MACHINE NATURAL LANGUAGE TRANSLATION USING WIKIPEDIA AS A PARALLEL CORPUS: A FOCUS ON SWAHILI

BY

ARTHUR BULIVA

UNITED STATES INTERNATIONAL UNIVERSITY - AFRICA

SUMMER 2017
MACHINE NATURAL LANGUAGE TRANSLATION USING WIKIPEDIA AS A PARALLEL CORPUS: A FOCUS ON SWAHILI

BY

ARTHUR BULIVA

A Project Report Submitted to the School of Science and Technology in Partial Fulfillment of the Requirement for the Degree of Master of Science in Information Systems and Technology

UNITED STATES INTERNATIONAL UNIVERSITY - AFRICA

SUMMER 2017
Declaration

I, the undersigned, declare that this is my original work and has not been submitted to any other college, institution or university other than the United States International University in Nairobi for academic credit.

Signed: ___________________________ Date: ___________________________
Arthur Buliva (ID 645381)

This project has been presented for examination with my approval as the appointed supervisor.

Signed: ___________________________ Date: ___________________________
Leah Mutanu

Signed: ___________________________ Date: ___________________________
Dean, School of Science and Technology
Copyright

All Rights Reserved. No part of this dissertation report may be reproduced in any form or by any means, electronic or mechanical, including photocopying, recording, or by any information storage and retrieval system, without prior permission of United States International University - Africa or the author.

Copyright © 2017 Arthur Buliva.
Abstract

The government of Kenya has undertaken an ambitious project to equip children with laptops and tablets for the purposes of facilitating electronic based learning. This initiative can only bear fruit provided that there is content relevant to the studies being undertaken. Many Kenyans learn English as a second language. Swahili or other African languages is the mother tongue. Therefore, with content in Swahili, a better and deeper understanding of subject matter takes place. Much of the academic content already exists albeit in English. Therefore, translating this content is the most practical method of getting the content in Swahili. This is especially so since the content is not necessarily new, but just needs to be interpreted.

There already exist machine translation engines, such as Microsoft Translator and Google Translate, which aim to make this task easier. However, African languages are generally under-represented in these engines. The translation results they produce are comparatively inaccurate when it comes to translating content to African languages. They are even more inaccurate when translating academic type of content. This can largely be attributed to the source of data used to train the translation engines. Many machine translation engines make use of corpora made up of phrases that are found in every day speech, into which academic terms are not adequately incorporated.

Wikipedia, an on-line crowd sourced encyclopedia, offers very good sources of data for purposes of translation works. This study has shown that using Wikipedia as a corpus can provide a viable source of data for academic related translations and specifically so when it comes to African languages.

Therefore, this project modeled an English to Swahili translation engine that uses Wikipedia as a source of translation corpus data. As an emphasis, this study did not set out to create yet another translation engine altogether, but to just improve on, and complement, a small aspect of the current existing engines. The approach that was used was to compare same language articles in Wikipedia and build a parallel corpus which is then used to create a translation database. It is worth noting that Wikipedia on its own cannot provide a comprehensive data set for
any machine translation engine. As proof of concept this model shows English to Swahili translations and presents preliminary results here. Indeed, further work is required for more accurate output alignment and combining the output to ensure fluency and accuracy.

This study was further motivated by the directive of the Communications Authority of Kenya that aims towards having at least 60% of the media content being local. This content therefore needs to be translated into local languages for presentation purposes. The study proposes a solution that can be scaled to learn and translate other local languages.

Finally it is worth noting that Kenya, like many other developing countries, imports numerous products from foreign countries. Many of these products have their labels and instructions written in these foreign languages, more-so English. This poses a potential threat to consumers who do not understand these languages for example in the case of medical drugs.
Acknowledgement

A special thanks goes to Leah Mutanu for the patience and endless guidance. I couldn’t have done it without you. Thanks Paul Warambo of Translators Without Borders for putting up with my endless demands to review translations. Thanks Lori Thicke of Translators Without Borders for introducing me to people who could review the French and Spanish translations I needed right from the beginning. Thank you Maite Suengas from the language department at the United Nations Office at Nairobi for all the help and encouragement with the French. I always secretly think you should have just been a Kenyan from the start. Thanks Martin Benjamin for fanning the spark that was my interest in languages into a roaring flame. Thank you Paa Kwesi Imbeah of kasahorow.org for publishing my first Swahili For Beginners book. You have no idea how much your input has helped me to get to this point. Thanks mum for putting good pressure on me to proceed to masters’ level in my education. It is paying off more than you can imagine. Thanks Kamusi Project. And last but definitely not least, thank you Wikipedia community.
Dedication

For Jerry Sinange, again. More than ever.
# Contents

Abstract iv  
Acknowledgement vi  
Dedication vii  

1 Introduction 1  
1.1 Background of the Problem 2  
1.2 Statement of the Problem 3  
1.3 General Objective 4  
1.4 Specific Objectives 4  
1.5 Rationale of the Study 4  
1.6 Scope of the Study 6  
1.7 Definition of Terms 6  
1.8 Chapter Summary 7  

2 Literature Review 8  
2.1 Introduction 8  
2.2 State of Machine Translation Engines 8  
2.2.1 General Status of MT Engines 8  
2.2.2 Status of MT Regarding African Languages 11  
2.2.3 Wikipedia as a Parallel Corpus 11  
2.2.4 Examples of Machine Translation Systems 15  
2.3 Methods & Techniques of Developing Machine Translation Engines 18  
2.3.1 Rule-based Methods 20  
2.3.2 Statistical Methods 21  
2.3.3 Example-based Methods 24  
2.4 Evaluating and Testing Machine Translation Engines 25
5 Results and Findings

5.1 Introduction.......................................................................................................................... 53

5.2 Status of current Machine Translation Engines................................................................. 53

5.2.1 Accuracy of Translation Engines for Common Language Usage.............................. 53

5.2.2 Status of Translation Engines in Academic Usage ...................................................... 56

5.3 Implementation Results of Custom Machine Translation Engine ................................... 57

5.3.1 Moses Statistical Engine ............................................................................................... 57

5.3.2 Custom Translation Engine using Java ....................................................................... 60

5.4 Evaluation of Custom Translation Engine ........................................................................ 69

5.4.1 Evaluation of Custom Translation Engine in Common Language Usage ...................... 69

5.4.2 Evaluation of Custom Translation Engine in Academic Usage .................................. 70

5.5 Chapter Summary.................................................................................................................. 71

6 Discussions, Conclusions and Recommendations

6.1 Introduction.......................................................................................................................... 72

6.2 Summary............................................................................................................................. 72

6.3 Discussions.......................................................................................................................... 73

6.3.1 Status of Existing MT engines .................................................................................... 73

6.3.2 Development of MT engine ....................................................................................... 76

6.3.3 Accuracy of Custom MT Engine .................................................................................. 79

6.4 Conclusions........................................................................................................................ 82

6.5 Recommendations.............................................................................................................. 84

6.5.1 Current MT Engines................................................................................................... 84

6.5.2 Development of a Custom MT Engine ....................................................................... 84

6.5.3 Evaluation of MT Engines.......................................................................................... 85

References................................................................................................................................... 86

Appendix A Part-of-speech tags used in the Penn Treebank Project .............................. 92
Appendix B  Common Phrases Used in Testing  
Google and Microsoft Translation Engines 94

Appendix C  Academic Phrases used to test Translation Engines 95

Appendix D  Source Code Listings 97
  D.1  Downloading Wikipedia Dumps using GNU Wget .......................... 97
  D.2  Extracting Parallel Titles from Wikipedia Dumps ........................ 97
  D.3  Wikipedia Content Extraction Using JSoup Java Library ................. 97
  D.4  Downloading Wikipedia Titles Using Wikipedia API ....................... 97
  D.5  Saving Wikipedia Pages into MongoDB ..................................... 98
  D.6  Indexing the MongoDB Collection ........................................... 98
List of Tables

2.1 Example of a Bilingual Parallel Corpus - Adapted from (Benjamin & Radetzky, 2014)) ................................................................. 9
2.2 EBMT Working Mechanism - Adapted from (Satanjeev Banerjee, 2008) 24
2.3 RTE Working Mechanism, as adapted from Google Translate.................. 26
6.1 Word Alignment Example ........................................................................................................ 77
6.2 Swahili Verb Breakdown ........................................................................................................... 78
# List of Figures

2.1 Interpreted French Translations of “Light” (Benjamin & Radetzky, 2014) ......................................................... 10
2.2 First paragraph of English Wikipedia entry for “Paper” – adapted from (Wikipedia, 2016) ............................................................... 12
2.3 First paragraph of Swahili Wikipedia entry for “Karatasi” – adapted from (Wikipedia, 2016) ............................................................... 12
2.4 Structure of a Wikipedia Page (Wikipedia, 2016) ................................................................................. 13
2.5 Kamusi Project Home Page (The Kamusi Project, 2016) .............................................................. 16
2.6 African Languages on Microsoft (Bing) Translator – adapted from (Microsoft Translator, 2016) ........................................................................ 16
2.7 African Languages on Google Translate – adapted from (Google Translate, 2016) ......................................................................... 18
2.8 METEOR metric matching of output against reference text (Satanjeev Banerjee, 2008) ........................................................................ 31
4.1 Workflow Diagram Describing Fetching Data From Wikipedia ........................................ 42
4.2 Sequence Diagram Describing Fetching Data From Wikipedia ........................................ 43
4.3 Database Structure .......................................................................................................................... 43
4.4 Physical Implementation of the MongoDB Database ........................................................................... 44
4.5 Workflow of the Translation Algorithm .............................................................................................. 45
4.6 Wikipedia Data Fetching Algorithm Class Diagram ........................................................................... 45
4.7 Class Diagram of the Translation Algorithm ........................................................................................... 46
4.8 The System Architecture ................................................................................................................... 47
4.9 GNU Wget Downloading Swahili Articles from Wikipedia ........................................................................ 48
4.10 Extracting Parallel Titles From Wikipedia Dumps ................................................................................... 49
4.11 Source Screenshot of Wikipedia Data Fetcher ........................................................................................ 50
4.12 Snippet of the Custom Translation Algorithm ...................................................................................... 52
5.1 Accuracy of English to Spanish Translations on Microsoft and Google Translation Engines .......................................................... 54
5.2 Accuracy of English to French Translations on Microsoft and Google Translation Engines .......................................................... 54
5.3 Accuracy of English to Afrikaans Translations on Microsoft and Google Translation Engines ......................................................... 55
5.4 Accuracy of English to Swahili Translations on Microsoft and Google Translation Engines .......................................................... 56
5.5 Accuracy of English to Swahili Translations on Microsoft and Google Translation Engines, for Academic Phrases........................... 57
5.6 CPU usage during the tuning process of Moses Engine ......................... 59
5.7 Moses engine running and doing a few translations which have been highlighted to show emphasis ................................................. 60
5.8 Swahili Article for Arabic Numeral 1 showing anomalies ...................... 61
5.9 Resulting Parallel Corpus File after Extracting Parallel Titles From Wikipedia Dumps ................................................................. 62
5.10 Data Being Fetched from Wikipedia ............................................. 63
5.11 MongoDB Database Displaying Data .............................................. 63
5.12 MongoDB Database Displaying Indexes .......................................... 64
5.13 Getting the translation Request ..................................................... 65
5.14 Look-up in MongoDB and chunking phrases using Apache OpenNLP 66
5.15 Translation Results ........................................................................ 67
5.16 Translation Results for “Kenya is a democratic country” ..................... 68
5.17 Fallback Chunking for “Kenya is a democratic country” ...................... 68
5.18 Fallback Translation for “Kenya is a democratic country” .................... 69
5.19 Translation Results of Common Phrases .......................................... 70
5.20 Accuracy of English to Swahili Translations on Microsoft, Google and the Custom Translation Engines With Respect to Academic Terms . . 71
6.1 Interpreted Translation of “This is a sink” on Google Translate .............. 74
6.2 Interpreted Translation of “We had a gay time” on Microsoft (Bing) Translate ...................................................................................... 75
6.3 Interpreted Translation of “Huyu ni Somo” on Microsoft (Bing)
Translate.................................................................................................................. 75

6.4 Translation of “Jupiter is a planet from outer space” on Google Translate 81

6.5 Translation of “Jupiter is a planet from outer space” on Microsoft
Translator ............................................................................................................ 81

6.6 Translation of “Jupiter is a planet from outer space” on the Custom
Engine ................................................................................................................. 82
CHAPTER ONE

1.0 INTRODUCTION

Translation is the process of getting content from one language to another. This may be performed either manually by one or more translators, or automatically by a machine. The manual method is usually impractical because of constraints such as costs, and the resulting translations may be inconsistent. Computers can work tirelessly and continuously and the comparative cost is cheaper than humans doing the job manually.

