UmobiTalk: Ubiquitous Mobile Speech Based Learning Language Translator for Sesotho Language

Master Dissertation

By

Makhetha John Nyetanyane

Supervised by

Dr. Muthoni Masinde

Co-supervised by

Mr Leon Grobbelaar

This dissertation was conducted and submitted in fulfilment of the requirements of degree

Masters in Information Technology,

At the

Central University of Technology, Free State, South Africa (CUT)

Declarations

I, Makhetha John Nyetanyane student number [redacted], declare that the work in this dissertation is a presentation of my original research work and has been submitted for the award of Magister Technologiae: Information Technology at the Central University of Technology, Free State. Contributions of other people involved, were indicated using the references and acknowledgements. Further, this work was done under the guidance of Dr Muthoni Masinde, Department of Information Technology, at the Central University of Technology, Free State.

Makhetha John Nyetanyane

Date: ______________________

Signature: _________________

In my capacity as supervisor of this dissertation, I certify that the above statements are true to the best of my knowledge.

Date: ______________________

Signature: _________________

In my capacity as co-supervisor of this dissertation, I certify that the above statements are true to the best of my knowledge.

Date: ______________________

Signature: _________________

Abstract

The need to conserve the under-resourced languages is becoming more urgent as some of them are becoming extinct; natural language processing can be used to redress this. Currently, most initiatives around language processing technologies are focusing on western languages such as English and French, yet resources for such languages are already available. The Sesotho language is one of the under-resourced Bantu languages; it is mostly spoken in Free State province of South Africa and in Lesotho. Like other parts of South Africa, Free State has experienced high number of migrants and non-Sesotho speakers from neighboring provinces and countries; such people are faced with serious language barrier problems especially in the informal settlements where everyone tends to speak only Sesotho. Non-Sesotho speakers refers to the racial groups such as Xhosas, Zulus, Coloureds, Whites and more, in which Sesotho language is not their native language.

As a solution to this, we developed a parallel corpus that has English as source and Sesotho as a target language and packaged it in UmobiTalk - Ubiquitous mobile speech based learning translator. UmobiTalk is a mobile-based tool for learning Sesotho for English speakers. The development of this tool was based on the combination of automatic speech recognition, machine translation and speech synthesis.

Acknowledgements

First and foremost, I would like to thank heaven God for strength, wisdom, passionate, love and courage. My sincere gratitude goes to my family (my late father Tsietsi Abrahm Nyetanyane, my brother Nkopane Isaac Nyetanyane and my mother Mankopane Augustina Nyetanyane) with their warmth support and instilling the mind that '*With God I can make It'*. I will also like to thank my supervisor, Dr Muthoni Masinde and co-supervisor Mr. Grobbelaar with their guidance and full support.

My sincere gratitude also goes to migrants and non-Sesotho speakers who took part in surveys, supported my work in this way and helped me get results of better quality, without them this research would have not been successful.

I would like to thank my fellow Masters in IT students for their feedback, cooperation and of course friendship. Specifically, Mpho Mbele whom we built, motivate, learn from each other, and fought together with one mind and one spirit.

Further, I am also grateful to my departmental manager Dr M. Oosthuizen, Mr Kompi and other colleagues with their mutual support and giving me a glimmer of hope that I will achieve it. Moreover, my humble thanks goes to Central University of Technology for giving me clean conducive environment whereby technology and finances needed are on my fingertips.

Lastly, thanks to my friends and other supporters for accepting nothing less than excellence from me.

Table of Contents

List of Figures

List of Tables

List of Appendices

List of Abbreviations and Definitions

ADJ:	Adjective
APK:	Android Package Kit
ASCII	American Standard Code for Information Interchange
ASR:	Automatic Speech Recognition
BCIC:	British Compared of International Corpus
CFG:	Context Free Grammar
CPU:	Central Processing Unit
CUT:	Central University of Technology
DT:	Determiner
GUI:	Graphical User Interface
HCL:	Human Computer Language
HYP:	Hypothesis Document
IDE:	Integrated Development Environment
IWAD:	In-Text Word Ambiguity Detection
LOG:	Logarithms
MT:	Machine translation
N:	Number of items
N-GRAMS:	Number of Grammatical Elements
NLP:	Natural Language Processing
NLU:	Natural Language Understanding
NN:	Noun
NP:	Noun Phrase
OCR:	Optical Character Recognition
P:	Preposition
POS:	Part of Speech
POSS:	Possessive
PP:	Preposition Phrase
PT:	Phrase Tag
QCA:	Quantitative content analysis
QP:	Questioning Phrase

RAM:	Random Access Memory
REF:	Reference Document
RocMe:	Recording of Oral Corpora Made Easy
SADC:	South African Developments Community
SMT:	Statistical Machine Translation
SWAD:	Single Word Ambiguity Detection
TTS:	Text To Speech
UML:	Unified Modeling Language
UmobiTalk:	Ubiquitous Mobile Speech Based Learning for Sesotho Language
VB:	Verb
VP:	Verb Phrase
WER:	Word Error Rate
WSA:	Word Sense Ambiguity
WSD:	Word sense disambiguation
XML:	Extensible Markup Language

Chapter 1: Introduction

1.1 Overview of the study

With over 1.7 million people speaking Sesotho, the language is the most used language in Free State (Census, 2011). South Africa is the largest economy in Sub-Saharan Africa and hence, it has more job/business opportunities compared to other countries in the region. Consequently, the country experiences high rates of migrants (seeking to improve their economic status) to the country and by extension to Free State. This is where language barrier becomes an issue; this is the gap the proposed application aimed to bridge by accommodating translation of Bantu languages that are not supported by widely used Google Translate and other machine translators. Many Bantu languages such as Southern Sotho are under-resourced languages; researchers express them as languages that lack unique writing systems or stable orthography, limited presence on the web and they have a lack of electronic resources for language processing (Besacier et al., 2013).

1.2 Problem Statement

The core research problem is that, despite the advancement and proliferation of speech-to-speech technologies and tools, there is no such a tool (mobile phone based) available for Southern Sotho language. It is envisioned that such a tool would be very useful to foreigners, non-Sesotho speaking as well as people with special needs who are faced with the tremendous problem on how to integrate themselves to Bantu (Southern Sotho) language speaking society here in Free State, South Africa. Although English is seen as an intermediary language to bridge between different races, people living in rural areas in the Free State do not know (speak, read and write) English, hence the Sesotho translating tool would be an asset. Furthermore, to address the research problem above, the following research objectives and questions were pursued:

1.3 Research Objectives

1.3.1 Main Research Objective

The main objective of the project was to develop a mobile speech based application putting in mind functionality, performance, design, usability, and moreover contribute towards the re-education of language barriers between Sesotho speaking and non-Sesotho speaking population in the Free State.

1.3.2 Secondary Research Objectives

The secondary objectives of the project were:

- To critically review literature related to the speech based technology (Speech-to-text, and text-to speech) and language processing tools;
- To identify the corpus elements exhibited in Southern Sotho language and build specialized parallel language corpus for a mobile application that will focus on only basic Sesotho language that migrants and non-Sesotho speakers need to know;
- Develop a speech based application and use it as an analysis tool to asses and evaluate the parallel corpus;
- To evaluate the functionality, performance, design and usability of the developed application.

1.4 Research Questions

1.4.1 Main Research Question

How can we leverage the use of speech-to-speech technology and mobile phone technology to remove language barriers between the Sesotho speaking and non-Sesotho speaking people (such as tourists and foreigners) in the Free State, putting in mind the functionality, performance, design and usability of the technology?

1.4.2 Secondary Research Questions

- What literature available relating to speech based technology and language processing tools?

- How will specialized parallel corpus that will cater for only basic Sesotho language that migrants and non-Sesotho speakers need to know, be implemented for mobile application?
- How will the developed parallel corpus be assessed to reduce the error rate and enhance the translation accuracy?
- How will the functionality, performance, design and usability be evaluated to improve the developed application?

1.5 Research Methodology

Given the nature of the proposed solution, mixed method approach was deemed most applicable because of the need to strengthen the reliability of data, validity of the findings and recommendations, and to broaden and deepen the understanding of the processes through which program outcome and impacts are achieved (Bamberger, 2012). Prototyping was applied in the development of the mobile application system prototype while experimentation, surveys and observation were used to evaluate the functionality, performance, design and usability of the prototype. Given the enormous scope of developing a parallel corpus, quantitative research design was adopted, number of participants were selected from the Free State population using purposive sampling and they were assessed using surveys; for this purpose, only, a selected representation of the Southern Sotho language was modelled and used to develop and evaluate the system prototype. The selected representation was focusing on basic Sesotho words and phrases that migrants need to know.

In evaluating the above mentioned aspects of the system prototype, sample method (involves taking a representative selection of the population and using data collected as research information (Latham , 2007)) was applied. The results obtained from the sample were generalized to the entire Free State population. Purposive sampling was used because the participants have some defining characteristics that make them the holders of the data needed for the study, e.g. foreigners and non-Sesotho speakers that are faced with the problem of integration to Sesotho speaking population.

1.6 Research Contributions

This dissertation proves that the under resourced languages can be processed through the use of mobile speech based technologies. We developed the mobile application that can be used by English speakers to learn Sesotho language, with the aim to bridge the gap caused by language barrier. For speech based technology, we developed the Sesotho speech synthesizer that can read the Sesotho words and it was collaborated with English speech synthesizer to read the English words such as proper nouns.

The machine translator is equipped with NLU unit that is first executed with the aim to understand the morphological, syntactic and semantic analysis of the source language.

The parallel corpus is designed using constituents (syntactically analyzed phrases); these constituents are flexible enough to allow the user to play around with existing words from the machine dictionary to form new sentences that can be translated. The corpus is designed in such a way it can keep up with some of the Sesotho's inflection and derivation of morphemes.

The system is also equipped with the list of over 200 English words and phrases that the user can select from and translate. These words and phrases are sectioned based on their domains such as greetings, small talk and others.

To improve the cost effective of the application, the corpus is stored in a phones memory than on internet. This allows the application to translate words and phrases without accessing the internet.

1.7 Layout of Chapters

In *Chapter 1,* the research problem and objectives were discussed. *Chapter 2* will discuss the Sotho languages, socio linguistic and economic challenges of Sesotho, Sesotho as a resource scarce language, the speech based technologies, the machine translator, characteristics of corpus, different types of corpora, and tools used to assess and evaluate the corpora. *Chapter 3* will focus on how the developed prototype was tested by the respondents, and how data was collected and analyzed. In *Chapter 4,* we will discuss the development of parallel corpus, whereas *chapter 5* will focus on the design and implementation of the Umobitalk which was integrated with the corpus. In *chapter 6*, we present system tests that were performed using a set of experiments to evaluate the translation speed and translation accuracy. Lastly in *Chapter 7*, we discuss the implications that we had during corpus development and further work that can be done on the system.

Chapter 2: Literature Review

2.1 Introduction

In chapter 1, an overview of the research study was presented. From the information presented in chapter 1, it is evident that the study investigates the development and evaluation of the functionality, performance, design and usability of UmobiTalk and contributes towards the re-education of language barriers between Sesotho speaking and non-Sesotho speaking population in the Free State. In chapter 2, Sesotho background, socio linguistic challenges and language as a resource scarce language are discussed. We briefly discuss corpus, its characteristics, different types of corpora and discuss corpus development stages such as corpus collection that focuses on how data (either written or spoken) is collected using existing technologies such as optical character recognition and web crawler for written text, and web crowdsourcing for speech based collection. Secondly, discussion of corpus cleaning stage where corpus is cleaned by removing noisy data that can degrade the performance of the machine that uses the data. Thirdly, corpus annotation focusing on how data is prepared for quick and accurate processing by performing morphological, syntactical and semantic annotation. Lastly, we assess the corpus performance through corpus analysis tools such as frequency list, concordance and collocation, n-Gram and keywords. We also accentuate other corpus analysis tools such as CQPWeb and Google translator that integrates the above mentioned analysis tools.

2.2 Sesotho language

Sesotho or Southern Sotho language is one of 11 official South African Bantu languages. According to statistics, Sesotho language is primarily spoken by 1,717,881 people in the Free State and secondly, is spoken by 1,395,089 people in the Gauteng province (Census, 2011). Southern Sotho, Northern Sotho (Setswana) and Western Sotho (Sepedi) are all derived from Sotho languages and all the speakers are called Basotho (Demuth, 1992). The Sotho languages are closely related to Southern Bantu language such as Nguni languages that comprise of Xhoza, Zulu, Swazi, Hlubi, Phuthi and

Ndebele. Although they are closely related, Faab (2010) classified word division of Sotho languages as disjunctive in contrast with Nguni languages which utilizes the conjunctive way of writing. Disjunctive language is a language in which linguistic units (words) are concatenated together with spaces in between in contrast with conjunctive language which allow no spaces in between such as “thank you” which is “ke a leboha” in Sesotho and “ngiyabonga” for Nguni language (Faab, 2010).

The Sesotho language is considered as a highly morphological language that makes use of numerous affixes to build the complete words (Johnson, 2008). Most of the Sesotho words are formed by the concatenation of morphemes which are known as linguistic units with minimal meaning. Earliest research of Johnson et al. (1999) describe all the Sotho languages as languages with rich morphemes and extensive “pronoun drop” consequently a single word may incorporate subject and object pronoun, and can go through verbal inflection and derivation. Verbal inflection and derivation are based on the addition of affixes (prefix, suffix) to stem word known as an infix in order to build new words from existing stem word (Jackson, 2014).

2.3 Socio Linguistic and Economic Challenges of Sesotho

South Africa is one of Africa’s economic giants thus it acts as a catalyst to motivate immigrants from the neighboring countries, especially those that are facing socio-economic challenges (Sibanda, 2010). Additionally, Gebre et al. (2011) claim that for the past few years, the country has experienced massive number of immigrants from non-South African Developments Community (SADC) countries, and the main motivation for this migration is economic factors. Moreover, conflict and political uncertainties are playing a vital role in the migration of refugees (Idemudia, et al., 2013). These migrants have successfully applied for their entrepreneurial flair in establishing small enterprises, but they are not comfortable due to language barriers (Kalitanyi & Visser, 2010). Gebre et.al. (2011) conducted a research on Ethiopians living in Durban, South Africa, and their findings denoted that these immigrants have opened up many small enterprises targeted at South Africans, however, they are faced with the challenge of integrating themselves in local communities due to language barriers.

In 2011, the Free State province experienced approximately 35,000 migrants coming from outside South Africa with in-migration of approximately 128,000 from different provinces (Census, 2011). The presence of migrants in our country brings to the table the issue of language as a form of communication. Socio linguistic problems are diminished through the use of English as intermediate language, however, most of Bantu language speakers especially elders from rural areas here in Free State do not know English. As such, migrants have to adapt learning the Sesotho language. Probyn (2006) further explained that English is also a problem in Bantu schools where the majority of Bantu learners do not have adequate English proficiency to successfully engage with the curriculum, as a result teachers resort to use the learners' home language for the sake of their understanding. Moreover, language barriers have been extended to the corporate world (Madera, et al., 2011). Foreign born workers are increasingly becoming a vital part of the community and local work force and while many foreign born workers are assimilated into the workforce, others are faced with more challenges to overcome the cultural and language barriers that exist (Madera, et al., 2011).

2.4 Sesotho Language as a Resource Scarce Language

Sesotho language is considered as a resource scarce language in view of the fact that it lacks many lexis in vocabulary and it has few digital resources, and it is a language with very few linguistic experts (De Vries et al., 2014). Researchers express resource scarce languages as languages that lack a unique writing system or stable orthography, limited presence on the web and they have lack of electronic resources for language processing (Besacier, et al., 2013). Ko & Mak (2013) indicate that research efforts on language processing technology such as speech recognition are focusing on fully resourced languages such as English, Japanese, French and many more from developed countries. Language with inadequate resources are difficult to computerize through the use of natural language processing (NLP) because large amounts of data is required to train the current recognizers (Besacier et al., 2013, Ko & Mak, 2013). Mustafa et al. (2016) explained that lack of financial resources, social and cultural support and political uncertainty have hindered the creation of such linguistic resources from scratch. Bantu languages were oppressed during the apartheid era; they were not supported in schools, in businesses or anywhere. Bantu learners were forced to learn only western languages (English and Afrikaans) as their language of instruction.

Although the principles have been attained to conserve the Bantu languages, they are still losing value attributable to the fact that western languages are now dominating promptly (Anodo, 2013). Bantu languages at a present time are becoming unpopular, less economically viable and doomed to lose currency because people and upcoming generations are focusing on learning western languages than their native languages (Anodo, 2013).To act on this problem, Rudwick and Parmegiani (2013) elucidated KwaZulu-Natal as the first province to enforce the learning of isiZulu language by all the schools regardless of the race. This idea will be effective if it can be implemented and integrated on learners' curriculum by all provinces. The results of language's resource inadequacy granted rise to the usage of many loan words from western languages that are now incorporated in Sesotho (Rose & Demuth, 2006).

2.5 Speech to Speech Technology

A speech-to-speech application must have the following components: Automatic Speech Recognition (ASR), Machine Translation (MT) and Text-to-Speech (TTS) (Ebert, 2014 & Hyman, 2014). As shown in figure 2.1 below, ASR receives a source language (input), MT converts a source language to target language (processing) and TTS speak (output) the target language.

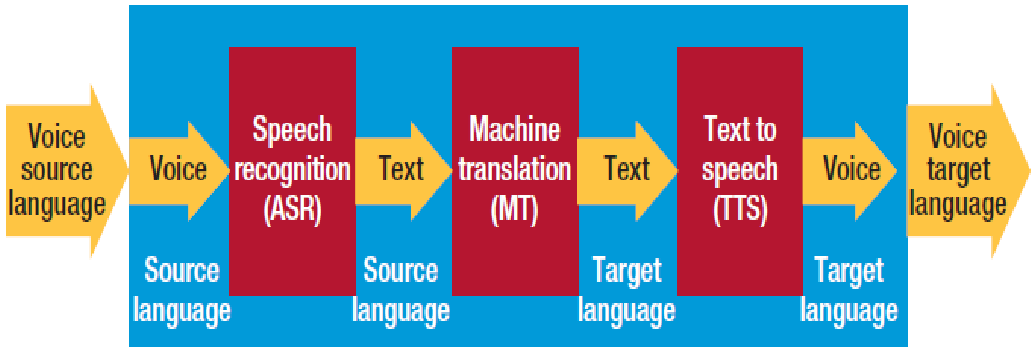


Figure 2.1: A speech-to-speech application's events sequence (Ebert, 2014 & Hyman, 2014).

ASR technology makes life easier because spoken words can be used to communicate with the machine. Kumar *et al*. (2011) state that speech is the easiest way to communicate and is faster than typing and more expressive than clicking. Still on the current topic, recent research of Reddy & Mahender (2013) revealed that speech to text applications can improve system accessibility by providing data entry options for blind, dyslexic, deaf or physically challenged users. The most recent example here is '*Be My Eyes*'; iPhone *app* that lets blind people contact a network of sighted volunteers for help with live video chat (http://www.bemyeyes.org/). Research into speech processing and communication for the most part, was motivated by people's desire to build a mechanical model to emulate human verbal communication capabilities (Anusuya & Katti, 2009). Speech is the most natural form of human communication; ASR has made it possible for computer to follow human voice commands and also understand the human languages.

Evidence from literature demonstrates the fact that, unlike entering input on a fully-sized keyboard; entering text on a mobile device is often sluggish and error prone (Alumae & Kaljurand, 2012). In addition, using a touch screen on small mobile device to input data is time consuming and frustrating. The use of ASR in mobile devices is more effective and flexible than in desktops because they can be used while a person is "on the move" (Kumar, *et al*., 2011). Researchers have suggested that ASR is very important especially for users with low literacy or little script knowledge such as those in the developing regions.

Despite the advantages/justification above, the use of ASR is not yet widely accepted in the society. Many enterprises and business sectors find it difficult and risky to integrate the use of ASR due to its limited offers and susceptibility to errors (Deng & Xuedong , 2004 & Kinoshita et al.,2013 & Hyman,2014). In retailers, the use of ASR is inadequate when using point of sale systems, where computational methods requires full sized keyboard. Feld et al., (2012) explained that unconstrained dictation of a speech remain error –prone, especially when the environment is noisy. Due to mobile portability and ubiquity in contrast with desktops, it is subjected to unconstrained dictation. The use of ASR will be effective in a discreet place. Further, the use of ASR can increase error rate since we have diverse speakers with diverse accents; some of the speeches are not properly articulated (Hyman, 2014). There is hope however, that the problem of accent is getting eradicated. ASR technology has improved by 40% in terms of reduction of word errors (Hyman, 2014). In the case of mobile devices, both ASR and Text-Typing technologies need to be integrated to enable provision of alternative options to users.

2.6 Machine Translation

Machine translation (MT) technology is a process of substituting a source text with a target text, but because of language implications, a well-constructed parallel corpus is essential to handle text and phrase translation processes (Ambati & Vogel, 2010). The viability of machine translator has been tested by several researchers. According to Nino (2009), MT has been seen as an asset that can be used by learners to learn foreign languages. MT has been used by most applications such as UmobiTalk, Google translate, dictionary applications, and many more. Hutchis (2001) expound the idea of using machines to translate started at 1940s and it was seen as an indispensable technology on the basis that it is economically viable compared to human translators, but on the other hand, it was posing a threat to professional human translators as it was taking over. Although the machine translators are quick, inexpensive, always available, and language independent that correlates highly with human translators, they will always have flaws compared to professional human translators (Green et al., 2013).

In Addition, Lin et.al, (2010) argues that machine translators will always have limitations in translation quality and are rarely used to translate documents with high requirements of quality. Thus, trained bilingual personnel will constantly be more advantageous than machine translators. For effective use of machines, users on the other hand should try to use words or sentences that are standard and meaningful that the machine can understand and respond back to them. To obtain an utmost performance and quality of machine translation, collaboration of machines and human performing post editing of language translation is fundamental (Green et al., 2013). The measuring of the machine translator is determined by its closest proximity to a professional human translator**.**

2.7 Google Translate, a Good Example of MT

In 2006, Google launched its pioneering app called Google Translate service which is now supporting up to 70 languages to support monolingual generations (Pollitt, 2014). According to Ebert (2014), Google Translate is an example of a text based machine translator system that applies statistical learning technique to build language and translation models from a large number of texts. Google Translate does an excellent job by using context to determine which is the speakers intended meaning (Hyman, 2014). To further enhance the tool, Google uplifted their standard by developing Google Translate Android app which makes use of speech based technology. On its onset, Google Translate Android application supported only 8 languages but now it supports up to 72 languages from Afrikaans to Yiddish (Hyman, 2014).

2.8 Language Corpus

To build any speech engine whether speech recognition or speech synthesis engine, one needs a corpus which can be described as a collection of pieces of language text in electronic form selected according to external criteria to represent as far as possible a language as a source of data for linguistic research (Jakubicek, et al., 2013 & Kennedy, 2014). Earliest research by Jackson (2005) expound corpus as vocabulary or collection of any text or speech and is used as a basis of statistical processing of natural language. Gries & Berez (2015) supplement an idea of corpus as a domain consisting of machine readable Unicode text files. Over the past few years, corpus has been radically used as an essential tool to help in language learning by non-native learners (Anthoney, 2005). Corpus can be a collection of more than one text, but in context of modern linguistics must cover four important aspects: sampling and representativeness, finite size, machine-readable form, and a standard reference (McEnery & Wilson, 2001 & Kilgarriff & Grefenstette, 2003 & Vinogradov, 2016).

Corpus is meant to be representative for a particular kind of speaker, particular genre, subject field, register (different classes of text such as books, magazine and newspaper (Maekawa, et al., 2014)), variety, or language as a whole meaning the sampling scheme of the corpus represents the variability of the population it is meant to represent (Kennedy, 2014 & Gries & Berez, 2015). Corpus is meant to be balanced, meaning the size of the sub sample (of speakers, registers, variety) are proportional to the proportions of such speakers, registers and varieties in the population the corpus is meant to represent (Gries & Berez, 2015). According to Maekawa et al. (2014), a corpus is balanced if the text collection do not depend heavily on a single class or context such as a newspaper, but rather focuses more on a variety of text classes and data must closely relate to the characteristics of the population that it will mirror.

Corpus building grew rapidly from 1960 to 1980 and corpus usage is not only becoming an important foundation of modern linguistic studies, but as other specialized academic research in the field of medicine, architecture, technology, law, English and other premises (Yang, et al., 2014). Earliest research of Anthony (2005) explains that corpus usage has extended to areas such as translation studies, stylistics, and grammar and dictionary developments. In addition, Graen et al. (2014) explain a corpus as a tool that can be used for many diverse language technology applications such as word sense disambiguation, anaphora resolution, information extraction, statistical machine translation, grammar projection, unsupervised part of speech tagging or learning multilingual semantic translation.

There are different types of corpora (collection of corpus) which are general, specialized, parallel, historical, multimodal, and learner corpus (Vaughan & O'Keefe, 2015 & Farr & Murray, 2016).

2.8.1 General Corpus

General corpus can be spoken (speech corpora) or written corpora aim to provide knowledge for the whole language and is considered as a very large monolingual corpus with millions of words that will be used to match the input (Farr & Murray, 2016). General corpora also known as sample corpora or reference corpora can be used to capture the language variety such as Britain English and American English or Lesotho's Sesotho and South African Sesotho (Vaughan & O'Keefe, 2015). It is a reference corpus in such a way that it can be used as a snapshot of a language collected at a particular point in time (Vaughan & O'Keefe, 2015).

2.8.2 Specialized corpus

Specialized corpus, on the other hand, is restricted to a certain domain and is compiled for a specific purpose and represents a particular context, genre, text or discourse and subject matter or topic (Farr & Murray, 2016). Yang et al. (2014) define specialized corpora as collecting a particular field of corpora to build ideal library collection. Vaughan & O'Keefe (2015) explained specialized corpora as a tool that captures the specific type of language use and dwells on it by using highly contextualized terminologies. Learner corpus is another type of a specialized corpus which focuses on some basic aspects of a language and is used specifically by non-native speakers of a language, represented to facilitate the teaching and learning processes and material.

2.8.3 Parallel Corpora

Parallel corpus is widely used for multilingual translation containing two or more language text samples aligned at sentence level in which one language represents the source and another one represents the target (Paulussen, et al., 2013 & Vaughan & O'Keefe, 2015). This technology is one of the indispensable resources that emerges a wide range of multilingual applications such as machine translation and cross lingual information extraction (Paulussen, et al., 2013). Additionally, parallel corpus can be explained as a valuable resource for cross-language information retrieval and data-driven natural language processing systems especially for statistical machine translation (SMT) (Tian, et al., 2014 & Nakazuwa, et al., 2015). The translation flow can either be unidirectional (one way direction from source to target) or bidirectional (from source to target and vice-versa) (Sundermeyer, et al., 2014).

2.8.4 Historical Corpus

Historical corpus known as diachronic corpus is a corpus that tracks the language and language writing used from centuries (ancient orthogonal) that can be compared with the currently used language with the aim to obtain rational finale on how language evolves (Vaughan & O'Keefe, 2015). For example, the convention of South African Sesotho orthography used in ancient times such as old testimonial bibles (20th centuries ago) is quite different from the one used lately (Demuth, 1992). Lesotho still retains an ancient original orthography while the one used in South Africa has evolved (Demuth, 1992). To prove a need of historical corpus, Hammo et al. (2015) said the development of historical corpus in Arabia assists linguistic and Arabic language learners to effectively explore, understand and discover interesting knowledge hidden in millions of instances of language use. A historical corpus of electronical art music has been successfully developed and now available online from UbuWeb art resource site (Collins, 2015). Despite its flaws in terms of bias whereby male composers are dominant, it provides interesting test ground for automated electronic music analysis in terms of historical and cultural coverage (Collins, 2015).

2.8.5 Multimodal Corpus

Multimodal corpus is a corpus that is done through audio and video recording normally during the discussion meeting (Gries & Berez, 2015). Multimodal corpus includes transcripts that are aligned or synchronized with the original audio or visual recording (Farr & Murray, 2016). Earliest research of Chen et al. (2005) said multimodal corpus is developed based on multimodal communicative behavior and can be recorded through visual display which can be writable such as speech or non-writable such as shoulder orientation, gesture, head orientation, and gaze relate to spoken content. Sign language is a good example of why multimodal corpus is necessary where it represents non-verbal language and non-verbal aspects of the language (Gries & Berez, 2015).

2.9 Corpus Development

A corpus development, irrespective of the category, involves four main activities: corpus compilation or collection, cleaning, annotation and retrieval (Rayson, 2015 & Bosco et al., 2015). A corpus compilation is the process of gathering or collecting text using scanners such as optical character recognition(OCR) which is a technology that converts the scanned images in to readable text (Kichuk, 2015) or sampling from online sources. Second stage is corpus cleaning whereby unnecessary data is removed. Annotation is the third stage where data is made ready for processing by adding tags, codes and documentation that identify textual and linguistic characteristics; and lastly retrieval stage where retrieval tools enable the actual linguistic investigations based on corpora for example frequency analysis, concordances, collocations, keywords and n-grams (Rayson, 2015).

The corpus development stages are shown below.

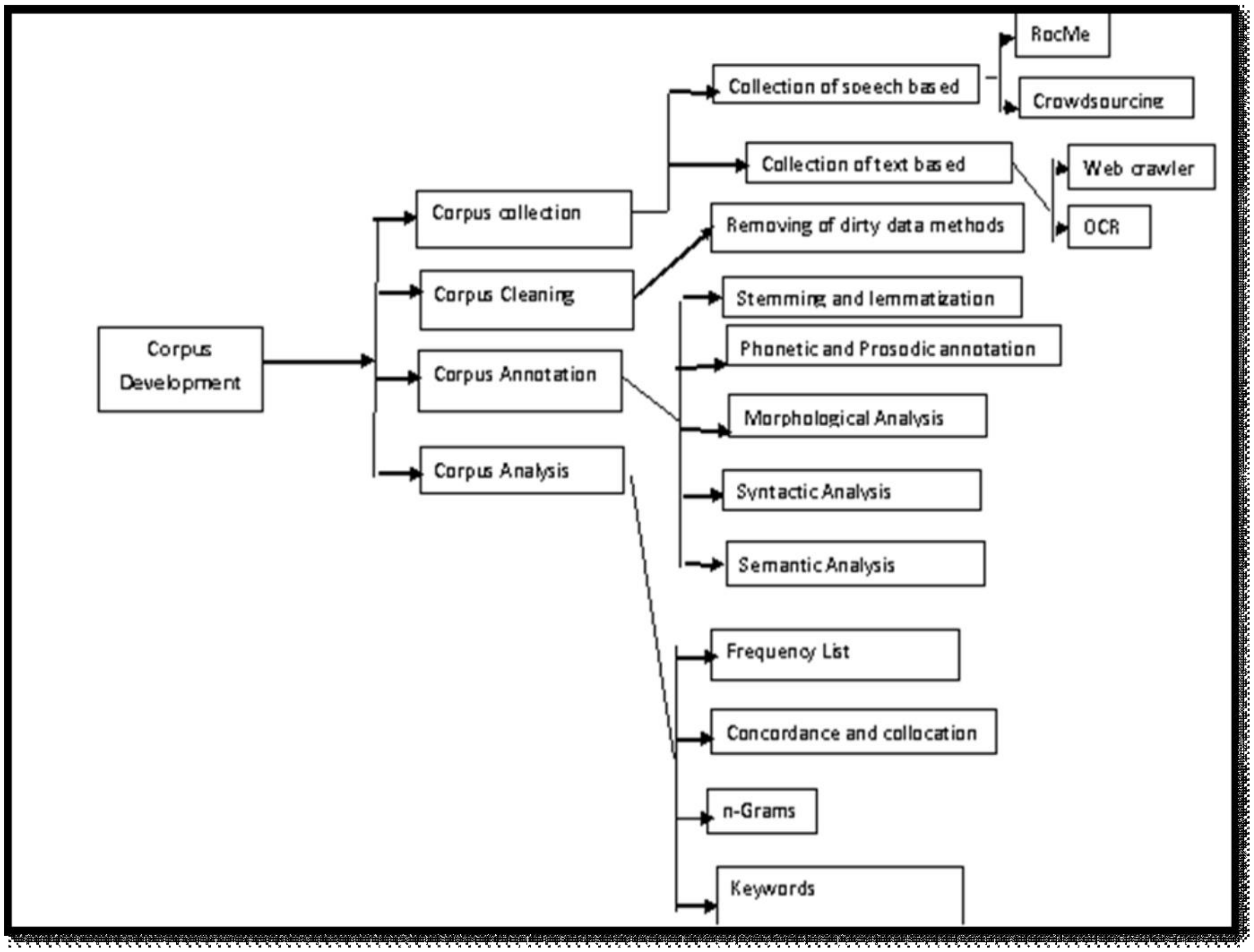


Figure 2.2: Four corpus development phases (Rayson, 2015 & Bosco, et al., 2015).

2.9.1 Corpus Collection

There are different ways to collect data for written corpora; for instance, from web pages using web crawler or from hard copies using OCR. Web crawler is an internet software that is traditionally used to collect data from a world wide web or intranets (Najork & Heydon, 2002). Web crawler acts as middle man between a server and database and is the search program that has direct contact with the database used to manipulate it (Hardie, 2012). A user enters a query into hypertext form from a client computer which will be sent to the server that will call web crawler to process the query and the results will be displayed back to a client in an html form (Hardie, 2012). OCR can be used to scan data from hardcopies such as books (bible, magazines, and newspapers) or any other written media in which their electronic form are no longer available (Kichuk, 2015).

Recordings are utilized as data collection tool for speech corpora. According to Caines et al. (2015), collection of speech corpora is scarcely performed comparable to written corpora, whereas there is a need for speech corpora for engineers working with automatic speech recognizers (ASR) and computational linguistics intending to build natural language processing resources trained on spoken rather than written corpora. Reasons observed for lack of speech corpora are cost and time consuming in performing recordings, transcription and speech annotation which is considered as more expensive and difficult to do (Casacuberta et al., 1991 & Caines, et al., 2015). Another important aspect that degrades the implementation of speech corpora is the implementation of machines that has to perform speech processing steps such as understanding the language phonetics, acoustics, intra, inter speaker, detect and remove background noise, integrate pronunciation variation detection algorithms and many more(Casacuberta et al.,1991). Some difficulties for speech recognition technologies include language linguistics, dialects, diacritics and many more.

Ferragne et al. (2013) presented Recording of Oral Corpora Made Easy (RocMe!) which is a web application designed to allow a sensible autonomous and dematerialized management of speech recordings. RocMe is an example of speech corpus collection via crowdsourcing (crowd of people who will perform the required job on internet). RocMe ease the collection of recordings for speech corpora by performing the following tasks: It provides the users on internet with series of sentences on a screen and let autonomously recording of those sentences and lastly collect the speaker metadata with fully customizable extensible mark-up language (XML) questionnaires (Ferragne et al., 2013). The problem with this method is, instead of enabling crowd workers to perform only transcription from the given set of audios, it provides crowd workers with the set of transcriptions and permits them to perform recording from those transcriptions (Ferragne et al., 2013). This idea can be seen as a compromise of quality audios because it is uncertain that crowd workers are well equipped with quality recording materials and discreet environment to perform recordings. On a road to ease the implementation of speech corpora, Caines et al. (2015) presented another method for speech corpus collection via crowdsourcing. Firstly, they collected two corpora by involving sample of native speakers of English and German-English bilinguals responding to questions based on business topics. Then they distribute both collected corpora through crowdsourcing for transcription purposes and results proved this method to be cost and time effective. However, accuracy level degradation and high level of flaws can be experienced because crowd workers can cheat especially when they are not compensated to do the job (Caines, et al., 2015).

2.9.2 Corpus Cleaning

The collected data for corpus must be cleaned to be valuable for usage by the applications, therefore this is an integral part of data processing and maintenance to reach error free data. Data cleaning process has been intrigued by the real world's dirty, incomplete and noisy data (Singhal & Jena, 2013). Incomplete in lacking attributes values, attributes of interest, noisy in terms of containing errors or outliers and inconsistent containing discrepancies in names and codes (Singhal & Jena, 2013). Data cleaning for text corpus is to remove duplicates, special characters or delimiters known as data anomalies that can pose danger to the application that will use the data.

Nakazuwa et al. (2015) explain that a clean parallel corpus is demanded by the machine translators to provide an outmost translation results. Earliest research by Hernandez & Stolfo (1998) explain that without accurate identification of duplicate information, frequency distribution, and various other aggregations will produce false or misleading statistics leading to untrustworthy results leading to loss of exposure, trust and reputation by companies using such data. Data cleaning algorithms can be set to detect and remove unknown words (words not in machine dictionary), duplicates, white spaces, articles and others. Error detection, validation and cleaning methods are normally used to automatically clean large amount of data such as thousands or millions of data especially the legacy data from museum and herbarium collected over the last 300 years (Chapman, 2005).

Process of data cleaning differ based on type of corpora implemented, Neunerdt et al.(2015) explained that web page cleaning is one of the most essential tasks in web corpus construction. The main aim of performing web page cleaning is to separate the content from boilerplate which is a definition given to comments, navigational elements, templates and advertisements which are out of interests for the corpus (Neunerdt, et al., 2015). Even though tools are used to perfect the corpora, most of huge corpora still have flaws that can be noticed in a long run and these flaws failed to be detected by the machines or the language experts during compilation process (Nakazuwa, et al., 2015). Nakazuwa et al. (2015) presented a method called three step crowdsourcing procedures that can quickly detect the flaws from huge parallel corpora and correct them over the internet. This method proved to be easy and cheap in contrast with other existing methods by allowing crowd workers to firstly detect the flaws, edit the flaws and lastly validate the edits for assurance of quality (Nakazuwa, et al., 2015).

In addition to corpora flaws, Graen et al. (2014) noticed general recurring errors in a current version of well-known Europarl corpus. Graen et al. (2014) said these errors are originating from the website of the European parliament and during corpus compilation stage. The aim was not just to use a crawler to crawl through the parliament web pages to detect and fix errors, they also stored a new version of edited corpus in an XML format which facilitates a more sophisticated selection of data than the original plain text file based corpus (Graen, et al., 2014). This new approach is also called corpus indexing which tokenizes a sentence in to elements and indexed starting from zero as the first position (Hardie, 2012). This approach will enable quick search of information by allowing algorithm to locate matches for a query without searching sequentially through all the text of the corpus (Hardie, 2012). Another approach that can be used for quick information extraction, is text sorting where by the text is placed in a certain order (Yang, et al., 2014).

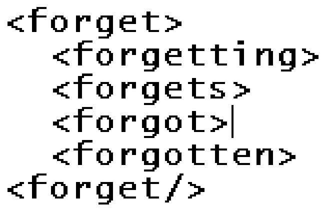
Despite the effectiveness of the machines used to clean the data, human post cleaning is quite substantial which is performed by the language expert to identify and rectify anomalies machines failed to detect.

2.9.3 Corpus Annotation

Corpus annotation can be explained as practice of adding interpretative linguistic information to a corpus with the purpose to organize and prepare it for quick machine processing (Leech & Smith, 2005). General rational behind annotating corpus is to understand the given language production by automatically assigning the linguistic analysis to a given sentence (Dickinsin & Lee, 2009). According to Gries and Berez (2015), annotation is preceded by the initial step called tokenization whereby a string of words is divided in to units or words that will be annotated.

2.9.3.1 Lemmatization and Stemming

Lemmatization is an example of annotation which is a process of identifying and marking each word in a corpus with its base; this is a process of stripping away inflection morphology on words such as verbs so that all forms of lemma “forget”- forgotten-forgets-forgetting-forgot-will represent a root word “forget” (Gries & Berez, 2015). Lemma is set of lexical forms having the same stem and belonging to the same word class but differ only in inflection (Hussein, 2015). The explanation of lemmatizing from Gries & Berez (2015) and Hussein (2015) can be illustrated using the xml on figure 2.3.



```
<forget>
  <forgetting>
  <forgets>
  <forgot>
  <forgotten>
<forget/>
```

Figure 2.3: illustration of lemma and its inflections

Another step is called stemming which is a pre-processing step in natural language processing that is used for quick information retrieval by truncating any form of affix, normally suffix of inflected or derivational words to represent as far as possible the base, for example a root word “forg-“ will represents forg-etting, forg-otten, forg-ets and so on (Jivani, 2011 & Gries & Berez, 2015). Search and indexing systems such as text mining applications, NLP systems and Information retrieval systems are now capitalizing the use of stemming as their most important feature (Kadri & Nie, 2006 & Jivani, 2011). Both stemming and lemmatization are used for the same purpose; they reduce morphological variants of a stem or lemma, however, a huge difference between the two is that in stemming, the 'stem' is obtained after performing set of rules without bothering about the part of speech of the context of the word occurrence, whereas lemmatizing involves a complex process of first understanding the context then determine the word's part of speech and finally finding the lemma (Jivani, 2011).

A problem with stemming is that these derived verbs "ate" and "ran" cannot be grouped to the same base "eat" and "run" due to the fact that they are different from their base structure, thus a need to use lemmatizing is substantial (Korenius et al., 2004 & Jivani, 2011). Stemming is quick and easy to do because it does not need any form of a dictionary; however, the precision lacks (Jivani, 2011). Some of the errors involving the use of stemming are over-stemming occurring when two words with different stem are stemmed to the same root known as false positive and another error is under-stemming occurring when two words that must be stemmed to the same stem are not known as false negative (Jivani, 2011 & Klinkmuller et al., 2013).

2.9.3.2 Phonetic Annotation

Phonetic annotation is an annotation of phonemes known as smallest speech sound produced by the articulated words (Safari & Nouza, 2015 & Gries & Berez, 2015). It takes in to account the duration of the message, the tone and the pitch, and the pronunciation of words (Gries & Berez, 2015). This operation is considered as extreme labor intensive task and as such there should be a proper balance between the expected outcomes and resources utilized (Casacuberta et al., 1991 & Bonaventura et al., 2000 & Caines, et al., 2015.

2.9.3.3 Prosodic Annotation

This is an annotation that is done on a spectrum, by analyzing change of pitch, juncture, nasalization and voicing (Arvaniti & Baltazani, 2005 & Gries & Berez, 2015). Given the speech signal, the labeler has to determine the word and phrase boundaries together with the pauses. Once a boundary is determined then they have to decide on a type of annotation to use.

2.9.3.4 Part of speech tagging

Earliest research by Armstrong et al.(1999) explain part of speech(POS) tagging as a process that consists of assigning to a word its disambiguated part of speech tag in a sentential context in which the word is used. Part of speech tagging known as primordial stage of language processing involves assigning each tokenized word a label that minimally identifies the part of speech of the word (Gries & Berez, 2015). It can also be explained as a simple process of tagging or labeling or clustering data to their specific data structures based on their grammatical properties such as grouping of nouns to noun tags. Part of speech tagging is considered as an essential preprocessing task in many NLP technologies followed by syntactic analysis and semantic disambiguation (Martinez, 2011). Part of speech tagging known as morphological annotation (McDonel, et al., 2014), is one of the most frequent and most exploited kinds of annotation because of its relevance to many corpus linguistic studies and it fits well with other form of annotation such as lemmatization, syntactic parsing and semantic annotation (Gries & Berez, 2015).

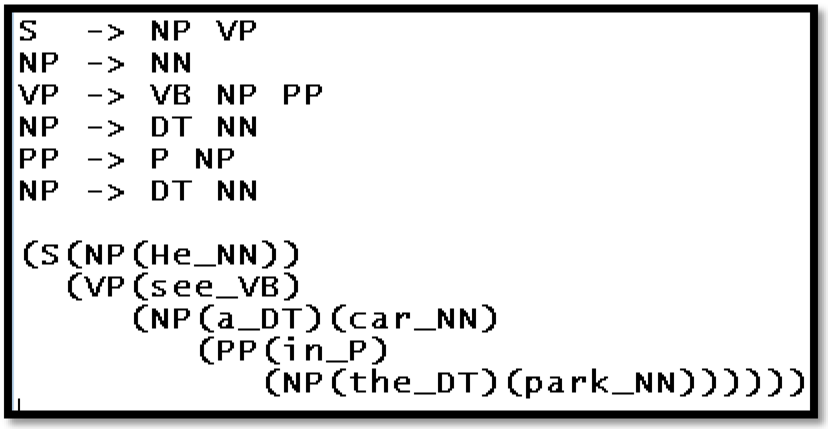
Part of speech taggers are machines that are used to reduce the human involvement by automatically annotate the huge corpora and these state-of-the-art machines have become more reliable with 96-97% accuracy (Martinez, 2011). Machines trained to perform part of speech tagging are expected to yield an accurate results, however, factors that are contributing to the precision of automatic annotation by the machines include; the morphological production of the language, especially highly inflected and derived morphological languages such as Sesotho that pose a challenge to taggers to understand the language and its implications (Armstrong, et al., 1999). Additionally, latest research of Gries and Berez (2015) explain some factors that have to be considered when training taggers are language represented by the corpus and its morphological characteristics (derivational and inflected words), complexity of the text in the corpus, the kind of tagger used (symbolic or statistical), the size, balance of corpora, the representation, the cleanness of corpora and many more.

2.9.3.5 Syntactic and Semantic Analysis

Since the POS annotation is considered as primordial step for almost linguistic processing tasks, syntactic analysis or parsing is the second stage (Lee, 2004 & Martinez, 2011 & Gries & Berez, 2015). These two linguistic annotation layers (POS and syntactic annotation) are used together to improve the exploitation of annotated corpus which will smooth up natural language processing (Paulussen, et al., 2013).

Syntactic parsing has evolved from symbolic approach to statistical approach that assign the most probable syntactic analysis, and its probability is determined on the basis of training corpus(supervised learning) or entirely data driven process(unsupervised learning) (Gries & Berez, 2015). The results of syntactic parsing come in a form of phrase structure or parse tree representation (Leech & Smith, 2005 & McDonel et al., 2014 & Gries & Berez, 2015). Well known corpora such as British Compared of International Corpus (BCIC), Penn Tree Bank, newly developed Tiger Corpus are examples of phrase structured parsed corpus (Gries & Berez, 2015).

