Kinyarwanda — English Translator

A Statistical Machine Learning Approach

Patrick Niyongabo
Hendrix College

A thesis submitted for the degree of
Bachelor of Arts in Computer Science

May 2017

I dedicate this thesis to my four young sisters:

Janvière, Joseline, Josiane, and Fortune. I miss y'all!

Acknowledgements

My deep gratitude goes to my thesis advisor, Dr. Gabrier Ferrer for his patience and guidance. I would also like to take this moment to thank every faculty member of the Hendrix College Math and Computer Science department for their relentless support. To all students who helped me or encouraged me in any way when I was working on this project, I can only say thank you!

Abstract

Statistical Machine Translation (SMT) is a statistical machine learning (SML) technique that uses statistical models to generate translations from one language to another. In this paper, I describe the statistical machine translation (SMT) approach that I am using to make an English to Kinyarwanda translator, and explain the phrase-based model used in some of the most prominent SMT systems such as Cdec, Joshua, and Moses. The latter SMT system being the one I am using to develop my English to Kinyarwanda translation model.

Contents

List of Figures

Chapter 1

Introduction

In recent years, due to an increase in the use of mobile phones and Internet around the World[17, p. 131-133], some developing nations are turning to a technology-based services industry with the hope that technology, and the Internet in particular, will serve as tools to speed up their economic growths towards sustainable development[14]. However this has proven to be a difficult challenge for countries whose popular, and in most cases, official languages are not widely used elsewhere around the World.

The reason why this is a big issue is because even though there is a lot of free and easily-accessible information on the Internet, around 77.9% of the content on the web is in the top ten most popular languages (English, Chinese, Spanish, Arabic, Portuguese, Japanese, Malay, Russian, French, and German)[5]. Therefore without the use of a translator, the information on the Internet is inaccessible, and somewhat useless, to people who can't read or understand at least one of these popular languages.

As an example, the Rwandan government, as part if its Vision 2020 plan, wanted to implement educational initiatives that would take advantage of free resources available on the Internet but most of these initiatives have been hindered by the fact that almost 90% of Rwandans are only proficient in Kinyarwanda [18] and there doesn't exist a competent translation service that can be used to translate to (or from) Kinyarwanda. As a Rwandan, after realizing this, I felt inclined to do something, I decided to explore possible technical solutions that could be used as a starting point in order to deal with this particular problem of information scarcity caused by linguistic barriers.

My first step when embarking on this project was to search around and see if there is any other work that has been done in this domain and to find out if there are any already-developed tools that I could use to translate any of the popular language to Kinyarwanda or vice-versa. After failing to find any competent service that supported Kinyarwanda translation[1], I started looking for research being done in the domain of natural language translation, and that is how I found out about Statistical Machine Translation (SMT) systems.

Before diving into the actual process of making a Kinyarwanda — English translator, I researched and tried out some of the freely-available SMT systems such as Moses (a complete SMT system), Cdec (a Python-based SMT decoder and aligner), and Joshua (a java-based SMT decoder).

For building my model, I ended up using Moses for reasons discussed in my concluding remarks. In the process, I also learnt about the phrase-based model that is used heavily in the field of statistical machine translation[13]. In this paper, I will explain how that model works and how it is implemented and how it is used in the above-mentioned SMT systems, while focusing on Moses and its role in my project.

[1]Kinyarwanda is one the langauges currently in development at Google Translate, but it is not clear whether it will be one of the supported languages anytime soon[2].

Chapter 2

Statistical Machine Translation

Even though there are many ways that translation from one language to another can be done[20, p. 5-6], many machine translation (MT) scholars now consider statistical machine translation (SMT) as the most effective way for developing efficient large-scale translation systems[10, p. xi]. The main idea behind SMT is to use large text translations from one language to another (parallel corpora) to generate translation tables, and then use good-quality content (monolingual corpus) of the target language to train its model, that is teaching the system about the general structure of that particular language so that is the system is able to evaluate generated translations and pick the most likely among them by using set statistical parameters.

To produce a good translation model, you need to have a large dataset of parallel corpora of high quality to use when developing translation rules for your model. According to Franz Josef Och, an SMT pioneer who created started the Google Translate project, you need at least a parallel corpora of 50-200 million words[16]. You also need to have a very large monolingual corpus, more than a billion words according to Och, to use when training the target language model[16]. In addition to this you have to *tune* the translation model. *Tuning* is the process in which the translation performance of the model is optimized by using a different parallel corpus that is of greater or equal quality than the one that that was used in the training stage. Given these two, and other, steps involved in developing an SMT system heavily employ statistical machine learning techniques, it is only reasonable that I discuss the basics of statistical machine learning before diving into the specifics of SMT systems.

2.1 Statistical Machine Learning

After the development of digital computers during the last half of the 20^{th} century, there have been multiple attempts of using machines to perform several tasks such as speech recognition, text translation, and data analysis. The process of teaching a machine the basics how to perform a certain task, and then letting the machine learn the rest on its own, is called "machine learning"[19].

Statistical machine learning is a branch of machine learning that uses data to generate statistical models that can then be used as functions when modeling new data inputs. To develop an SMT system, one uses large amounts of corresponding texts from both the source language and the target language as inputs of a learning algorithm that then generates statistics of equivalence between the two bodies of texts in form of probabilities. These probabilities are then used as inputs of another program that does the actual translation.

The resulting translating system can take a string of text in the source language and uses the rules, in this case statistical probabilities generated by the first algorithm, to output the best possible translation according to its rules. This is a very simplified example as it doesn't include all the steps involved in training an SMT model using prominent SMT systems such as Moses[12], but it provides enough details to showcase how basic concepts from machine learning are applied and used by SMT systems and SMT-based translation models.