Computer translation programs have been traditionally been built around phrase-based statistical machine translation (BBC, 2016b). Their working mechanism involves the analysis of already translated texts for example academic papers. After doing a parallel analysis, statistical probabilities are used in order to pick the most appropriate translation. This implies that the accuracy and effectiveness of the eventual translations heavily depend on the quantity and quality of the original texts.

An interesting point to note is word-for-word method of translation usually breaks the context of the translation especially when it comes to African languages. For example, the English phrase “A black cat” (English) directly translates to “Paka mweweusi” in Swahili. A bilingual Swahili and English speaker can tell that the translation is literally “reversed” and a reverse translation back to English yields “cat black”. Furthermore, a sentence like “The United States International University” can get translated to “United”, “States”, “International” and “University” as individual words, yet the whole phrase needs to be translated in its entirety as if it is just one word.

Because current statistical machine translators can get clunky and inaccurate, Alan Packer, director of engineering language technology at Facebook, said recently that the current method of statistical machine translation was reaching “the end of its natural life” (BBC, 2016b). Translators are moving from just words to actually
learning metaphors and the meaning behind the language – context.

Rules can be programmed into machine translation engines in order to take care of such issues. However, the rule bases can quickly grow to be extremely large and any slight changes in the rules will need a reprogramming of the software. Even though the rules can be manifested, a problem of context still lies within the rules. If an input such as “Light” comes in, what is also needed to be known is what type of light is being referred to; it could be light (energy), light (the object on a ceiling that gets switched on), light (as a feather) or light (orange is a lighter shade of red than is maroon)? With this example, it can be seen that indeed context is extremely important when doing translations..

1.1 Background of the Problem

Many of the current statistical machine based translation engines require human based parallel text input in order to process translations. This is often the weak point in translation engines since creating parallel texts takes time and people get easily bored of the mundane and repetitive task.

The field of academics is just but a subset of general language usage. With the Government of Kenya having passed a regulation that requires local content on local media, as well as the digital e-learning being introduced to children at an early age in Kenya, this study offers a good opportunity to bring native languages into academic usage. The terms and phrases used in academics can now accurately and efficiently be translated to Swahili.

Wikipedia has a lot of text and data in many different languages. Many of these texts are a form of translation between the languages. Theoretically, it should be possible to create a true parallel corpus for language pairs using this vast repository of multilingual texts, and then use this corpus to improve machine translation.

In practice, there are different ways to build any language corpus. Incorporating automated mechanisms to update the corpus will ensure continuous improvement
of the corpus. Different repositories such as newspaper websites, bilingual news websites, gossip sites and personal blogs can be used to incorporate contextual aspects to machine translation engines. Using these sources adds an extra leverage to the corpus by tapping into the everyday knowledge of the people who fluently speak the language of interest.

The most fundamental thing to note is that this study did not set out to create yet another translation engine. The main aim was to improve on a small aspect of the current existing engines, and that is the machine translation of academic content.

1.2 Statement of the Problem

African languages are resource scarce in the machine translation front (Wagacha, Pauw & Getao, 2006; Benjamin & Radetzky, 2014; Benjamin, 2014; Jesús, Jordi, Francisco & Jaime, 2008). This means that the current methods and rules of machine translation are not working well when it comes to African languages (Benjamin & Radetzky, 2014). Extending the rules used in European languages and trying to adapt them for African languages does not work either, owing to the structural difference between European and African languages. Besides, African languages use verbs with a complex conjugation rule set that cannot be deployed within the rule sets of European languages.

Academics is just but one of the facets of general language usage. Through this study, it was observed that the translation of African languages such as Swahili in academics is even more underrepresented. When it comes to Academic usage, African languages are dying because they are not being as widely used as is English. Indeed, English is the most commonly used language of instructions.

One of the goals of this project was to explore the possibility of creating a translation engine which applies machine learning and data mining techniques using data obtained from Wikipedia. The language in focus of the translation system was Swahili – an African language with approximately one-hundred million speakers spread over Kenya, Mozambique, Somalia and Tanzania ( Ethnologue, 2016).
1.3 General Objective

This study made use of freely available parallel corpus data from Wikipedia to develop a machine translation engine that aimed to provide academic translations from English to Swahili languages.

1.4 Specific Objectives

This research set out address the gap of inaccurate English to Swahili academic matter translations through the following research objectives:

1. To investigate the status of current popular machine translation engines with respect to Swahili as an African language
2. To create a machine translation tool that uses Wikipedia as its parallel corpus.
3. To evaluate the tool in order to determine how it fairs with English – Swahili translation.

1.5 Rationale of the Study

In the increasingly digital age, the world is becoming smaller and smaller. Products and services are getting to a global audience faster than ever before. The rate of learning of the skills of speaking different languages cannot keep up with this speed of globalization. As such, it is far easier to label products in the language of the target market than it is to teach the entire market region how to speak a new language whenever a new product gets launched. Using local languages means that the users of a particular product can then relate better to the products as it makes them feel that they have been adequately considered.

This project is aimed at helping in improving localization and internationalization works, and the focus is in African languages, since most popular MT engines
have focused on European languages and left the African languages quite underrepresented (Hyman, 2008; Okpor, 2014).

The current laptop project being undertaken by the government of Kenya is an interesting area to adapt this study. Content creators are a key group in this project, now that a lot of the content that will be used in the laptop project has to be localized to the Kenyan context (BBC, 2016a).

Another area that this project comes in handy is in the requirement by the Communication Authority of Kenya that at least 60% of the content that Kenyan media companies (television and radio) has to be local in nature (Communications Authority of Kenya, 2015). This directive gives a good platform for both local and international content providers to meet this directive.

Consider a scenario in which all health information worldwide is given in only English. Despite English being an official language in Kenya, the local understanding of it is limited (Translators Without Borders, 2015). This means that the health information provided in English is not being understood effectively. If the same information is available in Swahili, there will be a significant increase in understanding of the health data. Therefore, Translators Without Borders concludes, English is not the most suitable medium for the transfer of important information among representatives of these communities. Swahili seems to function better as a language of communication.

Consumers purchase more than just a product. In addition to the product, there is the experience that the product gives them, such as the packaging and the after-sales services (Oloko, 2014). Many products take advantage of localization for them to appeal to the market. Safaricom, a Kenyan telecommunications company, has shown that having services in the local context offers a significant advantage. For Safaricom, this has been made possible by using localized names of services for example *sambaza*, *skiza* and *M-PESA*. This makes the users feel that they are using a local product and consequently brand loyalty is achieved (Beauchesne, Dorion & Griggs, 2014).

Another reason for localizing content is that even though a certain large region may
generally speak the same language, the smaller areas within the region may make use of words in such a way that their meaning is different. For instance, even though both Kenya and Tanzania speak Swahili, there are subtle differences when it comes to the meaning of specific words in context. Cultural symbols such as flags may also differ, and when giving examples of the translations requested, it does well to determine the locale context in the example.

Localization is the process of conforming a product or service to a particular language or region. It involves modifying a product in order that it fits the culture in which the product or service will be used. Localization not only ensures that the language is right, but it also goes hand in hand with the context of usage. It involves taking a product and making it appropriate linguistically and culturally to the target region and and language where the product will be sold and used (Sanday, 2014).

1.6 Scope of the Study

This study investigated the methods of doing sentence extraction from Wikipedia entries in order to create a parallel language corpus, and in particular, the English and Swahili articles. Theoretically, this could be extrapolated so that the results of one language pair may be extended to cover any other language pairs.

The research also focused on mining general parallel language articles and not articles within just one particular genre. The data that was extracted from the Wikipedia articles was not just a blind dump of the contents of the entire page of the article, but rather just the first paragraph of the articles. This ensured that the process maximized on the possibility of the corpus being a parallel or a comparable one.

1.7 Definition of Terms

Corpus – A collection of texts (Nakayama, Hara & Nishio, 2008; Lenci, Zampolli & Calzolari, 2015)
**Parallel Corpus** – A collection of texts aligned side by side such that one is a translation of the other (Nakayama et al., 2008; Lenci et al., 2015)

**Translation** – the process of translating words or text from one language into another (Reese, 2015)

**Machine Translation** – Using an automated means of obtaining translation without human intervention (Reese, 2015)

**Machine learning** – A method of using artificial intelligence (AI) in order to make computers be able to learn new things without being explicitly programmed (Benjamin & Radetzky, 2014; Reese, 2015).

**Training set** – A data set that is used to obtain potential relationships (Benjamin & Radetzky, 2014)

### 1.8 Chapter Summary

This chapter has introduced basic terms that are used in the area of machine translation. The chapter has also shown the need for content translation to African languages now that interest in machine translation is just beginning to take root in African languages, even though it is already established in European languages. This chapter has also highlighted the need for ensuring that translations are performed in context as well as the importance of the translations in academic usage.

The next chapter outlines the various ways in which current machine translation engines perform when it comes to machine translation. The chapter will also delve into the various ways of developing machine translation engines, as well as on the different methods through which machine translation engines are evaluated.
CHAPTER TWO

2.0 LITERATURE REVIEW

2.1 Introduction

This chapter examines the current status of machine translation engines and explores the techniques used to develop and evaluate the translation engines. A critical summary of published research literature relevant to the topic under consideration is outlined. The chapter creates familiarity with current thinking and research on Wikipedia as a corpus for African languages, and justifies this research into this previously understudied area.

2.2 State of Machine Translation Engines

2.2.1 General Status of MT Engines

Machine translation (MT) is the attempt to automate all – or part – of the process of translating linguistic data from one natural human language to another. It is considered to be a sub-field of computational linguistics. Therefore, MT is defined as the use of computer software to translate text or speech from one natural language to another (Abiola O.B & A, 2015; Zens, 2008).

There are approximately 7000 languages that are spoken in the world today (Ethnologue, 2016). Out of these languages, there exists excellent sources of machine readable data as well as ICT resources for English, French, Italian, German and Spanish (the FIGS) (Benjamin & Radetzky, 2014). This leaves us with a lot of languages that are either dead or dying, meaning that they are not being passed on to the next generation.

In order for machines to be able to do translations, they need data in the language
from which the translation is needed, and also data in the language to which the translation will be produced. These data sets are known as the source and the target languages respectively. Machine translation engines make use of corpus (plural corpora) in order to do translations. In the field of languages and linguistics, a corpus is simply a collection of texts. A corpus may be based on a single language, in which case it is called a monolingual corpus. It may also be based on multiple languages so that it is known as a multilingual corpus. These corpus texts may be aligned such that the text in one language matches that of another language which now becomes a parallel corpus or aligned parallel corpus (Nakayama et al., 2008; Lenci et al., 2015). This can be illustrated as shown in Table 2.2.1.

Table 2.1: Example of a Bilingual Parallel Corpus - Adapted from (Benjamin & Radetzky, 2014))

<table>
<thead>
<tr>
<th>English</th>
<th>Swahili</th>
</tr>
</thead>
<tbody>
<tr>
<td>That is a black cat</td>
<td>Yule ni paka mweusi</td>
</tr>
<tr>
<td>A big chair!</td>
<td>Kiti kikubwa!</td>
</tr>
</tbody>
</table>

The United Nations has six official languages that are considered to be important to the work of the organization. These languages are Arabic, Chinese, English, French, Russian and Spanish (United Nations, 2016). Every official communication and document produced by the UN is translated into all these six languages. Indeed, this is a rich source of parallel corpus data, but with a downside that the data is biased on political happenings and not general knowledge.

Another rich source of parallel corpus data is religious texts such as the bible and the quran. The bible has been translated into 544 languages and this is a very good source of parallel corpus data (Wycliffe Global Alliance, 2015). The disadvantage of using the bible or quran as a primary source of corpus data is that the texts are nothing more beyond religious in nature, and this is definitely not representative of the broader aspects of human knowledge.
The efforts of doing machine translation depends on the context of the words. This means that the machines need much more than just the original word in order to make sense of the source data. There is a need to look into various aspects such as the roots of the word, misspellings, synonyms and the likes. Consequently, translation engines have to be "alive", continuously analyzing contexts and applying new meanings to literature in order to generate sensible translations (Benjamin & Radetzky, 2014).

Context is also important because different groups of people, cultures and semantics apply different meanings to the same set of words. In African languages, none of which is within the English + FIGS category, context is even more important now that they are underrepresented in modern popular machine translation engines.

In order to show how context is important, consider the concept of light as an example, it can be interpreted differently and consequently translated depending on the interpretation of the machine as illustrated in Figure 2.1.

Figure 2.1: Interpreted French Translations of “Light” (Benjamin & Radetzky, 2014)
2.2.2 Status of MT Regarding African Languages

One of the most important aspects of machine translation is the ability to do Named Entity Recognition (NER) (Shah, Lin, Gershman & Frederking, 2010). NER gives a translation engine the ability to distinguish real world objects, thereby placing them in already defined categories such as names of people, organizations and their abbreviations, places and locations, etc.

