TEXT-BASED IDENTIFICATION OF SELECTED NIGERIAN LANGUAGES USING CHARACTER N-GRAMS

BY

ODODO PEACE ONOYIZA
(130407025)

ADEYEMI MOFETOLUWA OLUWASEUN
(140407515)

A PROJECT SUBMITTED TO THE DEPARTMENT OF SYSTEMS ENGINEERING, UNIVERSITY OF LAGOS, AKOKA, IN PARTIAL FULFILMENT OF THE REQUIREMENT FOR THE AWARD OF BACHELORS OF SCIENCE (B.Sc. Hons) DEGREE IN SYSTEMS ENGINEERING.

SUPERVISOR: DR K.O. OROLU

OCTOBER 2018

DECLARATION

We hereby declare that we carried out the work reported in this report in the Department of Systems Engineering, University of Lagos, Akoka, under the supervision of Dr Kehinde Orolu. I solemnly declare that to the best of my knowledge, no part of this report has been submitted here or elsewhere in a previous application for award of a degree. All sources of knowledge used are duly acknowledged.

ODODO, PEACE

130407025

ADEYEMI, MOFETOLUWA

140407515

CERTIFICATION

This is to certify that the project titled “Text-Based Identification of Selected Nigerian Languages Using Character Ngrams” carried out by Ododo Peace and Adeyemi Mofetoluwa has been read and approved for meeting part of the requirements and regulations governing the award of Bachelor of Science degree in Systems Engineering of University, Akoka, Nigeria.

DR K.O. OROLU (PROJECT SUPERVISOR)	DATE

DR LADI OGUNWOLU (HEAD OF DEPARTMENT)	DATE

DEDICATION

This project is dedicated to our family for their guidance, support and love and also to our friends.

ACKNOWLEDGEMENT

We are grateful to the almighty God for the completion of this project, for his sustainance, protection and inspiration.

Our profound gratitude goes to our supervisor Dr. Kehinde Orolu for his guidance and excellent supervision during the course of this project.

Our appreciation and utmost gratitude goes to Dr. Tunde Adegbola who created time despite his busy schedules, to direct us on the necessary steps to taken in the course of this project.

A big thank you to Mr Isu Mohammed, who also helped us with important information, needed for our project. We are also grateful and thankful to the Department of Systems Engineering, Faculty of Engineering, University of Lagos for granting us the opportunity to work on this project.

A special thanks to our parents, Mr and Mrs Ododo, Mr and Mrs Adeyemi and our families for their unending love, emotional and financial support towards the success of this project. We also appreciate all our friends and colleagues for their love and care. May God bless you all. Amen.

ABSTRACT

Language Identification is the problem of determining which natural language a given content is in. This project focuses on creating a system that can identify the Nigerian Language in which a given document is written.

The Statistical approach is employed in creating the system, which involves modeling the languages and using a classification technique to classify the test data to the appropriate language. This approach tends to be less stressful and cumbersome when compared to the Linguistics approach.

N-grams were used as features in the language models created as it serves as an efficient way of breaking down a language to its bits. Based on certain factors such as availability of the training data and encoding of the document, Information Theoretic measure known as the Mutual Cross Entropy was used as a distance measure in classifying the test data to the training data.

In this project, three major Nigerian languages which are Yoruba, Igbo and Hausa were classified. From the level of accuracy attained in classification these languages, the system would be able to work with the other languages.

The Statistical approach to Language Identification using N-grams as modeling features and the Mutual Cross Entropy as classification algorithm yielded a very accurate system, even with a small language pool of Yoruba, Igbo and Hausa Languages

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 Background

The importance of Language Identification in Natural Language Processing (NLP) cannot be over-emphasized. Natural Language Processing is an aspect of Artificial Intelligence that involves how to program computers to efficiently process Natural Language data, providing a platform for exchanging useful information with humans. In achieving globalization and satisfying the need to communicate internationally, challenges involving the NLP process have evolved from being extremely difficult to become more tractable due to the rapid growth in computing power and the advancement made in language processing algorithms (Ahmed, Cha and Tappert, 2004). Dealing with these challenges (involving speech recognition, speech translation and text-to-speech synthesis) have led to the successful performance of tasks such as sentiments analysis, question answering, part-of-speech tagging and information retrieval generally. Language identification is seen as a prerequisite to performing such tasks due to the assumption that the language is already identified. For instance, in a multilingual environment such as Nigeria, a telephone help line may need to identify the spoken language before routing the call to the appropriate operator. A text-to-speech system may need to identify the language of a short piece of text to determine the pronunciation rules, prosodic models and phrasing strategies (Botha, Zimu and Barnard, 2012).

In creating systems for identifying the language of a given text, models of the languages involved are developed. Language modelling is the task of assigning probabilities to character sequences or sentences in a language (Kuhlmann, 2018). Sea veral approaches towards language modelling have been made depending on the amount of linguistic process needed. A set of these approaches involve the use of Linguistic models while the other uses Statistical or Probabilistic Models. The advantage of the latter over the former is: Linguistic modelling, although most of the time perfectly accurate, involves the use of language specific rules and could be computationally expensive. On the other hand, Statistical models make use of the statistics of features such as character or word sequences of the language and employing algorithms to perform the text-based Language Identification. Language Identification was one of the first NLP problems for which a statistical approach was used (Singh, 2006). The N-gram modelling approach is the most popular modelling and is what we are interested in this project. Several methods of feature extraction have been employed in classifying languages such as unique letter combinations, short word method, N-gram method and ASCII Vector of Character Sequences (Ahmed, Cha and Tappert, 2004), but the use of N-grams in Language modelling has surpassed the others, followed by the word-based method of language modelling. An N-gram is a contiguous sequence of *n* words or characters while an N-gram model is a language model defined on n-grams (Kuhlmann, 2018). To achieve accuracy and for comparison, two or more N-grams are usually used in modelling the language.

Even though the world is fast becoming one community as global communication keeps growing, there are few resources for the larger percentage of the world’s languages. Of the estimated 6500 languages that exist in the world, there are probably 5000 of them spoken by people with access to cell phones but very little is known about most of them (Munro, 2016). Of the about 400 indigenous Nigerian languages, only three – Yoruba, Igbo and Hausa; are listed as major Nigerian languages and nine – Hausa, Igbo, Yoruba, Efik/Ibibio, Fulani/Fulfulde, Kanuri, Edo, Tiv and Ijo; as regional major languages leaving the remaining three hundred and more as small minority languages (Agheyisi, 1984). There is a high possibility that the nine major Nigerian languages and a good percentage of the minor languages are part of the 5000 which have access to phones, therefore partaking in the advantage of using their languages over the phone and internet and even audio-recording so at least these languages can have records of existence. But these languages haven't had the advantage of being a part of Natural Language Processing due to lack of enough language data, a requirement for NLP (Munro, 2016). The peak we may have seen so far regarding language processing and Nigerian Languages is the Google translator as it also tends to identify the language of the text but with its limitation to just Yoruba, Igbo and Hausa. It also is not very accurate.

In this project, we contribute to bridging the gap between Nigerian Languages and Natural Language Processing by taking up the essential step of Language Identification. Using Statistical Language Modeling, N-grams were employed as features and determine the best form of N-gram classification that is suitable for the conditions met by our indigenous languages. A system that can accurately identify texts in our Yoruba, Igbo and Hausa is first created, then some of the remaining regional languages are considered for future work .

1.2 Statement of Problem

Today, Information is vital to human development and societal advancement and the best way to access it is in one's language: unbiased and understandable. Natural Language Processing plays a crucial role in information generation, dissemination, storage and retrieval as we have seen with the major languages of the world such as English, limiting a large percentage of the world's minor languages to the little information they can access. Therefore, the development and standardization of indigenous languages has become very important as there is a need for Nigerian indigenous languages to partake in Natural language processing (Obinyan, 2010).

Identification of our Nigerian languages would facilitate more of these tasks.

1.3 Aim and Objectives

The aim of this project is to create a Language Identification System based on Character N-grams that is efficient in Identifying and distinguishing various texts in Nigerian Languages.

To achieve this aim, the following objectives are set:

1. Obtain a dataset of Yoruba, Igbo and Hausa languages for the Language Modelling
2. Develop language models for the languages using N-grams as modelling features
3. Compare and classify the languages using an efficient N-gram based classifier
4. Build a language classifier application that identifies Nigerian languages.

1.4 Project Scope

This work is limited to standard Yoruba, standard Igbo and standard Hausa. In achieving the aim of distinguishing and identifying between texts in various Nigerian Languages using character N-gram probability as a feature for modelling, an efficient classification algorithm must be determined which would be able to achieve a reasonable amount of efficiency and classification speed.

1.5 Significance of the study

Creating a computational system that Identifies and distinguishes between various Nigerian Languages using N-grams as features is a step to facilitating and encouraging other Natural Language Processing Tasks that would involve Nigerian indigenous languages. It also encourages the idea of passing online information (the most efficient way of information exchange) in Nigerian Languages since the languages would be identifiable, therefore more written texts would be found in these languages.

1.6 Organization of Thesis

This thesis is organized as follows:

- Chapter 2 involves reviewing literature on past works related to the process of Language Identification.
- In Chapter 3, the approach taken towards creating the Language identification system and the building of the web application are discussed.
- The results gotten from implementing the system and those from the web application are discussed in Chapter 4.
- Chapter 5 conludes the report by discussing reccommendations, future works and giving a general summary of the work.

1.7 Operational Definition of Terms

Corpus: A collection of written texts.

Identification: The task of automatically detecting the language(s) present in a document based on the content of the document.

N-grams: An N-gram is a contiguous sequence of *n* words or characters

Natural Language Processing: is the ability of a computer program to understand human language as it is spoken.

Nigerian Indigenous Languages: three hundred and ninety-nine (399) Nigerian languages divided into the Major (9) and Minor (390) languages.