2.2 The Mathematics of Machine Translation

In this section, I explore the Mathematics behind some of the main SMT concepts as discussed in an article by Brown *et al.* entitled *"The Mathematics of Statistical Machine Translation: Parameter Estimation"*[7].

2.2.1 Probabilities in Translation

Suppose we are given an English sentence e, and we are trying to translate it to its corresponding Kinyarwanda sentence k, $P(k|e)$ is the probability that we will get the Kinyarwanda translation k when starting with the English sentence e. $P(k)$ is the

priori probability that k happens based on our system's Kinyarwanda language model. $p(e, k)$ is the joint probability that both e and k happen. If a Kinyarwanda sentence k is the output of our English to Kinyarwanda translator when English sentence e is the input, it is clear that e and k are related (not independent), therefore their chances of happening depend on each other. Thus,

$$P(e, k) = P(e) * P(k|e) \tag{2.1}$$

In the same fashion,

$$P(e, k) = P(k) * P(e|k) \tag{2.2}$$

The above two equations can be combined to get the Bayes' Theorem equation:

$$P(k|e) = \frac{P(k)P(e|k)}{P(e)} \tag{2.3}$$

The goal of the English to Kinyarwanda translation system is to return the best translation, that is finding the translation k_{best} that maximizes the probability $P(k|e)$ for any given input English sentence e as shown in the equation below:

$$\text{argmax}_e(P(k|e)) = \text{argmax}_e(P(k)P(e|k)) \tag{2.4}$$

Notice that $P(e)$ is eliminated in equation 2.4 as it is a constant. The probability that sentence e is in English doesn't change as we go through the translation process because e it is the sentence we are trying to translate. Now that we have equation 2.4 that maximizes $P(k|e)$, we can use it to learn more about how our English to Kinyarwanda translation model works. As an example, let's say we want to use our English to Kinyarwanda SMT-based model to translate the English sentence: "*cats and dogs are both mammals*". To translate this sentence, our translation model would have to go through these important three steps:

1. Use a trained translation system to generate all possible Kinyarwanda translations.[1]

2. For each Kinyarwanda translation, k, calculate:

[1]This can be done using an SMT system such as Moses. The process of training and using a bilingual translation system using Moses is explained in Chapter 3.

- $P(k)$, the priori probability that k is indeed a Kinyrawnda sentence.
- $P(e|k)$, the reverse probability that if we started with the Kinyarwanda sentence k, we would get e, the English sentence that we are trying to translate.
- Calculate the product of $P(k)$ and $P(e|k)$.

3. Output k_{best}, that is the sentence k that has the biggest $p(k|e)$. Remember that from equation 2.4, when maximized, $p(k|e)$ is equivalent to the maximized product of $p(k)$ and $p(e|k)$ as $p(k)$ doesn't change during the translation process.

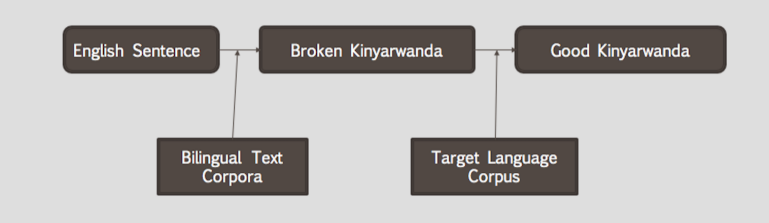

Figure 2.1: An illustration of how SMT works. A graphical representation of equation 2.4.

As it can be seen from the above example, Equation 2.4 is very important. In their article, Brown *et al.* described it as the "Fundamental Equation of Machine Translation"[7, p. 265] as it is capable of summarizing all the three steps involved in translating any given sentence from one language to another by using an SMT-based translation model.

At this point I will focus on the second step of the above-described process as the first and third steps can be easily implemented by a computer program that has access to the date of both the translation model and the target language model, and these two models can be generated by using an SMT system such as Moses in a process that I describe in chapter 3. However before addressing step 2, let's go back and ask one important question. **Why is it even necessary to to care about $P(k) * P(e|k)$? Can't we just directly calculate $P(k|e)$ if all we are trying to find is the Kinyarwanda sentence k_{best} that maximize the value of $P(k|e)$?**

Well the answer to that question is rather a devious one. To understand it, we have to look at this question as if we were computer programs. A computer program whose goal is to output the best possible translation k when given an English input sentence e will only care about two things:

1. The quality of the output sentence k in the target language (Kinyarwanda in this example).

2. The connection of that output sentence k to the input sentence e that it started with, that is how related are e and e compared to other sentence pairs seen by the translation model in the training and tuning stages.

However, unlike humans, a computer program can't just look at a sentence and decides if it is of good Kinyarwanda quality. It can't also deduce by just looking at two or more Kinyarwanda sentences, which one is the better translation for a given English sentence. Thus, when in a situation like, a machine program has no choice must reason backwards, at least that is how SMT-based systems are programmed to handle tasks like this one. As long the translation system has access to the translation tables from which it can generate several translations, and assigns to each one of them an exact conditional probability that if we started with that translation, and used the same translation system only going backwards, that is from the target language to the source language, we would get the original English sentence e that we are trying to translate to Kinyarwanda. Thos conditional probability combined with the priori probability $p(k)$ enables the translation system to maximize the chance of getting the best possible translation k.

