and Alternative Communication. Perspect. Augment. Altern. Commun. 20, 52. doi:10.1044/aac20.2.52

Hill, K., Romich, B., 2002. A Rate Index for Augmentative and Alternative Communication. Int. J. Speech Technol. 5, 57–64. doi:10.1023/A:1013638916623

Islam, M.A., Inkpen, D., 2006. Second Order Co-occurrence PMI for Determining the Semantic Similarity of Words.

Johansen, S., Hansen, P., 2006. Augmentative and Alternative Communication: The Future of Text on the Move. Univers Access Inf Soc 5, 125–149. doi:10.1007/s10209-006-0033-0

Johansson, S., Leech, G., Goodluck, H., 1978. Manual of Information to Accompany the Lancaster-Oslo/Bergen Corpus of British English, for Use with Digital Computers.

Jurafsky, D., Martin, J.H., 2000. Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition, 1st ed. Prentice Hall PTR, Upper Saddle River, NJ, USA.

Kay, M., 1984. Functional Unification Grammar: A Formalism for Machine Translation, in: Proceedings of the 10th International Conference on Computational Linguistics and 22Nd Annual Meeting on Association for Computational Linguistics, ACL '84. Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 75–78. doi:10.3115/980491.980509

Kehoe, A., Renouf, A., 2002. WebCorp: Applying the Web to linguistics and linguistics to the Web [WWW Document]. URL http://www.academia.edu/1826323/WebCorp_Applying_the_Web_to_linguistics_and_linguistics_to_the_Web (accessed 10.13.15).

Kent, G.H., Rosanoff, A.J., 1910. A study of association in insanity. [Baltimore, Md., American Medico-psychological Association].

Kiss, G.R., Armstrong, C., Milroy, R., 1973. An associative thesaurus of English and its computer analysis. ResearchGate.

Koester, H.H., Levine, S., 1998. Model simulations of user performance with word prediction. Augment. Altern. Commun. 14, 25–36. doi:10.1080/07434619812331278176

Koester, H.H., Levine, S., 1997. Keystroke-level models for user performance with word prediction. Augment. Altern. Commun. 13, 239–257. doi:10.1080/07434619712331278068

Kristensson, P.O., Vertanen, K., 2012. Performance Comparisons of Phrase Sets and Presentation Styles for Text Entry Evaluations, in: Proceedings of the 2012 ACM International Conference on Intelligent User Interfaces, IUI '12. ACM, New York, NY, USA, pp. 29–32. doi:10.1145/2166966.2166972