CHAPTER 2

LITERATURE REVIEW

Language Identification, is the task of determining the language a given document is written in and it has become an increasingly important topic in the Artificial Intelligence community as more documents keep making their way online (Baldwin and Lui, 2010). For instance, it is less stressful to identify the language in which most-hard copy documents are written, as a good Librarian can easily do that. But since most documents online do not pass before a human's eyes, computational systems are needed to perform the Language Identification Task (Sibun and Reynar, 2004). The identification of languages in a given text or document (online or not) is an important step to achieving language processing tasks.

In this chapter, previous approaches to solving the problem of Language Identification from written texts, otherwise called Text-based Language Identification, are reviewed. This involves the Language model used and features used for modelling. The use of N-grams as features for language modelling was especially considered, comparing it with other possible features. Classification methods used by previous authors are examined to determine which would perform best in attaining the desired accuracy for our computational system. Lastly, the structure of the Nigerian Languages and their level of participation in Technology and Natural Language Processing were examined, comparing it to that of other languages used in previous researches.

2.1 Language Modeling

Various approaches have been considered under the general topic of text-based Language Identification. These approaches differ according to the level of linguistic processing employed (Botha and Barnard, 2012). With respect to this factor, approaches can be classified into two main branches: Linguistic Models and Statistical models.

Linguistic Models deal with approaches that attempt to do a complete parse of text in order to determine the language and also the syntactic structure of the textual fragment (Botha and Bernard, 2012). Although these models are realistic, they are complex and require language specific rules. An example is the Weighted Finite-state Transducers which used a Lexical toolkit that allowed declarative description of lexicons, morphological rules, phonological rules etc. used by Sproat (1996) (Ahmed, Cha and Tappert, 2004). Johnson (1993) used a list of stop words from different languages to identify the language of a given document. Duiere-Lins and Goncalves (2004) considered the use of syntactically-derived closed grammatical class models, matching syntactic structure rather than words or character sequences. (Baldwin and Lui, 2010)

The second branch of approaches which involves Statistical Models makes use of statistics of features extracted from the text. Methods of feature extraction include unique letter combination, short word method, N-gram method and ASCII vectors of character sequences. Statistical models are generic and they utilize different features from training samples to categorize text through machine learning (Ahmed, Cha and Tappert, 2004). The use of Statistical models has more advantage over that of the Linguistic Models and has been employed in most Language Identification researches over the years.

Cavnar and Trenkle (1994) used the statistical modeling method in generating the Frequency profiles of the languages (Language Model) using N-grams.

Based on the statistical modelling of the languages, the algorithm to obtaining the N-grams and calculating their frequencies may differ. For instance, in the work done by Ahmed, Cha and Tappert (2004), the N-grams were obtained by reading each line of text and segmenting the string into different size N-grams, moving forward character by character and replacing the white space by an underscore. Also, pre-processing such as the removal of punctuation marks, numbers and special characters was done before collecting the N-grams.

2.2 Features Used for Language Modelling and N-grams

In identifying the language of a text, there are significant features gotten from the text which make up the Computational System used. These features could be characters, words, N-grams of either or ASCII vector of character sequences. Ziegler (1991) made use of the presence of particular characters. Souter et al. (1994) made use of the presence of particular words; Ahmed, Cha and Tappert (2004) and Duvenhage and Ntini and Ramonyai (2017) made use of the presence of particular character n-grams; Nakayama and Spitz (1993) and Sibun and Spitz (1994) made use of particularly-shaped words from images (Sibun and Reynar, 2004). N-grams are simple to compute for any given text and allow for a straightforward trade-off between accuracy and complexity.

2.2.1 N-Grams

According to Cavnar and Trenkle (1994), an N-gram is an N-character slice of a longer string. The N-grams of a string are gathered by extracting adjacent groups of n letters. In creating a Language Identification Computational model, N-grams of different lengths i.e. Unigrams, Bigrams, Trigrams and so on, are used simultaneously and compared in order to achieve accuracy as stated by Botha and Bernard (2012, p.1):

"The N-grams are also extremely simple to compute for any given text, which allows a straightforward trade-off between accuracy and complexity

The advantage of N-gram based matching lies in its ability to decompose strings into small parts. Therefore, errors that are present tend to affect only a limited number of those parts leaving the remaining parts intact. When N-grams common to two strings are counted, there is a measure of their similarity gotten which is resistant to a wide variety of textual errors. N-grams based matching has also been used to deal with noisy ASCII input in Natural Language Processing applications, Text retrieval and interpreting postal addresses.

In Cavnar and Trenkle's N-gram based Text Categorization, the advantage of the use of N-grams over the use of Lexicons was also stated. A Lexicon comprises all of the words that make up the Language of a people, individual, or knowledge area. It also involves morphemes and their combination present in the language. In classifying the languages, Lexicons for each language are kept, and then words in the sample text are checked to see which Lexicon they fall in. They lexicon that contains the most words from the sample document shows the language of that text. The problem with this technique is that it is quite difficult, especially for less popular languages. The Language may also be one that uses different forms of a word to indicate case, tense or other attributes. This would pose as a problem because larger lexicons would be needed to get the necessary word included or, a language-specific morphological processing must be developed to reduce different forms to their stems. Using N-grams, most of these problems will not be encountered. The occurrence of some N-grams gives strong evidence that a text belongs to a particular language and these N-grams are identified by creating the N-gram frequency profile for that text using the frequency profile technique. This is done without building any lexicon or morphological processing rules. (Cavnar and Trenkle, 1994)

Language models created from N-grams provide a simple and reliable way to categorize documents in a wide range of classification tasks.

2.3 N-gram Based Classification Algorithms/Techniques

There are several classifiers that have been employed in the task of Text-based Language Identification.

These include: Rank-Order statistics (also known as the Out-Of-Place Method), Naive Bayesian (Dunning,1994), Relative Entropy (Sibun and Reynar, 2004) Dot-product of word frequency vectors (Darnashek, 1995), Information theoretic measures of document similarity (Martins and Silva, 2005; Singh, 2006).

Pattern Recognition algorithms such as Support Vector Machines (Teytaud and Jalam, 2001), Monte Carlo sampling, Markov Models (Xia et al, 2009), decision trees, neural networks and multiple linear regression have also been used.

N-gram-based Classifiers are focused on, in this paper.

2.3.1 Rank-Order Statistics / Out-Of-Place Method

In creating an N-gram based Text categorization system, Cavnar and Trenkle (1994) made use of the Rank-Order Statistical method in classifying language documents by generating the Frequency profiles for documents in different languages. The system was based on comparing the N-gram frequency profiles of the training set data that represented the various categories and that of a particular document that is to be classified. Then the distance measure between the document's profile and each of the categories' profile is computed. The comparison that gives the smallest distance gives the category to which the document belongs.

The Rank-order method was used for both Language classification and Subject Classification of Text Categorization due to the following observations made:

- The top 300 or so N-grams are almost always highly correlated to the language i.e. if two documents are compared and there is a lot of similarity between their first 300 N-grams, then they most likely have the same language.

- From the rank 300 or so, the N-grams frequency profile begins to show N-grams that are more specific to the subject of the document i.e. if two documents talk mainly about Engineering, there would be a similarity between their N-grams from rank 300 below.

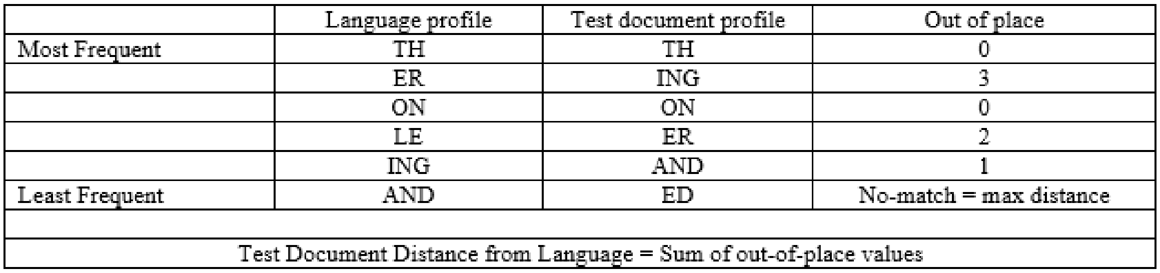
- It should be noted that there is nothing special about rank 300 since it was arrived at by inspection. One could do an elaborate statistics and an optimal cut-off rank can be chosen for a particular use.

These observations apply mostly to shorter documents. For longer documents, the shift from language-specific N-grams to Subject-specific N-grams would occur at a later rank.

After generating the frequency profile (Language model) for the test document, the Profile Distance Measure is found by calculating the simple rank-order statistic called the “Out-of-Place” measure between N-gram frequency profiles. The Out-of-Place measure determines how far out of place an N-gram in a profile is from its position in another profile and this is done for each N-gram in the document profile. That is, if an N-gram is at rank 3 and at a rank 5 in another profile, the out-of-place measure is 2. If an N-gram is not in the other document, it is given a maximum out of place value. An example is shown in the Table 2.1. The sum of all out-of-place measures for all N-grams gives the distance for the test document from that category.

Finally, the smallest distance for all the distance measures from all the category profiles to the document profiles gives the class the unknown language belongs to.

Table 2.1. Calculating the Out-of-Place Measure between Two Profiles (Ahmed, Cha and Tappert, 2004; Cavnar and Trenkle, 1994)



	Language profile	Test document profile	Out of place
Most Frequent	TH	TH	0
	ER	ING	3
	ON	ON	0
	LE	ER	2
	ING	AND	1
Least Frequent	AND	ED	No-match = max distance
Test Document Distance from Language = Sum of out-of-place values			

The distance measure indicates the closeness of the document profile to the category profile. The smaller the distance, the closer the document us to that category. Figure 2.1 gives the Data-flow for the N-gram based text categorization as descibed earlier.

The Rank-Order Statistics Classifier was termed as the Out-Of-Place Measure under Nearest-Neighbour/Nearest-Prototype Classification in the work of Baldwin and Lui (2010). A very similar method to it, termed as the Difference-In-Frequency method was also employed in the work of Botha and Bernard (2012).