For example, the correct Kinyarwanda translation of the English sentence (e): "*cats and dogs are both mammals*", is (k): "*injangwe n' imbwa ni inyamabere zombi*", with cats = injangwe, and = n', dogs = imbwa, are = ni, both = zombi, mammals = inyamabere. However, we can imagine that our translator, being very smart as it is, may generate the following translation: "*injangwe n' imbwa ni inyamaswa zombi*" (k'), which is the translation to "*cats and dogs are both animals*"(e'). So even though translation (k') is a good-structured and grammatically-correct Kinyarwanda sentence and would have a high $P(k')$, it is not the correct translation of (e), so its conditional

probability, $P(e|k')$, would be low, thus lowering the overall chance that sentence (k') is going to picked as the output by the translator.

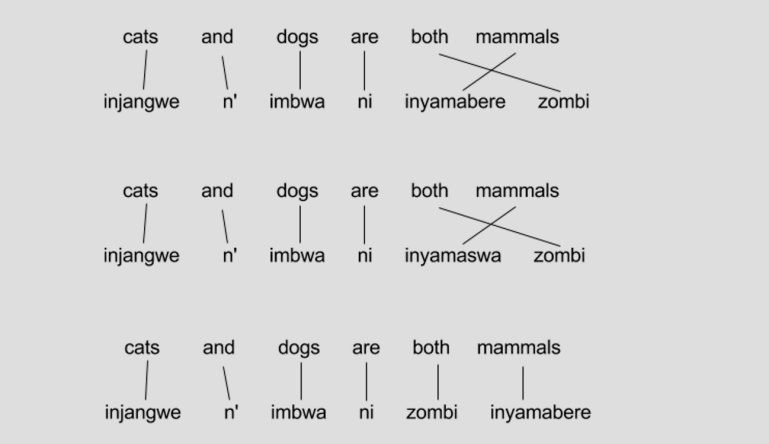

Figure 2.2: Comparison of several possible Kinyarwanda translations of the same English sentence: "*cats and dogs are both mammals*".

On another hand, we can imagine a misaligned version of sentence k. An example of this case is a sentence like "*injangwe n'imbwa ni zombi inyamabere*"(k'') which is a one-to-one translation of sentence (e) with each word with the words being in the same posistion as they were previously in e. Sentence k'' would likely be judged by the translation model as a very good translation. However, if our translation system's Kinyarwanda model was good enough, it would detect the inconsistency in the order of words in (k'') compared to other Kinyarwnda sentences, and $P(k'')$ would be low compared to $P(k)$; however training SMT language models to be very good at detecting minor inconsistencies like this one requires large (more than a billion words) monolingual corpora[16].

Back to our English to Kinyarwanda translation model, we can see that even though the combination of $P(k)$ and $P(e|k)$ is used to make sure that the results of our translation system is as correct as it can get, we can imagine some instances in which the two probabilities may contradict each other, that is when a bigger $P(k)$ means getting a lower $P(e|k)$ or vice versa. In these cases, it is recommended to add weight coefficients to equation 2.3. Added coefficients depend on the data used in the training and tuning stages, and on the structure of the languages being dealt with.

2.2.2 Alignments

As different languages have different structures, and one word from the source language can be translated using zero, one, or multiple words in the target language, it is important that we account for all these possibilities caused by changes in both the structure, and the *fertility* of words in, the language pairs we are working with. The *fertility* of a word is defined as the number of words that are generated by that particular word when translated from one language to another. To better understand this let's use the following example of a Kinyarwanda - English sentence pair.

```
Kinyarwanda: ‘‘Imana yirirwa ahandi igataha i Rwanda.’’
English: ‘‘God spends the day elsewhere but spends the night in Rwanda.’’
```

From the above translation, one can immediately deduce that the Kinyarwanda words in the above sentence have higher fertility rates when being translated to English; however, using this example and doing the translation from English to Kinyarwanda, the English words would have lower fertility rates.

In the above example, as it can be seen in figure 2.3, the word "Imana" has a fertility of 1 as it generates one English word, "God". On another hand, the word "yirirwa" has a fertility of 3, as it generates a total of three English words ("spends", "the", "day"). The relationship between Kinyarwanda words and English words illustrated in Figure 2.3 can also be represented using the Brown *et al.* notation[7, p. 266] as following: *(God spends the day elsewhere but spends the night in Rwanda | Imana(1) yirirwa(2, 3, 4) ahandi(5) igataha(7, 8, 9) i(10) Rwanda(11))*. In this notation, the numbers in parentheses represent the indexes that the Kinyarwanda words are equivalent to. As you may have noticed, index 6 is skipped because the English word "but" is not connected to any particular word in the Kinyarwanda sentence even. It is implicitly implied with the structure of the sentence. Such words that are generated somewhat out of nowhere are called "*spurious*" words[8, p. 2].

However the opposite can also happen. We can imagine translating a Kinyarwanda sentence to English, and losing one of its words through the translation process. In that scenario, that "*lost*" word may have been combined with another word and then translated together, or it may have been lost and just been translated through the context of the produced sentence. The "lost" word scenario is featured in fig 2.4.

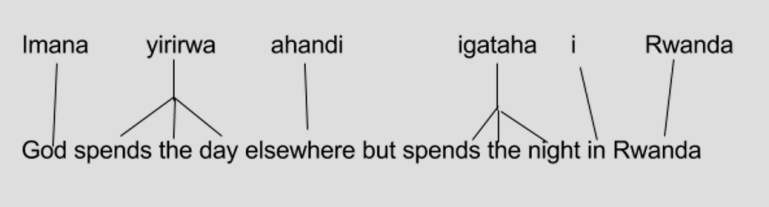

Figure 2.3: An example of sentence alignment with a "spurious" word: "*but*".