Implementation of syntactic parser must have dictionary entries together with the grammar rules that govern how the sentence is parsed for language processing (McDonel, et al., 2014). Syntactic parsers are composed of context free grammar (CFG) also called phrase structure rules used to parse the sentence in to phrase constituents that can be understood by the machine. One of the main phrase structure rule is S→NP VP whereby S denotes sentence or starting point, NP denotes noun phrase and VP denotes verb phrase (McDonel, et al., 2014, Gries & Berez, 2015). McDonel et al. (2014) made example of a sentence '*He see a car in the park'* which can be illustrated using context free grammar rules and how it is presented in a corpus for machine readability.



```
S   -> NP VP
NP  -> NN
VP  -> VB NP PP
NP  -> DT NN
PP  -> P NP
NP  -> DT NN

(S(NP(He_NN))
  (VP(see_VB)
     (NP(a_DT)(car_NN)
        (PP(in_P)
           (NP(the_DT)(park_NN))))))
```

Figure 2.4: Illustration of context free grammar rules and how sentence is parsed

The syntactic parser to its utmost performance must conceive disambiguation module of the semantic analysis that is used to disambiguate words known to have more than one grammatical functionality known as word sense ambiguity(WSA) and other form of ambiguities (McDonel, et al., 2014). Additionally, syntactic and semantic layer are used together to disambiguate strings with metaphors (Montemagni, 2003 & Gries & Berez, 2015).

Even though machines such as parsers can be used to annotate, human post- annotation by expert of a language is quite essential to correct the errors failed to be corrected by the parser (Ishita, et al., 2015). Although human annotation on its own(without machine's preprocessing) in a huge corpus is time consuming, Sabou et al.,(2014) presented a quick and simple method called human annotation through crowdsourcing by employing a group of linguistics to perform annotation on internet and the results' quality and accuracy outperforms the machine annotation.

All steps of language processing are now augmented by the machines such as data collection, data cleaning, data annotation and data analysis but post processing or evaluation stage of each step must be carried by the human to detect all errors failed by the machine (Ishita, et al., 2015).

2.9.4 Corpus Analysis

Corpus Analysis is a stage in which tools to exploit the power, effectiveness and usability of a corpus, and at the same time examination and evaluation of tools exploiting the corpus are effectively used (Hardie, 2012). Annotated corpora are useful in training and testing of machine learning statistical tools and the results are strongly influenced by quality and quantity of the corpus (Bosco, et al., 2015).

Tools that are used to find corpus errors include error detection, quality control techniques and many more (Bosco, et al., 2015). Most of these errors result from corpus annotation (Bosco, et al., 2015). Main corpus analysis tools that will be later explained include frequency analysis, concordances, collocations, keywords and n-grams (Rayson, 2015). To reduce the cost of using some existing tools to monitor and evaluate the corpus, one can develop software that will exploit the corpus to disclose the anomalies (Bosco, et al., 2015). As long as the developed software can effectively and efficiently performs concordance, collocation, queries and other critical analytical procedures (Hardie, 2012).

Another way to evaluate the reliability of the data is to compare the machine results with the results from human annotation, using algorithms such as word error rate(WER) and position independent word Error Rate(PERWER) (Popovic & Ney, 2007 & Bosco, et al., 2015).

The different types of corpus analysis are explained below.

2.9.4.1 Frequency List

Frequency list allows the user to load the corpus and investigate basic frequency patterns that show which words occurring regularly also known as high frequency items or less frequently known as low frequency items (Vaughan & O'Keefe, 2015). Word frequency is calculated as a number of occurrences of a word in a corpus (Hussein, 2015). Anomalies can be detected by having inappropriate occurrence of the target value; such as Biology specialized corpus that do contain less or none of word “photosynthesis” or some other main biological contextual terminologies. Frequency list is considered as first entry point in corpus analysis tools (Vaughan & O'Keefe, 2015).

2.9.4.2 N-Grams

N-Grams known as clusters, chunks or lexical bundles are a collection of words forming phrases in which their frequency can be measured (Vaughan & O'Keefe, 2015). N-Gram is one of examples of text classification approach and can also be explained as a number of elements in a grammatical order forming a string (Khreisat, 2006) or grammatical pattern of words (Anthoney, 2005). N-Gram shows how strings are morphologically and syntactically analyzed and moreover defining the syntactical dependency of texts, making it easy to identify any language grammar errors (Kanerva, et al., 2014).

2.9.4.3 Concordance and collocation analysis

Concordance and collocation analysis allow the researcher to empirically find the co-occurrence of the words in a corpus and how the co-occurrence affects their meaning (Vaughan & O'Keefe, 2015). Concordances are searched words with their own context in which they occur (Hussein, 2015). This context refers to words surrounding searched text normally from the left to right of the searched word (Hussein, 2015). These accompanying or surrounding words are known as collocation (Hussein, 2015).

Collocation provides a clear insight especially to the non-native learners on how a searched word can be used because it is displayed on more than one context, for example word “mouth” was searched using Dracula corpus and 38 occurrences were displayed in which some refer to the mouth of the river, harbor and bank instead of only the “mouth” as a body part (Vaughan & O'Keefe, 2015). According to Hussein (2015), concordance analysis extended its flexibility by housing search of lexical forms of words(lemma) therefore a searched word such as eat, will be displayed together with its linguistic word family such as *ate, eaten, eating,* and *eats*.

2.9.4.4 Keywords

Keywords can be used to list the words with high or low frequency list (Vaughan & O'Keefe, 2015). This approach can suggest the area for investigation based on the words frequency (Anthony, 2005). Keywords are words that draw attention for investigation and they seem to have abnormal frequency (higher or lower frequency) than what is expected (Hussein, 2015). Keywords can be classified using the keyness ranging, from the highest known as positive keywords list to the lowest known as negative keywords list (Hussein, 2015).

2.9.4.5 Google Translate as a Corpus Analysis Tool

Google translate is one of the machine translators used to overcome the language barrier. It is considered as corpus analysis tool that trains parallel corpora of about 70 languages (Banea, et al., 2008 & Pollit, 2014). Since there are no perfect corpora, language linguistics around the world can exploit language corpora using Google translate to locate, submit and document translation errors to Google translate developers (Herbert, et al., 2011). This approach will enhance the correctness of language corpora evaluated.

2.9.4.6 Other Analysis Tools

Some of the other tools that are used for corpus analysis are CQPWeb, Flax, Antcock and Concordancer. CQPWeb is a web based corpus analysis system that strives for equality between the power, usability and flexibility of the corpus (Hardie, 2012). Power in a sense that a tool can efficiently query a large corpus without compromising on speed and accuracy of the results (Hardie, 2012). The power of a corpus is based on its annotation which can allow for complex and sophisticated extraction of data such as morphological, syntactic and semantic extraction (Sinclair, 2004 & Hardie, 2012). Usability of the program that is used to query the corpus must at least have a GUI interface in which the user can interact in a traditional way (Vinyals & Friedland, 2008 & Hardie, 2012).

Flax is similar to CQPweb, it is an application introduced by Wu and Witten (2016) which can be used by learners to seek the language pattern by understanding the vocabulary, grammatical pattern and collocation.

Antcock is a freeware internet tool that is used to aid non-native learners in a classroom and comprise of powerful concordance, word and key word frequency generators, tools for cluster and lexical bundle analysis and word distribution plot (Anthoney, 2005).

Concordancer is one of the software tools that are normally used through website to help users search, access and analyze language in a corpus (Wu & Witten, 2016)

2.10 Conclusion

In this chapter we discussed the Sesotho language, socio linguistic and economic challenges of Sesotho, and Sesotho language as a language with insufficient resources. Furtherly, we discussed technical aspects such as speech to speech technologies, the machine translator, different types of corpora, how the corpora are implemented and the tools that are used to assess the corpora. The next chapter dwells on the research methodology.

Chapter 3: Methodology

3.1 Introduction

In this chapter, the mixed method approach and experimental research design are described. We also look at the geographical area where the research was conducted. The study population and sample are described, as well as data collection instruments including the methods that maintain validity and reliability of each instrument. Finally, data analysis is presented.

3.2 Research Approach and Design

The mixed method approach was followed where both quantitative and qualitative research methods are combined. According to Terrel (2012), mixed methods studies are products of the pragmatist paradigm and that combine qualitative and quantitative approaches within different phases of the research process. For quantitative research, described as a good explanation of the phenomena by collecting numerical data that are analyzed using mathematically based approach(Muijs,2010), the descriptive surveys were used, the surveys were comprised of both open and closed ended questions. For qualitative research, explained as a research that relates to understanding some aspects of social life, and its methods which generate words rather than numbers as data for analysis(Patton and Cochran, 2002),the prototype, experiments and observation were used.

Research design is a plan structure and strategy of investigation so conceived as to obtain answers to research questions or problems (Muijs, 2010). Under research design, the UmobiTalk was developed and it was tested by the group of respondents (sample). During the testing session, the researcher was experimenting the four variables of the system. These four variables were identified as a very important core aspects of UmobiTalk, so to reach all the targeted users' level of satisfaction irrespective of users' technological background or health condition.

- Usability: How easy it is for the user to accomplish the task without outside intervention.

- The design: How easy it is to read and understand the graphical user interface components such as buttons, labels, and textboxes.
- Functionality: Can the system recognize and translate the desired words and phrases without any inconveniences
- The performance or the speed: How fast it is to install the application, and start to operate it (recognize the speech, translate it and display the output). How the application performs on low level specifications mobiles. Under specifications, random access memory (RAM), central processing unit (CPU), and storage were reviewed.

3.3 Research Setting

Research setting refers to the location where the data was collected. In this instance, the study was conducted at Free State province (Welkom and Bloemfontein). From the selected sample, 15 individuals were residing in Welkom, while another 15 individual were from Bloemfontein.

3.4 The Study Population and Sample

Study population is the study of all elements (such as objects, individuals and events) sharing the same characteristics, such as age, gender, or health condition. In this research, the study population consisted of the individuals residing in Free State province. From this population, purposive sampling was exploited in consequence of participants retaining some defining characteristics that make them the holders of the data needed for the study. The purposive sampling technique also known as judgement sampling, is the deliberate selection of participants based on characteristics they possess (Tongco, 2007). A convenient research sample of 30 subjects was selected. This sample comprised of foreigners and non-Sesotho speakers that are faced with the problem of integration to Sesotho speaking population. The researcher physically contacted the respondents and he identified them by the color of their skin, accent and moreover their vulnerability when they failed to hear and respond back to the researcher as he was communicating with them in Sesotho language. Most of these respondents were from rural areas such as G Hostel (An area known to have many foreigners in Welkom). The respondents were selected based on their nationality, gender, age, workstations, literacy level and health condition. In terms of nationality, the sample was comprised of 10 blacks, 5 whites, 5 Indians, 5 Coloureds, and 5 Chinese. There was 15 females and 15 males ranging between the age of 18 and 75 years. Among the participants, others were students, business owners, academics, athletes and unemployed. Some of the respondents especially elders were technologically illiterate, while others were disabled.

The respondents were given a specific date, time and venue to come and test the software. The researcher was always contacting the respondents to remind them of the meeting. The researcher also provided the transport allowance to ensure that all the respondents make it on time without costs.

Once the sample was determined, the data collection instruments were used.

3.5 The Data Collection Tools

A questionnaire was chosen as a first data collection instrument. This instrument is explained as a document with series of questions (open and closed ended questions) for the purpose of gathering information. The majority of the questions from the questionnaire were closed ended questions, which were advantageous to both researcher (in terms of easier to analyze) and the respondents (in terms of quick to fill in and they are straight to the point). The open ended questions were used to procure a briefly explanation of the respondent opinions about the aspects of the application; they are useful in a sense that, they allow the respondents to respond to questions in their own words and they provide wide variety of details.

The questionnaires were written in English language, in accordance with subjects who can speak and write English. Moreover, the researcher was also available at all times to explain any questions that might pose uncertainty to the subjects. The respondents were given a maximum time of an hour to test the prototype (Install the Android Package Kit (apk), run and test the application on their phones) and a maximum of an hour to fill in the questionnaires. These time periods were decided by the researcher to accommodate all the participants including those that are technologically illiterate and those who are disabled.

The questionnaires were divided in to three sections, which are system design and usability, system functionality and system performance (speed).

The observation was a second instrument that was used, as the participants were testing the application, the researcher was circulating with a pen and a research diary to document some issues that users perceive do.

3.6 Reliability and Validity

The researcher was the only source to deliver and receive the questionnaires from respondents. The researcher ensured that questionnaires are filled and returned back to him. The researcher ensured that there was a space between each respondent to avoid any form of interaction between them. The respondents were motivated to be as honest as possible so to improve the prototype.

The environment chosen for respondents was conducive. The place was clean, air conditioners were available, the door was securely locked with a notice “do not disturb” to prevent outside interruption. The language used in questionnaires was at its simplest form.

3.7 Data Analysis

The data from surveys was collected and analyzed, both quantitative and qualitative. The open ended questions were analyzed through quantitative content analysis (QCA) performed by the researcher with the aim of quantifying close related or emerging characteristics and concept. The QCA is a measuring tool used to cluster close related facts and count their frequency.

3.7.1 Statistics Based on System Functionality

Question1 in the study asked the subjects their opinion in terms of application's usefulness, and the results were depicted on a figure 3.1 below.

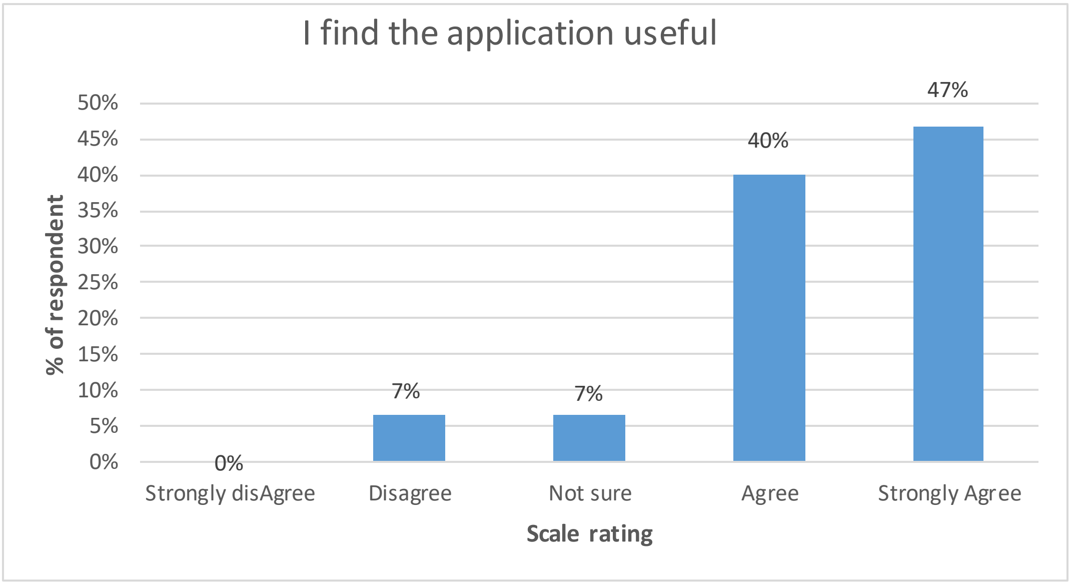


Figure 3.1: Presentation of results from the first question of system functionality

As depicted above, none of the respondents strongly disagreed, while 7% disagreed, 7% of the participants were not sure, while 40% agreed and 47% strongly agreed.

Question 2 of the system functionality, was also a close ended question where the participants were asked if they could complete the tasks smoothly without any inconvenience such as errors or exceptions that may halt the system functionality.

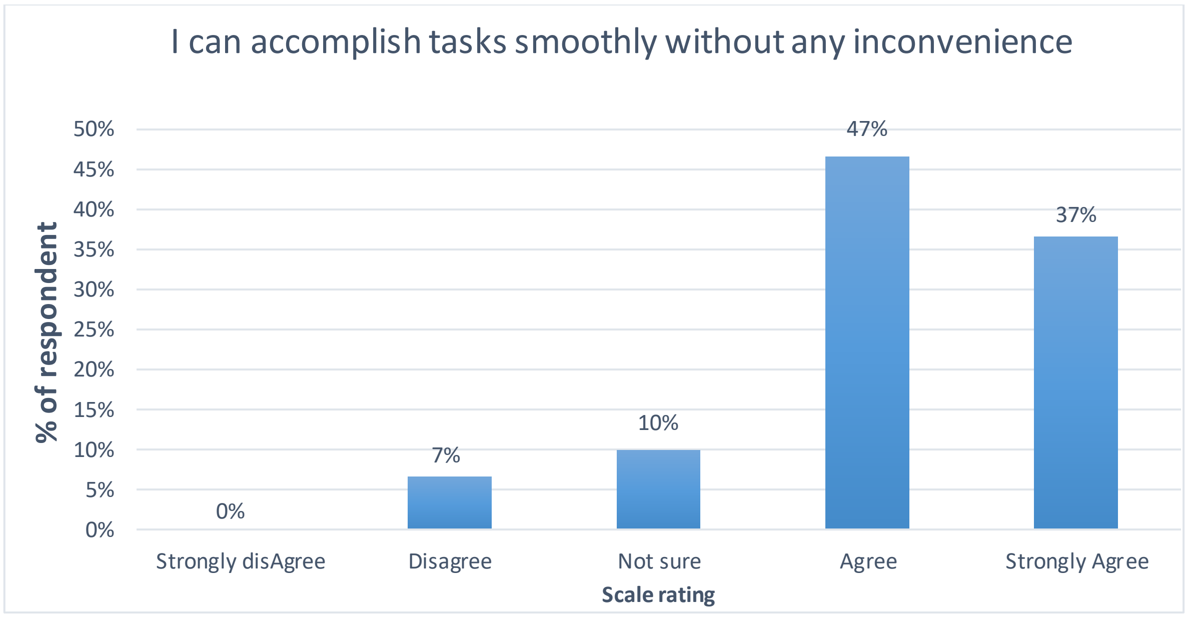


Figure 3.2: Results from the second question of system functionality

According to the graph above, none of the respondents strongly disagreed, while 7% disagreed. 10% of the respondents were not sure, while 47% of them agreed, and 37% of the respondents strongly agreed.

The researcher required to find out the respondents' satisfaction of the translation results. Hence, a closed ended question was posed to the respondents and as depicted on the graph below, none of respondents were strongly dissatisfied, while 27% of the respondents were dissatisfied, 10% of them were not sure, while 33% were satisfied and 30% of the participants were strongly satisfied.

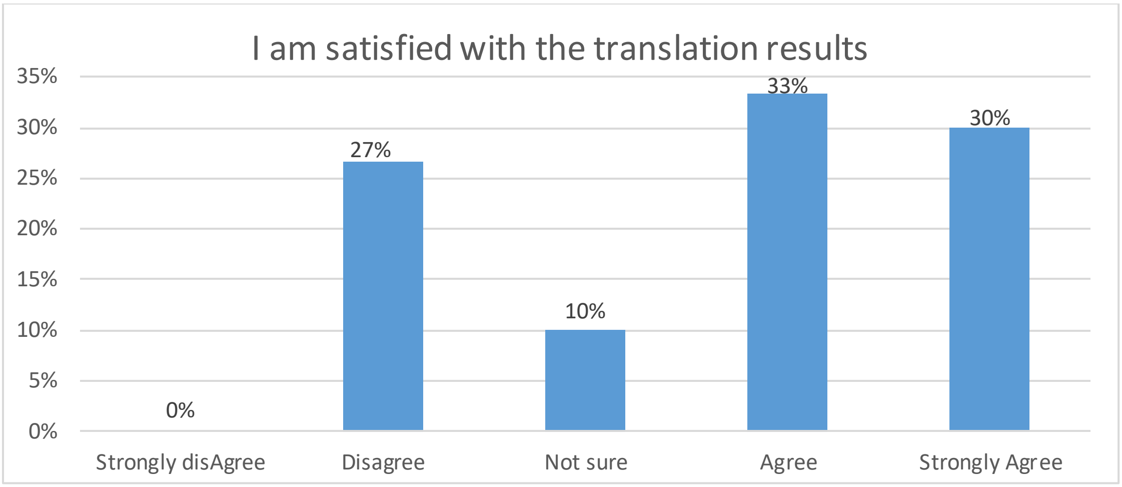


Figure 3.3: Results from the third question of system functionality

On the ground that the application's machine translation is the core component of the evaluated prototype's functionality, the researcher needed a brief explanation by posing an open-ended question on what translation errors or problems users came across while interacting with the system. According to the chart below, 27% of the respondents said application failed to translate long sentences, while 27% of the respondents said it cannot translate their desired words and phrases, and lastly 47% of them were not sure of the correctness of translated texts.

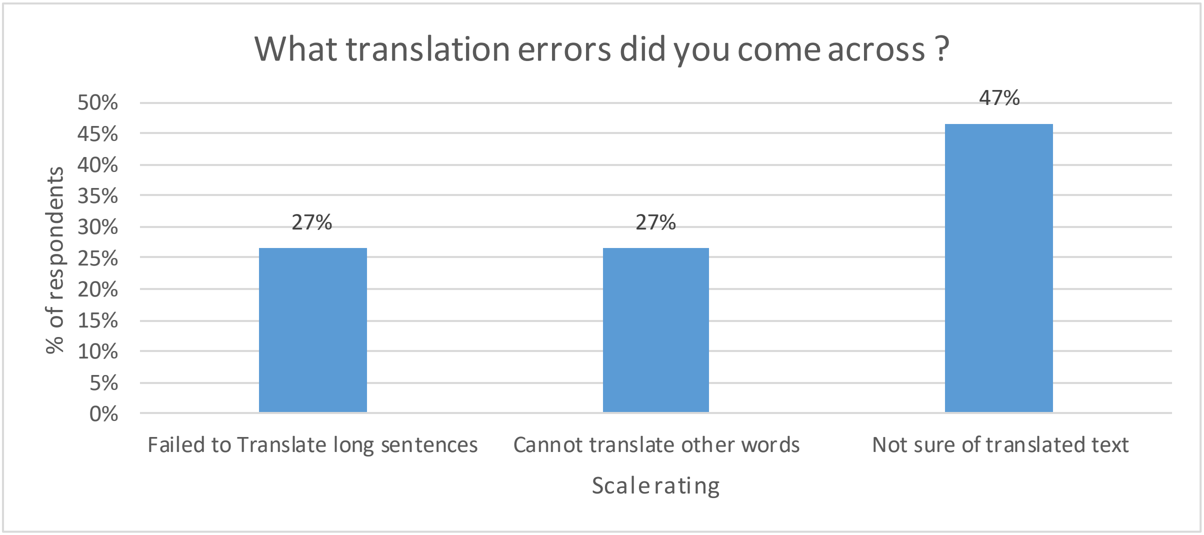


Figure 3.4: Results based on translation errors

Lastly, the respondents were asked using open ended question on what could be added, changed or removed based on system functionality.

According to the figure 3.5 below, 5% of the respondents requested the translation of scientific terms, while 5% required the translation of idiomatic terms, 8% of the respondents needed the system to provide the essay or summary writing and translation, while 5 % requested the system to be capable of translating complex sentences, 5% of the respondents said nothing should be added on a system functionality, while 14% of the respondents suggested that other South African Bantu languages should be included, 9% said the system must be bidirectional by including the Sesotho to English translation to help Sesotho speaking people to learn English language. 3% of the respondents said there must be a notification if the words cannot be translated. Further, 2% said the system should also provide the meanings of the translated words while 3% said the system must also include the examples of the translated words so to have broader understanding of Sesotho language. 8% of the respondents suggested the machine should incorporate the pronunciation practicing technology so they can quickly learn how to effectively pronounce Sesotho words, while 3% suggested the increase of many words. 8% said the application should also be compatible with non-Android mobile phones while 6% said the automatic speech recognition accuracy should be improved. Lastly, 8 % of the respondents said the system should improve audio quality.

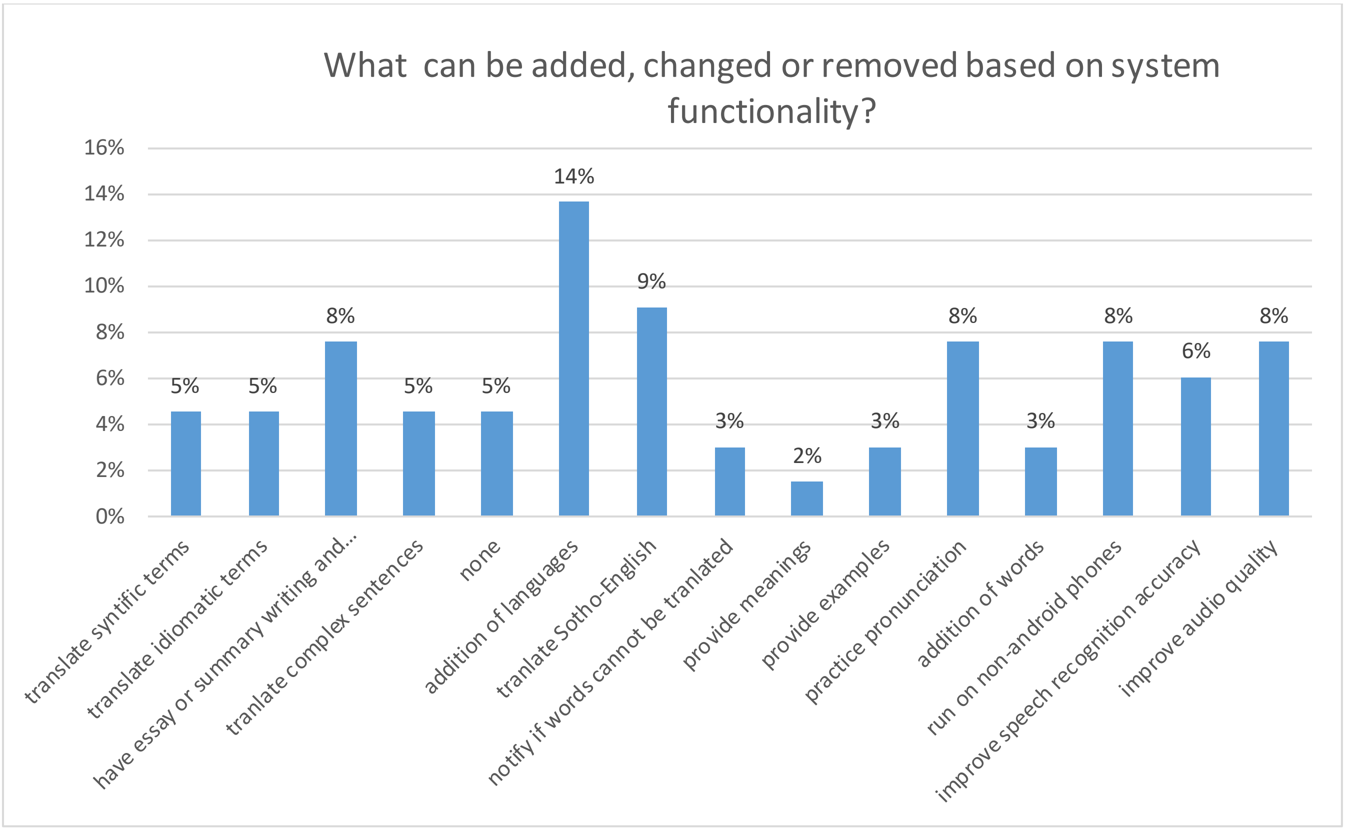


Figure 3.5: Results based on ways to improve the system functionality

3.7.2 Statistics Based on System Design and Usability

Question 1 was based on the system design, how explanatory the application was by reviewing graphical user interface (GUI). The respondents were asked if buttons, text and other controls are clear to understand. According to the chart below, 3% of the respondents strongly disagreed, while 13% disagreed, 17% of the respondents were not sure, while 30% agreed and lastly 37% of the respondents strongly agreed.

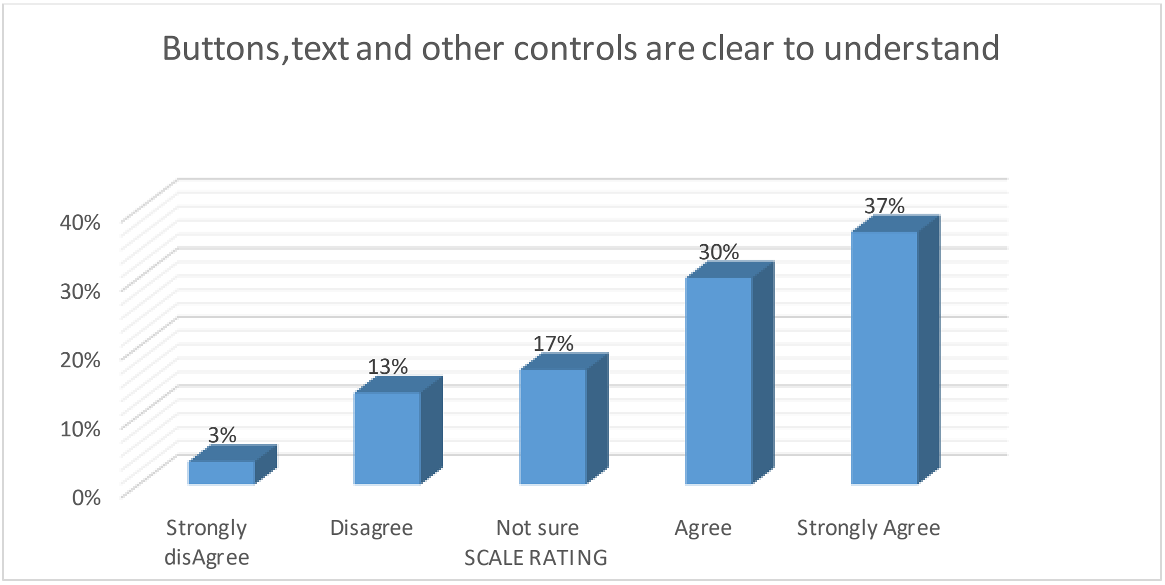


Figure 3.6: Results based on system design

Question 2 was based on how usable the system was in terms of easy enough to understand. The respondents were asked if they can use the application on their own without outside intervention. As depicted in the graph below 3% of respondents strongly disagreed, whereas none of them disagreed, 10% were not sure, 37% agreed while 50% of the respondents strongly agreed.

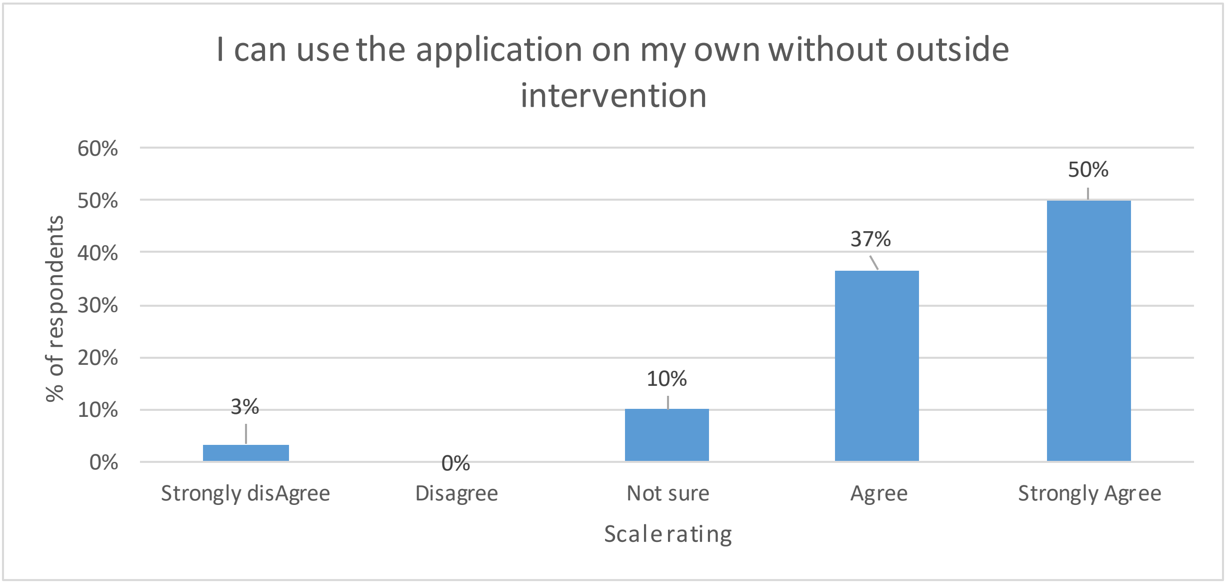


Figure 3.7: Results based on application usability

An open ended question was asked to the respondents to briefly explain what could be changed, removed or added based on the system design and usability. According to the graph below, 23% of the participants said background color must be changed, while 10% said none need to be changed, 10 % of the respondents said descriptive icons or images should be added on a system, while 13 % suggested the increase of font, 17% request the increase of audio sound quality, while 13% of the respondents required more instructions on controls and lastly 13% of the participants requested addition of tutorials that are embedded within the system on how to use the application.

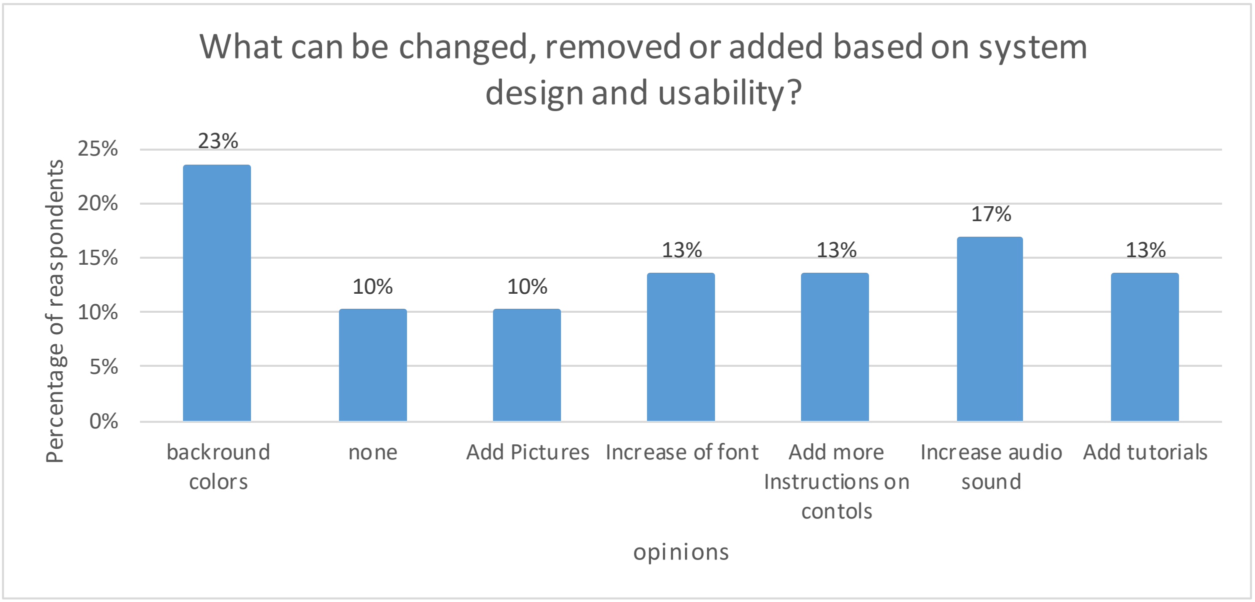


Figure 3.8: Results based on ways to improve system design and usability

3.7.3 Statistics Based on System Performance

Question 1 on the system performance was focusing on speed in terms of system installation and system loading. According to the graph below, 7% of the respondents were strongly dissatisfied, while 23% of the respondents were dissatisfied, 3% was not sure, while 33% were satisfied and lastly, 33% of our respondents were strongly satisfied.

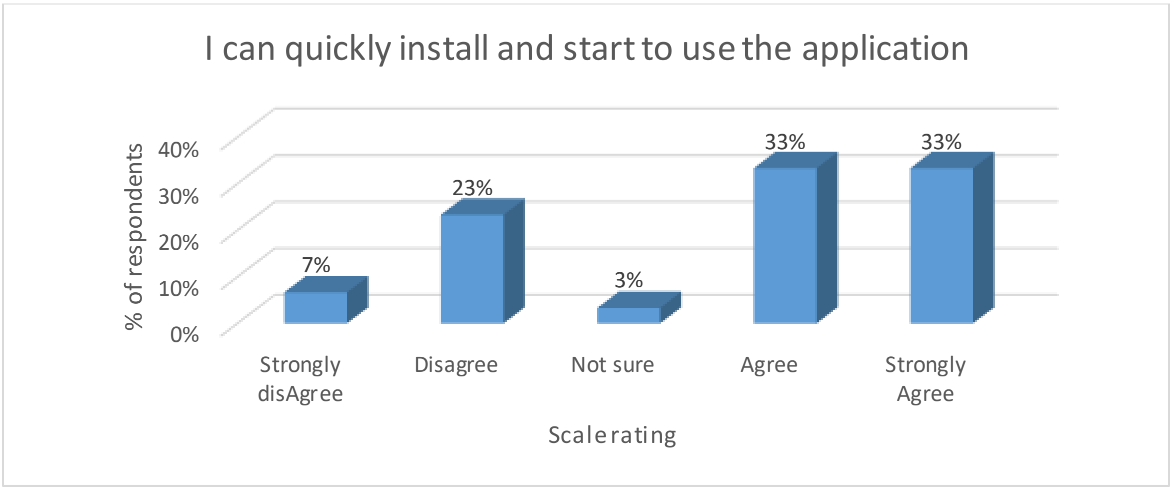


Figure 3.9: Results based on question one of system performance

Question 2 of the system performance, aimed at evaluating the response time from when button 'translate' is clicked to when the output is displayed on a screen. The respondents were asked if they could immediately get the translation feedback. According to the chart below, none of respondents strongly disagreed, while 17% disagreed. 13% were not sure, while 47% agreed and 23% strongly agreed.

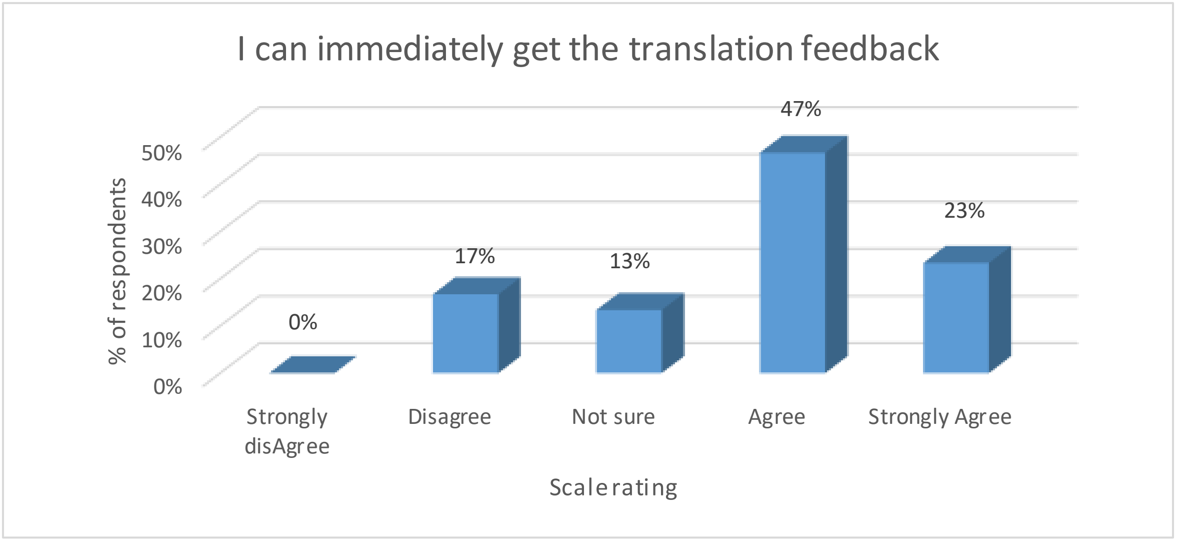


Figure 3.10: Results based on question two of system performance

The third question was evaluating the system performance on low level android phone devices which were provided to the respondents for testing purposes such as Huawei

Y220, Alcatel One Touch Evolve and others. These mobile devices were considered as low specification devices because they are using old version of android operating system (such as v2.3.6) with the phone storage of 40MB, CPU of Dual core 1.0 GHz Cortex-A, and 256 MB RAM.

The respondents were asked if the application can run smoothly on those devices. 10% of the respondents strongly disagreed, while 10% disagreed, 27% of the respondents were not sure, while 30% agreed and lastly 23% strongly agreed.

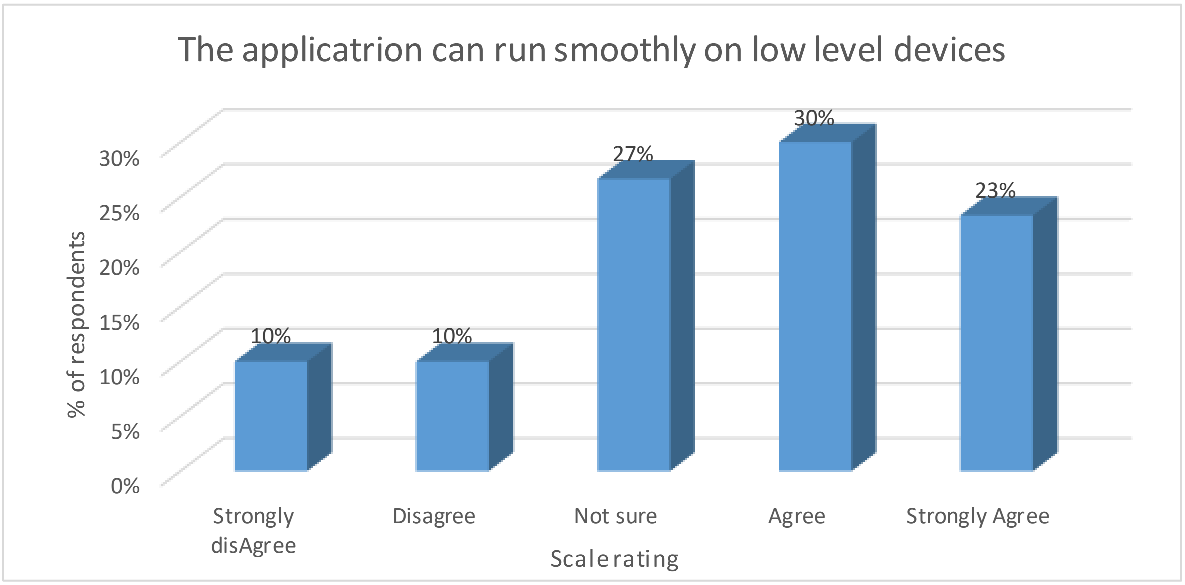


Figure 3.11: Results based on question three of system performance

Lastly, the respondents were asked what could be changed, removed or added based on system performance. According to the pie chart below, 19% said the translation speed is not satisfying and should be increased, while 31% suggested that the response time should be decreased when loading the menus, and 50% of the respondents said nothing should be improved.

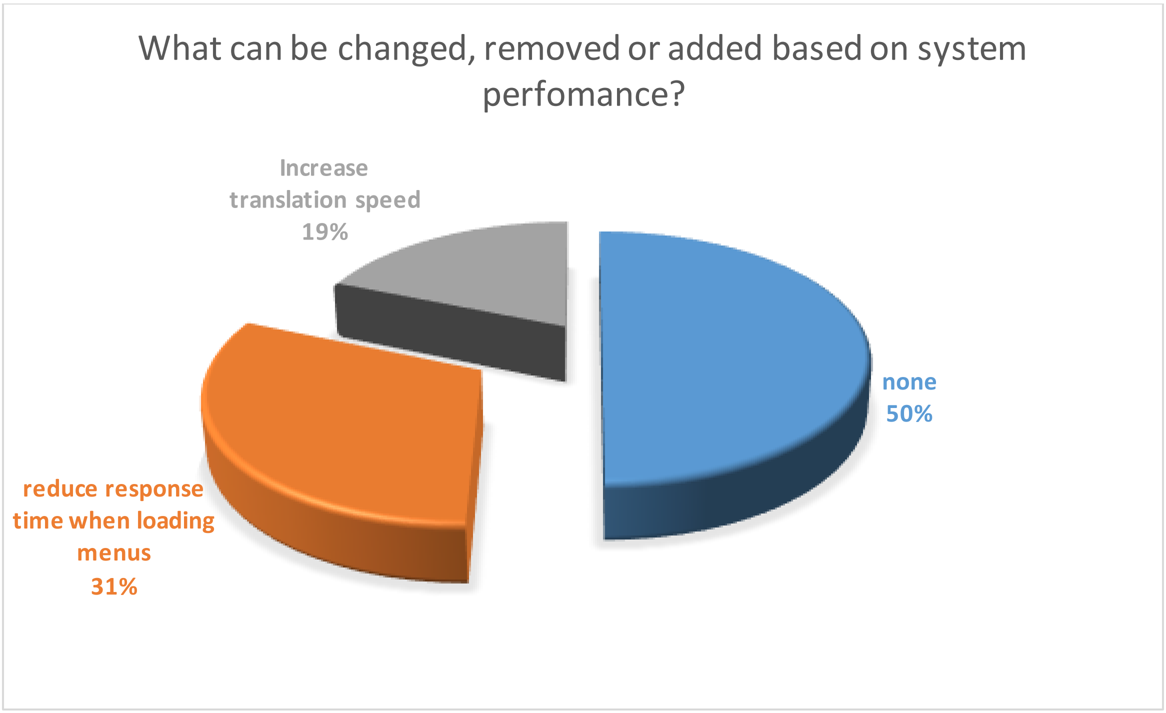


Figure 3.12: Results based on ways to improve system performance

3.8 Conclusion

This chapter was focusing on methodology approach and design, how population and the sample were selected, how data collection instruments were utilized and how data was analyzed in to valuable information. The following chapter focuses on corpus implementation.

Chapter 4: Corpus Implementation

4.1 Introduction

The parallel corpus provided in this research was constructed in a different way from some existing parallel corpora in terms of its flexibility and functionality. Instead of translating the given sentences as they are inside a corpus, it permits the user to use existing words from the machine dictionary to develop and translate their own sentences by following the correct language structure rules. What motivated the development of such a corpus is the limitation of phone storage since the corpus is stored in a phone's memory. The corpus is small but with higher functionality and flexibility. Considering a sentence such as "I eat an apple", it will be inefficient of the machine to iterate through the strings in a corpus to find the exact sentence as it is and translate it; instead the machine breaks and analyzes the sentence in to constituents (grammatically analyzed phrases), match the inputted constituents against set of source constituents in a source corpus; when the match is found then it extracts the Sesotho translation from the target corpus. The end translation result is "nna ja apole", which is close to the correct translation of "nna ke ja apole". From how constituents are developed, they allow modularity and replacement.