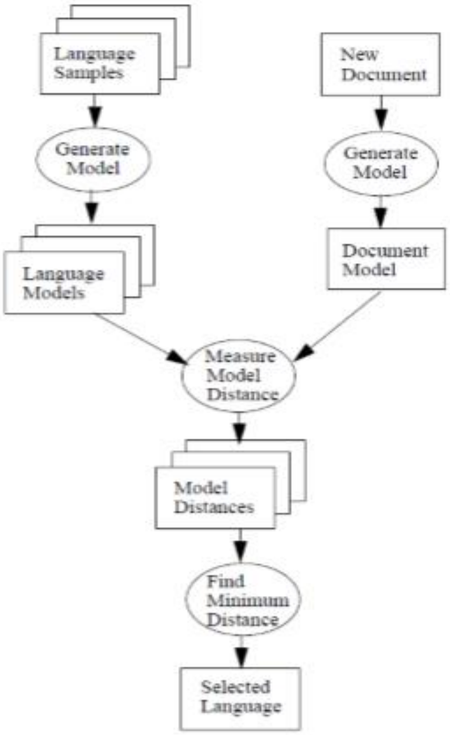


Figure 2.1 Dataflow for N-gram based Text Categorization (Canvar and Trenkle, 1994)

The Rank-Order Statistics method gave a very high accuracy and worked well on longer documents than on shorter ones.

2.3.2 The Cumulative Frequency Addition Method

For this classifier, a list of N-grams is generated from the input document, without considering their amounts or any sorting. This list can contain duplicates. Each N-gram in this list is searched in the language model of a considered language. The internal frequencies of the N-grams found are summed up.

Internal Frequency = Count of *i*th n-gram in the *jth* Language / Sum of the counts of all N-grams in the Language document.

The bigger the sum is, the smaller the distance between the language model and the document model. Finally, the language with the maximal sum is chosen as the language of the document (Panich, 2015).

The Cumulative Frequency Addition Method is a very simple and fast classifier.

2.3.3 The Naïve Bayesian (NB) Method

The Naive Bayesian Classifier is a popular text classification method, due to it being lightweight, robust and easy to update.

Botha and Barnard (2012) implemented the NB classifier in the Identification of eleven (11) South African Languages. For each language, a vector of N-gram probabilities was computed by

$$I_j = \frac{f_j}{|f_j|} \tag{2.1}$$

where f_j is a vector of n-gram frequencies is calculated from the language document of a class *j*.

The probability of the test string is calculated in the logarithm domain. The log likelihood simplifies calculations by adding logarithmic probabilities and this can be expressed as:

$$P(L|D) = \sum_{i=1}^{n-\alpha+1} In\ l_j(c_i) \tag{2.2}$$

After calculating the probabilities for all languages, the most likely language profile is selected as the language of the test string. For unseen n-grams a penalty value was given and tests were performed using various penalty values to decide on the best value for optimum classification accuracy.

2.3.4 Support Vector Machines (SVMs)

Support Vector Machines, a complex method of classification, is one of the most popular methods for text classification, largely because they automatically weigh large number of features, capturing feature interactions in the process. Vapnik (1995) stated that the basic principle behind Support Vector Machines is to maximize the margin between training instances and the calculated decision boundary based on structural risk minimization (Baldwin and Lui, 2010). The Support Vector Machine is a non-linear discriminant function that is able to classify in high dimensional space.

In using N-grams as features, the set of characters present in the n-gram combinations are considered as a space of X characters. Therefore, the total number of possible n-grams generated would be the number of characters X raised to the power of n. Thus the feature dimension of the SVM is equal to the number of n-gram combinations. The size of the feature space grows exponentially with n, which leads to long training times and extensive resource usage as n becomes large. Samples of the testing set are created using the same character space used to build the language model. After training the SVM language model the test samples can be classified according to the language.

In their paper, Botha and Barnard (2012) stated that the SVM used RBF Kernel, and overlap penalties were employed to allow non-separable data in the projected high-dimensional feature space. Classification is done in a one-against-one approach in which $\frac{k(k-1)}{2}$ classifiers are constructed (where k is the number of classes, thus 55 classifiers were constructed since they were dealing with 11 classes of South-African Languages) and each one trained from data of two different classes. Each binary classification was considered to be a vote for the winning class. All the votes are tallied, and the test sample is assigned to the class with the largest vote.

Baldwin and Lui (2010) made use of BSVM, an implementation of SVMs with multiclass classification support.

2.3.5 Information Theory Methods (Similarity/Distance Measures)

In an attempt to build a language and encoding identification tool, Singh (Singh, 2006) experimented with similarity measures used to measure the distance or similarity between language models. If the models are represented as distributions, then the similarity is basically a similarity of distributions. A lot of these similarity measures are based on Information theoretic approach.

According to Singh, two models P and Q over a variable X were considered. p and q are probabilities and r and s are ranks in P and Q respectively.

These measures are very simple ones and they all try to find the similarity between two models or distribution in a (more or less) information theoretic way, except the Out of rank measure proposed by Cavnar and Trenkle (Cavnar and Trenkle, 1994). Of all these measures, the performance of the Mutual Cross Entropy was the best, according to Singh (2006) in his effort to identify the language and encoding of South-Asian languages. The Mutual Cross Entropy can be seen as bidirectional or symmetric cross entropy. It is defined simply as the sum of the cross entropies of two distributions with each other. Cross entropy measures how similar one distribution is to the other, while Mutual cross entropy measures how similar the two distributions are to each other.

In modeling the languages, Singh mentioned that a few pages of text (2500-10000 words) were enough for training and Pruned Character n-grams is what he used. A similar model is made for the test data for which he mentioned that 5-10 words of test data would be enough. He evaluated his method using the various similarity measures listed above and tested with different test sizes. The similarity measures served as basis to the method he used. He also compared two cases, one in which the pruned Character n-gram model was used alone and the other in which it is augmented with a word n-gram model.

2.4. Comparing Classification Methods

According to Botha and Bernard (2012), there are factors which determine the accuracy of the Text-based Language identification. Crucial factors include: the size of the textual fragment to identify, the amount and variety of training data and the classification algorithm employed. Sibun and Reynar (2004) also identified the features to be used, the size of the input text, and the type of analysis used for identification as important issues in constructing an accurate, robust and fast system to identify Roman Alphabet Languages. Comparison between classification methods is done in relation to the factors mentioned above.

2.4.1 The Rank-Order Statistics Method, The Naive Bayesian Classifier and the N-gram Cumulative Frequency Addition Method

Ahmed, Cha and Tappert (2004) compared three Language classification methods: The N-gram Rank-Order Statistics Method, The Naive Bayesian Classifier and the N-gram Cumulative Frequency Addition Method based on the following.

- Speed of the Classifier
- Accuracy depending on length of document.
- Simplicity of Classifier

They discussed the results of an efficient language classifier using ad-hoc Cumulative Frequency Addition of N-grams. The technique was said to be simpler than the Conventional Naive Bayesian classification method and also classifies Short input strings more accurately than the Naive Bayesian Method, although they both perform similarly in speed. The Ad-hoc cumulative frequency addition technique was also described to be 5-10 times faster than the N-gram based rank-order statistical classifiers even though the rank-order statistical classifier has the advantage of being highly accurate and insensitive to typographical errors. But due to the required frequency counting and sorting of N-grams in the test document profile, the classification method has been recorded to be slower than other methods.

In their paper, they identified that the accuracy and speed of a Classifier as important in a high volume categorization environment. In trying to achieve this, their aim was to eliminate the frequency counting and sorting activities present in the Rank-Order Statistical method. Therefore, the Classifier used was able to achieve this using a new Cumulative Frequency Addition method.

In the analysis, a total of 291 files from 5 different languages were used for testing: Danish 52, English 66, French 53, Italian 60 and Spanish 60. For files of 150 bytes (characters) in length, the rank-order statistics and cumulative frequency addition methods attained an accuracy of 100%, while the Naïve Bayesian classifier attained 99.7%. The cumulative frequency addition and the Naïve Bayesian methods were of comparable speed and 3-10 times faster than the rank-order statistics method.

Panich (2015) also compared these three classification methods to find the effective one of them for the language identification of tweets where they used word-based models and N-grams based models which used the N-gram approach, the graph-based N-gram approached and the Improved graph-based. It was stated that for the Word-based method, the Cumulative Frequency Addition Classifier was 1-3% better than the Naive Bayesian Classifier and 1-6% better than the Rank-Order Statistics Classifier for the different Data sets used. For the N-gram approach, the Naive Bayesian Classifier worked in average better than other classifiers.

2.4.2 The Nearest-Neighbor/Nearest-Prototype Method (Cosine, Skew Divergence and Out-Of-Place) Methods, The Naïve Bayesian Method and The Support Vector Machines.

In the paper "Language Identification: The Long and Short of the Matter", Baldwin and Lui (2010) compared the use of the Support Vector Machine, Naive Bayesian Classifier and three Nearest-Neighbour Methods (Cosine, Skew Divergence and Out-of-place method (Rank-Order statistics)) to determine:

- The combination of tokenization strategy and classification method that achieves the best performance overall

- The Impact of the volume of training Data and Test document length on the accuracy of Language Identification.

- The interaction between character encoding and Language Identification

The tokenization used was Byte-based tokenization and Codepoint-based tokenization. N-grams used 1-gram, 2-gram and 3-gram.

Working with web documents under the assumption that the documents were monolingual, they were able to demonstrate that

- Language Identification is trivial over datasets with smaller numbers of languages and approximately even amounts of training data per language.

- Language identification is harder when dealing with dataset with larger numbers of languages with more skew in the amount of training data per language.

- Byte-based tokenisation without character encoding detection is superior to codepoint-based tokenisation with character encoding detection.

- Simple Cosine similarity-based Nearest-Neighbour Classification is equal to or better than models including Support Vector Machines and Naive Bayesian over language identification tasks.