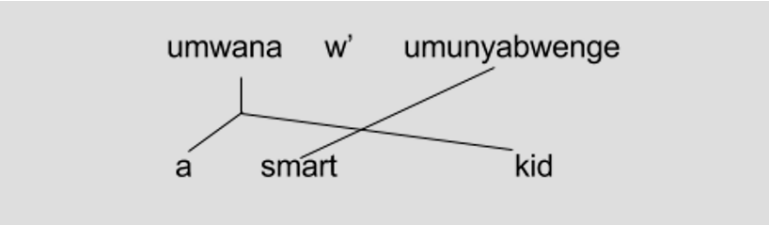

Figure 2.4: Example of a sentence alignment with a "lost" word: "*w'*".

However most sentence pairs that are dealt with during the training stage or the translation process do not just contain spurious words or lost words but a mixture of both (e.g.: Figure 2.5); and that is why alignment probabilities should be assigned to these two possibilities when developing a translation model. Depending on which language pair you are dealing with, "lost" and "spurious" words can have a very significant impact on the quality of the translation results. They ought to be taken seriously.

According to Brown *et al.* [7, p. 268], the way that SMT decoders use to handle these irregular cases of alignments is by adding one empty *cept*[2] to the sentence about to be translated, and using it as an alignment partner for any "spurious" word found in the final translation. As for "lost" words, Brown *at al.*[7, p. 268] suggest using their implied meaning or context to align them with their closest or related counterparts in the target language sentence.

By applying these suggestions to the example illustrated in Figure 2.4, we would get the following alternative alignment: *(a smart kid | k_0(1) umwana(1) w'(2) umunyabwenge(2))*, where k_0, stand for the null cept that we placed at the beginning of

[2]In this case, *cept* is defined as a word or a group of words whose context is translated together.

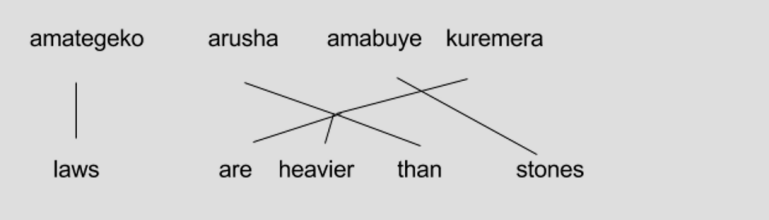

Figure 2.5: A sentence pair alignment containing both "lost" and "spurious" words.

the Kinyarwanda string before translating it to English. Also notice how instead of skipping the string *"w"'*, we now combine it with the string *"umunyabwenge"* into a context-based cept *"w' umunyabwenge"*, and then translate both words together together to the English cept, *"smart"*. In this case, the cept is just one word, but we can imagine cases in which it is empty or contains multiple words. One important

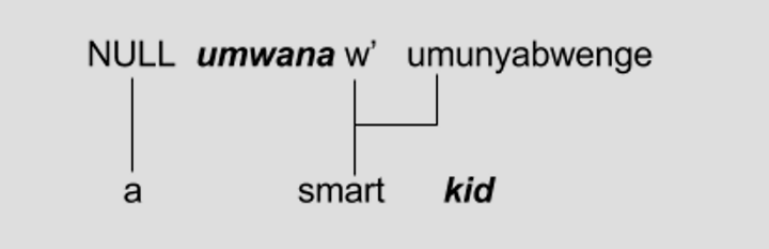

Figure 2.6: An alignment in which *"w' umunyabwenge"* is aligned with "*smart*"

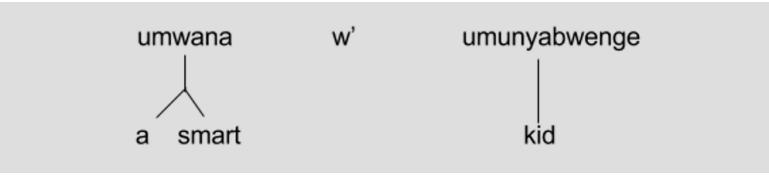

Figure 2.7: A graphical representation of a less likely (incorrect) alignment of the sentence pair: "*a smart kid | umwana w' umunyabwenge*".

question that arises as these two suggestions are implemented together is that there are many alignments for each translation pair. If you showed the alignments in Figures 2.4 and 2.5 to a group of ten people, it is unlikely that they all would agree that

one alignment is better than the other one. Based on this, one can imagine what an SMT-based translation model is going when presented with not one, but 2^{lm} possible alignments for each sentence pair with l being the number of words in the source language sentence, and m, the number of words in its target language sentence.

To handle this challenge of having several possible alignments for every sentence pair, according to Brown *at al.*[7, p. 269-270], an SMT model has to assign a likelihood probability to each possible alignment. For example, for the *(a smart kid | umwana w' umunyabwenge)* translation pair, both the alignments in Figures 2.4 and 2.5 would have bigger probabilities while alignments such as *(a smart kid | umwana(1, 2, 3) w' umunyabwenge)* and *(a smart kid | umwana(1) w'(2) umunyabwenge(3))* would have lower probabilities as they are incorrect, thus less likely to be considered by the translation model.

2.3 Phrase-Based Translation Model

Several SMT systems now use a phrase-based translation model to build phrase translation tables that are more effective than tables generated by word-based alignments [13, p. 48]. The reason why phrase-based models outperform word-based models is because phrase-based models break each sentence into phrase segments that are then translated and reconstructed later into the target language. This gives phrase-based translation systems the ability to account for contexts and meanings that words have used alongside some other particular words as opposed to when used alone or with different words. This is taking advantage of statistical probabilities that were in generated from the parallel corpora data used during the training and tuning stages.

An obvious case in which a phrase-based model would perform better than a word-based model is in the translation of sentences that contain idiomatic expressions. As an example, translating an expression like *cutting corners*[3] using a word-based translation system would result in the loss of contextual meaning that the words *cutting* and *corners* have when used together in the expression *cutting corners*.