Three stages of parallel corpus were implemented, which are corpus collection, corpus cleaning and corpus annotation. The corpus collection stage was proceeded by conducting the quantitative research on migrants and non-Sesotho speakers with the aim to collect basic Sesotho information (words and phrases) that migrants need to know. The corpus cleaning was proceeded on the data by first removing duplicates, articles, and other anomalies. Lastly, corpus annotation was then performed on the data to prepare it for machine processing.

4.2 Corpus Collection

Data that is stored in a corpus for the purpose of NLP was constrained based on the needs of migrants and non-Sesotho speakers. To have accurate, valid and reliable data, the research was conducted to obtain basic Sesotho language that migrants need to know and simultaneously prove the study hypothesis. Hypothesis behind this research was, despite the existing tools exploited to learn Sesotho, most migrants and non-Sesotho speakers still find it difficult to integrate themselves in to Sesotho speaking population due to language barrier.

To evidence the hypothesis, quantitative research method commenced on migrants and non-Sesotho speakers in Welkom and Bloemfontein, Free State, South Africa. Surveys were used to assess subjects' knowledge on basic Sesotho language words and phrases; they were structured based on few language domains such as greetings, small talk, tourism, business words and emergency words. These language domains were decided by the Sesotho language expert. The motive behind the selection of these domains is that they are socially and economically based, and moreover, they lay a firm language foundation that can enable the language learner to easily expand on other aspects of the language. The questionnaires were comprised of only close-ended questions to allow the respondents to quickly and effortlessly answer them.

Different types of population sampling were evaluated, the decision on purposive sampling was reached; we deliberately chose the individuals that would participate based on their location, gender, workstations, their health condition, their nationality, and their education level. In terms of selection based on nationality, some of the participants were identified by the color of their skin such as Indians, Coloureds, Whites and Chinese, while others were recognized by their accents when the researcher was communicating with them using English language. During purposive sampling, the researcher paid no attention to the migrants and non-Sesotho speakers' knowledge of Sesotho, as this will be handled by the questionnaires. The sample size of 100 subjects was reached, with the aim to have accurate and more comprehensive findings. The researcher physically submitted and collected the questionnaires from the respondents. As the respondents fill-in the questionnaires, the researcher was always available to answer any questions unclear to the respondents.

Pilot study was conducted on less than 10 participants to ensure that the administered questionnaires were accurate, comprehensible and straight to point.

4.2.1 Biographical Data Representation

In terms of gender participation, 49% of the participants were males and 51% were females.

Below is the presentation of the gender statistics together with the age group.

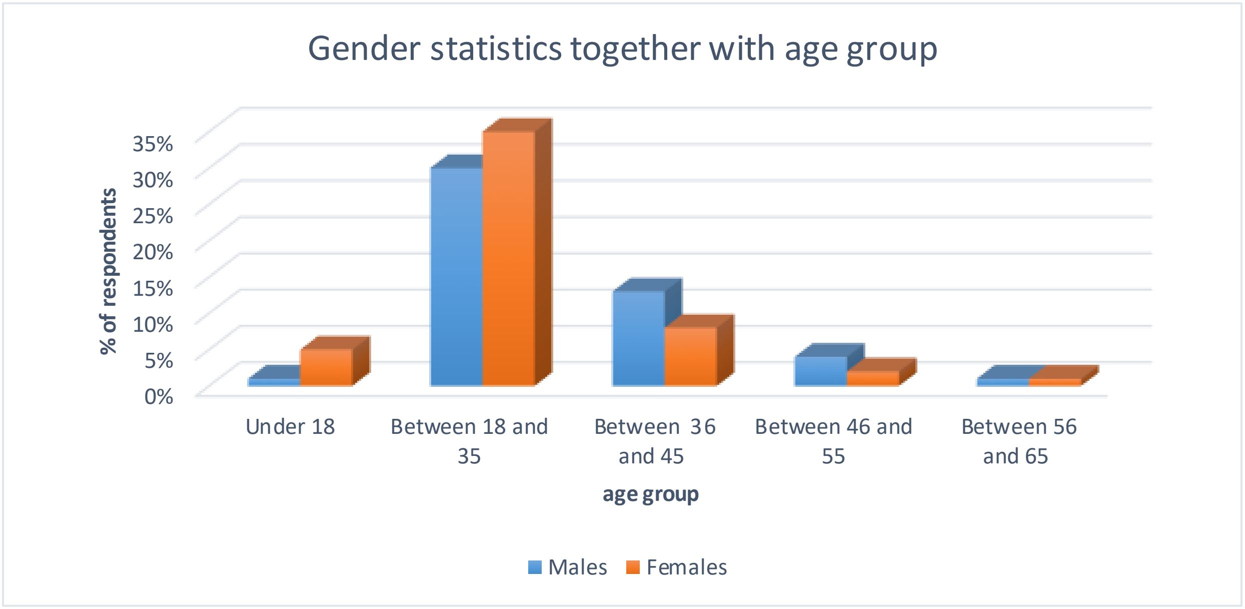


Figure 4.1: Presentation of gender statistics together with the age group

According to the graph above (figure 4.1), majority of participants were youth ranging between 18 and 35, Males: 30% and females 35%. The reason for this was their interest in technology. Additionally, mobile technology has been more widely adopted by youths than otherwise (Tunney et al., 2017).

From the graph below (figure 4.2), the relationship between the gender statistics and education level is presented, the post-secondary participants are dominating, with 28% males and 35% females. Most of the participants are from institutions of high learning such as Central University of Technology (CUT) and neighboring colleges. These institutions have number of migrants and non-Sesotho speakers who are here to pursue their academic dreams.

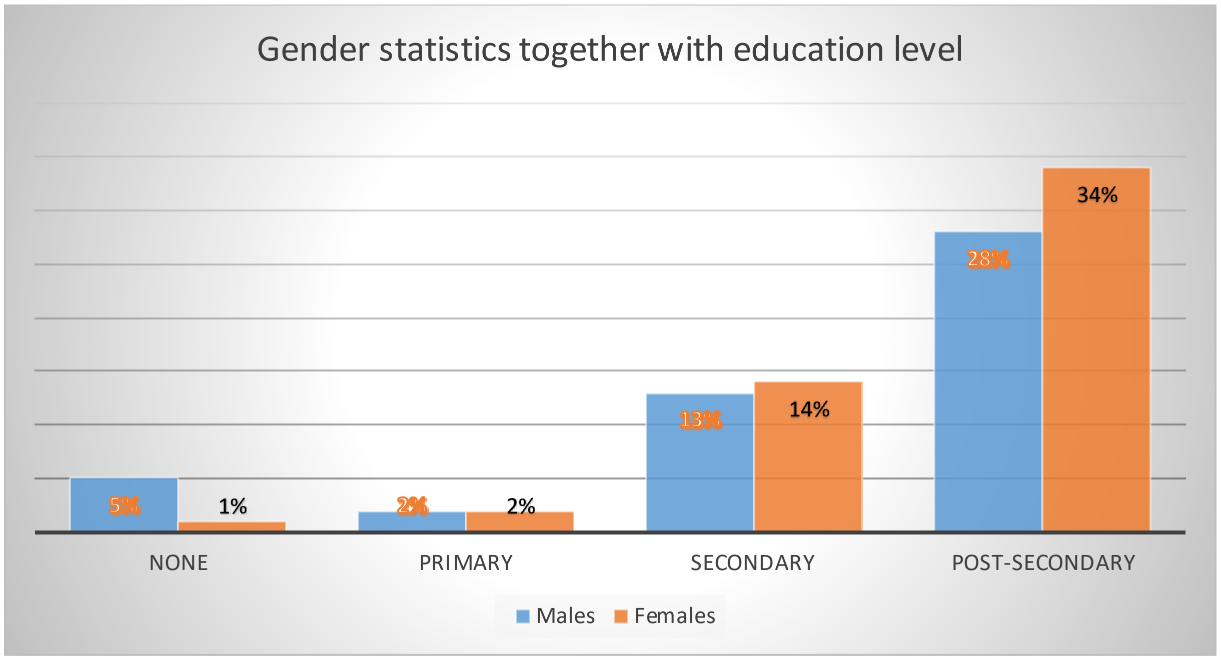


Figure 4.2: Presentation of gender statistics together with the education level

According to the chat below (figure 4.3), gender statistics with nationality is presented, aggregate of 70% of the participants(males and females) were black; the major reason is that most of non-Sesotho speaking groups residing in Free State are mostly Zulus, Xhoza and migrants from other neighbouring countries such as Botswana, Zimbabwe, Nigeria among others.

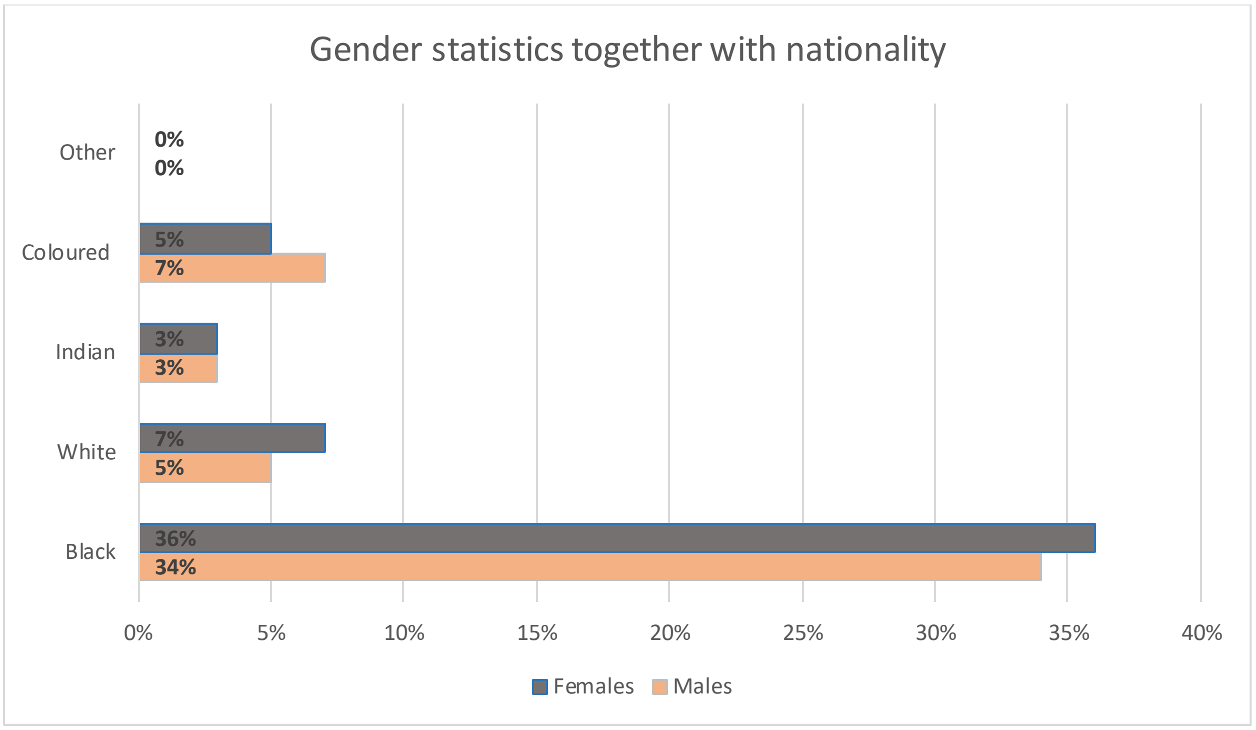


Figure 4.3: Presentation of gender statistics together with nationality

4.2.2 Core-Data Analysis

As it was explained earlier that the questionnaires were developed to cover five main domains of basic language learning, these domains were greetings, small talk, tourism (directions, culture, beliefs, ecosystem, food etc.), emergency and business language. The table below shows the average results obtained from the answers provided by the respondents.

Table 4.1: Average results of respondents who need to learn Sesotho based on different domains

Average results of respondents who need to learn :					
	Strongly Agree	**Agree**	**Not Sure**	**Disagree**	**Strongly disagree**
Sesotho greetings	29%	35%	11%	21%	3%
Sesotho small talk	33%	40%	10%	14%	4%
Sesotho tourism language	33%	34%	12%	16%	4%
Sesotho emergency language	31%	34%	11%	20%	4%
Sesotho business language	30%	34%	12%	19%	5%
Average Results	**31%**	**35 %**	**11 %**	**18%**	**4%**

According to the average statistics mentioned above, a need to learn the Sesotho language is substantial. This is identified by the higher average results of 31% of respondents who strongly agreed and 35 % of respondents who agreed that they need to learn Sesotho language domains. Therefore, these results satisfy the hypothesis previously stated in the chapter and yet would help to guide the development of the corpus.

For each domain, relevant data was collected and stored in a source corpus, and its Sesotho translation was stored in a target file. Words and sentences from both files were parallel aligned based on their translations.

4.3 Corpus Cleaning and Recording

To ensure consistency, the character case of the data in parallel corpus was changed to lowercase; an application that can remove duplicates and some delimiters such as full stops, commas and question marks from both files was developed. The application was helpful; nevertheless, post processing by language expert was made on the data. Articles known as determiners were removed from the source language because of their minimal meaning and moreover, they don't exist in Sesotho language. These articles includes words such as "the," "a", and "an". The contracted words such as "couldn't, didn't, can't," were changed to the standard words. The usage of technical words were avoided as possible. We ensured that all the words are correctly spelled, and correctly formatted. The end results on data was 200 sentences with 400 words for each file. However, some of the words were repeatedly used in different contexts.

Sesotho words were recorded for the purpose of Sesotho speech synthesizer. The office was used on Sundays to record the words with the intention to avoid any obstruction including the background noise. Four participants who were native Sesotho speakers were used to perform speech, among the participants, there were two males and two females. An hour break was applied during the recording sessions, lunch was catered, and participants were compensated. The Sesotho audio files of about 300 were transcribed and stored in a file called "raw" inside the application folder.

4.4 Corpus Annotation

Below is the presentation of POS tags together with phrase tags (PT) that were used during corpus annotation.

Table 4.2: Presentation of POS and PT tags together with their explanation

Tag	Description
POS tags	
NN	Noun
ADJ	Adjective
VB	Verb
P	Preposition
POSS	Possessive pronoun
Wh (questioning words)	Where, who, when, why, which, what
Hw (questioning words)	How
AND	And
OR	Or
Phrase Tags	
NP	Noun phrase
VP	Verb Phrase
PP	Preposition phrase
QP	Questioning phrase

Corpus annotation was considered as the complex and crucial stage; on a base that both language structures had to be further studied; moreover, researcher must be able to spot the language structural differences, and find a way to bring these languages together, to increase translation accuracy. Part of speech tagging was first implemented.

4.4.1 Part of Speech Tagging

Source language words were assigned POS tags, based on their grammatical features. This was performed by English language expert. At this stage the focus was on individual words; an underscore ("_") character was used to separate the word from its part of speech tag. The reason part of speech tagging was performed is because the machine can easily understand the word through its grammatical property. Example on how words were first tagged based on their domains and sub domains before phrase annotated.

Greetings

```
good_adj morning_nn
how_hw are_vb you_nn
I_nn am_vb fine_nn thank_vb you_nn
lovely_adj day_nn
```

Small Talk

```
Where_wh are_vb you_nn
I_nn am_vb Indian_nn
We_nn from_p China_nn
```

Work

```
I_nn use_vb computer_nn
Where_wh is_vb my_pos colleague_nn
I_nn teach_vb learners_nn
```

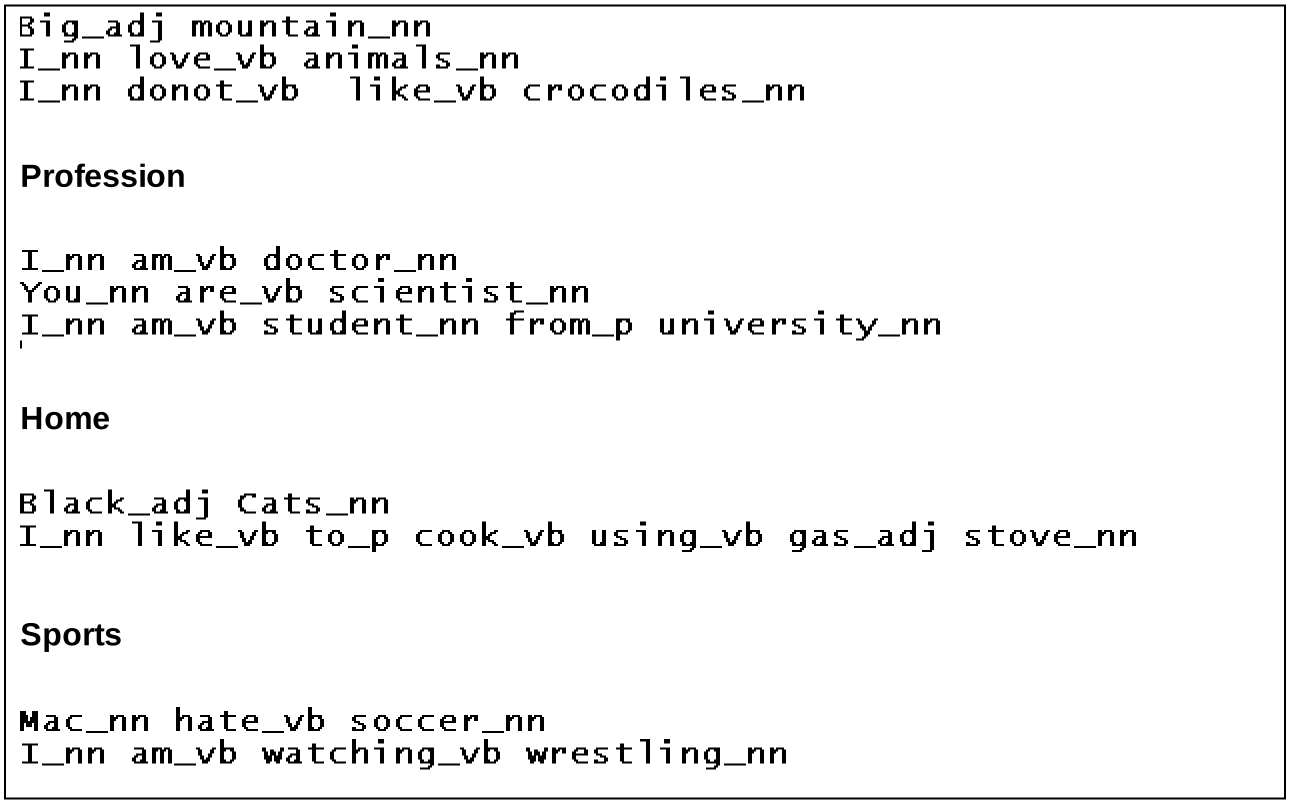
Emergency

```
Joseph_nn call_vb police_nn
Can_vb I_nn borrow_vb your_pos phone_nn
Stop_vb thief_nn
```

Shopping

```
How_h much_vb is_vb sugar_nn
Give_vb me_nn change_nn
```

Tourism



```
Big_adj mountain_nn
I_nn love_vb animals_nn
I_nn donot_vb  like_vb crocodiles_nn
```

Profession

```
I_nn am_vb doctor_nn
You_nn are_vb scientist_nn
I_nn am_vb student_nn from_p university_nn
```

Home

```
Black_adj Cats_nn
I_nn like_vb to_p cook_vb using_vb gas_adj stove_nn
```

Sports

```
Mac_nn hate_vb soccer_nn
I_nn am_vb watching_vb wrestling_nn
```

Figure 4.4: Example of how part of speech tagging layer is applied.

Sesotho words were difficult to be assigned part of speech tags in a similar way as English because of language differences. The part of speech tags applied in English language couldn't be applied and match the ones used in Sesotho language; for example, the phrase “good morning” is represented by a single word “mmorong”. Another example, a sentence such as “have lovely day” is translated to “eba le letsatsi le monate” ending up with five words, making it difficult to classify their part of speech tags. Hence, the solution was first to perform phrase annotation on a source language, then link the Sesotho phrases to their source phrases.

4.4.2 Phrase Annotation

Phrase annotation helps particularly in language to language translation, especially when two or more words in one language are confined or represented by a single word in another language.

At this stage we grouped source language words based on their word dependencies called phrases. Phrase tagging was performed to group closely grammatical related words in to constituents, whereby machine treated them as single items that can easily be translated. For example a partially analyzed phrase “good_adj night_nn” was tagged using a noun phrase tag embedded in parentheses in this form “NP (good_adj night_nn)” resulting in to what is called constituent (group of one or more grammatical analyzed words treated as a single word).

The use of constituents on each line in a source language was successful, however, they had to be linked or referenced to their relevant Sesotho translations from the target corpus. As displayed in figure 4.5 below.

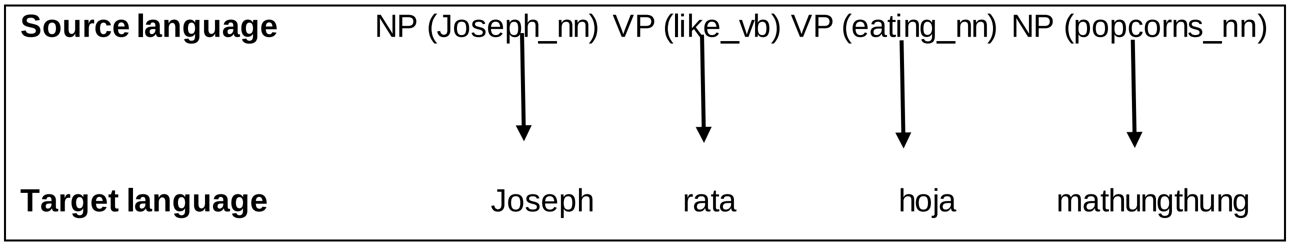


Figure 4.5: Link of constituents from source to target file using pointers

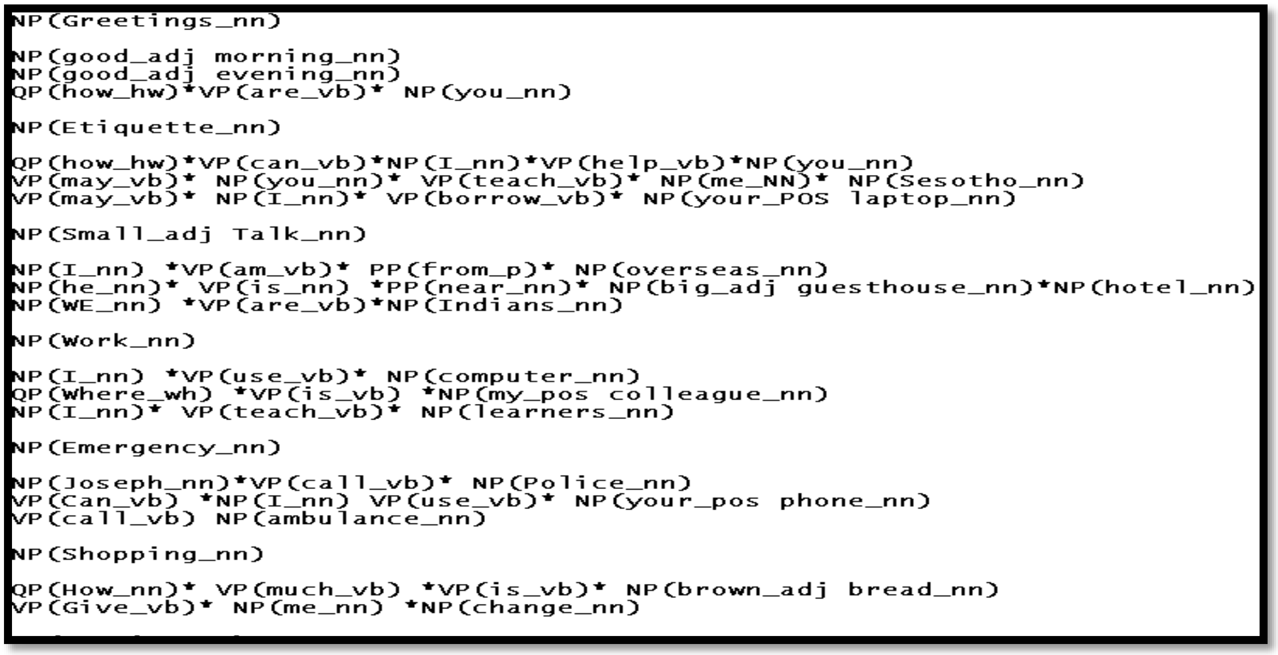
The approach that was first considered, was the use of pointers, in which a constituent points or reference to its relevant translation from two parallel sentences. A problem that deteriorated the implementation of pointers was the use of low level text files that could not support pointers. Corpus indexing was considered, whereby the constituents from each two parallel lines (source and target) are tokenized in to elements, and each element is indexed. Corpus indexing makes it easier to retrieve elements based on their locations. However, this approach also failed because plain text files cannot support indexing.

The successful solution was the use of numerical values to link the source constituents with target translations that are parallel aligned. Since when each character whether a letter or special character is represented by the American Standard Code for Information Interchange (ASCII) character code, a small program that can uniquely calculate and sum the character codes for each source constituent was developed. In details, the program has an operation that receives each source language constituent, it loops through the characters of the constituent and sum the character codes as it iterates. The final value was manually assigned to the relevant Sesotho translation. During the machine translation, the tagged value will be used to retrieve the requested Sesotho translation.

The proposed solution was also problematic because noun constituents such as NP (bread_nn) and NP (beard_nn) are resulting into the same ASCII value. Human post processing was done by either adding or subtracting on duplicates values to make them unique, and also at the same time ensuring that the changed values do not hamper with other calculated values.

We had to ensure that for each parallel sentences, the number of constituents from the source language are equals to the number of Sesotho translations. The delimiter ("*") was used to separate the constituents in a source corpus and Sesotho translations in a target corpus. The constituents with inflection of verbs such as eat, eats, eating were linked to a single Sesotho translation which is "ja". Example on how phrases for both source and target files are tagged.

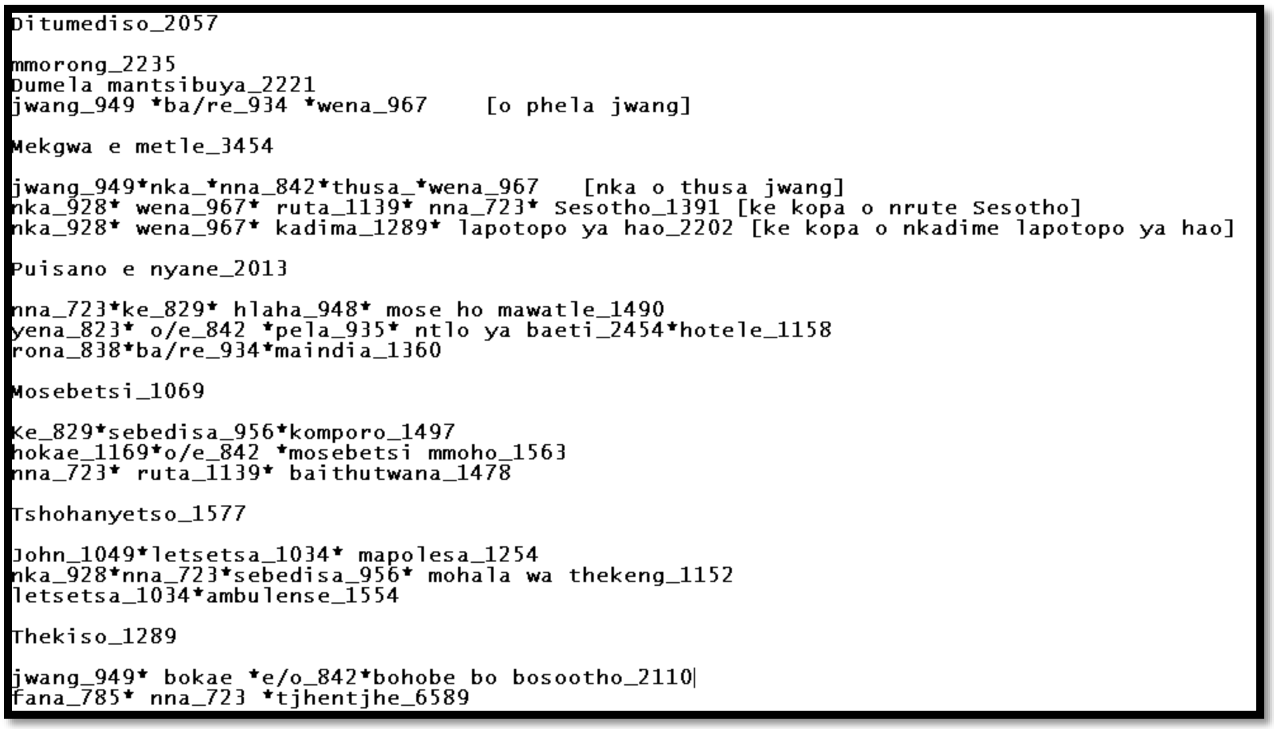
4.4.2.1 Source Language Annotation



```
NP(Greetings_nn)

NP(good_adj morning_nn)
NP(good_adj evening_nn)
QP(how_hw)*VP(are_vb)* NP(you_nn)

NP(Etiquette_nn)

QP(how_hw)*VP(can_vb)*NP(I_nn)*VP(help_vb)*NP(you_nn)
VP(may_vb)* NP(you_nn)* VP(teach_vb)* NP(me_NN)* NP(Sesotho_nn)
VP(may_vb)* NP(I_nn)* VP(borrow_vb)* NP(your_POS laptop_nn)

NP(Small_adj Talk_nn)

NP(I_nn) *VP(am_vb)* PP(from_p)* NP(overseas_nn)
NP(he_nn)* VP(is_nn) *PP(near_nn)* NP(big_adj guesthouse_nn)*NP(hotel_nn)
NP(WE_nn) *VP(are_vb)*NP(Indians_nn)

NP(Work_nn)

NP(I_nn) *VP(use_vb)* NP(computer_nn)
QP(Where_wh) *VP(is_vb) *NP(my_pos colleague_nn)
NP(I_nn)* VP(teach_vb)* NP(learners_nn)

NP(Emergency_nn)

NP(Joseph_nn)*VP(call_vb)* NP(Police_nn)
VP(Can_vb) *NP(I_nn) VP(use_vb)* NP(your_pos phone_nn)
VP(call_vb) NP(ambulance_nn)

NP(Shopping_nn)

QP(How_nn)* VP(much_vb) *VP(is_vb)* NP(brown_adj bread_nn)
VP(Give_vb)* NP(me_nn) *NP(change_nn)
```

Figure 4.6: Source corpus, how phrases are tagged

4.4.2.2 Target Language Annotation



```
Ditumediso_2057

mmorong_2235
Dumela mantsibuya_2221
jwang_949 *ba/re_934 *wena_967    [o phela jwang]

Mekgwa e metle_3454

jwang_949*nka_*nna_842*thusa_*wena_967   [nka o thusa jwang]
nka_928* wena_967* ruta_1139* nna_723* Sesotho_1391 [ke kopa o nrute Sesotho]
nka_928* wena_967* kadima_1289* lapotopo ya hao_2202 [ke kopa o nkadime lapotopo ya hao]

Puisano e nyane_2013

nna_723*ke_829* hlaha_948* mose ho mawatle_1490
yena_823* o/e_842 *pela_935* ntlo ya baeti_2454*hotele_1158
rona_838*ba/re_934*maindia_1360

Mosebetsi_1069

Ke_829*sebedisa_956*komporo_1497
hokae_1169*o/e_842 *mosebetsi mmoho_1563
nna_723* ruta_1139* baithutwana_1478

Tshohanyetso_1577

John_1049*letsetsa_1034* mapolesa_1254
nka_928*nna_723*sebedisa_956* mohala wa thekeng_1152
letsetsa_1034*ambulense_1554

Thekiso_1289

jwang_949* bokae *e/o_842*bohobe bo bosootho_2110
fana_785* nna_723 *tjhentjhe_6589
```

Figure 4.7: Target corpus, how phrases are tagged using numerical values

4.5 Conclusion

Despite combining two languages that are not even slightly related, a small corpus with high flexibility was developed. The use of constituents is the secret behind our corpus success. They allow modularity, in a sense that, a single constituent can be used in more than one instances, or in different inputted sentences. Therefore, there is no need of having duplicates of constituents. The constituents belonging to the same phrase group can be interchangeably used. For example, a sentence "**NP (I_nn)** live in Joburg" can be replaced with "**NP (Tall_adj man_nn)** live in Joburg". However, the constituents' flexibility can result in syntactically correct but semantically incorrect sentences.

This parallel corpus is generic enough to allow its extension to other Bantu languages. It has flaws due to language differences, time and resources. The study of these two languages can be further accentuated as there are wide range of stimulating differences and similarities when comparing these languages. Next chapter focuses on system design and development, the end product would be used to assess the translation accuracy, speed, power and flexibility of the corpus.

Chapter 5: System Design and implementation

5.1. Introduction

In this chapter the discussion of the system design together with implementation is presented. Unified modeling language (UML) diagrams were used to describe system analysis and design. Android programming language was used to implement the system. The analysis and implementation of the ASR, data cleaning stage, natural language understanding (NLU), machine translation and Sesotho speech synthesis are discussed. In summary, the spoken English utterance is recognized and converted in to text by the machine; the text is sent to the data cleaning stage to remove unneeded data such as delimiters, articles and contracted words are changed to standard words; the text is then sent through NLU which performs morphological, syntactical and semantic analysis on a text; at this stage the text is now in a form of constituent, this constituent is sent to the machine translator which compares it against the set of source constituents in a source corpus to find the best match, when the match is found, the corresponding Sesotho translation is extracted from the target corpus; the Sesotho speech synthesizer is used to read the translated text.

5.2. System Design

UmobiTalk similar to any speech based translator, is developed on three main processes that were discussed in chapter two (section 2.5). These processes (ASR, MT and TTS) are depicted below together with their sub processes. Each process is elaborated through UML diagrams.

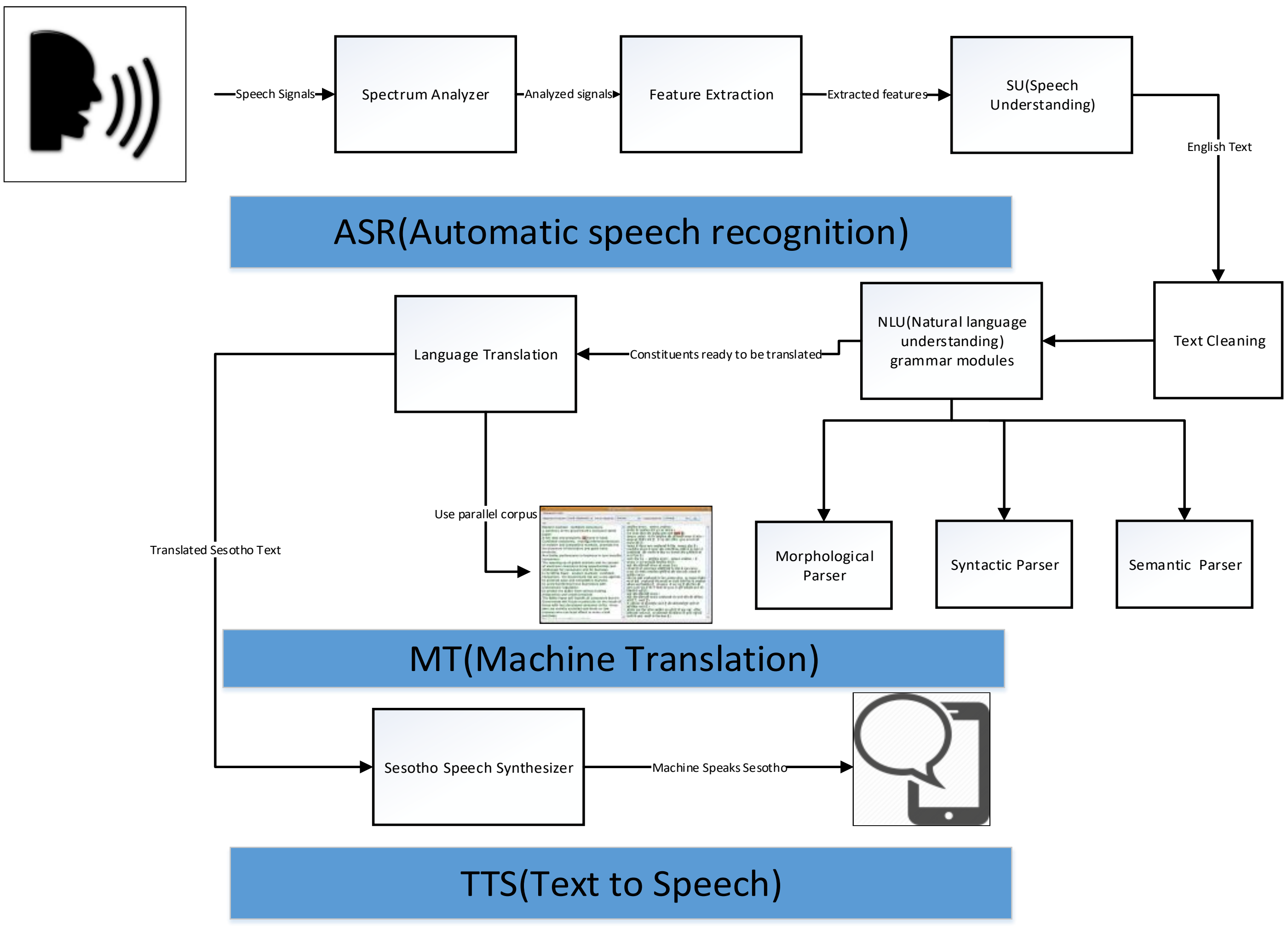


Figure 5.1: Umobitalk's framework on three main speech based processes

5.2.1 ASR Design

According to the figure 5.2 below, the first process that was analyzed is ASR. At this stage the machine receives the speech uttered by the user, recognize it and coverts it into English word(s); the converted text is displayed in a text box, to allow edits.

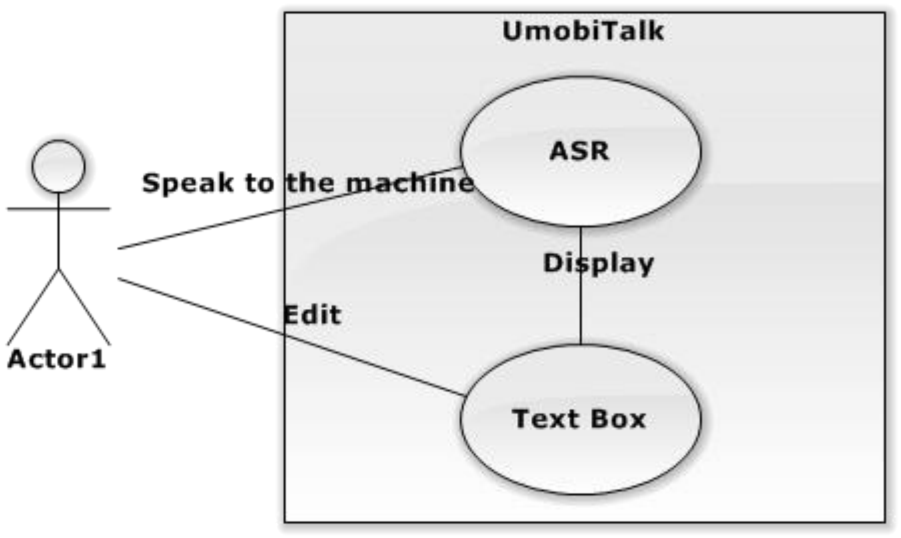


Figure 5.2: Use case of the user speaking to machine and edits the output

How the system work was elaborated using sequence diagram. The importance of sequence diagram is that it provides the brief overview, and it shows data flow from one object to the other during machine execution.

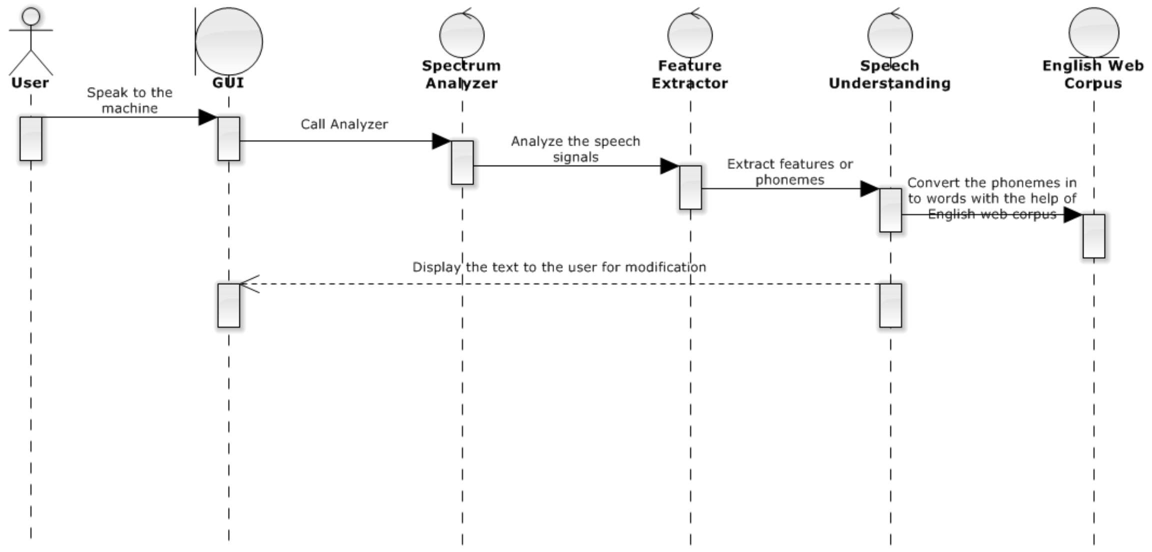

User
GUI
Spectrum Analyzer
Feature Extractor
Speech Understanding
English Web Corpus
Speak to the machine
Call Analyzer
Analyze the speech signals
Extract features or phonemes
Convert the phonemes in to words with the help of English web corpus
Display the text to the user for modification

Figure 5.3: Sequence diagram of the user speaking to the machine

From the above sequence diagram (figure 5.3), the user speaks to the machine by interacting with application's GUI, it calls the spectrum analyzer which acts like an ear of the machine, to analyze the speech signals. Feature extractor extracts the features also known as phonemes from the analyzed signals; speech understanding process converts the phonemes in to English text using English web corpus; finally, the recognized text is displayed on a screen, with the aim to allow the user to view and review the recognized words should the level of satisfaction not met by the ASR. This is because ASR cannot be fully reliable in terms of recognition.

5.2.2 Data Cleaning Design

Immediately the user presses the button ‘translate to Sesotho’, the data cleaning operation is called. For high translation accuracy, text must first be cleaned by checking and removing any attributes that are not needed by the system; see figure below (figure 5.4), these attributes are delimiters, articles and contracted words.

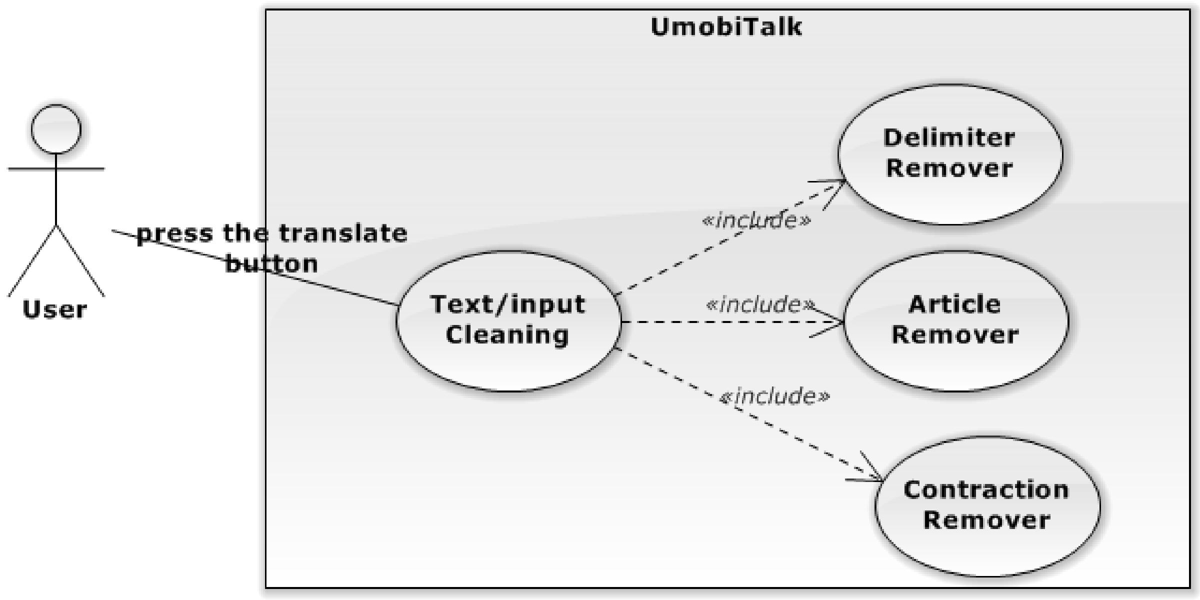


Figure 5.4: Use case diagram of input cleaning stage.

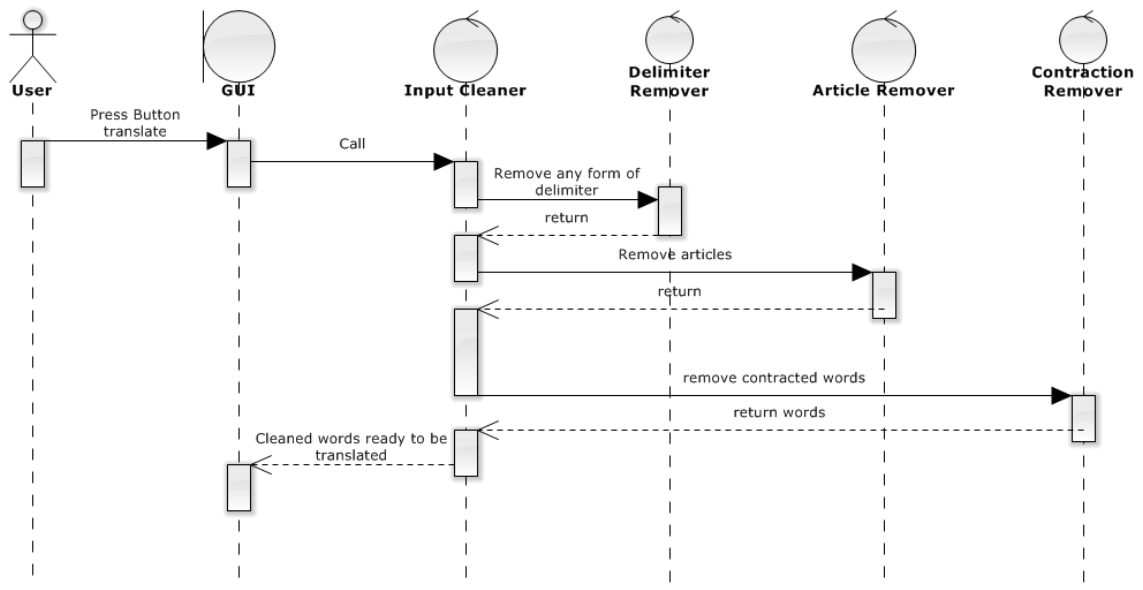


Figure 5.5. Sequence diagram of input cleaning stage.