2.4.3 The Support Vector Machine, Naïve Bayesian and Difference-In-Frequency Classifiers.

In the paper *"Factors that Affect the Accuracy of Text-based Language Identification"*, Botha and Barnard (2012) made mention of the fact that the Rank-Ordering classification is mostly outperformed in comparison tests. Therefore, they compared the Support Vector Machine, Naive Bayesian and a classifier similar to Rank-Order known as the Difference-In-Frequency classifier. Their comparison was done on the basis of examining the factors that determine the performance of text-based language identification with the focus on the 11 South-African official languages and using N-grams as features of classification. The factors considered were:

- Size of textual fragment/test document to identify.

- Amount and variety of training data

- Classification algorithm employed

In creating the N-grams, it was mentioned that increasing the size of n increases the accuracy of the classifier but beyond a certain level the accuracy decreases. As n increases, more computation and memory space is required. So the values of n were restricted to n=3 and n=6. The 6-gram was adopted because it yielded high accuracy when used in a Naive Bayesian classifier during a preliminary test. The training data sets used were the 200K, 400K, 800K, 1.6M and 2M character datasets. They also considered the 15 character window (2-3 words), 100 character window (A very long sentence) and 300 character window (A paragraph).

From their results, the following observations were made:

- All the classifiers showed improvements at larger training sets

- Overall, the 6-gram Naive Bayesian performed best with the exception of the 3-gram SVM performing slightly better than it with a 200k dataset and a 15 character window. The 6-gram did not do substantially better than the SVM for the other training sets and 15 character window.

- The 3-gram SVM performed better than the other 3-gram models with the exception of the 3-gram NB classifier performing better with the smallest data set using 100 and 300 character window.

- The Difference-In-Frequency classifier proved to be inferior to the other two.

An advantage the NB classifier had other than its excellent accuracy is that new language documents can simply be merged into an existing classifier by adding the n-gram statistics of these documents to the current language model.

Duvenhage, Ntini and Ramonyai (2017) in their *paper "Improved Text Language Identification for the South African Languages"* also used the Naïve Bayesian classifier, combined with a Lexicon based classifier to distinguish the specific South African Language, to predict the language family that a piece of text is written in. Their approach reduced language detection error by 31%.

2.4.4 Information Theory Approaches/Similarity Measures

Singh (Singh, 2006) in building a language and encoding identification tool specifically for South Asian languages compared various similarity measures based on N-gram characters. These measures were based on information theory except for the Out-of-Rank measure by Cavnar and Trenkle (1994). They include:

1. Log probability difference (LPD)
2. Absolute Log probability difference (ALPD)
3. Cross Entropy (CE)
4. RE measure based on relative entropy (RE)

5. JC measure (based on Jiang and Conrath's measure) (JC)

6. Cavnar and Trenkle's *out of rank measure* (CT)

7. MRE measure based on mutual (symmetric) relative entropy (MRE)

8. Mutual (symmetric) cross entropy. (MCE)

Their algorithms have been defined in Section 2.3.5 of this chapter.

From the results, it was seen that almost all the measures gave at least moderately good results but the best results were obtained from the Mutual Cross Entropy measure. The JC measure gave almost equally good results. The following observations were also made:

- More test data gives better results. But this did not always happen, which was also surprising. Therefore, other factors were involved.

- A factor was whether the size of the training data for the different models were the same.

- Another factor was noise in the data, which affected some measures more than some others.

- Using word n-grams to augment character n-grams improved the performance is some cases but not for JC, RE, MRE and MCE. It even reduced the performance for them in the case of smaller texts (100 and 200 characters).

Therefore, the JC, RE, MRE and MCE measures were observed to be the best with the Mutual Cross Entropy (MCE) topping the list.

2.5 The Advantage of Using Mutual Cross Entropy Method

From most of the previous work done on Language identification, the N-gram based classifiers that have out-performed others include the Naïve Bayesian Classifier, Support Vector Machines, Cumulative Frequency Addition Classifier. The Mutual Cross Entropy Method also showed high accuracy, although it was not compared with any of the above three.

Although these three groups of classifiers have shown high accuracy and relatively good speed in classification, the following are considered as advantages in using the Mutual Cross Entropy based on the work of Singh (2006).

a. It does not require a lot of words for both the training and testing data per language.

b. It shows high accuracy in identifying related languages.

c. The other methods mentioned above seem to do relatively better than each when considering certain factors. Using the Mutual Entropy method would reduce such bias.

2.6 Language Pool – Nigerian Languages

Most of the previous work done on Language Identification focused on the world's most popular languages such as English, Spanish, French, Italian, Chinese, Indian, Japanese, German etc. Distinguishing between these languages was not also very tedious as most of them were not similar. Obtaining the data set for training and testing also was not difficult as a lot of Corpora exist online with uniform encodings in most cases. Therefore, a large amount of data was easily gotten from newspaper websites (Cavnar and Trenkle, 1994), reader commentary (Zaiden and Callison-Burch, 2012), Periodicals, books, the Bible, government documents (Botha and Barnard, 2012), using Web-Crawlers and other means. Botha and Barnard also got some texts from an Academic Personnel.

However, the task of language identification gets harder as the language pool is narrowed to identical languages or languages with a lot of similarities. Most language identification approaches also require a large volume of documents to perform accurately. For most unpopular or native languages, getting enough corpora was not an easy task as seen in the work of Botha, Zimu and Barnard (2011).

Botha, Zimu and Barnard (2011) worked on the 11 Official South-African Languages: Afrikaans, English, isiNdebele, isiXhosa, isiZulu, Sepedi, Sesotho, Setswana, siSwati, tshiVenda and xiTsonga. These languages are grouped into families; Nguni (isiNdebele, isiXhosa, isiZulu and siSwati), Sotho (Sepedi, Sesotho, Setswana) and a pair not included in any of the families (tshiVenda and xiTsonga). Two data sets were used in their research:

- A limited domain set (consisting of the text of the South African national constitution in all eleven languages)
- An open-domain set which was created by an extension of web-crawling approach. The method employed an early language-identification system for automatic selection of Web-pages and suffered from two limitations: wrong identification of web pages

and web pages with mixed text. Therefore, they had to process those pages manually to create their open-domain set.

Special characters that exist in languages are also taken into consideration because they tend to affect the accuracy of identification especially for short texts. An example of this was in the misspelling of the Albanian letters “Ë” and “Ç” as “E” and “C” in the testing dataset of Hoxha and Baxhaku (2007), as Albanian keyboards are not very popular. This affected the accuracy of their system especially for shorter documents. Therefore, they had to create a custom training corpus that gave an accuracy of more than 99%. Also, the encoding of the texts in the training set is also considered as some encodings support special characters in languages. An instance was in the work of Botha and Barnard (2012), where all the text used were encoded in the UTF-8 format to support special characters found in Afrikaans (è, é, ê, ë, ï, ò, ó, ô, ö, ú, û, ü) and in Sesotho Sa Leboa and Setswanna.

Zaiden and Callison-Burch (2012) also worked on DID (Dialect Identification) for the Arabian Language. Their Automatic classifiers gave an 85% accuracy which is close to the 88% accuracy of human classification. The dataset used to train the Automatic classifiers was gotten from the dialect classifications done by humans.

Considering these factors in relation to Nigerian Indigenous languages, there are not a lot of Corpora available on most of our languages online, with the exception of the major languages Yoruba, Igbo and Hausa. Other Regional languages are found very scantily online. The encoding of the texts in the data set would also be considered due to the special characters that exist in Nigerian Languages. These are considerations looked at in our approach.

CHAPTER 3

METHODOLOGY

In this chapter, the method and approach used for the implementation of this project is developed. This include the use of statistical model, N-gram models for both training and testing data and a distance measure, Mutual cross entropy of both distributions and models (training and testing). To use this method, data was first collected.

3.1 Overview

Based on the literature review, a statistical model that uses N-grams for modeling the language identification system was used in this work, while mutual cross entropy distance measure is the classifier used for comparing and classifying the test data with the training data to determine which language the test data is categorized.

3.2 Data Collection

To create the model, corpus of several documents written in standard Yoruba, Igbo and Hausa languages was collected from sketch engine. In generating the corpora from sketch engine basic seed words were used such as ; man, woman, food, house written in Yoruba, Igbo and Hausa.

3.3 Tools and Techniques Used

- N-gram Model
- Mutual Cross Entropy Classification
- Python Program

3.4 Modeling the Languages

In modeling the languages, the following steps are taken.

1. Cleaning the language string/document i.e. removal of hyperlink tags and punctuation, conversion to lower case.
2. Splitting the string of language into a list of words.
3. Creation of trigrams using a list i.e. to ensure the words were correctly split into trigrams.
4. Creation of trigrams and frequency using a dictionary.
5. Sorting the dictionary in descending order.
6. Pruning the list of tuples containing trigrams and probability to a length of 1000.

3.4.1 Cleaning and splitting the Document

A program is written to clean the document by removing hyperlink tags and punctuation. Upper case is changed to lower case to ensure uniformity of character casing before splitting each word into N-grams.

3.4.2 Generating the Ngrams and Frequency

Each word was split into Ngrams. A dictionary was used to store the trigrams and their specific frequencies, as it has the ability to hold a key (Trigram) and the key's value (Frequency). An observation made when printing the trigrams and their respective frequency using the dictionary was that the order of trigrams created was not according to the order of words in the list i.e. the trigrams were scattered. In order to test that the program was correctly creating the trigrams, a sample in English was used. This was done for both checking the trigrams using a list and checking the trigrams with their frequency using a dictionary in order to compare both. The English sample used was:

3.4.3 N-gram Probability Calculation

$$\mathbf{P}_{i}(\mathrm{x}) = \frac{f_i(x)}{\boldsymbol{N}} \qquad (3.1)$$

Where;

$\mathbf{P}_{i}$(x) = Probability of the ith Ngram

$\boldsymbol{f}_i\,(x)$= Frequency of the *i*th Ngram

N= Total frequency of Ngrams

3.4.4 Statistics for each Language used for Training

The Table 3.1 gives the total number of words, n-grams and frequency of n-grams for each language.