.

[3]To cut corners is "to do something in the easiest, cheapest, or fastest way"[1].

Chapter 3

Using Moses

Moses is one of the most used SMT systems. It is a complete SMT system with a built-in decoder that can be used with several alignment algorithms. Moses is the SMT system that I used to train my English to Kinyarwanda translation model. In the next section, I describe the steps I followed to develop, train, and test my model using Moses.

3.1 Building a Baseline System

After successfully installing Moses and other required software (Giza++, Boost, etc), I used it to train my English to Kinyarwanda translation model by using the *King James Version*(KJV) Bible as my English corpus, and the *Bibiliya Yera*, the Kinyarwanda translation of the KJV, as my Kinyarwanda corpus.

The following is an excerpt of the first three verses of the New Testament in both the Kinyarwanda *Bibiliya Yera* and the English *King James Version* Bible (Matthew 1: 1-3).

```
1:1 Amasekuruza ya Yesu Kristo, mwene Dawidi, mwene Aburahamu ngaya: | 1:1
    The book of the generation of Jesus Christ, the son of David, the son
    of Abraham.
1:2 Aburahamu yabyaye Isaka, Isaka yabyaye Yakobo, Yakobo yabyaye Yuda na
    bene se, | 1:2 Abraham begat Isaac; and Isaac begat Jacob; and Jacob
    begat Judas and his brethren;
1:3 Yuda yabyaye Peresi na Zera kuri Tamari, Peresi yabyaye Hesironi,
    Hesironi yabyaye Ramu, | 1:3 And Judas begat Phares and Zara of Thamar;
     and Phares begat Esrom; and Esrom begat Aram;
```

As you can see from the above excerpt of a parallel corpus, you need to have two files two equivalent texts in two languages: the target language and the source language. The content in those two files need to correspond line by line. Line 100 in the target language's file should be the translation of line 100 in the file containing the source language. In my case, as I was trying to build an English to Kinyarwanda translation system, I started with two different files, one containing the *Bibiliya Yera* and the other one containing the *King James Version* Bible. When using Moses, the first step in training a translation model is a process called *"Corpus Preparation"*.

3.1.1 Corpus Preparation

Corpus Preparation consists of three steps: tokenization, truecasing, and cleaning.

During tokenization, spaces are added between all words and punctuation to make sure that different forms of the same word are counted as one. Without this step, the string "God", for example, would be considered to be different from the string "God!". To avoid this kind of behavior that would result in *data sparsity*, the *Moses tokenizer* adds an extra space in "God!" to make it "God !"; thus, when frequencies of the word "God" are calculated, the Moses engine is able to return a correct count that contains all the occurrences of "God!", "God,", etc.

In the next step, Moses uses a *truecasing* script, also known as the *truecaser*, to calculate the frequencies ratios of how many times a certain word is lower-cased compared to when it is capitalized. This is important as without this step, it would be almost impossible for the translation system to guess if the words at the beginning of a sentence are capitalized because they are normally capitalized (proper names) or if they are capitalized just because they are the at the beginning of their respective sentences.

For example, the word "God" and "god" have two different meanings and those two meanings should be handled correctly by the translator. However, when "God" is used at the beginning of a sentence, the translator can't know for sure if it is capitalized because it is meant to be "God" or if it is the word "god" that is capitalized because it is at the beginning of a sentence. In this case, the *Moses truecaser* trades off the 100% uncertainty for a statistical guess based on the frequencies of both strings in the corpus. In an instance in which the frequency of the word "God" in the corpus

is less than that of the string "god", all occurrences of "God" at the beginning of sentence will changed to "god", otherwise there will be no change. This is done for all words in the corpus and it helps to ensure that all meanings associated with capitalization are not all lost during the translation process.

The last but very important step is the cleaning step. In this step, a sentence pair is removed from the training data if one of its sentences has a character count greater than a set amount, or if the ratio of the character counts of its sentences is not proportional to the calculated or set ratio for the training data. The limiting character count is set according to the structure of the languages that are being dealt with or the quality/size of the parallel corpora being used.

In my case, I set to the character count to 180, which is very high compared to the ones used in the Moses tutorials that I looked at[11, 3]. The reason why I set the limiting character count of my translation system to a larger number is because Bible verses tend to be very long than normal sentences used in conversations and discussions, two major sources of data used in the above-mentioned Moses tutorials.[11, 3].

In addition to that, I am confident that the verse-by-verse translations that I used to train my model are correct as they are from the Bible, and thus have been proofread by humans. In addition, as the main goal of this step is to remove incorrect translations, and I am more than confident in the literary accuracy of training data I used, setting a lower limiting character count for my translation model would not had improved my model at all, it would actually have caused more data scarcity, and that would reduce the overall accuracy of my model.

Below are the first three verses (Matthew 1: 1-3) from the *King James Version* Bible after going through the three steps of the *"Corpus Preparation"* process.

```
1 : 1 The book of the generation of Jesus Christ , the son of David , the
    son of Abraham .
1 : 2 Abraham begat Isaac ; and Isaac begat Jacob ; and Jacob begat Judas
    and his brethren ;
1 : 3 And Judas begat Phares and Zara of Thamar ; and Phares begat Esrom ;
    and Esrom begat Aram ;
```

3.1.2 Language Model Training

Target language model training is the following step after corpus preparation. In this step, I used Moses' built-in KenLM 3-gram model tool to construct a target language mode based on my corpus. In this case, as I am working with an English to Kinyarwanda translation, Kinyarwanda is my target language, thus I am going to use the *Bibiliya Yera* corpus file generated the *truecaser*. At this point, there is no need of using the output generated after the cleaning stage of the *Corpus Preparation* process as a language model only depends on the structure of the target language being used, Kinyarwanda in this case, and not to its equivalent translation to the source language, English in this case. Therefore there is no need to take into consideration the effects of the sentence character count and the limiting ratio used to filter data in the cleaning stage of the *Corpus Preparation* process. After building the Kinyarwanda language model, I used the *Moses binarizing script* to turn the file containing the Kinyarwanda language model into a binary version that loads faster. At this step, I can use it to get the probability that any input sentence is part of the Kinyarwanda according to the language model that I built using data solely from the *Bibiliya Yera*.