NER already exists in many on-line MT engines but none exists for Swahili (Shah et al., 2010). In order to see its importance, a study that was previously conducted shows that when it comes to medical translation, Swahili scored lowest with only 10% correct, while Portuguese scored highest at 90% (Patil & Davies, 2014; Translators Without Borders, 2015). These poor results were attributed to the translation engines not being able to extract the concepts of people or places in the Swahili sentences.

Therefore, with results showing that Swahili translation is wanting in the day-to-day life as well as the critically important medical field, there exists an opportunity to contribute to the field of academics by developing a translation engine which is relevant for the Kenyan Swahili context.

2.2.3 Wikipedia as a Parallel Corpus

The amount of collaboratively translated and lexical texts on the Internet has been steadily increasing. By making use of contributors the world over, Wikipedia has been successful in creating a knowledge base that has a rich data set from diverse people and cultures. Since it covers a broad area of human knowledge, Wikipedia is an excellent source of data in many areas of languages and Artificial Intelligence (Nakayama et al., 2008).

Many of the cross-language Wikipedia articles are not a true parallel corpus, but are a comparable corpus. This is because many of the translations do not preserve the exact order of the sentences across articles, or some texts have been omitted between
language articles (Potthast, Stein & Anderka, 2008; Lenci et al., 2015).

Many Wikipedia articles begin by giving a definition of the subject matter. This can be shown by viewing the English definition entry for Paper as illustrated in Figure 2.2. The corresponding Swahili article, Karatasi, conforms to this standard by also offering the definition at the start of the article. This is illustrated in Figure 2.3.

Figure 2.2: First paragraph of English Wikipedia entry for “Paper” – adapted from (Wikipedia, 2016)

Figure 2.3: First paragraph of Swahili Wikipedia entry for “Karatasi” – adapted from (Wikipedia, 2016)

These definition texts are not exact parallel translations, but they are a comparable translations that can be used to create a simple translation engine. A Wikipedia page is structured as illustrated in Figure 2.4, which shows the initial paragraph as a definition of the article..

Monolingual aspects of Wikipedia have been extensively studied and they show that mining Wikipedia deserves further study (Jesús et al., 2008). There are important characteristics of Wikipedia make it very viable for web corpus creation. These include:

(a) Dense link structure

Wikipedia articles are interlinked, with one article containing references to other articles or external links. One page may link to another which in turn links back
to the original page. This shows that there is a possibility of developing a thesaurus in order to improve the synonyms (Nakayama & Hara, 2007), since if two articles link back to each other then they may be somehow related. This interlinking is what is referred to as being dense. This enables the integration of statistics of how many pages link to a particular page, and therefore rank the reliability of the page. A page is considered to be more reliable with an increase in the number of articles linking to the page.

The deep linking of articles is a feature that is particularly useful in determining relationships between articles. These relationships give the ability to establish a thesaurus. A thesaurus shows how words are related semantically (Nakayama et al., 2008). Human effort is the best way to create a thesaurus, but it is difficult to maintain such large data sets. New concepts are usually not captured and issues of ambiguity may still exist because of semantics and differences in spellings.

Based on the dense link structure, it is therefore possible to create a context-based corpus by extracting data from only one kind of genre, say Science or Mathematics. This enables the creation of a targeted corpus that omits articles that are not of
interest.

(b) Concept identification

Wikipedia has the characteristic of having links to ambiguous words or phrases included at the top of the page of any article that may have ambiguity. For example, the concept of “Windows” may have several meanings; There is the concept of Windows the operating system, Windows the 1980s film, not forgetting Windows the opening on a wall that lets in light. Each of these concepts, has their own unique Wikipedia link that allows for analyzing the concept of Windows while avoiding ambiguity (Nakayama et al., 2008). They all contain a link to clarify any ambiguity, https://en.wikipedia.org/wiki/Windows_(disambiguation).

This is what is known as concept identification. Ambiguity is not a problem for humans but poses quite a challenge to machine translation engines (Nakayama et al., 2008).

(c) Wide topic coverage

Wikipedia has the advantage of topic coverage (Nakayama et al., 2008). This means that new concepts being introduced get good and detailed coverage way ahead of paper-based corpora. In addition to text, Wikipedia also features multimedia images, videos and sounds all of which can be mined as part of a different project altogether.

Because different individual users with different knowledge areas can edit Wikipedia, it therefore follows that there is an advantage of broad topic coverage. There are various language options and collaborative modifications which improve the quality and quantity of the articles, resulting in a rich corpus.
(d) Third-party links

Just like any other web page, Wikipedia also links to external pages such as news websites or university articles. This is part of the requirements of Wikipedia editing that aims to make sure that all entries are cited and appropriately referenced. By also analyzing these external hyper-links, more information such as localization contexts can be extracted from an article. With other pages linking back to the Wikipedia articles, a mechanism of doing page ranking can be developed so that more attention is given to the most referenced page as a sort of weighted average (Viégas, Wattenberg & McKeon, 2007).

2.2.4 Examples of Machine Translation Systems

To date, several machine translation systems have been developed. Some of the engines include:

(a) The Kamusi Project

Using the same concepts as Wikipedia, The Kamusi Project makes use of on-line collaborator communities comprising volunteers, experts and authorities in the Swahili language in order to generate content (The Kamusi Project, 2016). Kamusi crowd-sources translations from the general public and holds them in an on-line database against which searches may be performed.

The Kamusi Project describes itself as being more of a dictionary rather than a machine translation engine (Benjamin, 2016). Indeed the word kamusi is Swahili for dictionary. The Kamusi Project offers excellent resources for one-word translation. It, however, was not designed as a phrase translator through which extraction of context is necessary for obtaining contextual meaning of words and phrases. Figure 2.5 shows the home page of Kamusi Project.
(b) Microsoft (Bing) Translator

Offering only two African languages on its menu as shown in Figure 2.6, Microsoft Translator indeed shows the underresourced nature of African languages in current machine translation software systems.

Figure 2.5: Kamusi Project Home Page (The Kamusi Project, 2016)

Figure 2.6: African Languages on Microsoft (Bing) Translator – adapted from (Microsoft Translator, 2016)
Microsoft collaborated with Translators Without Borders on the Swahili Translator “as a means to professionalize and standardize Swahili translations, especially in the areas of health and crisis relief, as part of the Words of Relief crisis relief translation network.” (Microsoft, 2016a). Translators Without Borders are a network of professional translators that collaboratively work on translation issues related to health, nutrition and education (Translators Without Borders, 2015).

(c) Google Translate

Arguably, this the most popular machine translation engine by virtue of usage. Developed by Google Inc., this machine translation engine incorporates a lot of languages spoken in the world. Google Translate is an acceptable way of getting clued in on what the source message is, but it should not be used in doing professional business translations (Ignacio, 2010; Benjamin, 2014). Google Translate has more than 200 million active users each month, and their chief scientist asserts that they provide “most of the translation on the planet” (Benjamin, 2014).

Google Translate works by analyzing statistical text corpora. These translated texts come from books, organizations like the UN and websites from all around the world (Och, 2012; Koehn, 2007). Even though Wikipedia could be part of the translated texts, this has not been explicitly confirmed nor denied by Google.

Offering a broader menu than Microsoft Translator as shown in Figure 2.7, Google Translate has a lot more African language coverage as compared to Microsoft Translator. However, just because there is a lot more options does not make the results any more accurate. Google still lacks enough African language data to enable it to give more accurate results when it comes to various aspects of African languages (Benjamin & Radetzky, 2014).
2.3 Methods & Techniques of Developing Machine Translation Engines

Translation machines require data in order to generate the translations. Much of the data already exists but it is in formats that cannot be used readily by computers, for example in printed format. Optical character recognition will not effectively convert the data in the printed words to computer text. Besides, such texts are usually heavily copyright protected, thereby making it a crime to make use of them without explicit permissions from the owners. The owners of the data may be located and are even known, but they are usually very reluctant to share (Benjamin & Radetzky, 2014).

Other translation data can be obtained from already existing texts such as
newspapers, magazines. Experts such as lexicographers or linguistic lecturers can also be sources of data and so is crowd-sourcing from Wikipedia and online forums. Considering all these sources, Wikipedia cuts across all of them by being able to offer all three sources at the same time (Nakayama et al., 2008). This is because Wikipedia, being wiki-based, offers its platform for anyone to contribute to articles – from subject matter experts to magazine editors all the way to the general public.

The translation data may be obtained either manually using human effort or by automation using machines that do the acquisition (Christian, 2010). The problem with existing data on African languages is that it is hard to come by, and it is noisy wherever it is found. Noisy means that it contains a lot of foreign words, or it is just simply inconsistent (Nothman, Ringland, Radford, Murphy & Curran, 2013).

Consequently, human intervention is needed in order to not only make the data more consistent, but to also proof-read it in order to remove ambiguity and noise (Wagacha et al., 2006). Working with humans poses several difficulties. The first hurdle lies in identifying the people who will work on the language. Experts in languages do not necessarily have the necessary skills needed in converting the data into a machine-readable format (Benjamin & Radetzky, 2014). Additionally, most language experts are usually employed somewhere else such as universities, and thus they may not have the time to keep a corpus up-to-date.

There is also the need for the people to be at least bilingual, in order to be able to proof-read translations for accuracy. Such expertise is expensive and funding for such an exercise may be unavailable. Wikipedia being free provides an excellent starting point in the process of creating a language corpus because the data is accessible free of charge (Christian, 2010).

As pointed out earlier in section 2.2.1, human language is dynamic and undergoes changes. Therefore, any machine translation engine has to be “alive”, continuously analyzing and searching for new meanings in the source literature in order to generate sensible translations. The most straightforward way is to indeed automate the mining of the texts instead of relying on human labour.

Indeed, many MT engines rely on roughly three ways of translating:
2.3.1 Rule-based Methods

Because there may not be enough corpora data that can be used in a translation engine, rule sets are generated that are able to translate certain simple words from the source to the target languages (EuroMatrix, 2012). Rule-based translation engines perform translation using extensive lexicons with morphological, syntactic and semantic information, and large sets of manually compiled rules, making them very labour-intensive to develop and maintain (Guy De Pauw, 2010). Rule-based machine translation engines may also employ the use of a word dictionary with entries of source and target languages. What this implies is that incoming words get translated by making use of a dictionary.

This type of approach is excellent when developing an engine for one particular language pair. The native speakers are able to add rules as needs change or new requirements arise. However, it is difficult to port such a particular system to other different language pairs. In order to add new rules, sometimes a recompilation of the algorithm may be necessary. Since language is dynamic, these rules need to be reviewed often and frequently in order to accommodate new rules or new ways of expressing natural languages (EuroMatrix, 2012).

Rule based methods require an analysis of a word or a particular phrase and comparing the phrase to a set of rules. This overhead of computational power means that some words which may be obvious to a human translator must be added into the rule dictionary because they may be foreign to the translation engine. Pattern matching and rewriting of rules makes this method of translation inefficient (Reese, 2015; Guy De Pauw, 2010).

A rule set is usually composed of a series of comparison statements in which the algorithm goes through the statements one by one and considering whether the
current request matches the rule. The rule set can easily grow to be even thousands of “if – else” statements. Sometimes it is the very last rule that matches the condition of translation, meaning that the computer has to go through thousands of lines in order to match just one translation (Reese, 2015).

Further to enormous rule sets, small irregularities in the input can cause the algorithm to go into an infinite loop. For example, a human can easily look beyond “spelinng mitsakes” but the rule based engine, which matches exactly all of the word or nothing, is unable to look beyond the error and try to see a probable solution. Without a properly defined exit condition, the algorithm can end up in a never-ending cycle (EuroMatrix, 2012; Reese, 2015; Guy De Pauw, 2010).

### 2.3.2 Statistical Methods

Statistical machine translation engines work by performing statistical analysis of bilingual text corpora, i.e. parallel corpora. This method attempts to determine the most probable translation of a sentence or phrase by analyzing all the places where the text or phrase occurs (Calzolari, Lenci & Zampolli, 2009). By default, these engines imagine that the source text has been encoded in some undefined way and then they try to translate it by decoding it first and reconstructing the necessary translation.

Statistical methods of translation were first developed by IBM in early 1990s (Reese, 2015; Calzolari et al., 2009). The algorithm works by analyzing each word at a time in a parallel corpus in the source language and finding its occurrence in every sentence in the source corpus. The target language is then analyzed to find the most common word that occurs whenever the word in the source occurs. This, therefore, is most likely the target word. A big criticism of the statistical method of translation is that it works well for languages that are structurally similar but fails whenever this is not the case (EuroMatrix, 2012; Jesús et al., 2008; Guy De Pauw, 2010).