Table 3.1. Table Showing Statistics for each Language used.

Language	Number of Words	Number of Specific N-grams	Total Number of N-grams

Hausa	35,027	3,206	91,870
Yoruba	32,874	4,880	63,340
Igbo	19,302	2,935	44,918

3.4.5 Sorting and Pruning the Dictionary

The program was able to successfully sort the trigrams in the dictionary, using the value of the probability (calculated from frequency) in descending order. It was sorted this way in order to prune by leaving only 1000 trigrams with the highest probabilities. In sorting the dictionary, it changes to a list of tuples i.e. (Trigram, Probability).

3.5 Classification Using Mutual Cross Entropy

Mutual cross entropy, which is a distance measure used for various problems, was used as a means of classification. Mutual cross entropy can be seen as bidirectional or symmetric cross entropy. It is defined simply as the sum of the cross entropies of two distributions with each other. Our motivation for using ‘mutual’ cross entropy was that many similarity measures like cross entropy and relative entropy measure how similar one distribution is to the other. This would not necessary mean the same thing as measuring how similar two distributions are to each other. Mutual information measures this bidirectional similarity, but it needs joint probabilities, which means that it can only be applied to measure similarity of terms within one distribution.

3.6 Comparison and Classification of the languages

a) Modeled test data using the statistical model.

b) Modeled training data was put in a list. This list contains the three lists for each of the 3 languages used for training data

c) Another list ‘result’ was created to contain distance measure result from comparison of test data with training data.

3.7 Training Algorithm

After getting the N-grams of the training data and the frequency occurrence sorted, the following steps are taken.

1. Train the system by preparing character based n-grams from the training data.
2. Sort them by rank using their frequency.
3. Prune by selecting only the top character N=grams, i.e the first 1000 character based n-grams (first ranked n-grams on sorted table). 1000 Ngrams are used in order to avoid combinatorial explosion and for a more accurate classification.
4. Calculate the character N-gram based score (sim_c) for the test data in relation with every model for which the system was trained according to equation 4

   Where

   $$sim_c = \sum_x (p(x) * log q(x) + q(x) * log p(x)) \quad (3.2)$$

   $p(x)$ = Frequency of occurrence of N-gram in the training data

$q(x)$ = Frequency of occurrence of N-gram in the test data

5. Based on the score (sim_c), the most likely language is then selected.

3.8 Language Identification Application

A web application was created to implement the language identification system. This serves as an interface between a user and the Language Identification system. The algorithm used for classification was implemented in the development of the web application. The Django Framework was used in creating the Application Performance Interface (API) for the application. The application identifies language of written text such as Yoruba, Igbo and Hausa languages.

CHAPTER 4

RESULTS AND DISCUSSIONS

In this chapter, the identification model result is discussed and analyzed.

4.1 Results from Modeling the Languages

In modeling the languages used for the training data and the test data, models which contained N-gram features of the language were obtained with their respective frequencies. The table below gives a sample of the N-grams in the Igbo training data and their respective frequencies.

Table 4.1 showing Sample of Igbo N-grams and their Frequencies

Trigrams	**Frequency**
Nwu	0.000267
yịn	0.000646
me	4.45E-05
wụọ	4.45E-05
nwe	0.007124
nwa	0.002939
esọ	8.91E-05
nwo	0.001692
esị	0.001603
ụm	4.45E-05
aug	2.23E-05

nọg	0.000156
ri	2.23E-05

4.2 Results from Using Different Lengths of Test Data

In testing the accuracy of the algorithm, different lengths of test data was used. The Table 4.2 below gives the distance measure for the different lengths of test data when compared with the training data. The language to which a test data belongs to gives the smallest distance measure when compared.

Table 4.2 Results obtained from comparing the test data with training data

		Distance Measure		
Test data	**Number of words**	**Yoruba**	**Igbo**	**Hausa**
Yoruba	1000	-9.500701084318257	-0.8712647116128442	-1.9553572537329478
Igbo	1000	-0.8886022661662601	-10.48453258827548	-2.848163771412795
Hausa	1000	-1.33809273173671	-3.0400068164764615	-10.704878858026913
Yoruba	600	-8.595261470660926	-0.42102127134545136	-0.8417957351490937
Igbo	600	-0.9327544371589473	-8.98765462877212	-2.397202770202894
Hausa	600	-1.4210713462903648	-3.1172888816087747	-9.709052204291888

Yoruba	300	-8.054151886583277	-0.6987989331246667	-1.6304199266437593
Igbo	300	-0.5915091233590072	-8.212140396983404	-2.599072565097221
Hausa	300	-1.4173161533868566	-2.3953021223765543	-8.94543623251282
Yoruba	100	-6.751829130242253	-0.37731862165361474	-0.5297671562531937
Igbo	100	-0.7550751604638133	-7.874231918703249	-2.0233752029629737
Hausa	100	-1.174060260297738	-2.4073005082888534	-7.248171017999588
Yoruba	30	-6.66069062344503	-0.22266556724052822	-0.8539716185038166
Igbo	30	-0.9295302470055893	-6.41912587691214	-2.2861103723414296
Hausa	30	-0.8927810335598655	-2.4840925922367716	-7.106783300794827
Yoruba	10	-5.574295693895815	0	-0.9560604111899353
Igbo	10	-0.9318583718920732	-5.371717090638054	-1.2874110611949938
Hausa	10	-1.5574468084995794	-2.3549199022184033	-6.1674540764452574
Yoruba	5	-3.693015449211475	0	-0.7817997589897497

Igbo	5	-0.4818304507918014	-5.776582972126477	-0.7980853290592751
Hausa	5	-1.7405736782475913	-2.0772908835250874	-5.885397718840419

It can be easily observed that the system classifies the test data accurately, even for a test data with a small length of 5 words. Also, it should be noted that proper classification was based on taking note of the language specific attributes such as the use of tone-marks where necessary.

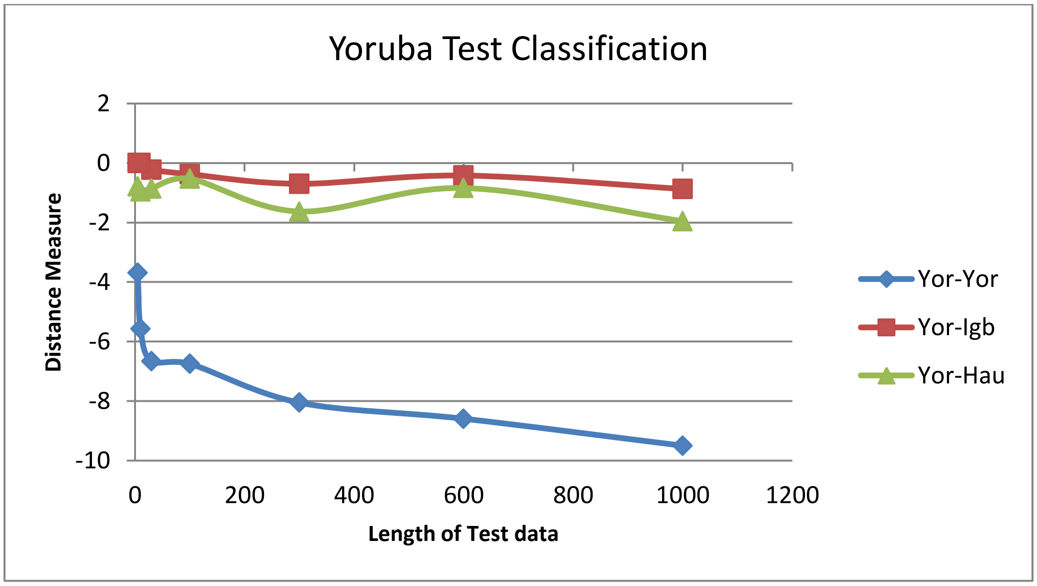


Figure 4.1 Graph showing the results from classifying Yoruba Test data with the 3 Training sets

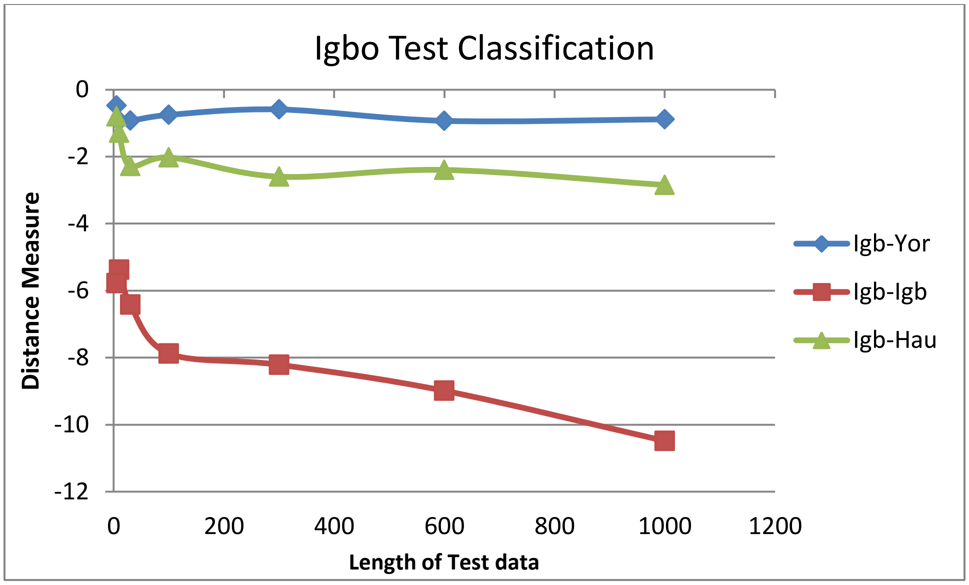


Figure 4.2 Graph showing the results from classifying Igbo Test data with the 3 Training sets