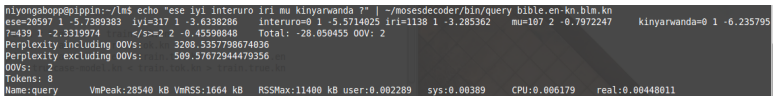

Figure 3.1: Checking if the sentence "ese iyi nteruro iri mu kinyarwanda ?" ("is this sentence in kinyarwanda ?") is indeed in Kinyarwanda according to a language model developed using corpus text from the *Bibiliya Yera*.

3.1.3 Training the Translation System

Now that we have trained our target language model, it is time to start training the translation system. For this step, I used Moses default word-alignment tool called Giza++. After running the commands for this step, *Moses* generated a `Moses.ini` configuration file that can be used to translate any English sentence to Kinyarwanda. However, at this point, there are two main issues that must be looked at. The first one is that translation takes a long time; to fix this we need to binarise the *phrase*

and reordering tables. The second one is that weights in our model configuration file are not adjusted, i.e.: they are dependent to the Bible data we used in training the model. In the next subsection, I talk about the process of tuning our model to make it more balanced and less dependent on the data used to train it.

3.1.4 Tuning the Translation Model

To tune our model, we need a separate small parallel corpus of high quality to add statistical variety to our model. The tuning process is important as it helps in making the translation model more balanced and less biased towards the data that was used during the training step. For example, as we only used data from the Bible, our model has only seen sentence structures, expressions, and words' combinations that are common in the Bible and nothing else from the commonly used conversational and/or formal language. Therefore, to make our model more efficient, it is crucial that we tune it using data that is very different from the Bible. As I couldn't easily find good-quality English data that is also translated in Kinyarwanda, I used the Rwandan constitution as it is freely-available on the web both in Kinyarwanda and in English. In the tuning process, as in the training process, I completed the steps of the *corpus preparation* process on the tuning corpus data before using it. I ran the tuning process by using the minimum error rate training (MERT) algorithm[15], the default option in *Moses*[4]. I also performed this step in a newly created folder to avoid overwriting my previous model's configuration files.

3.1.5 Binarising Phrase and Reordering Tables

Once the tuning process is over, it is advised to binarise the *phrase and reordering tables* in your translation model by using the Moses' built-in tools specialized for this process[3].

1. Change `PhraseDictionaryMemory` to `PhraseDictionaryCompact`
2. Set the path of the `PhraseDictionary` feature to point to `$HOME/working/binarised-model/phrase-table.minphr`
3. Set the path of the `LexicalReordering` feature to point to `$HOME/working/binarised-model/reordering-table`

Figure 3.2: Examples of changes made to the `Moses.ini` configuration file to make sure it correctly points to the *binarised* tables and the *binarised* model directory.

3.1.6 Testing the Translation Model

Now that we have completed the basic steps of building, training, and tuning a translation model using *Moses*, we can use it to do some simple translations. To do this, all you have to do is running terminal command, and that way you can translate a file containing sentences in English to Kinyarwanda. The sentences in the input file have to be in the same format as the ones in parallel corpora used in the training and tuning stages.

Below is the content of the English input file that I used to test my translation system followed by the Kinyarwanda file content that was generated the translator.

```
1  I am a computer science student
2  Augustus is the son of Ceasar
3  Augustus is the adopted son and heir of Ceasar
4  The word of the Lord.
5  In the beginning was the Word, and the Word was with God, and the Word was
       God.
6  In the beginning was the Word,
7  and the Word was with God,
8  and the Word was God.
9  beginning was the Word
10 the Word was with God
11 the Word was God
12 the word was god
```

```
1  ndi computer b student
2  Augustus &apos; umuhungu Ceasar
3  Augustus &apos; adopted mwene na atandukanaho mwene Ceasar
4  "" ry &apos; Lord.
5  "" mbere yari Word, Uwiteka Jambo God, n &apos; Jambo yari God.
6  "" mbere yari Word,
7  Uwiteka Jambo n God,
8  kandi Jambo yari God.
9  mbere Jambo
10 Jambo yari Imana
11 Jambo yari Imana
12 ijambo yari imana
```

3.2 Translation Results

As it can be seen from the results on the previous page, my translation system is still struggling with the translation of content that is not from the Bible. It can also be seen that it is almost incapable of translating long sentences as the one in line 5. However, I am still hopeful that it can be improved by using more data. The data I used to train and tune my translation model is like a drop in the ocean compared to the recommended quality and size as stressed by Och[16].

It should also be noticed that even though the quality of my translation results if of low quality, the results I am getting are consistent with the parallel corpora I used to train and tune my translation model. As I used the Bible in the training process, one should expect my model to do well on translating sentences from the Bible. To a certain degree, that is what my model is doing but with an slightly lower accuracy rate. I suspect that is mainly caused by two reasons.

The first one is that SMT systems work by using mathematical probabilities and decoding algorithms and not by directly looking up of each sentence in a large catalog of translations. The benefits of this is that SMTs systems are capable of translating sentences they have never seen before. However this comes at a cost that in some instances, the SMT model will return an incorrect translation for a sentence contained in the data that was used to train it.