Initially, many statistical translation systems were word based (Josef & Ney, 2003; Klementiev & Ann Irvine, 2009). Recent studies have enabled these systems to be
made better by having phrase-based models.

(a) Word-based Models

Many of the older statistical translation engines made use of this model. The model examines each of the words in a sentence independently, whilst ignoring the bordering words and any relations between them.

While this works perfectly well for single words, word-based models are not appropriate for compound words and phrases. Sometimes, one word in the source language can map to multiple words in the target language, or many words in the source mapping to only one word and this can be easily missed. Many natural languages depend on the surrounding words in order to derive meaning and explain the context of a particular word (Zens, 2008).

A popular example of a word-based statistical translation model is the GIZA++ system (Josef & Ney, 2003). GIZA++ was developed as a statistical word analyzer and is used as part of the translation process on Moses, an open source statistical translation engine (Koehn, 2007).

(b) Phrase-based Models

With time, phrase based models gained popularity over word-based models (Koehn, 2007; Zens, 2008). A phrase is simply a sequence of words. This enabled entire groups of words to be considered in their entirety. Phrase-based models ensure that context is maintained in the translation because words are analyzed in relation to each other.

The idea behind phrase-based models is to segment a given source sentence into phrases, then translate each phrase in turn and finally constructing a translation based on the translated phrases. It is interesting to note that in this model, punctuation marks are usually considered to be whole words by themselves (Zens, 2008; Reese, 2015; Josef & Ney, 2003).
Just as a form of artificial intelligence, statistical translation knowledge is automatically learned from the source data. Consequently, any new rules or grammar alterations are quickly captured and no new programming has to be done unlike as it is when it comes to the rule-based approach.

Taking an application such as a speech-to-text translation system, statistical engines have a pretty good chance of offering accurate results because much of the spoken word is usually natural language. If the database of the corpus used is based on natural language sentences, there is a high chance of scoring pretty accurate translations.

Phrase-based translation models do suffer from a few issues which include:

1. **Sentence Alignment**

   In one language, a particular sentence may have several possible correct translations. Sentence aligning can be achieved by making use of an algorithm called The Gale-Church alignment algorithm (Okpor, 2014).

2. **Statistical Anomalies**

   Real world training sets may replace those that are intended because of an abundance of a particular sentence in the training set. For example, “I need some toothpaste” may be replaced by “I need some colgate” because a particular locale may equate colgate to be any toothpaste of any brand (Okpor, 2014).

3. **Different Order of Words**

   This comes into play especially when it comes to English and question tags. “It doesn’t rain, does it?” “It is hot outside, isn’t it?” (Okpor, 2014). In this particular example, “It does rain” and “It doesn’t rain” may be interpreted as having the same meaning yet each of the sentences is the opposite of the other.
2.3.3 Example-based Methods

Initially proposed and developed by Japanese professor Makoto Nagao in 1984 (Nagao, 1984), example-based translation engines are similar to statistical methods in that they are parallel corpus driven. An Example-Based Machine Translator (EBMT) scans for patterns in both languages and relates them in a translation memory. (Jesús et al., 2008; Calzolari et al., 2009). Simply put, these translation software try to see if or how a source sentence been translated before and translates it the same way. The idea behind this is to reuse already existing knowledge as much as possible. The general working mechanism can be illustrated as shown in Table 2.2.

Table 2.2: EBMT Working Mechanism - Adapted from (Satanjeev Banerjee, 2008)

<table>
<thead>
<tr>
<th align="left">Input: He bought a book on African Geography</th>
</tr>
</thead>
</table>

| Data: |
| He bought a cow – Alinunua ng’ombe. |
| That is a book – Hicho ni kitabu |
| She is studying African Geography – Anasomea Jiografia ya Kiafrika |

<table>
<thead>
<tr>
<th align="left">Output: Alinunua kitabu on Jiografia ya Kiafrika</th>
</tr>
</thead>
</table>

Example-based translation method enjoys an advantage that it is highly probable that a small partial segment of a phrase can be found from a large collection of raw data. In addition to that, if there is a large enough corpus, the resulting examples are very well structured. It however suffers from the problem that the generated text may tend to be incomprehensible and difficult to understand. Also, if the examples don’t exist, then the software system does not have the algorithm to further deconstruct the source in order to search for a translation (Satanjeev Banerjee, 2008).
This can be evidenced in the example in table 2.2 where the word “on” is left untranslated because it is found nowhere in the corpus.

The main difference between example-based and statistical engines are that statistical engines compile the data into a binary file that is accessed every time there is a request, while example based methods hit the corpus each and every time a request is made (Nagao, 1984). Besides that, statistical engines tend to not break up a sentence into manageable chunks. Instead they perform a search for the entire phrase as it occurs in the corpus.

### 2.4 Evaluating and Testing Machine Translation Engines

There are many methods that can be used to test any particular software system and these methods can range from usability to efficiency (Sommerville, 2011). When it comes to the evaluation of machine translation systems, there are generally three methods of evaluation that can be used. These methods are:

#### 2.4.1 Round-trip Evaluation

The idea behind the assessment of machine translation quality through round-trip evaluation (RTE) is to translate from the source language to the given target language, then again translate the output back to the source language possibly using the same engine (Somers, 2005; Aaron L., 2013). This can be illustrated as shown in Table 2.3.

There are several reasons why RTE may not work. First of all, when the translated round trip is bad, it is difficult to tell whether the problem was with the original leg, or with the return trip (Gaspari F., 2006). Somers further points out that the other problem of RTE is the fundamental flaw on the foundation on which RTE is based. Even the best of human translators cannot translate a given sentence back to
Donald Trump was elected as President of the United States of America

Donald Trump alichaguliwa kuwa Rais wa Marekani

Donald Trump was elected president of the United States

its original, word-for-word. Finally, a return journey may result in a translation that is syntactically and grammatically correct but slightly different from the original, and therefore may be flagged as being incorrect (Benjamin & Radetzky, 2014).

On the other hand, RTE needs to be acknowledged that it may and it actually does work. While still considering Table 2.3 as an example, RTE works well in this case. It produces a reasonably understandable paraphrase and while en-route to our final output, the mid translation captures the intended meaning correctly.

The main assumption with RTE is that the machine that is doing the evaluation is knowledgeable enough in both the source and target languages.

### 2.4.2 Human Evaluation

The oldest way of judging the accuracy or quality of a translation is by making use of good old human judgment (Hutchins, 1996; Patil & Davies, 2014). Although using a human to evaluate translations is very much time-consuming, it is the most reliable method of MT evaluation (Christian, 2010). Using specially-trained human judges, the translations are judged based on **intelligibility** – that the translations make sense in terms of being understandable on a scale of 1 – 9, and **fidelity** – the amount of data in the original sentence that was captured in the translation and is measured on a scale of 0 – 9 (Hutchins, 1996). The more the information that the original contained that was not captured in the translation, the lower the system scored (Christian, 2010).

The scores are usually averaged because human judgment is subjective from one
person to another. Other means of evaluating the translations, therefore may be based on the fluency of the output, while others on the adequacy of the data that is obtained in the translation (Benjamin & Radetzky, 2014; Patil & Davies, 2014).

2.4.3 Automatic evaluation

In the context of Automatic Evaluation (AE), certain measurable metrics are defined that are be used to measure the output. Many of the metrics are usually aligned against those given under human evaluations and are designed to correlate with them (Benjamin & Radetzky, 2014; Reese, 2015; Christian, 2010).

Banerjee et al. (2005) highlights five characteristics that any good automatic metric should have: correlation, sensitivity, consistency, reliability and generality. These characteristics form the reference point from which the automatic evaluation algorithms start.

AE methods are relatively fast and cheap mainly because there is minimal human labour involved and there is minimum need for bilingual speakers of the source and target languages. Indeed, these methods can be used at any stage of an active system development process (Lavie, 2013).

The main disadvantage with AE systems is that they do not differentiate very well between minor differences in the output system compared to the reference (Papineni, 2012; Lavie, 2013). For example, a translation of “That is a cat” will be considered to be incorrect if the expected output was “That’s a cat”.

One method of automatically calculating translation accuracy especially on word based statistical machine models is by calculating the Levenshtein Distance between the two translations. The Levenshtein distance is a string metric for measuring the difference between two sequences. Informally, the Levenshtein distance between two words is the minimum number of single-character edits (i.e. insertions, deletions or substitutions) required to change one word into the other. It is named after Vladimir Levenshtein, who considered this distance in 1965 (Reese, 2015; V & PE, 2015).
Several other algorithms have been developed to do automatic evaluations (Satanjeev Banerjee, 2008; Reese, 2015). These include:

- BLEU
- METEOR
- General Text Matcher
- Translate Error Rate
- ParaEval precision and ParaEval recall
- Dependency overlap
- Semantic role overlap
- Word Error Rate over verbs
- Maximum correlation training on adequacy and on fluency

In the following section, close attention is paid to the first two, since they are the most popular amongst the set (Papineni, 2012; Lavie, 2013; Michael Denkowski, 2012).

(a) The BLEU Metric

BLEU was one of the first metrics and is currently one of the most popular in the field (Papineni, 2012). The foundational theory behind the idea is that a translation engine that gives output that is as close as possible to a human output is the best.

The BLEU metric works by calculating scores for individual segments and thereafter aggregating an average score for the segment scores (Papineni, 2012; Lavie, 2013).

For example, consider the following Swahili reference sentence:

- “Silaha zilizopatikana kwa wapiganaji wa kiyahudi zitaketezwa kwende mbuga ya wanyama baada ya wiki mbili”
Consider also an output of a machine translation engine that produces the following sentence:

— “Silaha kutoka kwa wanangambo wa kiyahudi zitachomwa baada ya wiki mbili”

The two sentences both portray the same meaning of weapons that have been found on fighters will be destroyed in a fortnight, by being burned. BLEU metric for the given output is therefore calculated as follows:

- 1-gram precision (how many single words are matching) : \( \frac{8}{11} \) (eight words match out of a total of 11 individual words)
- 2-gram precision (how many words occurring as a pair are matching) : \( \frac{4}{10} \) (eight words match out of a total of 10 pairs)
- 3-gram precision (how many words occurring in threes are matching) : \( \frac{2}{9} \)
- 4-gram precision : \( \frac{0}{8} \)

BLUE scores usually terminate at 4-gram precisions (Papineni, 2012). The total BLEU score is therefore calculated by doing a mathematical geometric average as shown in Equation 2.1

\[
BLEU = \left( \frac{8}{11} \right)^1 \left( \frac{4}{10} \right)^2 \left( \frac{2}{9} \right)^3 \left( \frac{0}{8} \right)^4
= \frac{668}{495} \approx 0.3374
= 33.74\%
\]

The advantage that BLEU score offers is that words get weighted individually as well as when they occur in phrases. It has a very measurable way of scoring metrics thus a machine can perform such a repetitive task autonomously. Given a reliable reference point, the scoring is fast and accurate.
One of the downsides of the BLEU score is what is known as The Brevity Penalty (Lavie, 2013). Suppose the previous output was “Silaha zilizopatikana kwa wapiganaji”. This implies a BLEU score of 100%! The MT output is very short and therefore this (erroneously) boosts the score.

(b) The METEOR Metric

The METEOR metric was developed so as to address the flaws of the BLEU metric (Michael Denkowski, 2014; Satanjeev Banerjee, 2008; Somers, 2005). It inherently works pretty much the same way as BLEU, but instead of calculating an arithmetic mean of the scores, it calculates a harmonic mean of the scores. In mathematics, a harmonic mean of a set of numbers is expressed as the reciprocal of the arithmetic mean of the reciprocals (Da-Feng Xia, 1999). For example, the harmonic mean of 1, 2, and 4 is

\[
\frac{3}{\frac{1}{1} + \frac{1}{2} + \frac{1}{4}} = \frac{3}{\frac{1}{1} + \frac{1}{2} + \frac{1}{4}} = \frac{12}{7}.
\]

This harmonic mean of calculation takes into account a metric that BLEU does not - recall. Recall, in the most basic of explanation, is the factor of not considering words that have already been averaged in the BLEU score. This indeed takes into effect a compensatory factor of punishing output that is too short – Brevity Penalty (Satanjeev Banerjee, 2008). This means that each word in a given output string can at most match only one word in the reference string. Further to this, METEOR also takes into account synonyms or alternative words (Lavie, 2013). METEOR, therefore, has been shown to outperform BLEU when comparing the scores that each give when measured against human judgments (Satanjeev Banerjee, 2008).

