17

Automatic Construction of a Kiswahili Corpus from the World Wide Web

Katherine Getao and *Evans Miriti*

A corpus is a large collection of language data either in written form or spoken form or both. It can be used to construct a language model that is used in many language technology applications. Some of these include speech to text, optical character recognition, machine translation and spell checking. The easiest way to create a text corpus is by putting together electronic text documents. For most languages, getting a huge collection of electronic texts is a time-consuming and challenging task. The monotonous nature of such a task will inevitably lead to much less attention being paid to the errors that might find their way into the text collection. This paper describes the working of an application that was used to build a Kiswahili corpus from the Internet to be used in natural language processing applications.

Introduction: Statistical Natural Language Processing

Statistical natural language processing methods are popular because one does not have to spend a long time learning and discovering all the rules of a language. Considering that natural language is so dynamic it is almost impossible to come up with exhaustive rules.

To use statistical language processing, there is need for data from which statistical information can be derived. This data is usually provided in the form of corpora.

Text corpus

A text corpus is a large collection of language data in written form. An electronic text corpus is a text corpus in a computer-readable format. It can be used to obtain statistical information about the language. The statistical information can range from word counts to the more sophisticated N-gram models.

In many natural language processing applications, a text corpus for a specific language usually comes in handy for the construction of a language model for that language. Some of the language models that can be constructed for a specific language are N-gram models. These include unigrams, bigrams, trigrams etc.

N-gram models are used in many applications, including speech to text, spelling correction and optical character recognition.

A labelled text corpus can also be used as training data for machine learning-based POS applications.

Other uses include dictionary creation and machine translation using parallel texts.

Corpus construction

Constructing a corpus is a challenging task, particularly when the electronic documents to be included in the corpus are not readily available. Traditional methods of correcting electronic texts include typing in the specific texts and scanning the texts. These methods are time-intensive and accuracy is not guaranteed. It is therefore necessary to get alternative methods for construction of a text corpus.

One of the alternative methods currently being used is to construct corpora from the Web (Rosie Jones and Rayid Ghani, 2000). The web currently contains several billion of pages. By February 2004, Google announced that they had a database of 4.28 billion web pages. If these documents could be accessed and converted into the appropriate format, then they could be used for the construction of an electronic corpus.

Several papers have been written on how to go about constructing a corpus from the web. These include automatically building a corpus for a minority language from the Web (Jones and Ghani, 2000) and improving the quality of a web corpus (Youchi Sekiguchi and Kazuhinde Yamamoto, 2004).

Nevertheless, each language presents its own unique challenges. In this paper, we describe the process we went through to develop an application that can be used for the automatic construction of a text corpus and a language model for Kiswahili. Though this application was developed and tested for the Kiswahili language, it can be used for any other language with minimal modification.

Steps for Creating a Web Corpus

Overview

- We create initial language models, i.e. Kiswahili unigram and bigram counts, and a list of words for other languages
- We will use these models to determine if a document is in Kiswahili or another language
- Unigrams refer to a simple words while bigrams refer to pairs of words
- We will also use the initial Kiswahili model to construct search queries

1. Creating the initial Models

It is necessary to use the initial models to determine which language a document is in. The initial Kiswahili unigram and bigram model is created by providing a document that is in "good Kiswahili". "Good Kiswahili" here refers to the absence of foreign words rather than grammatical correctness. While subsequent documents are downloaded and classified automatically, the initial document is downloaded and classified manually and used to create the initial model (unigram and bigram lists). Documents in other common languages are downloaded and used to create the other languages' unigram model (word list). In our case, we used four documents that were in English, French, Spanish and German. These are the languages that intersected most with Kiswahili particularly in sites that are used to give Kiswahili lessons.

NB: These initial documents are also known as seed documents.

2. Getting Kiswahili documents links

The next step is to get the links of the documents to be used in the construction of the corpus from the Web. There are several ways of getting these links. One method is through the use of a search engine and the second is by using a Web crawler. When using a search engine, you provide a search query such that the search engine will return links for documents of the type required (that is documents in the preferred language). This is the method adopted in our application. In the Web-crawler approach, the application is given an initial page, it then gets the links on the page and uses them to download more pages and get more links. This process is repeated until it has enough links.