Listing 3.1: Translation output of the English text on page 18 when using a translation model both trained and tuned using content from the Bible.

```
1  ndi computer science student
2  Awugusito ari mwene Ceasar
3  Awugusito ni adopted mwene n &apos; atari w &apos; Ceasar
4  "" Ijambo ry &apos; Lord.
5  "" Muri mbere na mbere Word, n &apos; Jambo yari kumwe n &apos; God, n &
       apos; Jambo yari God.
6  "" Muri mbere na mbere Word,
7  kandi Jambo yari kumwe n &apos; God,
8  kandi Jambo yari God.
9  mbere na mbere Jambo
10 hariho Jambo ; Jambo uwo yahoranye n &apos; Imana
11 Jambo yari Imana
12 ijambo ry &apos; imana yari
```

The second reason why my translation isn't doing well as expected at translating sentences from the Bible is because I used data from the Rwandan Constitution in the tuning process. I did that hoping that tuning my translation model using external data would help improve the overall performance of the model, however it is hard to judge that with the results I have right now because my corpora sizes are too small to have a significant impact. The only thing that I am sure of is that tuning reduced the accuracy of my model for translations of sentences from the Bible. This is illustrated by the sentences in the example in Listing 3.1 which are better translations of the input file than the results displayed on page 18.

Another important thing to notice is that my translator, like other SMT-based translation systems, is incapable of inferring the difference of homonym words. For example, when I tried to translate "Its not that Im so smart, its just that I stay with problems longer", a quote attributed to Albert Einstein, my translator translated the word "just" as "umukiranutsi", the Kinyarwanda translation for "a fair" or " an impartial" person! This shows that even though SMT-based translation systems perform better than rule-based systems, SMT is still inferior to human translation especially when it comes to these kind of situations in which words with the same spelling have more than one meaning.

Chapter 4

Concluding Remarks

First of all, before talking about what I thought about the overall progress of this project and what I think of the English to Kinyarwanda translation model that I current have, I think it is important that I first talk about some important decisions that I have had to make when working on this project.

- **Why English to Kinyarwanda?** Why not Kinyarwanda to English? The reason why I chose to work on Kinyarwanda to English translation model and not the other way around is because of need. Most Rwandans who would want to use a Translator would most likely use to translate content (books, articles, manuals instructions, etc.) from English to Kinyarwanda or from other popular languages to Kinyarwanda. I chose to work with English because it is one of the three official languages in Rwanda, the others being French and Kinyarwanda. English is also the language used in education, from primary schools to universities[18]. As for the reason I chose to work on an English to Kinyarwanda translator, and not the other way around, that can be explained by the fact that Rwanda had been an oral society for most of its existence, and there aren't so many writings in Kinyarwanda that one would possibly want to translate to another language[6, p. 59]. Even the majority of the ones that exist aren't digitalized or freely-available online, which makes using them or simply accessing even more challenging.

- **Why the Bible and the constitution?** I chose to use the Bible as my training data because it is the only large body of text that has both English and Kinyarwanda translations that I could find online. I used the Rwandan constitution for tuning as it is an official document that has good-quality content for both Kinyarwanda and English. I also used it because, as a non-religious text, it offers a good variation to the content from the Bible that I had used in the training process.

- **What is the main challenge that I faced?** The main challenge that I faced is data sparsity. Kinyarwanda is not a popular language by any metric. Even though it is estimated that Kinyarwanda is spoken by around 20 millions people, the majority of them live in Rwanda and its neighboring countries (Burundi, Uganda, Tanzania, and Democratic Republic of the Congo), and it isn't used elsewhere in the World[9]. That and the fact that Rwanda used to be an oral society and still is to some degree, guarantee that getting Kinyarwanda data or Kinyarwanda software is almost impossible. According to Samuelson *et al.*, despite its massive use in Rwanda and neighboring regions, "mass literacy in Kinyarwanda remains weak"[18]. One of the consequences that I faced when working on my project is that Moses, the SMT system that I used didn't even have a Kinyarwanda *tokenizer*. I had no choice but to use the default (English) one and hope for the best!

- **Why statistical machine translation?** I chose to use SMT as opposed to rule-based translation because SMT systems are easy to work with and easily scalable compared to rule-based translation models[16]. As my end goal for this project is to turn it into an open-source system that other people can contribute to, scalability was very important in taking this decision. I thought that implementing the translator using an SMT system would make it possible and easier for people, with different levels of technical expertise and Kinyarwanda proficiency, to contribute to the project either by providing more training data (in forms of translated texts), correcting translations generated by the translator, or writing code for translation tools specialized for Kinyarwanda.

- **Why Moses?** As I did not have any previous experience working with SMT systems, this wasn't an easy choice. I had to try all the popular SMT systems and decoders and decide which one to use. I tried Joshua, Cdec, and Moses. Unfortunately, Joshua and Cdec ran into compiling issues on my computer, and despite my multiple attempts, I couldn't get them to work past the tutorial stages found on their websites. With that, I had only one choice left: Moses. However, even though I had no other choice when I started using it, through the process of using Moses to develop my English to Kinyarwanda translation model, I have come to appreciate the fact that it has a lots of documentation and supporting documents that are easily-accessible and freely-available online.

- **What is next?** For now, my next goal is to try to get as much Kinyarwanda data as possible. As I already mentioned, it is not that difficult to get English data. The real challenge is getting Kinyarwanda data in right format (`.txt` file, line by line sentences, character count not too small or too large, etc.). However, what is even more difficult is getting Kinyarwanda data that has correct, human-proofread, line by line, equivalent English translation. Regarding this, my plan going forward is to set up a website that will crowd-source translations by generating English text line by line and asking users to translate it. Hopefully, I will get a decent amount of data this way. In addition to this, I intend to keep looking for and collecting English | Kinyarwanda data and adding it to my training corpus, that way I hope to keep improving my model.