Kucera, H., 1967. Computational analysis of present-day American English. Trove.
Landauer, T.K., Foltz, P.W., Laham, D., 1998. An introduction to latent semantic analysis. Discourse Process. 25, 259–284. doi:10.1080/01638539809545028
Leech, G. Garside, R., Bryant, M, 1994. CLAWS4: THE TAGGING OF THE BRITISH NATIONAL CORPUS, in: Proceedings of the 15th International Conference on Computational Linguistics. Presented at the COLING 94, Kyoto, Japan, pp. 622–628.
Levin, T., Scott, B., Borders, B., Hart, K., Lee, J., Decanini, A., 2007. Aphasia Talks: Photography as a Means of Communication, Self-Expression, and Empowerment in Persons with Aphasia. Top. Stroke Rehabil. 14, 72–84. doi:10.1310/tsr1401-72
Liddy, E., 2001. Natural Language Processing. Cent. Nat. Lang. Process.
Light, J., McNaughton, D., 2012. The changing face of augmentative and alternative communication: past, present, and future challenges. Augment. Altern. Commun. Baltim. Md 1985 28, 197–204. doi:10.3109/07434618.2012.737024
Light, J., Wilkinson, K., Drager, K., 2008. Designing Effective AAC systems: Research Evidence and Implications for Practice, in: Proceedings of ASHA 2008. Presented at the ASHA 2008, Chicago, IL.
Manning, C.D., Schütze, H., 1999. Foundations of Statistical Natural Language Processing. MIT Press, Cambridge, MA, USA.
McBride, D., 2011. AAC Evaluations and New Mobile Technologies: Asking and Answering the Right Questions. Perspect. Augment. Altern. Commun. 20, 9. doi:10.1044/aac20.1.9
McCoy, K.F., Bedrosian, J., Hoag, L., 2010. Implications of Pragmatic and Cognitive Theories on the Design of Utterance-based AAC Systems, in: Proceedings of the NAACL HLT 2010 Workshop on Speech and Language Processing for Assistive Technologies, SLPAT '10. Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 19–27.
McCoy, K.F., Pennington, C.A., 1998. Compansion: From Research Prototype to Practical Integration. Nat Lang Eng 4, 73–95. doi:10.1017/S1351324998001843
McNaughton, D., Light, J., 2013. The iPad and mobile technology revolution: benefits and challenges for individuals who require augmentative and alternative communication. Augment. Altern. Commun. Baltim. Md 1985 29, 107–116. doi:10.3109/07434618.2013.784930
Miller, G.A., 1990. Nouns in WordNet: A Lexical Inheritance System. Int. J. Lexicogr. 3, 245–264. doi:10.1093/ijl/3.4.245
Moore, R.C., 2004. On Log-likelihood-Ratios and the Significance of Rare Events. EMNLP.
Murphy, J., Markova, I., Collins, S., Moodie, E., 1996. AAC systems*: obstacles to effective use. Int. J. Lang. Commun. Disord. 31, 31–44. doi:10.3109/13682829609033150
Murray, J., Goldbart, J., 2009. Augmentative and alternative communication: a review of current issues. Paediatr. Child Health 19, 464 – 468. doi:http://dx.doi.org/10.1016/j.paed.2009.05.003
Nelson, D.L., McEvoy, C.L., 1998. USF Free Association Norms: Introduction [WWW Document]. URL http://web.usf.edu/FreeAssociation/Intro.html (accessed 9.2.15).

Pedersen, T., Patwardhan, S., Michelizzi, J., 2004. WordNet::Similarity: Measuring the Relatedness of Concepts, in: Demonstration Papers at HLT-NAACL 2004, HLT-NAACL–Demonstrations '04. Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 38–41.

Pennington, L., Marshall, J., Goldbart, J., 2007. Describing participants in AAC research and their communicative environments: guidelines for research and practice. Disabil. Rehabil. 29, 521–535. doi:10.1080/09638280600902794

Rapp, R., 2002. The Computation of Word Associations: Comparing Syntagmatic and Paradigmatic Approaches, in: Proceedings of the 19th International Conference on Computational Linguistics - Volume 1, COLING '02. Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 1–7. doi:10.3115/1072228.1072235

Rayner, K., 2009. Eye movements and attention in reading, scene perception, and visual search. Q. J. Exp. Psychol. 2006 62, 1457–1506. doi:10.1080/17470210902816461

Russell, W.A., Jenkins, J.J., University of Minnesota, Department of Psychology, 1954. The complete Minnesota norms for responses to 100 words from the Kent-Rosanoff word association test. Dept. of Psychology, University of Minnesota, Minneapolis.

Salovaara, A., 2009. Studying Appropriation of Everyday Technologies: A Cognitive Approach, in: CHI '09 Extended Abstracts on Human Factors in Computing Systems, CHI EA '09. ACM, New York, NY, USA, pp. 3141–3144. doi:10.1145/1520340.1520447

Sennott, S., Bowker, A., 2009. Autism, AAC, and Proloquo2Go. Perspect. Augment. Altern. Commun. 18, 137. doi:10.1044/aac18.4.137

Smith, L.E., Higginbotham, D.J., Lesher, G.W., Moulton, B., Mathy, P., 2006. The Development of an Automated Method for Analyzing Communication Rate in Augmentative and Alternative Communication. Assist. Technol. 18, 107–121. doi:10.1080/10400435.2006.10131910