As shown in figure 5.5 above, after button ‘translate to Sesotho’ is clicked, GUI calls the Input cleaner operation, that calls delimiter remover to remove the unnecessary delimiters, then it calls the article remover to remove the articles and lastly the contraction remover is called to change the contracted words to standard words.

The cleaned text is sent to NLU for further processing

5.2.3 NLU Design

Example below shows how phrase “good morning” is analyzed through NLU processes and translated by the machine translator.

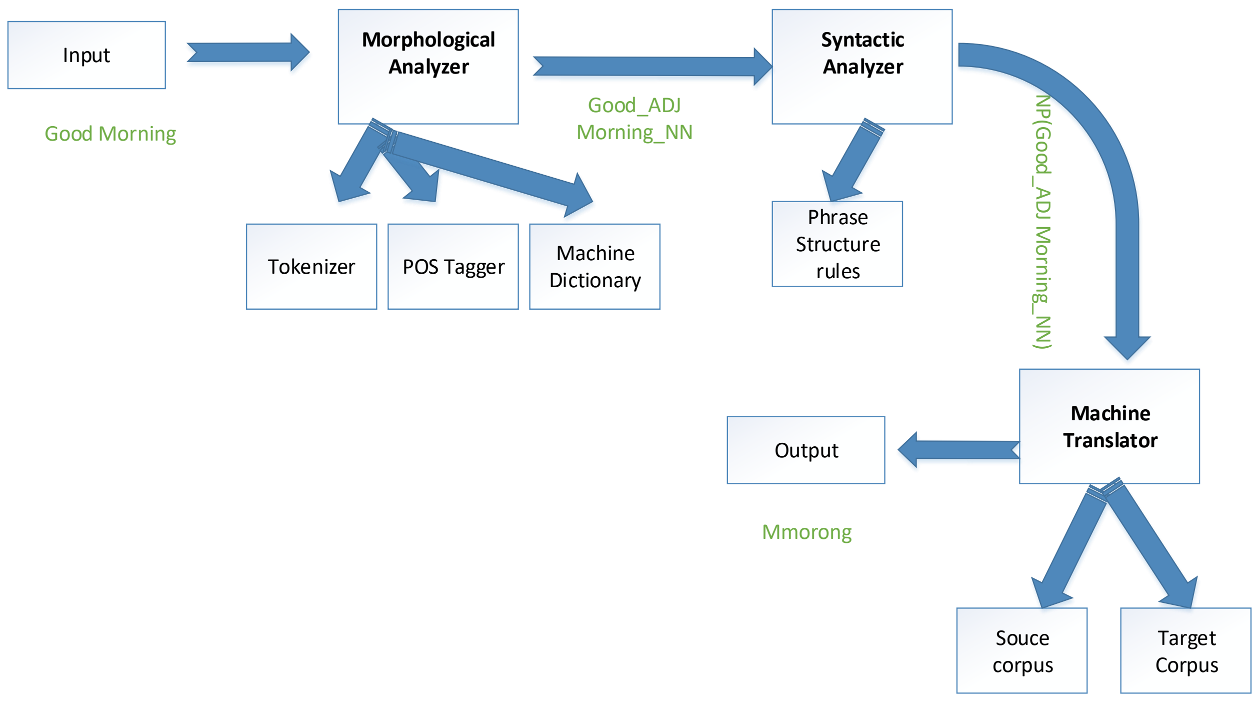


Figure 5.6: Data flow through NLU process

NLU is a part of the machine translator that is first executed before the language translation takes place, it first analyzes the grammatical features of the given input. It consists of three main operations, morphological, syntactical and semantic operations. Input must first go through NLU operations before it is declared as a legal input and ready to be translated. A legal inputted sentence is the sentence in which all of its words are in a machine dictionary database, and the structural pattern of those words forming a sentence are conforming to the grammatical rules of the language.

5.2.3.1 Morphological Analyzer Design

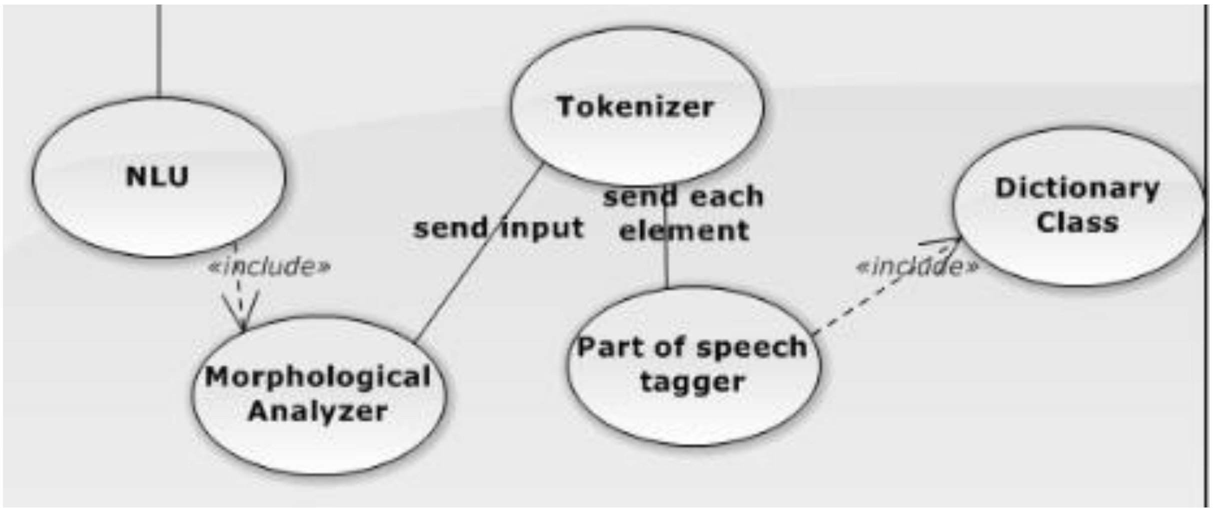


Figure 5.7: Use case diagram showing the morphological analysis of the NLU

According to figure 5.7 above, the morphological analysis stage can become more complicated, however, in this case, the focus was only on two operations which are tokenization and part of speech tagging.

When the morphological analyzer is called, it passes the inputted text to the tokenizer to split the sentence in to words known as tokens or elements using white space as a delimiter. Each word is sent to the part of speech tagger that uses dictionary database to assign a relevant part of speech tag(s) next to the word. The reason for using white space as a delimiter or divider is because both languages' orthography is disjunctive whereby words are separated by white spaces.

Machine dictionary database which is comprised of the table storing source language lexical items (nouns, verbs, adjectives, prepositions etc.) and their part of speech tags was developed. These tags are acronyms for POS words. Each word is parallel aligned with its part of speech tag. Some words with more than one grammatical features are aligned with more than one part of speech tag separated by a forward slash. For consistency, we ensured that all the words in a source language corpus are included in a machine dictionary database.

During execution, part of speech tagger loops inside the machine dictionary database to compare if each word forming part of an inputted sentence, is a legal word (exist in a database) and assigns it its relevant POS tag separated by an underscore for example “eat_vb”. Inputted words not found in a database are assumed as illegal, they cannot be translated. By default, they are declared as proper nouns, and they are assigned a noun tag.

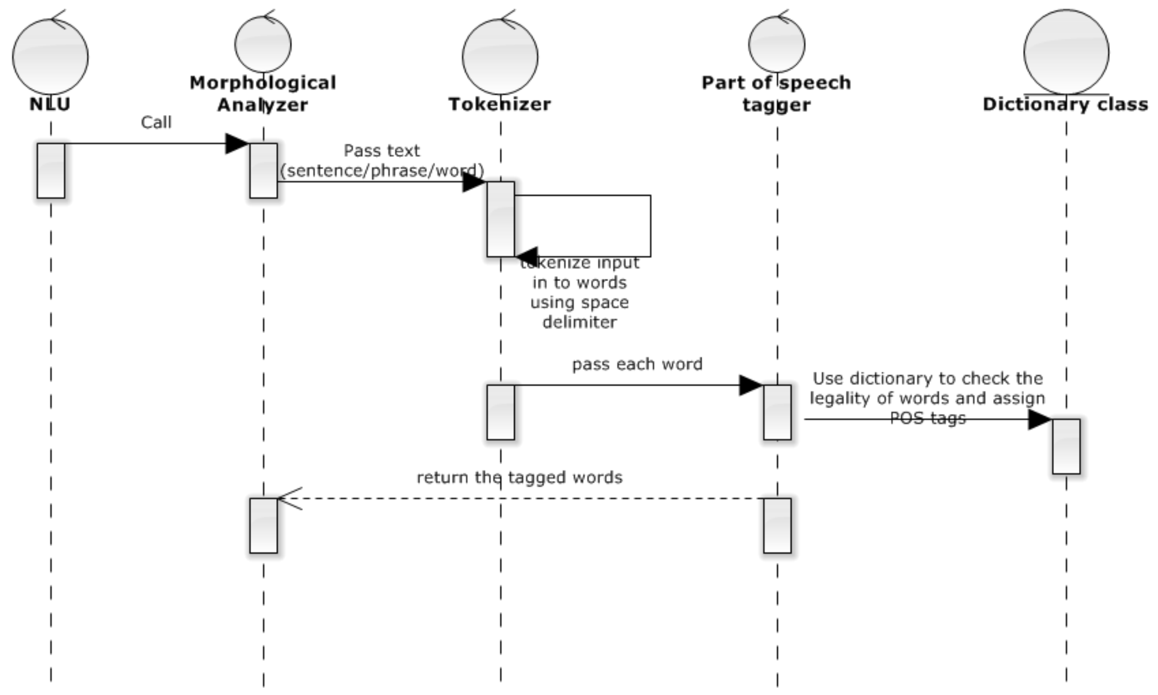

NLU
Morphological Analyzer
Tokenizer
Part of speech tagger
Dictionary class
Call
Pass text (sentence/phrase/word)
tokenize input in to words using space delimiter
pass each word
Use dictionary to check the legality of words and assign POS tags
return the tagged words

Figure 5.8: Sequence diagram showing the operations of morphological analyzer

The sequence diagram above (figure 5.8) shows the step by step processes of morphological analyzer that were previously discussed. After each word has been tagged, they are grouped back together forming a partially analyzed sentence. This sentence is then sent to the syntactic analyzer for further analysis.

5.2.3.2 Syntactic Analyzer Design

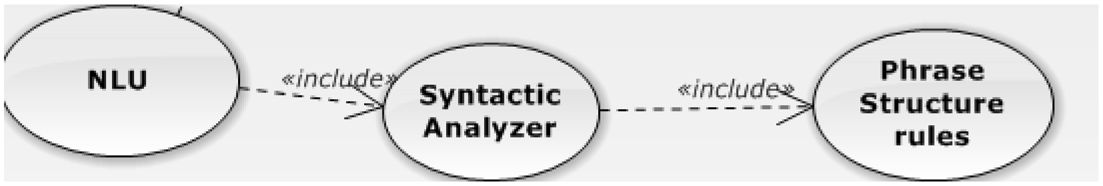


Figure 5.9: Use case of the syntactic analyzer, second part of the NLU

From the figure above (figure 5.9), the syntactic analyzer uses the language rules known as phrase structure rules or CFG rules to analyze and understand the grammatical structural pattern of the inputted sentence. Syntactic analyzer groups words based on their grammatical dependency, forming constituents or phrase chunks that can be easily translated by the MT. The formation of constituents, is based on the following phrase structure rules as shown below:

Noun, verb and prepositional phrase structure rules. Each phrase structure rule has a specific criteria to follow when determining a legal word dependency structure. The legal word dependency structure rules for noun phrase (NP) are modeled below using a parse tree like model.

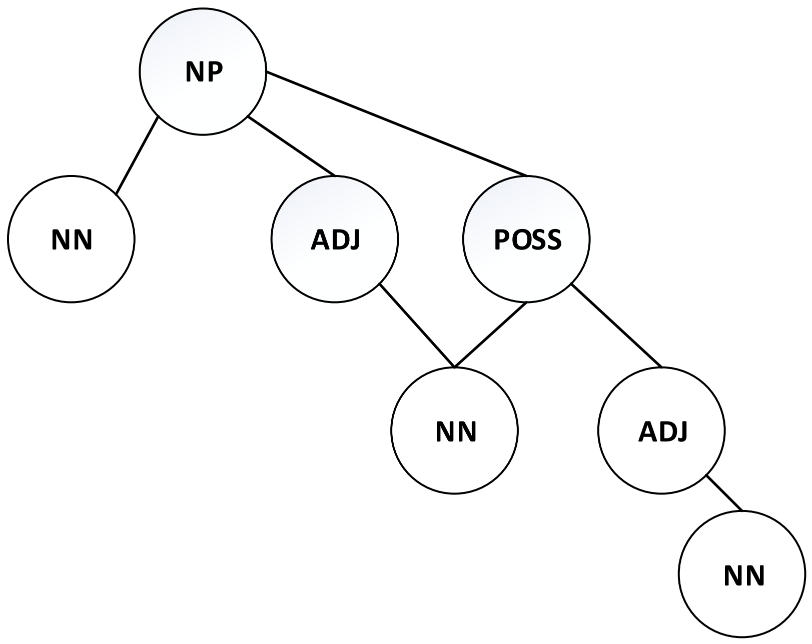


Figure 5.10: Presentation of NP structure rules using tree structure

From figure 5.10 above, a noun phrase constituent can be built using the following part of speech; a noun only, or adjective and noun, or possessive and a noun, or possessive, adjective and a noun. From figure 5.10 above, the following noun phrases can be modeled

1.NN example (Bread)
2.ADJ and NN example (Brown Bread)
3.POSS and NN example (My Bread)
4.POSS, ADJ and NN example (My Brown Bread)

Legal word dependency structure rules for verb phrase (VP) are presented below using the parse tree like model.

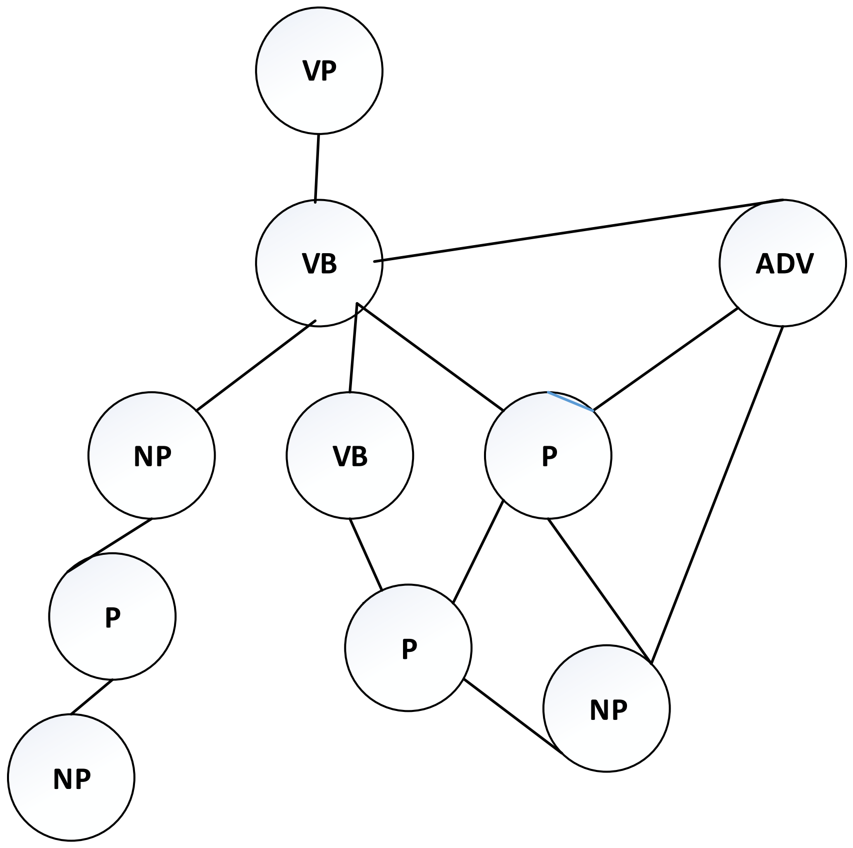


Figure 5.11. Presentation of VP rules using the tree structure

From figure 5.11 above, the following verb phrases can be formed:

1.VB and NP example (eat bread)

2. VB, NP, P and NP example (ate bread during lunch)

3. VB, P, P and NP example (is close to the market)

4.VB and ADV example (drinks quickly)

5.And many more.

The legal word dependency structure rules for preposition phrase (PP) are

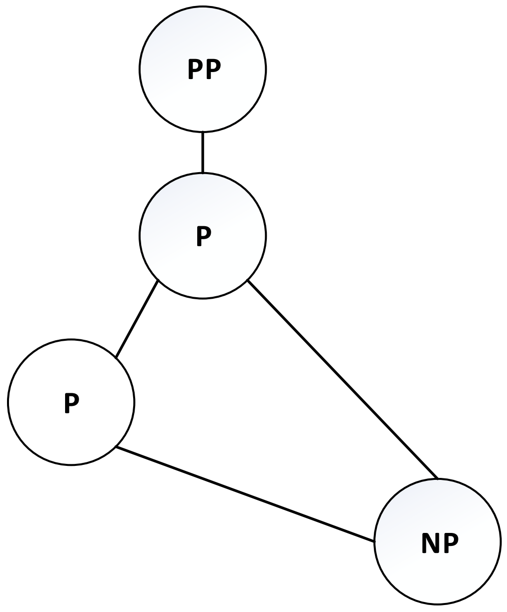


Figure 5.12: Presentation of PP rules using the tree

From the figure 5.12 above, the following prepositions phrases can be formed:

1. P and NP example (near the blue car)
2. P, P and NP example (far from my town/ close to the market)

The questioning phrase (QP) structure rule was also determined; the rule provided below can accommodate all sorts of questions such as where, who, what, when, why, which followed by the VP and NP.

QP VP NP example (QP (where_wh) VP (is_vb) NP (Thabo_nn))

The grammatical rules modeled above made it easy to organize, analyze and easily translate many inputted sentences. According to the machine's syntactic analyzer as described previously, the relationship or dependency of words is not determined by the actual words but by their grammatical properties. Two different words with the same syntactic structure are considered to have the same behavior.

Main phrase structure rule as explained in chapter 2, states that a sentence must at least be comprised of a subject (NP) and a verb (VP); this rule is modeled below.

S →NP VP

The grammatical properties that comes after the verb such as preposition followed by noun phrase falls under the parameters of the main verb phrase. Example on how sentences were analyzed using the main phrase structure rule.

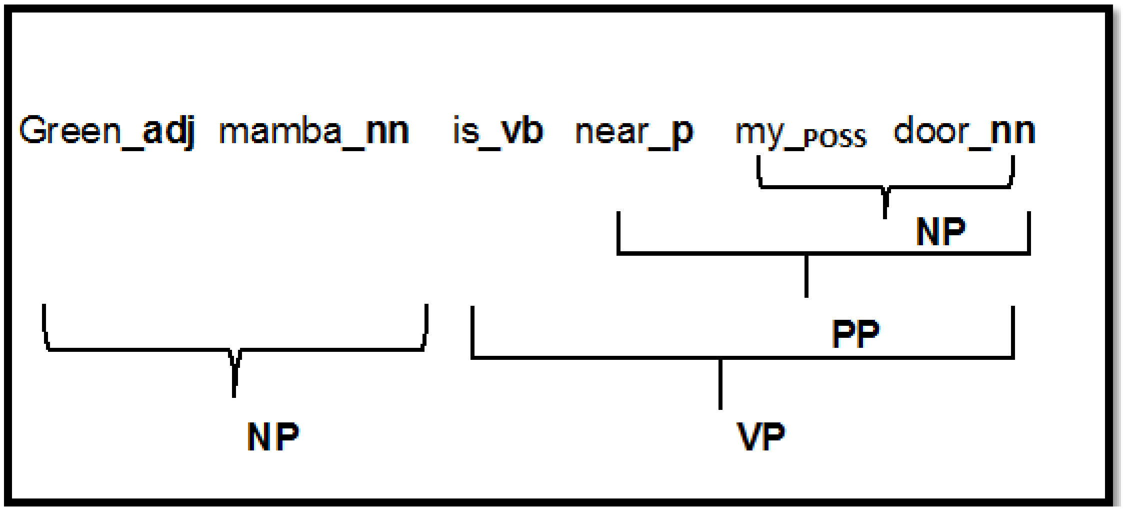


Figure 5.13: Syntactic structure of a sentence

From figure 5.13 above, **NP** is buildup of **ADJ** and **NN**, **VP** is buildup of **VB** and **PP**, **PP** is buildup of **P** and **NP**, **NP** is buildup of **POSS** and **NN**, this is simplified through the use of CFG rules presented below:

S→NP VP

NP→ADJ NN

VP→VB PP

PP→P NP

NP→POSS NN

Parse trees are formed by the CFG rules presented above. Parse trees are modeled to provide the clear view on how the syntactic analyzer understand, analyze and cluster different inputted sentences. Many inputted sentences can be analyzed using only one parse tree as long as they partake the same language structural pattern.

Parse tree below is formed by analyzing the sentence “green mamba is near my door”.

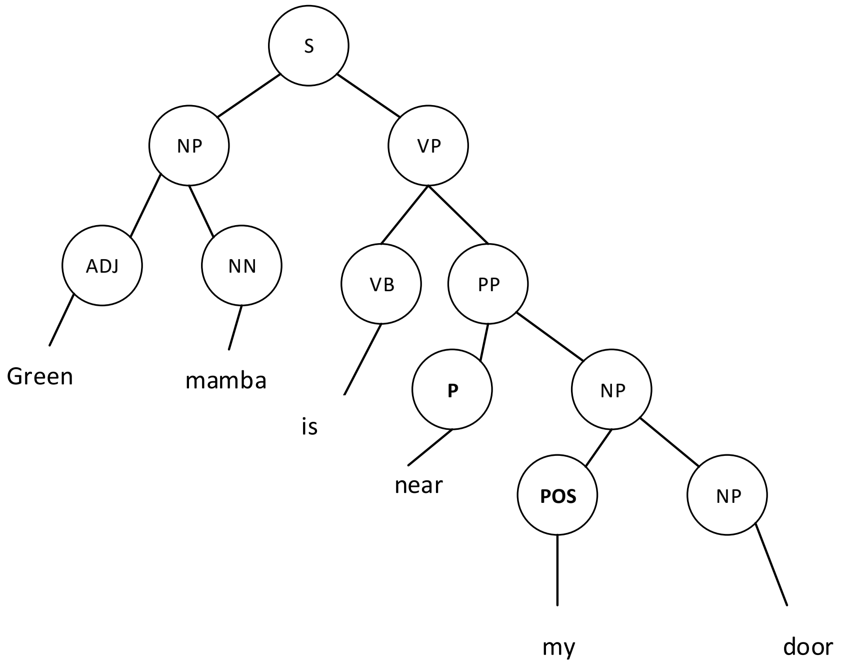


Figure 5.14: Parse tree formed by the CFG rules

From figure 5.14 above, the above part of a parse tree “S”, is known as a root, the tags known as branches and the words known as the leaves. The parse tree is normally modeled from top to bottom and left to right.

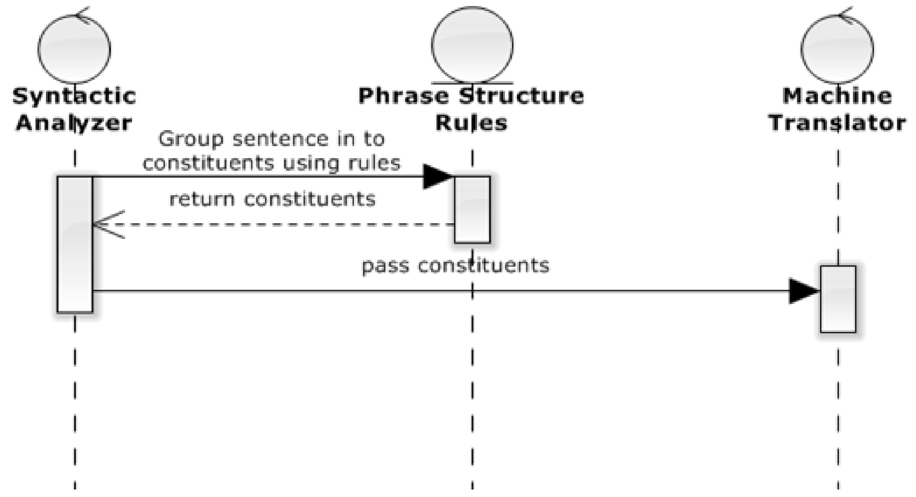


Figure 5.15: Sequence diagram of the syntactic analyzer

From figure 5.15 above, when an inputted sentence has been syntactically analyzed into constituents, each constituent is sent to the machine translator for translation.

5.2.3.3 Semantic Analyzer Design

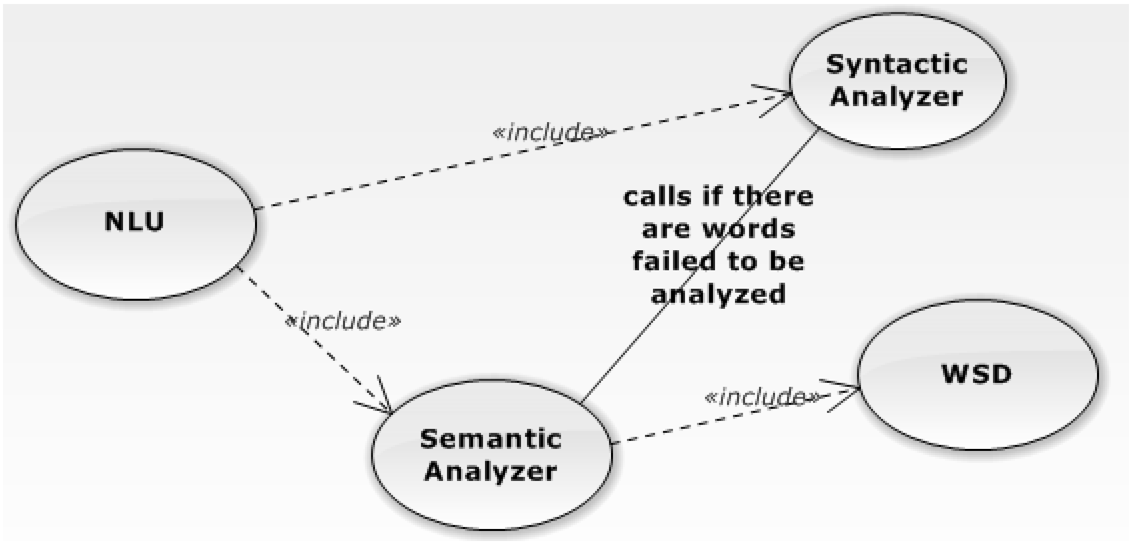


Figure 5.16: Use case of the semantic analyzer, the third stage of the NLU

From figure 5.16 above, the semantic analyzer is called by the syntactic analyzer if there are words in an inputted sentence that cannot be syntactically analyzed; this includes words with more than one part of speech tags. The words such as “book” or “close” inputted by the user, cannot be analyzed by the syntactic analyzer, since they have more than one meaning which is determined by more than one part of speech tag. With the help of semantic analyzer, the machine is capable of evaluating and resolving the ambiguity. The semantic analyzer is buildup of word sense disambiguation (WSD) model, this model is equipped with single word ambiguity detection (SWAD) and in-text word ambiguity detection (IWAD). SWAD is called when a single word such as “book” has been inputted by the user, it prompts the user to specify the “book” they referring to, “book” as noun or as a verb or “close” as a preposition or as a verb. The IWAD is designed in a more sophisticated way, to be able to disambiguate a sentence that is embedded with words ambiguity such as “*may_**NN/VB** I_**NN** book_**NN/VB** your_**POS** book_**NN/VB**”. It* disambiguates the word ambiguity using the word’s context. From above context, it should be noted that the IWAD will ‘notice’ that “may” is not a “may” of a month because it is followed by the noun instead of a verb, first “book” has been recognized as a verb because it is preceded by a noun instead of a verb. The second “book” is recognized as a noun because it is preceded by the possessive part of speech tag.

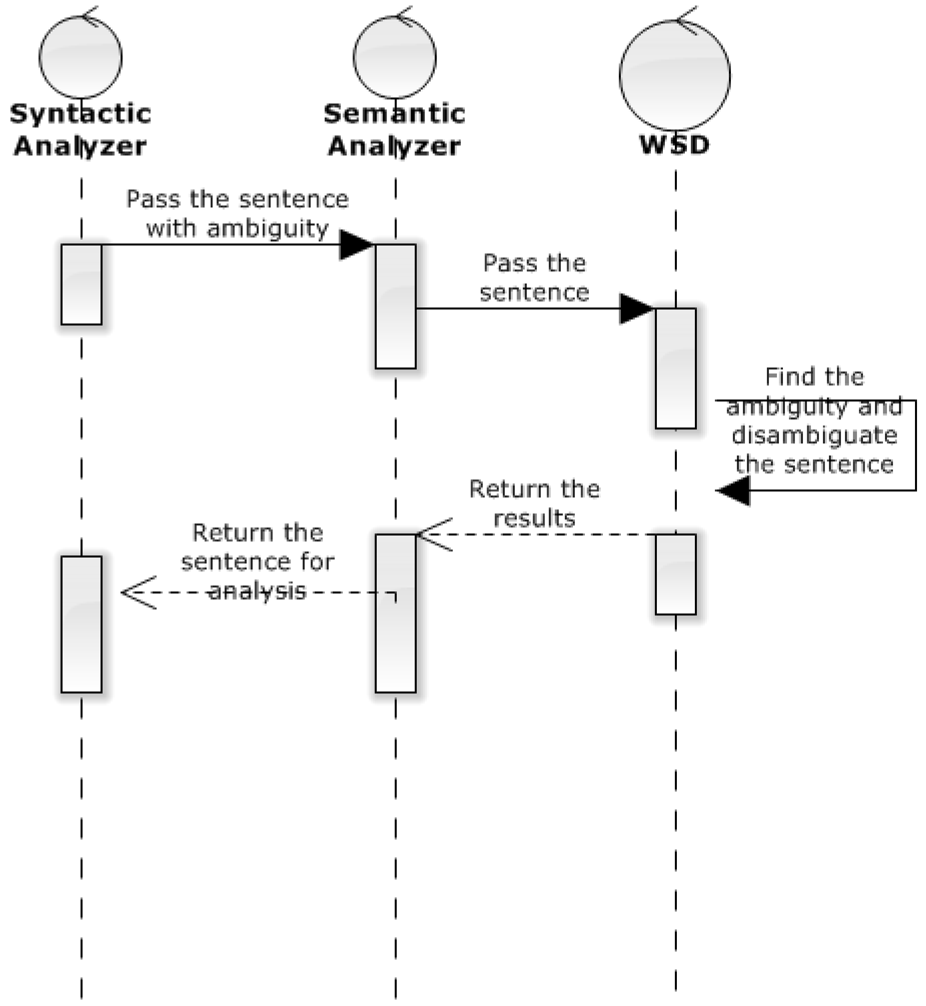

Syntactic Analyzer
Semantic Analyzer
WSD
Pass the sentence with ambiguity
Pass the sentence
Find the ambiguity and disambiguate the sentence
Return the results
Return the sentence for analysis

Figure 5.17: Sequence diagram of the semantic analyzer operations

Figure 5.17 above, after the sentence has been semantically analyzed, it is sent back to the syntactic analyzer for further processing.

The syntactically analyzed constituents are now sent to the machine translator for translation

5.4 Machine Translator Design

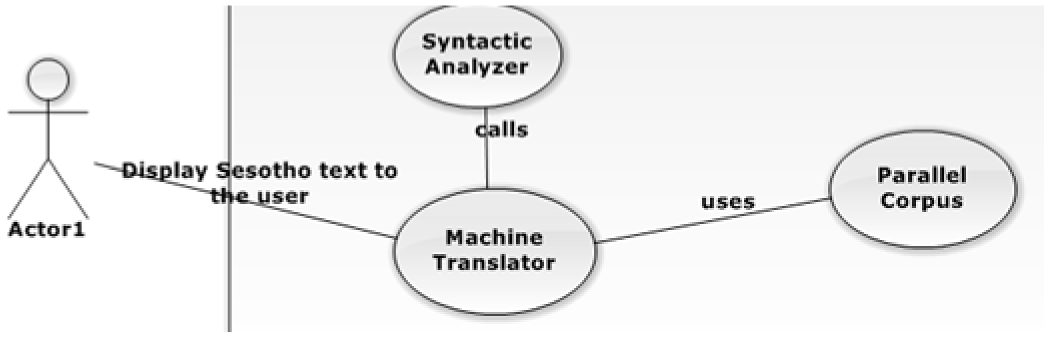


Figure 5.18: Use case of the machine translator

From figure 5.18 above, the machine translator compares each inputted constituent to the list of existing constituents in a parallel corpus's source language, when the match is found the Sesotho translated text is extracted and displayed on a screen for user view.

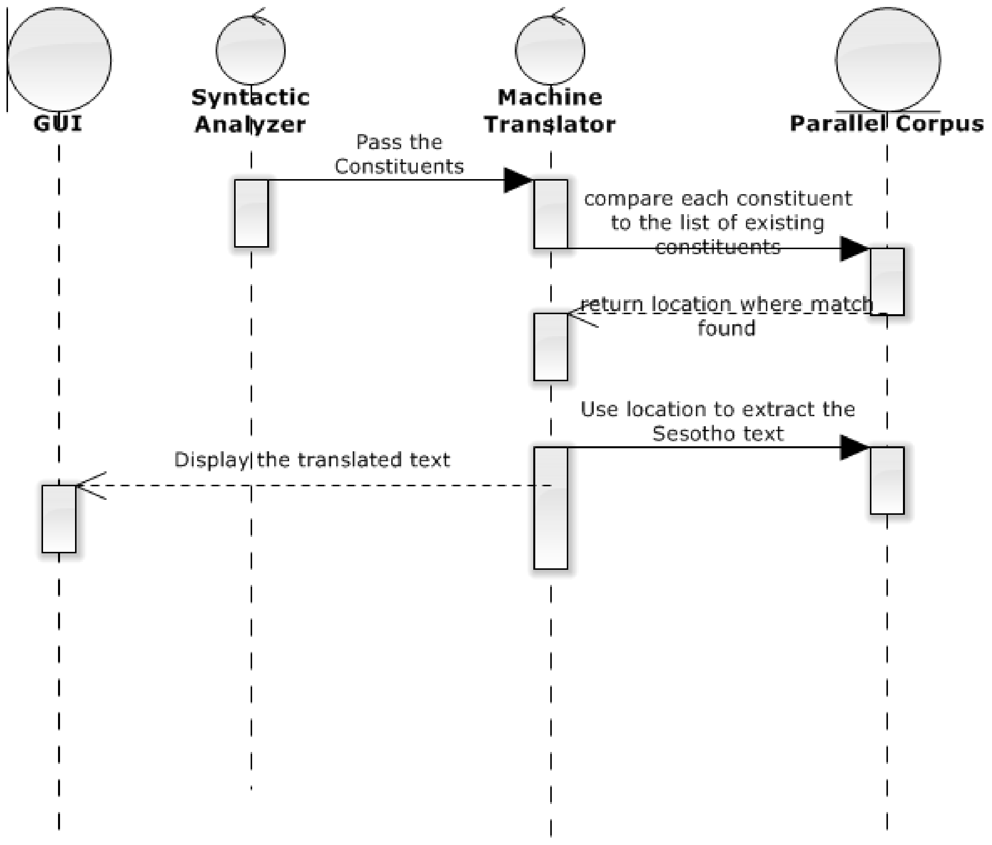

GUI
Syntactic Analyzer
Machine Translator
Parallel Corpus
Pass the Constituents
compare each constituent to the list of existing constituents
return location where match found
Use location to extract the Sesotho text
Display the translated text

Figure 5.19: Sequence diagram showing machine translator operations

From the figure 5.19 above, the syntactic analyzer passes each inputted constituent to the machine translator that compares them to the list of existing constituents in a parallel corpus's source language. For each inputted constituent, each line in a source corpus is broken down into constituents, and each source constituent is compared to the inputted constituent. The comparison takes place on each line, until the inputted constituent has been matched. Once the match is found then the specific Sesotho translation is extracted using the tag value (ASCII character code) and is stored in a list that is manipulated after machine translator terminates.

5.4 Sesotho and English TTS Design

The user is provided with the TTS technology to play the Sesotho translated text.

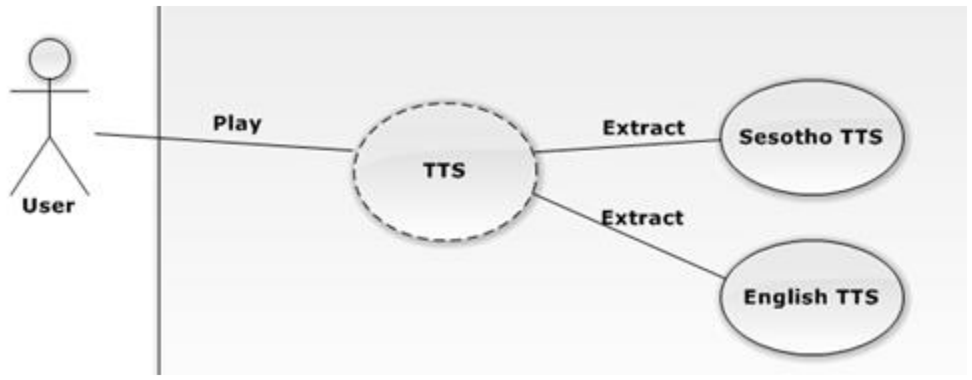


Figure 5.20: Use case of the TTS collaboration

The Sesotho words are played using Sesotho TTS. Words failed to be read by Sesotho TTS are assumed to be proper nouns, and they are read by English TTS.

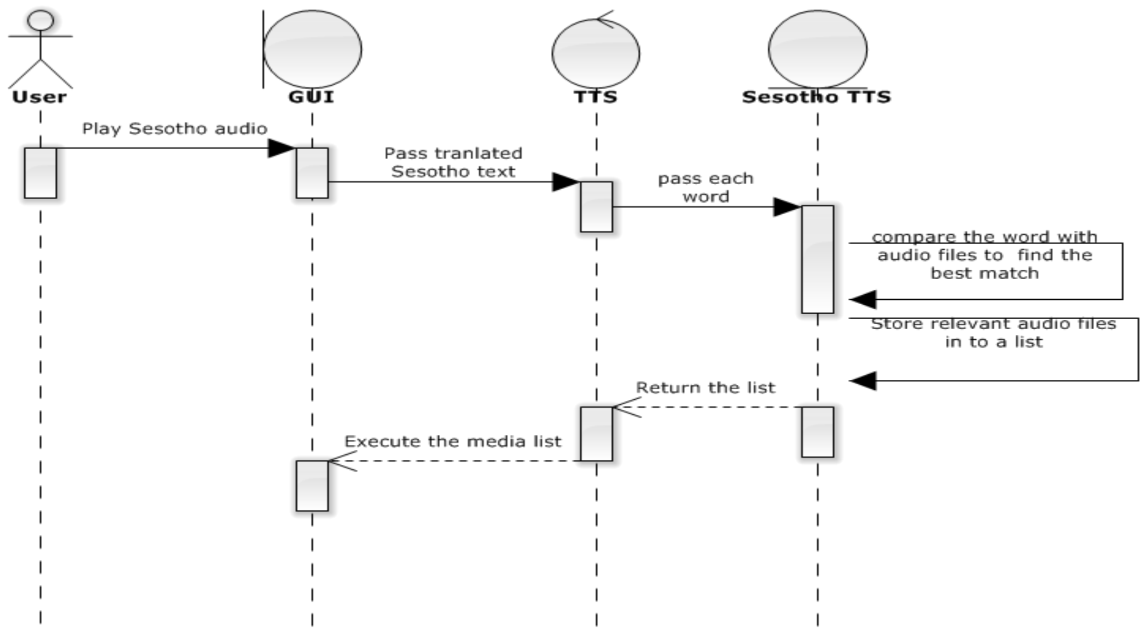

User
GUI
TTS
Sesotho TTS
Play Sesotho audio
Pass tranlated Sesotho text
pass each word
compare the word with audio files to find the best match
Store relevant audio files in to a list
Return the list
Execute the media list

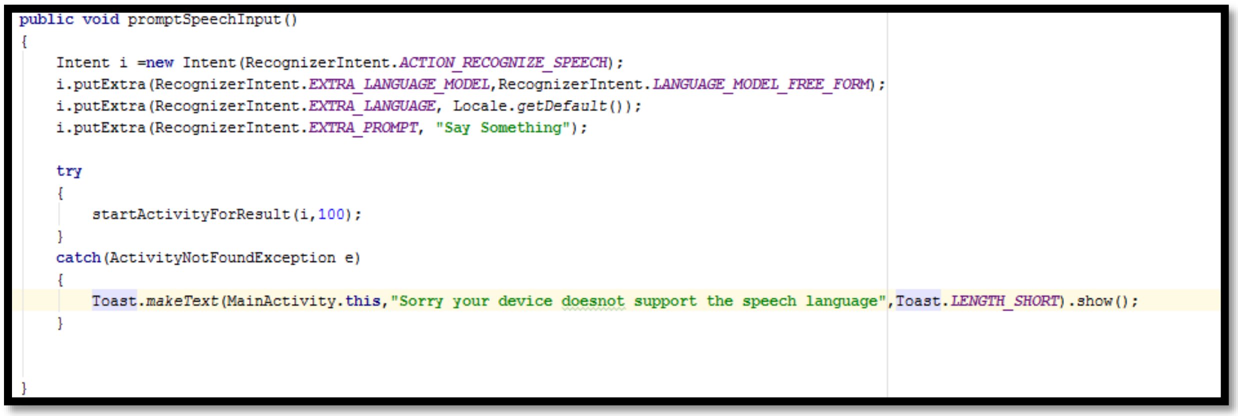
Figure 5.21: Sequence diagram of the Sesotho TTS

From figure 5.21 above, the user presses the play button on a screen, GUI passes the translated text(s) to the speech synthesizer, which tokenizes the sentence in to words, each word is sent to the Sesotho TTS. Sesotho TTS compares each word with the audio files' transcripts to find the best match. When the match is found, the relevant audio file is stored in a media player list that is later manipulated. After all audio files have been added to the media player list, then the list is returned back to the TTS module that is executed.

5.5 Implementation Phase

Android programing language was used to implement the system. The code segments that links with system design are elaborated. For ASR and English TTS we used Google's off-the-shelf source codes.

5.5.1 Speech Recognition implementation



```
public void promptSpeechInput()
{
    Intent i =new Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH);
    i.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL,RecognizerIntent.LANGUAGE_MODEL_FREE_FORM);
    i.putExtra(RecognizerIntent.EXTRA_LANGUAGE, Locale.getDefault());
    i.putExtra(RecognizerIntent.EXTRA_PROMPT, "Say Something");

    try
    {
        startActivityForResult(i,100);
    }
    catch(ActivityNotFoundException e)
    {
        Toast.makeText(MainActivity.this,"Sorry your device doesnot support the speech language",Toast.LENGTH_SHORT).show();
    }


}
```

Figure 5.22: Program code of the Google speech recognition

From the screen short above, the speech recognizer is set to receive the speech from the user by enabling the prompting option. The English language is set as a default language for speech understanding operation. The prompting message "say something" is added on a prompting menu. The "try catch" clause is used to catch "ActivityNotFound" exception. This exception is basically thrown when the machine does not support Google speech technology.

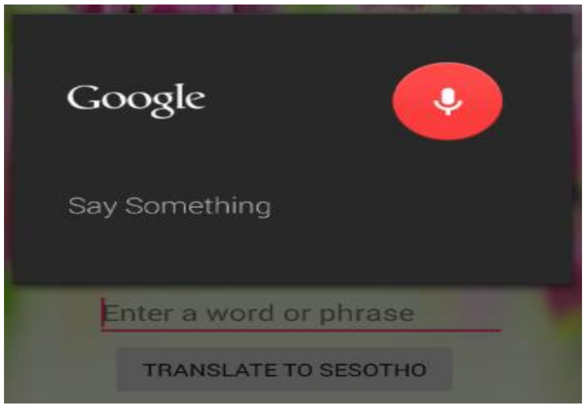
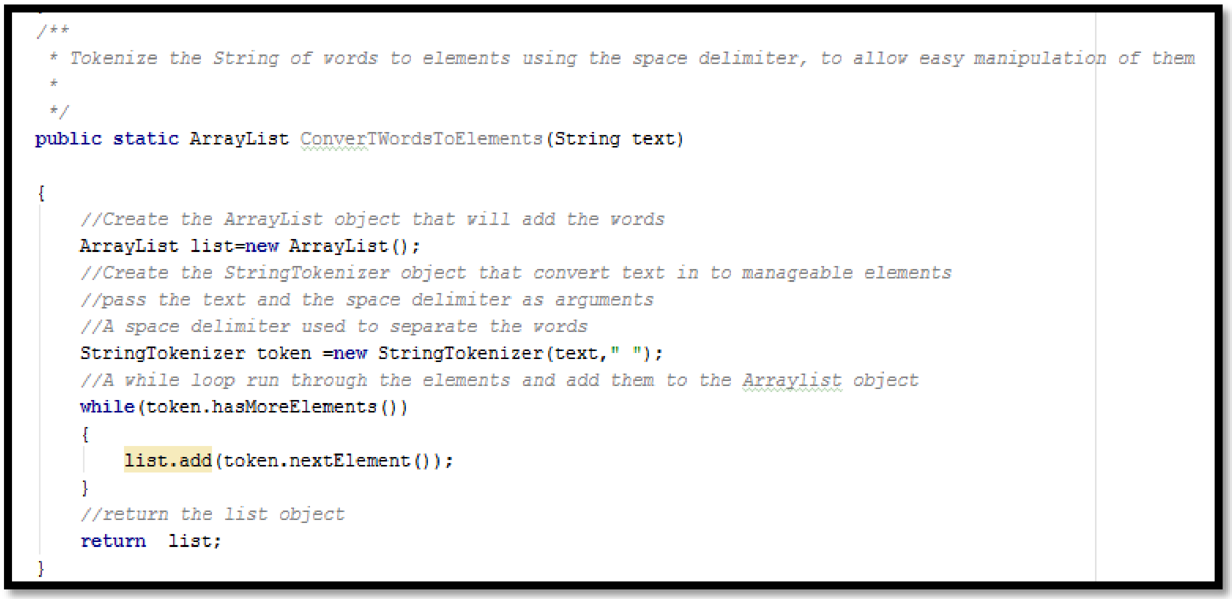


Figure 5.23: GUI of the speech recognizer prompting for user input

From figure 5.23 above, the user is prompted to input the speech, the spoken word(s) is displayed on a text box for user review.