Figure 2.8 shows how METEOR matches the reference and output texts.
2.5 Research Approach

The research approach that was applied was an inductive approach. An initial observation – that there already exist machine translation engines out there that do a good job in some languages compared to others – formed the basis of the study. From these observations, a generalized theory was drawn and contextual conclusion made upon the observations. An inductive approach ensures that context of the research is accounted for. The disadvantage of this approach, however, is that the observations may be based on a small number of observed instances (Denzin & Lincoln, 2011).

2.6 Chapter Summary

In this chapter, the working mechanism of current machine translation engines has been highlighted. Popular MT engines were also introduced, and their possible working mechanisms illustrated in conjunction with their short comings. Techniques used to develop machine translation engines were also examined as well as ways of evaluating translations.

This chapter has shown that it is possible to create a better machine translator for less resourced languages by tapping into information already available to the general public and also using existing sources instead of creating data from scratch.
CHAPTER THREE

3.0 METHODOLOGY

3.1 Introduction

This chapter details the methods that were used in achieving the objectives of this study. It provides information regarding the methods used in the study, design of the research, methods that were used to collect data, the procedure of the research and the data analysis methods.

3.2 Research Design

A research design details the plan to be undertaken in carrying out the research (Muaz, 2013). There are different types of methods that can be employed (Venkatesh, Brown & Sullivan, 2016), and these include:

- Descriptive for example a case-study or surveys
- Correlation for example an observational to see how one thing relates to another
- Semi-experimental for instance a field experiment or a quasi-experiment determined from observation
- Experimental involving random items and experimenting with them
- Review of the works of other people and is obtained through literature review, for example.

The research method that was used in respect to this study were based on experiments. The research design method is therefore experimental. Based on similar projects that use random sets of data to test them, the study applied the same
methods of analysis to design the experiments. As much as possible, the study did not re-invent the wheel but made use of already existing technologies in order to achieve its objectives.

The research was conducted under the following broad categories:

1. Data extraction from Wikipedia into a local database
2. The use of the extracted data as a parallel corpus
3. Data mining to determine appropriate translations

### 3.3 Population & Sampling Design

The language experts used in the evaluations of the output of the machine translators were professional translators selected from the Department of Languages and Translations at the United Nations Office at Nairobi. In addition to these translators, other linguists from Microsoft, Translators Without Borders and The Kamusi Project were also used in the human evaluation of the translations.

The average age of the language experts was about 45 years, implying that they have good experience when it comes to translations and evaluation of translations. In addition to this, the reviewers were also selected by virtue of their being fluent in both the language of the particular set that they were evaluating.

### 3.4 Data Collection Methods

Data collection methods, be it quantitative or qualitative, depend on random sampling and structured data collection instruments that ensure order and systematic analysis from random samples collected. This produces outcomes and results which can be summarized, compared or generalized (Pierce, 2016).

By logging the results that were obtain after running the system through various translation requests, human evaluation of the output allowed the ability to obtain
the results of the accuracy of various MT engines including the one developed in this study. The results of the accuracy were obtained through a form of questionnaire sent to the various language experts.

3.5 Research Procedures

This study followed several phases which will be detailed in this section.

3.5.1 Evaluating existing MT Engines

Having obtained a list of commonly used random English phrases, the evaluation of existing MT engines involved passing these phrases through these engines and evaluating the output. The evaluation is by the very same human–based evaluation that the custom MT engine was subjected to.

The evaluation was based on not only the accuracy of the translation, but also the integrity of the translation such that it captures as much information that occurred in the original phrase as possible.

3.5.2 Wikipedia Data Extraction

Wikipedia offers a fortnightly dump of its articles. The dumps are labeled by date therefore it is possible to programmatically determine the most recent dump. The implementation was to download the dump and extract the English language articles.

The first paragraph of a Wikipedia articles is usually a definition of the term. This definition can differ in terms of length of the paragraph, such that for instance the English language definition paragraph is four sentences long while the same definition in the Swahili language article is three sentences long. Because of such differences, a structured query language database was not suitable and therefore the data extracted was stored in a NoSQL database that was implemented.
3.5.3 Design of the MT Engine

The translation engine developed used data stored in the local NoSQL database. This enabled it to be fast in terms of data access. The overall technique to develop the engine was example based, whereby a phrase is broken down into chunks which are then individually translated. The chunks were broken in such a way as to maintain the context of the source sentence.

As an example, a sentence such as *Donald Trump is the president of The United States of America* can be chunked as follows:

- Donald Trump – proper noun
- is – Conjunction
- the president – noun
- of – conjunction
- The United States of America – proper noun

The parts of speech tags are determined from the Penn Treebank part-of-speech tags Project (See Appendix A).

In the event that the chunking was not done and the translation gets performed one word at a time without putting into consideration the neighbours of a particular word, the result will be such that the translation quickly loses the intended meaning. Using the previous example, each of the words of the proper noun, “United”, “States”, and “America” will be considered as individual words potentially yielding inaccurate results.

Using another example sentence of “Thanks for calling”, a search on the corpus is undertaken where the words have been used as intended. That is to mean that consideration is made only wherever “Thanks” has been used as a noun phrase, “for” has been used as a past tense and where “calling” has been used as a past continuous tense.
As a fallback mechanism, a direct word-based dictionary was implemented that takes care of the instances in which a word match was not found in Wikipedia. This dictionary used data that was obtained from The Kamusi Project, an online collaborative dictionary.

### 3.5.4 Development of the MT Engine

The developed engine was a desktop application with an allowance for a further enhancement as a web application. The database engine was be a NoSQL engine and specifically MongoDB. The primary programming language was Java and the system was deployed on a Fedora Linux operating system environment.

The main data collection method was through an output text field inbuilt into the application. This data was sent to the reviewers via the Internet through a spreadsheet. The collected data was the original sentence, the calculated translation and the results obtained from Microsoft and Google translate engines for the same data. The translation engine also made use of readily available and open source natural language processing tools, such at Apache OpenNLP.

### 3.5.5 Evaluation of the MT Engine

Since the automatic translation mechanisms do not work very well when it comes to languages that are not structurally related, this study did not make use of automated evaluation mechanisms. The evaluation was therefore human based, partly because the study was analyzing just a few phrases and also because automatic Swahili evaluation data is not readily available.

For every translation request that was made, the results of the output were compared with those obtained from translating using Google and Microsoft translation engines.
3.6 Implementation Approach

The main implementation approach made of already existing software wherever possible. For example, the tool that was used to download data from Wikipedia is an existing open source tool called GNU Wget that already does the job of retrieving data from URLs.

Modular design pattern was used in the system development so the several related items were grouped together into a single entity. For example, all the classes that handle the fetching of data from Wikipedia were collectively put together as one entity that is semi-independent from the module that reads and writes data to the MongoDB instance.

Some test-driven development paradigm was also used when it came to the features which the outcome is already known. For instance, when fetching data from a particular article in Wikipedia, a manual browsing of the page will show the kind of data to expect. Against this, a test case is written in order to assert that the code is working as expected.

The entire program was managed by a version control system, Git. This enables future collaboration with interested parties in further improvement of the software.

3.7 Data Analysis Methods

Automatic translation mechanisms work very well when it comes to languages that have a similar structure of words. Because this is not the case when it comes to English and Swahili, this study did not use any automated methods to test the results.

Therefore, the analysis mechanism was be human based evaluation in form of a questionnaire that was be sent out to the set of evaluators conducting the output analysis. Furthermore, there exist many ways to express a particular statement and these subtle differences can be more accurately captured by a human when
compared to a machine evaluator.

3.8 Chapter Summary

This chapter has outlined how the proposed system was developed and also how it was analyzed for the purpose of determining whether the objectives of the study were met. Technologies that were be used to develop the system and the reason why they were chosen have also been examined.
4.0 SYSTEM IMPLEMENTATION

4.1 Introduction

To aid in performing the translation work, a software system was developed. The first section of this chapter provides the overall structural design of the system. The second section focuses on the implementation of the translation engine.

4.2 Analysis

This section will document the requirement specifications of the software. These requirements were obtained by analyzing the current popular MT engines and determining their current functionality. Therefore the requirements of the developed custom system were based on the core functionalities provided by Google and Microsoft Translation engines.

4.2.1 Functional Requirements

The following are the functional requirements of the system:

1. This system will allow a user to input a source sentence that is to be translated
2. The system will provide functionality to download Wikipedia source texts while displaying the progress of the download
3. There will be a display showing the results of the translation
4. The system will have a logging system that displays results of the steps performed in obtaining the translation – for instance chunking it and searching individual words
4.2.2 Non-functional Requirements

The non-functional requirements of the system are:

1. One system with one point of launch
2. Integration of various individual scripts into just one
3. Need to provide a windowed graphical user interface
4. A response of one minute or less

4.2.3 Users and Roles

The system is accessed by two different types of users:

1. The administrator who also is the corpus manager. His/her responsibilities are:
   - Deleting the database for purposes of refreshing it with new data
   - Indexing the database
   - Update any changes to the software

2. Regular users
   The user is the individual or company that uses the software to obtain a translation. The user is expected to interact directly with the software. The user makes use of the system in the following way:
   - View the user interface and be able to input a source phrase to be translated
   - See the output of the translation
4.3 Modeling and Design

4.3.1 System Workflow

The workflow of the entire process is demonstrated in how the various classes interact with each other. The system comprises two separate activities which are described through the following workflows:

(a) Fetching Wikipedia Data

The general steps involved when fetching data from Wikipedia are:

1. Fetching the dump from Wikipedia data
2. Extracting parallel titles from the fetched data
3. For each of the parallel titles, fetch the definition paragraph, storing them into the local database

These steps are illustrated in Figure 4.1 which shows the workflow diagram for fetching data from Wikipedia.
Figure 4.1: Workflow Diagram Describing Fetching Data From Wikipedia

The sequence diagram depicted in Figure 4.2 shows how the data gets fetched from Wikipedia.
A NoSQL database, specifically MongoDB, was chosen for the database design. This is because the descriptions of the data vary in length and thus structured fixed sizes that are required in traditional relational SQL databases were not suitable. Figure 4.3 shows the design of the MongoDB database and Figure 4.3 shows the physical implementation of the MongoDB database schema.
The translation algorithm follows the following general sequence:

1. Match the entire phrase in the database and return the Swahili equivalent

2. If entire phrase is not found, break the phrase into segments and translate each segment

3. If segment is not found, break the segment into individual words and translate them

Figure 4.5 shows the workflow of the translation algorithm.
4.3.2 System Design

The application for fetching data from Wikipedia made use of two classes as illustrated in the class diagram on Figure 4.6.
The class diagrams showing the design of the translation algorithms are modeled in Figure 4.7

![Class Diagram of the Translation Algorithm]

**Figure 4.7: Class Diagram of the Translation Algorithm**

### 4.3.3 System Architecture

The system uses abstractions of the different components, thereby making use of a 3-tier layered architecture. Based on a N-Tier application framework, there is the separation of the presentation layer, the business logic and the data (Microsoft, 2016b).
The presentation layer provides the graphical user interface by making use of windows and window-driven events such as mouse clicks. The business layer provides the actual functionality of the application. The data layer provides all the necessary access to the data in the databases or files or online streams.

The system architecture is modeled in figure 4.8

![System Architecture Diagram](image)

Figure 4.8: The System Architecture

### 4.4 Proof of Concept

As much as possible, available open source tools and scripts were used in order to ease the task at hand and also to not re-invent what had already been done.
4.4.1 Fetching data from Wikipedia

The first important step was to obtain a list of all Swahili articles from Wikipedia. Ordinarily, it will take a human a lot of effort and time to manually go through each and every article in order to extract text. Already, Wikipedia provides a dump of the articles that it contains. These dumps may be accessed from https://dumps.Wikipedia.org.

Wikipedia-parallel-titles is a tool used to build parallel titles from Wikipedia using any given language pairs (Dyer, 2015). In order to make processing faster, Wikipedia-parallel-titles recommends that the smaller of the language pairs be the one to be downloaded. In this case, since the task was to build Swahili-English corpus, the dump that was downloaded from the Wikipedia dump page was the Swahili wiki dump files. The reason is that there are fewer Swahili language articles on Wikipedia than there are English language articles.

(a) Downloading Wikipedia data

The necessary Wikipedia data was fetched using wget tool. GNU Wget is an open source software package for retrieving files and can make use of HTTP, HTTPS and FTP protocols (The Free Software Foundaton, 2016). Figure 4.9 shows Wget in action downloading the necessary files.

![Figure 4.9: GNU Wget Downloading Swahili Articles from Wikipedia](image)

Figure 4.9: GNU Wget Downloading Swahili Articles from Wikipedia
(b) Extracting Wikipedia titles

Figure 4.10 shows the Wikipedia-parallel-titles script running in order to extract the Swahili articles titles from the dumps that have just been obtained.