To sum up, having asked and answered these important questions, all I have to add is that despite all the challenges I have faced when working on this project, I have more determination now more then ever. I am more determined because I am learning a lot by working on this project, and because I know that as long I keep working on this project, the quality of my translation model results will keep improving.

Appendix A

Code Reference: `Moses.ini`

`Moses.ini` file generated by my English to Kinyarwanda translation model after training and tuning[1] it using data from the Bible [2] and the Rwandan Constitution respectively.

```
1  # MERT optimized configuration
2  # decoder /export/home/s14/niyongabopp/mosesdecoder/bin/moses
3  # BLEU 0.811884 on dev /export/home/s14/niyongabopp/corpus/train.true.en
4  # We were before running iteration 5
5  # finished Mon Dec 12 14:25:47 CST 2016
6  ### MOSES CONFIG FILE ###
7  #########################
8
9  # input factors
10 [input-factors]
11 0
12
13 # mapping steps
14 [mapping]
15 0 T 0
16
17 [distortion-limit]
18 6
19
20 # feature functions
21 [feature]
```

[1] Using the minimum error rate training (MERT) tuning algorithm.

[2] The *King James Version* Bible for English and the *Bibiliya Yera* for Kinyarwanda.

```
22 UnknownWordPenalty
23 WordPenalty
24 PhrasePenalty
25 PhraseDictionaryMemory name=TranslationModel0 num-features=4 path=/export/
       home/s14/niyongabopp/working/train/model/phrase-table.gz input-factor=0
        output-factor=0
26 LexicalReordering name=LexicalReordering0 num-features=6 type=wbe-msd-
       bidirectional-fe-allff input-factor=0 output-factor=0 path=/export/home
       /s14/niyongabopp/working/train/model/reordering-table.wbe-msd-
       bidirectional-fe.gz
27 Distortion
28 KENLM name=LM0 factor=0 path=/export/home/s14/niyongabopp//lm/bible.en-kn.
       blm.kn order=3
29
30 # dense weights for feature functions
31 [weight]
32
33 LexicalReordering0= 0.0511426 0.0255929 0.0391846 -0.280932 0.0141974
       0.0215455
34 Distortion0= 0.112974
35 LM0= 0.0269908
36 WordPenalty0= 0.0333231
37 PhrasePenalty0= 0.0332891
38 TranslationModel0= 0.0467293 0.0066613 0.0107129 0.296725
39 UnknownWordPenalty0= 1
```

Bibliography

[1] Definition of "Cut Corners". http://dictionary.cambridge.org/us/dictionary/english/cut-corners. Accessed: 2016-12-10.

[2] Google Translate Community. https://translate.google.com/community. Accessed: 2016-12-05.

[3] Moses - Baseline System. http://www.statmt.org/moses/?n=Moses.Baseline. Accessed: 2016-12-12.

[4] Moses - Factored Training and Tuning. http://www.statmt.org/moses/?n=FactoredTraining.Tuning. Accessed: 2016-12-12.

[5] Top Ten Internet Languages - Word Internet Statistics. http://www.internetworldstats.com/stats7.htm. Accessed: 2016-12-05.

[6] Julius Adekunle. *Culture and Customs of Rwanda*. Greenwood Publishing Group, 2007.

[7] Peter F Brown, Vincent J Della Pietra, Stephen A Della Pietra, and Robert L Mercer. The mathematics of statistical Machine Translation: Parameter Estimation. *Computational linguistics*, 19(2):263–311, 1993.

[8] Ulrich Germann. Greedy Decoding for Statistical Machine Translation in Almost Linear Time. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, pages 1–8, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics.

[9] Lwaboshi Jacques Kayigema. Loanword Allocation in Kinyarwanda. 2010.

[10] Philipp Koehn. *Statistical Machine Translation*. Cambridge University Press, 2009.

[11] Philipp Koehn. Moses - Statistical Machine Translation System - User Manual and Code Guide, 2010.

[12] Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. Moses: Open Source Toolkit for Statistical Machine Translation. In *Proceedings of the 45th annual meeting of the ACL on interactive poster and demonstration sessions*, pages 177–180. Association for Computational Linguistics, 2007.

[13] Philipp Koehn, Franz Josef Och, and Daniel Marcu. Statistical Phrase-Based Translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 48–54. Association for Computational Linguistics, 2003.

[14] Sanjaya Lall and Morris Teubal. "Market-Stimulating" Technology Policies in Developing Countries: a Framework with Examples from East Asia. *World development*, 26(8):1369–1385, 1998.

[15] Franz Josef Och. Minimum Error Rate Training in Statistical Machine Translation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 160–167. Association for Computational Linguistics, 2003.

[16] Franz Josef Och. Statistical Machine Translation: Foundations and Recent Advances. *Tutorial at MT Summit*, 2005.

[17] James Brian Quinn. The Impacts of Technology in the Services Sector. *Technology and Global Industry*, 1987.

[18] Beth Lewis Samuelson and Sarah Warshauer Freedman. "Language Policy, Multilingual Education, and Power in Rwanda". *Language Policy*, 9(3):191–215, 2010.

[19] Rob Schapire. Theoretical Machine Learning. `http://www.cs.princeton.edu/courses/archive/spr08/cos511/scribe_notes/0204.pdf`. Accessed: 2016-12-05.

[20] Arturo Trujillo. *Translation Engines: Techniques for Machine Translation*. Springer Science & Business Media, 2012.