5.5.2 Data Cleaning implementation

Immediately after button “translate to Sesotho” is clicked, the inputted string is sent to the static method ConvertSentenceToElements() of type Array List, it accepts the string as a parameter. It tokenizes the string in to tokens (words), each token is added to the Array List object with the aim to treat tokens as elements to allow easy manipulation of them.



```
/**
 * Tokenize the String of words to elements using the space delimiter, to allow easy manipulation of them
 *
 */
public static ArrayList ConverTWordsToElements(String text)

{
    //Create the ArrayList object that will add the words
    ArrayList list=new ArrayList();
    //Create the StringTokenizer object that convert text in to manageable elements
    //pass the text and the space delimiter as arguments
    //A space delimiter used to separate the words
    StringTokenizer token =new StringTokenizer(text," ");
    //A while loop run through the elements and add them to the Arraylist object
    while(token.hasMoreElements())
    {
        list.add(token.nextElement());
    }
    //return the list object
    return  list;
}
```

Figure 5.24: Program code of method ConvertSentenceToElements().

The list of elements is sent to the method “removeDeterminers()”; It loops through the elements to find determiners using the Equals() method of a String class to perform comparison; when they are found the remove() method of the Array List class is used to remove each determiner at a specified index.

The list is then sent to “removeDelimiters()” method, it loops through the elements, it uses the contains() method of String class to check words with specified delimiters, that are passed as arguments to the method. The delimiters are removed from the text, using the StringTokenizer object.

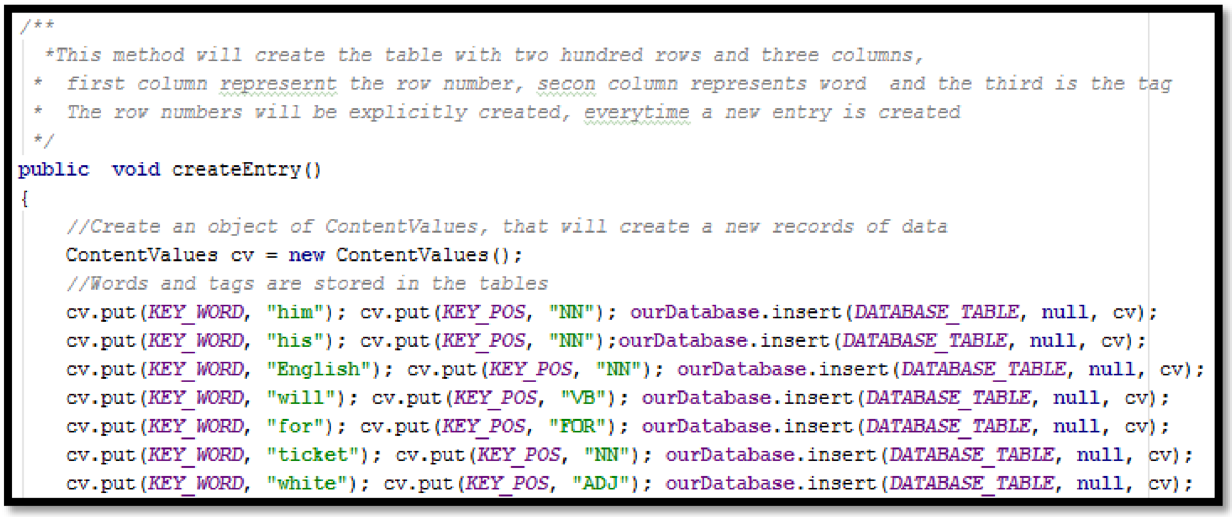
Finally, the list is then sent to “contractionRemover()” method to resolve the contracted words in to standard words. This is performed by looping through the list, use the decision structures to compare each word to existing contracted words, if a word is found, the set() method of the Array List is used to modify the contracted word at a specified position to a standard word.

5.5.3 Natural Language Understanding implementation

Implementation of three NLU processes:

5.5.3.1 Morphological Analyzer implementation

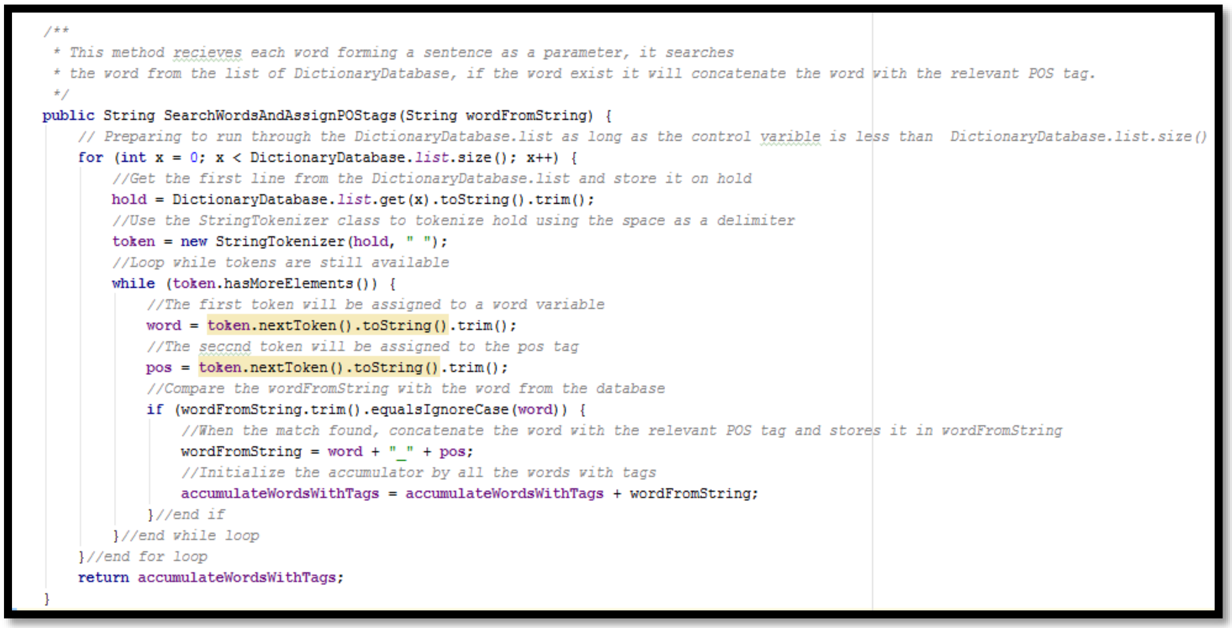
The table of a machine dictionary database was manually implemented with two columns and approximately 200 rows. The first column stores lexical items and second column stores their POS tags.



```
/**
  *This method will create the table with two hundred rows and three columns,
 *  first column represernt the row number, secon column represents word  and the third is the tag
 *  The row numbers will be explicitly created, everytime a new entry is created
 */
public  void createEntry()
{
    //Create an object of ContentValues, that will create a new records of data
    ContentValues cv = new ContentValues();
    //Words and tags are stored in the tables
    cv.put(KEY_WORD, "him"); cv.put(KEY_POS, "NN"); ourDatabase.insert(DATABASE_TABLE, null, cv);
    cv.put(KEY_WORD, "his"); cv.put(KEY_POS, "NN");ourDatabase.insert(DATABASE_TABLE, null, cv);
    cv.put(KEY_WORD, "English"); cv.put(KEY_POS, "NN"); ourDatabase.insert(DATABASE_TABLE, null, cv);
    cv.put(KEY_WORD, "will"); cv.put(KEY_POS, "VB"); ourDatabase.insert(DATABASE_TABLE, null, cv);
    cv.put(KEY_WORD, "for"); cv.put(KEY_POS, "FOR"); ourDatabase.insert(DATABASE_TABLE, null, cv);
    cv.put(KEY_WORD, "ticket"); cv.put(KEY_POS, "NN"); ourDatabase.insert(DATABASE_TABLE, null, cv);
    cv.put(KEY_WORD, "white"); cv.put(KEY_POS, "ADJ"); ourDatabase.insert(DATABASE_TABLE, null, cv);
```

Figure 5.25: Program code of machine dictionary

The part of speech tagger was implemented, it has the method "SearchWordsAndAssignPOStags()", this method is called repeatedly based on the number of words forming an inputted sentence. It uses a loop and decision structure to iterate and compare each inputted word against dictionary words, to find the best match. During the comparison, it uses the trim() method of a String class to truncate the white spaces before and after the tested words to diverge unnecessary data mismatch so to alleviate comparison. When the match is found, String variable "accumulateWordsWithTags" concatenates a word with its tag(s) separated by an underscore character.



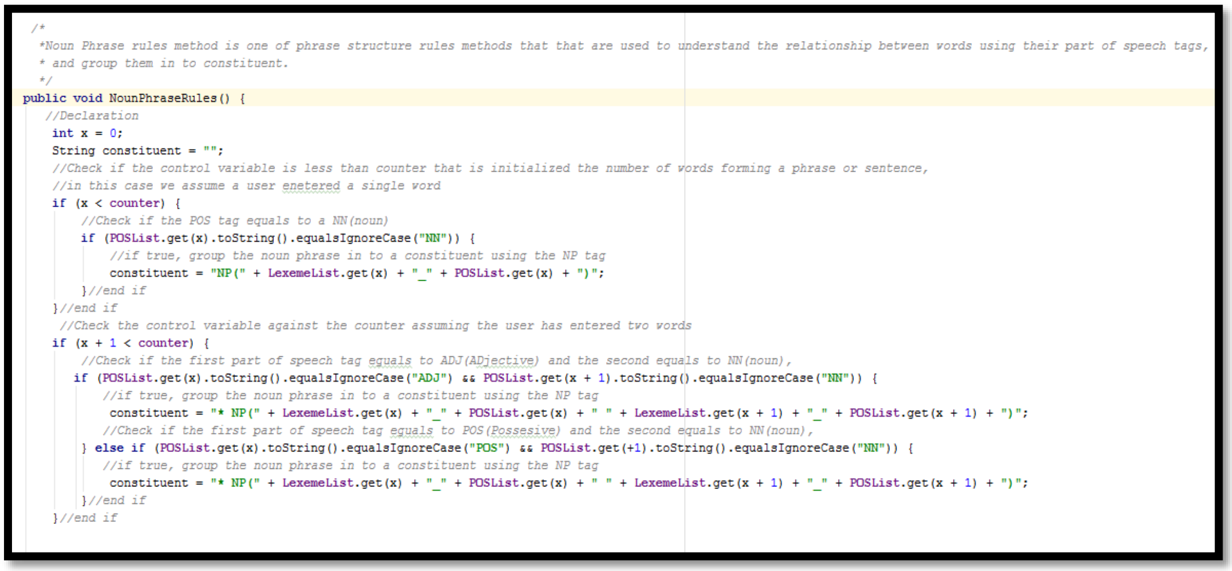
```
/**
 * This method recieves each word forming a sentence as a parameter, it searches
 * the word from the list of DictionaryDatabase, if the word exist it will concatenate the word with the relevant POS tag.
 */
public String SearchWordsAndAssignPOStags(String wordFromString) {
    // Preparing to run through the DictionaryDatabase.list as long as the control varible is less than  DictionaryDatabase.list.size()
    for (int x = 0; x < DictionaryDatabase.list.size(); x++) {
        //Get the first line from the DictionaryDatabase.list and store it on hold
        hold = DictionaryDatabase.list.get(x).toString().trim();
        //Use the StringTokenizer class to tokenize hold using the space as a delimiter
        token = new StringTokenizer(hold, " ");
        //Loop while tokens are still available
        while (token.hasMoreElements()) {
            //The first token will be assigned to a word variable
            word = token.nextToken().toString().trim();
            //The seccnd token will be assigned to the pos tag
            pos = token.nextToken().toString().trim();
            //Compare the wordFromString with the word from the database
            if (wordFromString.trim().equalsIgnoreCase(word)) {
                //When the match found, concatenate the word with the relevant POS tag and stores it in wordFromString
                wordFromString = word + "_" + pos;
                //Initialize the accumulator by all the words with tags
                accumulateWordsWithTags = accumulateWordsWithTags + wordFromString;
            }//end if
        }//end while loop
    }//end for loop
    return accumulateWordsWithTags;
}
```

Figure 5.26: Program segment of the part of speech tagger operation

After the execution of part of speech tagger, the partially analyzed sentence whereby each word has its own part of speech tag is an end results. This sentence is sent to syntactic analyzer.

5.5.3.2 Syntactic Analyzer implementation

As it was explained that the syntactic analyzer understand the inputted sentence through its syntactic structure; the part of speech tags are extracted from the partially analyzed sentence and are stored in an Array List called POSTagList in their context order. The words are also extracted and stored in another Array List called LexisList. The data from these two arrays is parallel aligned. The tags from POSTaglist are manipulated, to check the grammatical dependency; the tags go through phrase structure rules conditions or decision structures to determine their dependencies. Where there is a dependency, words with tags are grouped in to constituent, using a specific phrase tag and parentheses. For the inputted string that has the subject and a verb; the location of the verb is first determined; a constituent from the left hand side of a verb is evaluated using the noun phrase structure rules, and the right portion including the verb is evaluated using the verb phrase structure rules.



```
 /*
  *Noun Phrase rules method is one of phrase structure rules methods that that are used to understand the relationship between words using their part of speech tags,
  * and group them in to constituent.
  */
public void NounPhraseRules() {
   //Declaration
    int x = 0;
    String constituent = "";
    //Check if the control variable is less than counter that is initialized the number of words forming a phrase or sentence,
    //in this case we assume a user enetered a single word
    if (x < counter) {
        //Check if the POS tag equals to a NN(noun)
        if (POSList.get(x).toString().equalsIgnoreCase("NN")) {
            //if true, group the noun phrase in to a constituent using the NP tag
            constituent = "NP(" + LexemeList.get(x) + "_" + POSList.get(x) + ")";
        }//end if
    }//end if
     //Check the control variable against the counter assuming the user has entered two words
    if (x + 1 < counter) {
        //Check if the first part of speech tag equals to ADJ(ADjective) and the second equals to NN(noun),
       if (POSList.get(x).toString().equalsIgnoreCase("ADJ") && POSList.get(x + 1).toString().equalsIgnoreCase("NN")) {
           //if true, group the noun phrase in to a constituent using the NP tag
            constituent = "* NP(" + LexemeList.get(x) + "_" + POSList.get(x) + " " + LexemeList.get(x + 1) + "_" + POSList.get(x + 1) + ")";
           //Check if the first part of speech tag equals to POS(Possesive) and the second equals to NN(noun),
        } else if (POSList.get(x).toString().equalsIgnoreCase("POS") && POSList.get(+1).toString().equalsIgnoreCase("NN")) {
           //if true, group the noun phrase in to a constituent using the NP tag
            constituent = "* NP(" + LexemeList.get(x) + "_" + POSList.get(x) + " " + LexemeList.get(x + 1) + "_" + POSList.get(x + 1) + ")";
        }//end if
    }//end if
```

Figure 5.27: Program segment of the syntactic analyzer operations

5.5.3.3 Semantic Analyzer implementation

From figure 5.28 below, SWAD is invoked when an inputted single word with ambiguity is translated. GUI example on how the machine respond is presented below.

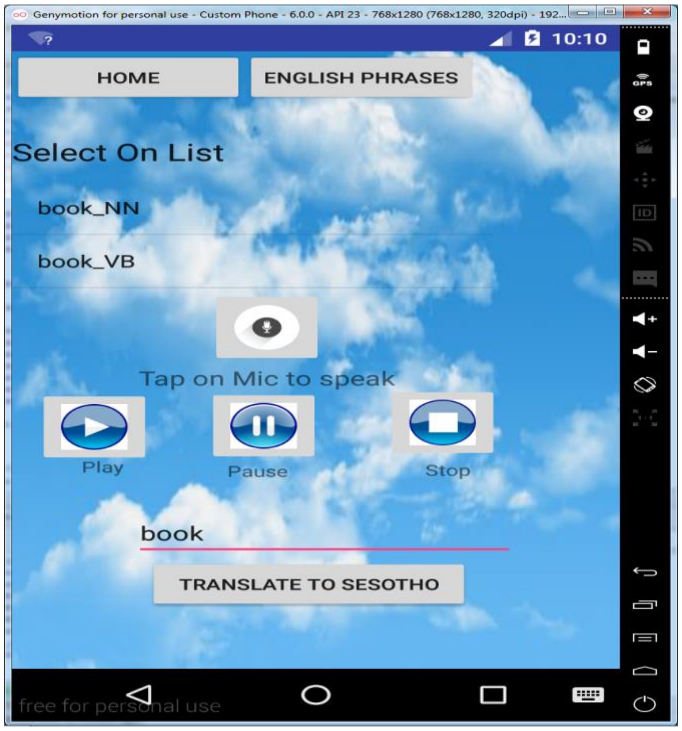
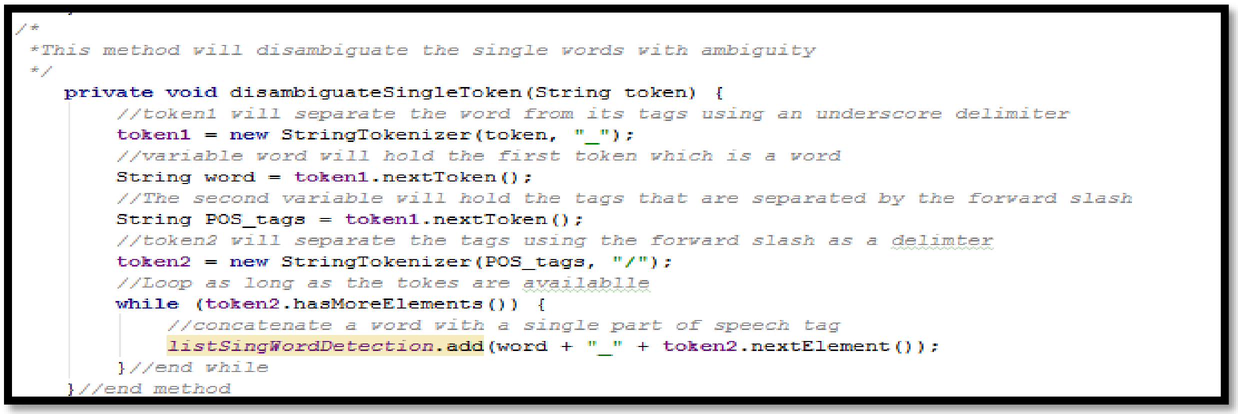


Figure 5.28: Single word ambiguity detection

This is achieved by small block of code presented below (figure 5.29).

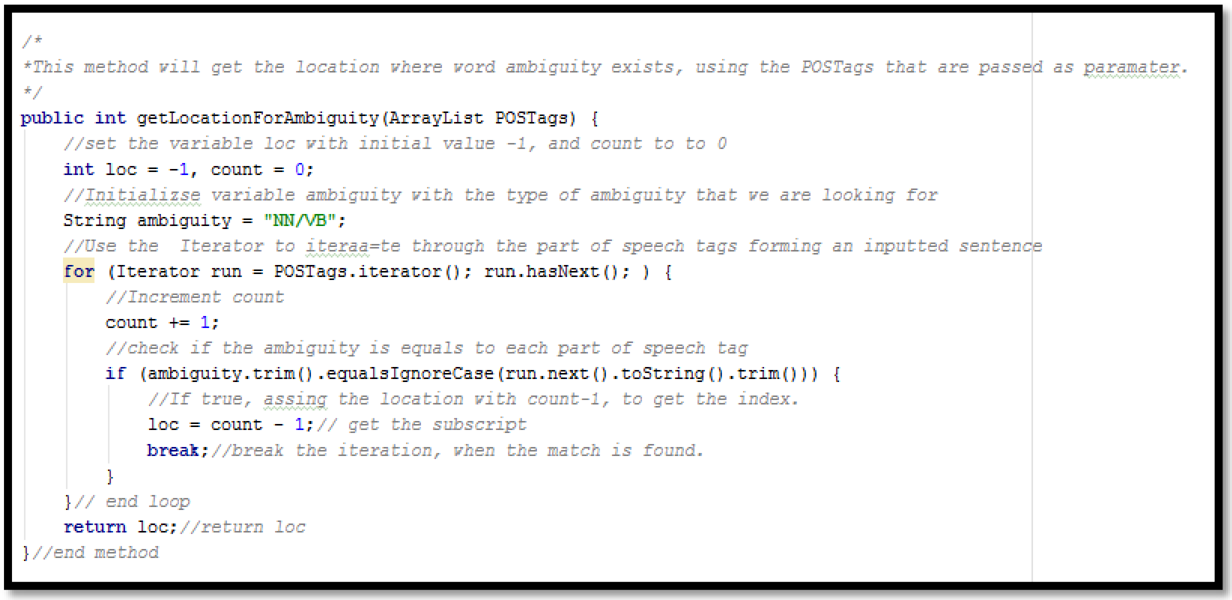


```
/*
 *This method will disambiguate the single words with ambiguity
 */
    private void disambiguateSingleToken(String token) {
        //token1 will separate the word from its tags using an underscore delimiter
        token1 = new StringTokenizer(token, "_");
        //variable word will hold the first token which is a word
        String word = token1.nextToken();
        //The second variable will hold the tags that are separated by the forward slash
        String POS_tags = token1.nextToken();
        //token2 will separate the tags using the forward slash as a delimter
        token2 = new StringTokenizer(POS_tags, "/");
        //Loop as long as the tokes are availablle
        while (token2.hasMoreElements()) {
            //concatenate a word with a single part of speech tag
            listSingWordDetection.add(word + "_" + token2.nextElement());
        }//end while
    }//end method
```

Figure 5.29: Program code of the single word ambiguity detection

From the figure 5.29 above, the method “disambiguateSingleToken()” receives the ambiguity token such as “book_NN/VB”. It uses the StringTokenizer class to separate the word from its tags using an underscore character as a delimiter; the combined tags separated by a forward slash “NN/VB” are further tokenized using forward slash as a delimiter. The “while loop” is used to add a word to both tags. Ending up with duplicate words tagged to different POS tags. The words with their tags are stored in to a List View that is displayed to the user, for selection.

The IWAD has a method “getLocationOfAmbiguity()” as depicted on a code segment below. This method uses an ArrayList.Iterator to iterate through the POSTagList to find the ambiguity.



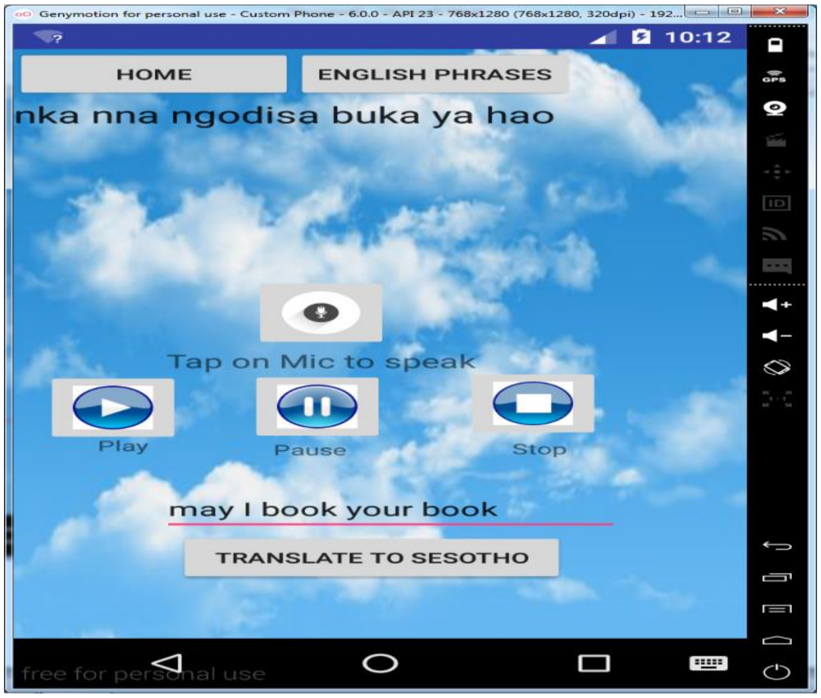
```
/*
*This method will get the location where word ambiguity exists, using the POSTags that are passed as paramater.
*/
public int getLocationForAmbiguity(ArrayList POSTags) {
    //set the variable loc with initial value -1, and count to to 0
    int loc = -1, count = 0;
    //Initializse variable ambiguity with the type of ambiguity that we are looking for
    String ambiguity = "NN/VB";
    //Use the  Iterator to iteraa=te through the part of speech tags forming an inputted sentence
    for (Iterator run = POSTags.iterator(); run.hasNext(); ) {
        //Increment count
        count += 1;
        //check if the ambiguity is equals to each part of speech tag
        if (ambiguity.trim().equalsIgnoreCase(run.next().toString().trim())) {
            //If true, assing the location with count-1, to get the index.
            loc = count - 1;// get the subscript
            break;//break the iteration, when the match is found.
        }
    }// end loop
    return loc;//return loc
}//end method
```

Figure 5.30: semantic analyzer code, showing method getLocationOfAmbiguity()

Once the ambiguity’s location is found, then the IWAD resolves ambiguity by analyzing the POS tags that precedes or success the ambiguity. For the ambiguity (NN/VB), the method “testForVerb()” is invoked; this method analyzes the successive and preceding POS tags, it has the condition that says if a preceding part of speech tag is a noun, or successive POS tag is a noun or possessive or preposition, then the ambiguity is resolved in to a verb. Another method that is secondly invoked is “testForNoun()”, this method analyzes the preceding POS tag before ambiguity. It has the condition that says if the preceding POS tag is a determiner or adjective or possessive or verb, then the ambiguity is resolved in to a noun.

Lastly method “testForPreposition()” is called when the (VB/P) ambiguity is detected. The word “close” result in to this ambiguity. The method checks if the successive part of speech tag is a preposition, then assigns the ambiguity the preposition tag; else it assigns the ambiguity the verb tag.

As depicted in a figure 5.31 below, the sentence with in-text ambiguity is inputted by the user; the machine managed to resolve the ambiguity and successfully translate the sentence.


Genymotion for personal use - Custom Phone - 6.0.0 - API 23 - 768x1280 (768x1280, 320dpi) - 192...
10:12
HOME
ENGLISH PHRASES
Tap on Mic to speak
Play
Pause
Stop
TRANSLATE TO SESOTHO
free for personal use

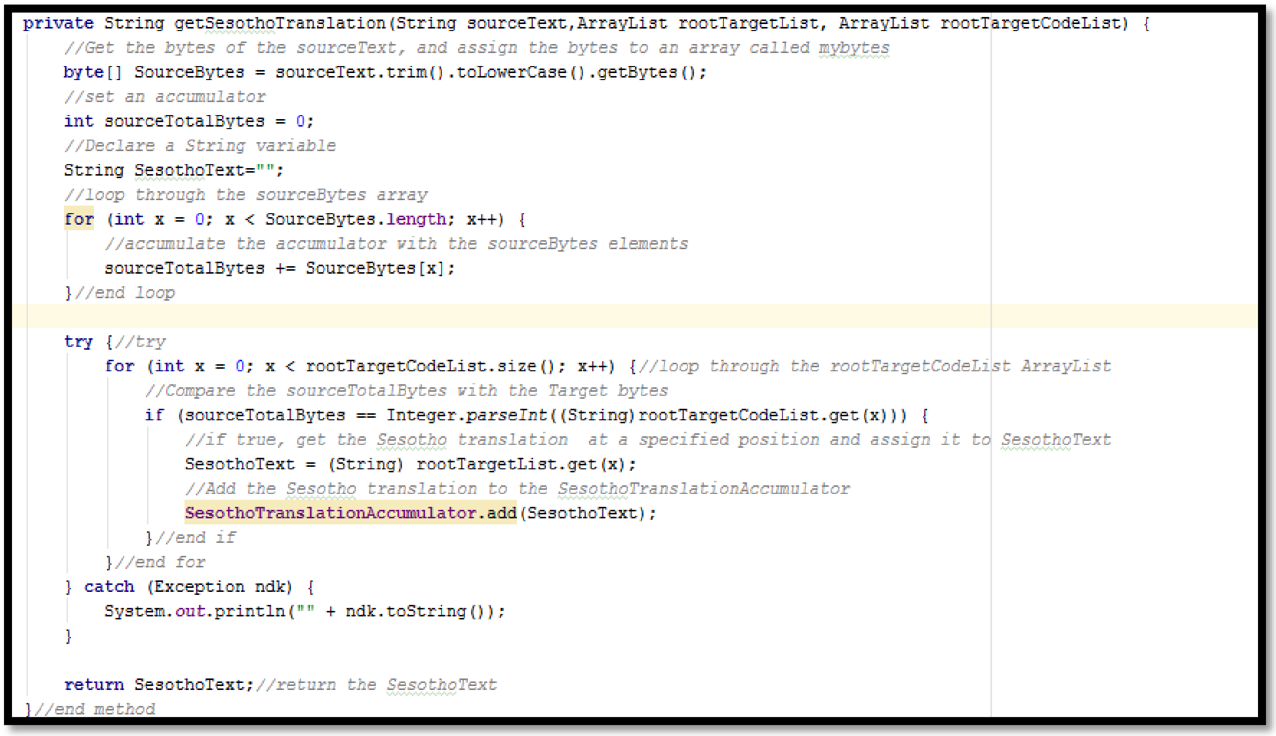
nka nna ngodisa buka ya hao

may I book your book

Figure 5.31: GUI showing the disambiguated in-text ambiguity

5.5.4 Machine Translator Implementation

The machine translator receives the syntactically analyzed input from the syntactic analyzer, each line from the source corpus and the target corpus; that are sequentially passed as arguments using the loop. At this stage the data from both corpora and input have already been analyzed in to constituents separated by the delimiter (“*”). Each inputted constituent is compared against the source corpus constituents on each line to find the best match. When the match is found, method “getSesothoTranslation()” is provoked. This method receives three arguments which are the matched source constituent, array holding target Sesotho translations that are parallel aligned with the matched source constituent, and array holding their target numerical codes (codes that link the source constituents with the target translations, as described in chapter 4) as parameters. This method calculates the total character codes of the matched source constituent; it compares the calculated total against the target numerical codes. When the match is found the linked target translation is retrieved and added into the Array List "SesothoTranslationAccumulator" that is later manipulated.

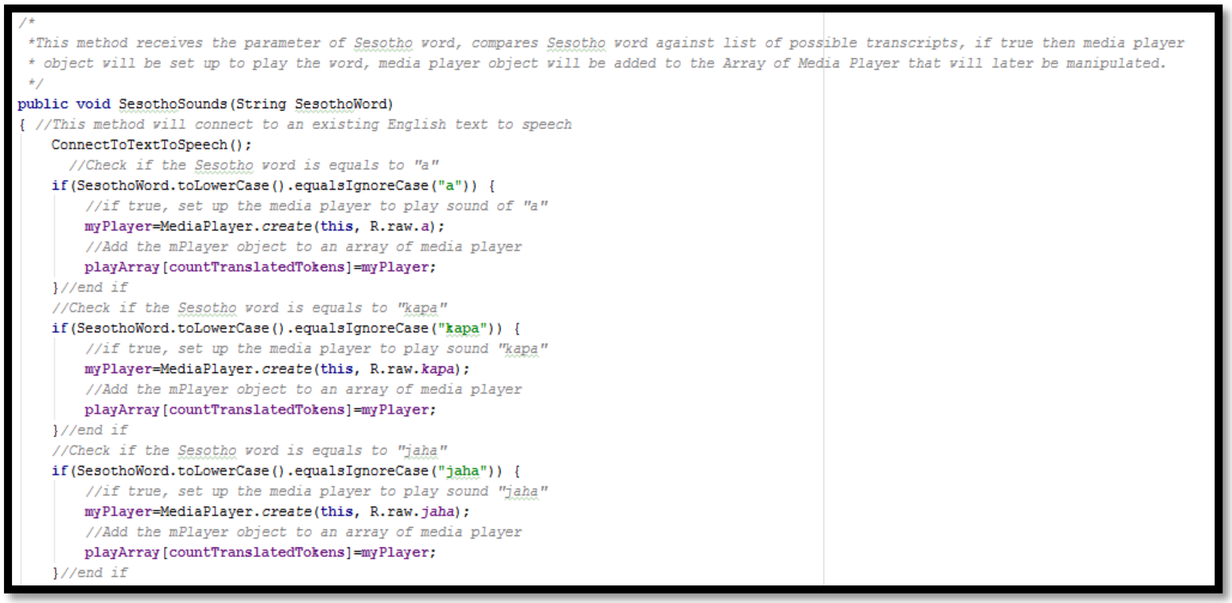


```java
private String getSesothoTranslation(String sourceText,ArrayList rootTargetList, ArrayList rootTargetCodeList) {
    //Get the bytes of the sourceText, and assign the bytes to an array called mybytes
    byte[] SourceBytes = sourceText.trim().toLowerCase().getBytes();
    //set an accumulator
    int sourceTotalBytes = 0;
    //Declare a String variable
    String SesothoText="";
    //loop through the sourceBytes array
    for (int x = 0; x < SourceBytes.length; x++) {
        //accumulate the accumulator with the sourceBytes elements
        sourceTotalBytes += SourceBytes[x];
    }//end loop

    try {//try
        for (int x = 0; x < rootTargetCodeList.size(); x++) {//loop through the rootTargetCodeList ArrayList
            //Compare the sourceTotalBytes with the Target bytes
            if (sourceTotalBytes == Integer.parseInt((String)rootTargetCodeList.get(x))) {
                //if true, get the Sesotho translation  at a specified position and assign it to SesothoText
                SesothoText = (String) rootTargetList.get(x);
                //Add the Sesotho translation to the SesothoTranslationAccumulator
                SesothoTranslationAccumulator.add(SesothoText);
            }//end if
        }//end for
    } catch (Exception ndk) {
        System.out.println("" + ndk.toString());
    }

    return SesothoText;//return the SesothoText
}//end method
```

Figure 5.32: Program code of the machine translator operation

5.5.5 Collaboration of Sesotho TTS and Google TTS Implementation

The translated Sesotho text(s) is displayed on a screen. The user presses the "play" button to read the translation using Sesotho TTS. This code segment below, shows how Sesotho words are read by the machine using Media player list. Since the Sesotho TTS it designed to forecast the existing Sesotho words from the corpus; Google TTS is used to read unknown words.



```
/*
 *This method receives the parameter of Sesotho word, compares Sesotho word against list of possible transcripts, if true then media player
 * object will be set up to play the word, media player object will be added to the Array of Media Player that will later be manipulated.
 */
public void SesothoSounds(String SesothoWord)
{ //This method will connect to an existing English text to speech
    ConnectToTextToSpeech();
      //Check if the Sesotho word is equals to "a"
    if(SesothoWord.toLowerCase().equalsIgnoreCase("a")) {
        //if true, set up the media player to play sound of "a"
        myPlayer=MediaPlayer.create(this, R.raw.a);
        //Add the mPlayer object to an array of media player
        playArray[countTranslatedTokens]=myPlayer;
    }//end if
    //Check if the Sesotho word is equals to "kapa"
    if(SesothoWord.toLowerCase().equalsIgnoreCase("kapa")) {
        //if true, set up the media player to play sound "kapa"
        myPlayer=MediaPlayer.create(this, R.raw.kapa);
        //Add the mPlayer object to an array of media player
        playArray[countTranslatedTokens]=myPlayer;
    }//end if
    //Check if the Sesotho word is equals to "jaha"
    if(SesothoWord.toLowerCase().equalsIgnoreCase("jaha")) {
        //if true, set up the media player to play sound "jaha"
        myPlayer=MediaPlayer.create(this, R.raw.jaha);
        //Add the mPlayer object to an array of media player
        playArray[countTranslatedTokens]=myPlayer;
    }//end if
```

Figure 5.33: Program code of the TTS operation, showing the Sesotho sounds

From the figure 5.33 above, each translated Sesotho text, is passed as an argument to the method “SesothoSounds()”. This method will compare the Sesotho text with the audio transcriptions to find the best match. When the match is found, the relevant audio file will be retrieved from the file “raw” and will be stored in the media player list that will later be manipulated. When the match is not found, the Google TTS is provoked to read the word.

5.6 Conclusion

The chapter presented was focusing on how the prototype was analyzed and designed, we elaborated the use of UML diagrams by touching on use cases and sequence diagrams. We used Android Studio to develop the prototype. The following chapter dwells on prototype testing and evaluation

Chapter 6: System Testing and Evaluation

6.1 Introduction

Testing and evaluation process is an integral part of the system engineering process which identifies levels of performance and assists the developer in correcting the deficiency (Buede & Miller, 2016). The system testing and evaluation was conducted by the researcher through experiments, to meticulously observe and document the machine translation's speed to process data and translation accuracy.

6.2 Evaluation of Searching Algorithms Using Big O

The asymptotic function which is also known as Big O notation was used to describe the performance or complexity of search algorithms used in the application. Big O notation also known as Big Omega notation is a mathematical notation used to evaluate the performance of an algorithm based on given data set, it evaluates the algorithm using three time-complexity cases which are worst, best and average cases (Danziger, 2015).

The efficiency of a search algorithm was based on number of executions or operations taking place to search the target value, from the given array as an input. The less the number of executions, the faster the algorithm. For a search algorithm, the number of executions are directly proportional to the number of comparisons. Two types of algorithms that were tested were linear and binary search algorithms.

For testing purpose, a single line from the source corpus was broken down in to constituents and were stored in array; these constituents were now considered as the elements of an array. The size of an array is determined by the number of elements it has.

Different types of time complexity cases as described earlier, were evaluated on both algorithms, to find the one with very lowest comparisons, or iterations, resulting in to high performance.

The best case approach was first tested, for this approach the researcher needed to find out fewest comparisons an algorithm can perform to find the desired item (constituent) in an array. An array and a search item (a constituent that will be compared against the constituents in an array, to find the best match) were passed to linear search algorithm as arguments; the desired item was in a first slot of an array. Linear search algorithm took a single comparison to get the search item. The same arguments (array and a search item) were sent to binary search algorithm, the single comparison was also made. Therefore, for best case approach the speed of both algorithms was equal.

The second case was worst case, at this stage the researcher needed the most number of comparisons an algorithm can necessarily take to find the desired item. To comply with this stage, the desired item was in the last slot of an array. We passed the same arguments to linear search algorithm. The results showed that the number of comparisons were proportional or equivalent to the number of elements in an array. This is because, the comparisons were done sequentially on each element to find the best match. Therefore, when the array size grows then the number of comparisons increases. This was determined by the Big O function O (n), meaning if an array has *n* items, then it takes *n* comparisons in the worst case. This is modeled on a graph below.

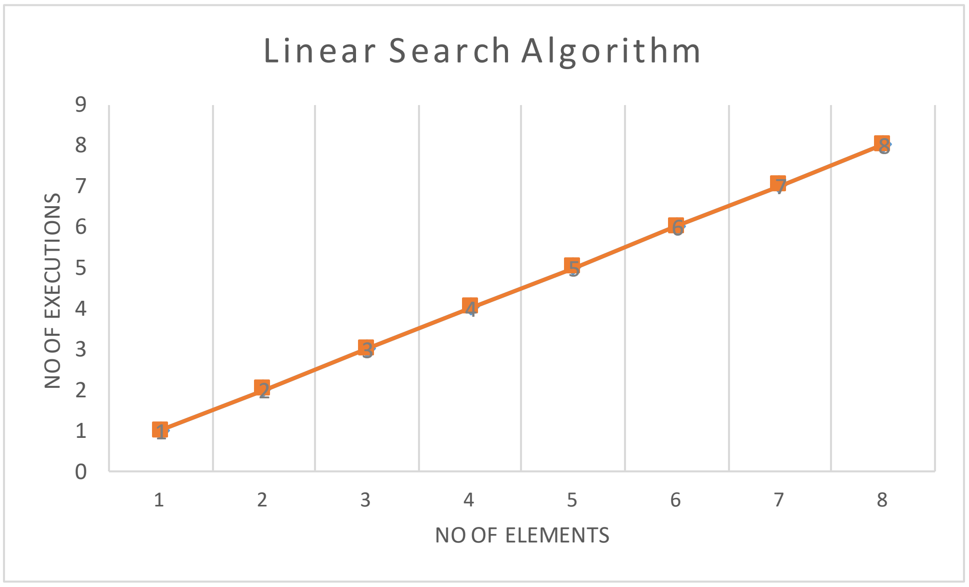


Figure 6.1: Execution time using linear search algorithm

The same arguments were sent to the binary search, the algorithm compares the middle element of an array with the target value; if the target value matches the middle element then loop terminates; if the target value is less than or greater than the middle element, then the search continues in the lower or upper half of the array, respectively, eliminating the other half from consideration (Bentley & Sedgewick, 1997). The end result was represented as O ($\log_2$ n), meaning the array size *n* was broken down in to halves for each comparison; using the base value 2 as a divider. For example, O ($\log_2$ 8) will result in to three comparisons (4, 2, 1). In other words, an array with 8 elements, will require 3 comparisons when using binary search algorithm; and it will require 8 comparisons when using the sequential search known as linear search. This is modeled below.

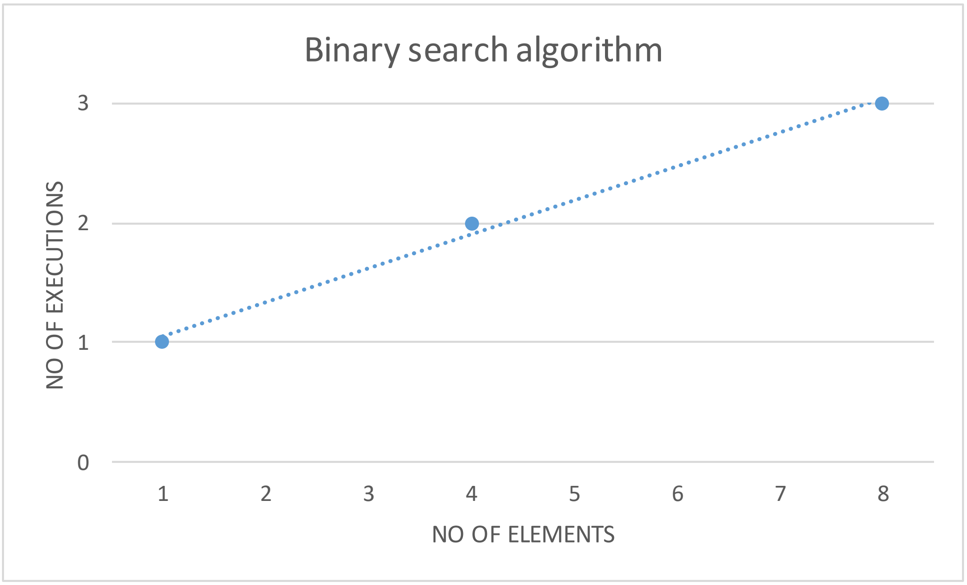


Figure 6.2: Execution time using the binary search algorithm

The average case was thirdly evaluated, in this stage the researcher needed to find out how many comparisons it will take to find an item in the middle of the list. So we placed the desired item in a middle slot of the array. The same arguments were sent to the linear search, the results showed that the number of comparisons were half the size of the input represented by O (n/2). The same arguments were sent to binary search; this algorithm performed only one comparison to find the match.

Based on evaluations of time complexity cases made, the binary search algorithm was selected as the quickest algorithm because of minimal number of comparisons it takes to find a desired item. However, data in an array must first be sorted in to alphabetical order to effectively utilize the binary search algorithm.

We had to evaluate different types of sorting algorithms used and select the best in terms of speed.

6.3 Evaluation of Sorting Algorithms Using Big O

The first algorithm was selection sort. The idea behind this algorithm is that the first element is considered as the minimum element, it will be compared against the rest of the elements or values in the list. If the minimum is greater than the tested value, then the swapping process takes place. During the swapping process, temporary variable is declared and stores the minimum element, then the minimum element is assigned the tested value, and the tested value is assigned the temp. The new minimum element will be tested against the remaining portion of the list. This process takes place until all elements are in alphabetical order.

We evaluated the selection sort algorithm by evaluating its time complexity in terms of swapping and comparisons.

Best case approach was when the array of elements passed to the algorithm as an argument, was already sorted in alphabetical order; worst case approach was when the first value of the array was the largest value and the rest in the list were in alphabetical order; and lastly average case, whereby the largest value was at the middle of the list.

The selection sort's summary time complexity for best case was O (n^2); worst case was O (n^2) and average case was O (n^2). This is a worst performance because the number of comparisons and swapping are squared the number of array items for each time-complexity case. This is because of nested loops, therefore the performance is directly proportional to the square of the input (array) data set.

The second algorithm that was evaluated was bubble sort. For each pass the largest value is swapped until it is dissolved to its proper location. In other words, the largest value is moved through the elements to the last position. According to summary results, best case was O (n), the average case was O (n^2) and the worst case was O (n^2). The performance of this algorithm was poor, however, better than the selection sort in terms of best case approach. In bubble sort's best case approach, the number of swapping and comparisons are equal to the size of the elements in the list.

Among the sorting algorithms that were evaluated, the quick sort algorithm was pointed as the best; it is also known as divide and conquer algorithm. This algorithm specify one element in the array as pivot point. Then values that are smaller than the pivot will be on the left hand side of the pivot, and values greater than pivot will be on the right hand side of the pivot. The values on the left hand side of the pivot will be divided in to halves and be sorted, and the ones on the right hand side will be done the same.