![Figure 4.10: Extracting Parallel Titles From Wikipedia Dumps](image)

Now that parallel titles have been obtained, the next step was to retrieve the first paragraphs from the actual Wikipedia titles. For this to take place, a Java program was written in order to go through the created titles.txt file line by line.

(c) Extracting Definition Paragraphs from Wikipedia

The MediaWiki action API is a web service that provides convenient access to wiki features, data, and meta-data over HTTP, via a URL (MediaWiki, 2016). This Wikipedia API presents output data in JSON format. This JSON data was parsed in order to extract the relevant data. The snippet of the fetching data from Wikipedia and the extraction of the JSON data to relevant tokens can be seen on the source code listing D.4 while Figure 4.11 shows the implementation of the code on NetBeans IDE.
Because fetching data using the API provided the most accurate results which had fewer comparative errors, it is the main method that this study settled for when it came to fetching the first paragraph from Wikipedia.

4.4.2 The Database

Now that the data has been fetched, it needed to be stored locally for ease and speed of access. The initial choice was the traditional MySQL. Quick and easy to deploy, MySQL offers a relational SQL schema which can be used to construct a parallel corpus from the data that has been fetched.

MySQL however did not work in this case because it requires structured data. One particular set of data coming in from Wikipedia could be one sentence long while another set may be seven sentences long or even just one word long.

The option was to go for MongoDB which provides a NoSQL structure. This database offers a good schema for data that is not structured. The database engine
even offers a way of having a record within another record. Most importantly, MongoDB has an inbuilt mechanism that was absolutely necessary in implementing the searching mechanisms of the algorithm: indexing for the purpose of full-text search.

In database technologies, full-text search is the technique of searching a database containing texts based on a search criteria specified by the user. This is similar to the way searching is performed on Google, by keying in certain string keywords or phrases and getting back the relevant results sorted by how they rank (Trivedi, 2015).

Depending on the specified language, MongoDB text-search indexes drop language-specific stop words. In English as an example, the words ”the”, ”an”, ”a” and ”and” are examples of stop words. It is interesting to note that no African languages are represented on MongoDB’s text search languages (MongoDB, 2016).

Also inbuilt in MongoDB text search is stemming. Stemming is the process of finding the word stem of a word. For example, words such as ”walking”, ”walked”, or ”walks” all have the same word stem of ”walk” (Reese, 2015).

4.4.3 The Translation Algorithm

Having implemented the custom translation algorithm based on the illustrated workflow, the next step was to prepare MongoDB for string based searches. This was done by creating a text index on the data. Text indexes work by searching for occurrences through the combination of any words from a sentence, irrespective of the order in which they occur. Text indexing in MongoDB also helps in removing stop words form key search parameters resulting in only the most relevant results.

The text score is another feature of MongoDB that was used since the results produced are ranked according to the most probably match to the least probable match.