Smith, M.M., Connolly, I., 2008. Roles of aided communication: perspectives of adults who use AAC. Disabil. Rehabil. Assist. Technol. 3, 260–273. doi:10.1080/17483100802338499

Todman, J., 2000. Rate and quality of conversations using a text-storage AAC system: Single-case training study. Augment. Altern. Commun. 16, 164–179. doi:10.1080/07434610012331279024

Todman, J., Alm, N., 2003. Modelling conversational pragmatics in communication aids. J. Pragmat. 35, 523–538. doi:10.1016/S0378-2166(02)00130-3

Todman, J., Alm, N., Higginbotham, J., File, P., 2008a. Whole utterance approaches in AAC. Augment. Altern. Commun. Baltim. Md 1985 24, 235–254. doi:10.1080/08990220802388271

Todman, J., Alm, N., Higginbotham, J., File, P., 2008b. Whole Utterance Approaches in AAC. Augment. Altern. Commun. 24, 235–254. doi:10.1080/08990220802388271

Trnka, K., McCoy, K.F., 2008. Evaluating Word Prediction: Framing Keystroke Savings, in: Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Short Papers,

HLT-Short '08. Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 261–264.

Trnka, K., Yarrington, D., McCaw, J., McCoy, K.F., Pennington, C., 2007. The Effects of Word Prediction on Communication Rate for AAC, in: Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Companion Volume, Short Papers, NAACL-Short '07. Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 173–176.

United Nations, 2006. Convention on the rights of persons with disabilities [WWW Document]. URL http://www.un.org/disabilities/convention/conventionfull.shtml (accessed 12.21.15).

Volk, M., 2002. Using the web as corpus for linguistic research. Catcher Mean. Pajusalu R Hennoste TEds Dept Gen. Linguist. 3.

Waller, A., 2006. Communication Access to Conversational Narrative. Top. Lang. Disord. 26, 221.

Washtell, J., Markert, K., 2009. A Comparison of Windowless and Window-based Computational Association Measures As Predictors of Syntagmatic Human Associations, in: Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2 - Volume 2, EMNLP '09. Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 628–637.

Wermter, S., Riloff, E., Scheler, G., 1995. Learning approaches for natural language processing, in: Wermter, S., Riloff, E., Scheler, G. (Eds.), Connectionist, Statistical and Symbolic Approaches to Learning for Natural Language Processing, Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 1–16.

Wettler, M., Rapp, R., 1993. Computation of Word Associations Based on the Co-Occurrences of Words in Large Corpora, in: In Proceedings of the 1 St Workshop on Very Large Corpora: Academic and Industrial Perspectives. pp. 84–93.

Wiegand, K., Patel, R., 2012a. SymbolPath: A Continuous Motion Overlay Module for Icon-based Assistive Communication, in: Proceedings of the 14th International ACM SIGACCESS Conference on Computers and Accessibility, ASSETS '12. ACM, New York, NY, USA, pp. 209–210. doi:10.1145/2384916.2384957

Wiegand, K., Patel, R., 2012b. Non-syntactic Word Prediction for AAC, in: Proceedings of the Third Workshop on Speech and Language Processing for Assistive Technologies, SLPAT '12. Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 28–36.

Wojtinnek, P.-R., Pulman, S., Völker, J., 2012. Building semantic networks from plain text and wikipedia with application to semantic relatedness and noun compound paraphrasing. Int. J. Semantic Comput. 06, 67–91. doi:10.1142/S1793351X12400041

Yule, G., Kendall, M.G., 1968. An Introduction to the theory of Statistics (14th ed.), in: An Introduction to the Theory of Statistics (14th Ed.). Charles Griffin & Co, p. 268.

Zipf, G.K., 1935. The psycho-biology of language. Houghton, Mifflin, Oxford, England.