In the first method the search engine will return a set of pages each with links to documents found in the search. We need to retrieve and save these links from the search-result pages.

The search query is created automatically using the bigram in the Kiswahili bigram model with the highest frequency. This bigram is then marked so that it is not used again. Subsequent searches used the bigram with the highest frequency that had not been used previously.

3. Downloading and processing the files

After we have extracted the links, the next step is downloading the documents and processing them to the correct format for inclusion in the text corpus. The documents that are downloaded are .htm, .html and .txt files which are easier to process. In addition, they are the most numerous for the Kiswahili language (hence few documents are left out).

The objective of the processing is to obtain sentences that are in an acceptable format for inclusion in the corpus.

Since this process is automated, this cannot be perfectly accomplished, but the steps outlined significantly improve the quality of the text.

There are several processing steps that need to be carried out on a downloaded file to convert it to a proper format for inclusion in the corpus. Some of the processing steps carried out include:

Identification of sentences: This is done using tags, full stops, exclamation marks, and question marks.

Removal of special tags and the text in between: These include <style> <script> <xml> <a> and <title> tags. These will most likely contain ungrammatical sentences.

Removal of all html tags

Removal of sentences with numbers in the middle: The best technique would be to turn the numbers to text but we did not do this due to time constraints.

Removal of brackets [], (), {}, and text in between: This is because they tend to interfere with the grammatical correctness of a sentence.

Removal of sentences that are less than five words long: These type of sentences tend to be grammatically wrong.

Creating the Text Corpus and the Models

After a file is processed, it can then be added to the corpus and also used to augment one of the language models. Two models are created. One model is for the target language (Kiswahili) and the other model for the other languages. (Jones et al, 2000). Initially, the Kiswahili model consists of both a unigram frequently count and a bigram. The unigram model is created by getting each unique word in the file and adding it to a table in a database if it is not already there. The number of times the word appears in the text is also counted and added to its current count in the unigram's table. The bigram table consist of each two words that appear following each other in sentences in the text, i.e. word1 word2 and how many times they appear in that order in the texts considered so far (count). To be able to count how many times a word appears at the beginning of a sentence a unique word form is introduced at the beginning of each sentence. The bigram count of this word and other words indicates how many times the other word is used to start a sentence. These entries are used to calculate the unigram and bigram probabilities after the corpus construction process is complete.

The other model is the unigram model for words in other languages. This model is necessary because not all the documents downloaded will be in Kiswahili. This model is used in conjunction with the Kiswahili model to determine which language the document belongs to. In our case, we defined upper limit and lower limit constants. If the number of words in the document that are also in the Kiswahili unigram model exceeds the upper limit and, in addition, the number of words that are in the document and in the other model is less than the lower limit the document is in Kiswahili and can be added to the Kiswahili corpus. If the number of words in the document that are also in the other languages unigram model exceeds the upper limit and in addition the number of words that are in the document and in the Kiswahili unigram model is less than lower limit, the document is in other languages and can be added to the other languages corpus. Otherwise the document is deemed ambiguous and discarded. The setting of the upper limit and the lower limit constants is vital as it ensures that documents in other languages, e.g. English, do not do not find their way into the Kiswahili corpus. Their values can be adjusted until a desirable result is obtained. This also ensures that Kiswahili documents do not find their way into the other languagesmodel. If this happens, the application will discard most documents as it will not be able to tell which model it fits into. They also take care of the fact that there are many bilingual documents on the Web. These need to be discarded to avoid having a Kiswahili corpus replete with many words from other languages. These techniques are borrowed and adapted from *Learning a Monolingual Language Model from a Multilingual Text Database* (Jones et al., 2000).