The translation algorithm itself was then developed that followed the workflows
that were illustrated in Figure 4.5. Figure 4.12 shows a screen capture of part of the implementation of the custom translation algorithm.

```java
HashMap<String, String> mongoData = connection.fetchFromMongoB(String.format("\"\"", rawObject])
    original
    }});

if (mongoData.isEmpty()) // whole phrase not in wikipedia. break it down:
    String segments = (worder.tokenize(original));
    System.out.println("segments: "+ segments);

if (segments.isEmpty()) || segments.indexOf().equalsIgnoreCase(original.toString())
    String[] tokenizer = SimpleTokenizer.INSTANCE;
    String[] wordTokens = tokenizer.tokenize(original.trim());
    for (String token : wordTokens)
        translation.add(translate(token));
    else
        for (String segment : segments)
            
Figure 4.12: Snippet of the Custom Translation Algorithm

4.5 Chapter Summary

This chapter has described how the custom translation engine was designed and developed. In addition to the system itself, the underlying database structure and design have also been outlined.

The next chapter will show the results that were obtained within the given the system parameters.
5.0 Results and Findings

5.1 Introduction

This chapter shows the results that were obtained from the evaluation of the machine translation engines, both existing and custom, with the setup that was developed.

5.2 Status of current Machine Translation Engines

In order to evaluate the status of current machine translation engines, experiments were conducted in order to determine how accurately the engines managed to translate European languages compared to African languages.

5.2.1 Accuracy of Translation Engines for Common Language Usage

To establish the status of existing machine translation engines, commonly used English phrases were obtained and they were run through the different translation engines. The phrases may be seen in Appendix B. The aim was to establish the accuracy of these translation engines when performing translations from English to French, Spanish, Swahili and Afrikaans.

Google and Bing translation engines were able to perform English to Spanish translation pretty accurately according to the evaluators. From the results, Google and Microsoft engines scored an average accuracy of 93.50% and 86.50% respectively. Figure 5.1 illustrates these results.
The next language to be tested against Microsoft and Google Translate was French. Figure 5.2 shows the accuracy of the French phrases’ translation that were used when viewed against Microsoft as well as Google translation engines. The results obtained here are that on average, Microsoft Translate translated the sentences with a 69.25% accuracy while Google did the same with an accuracy of 71.25%.

Figure 5.2: Accuracy of English to French Translations on Microsoft and Google Translation Engines
Afrikaans was the first African language that was tested. The results gave an accuracy of 64.50% and 68.00% for Google and Microsoft engines respectively. Figure 5.3 shows the accuracy of the Afrikaans phrases’ translation that were used when viewed against Microsoft as well as Google translation engines.

Swahili was the next African language that was tested for accuracy. Figure 5.4 shows the accuracy of the Swahili phrases’ translation that were used when viewed against Microsoft as well as Google translation engines. 29.5% and 53.5% were the respective average scores for the engines.
The next step was to determine the accuracy of existing translation engines when it comes to translating academic content. In this context, academic phrases are those that are used in a school setting. Such phrases are typically less commonly used in general everyday speak. An example of an academic phrase is “The pH of water is 7” or “The Alcohol is a chemical compound from carbon, hydrogen and oxygen”.

Figure 5.5 shows the accuracy of the translation of academic phrases against Microsoft as well as Google translation engines of which the engines scored an average of 29% and 45% for Google and Microsoft respectively.
5.3 Implementation Results of Custom Machine Translation Engine

Two methods were attempted in implementing the custom translation engine. This section discusses the two methods in detail.

5.3.1 Moses Statistical Engine

The first attempt was to create the translation system based on a pre-existing statistical translation system called Moses Statistical Machine Translation System. As the name suggests, Moses is a statistical based machine translation engine in
that it attempts to make sense of parallel data by making use of a massive collection of parallel and aligned sentences whereby sentences in one language (known as the source language) map to the the other (known as the target language) in a relationship format.

Moses has a standard method of operation. It comes with a basic user guide that provides instructions on how to develop a French to English translation engine (Koehn, 2007). These instructions were adapted to the language set of English to Swahili to create an English-to-Swahili translation engine.

The translation engine followed the following steps to perform the translation:

1. **Preparation of the Corpus** At the start of this step, the data from Wikipedia had already been successfully fetched into the local MongoDB. This data was then prepared for the training of Moses by performing the following steps:

   (a) **Tokenization**: This means that separators (for example spaces and commas) have to be inserted between the individual words and any other punctuation.

   (b) **True-casing**: The initial words in each of the sentences are then converted to their most probable casing – uppercase or lowercase – so as to reduce data sparsity.

   (c) **Cleaning**: Long or empty sentences are removed or trimmed to a much manageable length.

2. **Language Model Training** The language model training was then done. This ensured a fluent output by removing non-English words in the source data. Therefore, the training is built with the source language (i.e. English in this case).

3. **Training the Translation System** Finally to the main event – training the translation model. In order to do this, Moses invoked a script that performs word-alignment. For example, whenever the word *August* appears in the source English language, *Agosti* appears in the target language. The engine therefore interprets this
as an indicator that there is a strong possibility that Agosti is the English equivalent of August. GIZA++ was the application used for word alignment.

This eventually produced a final binary file that contains the statistical data that was used to do the translations. This file was stored on the local filesystem.

4. **Tuning** Tuning is the process where items that are way off the average are removed. This was the slowest part of the process. Because it needs a different data set from the one that was used to do the training, a separate data set was downloaded in order to perform this step. The end result of tuning was a moses.ini file with trained weights.

Multiple cores were very useful here. While specifying to Moses to use 4 cores, the process took 3.5 hours to complete in this case. Figure 5.6 shows the CPU usage during the tuning process, showing that all the four cores of the CPU were maxed out at 100%.

![System Monitor](image)

**Figure 5.6: CPU usage during the tuning process of Moses Engine**

5. **Testing** Finally Moses was run and a few test phrases were input. Figure 5.7 shows Moses engine running and performing a few translations.
5.3.2 Custom Translation Engine using Java

Moses is based on GIZA++. This word alignment tool does not perform well when it comes to languages that are not structurally similar, as shown in section 2.3.2. Furthermore, Moses did not perform well in translation cases involving phrases, but only worked when the requests involved single words.

Therefore, an example-based translation engine was implemented using standard Java and a MongoDB database. This engine worked by chunking input phrases from the original input sentence, and the chunks were then individually looked up in the database.

The development of the custom translation engine followed the workflows described in Figure 4.5. Each of the steps was individually tested to conform to the requirements detailed in the workflow.
Step 1: Fetching Titles of Articles from Wikipedia

Extracting Wikipedia Data Before settling for the Wikipedia API for purposes of extracting data from Wikipedia, the initial attempt was to use a Java library called JSoup for this exercise. JSoup provides a means of scraping and parsing HTML from any URL, file, or string (Hedley, 2016). With this library, the specific paragraph of Wikipedia is specified from a certain page. Code snippet at Appendix D.3 shows the working code of jsoup library.

For the majority of the articles, the data came as was expected but there were some anomalies that were found:

1. For articles which start with a disambiguation help text, jsoup fetched the disambiguation text instead of the actual text from the article. For example, in the Swahili article about the digit 1, obtained from the link [https://sw.wikipedia.org/wiki/1](https://sw.wikipedia.org/wiki/1), jsoup extracted the initial disambiguation text *Lango la Historia | Lango la Biografia | Karibuni | Orodha ya Miaka*, instead of the initial paragraph, *Makala hii inahusu mwaka 1 BK (Baada ya Kristo)*. Figure 5.8 shows this extraction anomaly.

2. Even though Exception handling was implemented, jsoup did not fail silently and kept returning exceptions. The internal error handling for jsoup appears to have an inbuilt class that is pre-compiled and not easy to reverse-engineer. The implication of this is that the data in the database was exception traces instead of actual data or perhaps even blank entries.

![Figure 5.8: Swahili Article for Arabic Numeral 1 showing anomalies](image-url)
Because of these failings, the next option was to fetch the data using the API Wikipedia provides. The automated script written provided more accurate and reliable results which had fewer comparative errors, although it ran comparatively slower than JSoup library implementation.

The script ran successfully and stored the output of the data in a file named titles.txt. A snapshot of the contents of the file is shown in Figure 5.9.

![Figure 5.9: Resulting Parallel Corpus File after Extracting Parallel Titles From Wikipedia Dumps](image)

After the titles had been fetched, the next step was to fetch the first paragraphs of the titles from Wikipedia and insert them into MongoDB.

Figure 5.10 shows how the program ran as it was fetching the first paragraphs of each title from Wikipedia, and saving the data into the local MongoDB instance.
The data was indeed successfully inserted into the database, as illustrated in Figure 5.11.
Step 2: The MongoDB NoSQL Database Preparation

The next step after data had been fetched into MonGODB was to create an index for the data. Source code listing in appendix shows how the index was created. An important thing to note is that the index uses English as a default language. This removes specific English language stop words. It if were a Spanish based index, then Spanish stop words will be removed.

Figure 5.12 shows the results of creating the index in MongoDB.

![MongoDB database displaying indexes](image)

Figure 5.12: MongoDB Database Displaying Indexes
Step 3: The Translation Application

In order to walk through the workflow, this step will demonstrate the process of translation of a sample sentence;

"Rabies is an infectious disease of the nervous system from a cat and a dog"

(i) Obtain the translation request from the user

The user puts in the translation request from the graphical interface and clicks on the "translate" button as shown in Figure 5.13

(ii) Look up the whole phrase in MongoDB

The whole phrase is now matched in MongoDB in order to determine if it exists wholly or partially in any of the WIkipedia entries. In this particular case, the entire phrase was not found and in this case, it was broken down into chunks using
Apache OpenNLP. In this case, the phrase was broken down into eight segments: "Rabies", "is", "an infectious disease", "of", "the nervous system", "from", "a cat" and "a dog".

The results of the lookup and chunking process are displayed in Figure 5.14.

Figure 5.14: Look-up in MongoDB and chunking phrases using Apache OpenNLP

(iii) Translating the Chunks

With the sentence chopped up into chunks that need to be searched, the program now looped through the chunks and translated them one after another. Once again, if the whole chunk of a phrase, for example, "The Nervous System" is not to be found then the application split the chunk further into its individual constituent words and translate them directly. In the scenario whereby a further word split is not found, a fall-back basic dictionary was used to translate the words. The dictionary uses data obtained from The Kamusi Project.

(iv) Concatenating the Partial Translations

The program works such that the partial translations of the chunks are held in an array. At the end of looking up all the chunks, the array contents gets stitched up. In the particular case example in use, the output was:

"Kichaa cha mbwa ni Kichaa ni Ugonjwa wa kuambukiza ya Kichaa ya Mfumo wa neva kutoka Pakakaya Mbwakaya"

Figure 5.15 shows the final output of the process.
Sometimes the output contained a lot of noise. As such, the algorithm could not get a properly weighted match, because each of the entries in the database had an equal number of occurrences thereby making all the possible translations have a pretty much equal probability. An example of such a sentence was “Kenya is a democratic country”. The results of chunking this phrase gives “democratic country” as one of the chunks. “democratic country” matched with a lot of entries, some of which included irrelevant results. Some of the matched entries were Ghana, Burkina Faso and the Democrats and Republicans from the United States.

When this was the case, the algorithm was designed to use a fall-back mechanism in order to try to eliminate ambiguity and noise. Using a basic dictionary from The Kamusi Project, the algorithm is designed to break up the original phrase into its constituent words. These words are then looked up against the dictionary for a verbatim translation.

Figure 5.16 shows the raw output of this translation, which was then chunked as illustrated in Figure 5.17 in order to produce the final translation of “Kenya ni
Figure 5.16: Translation Results for "Kenya is a democratic country"

Original number of words: 5
Translated number of words: 204

[Kenya, is, a, democratic, country]
5.4 Evaluation of Custom Translation Engine

5.4.1 Evaluation of Custom Translation Engine in Common Language Usage

The same common English phrases (as seen in Appendix B) were passed through our custom translation Engine.

The results of this are displayed in Figure 5.19 which show an average accuracy of 15% for common phrases.
5.4.2 Evaluation of Custom Translation Engine in Academic Usage

Further to that, a new set of phrases that focused on academic matters was passed through the engine. These phrases are listed in Appendix C.

The results obtained from the academic phrases are shown in figure 5.20. The results indicated average accuracies of 27% for Google, 45% for Bing and the custom engine averaged 91%.
5.5 Chapter Summary

This chapter has shown the results of the analysis of existing machine translation engines. The chapter has also shown the outcome of developing an example-based custom translation engine, and evaluation of the developed translation engine. The next chapter gives a detailed analysis of these results so as to explain why the results came out the way they did.
6.0 Discussions, Conclusions and Recommendations

6.1 Introduction

This chapter gives detailed discussions and analysis of the outcome of the objectives that this study set out to achieve. The aim is to help the reader understand why the tests conducted gave the results obtained, as well as the merits and demerits of the approaches used. The chapter will also challenges encountered, and a brief outline of the shortcomings of the study. Furthermore, it outlines that which was achieved with the available resources and within the given time-frames. Recommendations on how to further improve on the research are also made.

6.2 Summary

Having accomplished the set out objectives of analyzing current machine translation engines with respect to African languages, this research project has shown that African language resources in currently available machine translation engines are limited. Wikipedia can be used to push the boundaries of educational content translation in order to make them more accurate.

In this research project, the main objective was to model a machine translation system. The initial attempt was to adapt an already existing software – Moses. Having understood the merits and demerits of Moses, the second attempt was aimed at solving those limitations of Moses by designing and implementing a custom engine. The translation engine made use of example-based methods of machine translation algorithms.

There exists adequate data regarding machine translation engines, but not much data is readily available for African languages, Swahili being just but an example.
The resulting system, therefore, needs to be improved not just by itself, but also through making Swahili data accessible through engaging relevant stakeholders.

A major and important note is that the structural differences between English and Swahili created a bit of a challenge. If the two languages were as structurally similar as Spanish is to English, perhaps this project may have just fully adapted Moses and not had to design and implement a custom engine altogether.

6.3 Discussions

6.3.1 Status of Existing MT engines

When compared to French, Spanish gave a better translation accuracy when evaluated on Google and Microsoft engines. Indeed, Spanish is very much structurally similar to English than French. So structurally and syntactically similar is Spanish to English, that it can be considered to be broken English. For example, “much” is mucho in Spanish and beacoup in French. A car is carro in Spanish, voiture in French.

African Bantu languages, aside from having limited dictionary resources, have an especially complicated verb system which combines the noun, tense and even negation into one word. Thus an English phrase that is represented by five words, “I did not visit him”, is only one word in Swahili, sikumtembelea. Interestingly enough, there is no way of extracting the context of gender from sikumtembelea since the same translation applies for the phrase “I did not visit her”.

That aside, the little resources that exist in African languages are generally common everyday usages that range from greetings to basic phrases or gossip on online forums. Advanced topics and academic data are not so much available because, as outlined in the literature review section, English is the most widely used language of instruction in schools at least in the East African region.

Having done a simple comparison experiment on popular translation engines of
Microsoft and Google, this study set out to try and identify why Swahili translation was poor. A closer examination of the translation showed that the translation engines were not factoring in the context in their translation. For example when translating the word “Sink” as a noun, Google Translate treated it as if it was a verb and consequently gave results of sink as an action, as illustrated in Figure 6.1. Figure 6.2 and Figure 6.3 shows other phrases that were put through the same analysis, this time on a different translation engine.

Figure 6.1: Interpreted Translation of “This is a sink” on Google Translate
Figure 6.2: Interpreted Translation of “We had a gay time” on Microsoft (Bing) Translate

Figure 6.3: Interpreted Translation of “Huyu ni Somo” on Microsoft (Bing) Translate
As these results show, the effects of not factoring in context in translation may end up twisting the intended meaning thereby making the resulting translation mean something different to the intended meaning of the original phrase.

### 6.3.2 Development of MT engine

#### (i) Fetching data from Wikipedia

The translation engine developed in this study involved corpus data from Wikipedia. This Wikipedia data is freely available, and it contains all the available data in any particular language set that has been selected (Wikipedia, 2016). The data comes in a raw wiki format, and therefore this means that an additional effort has to be made in order to extract the exact relevant data that is needed. Consequently, this implies that there needs to be written a script that does the extraction of the exact needed data.

So as to not reinvent the wheel, two open source data extraction tools were explored to extract the necessary data from Wikipedia. The tools require just the link to the Wikipedia entry of interest. Therefore, since an open source tool already existed that extracts the titles from the dumps (Dyer, 2015; Hedley, 2016), the work of extracting the data was drastically reduced.

The first data extraction attempt involved using jsoup library. As a native Java library that can be plugged into a Java application, it ran pretty fast to connect to Wikipedia and extract the text. The downside that was noticed was that jsoup often returned faulty data. Consequently, the choice was made to switch to the Wikipedia API. The only disadvantage of Wikipedia API was that the data fetching process was a little bit slower in comparison to jsoup. This was a compromise that had to be made so as to sacrifice speed but obtain reliable data.

MongoDB was the database that was used to store the data locally. Having been designed to handle big data of millions or billions of records, MongoDB experienced problems in handling the 27,000 records that were fetched from Wikipedia.
the data took approximately one minute. Apart from the ability to handle large
data sets, MongoDB was selected because it is free and open source. There are no
licensing fees involved and therefore it was cheap to deploy. Plenty of help through
online forums was also available.

(ii) Moses Statistical MT Engine

The first attempt in creating the translation engine involved adapting Moses for
English to Swahili translation. Moses is a statistical machine translation engine. This
means that it relies on the analysis of parallel language sentences, and determining
the most common matches for a particular word by checking how many times it
occurs in the opposite language data set (Koehn, 2007).

The task of generating these statistical matches depends on the GIZA++ package.
The GIZA++ algorithm handles very well closely-related words in either languages.
Consider the English sentence “I ate an apple”. This sentence has the Spanish
and Swahili equivalent translations of “Yo come una manzana” and “Niliila tofaa”
respectively. Deconstruction of the sentence and its translations morphologically
yields the analysis shown in Table 6.1:

<table>
<thead>
<tr>
<th>English:</th>
<th>I</th>
<th>eat</th>
<th>an</th>
<th>apple</th>
</tr>
</thead>
<tbody>
<tr>
<td>French:</td>
<td>Je</td>
<td>mange</td>
<td>une</td>
<td>pomme</td>
</tr>
<tr>
<td>Spanish:</td>
<td>Yo</td>
<td>como</td>
<td>una</td>
<td>manzana</td>
</tr>
<tr>
<td>Swahili:</td>
<td>Niliila</td>
<td>tofaa</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

As illustrated, English and other FIGS languages align very well when using
GIZA++. Swahili on the other hand presents quite a challenge. This is evident
since three words in English – “I”, “eat” and “an” – have been translated into only
one “super-verb”.

This is a characteristic of many Bantu languages under which Swahili falls. Verbs get conjugated such that one word expresses all the elements of the actor (first person, second person), the tense (past, present, future), the subject of the verb (you, him, they, us), the actual action as the root or stem and finally a suffix that indicates elements that, for instance, tells if the verb was done on behalf of someone else. Table 6.2 illustrates this in a bit more detail, using the translation output of “Niliila tofaa” as initially described in Table 6.1.

<table>
<thead>
<tr>
<th>Ni</th>
<th>li</th>
<th>i</th>
<th>la</th>
<th>tofaa</th>
</tr>
</thead>
<tbody>
<tr>
<td>Me</td>
<td>did (past tense)</td>
<td>it (non-living)</td>
<td>eat (root)</td>
<td>apple (noun)</td>
</tr>
</tbody>
</table>

A negation of the sentence adds even more confusion to GIZA++ because a prefix is appended to the compounded verb. Still using the example of “I did not eat an apple”, the translation now becomes simply “Sikuila tofaa”. Now, one Swahili word represents five English words!

This was the reason that Moses was inappropriate for this study. As seen in the literature section, GIZA++ considers punctuation marks such as commas and question marks to be full words. It is because of this that Moses translated many of the words to Swahili as commas or full stops.

Just as a side note, Moses is processor and resource intensive. Indeed, the documentation recommends making use of as much parallelization as possible. In the instance of this attempt, the entire tokenization, alignment and training process took about 2.5 hours using 4 cores on an IBM laptop (Intel i7-2640M, 8GB RAM, SSD). This was not worth the while, especially considering that the corpus keeps changing frequently and needs frequent updating.
(iii) Custom Machine Translation Engine

The main functional requirement of this study was to model a machine translation engine that could perform translations targeting academic content. The likes of Microsoft and Google do common language translations pretty fast because they have pre-analyzed the data and they store it ready for requests. The custom engine however does the translations on the fly, going through the data collections and calculating the relevant translations as needed.

The custom translation engine initially kept returning a lot of unnecessary and unwanted translations. Noise is the term used to describe such. An analysis of the returned data revealed that the cause was the inclusion of stop words. These stop words occur in almost all the English database.

Removing stop words ensured that only the relevant key words or subjects within a sentence were considered. For example, Rothschild is a species of giraffe without the stop words results in Rothschild species giraffe. Notably, this makes it possible to focus on only the relevant subject matter through searching the databases and not including the very common word “is”.

Another failing point that was realized is that the custom algorithm seemed to fail when it came to plurals. This is because items in Wikipedia are mostly singular in nature, such that for example, an entry will be on “Thief” and not “Thieves”.

6.3.3 Accuracy of Custom MT Engine

(i) Common phrases

From the results, the custom translation engine did not fair well with common English phrases. This was notable through two particular reasons:

1. The data set from Wikipedia with respect to Swahili is comparatively shallow – 25,000 articles in the Swahili Wikipedia compared to 25,000,000 in the Spanish Wikipedia and over 500,000,000 in the English Wikipedia. Thus the statistical
learning set was not rich enough to be able to provide concrete comparisons between certain words.

2. Most of the words used in common English vocabulary existed in so many of the Wikipedia articles. The academic phrases offered a particularly unique set because the academic phrases appeared in very few, perhaps only one, article thereby making the translations pretty obvious to our algorithm.

(ii) Academic phrases

On the other hand, the likes of Google Translate and Microsoft Translator did not fare so well when it comes to academic matters because they probably use a lot of common usage data sets in order to statistically analyze possible translations. Because these data sets lack academic content, the translators are not able to accurately translate such content.

A closer examination of the results from the translation engines show how the engines performed with specific examples as illustrated on Figures 6.4, 6.5 and 6.6. From Figure 6.6 it can be seen that the custom translation engine was able to correctly translate the Swahili name for the planet Jupiter as Mshtarii and it also incorporated the term “outer” in its translation, something that Google and Microsoft did not.
Figure 6.4: Translation of "Jupiter is a planet from outer space" on Google Translate

Figure 6.5: Translation of "Jupiter is a planet from outer space" on Microsoft Translator
6.4 Conclusions

The government of Kenya has undertaken an ambitious project to equip children with laptops and tablets for the purposes of facilitating electronic based learning. This initiative can only bear fruit provided that there is content relevant to the studies being undertaken. Many Kenyans learn English as a second language. Swahili or other African languages is the mother tongue. Therefore with content in Swahili, a better and deeper understanding of subject matter takes place. Much of the academic content already exists albeit in English. Therefore, translating this content is the most practical method of getting the content in Swahili. This is especially so since the content is not necessarily new, but just needs to be interpreted.

There already exist machine translation engines, such as Microsoft Translator and Google Translate, which aim to make this task easier. However, African languages are generally under-represented in these engines. The translation results they
produce are comparatively inaccurate when it comes to translating content to African languages. They are even more inaccurate when translating academic type of content. This can largely be attributed to the source of data used to train the translation engines. Many machine translation engines make use of corpora made up of phrases that are found in every day speech, into which academic terms are not adequately incorporated.

Wikipedia, an on-line crowd sourced encyclopedia, offers very good sources of data for purposes of translation works. This study has shown that using Wikipedia as a corpus can provide a viable source of data for academic related translations and specifically so when it comes to African languages.

Therefore, this project modeled an English to Swahili translation engine that uses Wikipedia as a source of translation corpus data. As an emphasis, this study did not set out to create yet another translation engine altogether, but to just improve on, and complement, a small aspect of the current existing engines. The approach that was used was to compare same language articles in Wikipedia and build a parallel corpus which is then used to create a translation database. It is worth noting that Wikipedia on its own cannot provide a comprehensive data set for any machine translation engine. As proof of concept this model shows English to Swahili translations and presents preliminary results here. Indeed, further work is required for more accurate output alignment and combining the output to ensure fluency and accuracy.

This study was further motivated by the directive of the Communications Authority of Kenya that aims towards having at least 60% of the media content being local. This content therefore needs to be translated into local languages for presentation purposes. The study proposes a solution that can be scaled to learn and translate other local languages.

Finally it is worth noting that Kenya, like many other developing countries, imports numerous products from foreign countries. Many of these products have their labels and instructions written in these foreign languages, more-so English. This poses a potential threat to consumers who do not understand these languages for example
in the case of medical drugs.

6.5 Recommendations

6.5.1 Current MT Engines

Current MT engines are proprietary and their exact working mechanism is not public information. Their sources of data are also not in the public domain. It can, however, be assumed that these engines make use of data sources from the bible, quran, the UN corpus or from social media and news websites. Whether or not they make use of Wikipedia as a corpus is also not known.

Therefore, based on the results of this research, this study recommend that the engines pay attention to Wikipedia as a rich source of corpus data. This is of particular importance when it comes to translating academic texts from English to Swahili or any other African language.

6.5.2 Development of a Custom MT Engine

This study shown that the model can work for English to Swahili translation works, even though the two languages are not structurally similar. Extending the algorithm to incorporate many more African languages will be an added advantage. This project heavily depended on available Wikipedia data. Consequently, one way of improving the accuracy of the translations is to improve Wikipedia articles in terms of quality and quantity. Indeed, this custom engine can be extended and used to improve Wikipedia by adding a feature that enables adding or editing Wikipedia entries.

The custom MT engine was developed so that it fulfills its main functional requirement of performing translations. The non-functional requirements such as efficiency have only but been partially taken care of. Therefore, it will be good to improve the algorithm by making use of a few integrated rule sets. This will help
because many of the stop words have only one translation and thus returning the translations through a rule set is faster than opening and closing several database connections.

6.5.3 Evaluation of MT Engines

The evaluation of translation engines, both existing as well as the custom engine, was based on a questionnaire. The main reason behind this is that the data needed for automatic evaluation for Swahili is not readily available. Even MongoDB indexing does not have stop word data entries for any African language, let alone Swahili. As an extension of this project or perhaps a different project altogether, Swahili stop words need to be included in MongoDB as well as Swahili Name Entity Recognition. This will be very useful when it comes to enabling automatic evaluation mechanisms.

Finally, further work is required for more accurate output alignment and combining the output to ensure fluency and higher percentages of accuracy. Indeed a recommendation for further improvement of this study is to explore the possibility of creating a round-trip evaluation mechanism for automatic evaluation of machine translation output.


APPENDIX ONE

1.0 PART-OF-SPEECH TAGS USED IN THE PENN TREEBANK PROJECT

This list is adapted from
https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html.
The complete list of The University of Pennsylvania (Penn) Treebank Tag-set can be found at

<table>
<thead>
<tr>
<th>Tag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CC</td>
<td>Coordinating conjunction</td>
</tr>
<tr>
<td>CD</td>
<td>Cardinal number</td>
</tr>
<tr>
<td>DT</td>
<td>Determiner</td>
</tr>
<tr>
<td>EX</td>
<td>Existential there</td>
</tr>
<tr>
<td>FW</td>
<td>Foreign word</td>
</tr>
<tr>
<td>IN</td>
<td>Preposition or subordinating conjunction</td>
</tr>
<tr>
<td>JJ</td>
<td>Adjective</td>
</tr>
<tr>
<td>JJR</td>
<td>Adjective, comparative</td>
</tr>
<tr>
<td>JJS</td>
<td>Adjective, superlative</td>
</tr>
<tr>
<td>LS</td>
<td>List item marker</td>
</tr>
<tr>
<td>MD</td>
<td>Modal</td>
</tr>
<tr>
<td>NN</td>
<td>Noun, singular or mass</td>
</tr>
<tr>
<td>NNS</td>
<td>Noun, plural</td>
</tr>
<tr>
<td>NNP</td>
<td>Proper noun, singular</td>
</tr>
<tr>
<td>NNPS</td>
<td>Proper noun, plural</td>
</tr>
<tr>
<td>PDT</td>
<td>Predeterminer</td>
</tr>
<tr>
<td>POS</td>
<td>Possessive ending</td>
</tr>
<tr>
<td>PRP</td>
<td>Personal pronoun</td>
</tr>
<tr>
<td>PRPS</td>
<td>Possessive pronoun</td>
</tr>
<tr>
<td>RB</td>
<td>Adverb</td>
</tr>
<tr>
<td>RBR</td>
<td>Adverb, comparative</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td><strong>RBS</strong></td>
<td>Adverb, superlative</td>
</tr>
<tr>
<td><strong>RP</strong></td>
<td>Particle</td>
</tr>
<tr>
<td><strong>SYM</strong></td>
<td>Symbol</td>
</tr>
<tr>
<td><strong>TO</strong></td>
<td>to</td>
</tr>
<tr>
<td><strong>UH</strong></td>
<td>Interjection</td>
</tr>
<tr>
<td><strong>VB</strong></td>
<td>Verb, base form</td>
</tr>
<tr>
<td><strong>VBD</strong></td>
<td>Verb, past tense</td>
</tr>
<tr>
<td><strong>VBG</strong></td>
<td>Verb, gerund or present participle</td>
</tr>
<tr>
<td><strong>VBN</strong></td>
<td>Verb, past participle</td>
</tr>
<tr>
<td><strong>VBP</strong></td>
<td>Verb, non-3rd person singular present</td>
</tr>
<tr>
<td><strong>VBZ</strong></td>
<td>Verb, 3rd person singular present</td>
</tr>
<tr>
<td><strong>WDT</strong></td>
<td>Wh-determiner</td>
</tr>
<tr>
<td><strong>WP</strong></td>
<td>Wh-pronoun</td>
</tr>
<tr>
<td><strong>WP$</strong></td>
<td>Possessive wh-pronoun</td>
</tr>
<tr>
<td><strong>WRB</strong></td>
<td>Wh-adverb</td>
</tr>
</tbody>
</table>
APPENDIX TWO

2.0 COMMON PHRASES USED IN TESTING

GOOGLE AND MICROSOFT TRANSLATION ENGINES

– What have you been up to lately?
– I’m fine, thanks. How about you?
– I should have called you and told you I’d be late
– Do you have a stomachache?
– How do you say chekechea in English?
– Don’t worry about it.
– I don’t think we’ve met before. My name is Leah
– I’d like to introduce you to Prof. Mutanu
– It’s getting late, I have to leave.
– Thanks for calling.
– I’m calling to find out when the bar closes.
– My lawyer friend.
– I’m sorry for your loss.
– Are you free on Saturday night?
– Tell me the difference between A and B
– Can I borrow a pen?
– Get together in groups of four
– I have a question for you
– Would you mind repeating that?
– Are you with me?
APPENDIX THREE

3.0 ACADEMIC PHRASES USED TO TEST TRANSLATION ENGINES

- A chemical compound is an entity consisting of two or more atoms, at least two from different elements, which associate via chemical bonds
- A tissue is an ensemble of similar cells from the same origin that together carry out a specific function
- A bone is a rigid organ that constitutes part of the vertebral skeleton
- Cancer is a group of diseases involving abnormal cell growth with the potential to invade or spread to other parts of the body
- Vertebrates comprise all species of animals with backbones
- Jupiter is the fifth planet from the Sun and the largest in the Solar System
- In Chemistry, pH is a numeric scale used to specify the acidity or basicity of an aqueous solution
- A glacier is a persistent body of dense ice that is constantly moving under its own weight
- Iron is a chemical element with symbol Fe (from Latin: ferrum) and atomic number 26
- The periodic table is a tabular arrangement of the chemical elements, ordered by their atomic number
- In physics, a force is any interaction that, when unopposed, will change the motion of an object
- The nucleus is the small, dense region consisting of protons and neutrons at the center of an atom
- Solid is one of the four fundamental states of matter
- The newton (symbol: N) is the SI unit of force
- A volcanic crater is a roughly circular depression in the ground caused by volcanic activity
- Lava is the molten rock expelled by a volcano during an eruption
- Electromagnetic waves are waves that contain an electric field and a magnetic field and carry energy
- Wikipedia is an Internet encyclopedia project in many languages
- A comet in the solar system
- Calcium for bone
- The Iron Age is an archaeological era
- Europa is a moon from outer space
- Moses is a prophet in Abrahamic Religions
- Beetles are a group of insects that form the order Coleoptera
- The Habitat of Polar Bear is at the North Pole
- The computer keyboard is red in color
- 7 Penguin from Antarctica
APPENDIX FOUR

4.0 SOURCE CODE LISTINGS

D.1 Downloading Wikipedia Dumps using GNU Wget