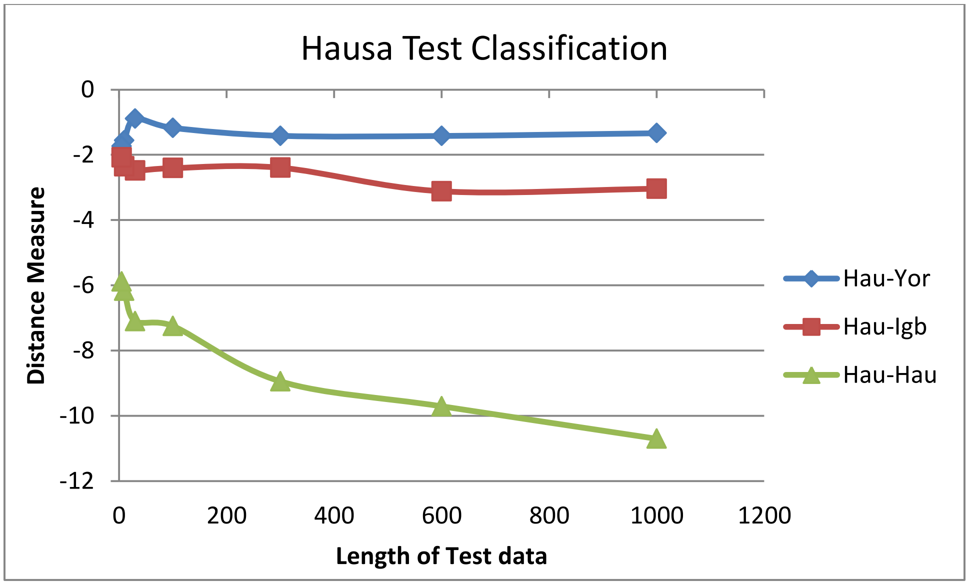


Figure 4.3 Graph showing the results from classifying Hausa Test data with the 3 Training sets

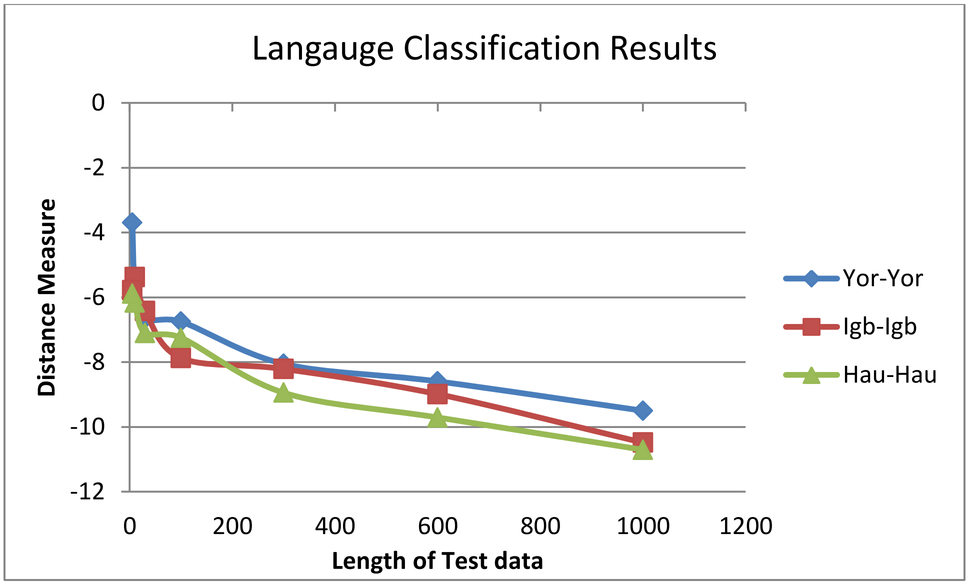


Figure 4.4 Graph showing the results of classification for each language set

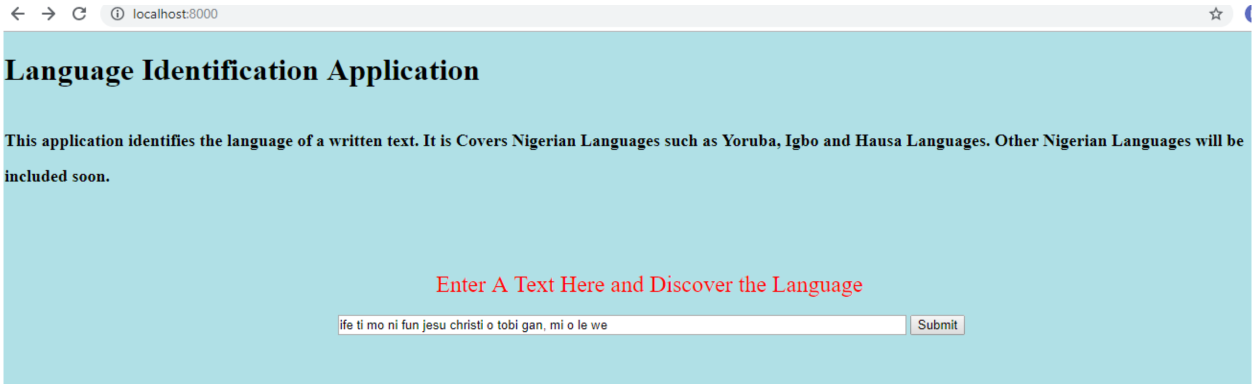
The graphs above show the result trends obtained by classifying the different length of test data ranging from 5 words to 1000 words. From Figure 4.1, the Yoruba test data is seen to have been classified accurately as the curve for the Yoruba language takes the least values, therefore the curve is the lowest on the graph. This signifies that in classifying this test data, the distance measure between the Yoruba test data and Yoruba training data are the smallest for all lengths of tests data when compared with the Igbo and Hausa languages. This is also the same case in classifying the Igbo and Hausa languages as seen in Figure 4.2 and 4.3. The system was able to accurately classify the test data with the languages required.

Although the system classified the test data accurately, the graph in Figure 4.4 shows that classification accuracy decreases as the length of test data decreases for the 3 languages. The

Test data with the length of 5 words has the highest distance measure (least expected result) when classified with its respective training data.

4.3 Performance of the Web Application

The web application created served as an interface between a user and the system.



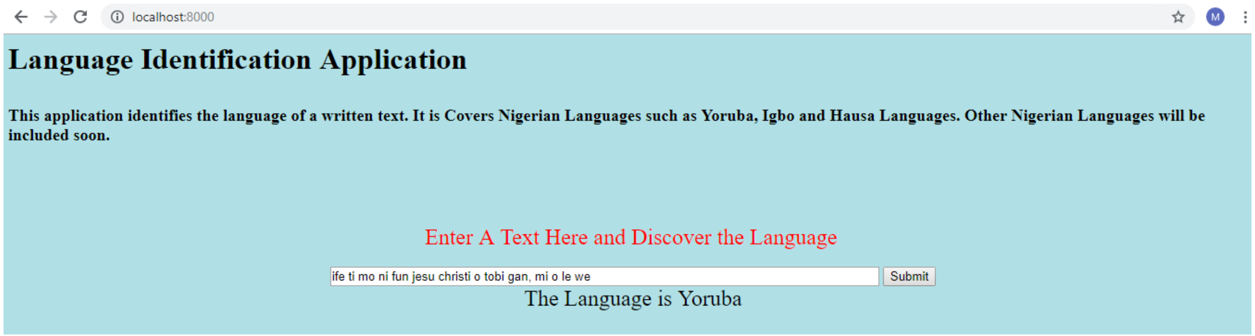
Language Identification Application

This application identifies the language of a written text. It is Covers Nigerian Languages such as Yoruba, Igbo and Hausa Languages. Other Nigerian Languages will be included soon.

Enter A Text Here and Discover the Language

ife ti mo ni fun jesu christi o tobi gan, mi o le we

Figure 4.5: Image showing the page where the user has put in a text in Yoruba in the user interface



Language Identification Application

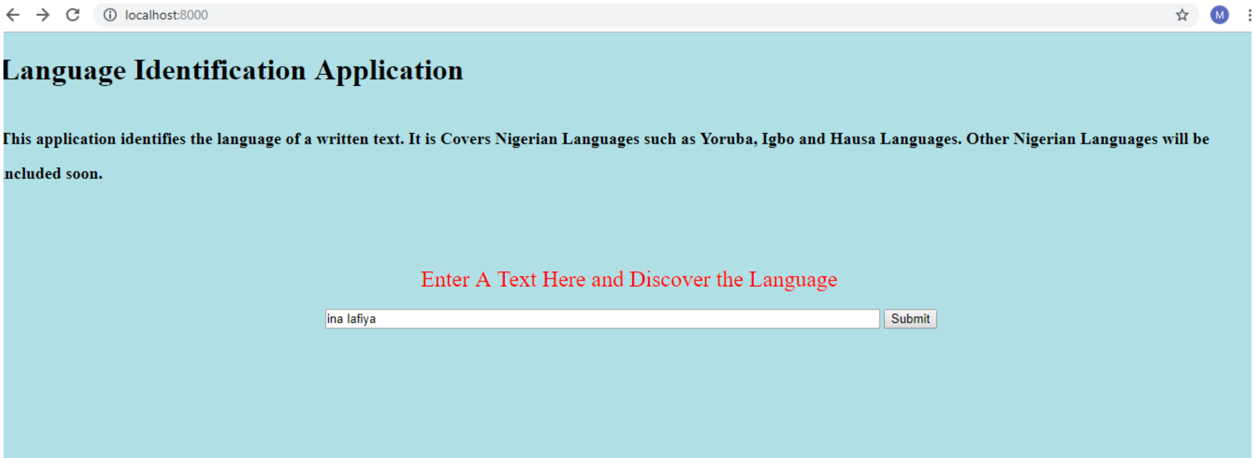
This application identifies the language of a written text. It is Covers Nigerian Languages such as Yoruba, Igbo and Hausa Languages. Other Nigerian Languages will be included soon.