Summary of the Web-Corpus Construction Process

1. Create the initial models

Repeat

2. Create a search query using the bigram with the highest count (frequency) in the Kiswahili language model

3. Make a get call to the search engine
4. Process the results of the search engine to get the links
5. Save the links

   /*Note that only .html, .htm and .txt files are saved*/

For each link

6. Download the file referred to by the link url.

   If the get is not successful continue to the next link.
7. Process the downloaded file so that you are left with only acceptable sentences.
8. Determine the language of the document.
9. If the document is in Kiswahili, add the document to the Kiswahili corpus and use it to augment the Kiswahili unigram and bigram count models.
10. If the document is in the other language(s) model, update the other language(s) word list.
11. If the document is ambiguous, discard it.

End For

Until termination condition

Application Architecture

The application consists of five main components. These are:

i. Graphical user interface
ii. HTTP module
iii. PHP scripts
iv. Web server
v. Database

The role of the various components is described below.

The database

The database is used to store the data needed and used in the application. This includes the urls of the documents to be downloaded, the Kiswahili unigrams (word list) and bigrams (pairs of words) and the other languages unigrams (word lists).

Web server

The web server is used to run the PHP scripts. Local documents are also accessed through the Web server.

HTTP module

This consists of an interface to enable us to issue get requests much in the same way as we do using a Web browser. It provides an interface to which we give a url, it downloads the document referred to in the url and stores it locally or returns a failure message if the get command is not successful. The java programming language provides such an interface using the http url connection and url classes in the java.net package.

To run the PHP scripts we will also have to make get requests to the Web server.

Graphical user interface

The graphical user interface provides a front from which commands can be executed. The commands that can be run from the interface include: entering and saving the initial models urls, creating the initial Kiswahili model, creating other languages initial model, creating the corpus.

i) Entering and saving the initial models urls

As discussed in the section on steps for creating a Web corpus (the sub-section on creating the initial models), we need to provide documents to be used to create initial models. These can be provided in the form of urls where the documents are contained. If they are available in the local machine, then they can be served by the local Web server. The application provides an interface where these urls can be entered and saved. In entering a url one has to specify the initial model it will be used to create, i.e. either Kiswahili or other languages.

ii) Creating the Kiswahili initial model

Before this command is run, the urls for creating initial models ought to have been saved in the urls table in the database. Each of the urls designated for creating the initial Kiswahili model is obtained, and a get request is issued using the http interface to download the file which is saved locally. A script to process a downloaded file as described in the section on steps for creating a Web-corpus (the sub-section on downloading and processing the files) is then called using the http interface. If this script completes successfully the processed file is saved. Another script to create the initial Kiswahili model is called using the http interface. This script picks the unique words in the processed file and adds them to the unigrams table together with their counts. If a word already exists in the unigrams table, its count in the processed file is used to update its count in the unigrams table by adding the value of the count in the processed file. The script also picks the unique bigrams (pairs of words) in the processed file and adds them to the bigrams table together with their counts. If a bigram already exists in the unigrams table, its count in the processed file is used to update its count in the bigrams table by adding the value of the count in the processed file. The processed file is then added to the corpus.

iii Creating the initial other languages model

Before this command is run, the urls for creating initial models ought to have been saved in the urls table in the database. Each of the urls designated for creating the initial other languages model is obtained, and a get request is issued to download the file, which is saved locally. A script to process a downloaded file as described in the section on steps for creating a Web corpus (su-section on downloading and processing the files)

is then called using the http interface. If this script completes successfully the processed file is saved. Another script to create the initial other languages model is called using the http interface. This script picks the unique words in the processed file and adds them to the other-unigrams table together with their counts. If a word already exists in the other-unigrams table, its count in the processed file is used to update its count in the other-unigrams table by adding the value of the count in the processed file. Bigrams are not needed for the other languages model.

iv) Creating a corpus from the Web

Before this command is run, the initial models ought to have been created. The bigrams table is accessed and the bigram with the highest frequency is used to create a search query for a search engine. A get request to the search engine is made using the http interface. The get call to the search engine returns a search-results page that is saved as a file. A call is made to a script that processes this file and extracts the links which are usually between <A> </A> tags. The links are stored in the urls table.