Appendix

NLP pre-processor pseudo code

1. Vocabulary generation for XML and text files
    a. If file is a text file the app performs regex match to split all words (including contracted words) regex expression = "**[A-Za-z]+'?[a-z]**"
        i. All stop words are removed
        ii. All other words are counted using a dictionary <string,int>. If a word is encountered for the first time a new key is added and counter is set at 1. If a word has been encountered before the value is fetched and incremented by 1
        iii. Once vocabulary is created, it is ranked based on frequency (high to low)
    b. If file is an XML file (in BNC format) the application looks for all nodes with tag w.
        i. Each file is stored in array for random access (needed for contracted words)
        ii. Words are trimmed. Whitespace at the end of word is removed
        iii. Search to look for contracted words. BNC format stores contracted words in the following way (word can't as example) :<w> can </w><w> n't</w>. the app looks at the coming word, if it starts with an apostrophe or it is equal to n't it merges it with the current word and skips to the next word
        iv. All stopwords are ignored
        v. Frequency is counted and results are ranked as done with text files
        vi. Sum up the total number of words for each document to be used as total in probability counting.

2. File parsing and co occurrence file generation . For each word in the vocabulary each file is parsed and word pair frequency is counted based on
    a. If file is text file is split with regex as before
        i. Stopwords are removed
    b. If file is XML file is parsed using w tag as before
        i. Contracted words are merged to generate final word list
        ii. Stopwords are removed
    c. Word pair frequency counting using window
        i. Check the position of the word in the current text and determine the left side of the window.
            1. If the number of preceding words is less than the window size, the window left side is the first word in the document.
            2. If the number of preceding words is greater than or equal to the window size window, left side starts at current position minus window size.
        ii. Check the position of the word in the current text and determine the right side of the window
            1. If number of succeeding words is less than or equal to the window size the window right side is the last element in the document.
            2. If number of succeeding words is greater than the window, window right side is current position + window size.
        iii. For each word within window left and window right produce wordpairs, in the format “w1,w2”
        iv. Update pair frequency in the same fashion as word vocabulary counting

Food	Reasons for discarding category	Music	Reasons for discarding category
Agriculture	Not relevant to a discussion about food and drink	Aspects of Music	Introduces sociological terms about music
Beverages	Appropriate	Essays about music	Literary based discussion on music
Cuisine	Appropriate	Music and politics	Introduces political terms
Eating behaviours	Could introduce scientific terms	**Music audio**	Appropriate
Food allergies	Could introduce medical terms	Music books	Introduces names of literary works

			about music
Food and drink by country	Could introduce geographical terminology	**Music by genre**	Appropriate
Food and drink preparation	Appropriate	Music by geographical categorisation	Introduces names of places
Food and drink stubs	Incomplete category	**Music by source**	Repetition of previous categories

Food and drink terminology	Approrpiate	**Music by theme**	Repetition of previous categories
Food and the environment	Introduces environmental terms	Music databases	Introduces names of databases about music
Food awards	Names of awards	Music education	Introduces terms about

			musical education
Food culture	Introduces sociological terms	**Music genres**	Appropriate
Food decorations	Introduces terms related to food decoration	Music history	Introduces historical terms
Food festivals	Introduces names of places and of festivals	Music images	Introduces names of images
Food industry	Introduces names of companies	Music in fiction	Introduces names of fictional works about music
Food law	Introduces legal terms	Music industry	Introduces names of record labels
Food museums	Introduces names of places and museums	**Music libraries**	Appropriate
Food safety	Introduces names of diseases, scientific terms and legal terms	**Music media**	Appropriate
Food related lists	Appropriate	Music	Introduces names of

		organisations	organisations
Food related organisations	Introduces names of organisations	Music people	Introduces names of people in the music industry
Foods	Appropriate	**Music performance**	Appropriate
Gustation	Introduces technical terms	Music psychology	Introduces terms psychological science terms
History of food and drink	Introduces historical terms	Music stubs	Incomplete works
Hunger	Introduces names of places and terms related to political issues	**Music technology**	Appropriate
Nutrition	Introduces scientific terms	Music therapy	Introduces medical terms
Organic food	Appropriate	**Musical**	Introduces technical