Enter A Text Here and Discover the Language

ife ti mo ni fun jesu christi o tobi gan, mi o le we

The Language is Yoruba

Figure 4.6: Image showing Yoruba as the identified language of the user input from Figure 4.5



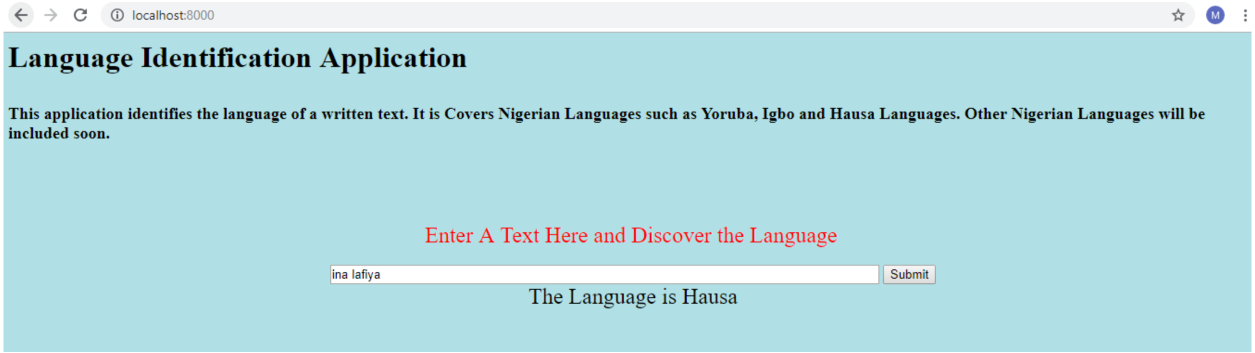
Language Identification Application

This application identifies the language of a written text. It is Covers Nigerian Languages such as Yoruba, Igbo and Hausa Languages. Other Nigerian Languages will be ncluded soon.

Enter A Text Here and Discover the Language

ina lafiya

Figure 4.7: Image showing the page where the user has put in a text in Hausa in the user interface



Language Identification Application

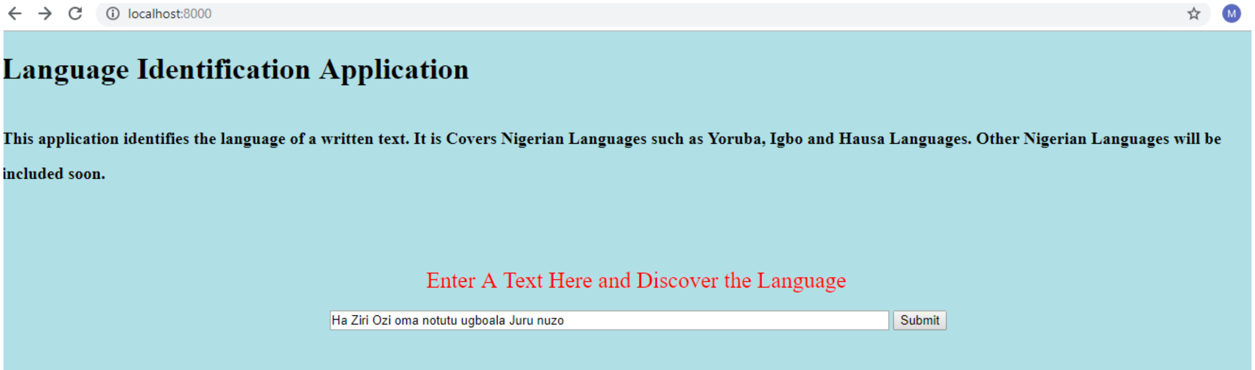
This application identifies the language of a written text. It is Covers Nigerian Languages such as Yoruba, Igbo and Hausa Languages. Other Nigerian Languages will be included soon.

Enter A Text Here and Discover the Language

ina lafiya

The Language is Hausa

Figure 4.8: Image showing Hausa as the identified language of the user input from Figure 4.7



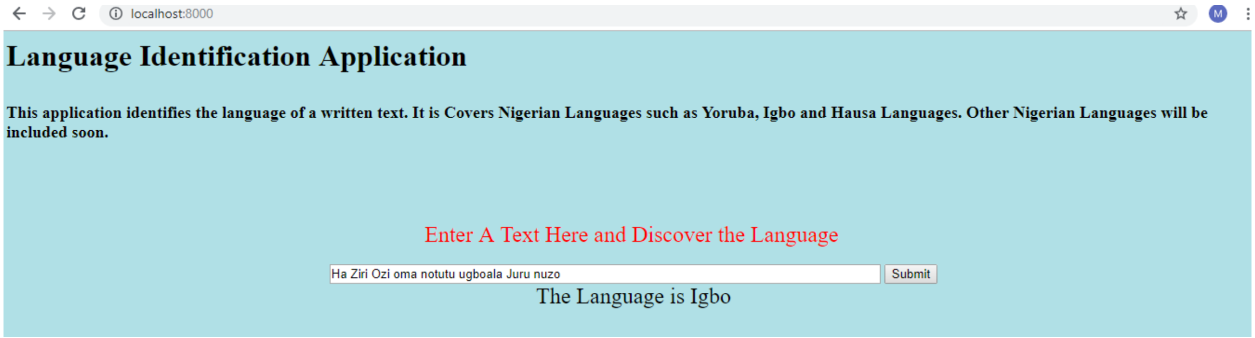
Language Identification Application

This application identifies the language of a written text. It is Covers Nigerian Languages such as Yoruba, Igbo and Hausa Languages. Other Nigerian Languages will be included soon.

Enter A Text Here and Discover the Language

Ha Ziri Ozi oma notutu ugboala Juru nuzo

Figure 4.9: Image showing the page where the user has put in a text in Igbo in the user interface



Language Identification Application

This application identifies the language of a written text. It is Covers Nigerian Languages such as Yoruba, Igbo and Hausa Languages. Other Nigerian Languages will be included soon.

Enter A Text Here and Discover the Language

Ha Ziri Ozi oma notutu ugboala Juru nuzo

The Language is Igbo

Figure 4.10: Image showing Igbo as the identified language of the user input from Figure 4.9

The application accurately identified the language of a given text for texts with 6 words and above. For texts with 1 to 5 words, classification was not absolutely accurate as there was misclassification in a few cases.

CHAPTER 5

CONCLUSION

5.1 Conclusion

The Text-based Identification of Nigerian Languages using N-grams is a project done with the aim of training the machine to identify the Nigerian language a document is written in. From researching and reviewing previous work done in this area, certain decisions such as the classification algorithm to use, the N-gram to implement and the size of the training data and test data, were made based on certain factors and limitation of the language pool in consideration. The Mutual cross entropy was used as the classification algorithm and the nature of N-grams used was that of n = 3, since the number 3 represents balance; words of length 1 or 2 were left the way they were therefore there were a few unigrams and trigrams also. To avoid combinatorial explosion, the language models used for the training data were pruned to a length of 1000 n-grams. Implementing all these was able to give quite an accurate language classification system.

The Statistical approach to Language Identification using N-grams as modeling features and the Mutual Cross Entropy as classification algorithm yielded a very accurate system, even with a small language pool of Yoruba, Igbo and Hausa Languages.

.

5.2 Recommendation

In bringing Nigerian Languages into the lime-light of Natural Language processing, this project can only play a part if it is made more robust i.e. by adding more Nigerian languages to the model. This would give way to other Natural language processes for the Nigerian languages as Language identification is a basic step in Natural Language Processing. Therefore, it is recommended that the system should be made more robust as much as possible. In increasing the language pool of the system, similar languages would be introduced e.g. Itsekiri is similar to Yoruba. This would help observe the results from such similar languages and improve on the accuracy of the system.

5.3 Future Work

Further work can be done on this project by including more indigenous Nigerian languages to the training data for training the system, such that the system does not only identify standard Yoruba, Igbo and Hausa but can also identify other Nigerian Indigenous languages.

References

Agheyisi R.N. (1984). Minor Languages in the Nigerian context: Prospects and Problems. *Word* 35.3 (1984): 235-253

Ahmed, B., Cha, S. H., & Tappert, C. (2004,May). Language identification from text using n-gram based cumulative frequency addition.In *Proceedings of Student/Faculty Research Day, CSIS, Pace University*, 12-1.

Baldwin, T., & Lui, M. (2010, June). Language identification: The long and the short of the matter. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics* (pp. 229-237). Association for Computational Linguistics.

Botha G. & Barnard E. (2005). Two Approaches To Gathering Text Corpora fron the WorldWideWeb. *Sixteenth Annual Symposium of the Pattern Recognition Association of South Africa, Langebaan, South Africa, 23-25 November 2005.*

Botha, G. R., & Barnard, E. (2012). Factors that affect the accuracy of text-based language identification. *Computer Speech & Language*, *26*(5), 307-320.

Botha G., Zimu V & Barnard E. (2006). Text-based Language Identification for South African Langauges. *SAIEE Africa Research Journal*, *98*(4), 141-146.

Cavnar, W. B., & Trenkle, J. M. (1994). N-gram-based text categorization. *Ann arbor mi*, *48113*(2), 161-175.

Dunning T. (1994). *Statistical Identification of Language*(pp 94-273). Computing Research Laboratory, New Mexico State University.

Duvenhage, B., Ntini, M., & Ramonyai, P. (2017, November). Improved text language identification for the South African languages. In *Pattern Recognition Association of South Africa and Robotics and Mechatronics (PRASA-RobMech), 2017*(pp. 214-218). IEEE.

Gebre, B. G., Zampieri, M., Wittenburg, P., & Heskes, T. (2013). Improving native language identification with tf-idf weighting. In *the 8th NAACL Workshop on Innovative Use of NLP for Building Educational Applications (BEA8)* (pp. 216-223).

Hoxha K. & Baxhaku A. (2007). Albanian Language Identification in Text Documents. *University of Tirana, Faculty of Natural Sciences, Department of Informatics and Department of Mathematics*

Ionescu, R. T., Popescu, M., & Cahill, A. (2014). Can characters reveal your native language? A language-independent approach to native language identification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 1363-1373).