For each of the links, a get call is made using the http interface to download the file and store it locally.

If the get is successful, a script to process a downloaded file as described in the section on steps for creating a Web corpus (sub-section on downloading and processing the files) is then called using the http interface. If this script completes successfully the processed file is saved.

Another script to create the Kiswahili model and corpus is called using the http interface. This script picks the unique words in the processed file and uses them together with the unigrams and other-unigrams table to determine if the document is in Kiswahili, other languages or ambiguous as described in the section on steps for creating a Web corpus (sub-section on creating the text corpus and the models).

If the document is ambiguous, the script discards it.

If it is in Kiswahili it adds the unique words to the unigrams table together with their counts. If a word already exists in the unigrams table, its count in the processed file is used to update its count in the unigrams table by adding the value of the count in the processed file. The script also picks the unique bigrams (pairs of words) in the processed file and adds them to the bigrams table together with their counts. If a bigram already exists in the bigrams table, its count in the processed file is used to update its count in the bigrams table by adding the value of the count in the processed file. The processed file is then added to the corpus.

If the document is in other languages, it adds the unique words to the other-unigrams table together with their counts. If a word already exists in the other-unigrams table, its count in the processed file is used to update its count in the other-unigrams table by adding the value of the count in the processed file.

After all the links have been used, we make another search request using the next most frequent bigram and repeat the process.

PHP Scripts

A lot of text processing is necessary in the corpus-construction process. Regular expressions are needed to carry out the text processing. PHP is a good candidate for

writing the scripts for this process because of its ease-of-use and the powerfulness of its regular expressions. It also provides an easy to use interface for most database management systems. The following are the scripts needed in our application:

i) Script to process search results

This script takes a file containing the results returned by a search engine and extracts the links. The links are in between <A></A> tags. Only .html, .htm and .txt links are picked. This can easily be done using regular expressions. The links are then stored in the urls table.

ii) Script to process a downloaded file

A file downloaded from the Internet is formatted for display on a browser. It thus contains a lot of formatting words and tags we would not want to include in a corpus. The purpose of this script is to carry out this formatting to remove unwanted words and tags while leaving the desirable sentences in the file. This script implements the steps described in the section on steps for creating a Web corpus the (sub-section on downloading and processing the files). This is done using regular expressions.

iii) Script to create initial Kiswahili model

The working of this script is described in the section on the graphical user interface on (sub-section creating the Kiswahili initial model).

iv) Script to create initial other languages model

The working of this script is described in the section on graphical user interface (the sub-section on creating the initial other languages models.

v) Script to create models and corpus

The working of this script is described in the section of graphical user interface (the sub-section on creating corpus from the Web).

Descriptive Statistics

Table 16.1: Urls (search) statistics

1	Execution time (for application)	21 hours
2	Number of searches	21 (21 bigrams used to search)
3	Number of urls collected	2,253
4	Urls used for Kiswahili corpus	1,202 (53%)
5	Urls used for other languages corpus	1(0%)
6	Discarded (ambiguous)	800 (36%)
7	Not used (application stopped)	246(11%)
8	Urls for initial models	4 (0%)

Table 16.2: Ten most frequently used words

1	Number of words in the corpus (tokens)	4,843,581
2	Number of unique words in the corpus (types)	150,274
3	Number of words appearing just once	81,152 (54 %)
4	Number of words appearing two times and below	98,666 (66%)
5	Number of words appearing three times and below	106,641 (70%)
6	Words appearing 125 times and above	3,046 (0.02 %)

Table 16. 3: Unigram Statistics

1	na	308,655	6.37%
2	ya	219,197	4.52%
3	kwa	147,973	3.05%
4	wa	132,981	2.74%
5	ni	78,509	1.62%
6	katika	66,589	1.37%
7	Mungu	41,229	0.85%
8	kuwa	41,033	0.84%
9	za	38,572	0.79%
10	yake	34,254	0.70%