The summary time complexity for quick sort's best case was O (n log n), average case was O (n log n) and worst case was O (n log n). This algorithm was considered as the best compared to other sorting algorithms. Because a function O (n log n) which is similar to O ($\log_n n$), means the array size is broken down into halves for each comparison. The first *n* denotes the base used to divide the array size and the second *n* denotes the array size.

The binary search and quick sort algorithms were both utilized in the whole system because of their effectiveness and their efficiency.

6.4 Measuring of Translation Accuracy Using WER

Among existing language translation accuracy measures, word error rate (WER) algorithm was used in terms of its simplicity. The aim of using WER was to evaluate the machine translation output known as hypothesis document (hyp) against given translation reference (ref), known to be correct by Sesotho language expert. The machine translation accuracy was evaluated and reported by the Sesotho language expert.

The words from the machine dictionary were used to train the machine translator. These words were interchangeably used to create over 200 different grammatical meaningful sentences that were translated. The size of inputted sentences was ranging from lowest (one word) to highest (6 words), respectively. Each translated Sesotho text (hypothesis document) together with its reference text were sent to the WER algorithm as arguments.

This algorithm was computed as:

$$WER = \frac{S + D + I}{N}$$

S is the number of text substitutions, *D* is the number of deletions, *I* is the number of insertions and N is the number of words in the reference text.

An example on how WER works, given Sesotho hypothetical output and reference text

Hyp: Rona _ ja bohobe

Ref: Rona re ja bohobe

$$WER = \frac{0 + 1 + 0}{4} = \frac{1}{4} = 25\%$$

From the above results, the word error rate is 25%. The lower the word error rate the higher the translation accuracy. The table below shows the number of words per sentence inputted, WER average and translation accuracy.

Table 6.1: Results of WER based on number of inputted words

No of inputted words	**WER(average)**	**Translation Accuracy**
1	0%	100%
2	10%	90%
3	25%	75%
4	32%	68%
5	40%	60%
6	60%	40%

According to the chart above, the higher the number of inputted words, the higher the word error rate, hence, the less the translation accuracy. The degradation of translation accuracy is caused by different languages' grammatical structures. The average WER is 27.8% and the average translation accuracy is 72.2%.

6.5 Conclusion

In this chapter, we evaluated different types of searching algorithms, binary search algorithm was selected as the quickest. However, the effectiveness of this algorithm required the data to be sorted in alphabetical order. We evaluated different types of sorting algorithms using time complexity cases, the decision was made on quick sort algorithm. Both binary search and quicksort were utilized in the application's search algorithms. The WER algorithm was used to calculate the translation accuracy of the machine, and the average results were presented. The next chapter will be discussions and implications that took place during corpus annotation.

Chapter 7: Discussions and Language Implications

7.1 Discussions

In this dissertation the researcher has presented the Umobitalk application, to improve the Sesotho language as one of Bantu languages known to receive less attention in the field of NLP and human computer language(HCL) due to lack of resources.

The literature was fully reviewed based on speech based technologies and language processing tools. With the knowledge obtained, we developed Umobitalk Android application, which capitalizes three important phases which are Google ASR, MT and TTS (Sesotho TTS and Google TTS).This mobile application was implemented using Android Studio, in a sense that android mobiles are concurring the globe. The idea behind Umobitalk, is to recognize an English spoken word or phrase, translate it in to Sesotho language and read the translated text.

Google ASR was used to recognize and convert the speech into text with the aim for user to view or review the recognized text. To support the MT operation, firstly, data cleaning operation was implemented to clean the inputted text by removing delimiters, articles and determiners. Secondly, NLU was implemented with the purpose to understand and analyze the syntactical structure of the inputted text, this operation is encapsulated with morphological, syntactical and semantic analysis. Thirdly, specialized parallel corpus which constitute the English as a source language and Sesotho as a target language was developed. The parallel corpus was chosen as a tool to support language to language translation by the MT, it was implemented in such a way it will cater only basic Sesotho language (words and phrases) that migrants need to know. To guide the implementation of the corpus, the research was conducted to obtain those basic words and phrases based on different language domains that were selected by the Sesotho language expert.

The UmobiTalk was then used to evaluate the correctness and accuracy of the parallel corpus. Since the Sesotho language is highly morphological and syntactical language, the Sesotho corpus was designed in such a way it can keep up with some Sesotho inflection and derivation of words.

For TTS operation, the Sesotho speech synthesizer was manually implemented, and it was integrated with off-the-shelf Google TTS to read existing Sesotho words from Sesotho corpus and unknown words such as English proper nouns. The accessibility of the system was proved by the fact that it can work without accessing the internet, unless Google speech technology is used. If so, downloads of requested files will take place once only.

The resulting system's functionality, performance, design and usability was evaluated by the group of respondents from Free State population. These respondents were foreigners and non-Sesotho speakers that do not know Sesotho language.

Some aspects that needs to be improved based on respondents perception and observations conducted by the researcher

1. Automatic speech recognition,

   Accuracy of Google ASR was poor, it could not recognize the continuous speech. The user has to speak loud with a close talk towards the mic. The machine lacks in differentiating between different dialects, or different pronunciation. The Google ASR also failed to extract the speech from the background noise.

2. Addition of words in a machine dictionary database and in corpus(migration from specialized parallel corpus to general parallel corpus)

   Most of the participants had a problem with the machine when failing to translate their desired input. This problem was caused by the fact that the corpus was designed to specialize only on basic Sesotho words and phrases that need to be known by the users, not the whole language. The major challenge in migrating to general parallel corpus consisting of millions of words was the lack of cellphone's processing speed, limited memory and storage capabilities. To solve these problems, the client server approach can be adopted whereby the dictionary database together with the corpus will be removed from the app and be located on web server. This is an effective approach in terms of improving system functionality; however, it cannot be viable due to cost ineffectiveness, since the user has to always access the internet to operate the system.

3. Allow translation to other languages

This can further be implemented by extending the target corpus to accommodate other languages. The functionality level will increase, because single application solves more than one problem. However, the major challenge is the level of complexity which will increase not only on a technical basis but also on graphical user interface. The user before performing translation must first go through the process of using some sort of combo box to select the source and the target language.

4. Bidirectional translation between two languages

Another further job stipulated by the respondents is vice-versa translation between two languages; the application should translate back from Sesotho to English. This approach will help Sesotho speakers who would like to learn English languages. As a solution, Sesotho dictionary database and parallel corpus where the Sesotho language is a source language, must be developed. One main challenge is the implementation of ASR for Sesotho language. This technology will prompt the user to speak Sesotho words or phrases to the machine, recognize and display them on a screen for processing. Lots of training data for speech processing is required; resource scarce languages like Sesotho might not be viable for this technology.

5. Pronunciation learning technology

UmobiTalk similar to other web language learning translators such as Android speech based Google translate, allow the user to speak the words they wish to translate and generate the output normally through a text or speech. That is what is expected from any translator. However, providing the user with a foreign text, will not help a user to read and learn the proper pronunciation of translated text. For example a phrase such as “setjhaba sa Qwaqwa” will be difficult to be read and properly articulated by a non-Sesotho speakers such as Indians or Chinese speaking people, thus a need to learn pronunciation is vivid. As a further work

UmobiTalk will integrate Sesotho pronunciation training, to enforce a proper pronunciation of Sesotho words for effective communication principles. This technology will also help with the learning of complex phonological aspects of Sesotho consonants such as “kg, tl, tlh, qwa, qha, ts,tsh,ph” and many more.

6. Lemmatization or stemming of inputted words

The morphological analyzer of the application do not apply the lemmatization or stemming of words. This process is important because inputted verbs such as works, worked, working, has to be grouped to a single lemma “work”.

Based on analyzed results from research methodology, it is evident that the system can bridge the gap caused by language barrier between Sesotho speaking and non-Sesotho speaking people in Free State. The system’s design and usability proved that they can cater for disadvantaged individuals who are technologically illiterate and disable such as those who are blind, deaf and those suffering from dyslexia.

To further enhance the effectiveness of the machine to quickly processing data, system testing and evaluation was conducted by the researcher. Since the application is equipped with many search algorithms inside the MT operation, we decided to evaluate and optimize these search algorithms so to enhance the processing speed. The Big O notation was used as a tool to assess two types of search algorithms which are sequential and binary search. For evaluation, we used Big O’s time complexity cases which are best case, worst case and average case. The criteria for each time complexity case was set by the researcher. Binary search algorithm was proved as the quickest search algorithm. However the searched data has to be sorted to support the effectiveness of binary search, Big O notation’s time complexity cases were used again to evaluate different types of sorting algorithms, and the decision was made on quick sort. Both binary search and quick sort algorithms were utilized for all search algorithms in MT operation.

The translation accuracy of the machine was evaluated by the Sesotho language expert using word error rate (WER) algorithm. This algorithm accepts two arguments which are hypothesis document (translation made by the machine) and translation reference which is known to be correct by the language expert. It compares the two arguments and calculate the word error rate. The higher the word error rate the lower the translation accuracy. More than 200 English sentences were formed using words from machine dictionary and were translated. According to the statistics made 72.2 % translation accuracy was reached.

7.2 Language Implications during Corpus Annotation

The words with unresolved ambiguity such as a word “like” were difficult to be annotated; “like” can act as a verb and as a comparison tool in a same context in which it appears; therefore, the meaning of the whole context results into ambiguity. For example the sentence “men like others” has two semantics result in to an arbitrary decision by the compiler.

1. Like, means men like others in terms of attraction.
2. Like, means men like others in terms of comparison.

Some words have ambiguity that can easily be resolved based on a context in which they appear, such as “close”, “fish”, “book”, and others.

Some implications that were noticed when trying to annotate both languages were Sesotho inflection of words. Sesotho nouns are inflected based on the context they are locating on. For example a noun “tree” is translated as “sefate”; however, a clause “on a tree” is translated as “sefateng”, whereby the locative suffix “ng” has been fused to a morpheme. A noun “love” in Sesotho language is “lerato”, but when is joined to a sentence such as “I love you”, it changes to “rata” same as a word “hloya” (hate) which changes to “hloile”.

The corpus that took into consideration the inflection of Sesotho words was developed, however, it couldn’t always keep up with each morpheme various inflections and derivations. Thus affect the translation accuracy.

The verb “is” and “are” are known as subject or relative concords in Sesotho language. These concords are not fixed, for example verb “is” can be represented by the concord “o” or “e” depending on the type of a noun preceding the concord, for example any type of a noun whether proper noun or pronoun referencing a person is followed by subject concord “o”, a noun referencing an object or animal is followed by subject concord “e”. This resulted into a problem because before selecting a correct concord, the type of the preceding subject must first be studied. The complex part was that unknown words (words not in a corpus) are default classified as proper nouns such as “Jacob” or “Shoprite” by the machine; so it is impossible to determine the reference of the proper noun inputted by the user, whether it reference human being or an object, so to select the correct concord. The solution developed was either to separate possible subject concords with a forward slash key such as “o/e/ke”, or to use a default concord that will stand all form of proper nouns. None of the solutions are effective because having a sentence with forward slashes will definitely confuse the language learner and moreover cannot be read by the Sesotho speech synthesizer, and using one type of subject concord will result in to semantically incorrect sentences.

Other dynamics of concords that the machine had to contest with, are concords used to stand the pronouns.

Table 7.1 Concords based on absolute pronouns

	Singular pronoun	**Concord**	**Plural pronoun**	**Concord**
First Person	I(nna)	Ke	We(rona)	Re
Second Person	You(wena)	O	You(lona)	Le
Third Person	He/She(yena)	O	They(bona)	Ba

The selection of correct concords had a great impact in machine processing speed, resources and capabilities.

The questioning phrases used in two languages posed a challenge, in English language questioning words normally start a sentence and in Sesotho language they end a sentence. For example a phrase “where are you” is translated as “wena o hokae”, whereby “hokae” is a translation of “where”.

To make the translation possible, the questioning words had to be swapped in a target file to match the source words.

Moreover, the questioning word “hokae” can vanish in a target language, when its translation in a source language is not used to start the sentence, for example a sentence “this is where he stays”, is translated as “ke moo a dulang teng”. The same as the questioning word “who”, example “this is who I am”, which is translated as “ke moo ke leng ka teng”.

The Sesotho pronouns are failing gender agreement qualification. Pronoun he/she is translated as “yena”, making it difficult to differentiate the subject in terms of their gender.

References

1. Alumae,T, Kaljurand,K., 2012. Open and Extantadable Speech Recognition Application Architecture for Mobile. Environments.. s.l., Estonian Ministry of Education and Research targetifinanced research theme.

2. Ambati, V. & Vogel, S., 2010. *Can Crowds build parallel corpus for machine translation systems?* Los Angel California, Association for Computational Linguistics, pp. 62-65.

3. Anodo, T., 2013. *Open Source Spelling Checker for Kimiiru language (Doctoral dissertation, University of Nairobi).* s.l., s.n.

4. Anthoney, L., 2005. *AntConc: Design and Development of a Freeware Corpus Analysis Toolkit for the Technical Writing Classroom.* s.l., IEEE.

5. Anusuya, M & Katti,S., 2009. Speech Recognition by Machine. *International Journal of Computer Science and Information Security,* 6(3), pp. 181-205.

6. Armstrong, S. et al., 1999. *Natural language processing using very large corpora,* s.l.: Kluwer Academic Publishers.

7. Arvaniti, A. & Baltazani, M., 2005. International analysis and prosodic annotation of Greek spoken corpora. Prosodic typology: The phonology of intonation and phrasing, pp.84-117.

8. Bamberger, M., 2012. Introduction to mixed methods in impact evaluation. Impact Evaluation Notes 3, 3, pp.1-38.

9. Banea, C., Mihalcea, R., Wiebe, J. & Hassan, S., 2008, October. Multilingual subjectivity analysis using machine translation. In proceedings of the Conference on Empirical Methods in Natural Language Processing (pp. 127-135). Association for Computational Linguistics

10. Bentley, J. & Sedgewick, R., 1997, January. Fast algorithms for sorting and searching strings. In *SODA* (Vol. 97, pp. 360-369).

11. Besacier, L., Barnard, E., Karpov, A. & Schultz, T., 2013. Automatic speech recognition for under resourced languages: A survey. *Speech communication,* Volume 56, pp. 85-100.

12. Bonaventura, P., Howarth, P. and Menzel, W., 2000, August. Phonetic annotation of a non-native speech corpus. In Proceedings International Workshop on Integrating Speech Technology in the (Language) Learning and Assistive Interface, Instil (pp 10-17).

13. Bosco, C., Patti, V. & Bolioli, A., 2015. *Developing Corpora for Sentiment Analysis:The Case of Irony and Senti–TUT (Extended Abstract).* s.l., s.n., pp. 4158-4162.

14. Buede, D. & Miller, W., 2016. The engineering design of systems: models and methods. John Wiley & Sons.

15. Caines, A., Bentz, C., Greham, C., Polzehl, T. & Buttery, P., 2015. *Crowdsourcing a multi-lingual speech corpus: recording,transcription and natural language processing,* s.l.: s.n.

16. Casacuberta, F., Garcia, R., Llisterri, J., Nadeu, C., Pardo, J.M and Rubio, A., 1991, September. Development of Speech Corpora for speech research (ALBAYZIN). In Workshop on International Cooperation and Standardization of Speech Databases and Speech I/O Assessment Methods, Chiavari, Italy(pp.26-28)

17. Chapman, A., 2005. *Principles and Methods of Data cleaning- Primary species and species-occurance data,* Copenhagen: Global Biodiversity Information Facility.

18. Chen, L., Rose, R., Qiao, Y., Kimbara, I., Parrill, I., Welji, H., Han, T., Tu, J., Huang, Z., Harper, M., Quek, F., Xiong, Y., McNeill, R., Tuttle, R. & Huang, T., 2005. *Vace multimodal meeting corpus,* s.l.: s.n.

19. Collins, N., 2015. *The UbuWeb electronic music corpus on MIR investigation of a historical corpus,* Durham: Cabridge University Press.

20. Danziger, P., 2015. Big O Notation.

21. Demuth, K., 1992. Accessing Functional Categories in Sesotho: Interactions at the Morpho Syntax Interface. *The acquisition of Verb Placement:,* pp. 83-107.

22. Deng,L. & Xuedong,H., 2004. Challenges in adopting speech recognition. *Communication of the ACM,* 47(1), pp. 69-75.

23. Devlin, J., Zbib, R., Huang, Z., Lamar,T., Schwartz, R.M. and Makhoul, J.,2014, June. Fast and Robust Neural Network Joint models for statistical machine translation. In ACL(1) (pp. 1370-1380)

24. De Vries, N.J., Davel, M.H., Badernhorst, J., Basson, W.D., De Wet, F., Barnard, E. and De Waal, A., 2014. A smartphone based ASR data collection tool for under-resourced languages. Speech communication, 56, pp. 119-131.

25. Dickinsin, M. & Lee, C., 2009. Modifying corpus annotation to support the analysis of learner language. *CALICO journal,* 26(3), pp. 545-561.

26. Ebert, C., 2014. Speech- Recognition For Voice Based Machine Translation. *Software Technology,* pp. 21-34.

27. Faab, G., 2010. A morphosyntactic description of Northern Sotho as a basis for an automated translation from Northern Sotho into English.

28. Farr, F. & Murray, L., 2016. *The Routledge Handbook of Language Learning and Technology.* New York: Routledge.

29. Feld, M., Montazi,S., Freigang,F., Klakow,D. & Muller,C., 2012. *Mobile texting: Can Post ASR Correction Solve the Issues? An experimental Study on Gain vs Costs.* Austin,TX,USA, CHI 2012.

30. Ferragne, E., Flavier, S. & Fressard, C., 2013. *Rocme! Software for the recording and management of speech corpora,* s.l.: s.n.

31. Gebre, L., Maharaj, P. & Pillay, K., 2011. The experiences of immigrants in South Africa: a case study of Ethiopians in Durban, South Africa. 22(1), pp. 23-35.

32. Graen, J., Batinic, D. & Volk, M., 2014. *Cleaning the Europarl corpus for linguistic applications,* s.l.: s.n.

33. Green, S., Heer, J. and Manning, CD., 2013, April. The efficacy of human post-editing for language translation. In proceeding of the SIGCHI Conference on human factors in Computing Systems (pp. 439-448). ACM.

34. Gries, S. & Berez, A., 2015. *Linguistic annotation in or for corpus linguistics,* s.l.: s.n.

35. Hammo, B., Yagi, S., Ismail, O. & AbuShariah, M., 2015. Exploring and exploiting a historical corpus for Arabic. pp. 1-23.

36. Hardie, E., 2012. CQPweb- combining power, flexibility and usability in a corpus analysis tool. *International Journal of Corpus Linguistics,* 17(3), pp. 380-409.

37. Herbert, B., Szarvas, G. & Gurevych, I., 2011, April. Combining query translation techniques to improve cross-language information retrieval. In European Conference on Information Retrieval (pp. 712-715). Springer Berlin Heidelberg

38. Hernandez, M. & Stolfo, S., 1998. Real world data is dirty: Data cleansing and the merge/purge problem. *Data Mining and Knowledge Discovery,* Volume 2, pp. 9-37.

39. Hussein, K., 2015. The potentialities of Corpus based technique for analyzing literature. *Journal of Literature, Language and Culture (COE & RJ-JLLC),* 1(1), pp. 24-38.

40. Hutchis, J., 2001.From first conception to first demonstration, 1947-1954. A chronology. Machine translation 12, no.3 (1997): 195-252.

41. Hyman, P., 2014. Speech-To-Speech Translators Stutter, but researchers see mellifluous future. *Communication of the ACM,* 57(4), pp. 16-19.

42. Idemudia, E., Williams, J. & Wyatt, G., 2013. Migration challenges among Zimbabwean refugees before, during and post arrival in South Africa. *Journal of injury and violence,* 5(1), pp. 17-27.

43. Ishita, E., Oard, D. & Fleischman, K., 2015. Learning curves for automating content analysis: How much human annotation is needed? *In Advanced Applied Informatics (IIAI-AAI), 2015 IIAI 4th international congress,* pp. 171-176.

44. Jackson, M., 2005. Automatic Speech recognition: Human Computer interaction for Kinyarwanda Language.

45. Jackson, H., 2014. *Words and their meaning,* s.l.: s.n.

46. Jakubicek, M., Kilgarriff, A., Kovar, V., Rychly, P. & Suchomel, V., 2013. The Ten Ten corpus family.

47. Jivani, A., 2011. Comparative study of stemming algorithm. *International Journal of Computer technology application,* 2(6), pp. 1930-1938.

48. Johnson, M., 2008. Unsupervised word segmentation for Sesotho using adaptor grammars. *Proceedings of the tenth meeting of ACL Special.*

49. Johnson, M., Demuth, K. & Canon, S., 1999. Tagging and Glossing Sesotho.

50. Kadri, Y. & Nie, J.Y., 2006. Effective stemming for Arabic information retrieval. In the challenge of Arabic for NLP/MT, Intl Conf. at the BCS (pp. 68-74)

51. Kalitanyi, V. & Visser, K., 2010. African immigrants in South Africa: job takers or job creators? *South African Journal of Economic and management sciences,* 13(4).

52. Kanerva, J., Luotalahti, J., Laippala, V. & Ginter, F., 2014. *Syntactic N-Gram Collection from a Large-Scale Corpus of Internet Finnish,* s.l.: IOS Press.

53. Kennedy, G., 2014. *Introduction to corpus linguistics.* New York: Routledge.

54. Khreisat, L., 2006. *Arabic Text Classification Using N-Gram Frequency Statistics A Comparative Study,* s.l.: s.n.

55. Kichuk, D., 2015. Loose,falling characters and sentences. *The persistence of the OCR problem in degital repository E-books,* 15(1), pp. 59-91.

56. Kilgarriff, A. & Grefenstette, G., 2003. Introduction to the Special Issue on the Web as Corpus. *Computational Linguistics,* 29(3), pp. 334-347.

57. Kinoshita, K., Delcroix, M., Yoshioka, T., Nakatani, T., Sehr, A., Kellermann, W. & Maas, R., 2013, October. The REVERB challenge: A common evaluation framework for dereverberation and recognition of reverberant speech. In Applications of Signal Processing to Audio and Acoustics(WASPAA),2013 IEEE Workshop on (pp.1-4).IEEE.

58. Klinkmuller, C., Weber,I., Mendling, J., Leopold,H. & Ludwig, A., 2013. Increasing recall of process model matching by improved activity label matching. In Business Process Management (pp. 211-218). Springer Berlin Heidelberg.

59. Ko, T. & Mak, B., 2013. Eigentrigraphemes for under-resourced languages. *Speech Communication.*

60. Korenius, T., Laurikkala, J., Jarvelin, K. & Juhola, M., 2004, November. Stemming and lemmatizing in the clustering of finnish text documents. In proceedings of the thirteenth ACM international conference on information and knowledge management (pp. 625-633). ACM.

61. Kumar, A. et al., 2011. Rethinking Speech Recognition on Mobile Device, California, USA: s.n.

62. Latham, B., 2007. Quantitative Research Method. Sampling: What is it? 8 March.

63. Lee, Y., 2004, May. Morphological analysis for statistical machine translation. In Proceedings of HLT-NAACL 2004: Short Papers (pp. 57-60). Association for Computational Linguistics.

64. Leech, G. & Smith, N., 2005. Extending the possibilities of corpus based research on English in the twentieth century: A prequel to LOB and FLOB. ICAME Journal, 29, pp.83-98.

65. Lin, D., Murakami, Y., Ishida, T., Murakame, Y. & Tanaka, M., 2010. *Composing Human and Machine Translation Service: Language Grid for Improving Localization Processes.* s.l., Navix Co.,Ltd.

66. Madera, J., Neal, J. & Dawsonm M., 2011. A STRATEGY FOR DIVERSITY TRAINING: FOCUSING ON EMPATHY IN THE WORKPLACE. *Journal of hospitality and tourism research,* 35(4), pp. 469-487.

67. Maekawa, K., Yamazaki, M., Ogiso, T., Maruyama, T., Ogura, H., Kashino, W., Koiso, H., Yamaguchi, M. & Tanaka, M., 2014. Balanced Corpus of contemporary written Japanese. *Language Resources and Evaluation,* 48(2), pp. 345-371.

68. Martinez, A., 2011. Part of speech tagging. *Wiley Interdisciplinary reviews: computational statistics,* 4(1), pp. 107-113.

69. McDonel, S., Min, K. & Connor, A., 2014. *Autonomous requirements specification processing using Natural Language Processing,* s.l.: s.n.

70. McEnery, T. & Wilson, A., 2001. *Corpus linguistics.* 2nd ed. s.l.:s.n.

71. Montemagni, S., Barsotti, F., Battista, M., Calzolari, N., Corazzari, O., Lenci, A., Zampolli, A., Fancuilli, F., Massetani, M., Raffaelli, R. & Basili, R., 2003. Building the Italian syntactic-semantic treebank. Treebanks, pp.189-210.

72. Muijs, D., 2010. Doing quantitative research in education with SPSS. Sage.

73. Mustafa, M., Salim, S. & Rahman, F., 2016. A two-stage adaptation towards automatic speech recognition system for Malay-speaking children. *International Journal of Computer, Electrical, Automation, Control and Information Engineering,* 10(3).

74. Najork, M. & Heydon, A., 2002. High performance web crawling. In Handbook of massive data sets (pp. 25-45). Springer US.

75. Nakazuwa, T., Kuwohashi, S., Kobayashi, H., Ishikawa, H. & Sassano, M., 2015. *3-step parallel corpus cleaning using monolingual crowd workers,* s.l.: s.n.

76. Neunerdt, M., Reyer, M. & Mathar, R., 2015. *Enhanced webpage cleaning for constructing social media corpora,* s.l.: s.n.

77. Nino, A., 2009.Machine translation in foreign language learning: language learners 'and tutors' perceptions of its advantages and disadvantages. Recall, 21(02), pp.241-258.

78. Patton, M. & Cochran, M., 2002. A guide to using qualitative research methodology. Retrieved with permission by Nouria Brikci-Research Officer, MSF UK (February 2007) http://evaluation. Msf. At/fileadmin/evaluation/files/documents/resources_MSF/MSF_Qualitative_ Methods.pdf.

79. Paulussen, H., Macken, L., Vandeweghe, V. & Desmet, P., 2013. *Dutch Parallel Corpus: A Balanced Parallel Corpus for Dutch-English and Dutch-French,* s.l.: s.n.

80. Pollitt M, 2014. History Matters. *History Today,* pp. 3-4.

81. Popovic, M., & Ney, H., 2007, June. Word error rates: decomposition over pos classes and applications for error analysis. In Proceedings of the Second Workshop on Statistical Machine Translation (pp. 48-55). Association for Computational Linguistics.

82. Probyn, M., 2006. Language and Learning Science in South Africa. *Language and education,* 20(5).

83. Rayson, P., 2015. *Handbook of English corpus linguistics.* s.l.:s.n.

84. Reddy, R & Mahender, E., 2013. Speech to Text Conversion Using Android Platform. International Journal of Engineering Research and Applications (IJERA), 3(1), pp. 253-258.

85. Rose, Y. and Demuth, K., 2006. Vowel epenthesis in loanword adaption: Representational and phonetic considerations. Lingua, 116(7). Pp. 1112-1139.

86. Rudwick,S. & Parmegiani,A., 2013. Divided loyalities: Zulu vis-vis English at the University of KwaZulu-Natal. Language Matters, 44(3), pp.89-107.

87. Sabou, M., Bontcheva, K., Derczynski, L. & Scharl, A., 2014. Corpus Annotation through Crowdsourcing: Towards Best Practice Guidelines. *LREC,* pp. 859-866.

88. Safari, R. and Nouza, J., 2015, June. Methods for rapid development of ASR system for Russia. In Electronics, Control, Measurement, Signals and their Application to Mechatronics (ECMSM), 2015 IEEE International Workshop of (pp. 1-6). IEEE.

89. Sibanda, O., 2010. Social tiles and the Dynamics of integration in the City of Johannesburg among Zimbabwe Migrants.

90. Sinclair, J., 2004. Trust the text: Language, corpus and discourse. Routledge

91. Singhal, S. & Jena, M., 2013. A study on Weka Tool for data processing, classification and clustering. *International Journal of innovative technology and exploring engineers,* 2(6), pp. 250-253.

92. Statistics South Africa, 2011. Population Census.

93. Sundermeyer, M.et al., 2014. Translation modeling with bidirectional recurrent neural networks. s.l., s.n., pp. 14-25.

94. Terrel, S.R., 2012. Mixed – methods research methodologies. The qualitative report, 17(1), pp. 254-280

95. Tian, L., Wong, D., Chao, L., Quaresma, P., Oliveira, F., Lu, Y., Li, S., Wang, Y. & Wang, L., 2014. *UM-Corpus: A large English-Chinese Parallel Corpus for Statistical Machine Translation,* s.l.: s.n.

96. Tongco, M.D.C., 2007. Purposive sampling as a tool for informant selection. Ethnobotany Research and Applications, 5. Pp. 147-158.

97. Tunney, C., Cooney, P., Coyle, D. and O'reilly, G., 2017. Comparing young people's experience of technology-delivered v. face-to-face mindfulness and relaxation: two-armed qualitative focus group study. The British Journal of Psychiatry, 210(4), pp.284-289.

98. Vaughan, E. & O'Keefe, A., 2015. Corpus Analysis. *The International Encyclopedia of Language and Social Interaction.*

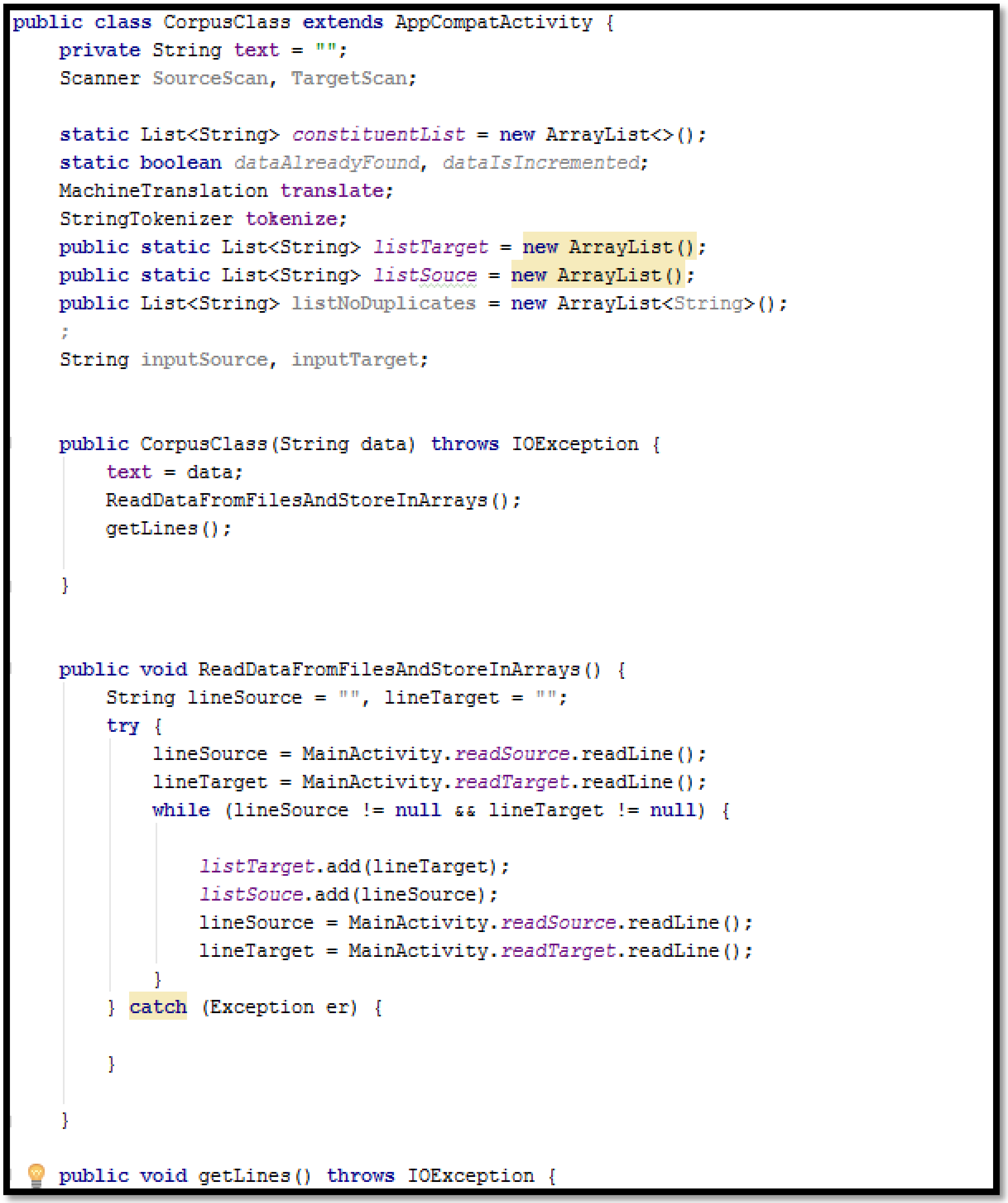
99. Vinogradov, I., 2016. Linguistic Corpora of understudied languages; do they make sense? Kanina, 40(1), pp.116-130.

100. Vinyals, O. and Friedland, G., 2008, August. Towards semantic analysis of conversations: A system for the live identification of speakers in meetings. In semantic computing, 2008 IEEE International Conference on (pp. 426-431). IEEE.

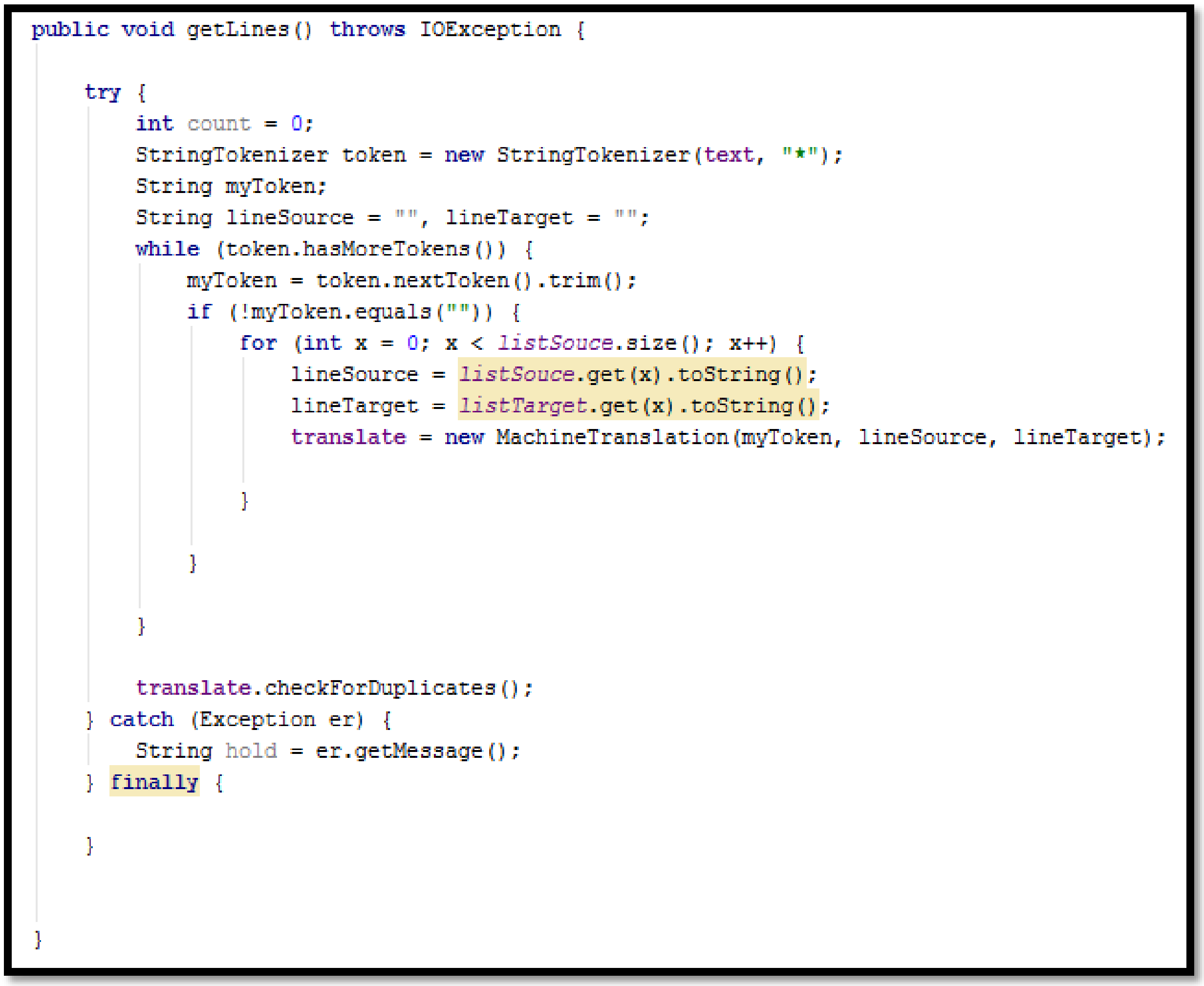
101. Wu, S. & Witten, L., 2016. Transcending Concordance: Augmenting Academic Text for L2 Writing. *International Journal of Computer-Assisted Language Learning and Teaching (IJCALLT,* 6(2).

102. Yang, L., Zhang, D. & Tang Y., 2014. The realization of food corpus based on database technology. *Journal of Simulation,* 2(6), pp. 327-330.

List of Appendices

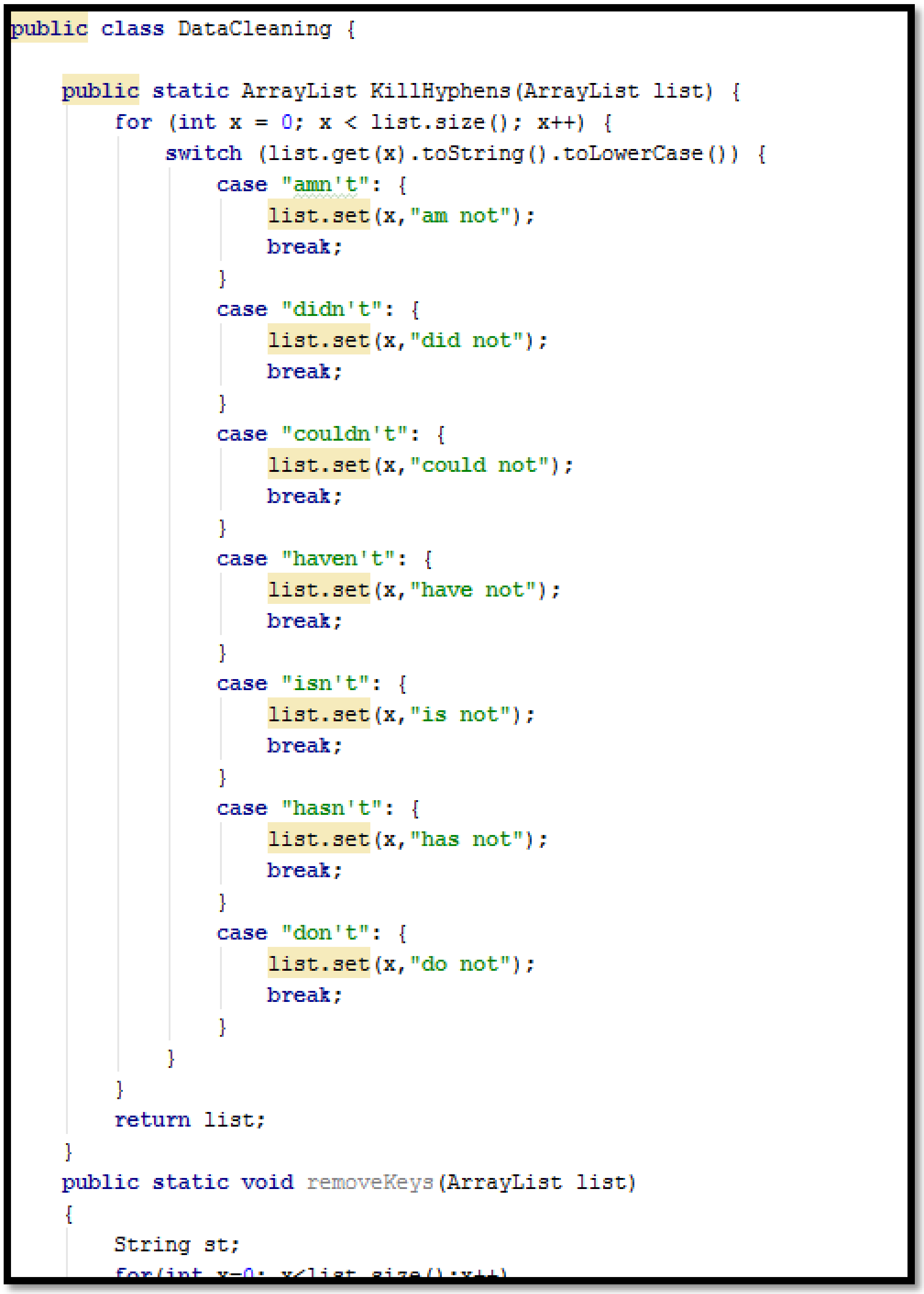


```java
public class CorpusClass extends AppCompatActivity {
    private String text = "";
    Scanner SourceScan, TargetScan;

    static List<String> constituentList = new ArrayList<>();
    static boolean dataAlreadyFound, dataIsIncremented;
    MachineTranslation translate;
    StringTokenizer tokenize;
    public static List<String> listTarget = new ArrayList();
    public static List<String> listSouce = new ArrayList();
    public List<String> listNoDuplicates = new ArrayList<String>();
    ;
    String inputSource, inputTarget;


    public CorpusClass(String data) throws IOException {
        text = data;
        ReadDataFromFilesAndStoreInArrays();
        getLines();

    }


    public void ReadDataFromFilesAndStoreInArrays() {
        String lineSource = "", lineTarget = "";
        try {
            lineSource = MainActivity.readSource.readLine();
            lineTarget = MainActivity.readTarget.readLine();
            while (lineSource != null && lineTarget != null) {

                listTarget.add(lineTarget);
                listSouce.add(lineSource);
                lineSource = MainActivity.readSource.readLine();
                lineTarget = MainActivity.readTarget.readLine();
            }
        } catch (Exception er) {

        }

    }

    public void getLines() throws IOException {
```

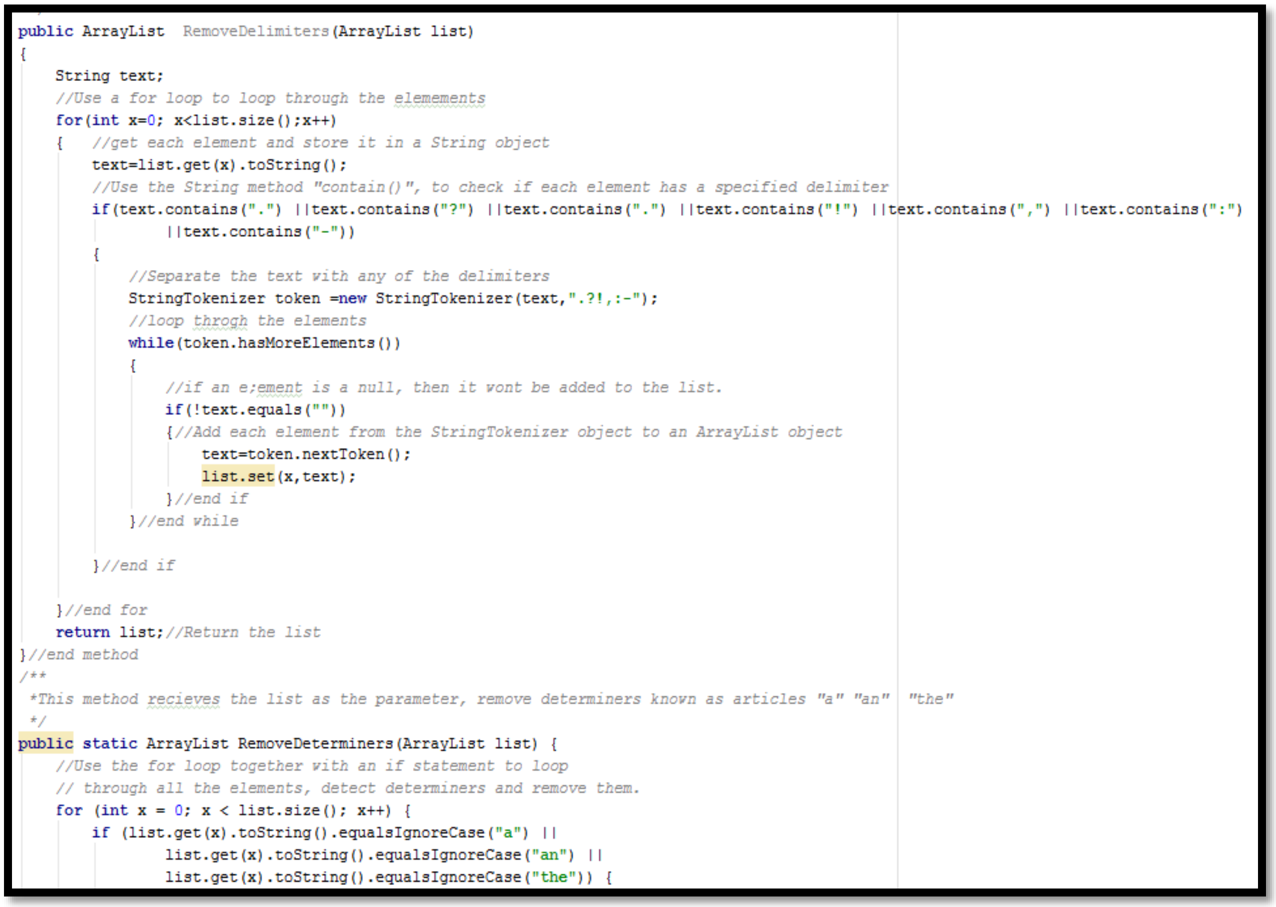


```
public void getLines() throws IOException {

    try {
        int count = 0;
        StringTokenizer token = new StringTokenizer(text, "*");
        String myToken;
        String lineSource = "", lineTarget = "";
        while (token.hasMoreTokens()) {
            myToken = token.nextToken().trim();
            if (!myToken.equals("")) {
                for (int x = 0; x < listSouce.size(); x++) {
                    lineSource = listSouce.get(x).toString();
                    lineTarget = listTarget.get(x).toString();
                    translate = new MachineTranslation(myToken, lineSource, lineTarget);

                }

            }

        }

        translate.checkForDuplicates();
    } catch (Exception er) {
        String hold = er.getMessage();
    } finally {

    }


}
```