Jauhiainen, T., Lindén, K., & Jauhiainen, H. (2017). Evaluation of language identification methods using 285 languages. In *Proceedings of the 21st Nordic Conference on Computational Linguistics* (pp. 183-191).

Li, H., Ma, B., & Lee, C. H. (2007). A vector space modeling approach to spoken language identification. *IEEE Transactions on Audio, Speech, and Language Processing*, *15*(1), 271-284.

Lui, M., Lau, J. H., & Baldwin, T. (2014). Automatic detection and language identification of multilingual documents. *Transactions of the Association of Computational Linguistics*, *2*(1), 27-40.

Martins, B., & Silva, M. J. (2005, March). Language identification in web pages. In *Proceedings of the 2005 ACM symposium on Applied computing* (pp. 764-768). ACM.

Muthusamy, Y. K., Barnard, E., & Cole, R. A. (1994). Automatic language identification: A review/tutorial. *IEEE Signal Processing Magazine*, *11*(4), 33-41.

Obinyan, G. A. (2010). The development of indigenous Nigerian languages for effective communication and professional use: The case of Esan language. *EJOTMAS: Ekpoma Journal of Theatre and Media Arts*, *3*(1-2).

Panich, L. (2015). Comparison of Language Identification Techniques (Doctoral dissertation, Bachelor's Thesis, Heinrich Heine Universität Düsseldorf).

Sibun P. & Jerey C. (2004). Language Identification: Examining the Issues. *The Institute for the Learning Sciences, Northwestern University, 1800 Maple Avenue, Evanston, IL 60201*

Singh, A. K. (2006, July). Study of some distance measures for language and encoding identification. In *Proceedings of the Workshop on Linguistic Distances* (pp. 63-72). Association for Computational Linguistics.

Zaidan, O. F., & Callison-Burch, C. (2014). Arabic dialect identification. *Computational Linguistics*, *40*(1), 171-202.

Appendix

Code for Modeling

```
import os

import string



def getFilePath(filename):

    path = os.path.join(os.path.dirname(__file__), filename)

    return path



def lang_model(filepath):

    file_object = open(getFilePath(filepath),"r")

    file_document = file_object.read()

    out_file = file_document.replace('\n', ' ')

    file_object.close()

    return out_file



def normalstring(file_document):

    pos = 0
```

```
language_nab = ""

while pos < len(file_document):

    if file_document[pos] == "<":

        temp = "<"

        while file_document[pos] <> ">":

            temp += file_document[pos]

            pos += 1

    else:

        language_nab +=  file_document[pos]

        pos += 1





split_language = language_nab.split()



wordlist = []

for word in split_language:

    for char in word:
        if char in string.punctuation:  #to remove punctuation marks
            word = word.replace(char, '')


    word = word.decode("utf-8").lower()
    wordlist.append(word)
    #print word


'''testlist = []
for word in wordlist:
    if len(word) == 1 or len(word) == 2:
        testlist.append(word)
    else:
        for i in range(0,len(word)-2):
            trigram = word[i:i+3]
            testlist.append(trigram)
for tri in testlist:
    print tri'''
```

```
trigfreq = {}

for word in wordlist:

    if len(word)==1 or len(word)==2:

        if word not in trigfreq:

            trigfreq[word] = 1

        else:

            trigfreq[word] += 1

    else:

        for i in range(0,len(word)-2):

            trigram = word[i:i+3]

            if trigram not in trigfreq:

                trigfreq[trigram] = 1

            else:

                trigfreq[trigram] += 1




#Implementing the Dictionary

total = 0
```

```
for element in trigfreq:

    total += trigfreq[element]



for element in trigfreq:

    trigfreq[element] = float(trigfreq[element])/float(total)

    #print element, trigfreq[element]



#print "Number of Trigrams: " + str(len(trigfreq))

#print "Total frequency of Trigrams: " + str(total)




#sorting and pruning the dictionary

sort_dict = sorted(trigfreq.iteritems(), key=lambda (k,v): (v,k), reverse= True)

keep_sorted = sort_dict[0:1000]



#for element in keep_sorted:

    #print element[0], element[1]
    return keep_sorted;



#lang_model("igbo.txt")
```

Code for Classification

```
import Project_code

import math



yoruba_data = Project_code.normalstring(Project_code.lang_model("New_yoruba.txt"))

igbo_data = Project_code.normalstring(Project_code.lang_model("igbo.txt"))

hausa_data = Project_code.normalstring(Project_code.lang_model("hausa.txt"))



Language_Lists = [yoruba_data, igbo_data, hausa_data]




def compare(test_file):

    #test_data = Project_code.lang_model(test_file)
```

```
#print("CURRENT_LOG",test_file)

test_data = Project_code.normalstring(test_file)

#for element in test_data:

    #print element



result = [0,0,0]

for index,language in enumerate(Language_Lists):

    for ngram,prob in language:

        for ng,pb in test_data:

            if ngram == ng:

                sim = float((prob * float(math.log(pb))) + (pb * float(math.log(prob))))

                result[index]+=sim



if result[0] < result[1] and result[0] < result[2]:

    print "Language is Yoruba"

elif result[1] < result[0] and result[1] < result[2]:

    print "Language is Igbo"

elif result[2] < result[0] and result[2] < result[1]:
        print "Language is Hausa"

    else:

        print "Not in database"



compare("Mo n lo si oko")
```

Code for Web Application

Index.html

```
<!DOCTYPE html>

<html>

<head>

    <title>

        Nigerian Language Identification Application

    </title>


</head>

<body style="background-color:powderblue;">

    <h1>

        Language Identification Application

        <p>

            <font size="+1">

                This application identifies the language of a written text. It covers Nigerian Languages such as Yoruba, Igbo and Hausa Languages. Other Nigerian Languages will be included soon.

            </font>
        </p>
    </h1><br><br><br>

    <form action="/result/" method="post">
        <center>
            <label style="color:rgb(255,0,0);" for="user_input"> <font size="+2">Enter A Text Here and Discover the Language</font> </label><br><br>
            <input id="user_input" type="text" name='user_input' size = "80">
            <input type="submit" value="Submit">
        </center>



    </form>

</body>
</html>
```

Result.html

```
<html>

<head>

    <title></title>

</head>



    <body style="background-color:powderblue;">

        <h1>
        Language Identification Application
        <p>
            <font size="+1">
                This application identifies the language of a written text. It is Covers Nigerian Languages such as Yoruba, Igbo and Hausa Languages. Other Nigerian Languages will be included soon.
            </font>
        </p>
    </h1><br><br><br>


        <center>
            <font size = "+2">
            {% if result %}
            {{result}}
            {% endif %}


        </font>
        </center>
```

```
	</body>

</html>
```

Views.py

```
from django.http.response import *

from django.shortcuts import *

from LangIdenApp.Comparison_function import *

import os

from django.views.decorators.csrf import csrf_exempt




def index(request):



	return render(request, 'index.html')



@csrf_exempt

def result(request):

	user_input = request.POST.get('user_input')
```

```
    print('VIEW_LOG', user_input)



    result = compare(user_input)

    return render(request, 'result.html', {

        'result': result



    })



def __str__(self):

    return self.result.encode('utf8')
```

url.py

```
""wapp URL Configuration



The `urlpatterns` list routes URLs to views. For more information please see:

   https://docs.djangoproject.com/en/1.11/topics/http/urls/

Examples:

Function views

   1. Add an import:  from my_app import views
```

```
    2. Add a URL to urlpatterns:  url(r'^$', views.home, name='home')
Class-based views
    1. Add an import:  from other_app.views import Home
    2. Add a URL to urlpatterns:  url(r'^$', Home.as_view(), name='home')
Including another URLconf
    1. Import the include() function: from django.conf.urls import url, include
    2. Add a URL to urlpatterns:  url(r'^blog/', include('blog.urls'))
"""
from django.conf.urls import url
from django.contrib import admin
from views import *


urlpatterns = [
    url(r'^admin/', admin.site.urls),
    url(r'^$', index),
    url(r'^result/$', result),


]
```

Manage.py

```
#!/usr/bin/env python
import os
import sys

if __name__ == "__main__":
    os.environ.setdefault("DJANGO_SETTINGS_MODULE", "wapp.settings")
    try:
        from django.core.management import execute_from_command_line
    except ImportError:
        # The above import may fail for some other reason. Ensure that the
        # issue is really that Django is missing to avoid masking other
        # exceptions on Python 2.
        try:
            import django
        except ImportError:
            raise ImportError(
                "Couldn't import Django. Are you sure it's installed and "
                "available on your PYTHONPATH environment variable? Did you "
                "forget to activate a virtual environment?"
            )
        raise
    execute_from_command_line(sys.argv)
```

settings.py

```
"""
Django settings for wapp project.

Generated by 'django-admin startproject' using Django 1.11.16.

For more information on this file, see
https://docs.djangoproject.com/en/1.11/topics/settings/

For the full list of settings and their values, see
https://docs.djangoproject.com/en/1.11/ref/settings/
"""

import os

# Build paths inside the project like this: os.path.join(BASE_DIR, ...)
BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))



# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/1.11/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'eq&mme6ux!&u5&o@j91(eh4*izn5ll1)le=-l)3jc!epp9bl)s'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True
```

```
ALLOWED_HOSTS = []


# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'wapp.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [os.path.join(BASE_DIR, 'templates')],``
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]

WSGI_APPLICATION = 'wapp.wsgi.application'


# Database
# https://docs.djangoproject.com/en/1.11/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
    }
}
```

```
# Password validation
# https://docs.djangoproject.com/en/1.11/ref/settings/#auth-password-validators

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]


# Internationalization
# https://docs.djangoproject.com/en/1.11/topics/i18n/

LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'UTC'

USE_I18N = True
```

```
USE_L10N = True


USE_TZ = True



# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/1.11/howto/static-files/


STATIC_URL = '/static/'
```