Table 16. 4: Bigram Statistics

1	Number of bigrams (total)	214,8275
2	Number of unique bigrams	58,3871
3	Number of bigrams occurring just once	419,171 (72%)
4	Number of bigrams occurring twice or less	479,760 (82%)
5	Number of bigrams occurring three times or less	504,67 (86%)

Table 16.5: Ten most frequent bigrams

1	mwenyezi Mungu	mighty God	8156	0.379653%
2	juu ya	On	4393	0.20449%
3	baada ya	after	3786	0.176234%
4	pamoja na	with	3448	0.160501%
5	kwa sababu	because	3445	0.160361%
6	kwa ajili	because	3147	0.14649%
7	ndani ya	in (inside)	2569	0.119584%
8	ajili ya	due to	2511	0.116884%
9	kutokana na	as a result of	2210	0.102873%
10	na kwa	and "… because"	2009	0.093517%

Discussion

Table 1 shows that with more searches, a bigger corpus can be constructed. It also shows that there is a lot of intersection between Kiswahili and other languages. These discarded urls can be investigated to find out what the languages are and also the kind of documents they are.

Table 2 shows that most words appear only once (54%). Seventy percent of the words appear three or less times. This translates into sparseness in an N-gram probability model. Thus there is need for smoothing techniques to be used. Also the corpus needs to be as large as possible.

Table 2 and ***Table 3*** show some information about the most frequent words in the Kiswahili language. A large percentage of these can be assumed to be in any Kiswahili document. Thus they can be used to automatically determine if a document is in Kiswahili or not.

Table 4 shows that most bigrams occur just once (72%). The bigrams occurring three or less times account for 86% of the bigrams.

Table 5 shows "Mwenyezi Mungu" as the most frequent bigram. This is a pointer to the fact that a lot of the Kiswahili documents on the web have religious content. The other most frequent bigrams are pairs of words used as prepositions.

Limitations

More statistics need to be done on the corpus. For instance, we could compare this corpus with a manually generated corpus.

More processing of a document before it is included in the corpus in needed. One can go through processed documents to pick out what mistakes the current processing is making. One can then add a feature to the processing process to remove these mistakes. This should aim to ensure that only well formed sentences and bigrams are included in the text.

Further Work

- Use of most frequent words (Table 3) to speed up recognition of documents.
- Compare with other approaches.
- Investigate quality and type of documents in the corpus.
- Make an attempt to classify the type of documents, e.g. newspaper, religious, sports, academic, corporate etc. using the words "unique" to label the different kinds of documents.

Conclusion

Using the process discussed, it was possible to create a corpus of about 5 million words within about 24 hours. The limitation of quality that might result from a corpus created by this method is compensated for by the size. Thus, for the purpose of quickly creating an individualised corpus this application and the methods discussed in this paper can be used. The methods discussed here can be applied to any language. The only parameters that will need to be changed are the seed documents.

References

Callan J., M. Connell, Du (1999). Automatic Discovery of language models for text databases. Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data, (pp. 479–490). Philadelphia, USA.

Gilles-Maurice De Schryver (2002). "Web for/as Corpus: A Perspective for The African Languages". Nordic Journal of African Studies 11(2),(pp 266-282).

Jones, R. and R. Ghani (2000). Automatically Building a Corpus for a Minority Language From the Web. Proceedings of the Student Workshop at the 38th Annual Meeting of the Association for Computational Linguistics, (pp. 29-36).

Jones, R. and R. Ghan. (2000). "Learning a Monolingual Language Model from a Multilingual Text Database". Proceedings of the Ninth International Conference on Information and Knowledge Management, (pp. 187 – 193).

Jurafsky, D. and J. Martin (2000). *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition*. Pearson Education, Delhi, India.

Notess G. R. (2004). Review of Google. [online]. Available: *http://www.searchengineshowdown.com/features/google/review.html*

Sekiguchi Y., and K. Yamamoto (2004). "Improving Quality of the Web Corpus". Proceedings of The First International Joint Conference on Natural Language Processing, (pp.201-206).