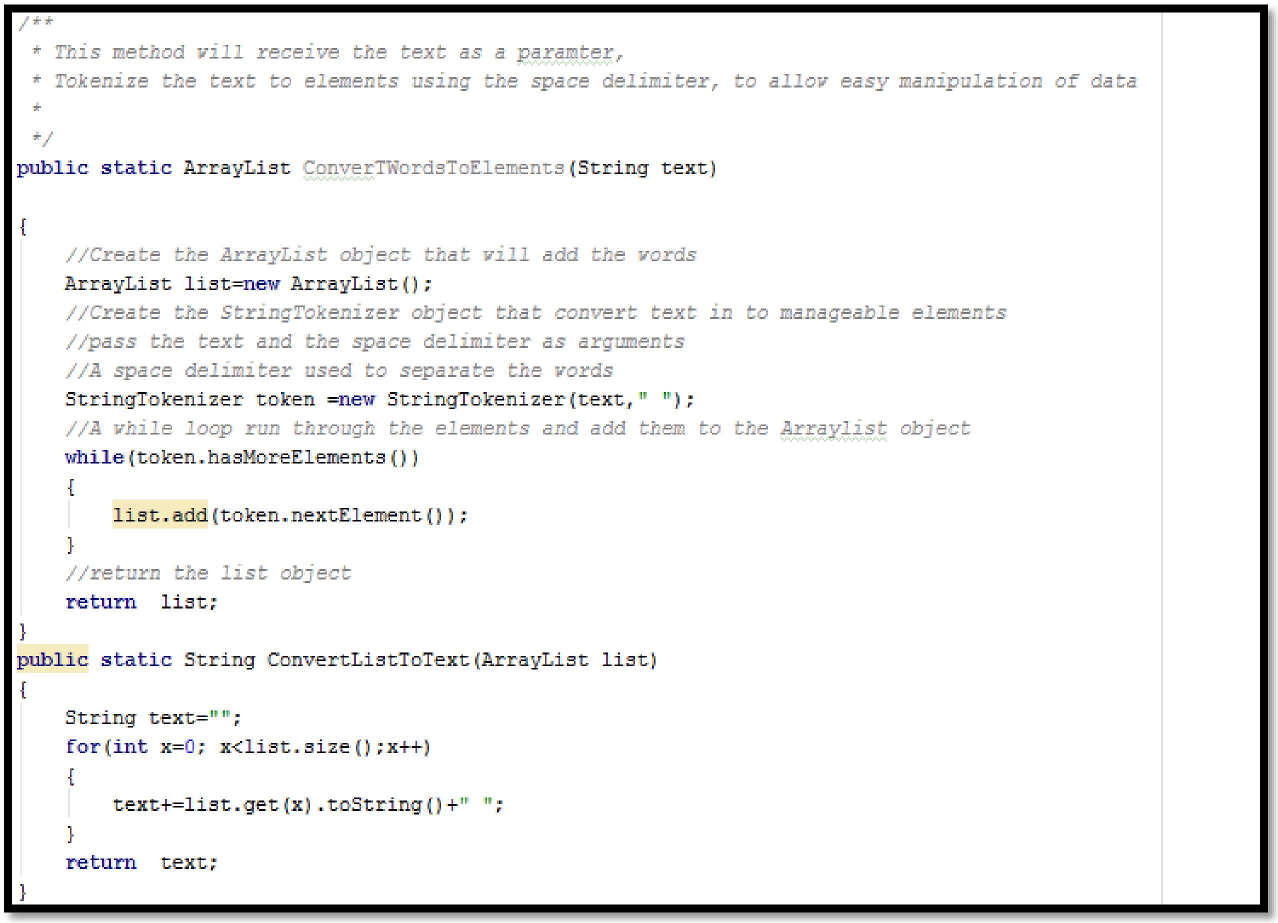
Appendix 1: Corpus Class Code



```
public class DataCleaning {

    public static ArrayList KillHyphens(ArrayList list) {
        for (int x = 0; x < list.size(); x++) {
            switch (list.get(x).toString().toLowerCase()) {
                case "amn't": {
                    list.set(x,"am not");
                    break;
                }
                case "didn't": {
                    list.set(x,"did not");
                    break;
                }
                case "couldn't": {
                    list.set(x,"could not");
                    break;
                }
                case "haven't": {
                    list.set(x,"have not");
                    break;
                }
                case "isn't": {
                    list.set(x,"is not");
                    break;
                }
                case "hasn't": {
                    list.set(x,"has not");
                    break;
                }
                case "don't": {
                    list.set(x,"do not");
                    break;
                }
            }
        }
        return list;
    }
    public static void removeKeys(ArrayList list)
    {
        String st;
        for(int x=0; x<list.size();x++)
```

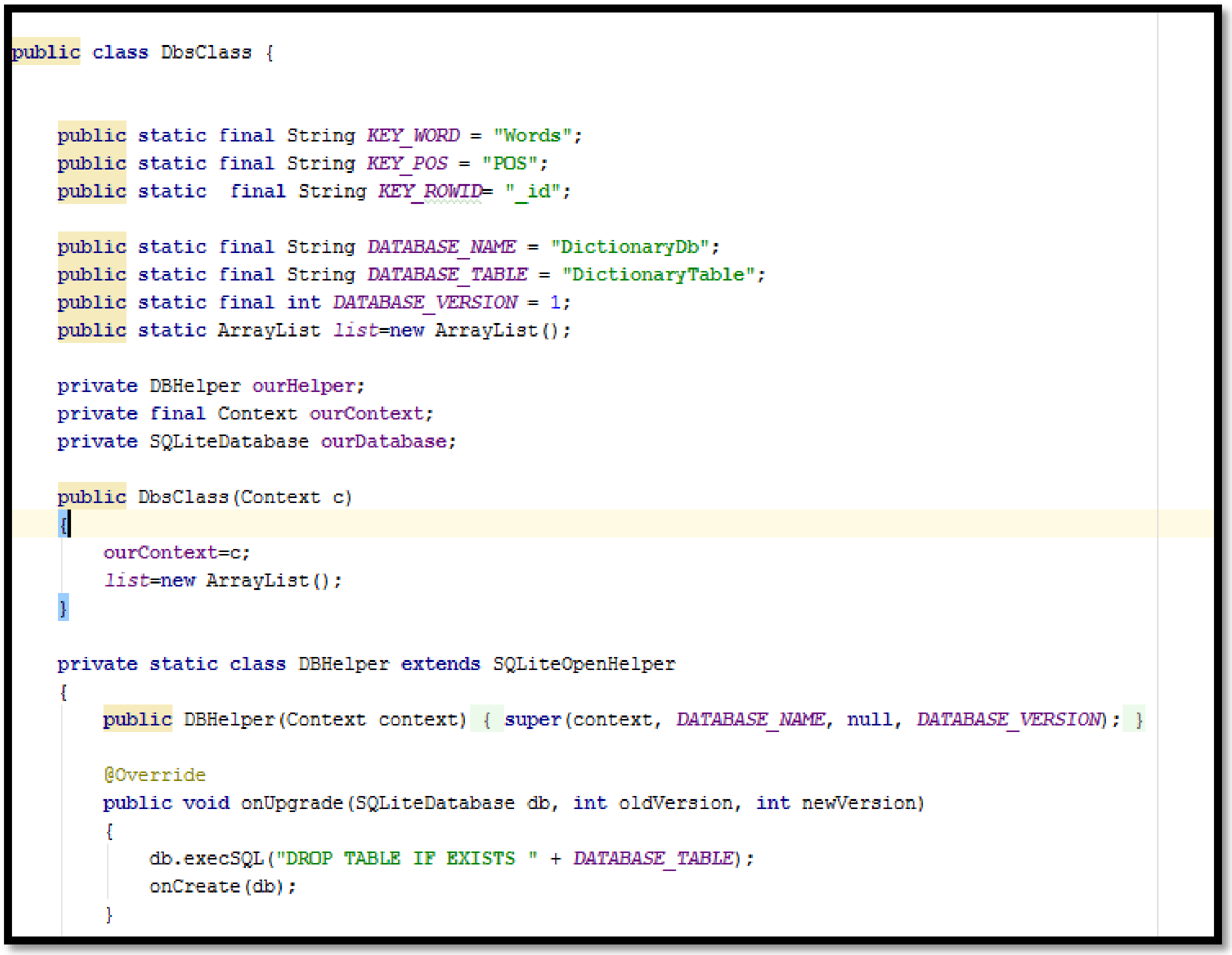


```
public ArrayList  RemoveDelimiters(ArrayList list)
{
    String text;
    //Use a for loop to loop through the elemements
    for(int x=0; x<list.size();x++)
    {   //get each element and store it in a String object
        text=list.get(x).toString();
        //Use the String method "contain()", to check if each element has a specified delimiter
        if(text.contains(".") ||text.contains("?") ||text.contains(".") ||text.contains("!") ||text.contains(",") ||text.contains(":")
                ||text.contains("-"))
        {
            //Separate the text with any of the delimiters
            StringTokenizer token =new StringTokenizer(text,".?!,:-");
            //loop throgh the elements
            while(token.hasMoreElements())
            {
                //if an e;ement is a null, then it wont be added to the list.
                if(!text.equals(""))
                {//Add each element from the StringTokenizer object to an ArrayList object
                    text=token.nextToken();
                    list.set(x,text);
                }//end if
            }//end while

        }//end if

    }//end for
    return list;//Return the list
}//end method
/**
 *This method recieves the list as the parameter, remove determiners known as articles "a" "an"  "the"
 */
public static ArrayList RemoveDeterminers(ArrayList list) {
    //Use the for loop together with an if statement to loop
    // through all the elements, detect determiners and remove them.
    for (int x = 0; x < list.size(); x++) {
        if (list.get(x).toString().equalsIgnoreCase("a") ||
                list.get(x).toString().equalsIgnoreCase("an") ||
                list.get(x).toString().equalsIgnoreCase("the")) {
```

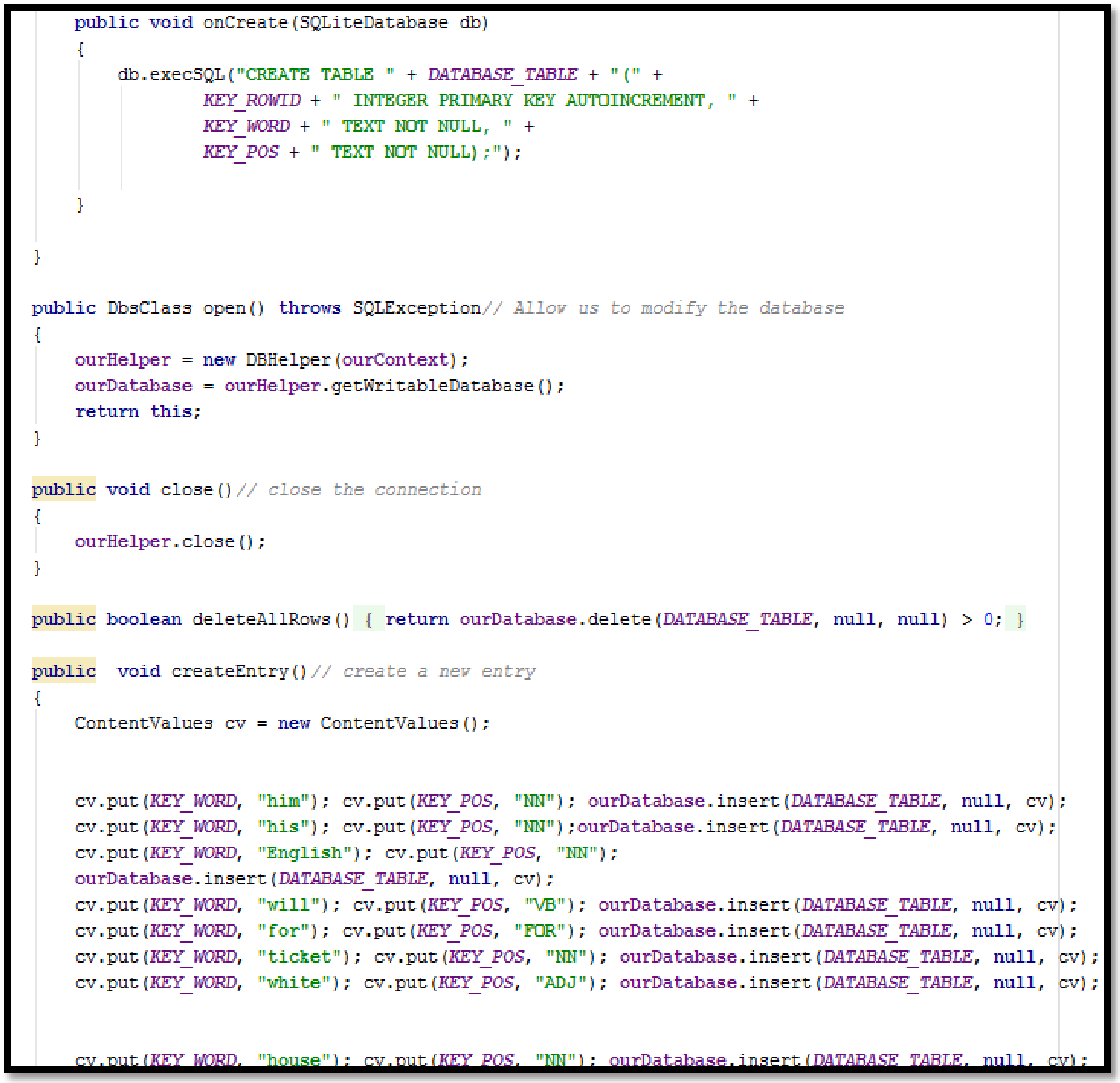


```
/**
 * This method will receive the text as a paramter,
 * Tokenize the text to elements using the space delimiter, to allow easy manipulation of data
 *
 */
public static ArrayList ConverTWordsToElements(String text)

{
    //Create the ArrayList object that will add the words
    ArrayList list=new ArrayList();
    //Create the StringTokenizer object that convert text in to manageable elements
    //pass the text and the space delimiter as arguments
    //A space delimiter used to separate the words
    StringTokenizer token =new StringTokenizer(text," ");
    //A while loop run through the elements and add them to the Arraylist object
    while(token.hasMoreElements())
    {
        list.add(token.nextElement());
    }
    //return the list object
    return  list;
}
public static String ConvertListToText(ArrayList list)
{
    String text="";
    for(int x=0; x<list.size();x++)
    {
        text+=list.get(x).toString()+" ";
    }
    return  text;
}
```

Appendix 2: Data Cleaning Operations Code

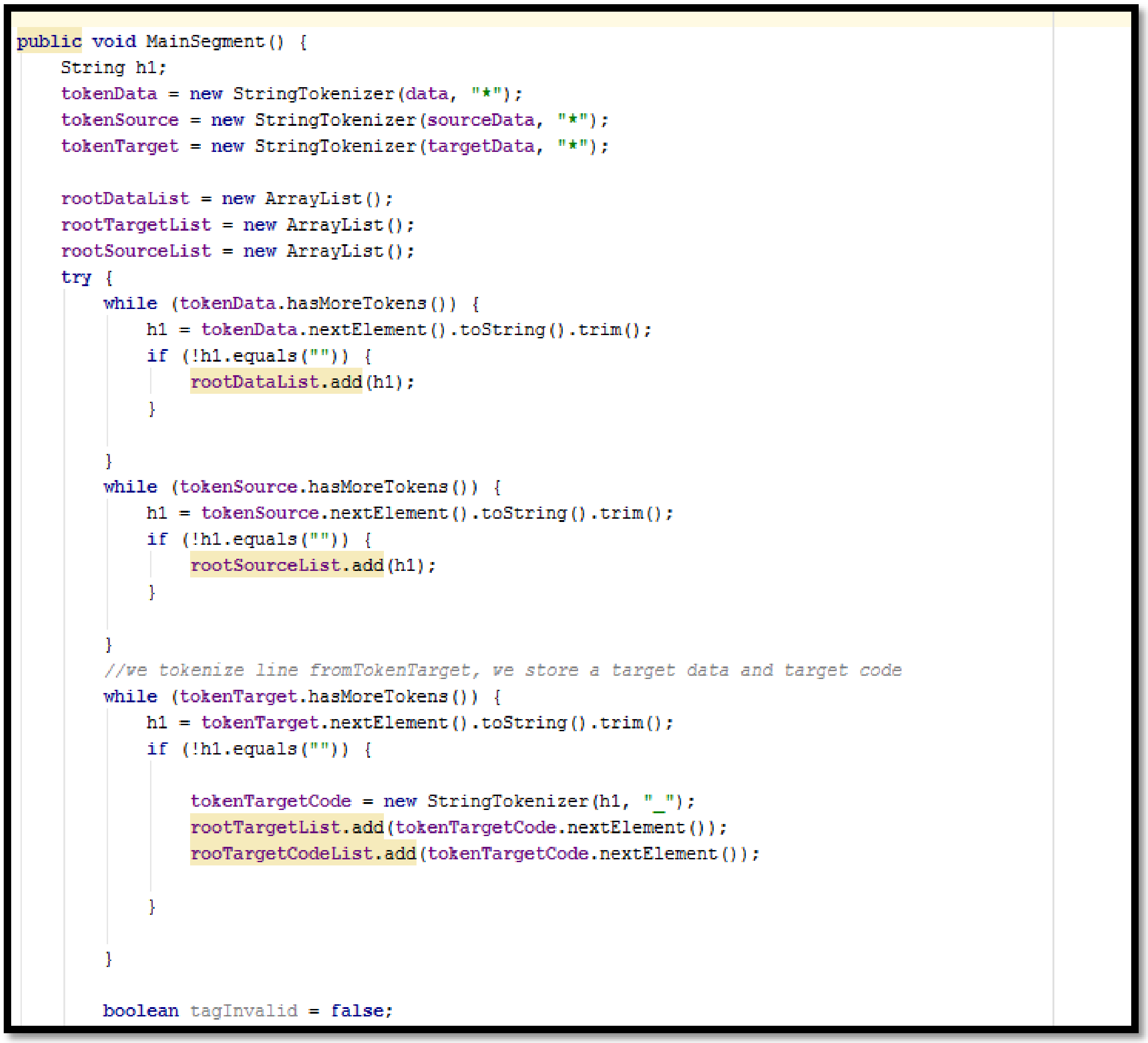


```
public class DbsClass {


    public static final String KEY_WORD = "Words";
    public static final String KEY_POS = "POS";
    public static  final String KEY_ROWID= "_id";

    public static final String DATABASE_NAME = "DictionaryDb";
    public static final String DATABASE_TABLE = "DictionaryTable";
    public static final int DATABASE_VERSION = 1;
    public static ArrayList list=new ArrayList();

    private DBHelper ourHelper;
    private final Context ourContext;
    private SQLiteDatabase ourDatabase;

    public DbsClass(Context c)
    {
        ourContext=c;
        list=new ArrayList();
    }

    private static class DBHelper extends SQLiteOpenHelper
    {
        public DBHelper(Context context) { super(context, DATABASE_NAME, null, DATABASE_VERSION); }

        @Override
        public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion)
        {
            db.execSQL("DROP TABLE IF EXISTS " + DATABASE_TABLE);
            onCreate(db);
        }
```

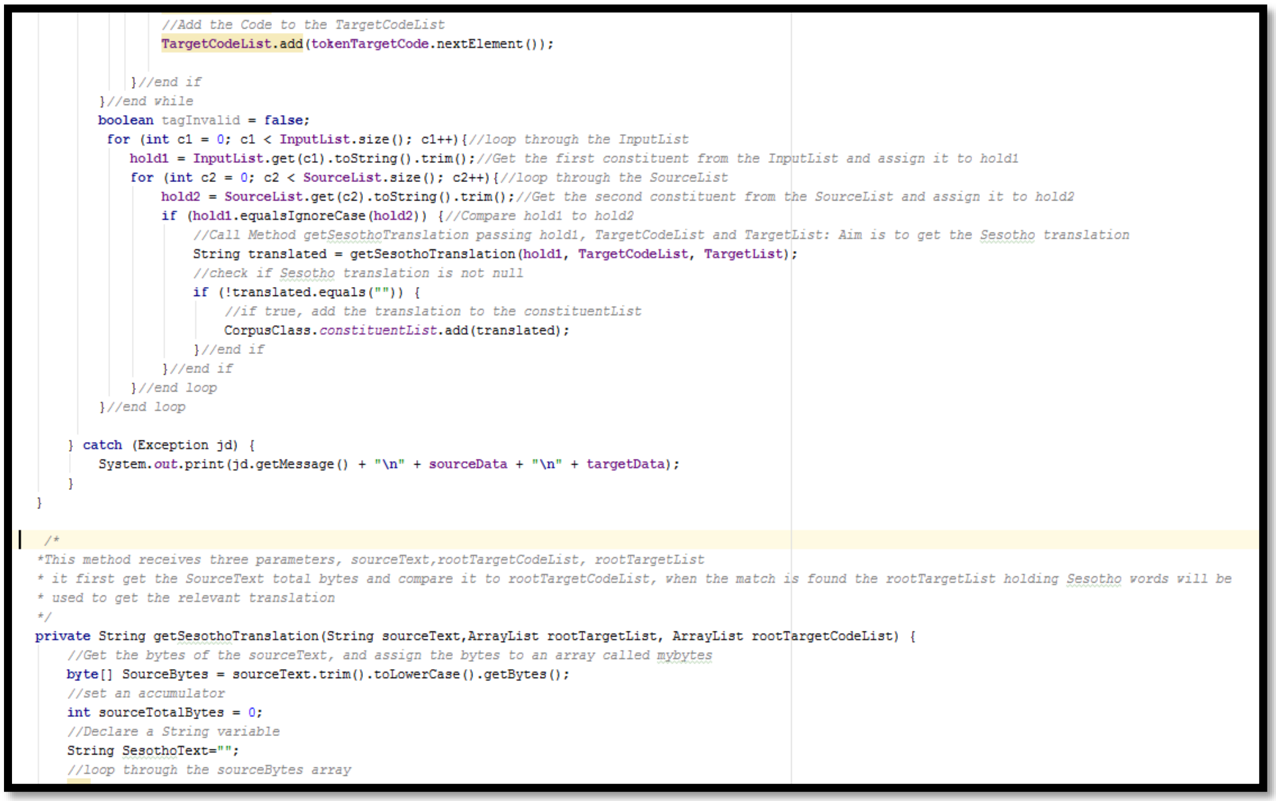


```
    public void onCreate(SQLiteDatabase db)
    {
        db.execSQL("CREATE TABLE " + DATABASE_TABLE + "(" +
                KEY_ROWID + " INTEGER PRIMARY KEY AUTOINCREMENT, " +
                KEY_WORD + " TEXT NOT NULL, " +
                KEY_POS + " TEXT NOT NULL);");

    }

}

public DbsClass open() throws SQLException// Allow us to modify the database
{
    ourHelper = new DBHelper(ourContext);
    ourDatabase = ourHelper.getWritableDatabase();
    return this;
}

public void close()// close the connection
{
    ourHelper.close();
}

public boolean deleteAllRows() { return ourDatabase.delete(DATABASE_TABLE, null, null) > 0; }

public  void createEntry()// create a new entry
{
    ContentValues cv = new ContentValues();


    cv.put(KEY_WORD, "him"); cv.put(KEY_POS, "NN"); ourDatabase.insert(DATABASE_TABLE, null, cv);
    cv.put(KEY_WORD, "his"); cv.put(KEY_POS, "NN");ourDatabase.insert(DATABASE_TABLE, null, cv);
    cv.put(KEY_WORD, "English"); cv.put(KEY_POS, "NN");
    ourDatabase.insert(DATABASE_TABLE, null, cv);
    cv.put(KEY_WORD, "will"); cv.put(KEY_POS, "VB"); ourDatabase.insert(DATABASE_TABLE, null, cv);
    cv.put(KEY_WORD, "for"); cv.put(KEY_POS, "FOR"); ourDatabase.insert(DATABASE_TABLE, null, cv);
    cv.put(KEY_WORD, "ticket"); cv.put(KEY_POS, "NN"); ourDatabase.insert(DATABASE_TABLE, null, cv);
    cv.put(KEY_WORD, "white"); cv.put(KEY_POS, "ADJ"); ourDatabase.insert(DATABASE_TABLE, null, cv);


    cv.put(KEY_WORD, "house"); cv.put(KEY_POS, "NN"); ourDatabase.insert(DATABASE_TABLE, null, cv);
```

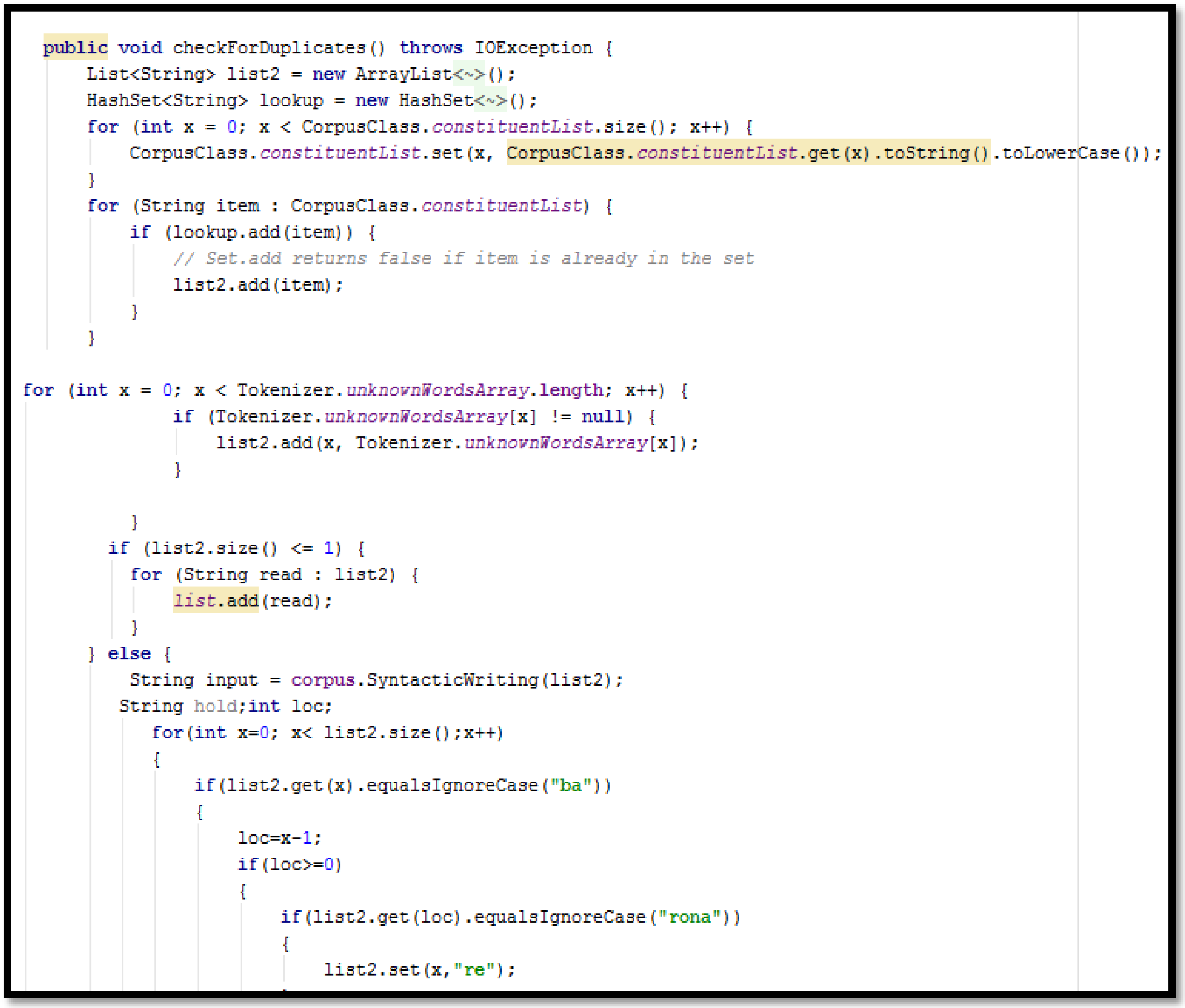
Appendix 3: Machine dictionary database code



```
public void MainSegment() {
    String h1;
    tokenData = new StringTokenizer(data, "*");
    tokenSource = new StringTokenizer(sourceData, "*");
    tokenTarget = new StringTokenizer(targetData, "*");

    rootDataList = new ArrayList();
    rootTargetList = new ArrayList();
    rootSourceList = new ArrayList();
    try {
        while (tokenData.hasMoreTokens()) {
            h1 = tokenData.nextElement().toString().trim();
            if (!h1.equals("")) {
                rootDataList.add(h1);
            }

        }
        while (tokenSource.hasMoreTokens()) {
            h1 = tokenSource.nextElement().toString().trim();
            if (!h1.equals("")) {
                rootSourceList.add(h1);
            }

        }
        //we tokenize line fromTokenTarget, we store a target data and target code
        while (tokenTarget.hasMoreTokens()) {
            h1 = tokenTarget.nextElement().toString().trim();
            if (!h1.equals("")) {

                tokenTargetCode = new StringTokenizer(h1, "_");
                rootTargetList.add(tokenTargetCode.nextElement());
                rooTargetCodeList.add(tokenTargetCode.nextElement());

            }

        }

        boolean tagInvalid = false;
```

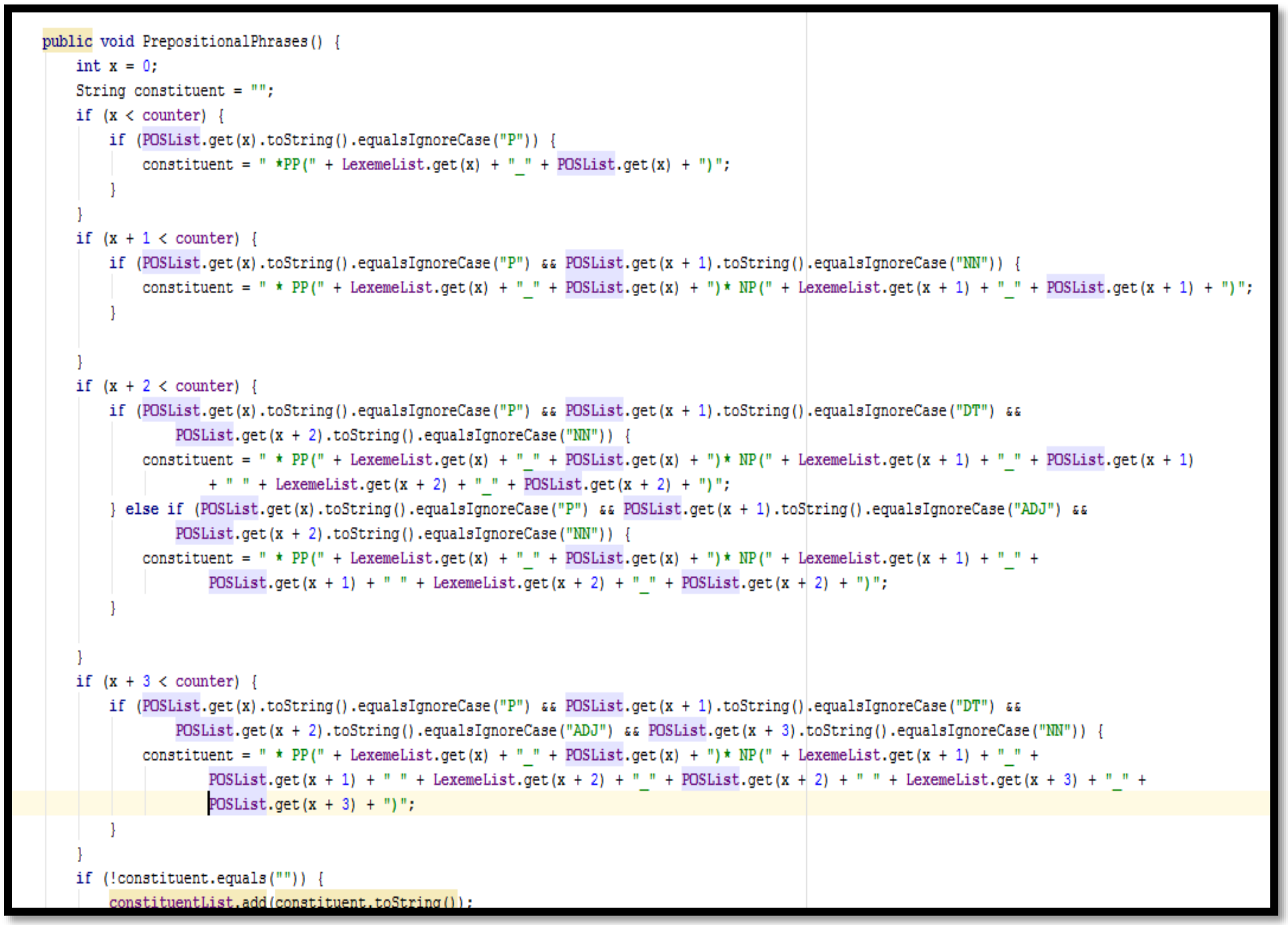


```
                    //Add the Code to the TargetCodeList
                    TargetCodeList.add(tokenTargetCode.nextElement());

                }//end if
            }//end while
            boolean tagInvalid = false;
             for (int c1 = 0; c1 < InputList.size(); c1++){//loop through the InputList
                hold1 = InputList.get(c1).toString().trim();//Get the first constituent from the InputList and assign it to hold1
                for (int c2 = 0; c2 < SourceList.size(); c2++){//loop through the SourceList
                    hold2 = SourceList.get(c2).toString().trim();//Get the second constituent from the SourceList and assign it to hold2
                    if (hold1.equalsIgnoreCase(hold2)) {//Compare hold1 to hold2
                        //Call Method getSesothoTranslation passing hold1, TargetCodeList and TargetList: Aim is to get the Sesotho translation
                        String translated = getSesothoTranslation(hold1, TargetCodeList, TargetList);
                        //check if Sesotho translation is not null
                        if (!translated.equals("")) {
                            //if true, add the translation to the constituentList
                            CorpusClass.constituentList.add(translated);
                        }//end if
                    }//end if
                }//end loop
            }//end loop

        } catch (Exception jd) {
            System.out.print(jd.getMessage() + "\n" + sourceData + "\n" + targetData);
        }
    }

     /*
    *This method receives three parameters, sourceText,rootTargetCodeList, rootTargetList
    * it first get the SourceText total bytes and compare it to rootTargetCodeList, when the match is found the rootTargetList holding Sesotho words will be
    * used to get the relevant translation
    */
    private String getSesothoTranslation(String sourceText,ArrayList rootTargetList, ArrayList rootTargetCodeList) {
        //Get the bytes of the sourceText, and assign the bytes to an array called mybytes
        byte[] SourceBytes = sourceText.trim().toLowerCase().getBytes();
        //set an accumulator
        int sourceTotalBytes = 0;
        //Declare a String variable
        String SesothoText="";
        //loop through the sourceBytes array
```

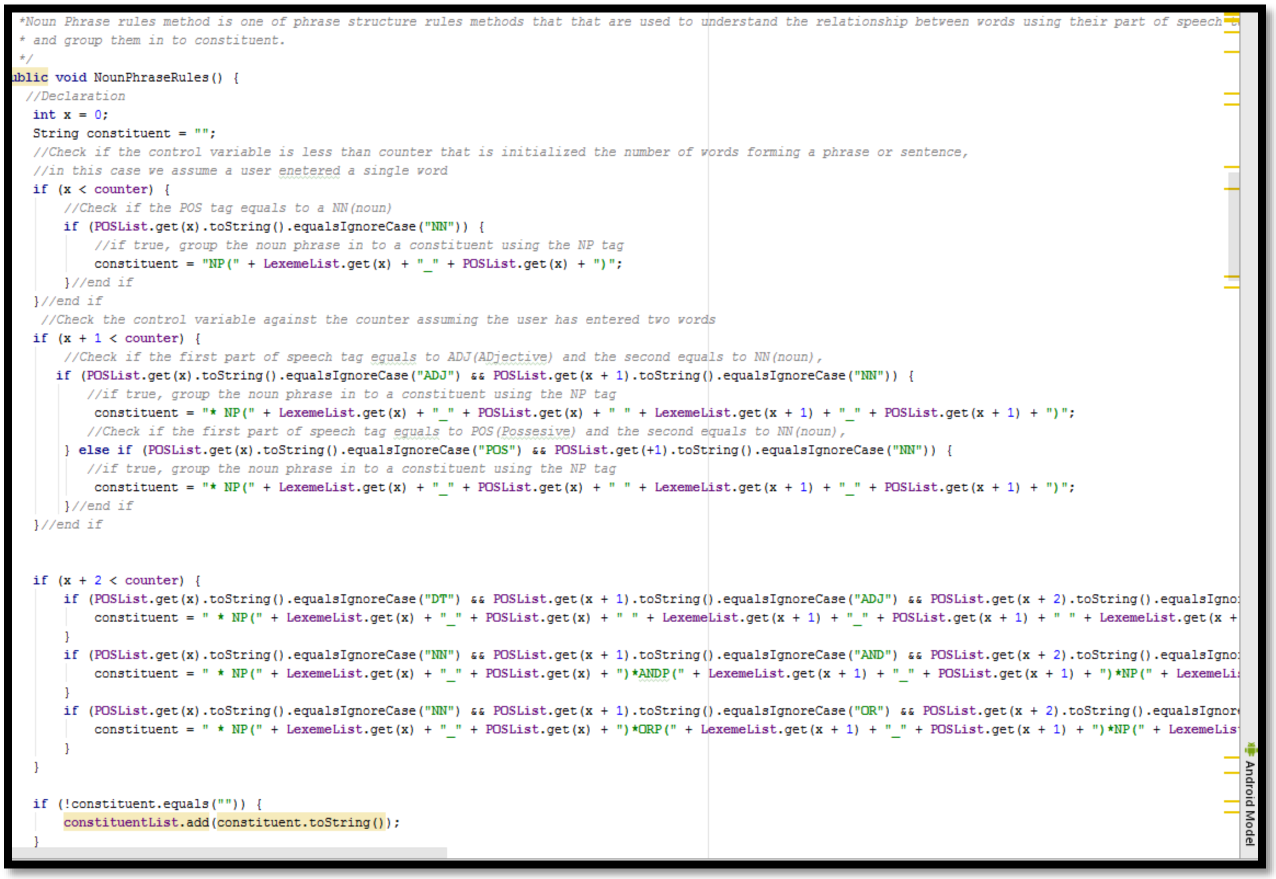


```
  public void checkForDuplicates() throws IOException {
      List<String> list2 = new ArrayList<~>();
      HashSet<String> lookup = new HashSet<~>();
      for (int x = 0; x < CorpusClass.constituentList.size(); x++) {
          CorpusClass.constituentList.set(x, CorpusClass.constituentList.get(x).toString().toLowerCase());
      }
      for (String item : CorpusClass.constituentList) {
          if (lookup.add(item)) {
              // Set.add returns false if item is already in the set
              list2.add(item);
          }
      }

for (int x = 0; x < Tokenizer.unknownWordsArray.length; x++) {
              if (Tokenizer.unknownWordsArray[x] != null) {
                  list2.add(x, Tokenizer.unknownWordsArray[x]);
              }

          }
        if (list2.size() <= 1) {
          for (String read : list2) {
              list.add(read);
          }
      } else {
          String input = corpus.SyntacticWriting(list2);
         String hold;int loc;
            for(int x=0; x< list2.size();x++)
            {
                if(list2.get(x).equalsIgnoreCase("ba"))
                {
                    loc=x-1;
                    if(loc>=0)
                    {
                        if(list2.get(loc).equalsIgnoreCase("rona"))
                        {
                            list2.set(x,"re");
```

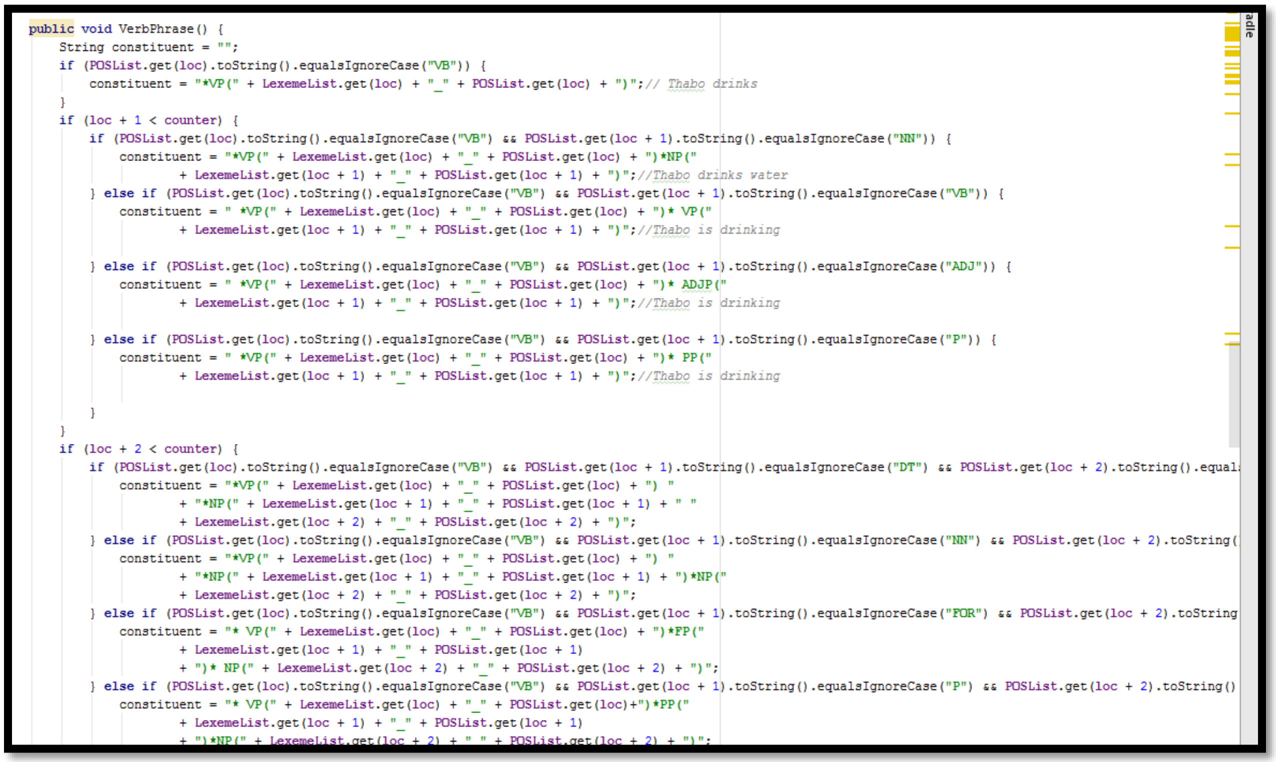
Appendix 4: Machine Translator Code



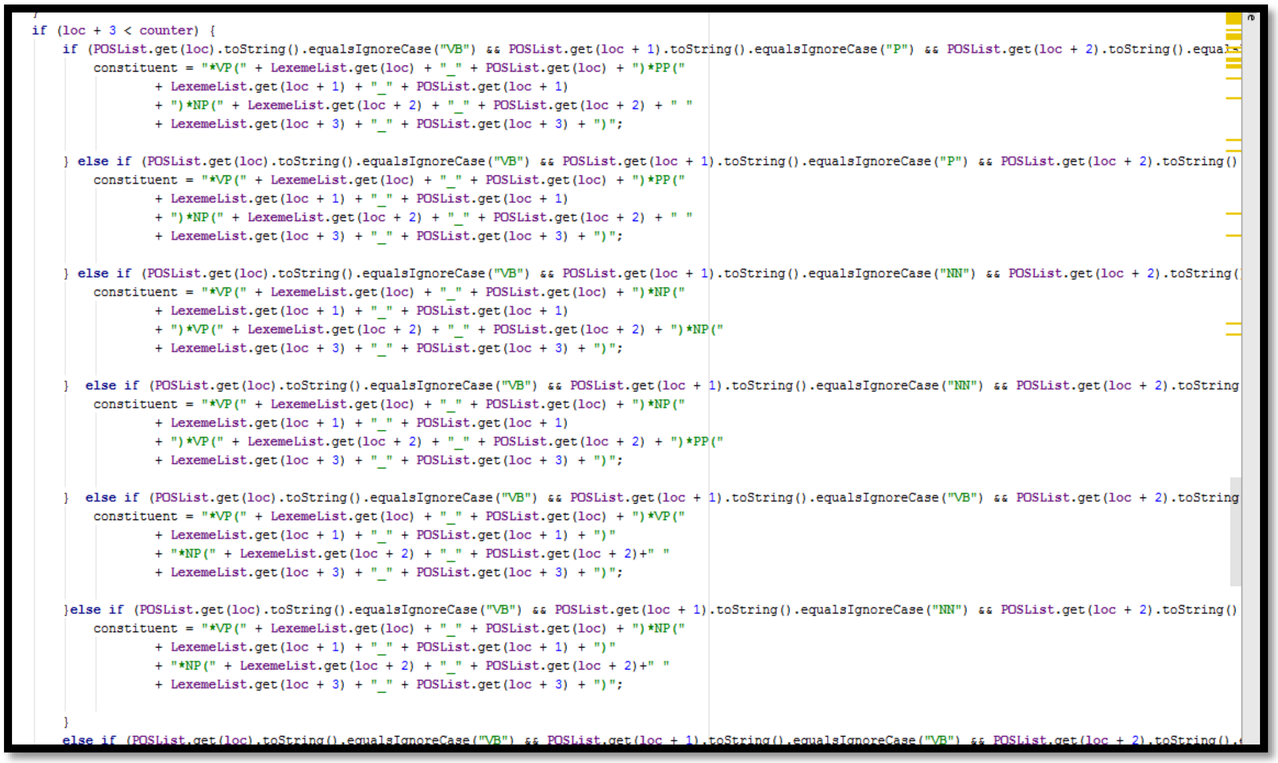
```java
public void PrepositionalPhrases() {
    int x = 0;
    String constituent = "";
    if (x < counter) {
        if (POSList.get(x).toString().equalsIgnoreCase("P")) {
            constituent = " *PP(" + LexemeList.get(x) + "_" + POSList.get(x) + ")";
        }
    }
    if (x + 1 < counter) {
        if (POSList.get(x).toString().equalsIgnoreCase("P") && POSList.get(x + 1).toString().equalsIgnoreCase("NN")) {
            constituent = " * PP(" + LexemeList.get(x) + "_" + POSList.get(x) + ")* NP(" + LexemeList.get(x + 1) + "_" + POSList.get(x + 1) + ")";
        }

    }
    if (x + 2 < counter) {
        if (POSList.get(x).toString().equalsIgnoreCase("P") && POSList.get(x + 1).toString().equalsIgnoreCase("DT") &&
                POSList.get(x + 2).toString().equalsIgnoreCase("NN")) {
            constituent = " * PP(" + LexemeList.get(x) + "_" + POSList.get(x) + ")* NP(" + LexemeList.get(x + 1) + "_" + POSList.get(x + 1)
                    + " " + LexemeList.get(x + 2) + "_" + POSList.get(x + 2) + ")";
        } else if (POSList.get(x).toString().equalsIgnoreCase("P") && POSList.get(x + 1).toString().equalsIgnoreCase("ADJ") &&
                POSList.get(x + 2).toString().equalsIgnoreCase("NN")) {
            constituent = " * PP(" + LexemeList.get(x) + "_" + POSList.get(x) + ")* NP(" + LexemeList.get(x + 1) + "_" +
                    POSList.get(x + 1) + " " + LexemeList.get(x + 2) + "_" + POSList.get(x + 2) + ")";
        }

    }
    if (x + 3 < counter) {
        if (POSList.get(x).toString().equalsIgnoreCase("P") && POSList.get(x + 1).toString().equalsIgnoreCase("DT") &&
                POSList.get(x + 2).toString().equalsIgnoreCase("ADJ") && POSList.get(x + 3).toString().equalsIgnoreCase("NN")) {
            constituent = " * PP(" + LexemeList.get(x) + "_" + POSList.get(x) + ")* NP(" + LexemeList.get(x + 1) + "_" +
                    POSList.get(x + 1) + " " + LexemeList.get(x + 2) + "_" + POSList.get(x + 2) + " " + LexemeList.get(x + 3) + "_" +
                    POSList.get(x + 3) + ")";
        }
    }
    if (!constituent.equals("")) {
        constituentList.add(constituent.toString());
```



```
*Noun Phrase rules method is one of phrase structure rules methods that that are used to understand the relationship between words using their part of speech
* and group them in to constituent.
*/
ublic void NounPhraseRules() {
 //Declaration
  int x = 0;
  String constituent = "";
  //Check if the control variable is less than counter that is initialized the number of words forming a phrase or sentence,
  //in this case we assume a user enetered a single word
  if (x < counter) {
      //Check if the POS tag equals to a NN(noun)
      if (POSList.get(x).toString().equalsIgnoreCase("NN")) {
          //if true, group the noun phrase in to a constituent using the NP tag
          constituent = "NP(" + LexemeList.get(x) + "_" + POSList.get(x) + ")";
      }//end if
  }//end if
   //Check the control variable against the counter assuming the user has entered two words
  if (x + 1 < counter) {
      //Check if the first part of speech tag equals to ADJ(ADjective) and the second equals to NN(noun),
     if (POSList.get(x).toString().equalsIgnoreCase("ADJ") && POSList.get(x + 1).toString().equalsIgnoreCase("NN")) {
         //if true, group the noun phrase in to a constituent using the NP tag
          constituent = "* NP(" + LexemeList.get(x) + "_" + POSList.get(x) + " " + LexemeList.get(x + 1) + "_" + POSList.get(x + 1) + ")";
         //Check if the first part of speech tag equals to POS(Possesive) and the second equals to NN(noun),
      } else if (POSList.get(x).toString().equalsIgnoreCase("POS") && POSList.get(+1).toString().equalsIgnoreCase("NN")) {
         //if true, group the noun phrase in to a constituent using the NP tag
          constituent = "* NP(" + LexemeList.get(x) + "_" + POSList.get(x) + " " + LexemeList.get(x + 1) + "_" + POSList.get(x + 1) + ")";
      }//end if
  }//end if


  if (x + 2 < counter) {
      if (POSList.get(x).toString().equalsIgnoreCase("DT") && POSList.get(x + 1).toString().equalsIgnoreCase("ADJ") && POSList.get(x + 2).toString().equalsIgno
          constituent = " * NP(" + LexemeList.get(x) + "_" + POSList.get(x) + " " + LexemeList.get(x + 1) + "_" + POSList.get(x + 1) + " " + LexemeList.get(x +
      }
      if (POSList.get(x).toString().equalsIgnoreCase("NN") && POSList.get(x + 1).toString().equalsIgnoreCase("AND") && POSList.get(x + 2).toString().equalsIgno
          constituent = " * NP(" + LexemeList.get(x) + "_" + POSList.get(x) + ")*ANDP(" + LexemeList.get(x + 1) + "_" + POSList.get(x + 1) + ")*NP(" + LexemeLi
      }
      if (POSList.get(x).toString().equalsIgnoreCase("NN") && POSList.get(x + 1).toString().equalsIgnoreCase("OR") && POSList.get(x + 2).toString().equalsIgnore
          constituent = " * NP(" + LexemeList.get(x) + "_" + POSList.get(x) + ")*ORP(" + LexemeList.get(x + 1) + "_" + POSList.get(x + 1) + ")*NP(" + LexemeLis
      }
  }

  if (!constituent.equals("")) {
      constituentList.add(constituent.toString());
  }
```