		composition	terms
People in food and agriculture occupations	Introduces names of occupations related to food industry	Musical culture	Introduces sociological terms
Serving and dining	Appropriate	**Musical instruments**	Appropriate
Works about food and drink	Introduces titles of works about food and drink	**Musical terminology**	Appropriate
		Music related lists	Appropriate
		Works about music and musicians	Introduces names of fictional works and characters
Transport	**Reasons for keeping/discarding category**	**Sports**	**Reasons for keeping/ discarding**

			category
Energy in transport	Introduces scientific terms and names of fuels	**Categories by sport**	Appropriate
Hijacking	Introduces names of places, terrorist organisations, and perpetrators of Hijcakings	Gender and sport	Not appropriate
History of transport	Introduces historical terms	History of sport	Introduces historical terms
Intermodal transport	Appropriate	Physical exercise	Introduces medical terms
International transport	Appropriate	**Playing field surfaces**	Specific to types of surfaces
Logistics	Introduces names of places and technical terms related to logistics	Politics and sports	Introduces political terms
Military transport	Appropriate	Religion and sports	Goes into specifics of religion

People in transport	Introduces roles related to transport	Sports by place	Introduces names of places
Private transport	Appropriate	**Sport disciplines**	
Public transport	Appropriate	Sports administration	Introduces administrative sports terminology
Shipping	Appropriate to balance the number of terms related to sea/water transport	Sports business	Introduces business related terms
Student transport	Not related	**Sports by type**	
Sustainable transport	Introduces uncommon modes of transport and terms related to fuel and sustainable energy	Sports by year	Duplicate categories
Test tracks	Introduces names of location and terms related to tracks	**Sports controversies**	Goes into specifics of sports controversies
Transport and the	Reason similar to discarding sustainable	**Sports culture**	Could introduce not

environment	transport		related terms
Transport authorities	Appropriate	**Sports events**	
Transport by cargo	Appropriate	Sports instruction	Not relevant
Transport by city	Introduces names of cities	Sports journalism	Introduces journalistic terms
Transport by continent	Introduces names of continents	Sports museums	
Transport by country	Introduces names of countries	**Sports occupations and roles**	Appropriate
Transport by mode	Appropriate	**Sports organisations**	Introduces names of organisations
Transport controversies	Kept to increase vocabulary count	**Sports rules and regulations**	Introduces technical terms
Transport culture	Not relevant	Sports science	Introduces scientific terms

Transport economics	Introduces economic terms	**Sports seasons**	
Transport images	Talks about	Sports sponsorships	Not relevant
Transport infrastructure	Appropriate as it introduces terms related to roads	**Sports squads**	Appropriate
Transport law	Introduces legal terms	**Sports strategy**	
Transport lists	Introduces appropriate terms therefore was kept	Sports stubs	
Transport occupations	Appropriate	**Sports techniques**	
Transport of heads of state	Introduces names of heads of states	**Sports terminology**	
Transport operations	Appropriate	**Sports trophies and awards**	Introduces names of awards
Transport	Introduces names of places and	Sports venues	Introduces names of

organisations	organisations		places
Transport safety		**Sports related lists**	Appropriate
Transport stubs	Incomplete work	Works about sports	Introduces names of books and authors
Transport systems	Appropriate		
Transportation engineering	Appropriate		
Transportation geography	Introduces places		
Transportation planning	Introduces technical terms		
Wikipedia books on transport	Names of books		
Works about transport	Names of books		

List of Data and Software digitally stored for future use

- Recording of meeting with Speech Therapist
- Wikipedia articles
- AAC application APK and source code
- Source code of NLP pre-processor
- Files Generated by the
- Source code of NLG Sentence Generator
- Source code of PHP server to call NLG Server
- Source code of PERL meronym detector
- CrowdFlower data and results
- Hick Hyman simulation data and results spreadsheets