```
```

Listing D.1: Downloading Wikipedia Dumps

D.2 Extracting Parallel Titles from Wikipedia Dumps

```
./build-corpus.sh en arwiki-20160630 > titles.txt
```

Listing D.2: Extracting Parallel Titles from Wikipedia Dumps

D.3 Wikipedia Content Extraction Using JSoup Java Library

```
String link = String.format("http://%s.wikipedia.org/wiki/%s", parameters);
Document doc = Jsoup.connect(link).timeout(5000).get();
Elements paragraphs = doc.select(".mw-content-ltr p, .mw-content-ltr li")
    .first();
Element firstParagraph = paragraphs.first();
return firstParagraph.text(); //Print out just the first paragraph
```

Listing D.3: Wikipedia Content Extraction Using JSoup Java Library

D.4 Downloading Wikipedia Titles Using Wikipedia API
```java
String link = String.format("https://%s.wikipedia.org/w/api.php?format=json&action=query&prop=extracts&exlimit=max&explaintext&exintro&titles=%s", parameters);
URI uri = new URI(link);
JSONTokener tokenizer = new JSONTokener(uri.toURL().openStream());
JSONObject root = new JSONObject(tokenizer);
JSONObject query = (JSONObject) (root.get("query"));
JSONObject pages = (JSONObject) (query.get("pages"));
```

Listing D.4: Downloading Wikipedia Titles Using Wikipedia API

### D.5 Saving Wikipedia Pages into MongoDB

```java
WikipediaDataFetcher wikiFetcher = new WikipediaDataFetcher();
String wikiTextEnglish = wikiFetcher.fetchData("en", englishTitle);
String wikiTextSwahili = wikiFetcher.fetchData("sw", swahiliTitle);
```

```java
db.getCollection("wikipedia").insertOne(
    new Document()
      .append("title", englishTitle)
      .append("en", wikiTextEnglish)
      .append("kichwa", swahiliTitle)
      .append("sw", wikiTextSwahili));
```

Listing D.5: Saving Wikipedia Pages into MongoDB

### D.6 Indexing the MongoDB Collection

```java
db.wikipedia.createIndex({ "en" : "text", "sw" : "text" }, {
    default_language: "english" });
db.wikipedia.getIndexNames();
```