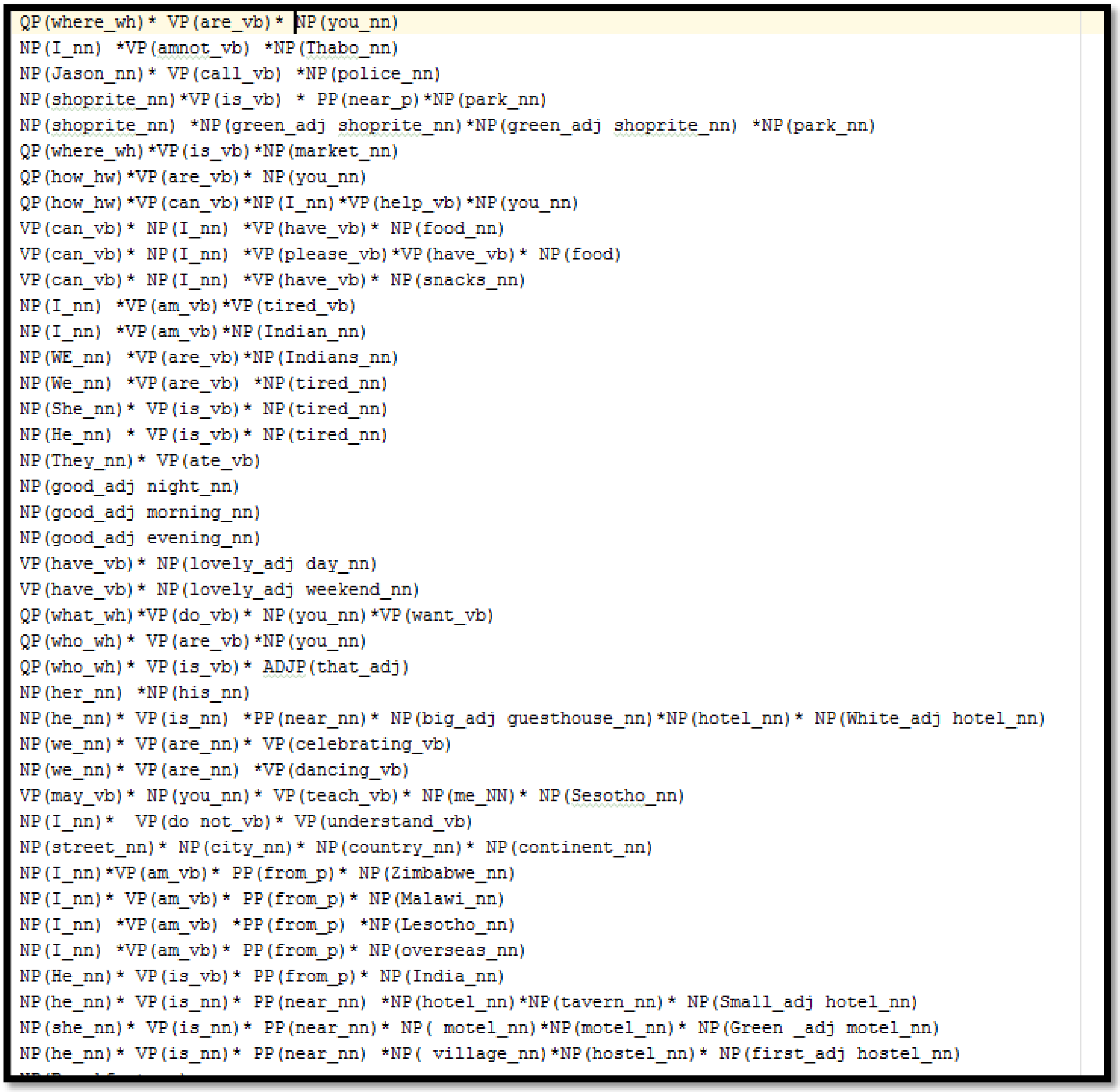


```
public void VerbPhrase() {
    String constituent = "";
    if (POSList.get(loc).toString().equalsIgnoreCase("VB")) {
        constituent = "*VP(" + LexemeList.get(loc) + "_" + POSList.get(loc) + ")";// Thabo drinks
    }
    if (loc + 1 < counter) {
        if (POSList.get(loc).toString().equalsIgnoreCase("VB") && POSList.get(loc + 1).toString().equalsIgnoreCase("NN")) {
            constituent = "*VP(" + LexemeList.get(loc) + "_" + POSList.get(loc) + ")*NP("
                    + LexemeList.get(loc + 1) + "_" + POSList.get(loc + 1) + ")";//Thabo drinks water
        } else if (POSList.get(loc).toString().equalsIgnoreCase("VB") && POSList.get(loc + 1).toString().equalsIgnoreCase("VB")) {
            constituent = " *VP(" + LexemeList.get(loc) + "_" + POSList.get(loc) + ")* VP("
                    + LexemeList.get(loc + 1) + "_" + POSList.get(loc + 1) + ")";//Thabo is drinking

        } else if (POSList.get(loc).toString().equalsIgnoreCase("VB") && POSList.get(loc + 1).toString().equalsIgnoreCase("ADJ")) {
            constituent = " *VP(" + LexemeList.get(loc) + "_" + POSList.get(loc) + ")* ADJP("
                    + LexemeList.get(loc + 1) + "_" + POSList.get(loc + 1) + ")";//Thabo is drinking

        } else if (POSList.get(loc).toString().equalsIgnoreCase("VB") && POSList.get(loc + 1).toString().equalsIgnoreCase("P")) {
            constituent = " *VP(" + LexemeList.get(loc) + "_" + POSList.get(loc) + ")* PP("
                    + LexemeList.get(loc + 1) + "_" + POSList.get(loc + 1) + ")";//Thabo is drinking

        }
    }
    if (loc + 2 < counter) {
        if (POSList.get(loc).toString().equalsIgnoreCase("VB") && POSList.get(loc + 1).toString().equalsIgnoreCase("DT") && POSList.get(loc + 2).toString().equal
            constituent = "*VP(" + LexemeList.get(loc) + "_" + POSList.get(loc) + ") "
                    + "*NP(" + LexemeList.get(loc + 1) + "_" + POSList.get(loc + 1) + " "
                    + LexemeList.get(loc + 2) + "_" + POSList.get(loc + 2) + ")";
        } else if (POSList.get(loc).toString().equalsIgnoreCase("VB") && POSList.get(loc + 1).toString().equalsIgnoreCase("NN") && POSList.get(loc + 2).toString(
            constituent = "*VP(" + LexemeList.get(loc) + "_" + POSList.get(loc) + ") "
                    + "*NP(" + LexemeList.get(loc + 1) + "_" + POSList.get(loc + 1) + ")*NP("
                    + LexemeList.get(loc + 2) + "_" + POSList.get(loc + 2) + ")";
        } else if (POSList.get(loc).toString().equalsIgnoreCase("VB") && POSList.get(loc + 1).toString().equalsIgnoreCase("FOR") && POSList.get(loc + 2).toString
            constituent = "* VP(" + LexemeList.get(loc) + "_" + POSList.get(loc) + ")*FP("
                    + LexemeList.get(loc + 1) + "_" + POSList.get(loc + 1)
                    + ")* NP(" + LexemeList.get(loc + 2) + "_" + POSList.get(loc + 2) + ")";
        } else if (POSList.get(loc).toString().equalsIgnoreCase("VB") && POSList.get(loc + 1).toString().equalsIgnoreCase("P") && POSList.get(loc + 2).toString()
            constituent = "* VP(" + LexemeList.get(loc) + "_" + POSList.get(loc)+")*PP("
                    + LexemeList.get(loc + 1) + "_" + POSList.get(loc + 1)
                    + ")*NP(" + LexemeList.get(loc + 2) + " " + POSList.get(loc + 2) + ")";
```



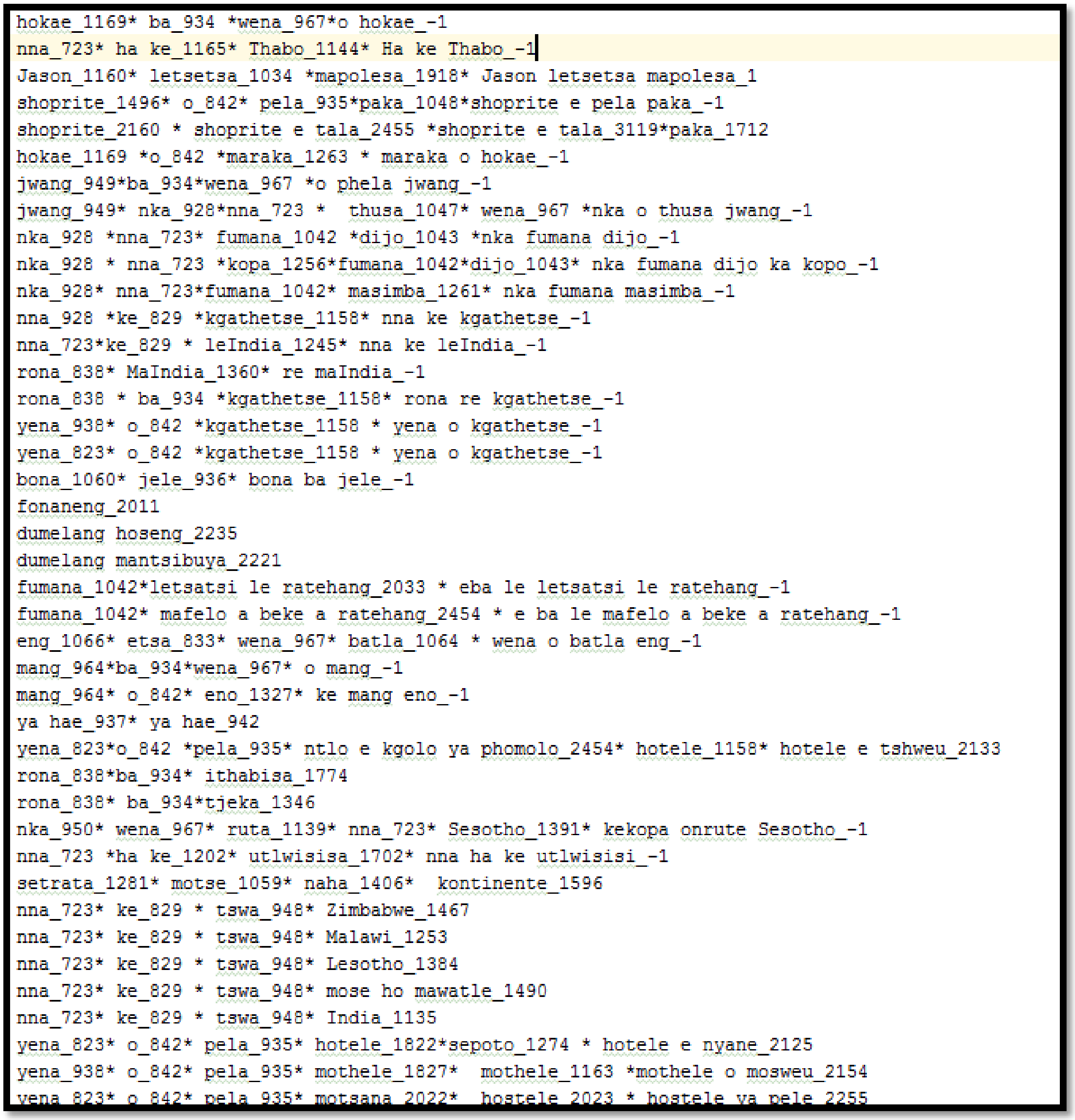
```
if (loc + 3 < counter) {
    if (POSList.get(loc).toString().equalsIgnoreCase("VB") && POSList.get(loc + 1).toString().equalsIgnoreCase("P") && POSList.get(loc + 2).toString().equal
        constituent = "*VP(" + LexemeList.get(loc) + "_" + POSList.get(loc) + ")*PP("
                + LexemeList.get(loc + 1) + "_" + POSList.get(loc + 1)
                + ")*NP(" + LexemeList.get(loc + 2) + "_" + POSList.get(loc + 2) + " "
                + LexemeList.get(loc + 3) + "_" + POSList.get(loc + 3) + ")";

    } else if (POSList.get(loc).toString().equalsIgnoreCase("VB") && POSList.get(loc + 1).toString().equalsIgnoreCase("P") && POSList.get(loc + 2).toString()
        constituent = "*VP(" + LexemeList.get(loc) + "_" + POSList.get(loc) + ")*PP("
                + LexemeList.get(loc + 1) + "_" + POSList.get(loc + 1)
                + ")*NP(" + LexemeList.get(loc + 2) + "_" + POSList.get(loc + 2) + " "
                + LexemeList.get(loc + 3) + "_" + POSList.get(loc + 3) + ")";

    } else if (POSList.get(loc).toString().equalsIgnoreCase("VB") && POSList.get(loc + 1).toString().equalsIgnoreCase("NN") && POSList.get(loc + 2).toString(
        constituent = "*VP(" + LexemeList.get(loc) + "_" + POSList.get(loc) + ")*NP("
                + LexemeList.get(loc + 1) + "_" + POSList.get(loc + 1)
                + ")*VP(" + LexemeList.get(loc + 2) + "_" + POSList.get(loc + 2) + ")*NP("
                + LexemeList.get(loc + 3) + "_" + POSList.get(loc + 3) + ")";

    }  else if (POSList.get(loc).toString().equalsIgnoreCase("VB") && POSList.get(loc + 1).toString().equalsIgnoreCase("NN") && POSList.get(loc + 2).toString
        constituent = "*VP(" + LexemeList.get(loc) + "_" + POSList.get(loc) + ")*NP("
                + LexemeList.get(loc + 1) + "_" + POSList.get(loc + 1)
                + ")*VP(" + LexemeList.get(loc + 2) + "_" + POSList.get(loc + 2) + ")*PP("
                + LexemeList.get(loc + 3) + "_" + POSList.get(loc + 3) + ")";

    }  else if (POSList.get(loc).toString().equalsIgnoreCase("VB") && POSList.get(loc + 1).toString().equalsIgnoreCase("VB") && POSList.get(loc + 2).toString
        constituent = "*VP(" + LexemeList.get(loc) + "_" + POSList.get(loc) + ")*VP("
                + LexemeList.get(loc + 1) + "_" + POSList.get(loc + 1) + ")"
                + "*NP(" + LexemeList.get(loc + 2) + "_" + POSList.get(loc + 2)+" "
                + LexemeList.get(loc + 3) + "_" + POSList.get(loc + 3) + ")";

    }else if (POSList.get(loc).toString().equalsIgnoreCase("VB") && POSList.get(loc + 1).toString().equalsIgnoreCase("NN") && POSList.get(loc + 2).toString()
        constituent = "*VP(" + LexemeList.get(loc) + "_" + POSList.get(loc) + ")*NP("
                + LexemeList.get(loc + 1) + "_" + POSList.get(loc + 1) + ")"
                + "*NP(" + LexemeList.get(loc + 2) + "_" + POSList.get(loc + 2)+" "
                + LexemeList.get(loc + 3) + "_" + POSList.get(loc + 3) + ")";

    }
    else if (POSList.get(loc).toString().equalsIgnoreCase("VB") && POSList.get(loc + 1).toString().equalsIgnoreCase("VB") && POSList.get(loc + 2).toString().
```

Appendix 5: Phrase Structure rules code



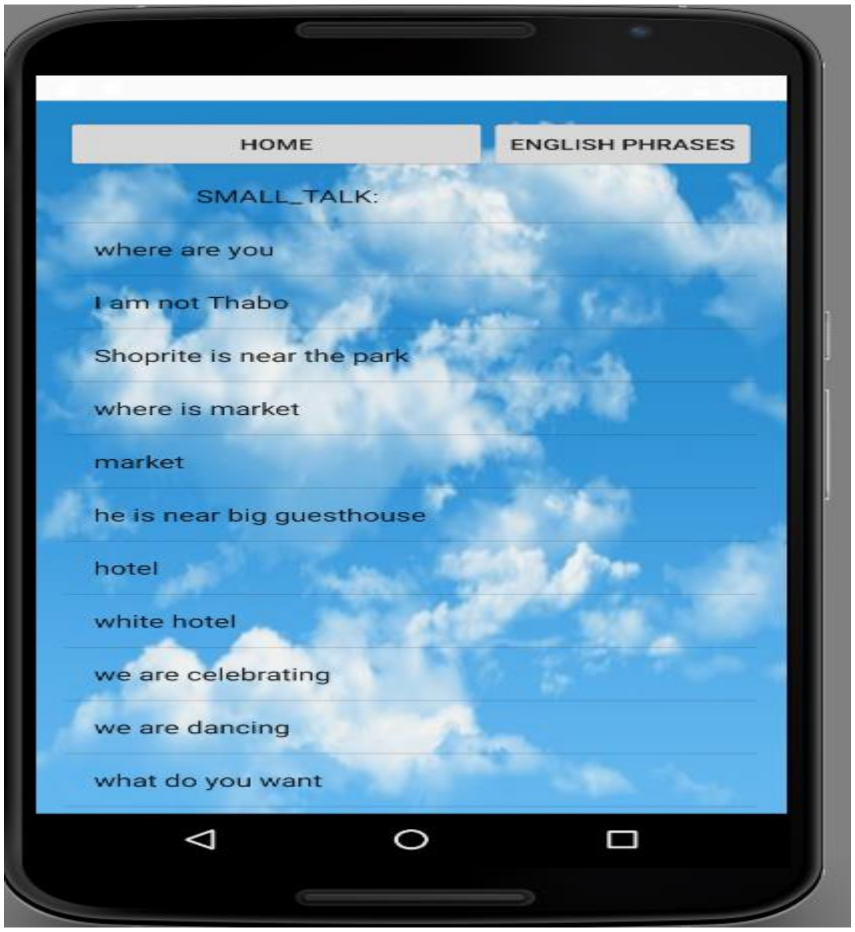
```
QP(where_wh)* VP(are_vb)* NP(you_nn)
NP(I_nn) *VP(amnot_vb) *NP(Thabo_nn)
NP(Jason_nn)* VP(call_vb) *NP(police_nn)
NP(shoprite_nn)*VP(is_vb) * PP(near_p)*NP(park_nn)
NP(shoprite_nn) *NP(green_adj shoprite_nn)*NP(green_adj shoprite_nn) *NP(park_nn)
QP(where_wh)*VP(is_vb)*NP(market_nn)
QP(how_hw)*VP(are_vb)* NP(you_nn)
QP(how_hw)*VP(can_vb)*NP(I_nn)*VP(help_vb)*NP(you_nn)
VP(can_vb)* NP(I_nn) *VP(have_vb)* NP(food_nn)
VP(can_vb)* NP(I_nn) *VP(please_vb)*VP(have_vb)* NP(food)
VP(can_vb)* NP(I_nn) *VP(have_vb)* NP(snacks_nn)
NP(I_nn) *VP(am_vb)*VP(tired_vb)
NP(I_nn) *VP(am_vb)*NP(Indian_nn)
NP(WE_nn) *VP(are_vb)*NP(Indians_nn)
NP(We_nn) *VP(are_vb) *NP(tired_nn)
NP(She_nn)* VP(is_vb)* NP(tired_nn)
NP(He_nn) * VP(is_vb)* NP(tired_nn)
NP(They_nn)* VP(ate_vb)
NP(good_adj night_nn)
NP(good_adj morning_nn)
NP(good_adj evening_nn)
VP(have_vb)* NP(lovely_adj day_nn)
VP(have_vb)* NP(lovely_adj weekend_nn)
QP(what_wh)*VP(do_vb)* NP(you_nn)*VP(want_vb)
QP(who_wh)* VP(are_vb)*NP(you_nn)
QP(who_wh)* VP(is_vb)* ADJP(that_adj)
NP(her_nn) *NP(his_nn)
NP(he_nn)* VP(is_nn) *PP(near_nn)* NP(big_adj guesthouse_nn)*NP(hotel_nn)* NP(White_adj hotel_nn)
NP(we_nn)* VP(are_nn)* VP(celebrating_vb)
NP(we_nn)* VP(are_nn) *VP(dancing_vb)
VP(may_vb)* NP(you_nn)* VP(teach_vb)* NP(me_NN)* NP(Sesotho_nn)
NP(I_nn)*  VP(do not_vb)* VP(understand_vb)
NP(street_nn)* NP(city_nn)* NP(country_nn)* NP(continent_nn)
NP(I_nn)*VP(am_vb)* PP(from_p)* NP(Zimbabwe_nn)
NP(I_nn)* VP(am_vb)* PP(from_p)* NP(Malawi_nn)
NP(I_nn) *VP(am_vb) *PP(from_p) *NP(Lesotho_nn)
NP(I_nn) *VP(am_vb)* PP(from_p)* NP(overseas_nn)
NP(He_nn)* VP(is_vb)* PP(from_p)* NP(India_nn)
NP(he_nn)* VP(is_nn)* PP(near_nn) *NP(hotel_nn)*NP(tavern_nn)* NP(Small_adj hotel_nn)
NP(she_nn)* VP(is_nn)* PP(near_nn)* NP( motel_nn)*NP(motel_nn)* NP(Green _adj motel_nn)
NP(he_nn)* VP(is_nn)* PP(near_nn) *NP( village_nn)*NP(hostel_nn)* NP(first_adj hostel_nn)
```

Appendix 6: Source Corpus



```
hokae_1169* ba_934 *wena_967*o hokae_-1
nna_723* ha ke_1165* Thabo_1144* Ha ke Thabo_-1
Jason_1160* letsetsa_1034 *mapolesa_1918* Jason letsetsa mapolesa_1
shoprite_1496* o_842* pela_935*paka_1048*shoprite e pela paka_-1
shoprite_2160 * shoprite e tala_2455 *shoprite e tala_3119*paka_1712
hokae_1169 *o_842 *maraka_1263 * maraka o hokae_-1
jwang_949*ba_934*wena_967 *o phela jwang_-1
jwang_949* nka_928*nna_723 *  thusa_1047* wena_967 *nka o thusa jwang_-1
nka_928 *nna_723* fumana_1042 *dijo_1043 *nka fumana dijo_-1
nka_928 * nna_723 *kopa_1256*fumana_1042*dijo_1043* nka fumana dijo ka kopo_-1
nka_928* nna_723*fumana_1042* masimba_1261* nka fumana masimba_-1
nna_928 *ke_829 *kgathetse_1158* nna ke kgathetse_-1
nna_723*ke_829 * leIndia_1245* nna ke leIndia_-1
rona_838* MaIndia_1360* re maIndia_-1
rona_838 * ba_934 *kgathetse_1158* rona re kgathetse_-1
yena_938* o_842 *kgathetse_1158 * yena o kgathetse_-1
yena_823* o_842 *kgathetse_1158 * yena o kgathetse_-1
bona_1060* jele_936* bona ba jele_-1
fonaneng_2011
dumelang hoseng_2235
dumelang mantsibuya_2221
fumana_1042*letsatsi le ratehang_2033 * eba le letsatsi le ratehang_-1
fumana_1042* mafelo a beke a ratehang_2454 * e ba le mafelo a beke a ratehang_-1
eng_1066* etsa_833* wena_967* batla_1064 * wena o batla eng_-1
mang_964*ba_934*wena_967* o mang_-1
mang_964* o_842* eno_1327* ke mang eno_-1
ya hae_937* ya hae_942
yena_823*o_842 *pela_935* ntlo e kgolo ya phomolo_2454* hotele_1158* hotele e tshweu_2133
rona_838*ba_934* ithabisa_1774
rona_838* ba_934*tjeka_1346
nka_950* wena_967* ruta_1139* nna_723* Sesotho_1391* kekopa onrute Sesotho_-1
nna_723 *ha ke_1202* utlwisisa_1702* nna ha ke utlwisisi_-1
setrata_1281* motse_1059* naha_1406*  kontinente_1596
nna_723* ke_829 * tswa_948* Zimbabwe_1467
nna_723* ke_829 * tswa_948* Malawi_1253
nna_723* ke_829 * tswa_948* Lesotho_1384
nna_723* ke_829 * tswa_948* mose ho mawatle_1490
nna_723* ke_829 * tswa_948* India_1135
yena_823* o_842* pela_935* hotele_1822*sepoto_1274 * hotele e nyane_2125
yena_938* o_842* pela_935* mothele_1827*  mothele_1163 *mothele o mosweu_2154
yena_823* o_842* pela_935* motsana_2022*  hostele_2023 * hostele ya pele_2255
```

Appendix 7: Target Corpus



SMALL_TALK:

where are you

I am not Thabo

Shoprite is near the park

where is market

market

he is near big guesthouse

hotel

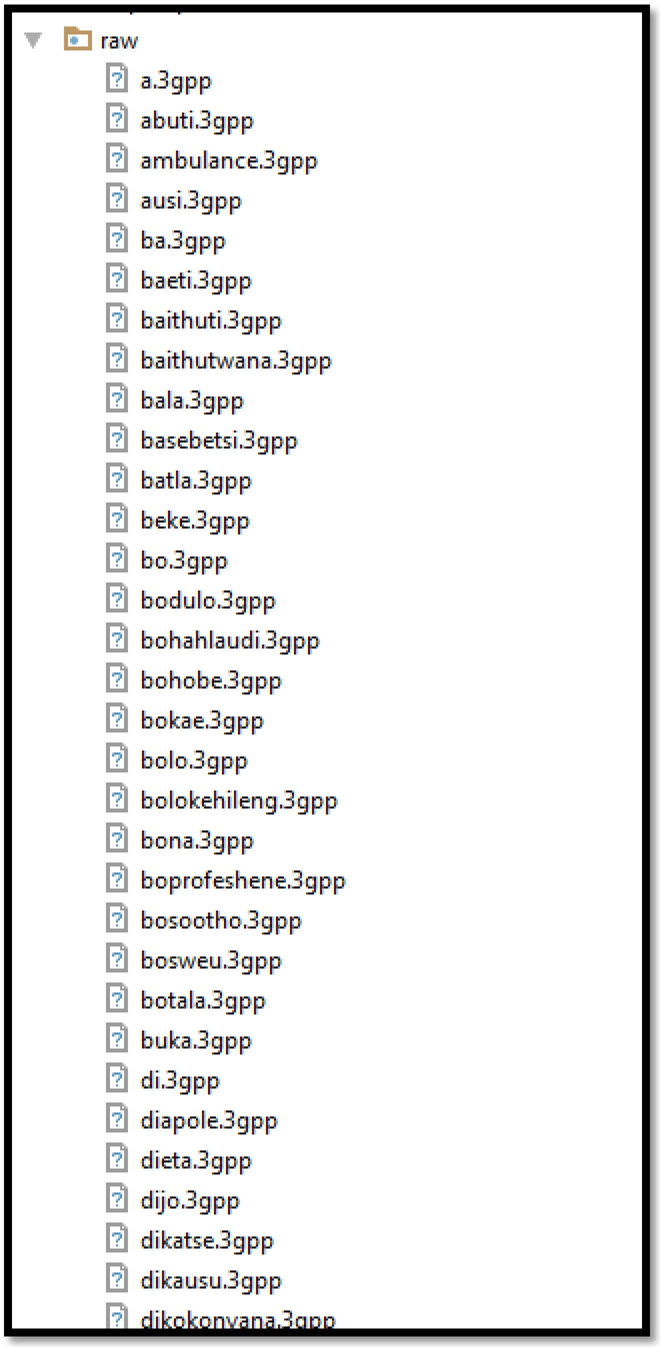
white hotel

we are celebrating

we are dancing

what do you want

Appendix 8: English phrases that a user can select from and translate


raw
a.3gpp
abuti.3gpp
ambulance.3gpp
ausi.3gpp
ba.3gpp
baeti.3gpp
baithuti.3gpp
baithutwana.3gpp
bala.3gpp
basebetsi.3gpp
batla.3gpp
beke.3gpp
bo.3gpp
bodulo.3gpp
bohahlaudi.3gpp
bohobe.3gpp
bokae.3gpp
bolo.3gpp
bolokehileng.3gpp
bona.3gpp
boprofeshene.3gpp
bosootho.3gpp
bosweu.3gpp
botala.3gpp
buka.3gpp
di.3gpp
diapole.3gpp
dieta.3gpp
dijo.3gpp
dikatse.3gpp
dikausu.3gpp
dikokonyana.3gpp

Appendix 9: Sesotho recordings inside a file “raw”

Chapter 4: Questionnaires

RESEARCH INVITATION LETTER
I am pleased to invite you to participate in research aimed at identifying basic English words and phrases that will be translated to the foreign language (Sesotho) using UmobiTalk(Ubiquitous mobile based speech learning for Sesotho). UmobiTalk is a proposed speech based application that aims to remove the language barrier between migrants and Sesotho speakers by training them to learn words and proper pronunciation of words. Be assured that any information you provide will be treated in the strictest confidence and your participation will not be identifiable in the resulting report. You are entirely free to discontinue your participation at any time or to decline to answer particular questions. I will seek your consent, on the attached form on which I commit to ensure that your name or identity is not revealed. Direct any enquiries concerning this study to the **main** Researchers contacts below. Thank you for your assistance. Researcher Central University of Technology, Free State, South Africa

Questionnaire page 1

CONSENT FORM

I, the undersigned, confirm that (please tick box as appropriate using ✓ or ✗):

[1]	I have read and understood the information about the research,	☐
[2]	I have been given the opportunity to ask questions about the research and my participation.	☐
[3]	I voluntarily agree to participate in the research.	☐
[4]	I understand I can withdraw at any time without giving reasons and that I will not be penalized for withdrawing	☐
[5]	The procedures regarding confidentiality have been clearly explained to me.	☐
[6]	If applicable, separate terms of consent for forms of data collection have been explained and provided to me.	☐
[7]	The use of the data in research, publications, sharing and archiving has been explained to me.	☐
[8]	I understand that other researchers will have access to this data only if they agree to preserve the confidentiality of the data and if they agree to the terms I have specified in this form.	☐
[9]	Select only **ONE** of the following:	
	• I would like my name used and understand what I have said or written as part of this research will be used in reports, publications and other research outputs so that anything I have contributed to this project can be recognised.	☐
	• I do not want my name used in this research.	☐
[10]	I agree to sign and date this informed consent, along with the Researcher.	☐

Name of Respondent	**Signature**	**Date**
Name of Researcher	**Signature**	**Date**

Questionnaire page 2

SCHEDULED FOR SEPTEMBER 2015

PART A: INTRODUCTION

The motivation behind the development of this tool, Umobitalk is a problems that brought by language barrier in our community. Migrants and other non- Sesotho speakers find it difficult to integrate themselves in to the Sesotho speaking population here in Free-State. This isolation can bring hatred, less understanding of others beliefs and cultural differences etc. So proposed tool come in to rescue by initiating the learning of basic words and phrases so to share ideas, cultural activities, and beliefs with the Sesotho speaking group. The reason behind selection of Sesotho language among other Bantu languages as a case study is findings that proved Sesotho as a major language spoken by a majority of people in Free-State. Another valuable element is to enhance and up rise Bantu developing languages through HLC (human language computing) the same as other western languages. A proposed tool will accommodate almost individuals with different literate level, and special needs such as dyslexic individuals. UmobiTalk is a speech based application that will run on Android smart phones which are now concurring the globe. The user will speak or type a word or phrase they wish to translate in English, the machine will translate it for them in Sesotho, to facilitate the learning, the machine will also read the translated word or phrase. This technology will further be enhanced to allow proper pronunciation of a translated word or phrase; the user will repeat an utterance until they got it right. What makes it outstanding and quite different from human tutor, it is available 24/7, patience, 100% attention, very easy to use and cost is reasonable.

You are requested to participate in this valuable research by completing this questionnaire. You are required to put a mark (√ or X) in the check box to select an option or write down a response for open ended questions.

PART B: MIGRANTS AND NON SESOTHO SPEAKERS QUESTIONNAIRE

Q 1 Names:____ (Optional)

Q 2 Gender? ❑ Male ❑ Female

Q 3 Age bracket?

❑Under 18 ❑18-35 ❑36-45 ❑46-55 ❑56-65 ❑ above 66

Q 4 Highest Education Level:

❑None ❑Primary ❑Secondary ❑Post-Secondary

Q 5 Nationality

❑Black ❑White ❑Indian ❑Colored ❑Other

GREETINGS USING FOREIGN LANGUAGE(SESOTHO)

Indicate to what extent you agree/disagree with each of the following statements		Strongly disagree	Disagree	Not sure	Agree	Strongly Agree
Q 6	I know several Sesotho greetings together with their nouns such as Sir, Madam, father, son etc	1	2	3	4	5

Questionnaire page 4

Q 7	I know Sesotho greeting questions together with their expected answers such as "Good thanks, and how are you"(Ke hantle, wena o phela jwang?)	1	2	3	4	5
Q 8	I know several Sesotho greetings and their pronunciation	1	2	3	4	5

SMALL TALK USING FOREIGN LANGUAGE(SESOTHO)

Indicate to what extent you agree/disagree with each of the following statements		Strongly Disagree	Disagree	Not sure	Agree	Strongly Agree
Q 9	I know Sesotho vocabulary and phrases that can help to socialize with Sesotho speakers	1	2	3	4	5
Q 10	I can pronounce Sesotho language fluently to be clearly understood	1	2	3	4	5

TOURING USING FOREIGN LANGUAGE(SESOTHO)

Indicate to what extent you agree/disagree with each of the following statements		Strongly disagree	Disagree	Not sure	Agree	Strongly Agree
Q 11	I know how to give and ask for direction using Sesotho	1	2	3	4	5
Q 12	I can use the Sesotho knowledge acquired to learn culture, beliefs and ethics of Sesotho speakers	1	2	3	4	5
Q 13	I can use the Sesotho knowledge acquired to learn ecological factors such animals, plants and environment	1	2	3	4	5

EMERGENCY USING FOREIGN LANGUAGE(SESOTHO)

Indicate to what extent you agree/disagree with each of the following statements		Strongly disagree	Disagree	Not sure	Agree	Strongly Agree
Q 14	I can ask for help when I am in emergency or in danger	1	2	3	4	5

Questionnaire page 6

Q 15	I can call and explain to Sesotho speakers	1	2	3	4	5
Q 16	I can notify the intruder to leave because I am armed	1	2	3	4	5
Q 17	I can call and explain the situation to the emergency service providers	1	2	3	4	5

ECONOMY(CUSTOMER) USING FOREIGN LANGUAGE(SESOTHO)

Indicate to what extent you agree/disagree with each of the following statements		Strongly disagree	Disagree	Not sure	Agree	Strongly Agree
Q 18	I can point to some items in a shop and ask for a price.	1	2	3	4	5
Q 19	I can explain to the seller the quantity of items I request	1	2	3	4	5

ECONOMY(SELLER) USING FOREIGN LANGUAGE(SESOTHO)

Q 20	I can understand what the Sesotho speaking customer wants. E.g. “Dumela Ntate, ke kopa dilamunu tse pedi, masimba a mararo le pakete ya matokomane”.	1	2	3	4	5

Questionnaire page 7

Q 21	**I can identify and explain items I'm selling to Sesotho speaking customers.**	**1**	**2**	**3**	**4**	**5**
Q 22	**I can ask the quantity requested by a customer and amount a customer has.**	**1**	**2**	**3**	**4**	**5**

PART E: REQUEST FOR RESPONDENT'S FURTHER INVOLVEMENT (OPTIONAL)

Phase II of this study, will involve development of an Umobitalk application for migrants and other non-Sesotho speakers. Should you be interested to participate, please you are requested to provide your contact details below.

Full Name:	
ID Number:	
Email:	
Your	
Phone Number:	

Chapter 3: Questionnaires

RESEARCH INVITATION LETTER

I am pleased to invite you to participate in research aimed at identifying basic English words and phrases that will be translated to the foreign language (Sesotho) using UmobiTalk(Ubiquitous mobile based speech learning for Sesotho). UmobiTalk is a proposed speech based application that aims to remove the language barrier between migrants and Sesotho speakers by training them to learn words and proper pronunciation of words.

Be assured that any information you provide will be treated in the strictest confidence and your participation will not be identifiable in the resulting report. You are entirely free to discontinue your participation at any time or to decline to answer particular questions.

I will seek your consent, on the attached form on which I commit to ensure that your name or identity is not revealed.

Direct any enquiries concerning this study to the **main** Researchers contacts below.

Thank you for your assistance.

Researcher

Central University of Technology, Free State, South Africa

Questionnaire page 1

CONSENT FORM

I, the undersigned, confirm that (please tick box as appropriate using ✓ or ✗):	
I have read and understood the information about the research,	☐
I have been given the opportunity to ask questions about the research and my participation.	☐
I voluntarily agree to participate in the research.	☐
I understand I can withdraw at any time without giving reasons and that I will not be penalized for withdrawing	☐
The procedures regarding confidentiality have been clearly explained to me.	☐
If applicable, separate terms of consent for forms of data collection have been explained and provided to me.	☐
The use of the data in research, publications, sharing and archiving has been explained to me.	☐
I understand that other researchers will have access to this data only if they agree to preserve the confidentiality of the data and if they agree to the terms I have specified in this form.	☐
Select only **ONE** of the following:	
• I would like my name used and understand what I have said or written as part of this research will be used in reports, publications and other research outputs so that anything I have contributed to this project can be recognised.	☐
• I do not want my name used in this research.	☐
I agree to sign and date this informed consent, along with the Researcher.	☐

______________________	______________	______________
Name of Respondent	**Signature**	**Date**
______________________	______________	______________
Name of Researcher	**Signature**	**Date**

Questionnaire page 2

SCHEDULED FOR SEPTEMBER 2015

PART A: INTRODUCTION

The motivation behind the development of this tool, Umobitalk is a problems that brought by language barrier in our community. Migrants and other non- Sotho speakers find it difficult to integrate themselves in to the Sotho speaking population here in Free-State. This isolation can bring hatred, less understanding of others beliefs and cultural differences etc. So proposed tool come in to rescue by initiating the learning of basic words and phrases so to share ideas, cultural activities, and beliefs with the Sesotho speaking group. The reason behind selection of Sesotho language among other Bantu languages as a case study is findings that proved Sesotho as a major language spoken by a majority of people in Free-State. Another valuable element is to enhance and up rise Bantu developing languages through HLC (human language computing) the same as other western languages. A proposed tool will accommodate almost individuals with different literate level, and special needs such as dyslexic individuals. UmobiTalk is a speech based application that will run on Android smart phones which are now concurring the globe. The user will speak or type a word or phrase they wish to translate in English, the machine will translate it for them in Sesotho, to facilitate the learning, the machine will also read the translated word or phrase. This technology will further be enhanced to allow proper pronunciation of a translated word or phrase; the user will repeat an utterance until they got it right. What makes it outstanding and quite different from human tutor, it is available 24/7, patience, 100% attention, very easy to use and cost is reasonable.

You are requested to participate in this valuable research by completing this questionnaire. You are required to put a mark (√ or X) in the check box to select an option or write down a response for open ended questions.

PART B: MIGRANTS AND NON SESOTHO SPEAKERS QUESTIONNAIRE

1. Names:____ (Optional)
2. Gender? ❑ Male ❑ Female
3. Age bracket?

❑Under 18 ❑18-35 ❑36-45 ❑46-55 ❑56-65 ❑ above 66

4. Highest Education Level:

❑None ❑Primary ❑Secondary ❑Post-Secondary

5. Nationality

❑Black ❑White ❑Indian ❑Colored ❑Other

SYSTEM FUNCTIONALITY

6.	**I can accomplish tasks smoothly without any inconveniences**	**1**	**2**	**3**	**4**	**5**
8.	**I am satisfied with the translation results**	**1**	**2**	**3**	**4**	**5**
7.	**I find the application useful**	**1**	**2**	**3**	**4**	**5**

9. What translation errors did you come across?

____

____

Questionnaire Page 4

10. What can be changed, removed or added of the system functionality?

SYSTEM DESIGN AND USABILITY

Indicate to what extent you agree/disagree with each of the following statements		Strongly disagree	Disagree	Not sure	Agree	Strongly Agree
11.	I can use the application on my own without outside intervention	1	2	3	4	5
12.	Buttons, text and other controls clear to understand	1	2	3	4	5

13. What can be changed, removed or added based on system design and usability

SYSTEM PERFOMANCE

Indicate to what extent you agree/disagree with each of the following statements		Strongly disagree	Disagree	Not sure	Agree	Strongly Agree
14.	I can quickly install and start to use a the application	1	2	3	4	5
15.	I can quickly get the translation feedback	1	2	3	4	5
16.	The application can run smoothly in a low level device	1	2	3	4	5

What can be changed, removed or added based on system performance?

Questionnaire Page 6

PART E: REQUEST FOR RESPONDENT'S FURTHER INVOLVEMENT (OPTIONAL)

Phase II of this study, will involve development of an Umobitalk application for migrants and other non-Sesotho speakers. Should you be interested to participate, please you are requested to provide your contact details below.

Full Name:	
ID Number:	
Email:	
Your	
Phone Number:	