Language identification

by

Andreas Socratous

Final Year project submitted in partial fulfilment of the requirements for the Degree of

Bachelor of Science in Computer Science

Department of Computer Science

School of Sciences

University of Nicosia

May, 2016

***Language Identification***

***By***

***Andreas Socratous***

This Final Year Project has been accepted in partial fulfilment of the requirements for the Degree of

Bachelor of Science in Computer Science

Project Advisor:

/ /

(Name) (Signature) (Date)

Examiner:

/ /

(Name) (Signature) (Date)

Examiner:

/ /

(Name) (Signature) (Date)

Department of Computer Science

School of Sciences

University of Nicosia

**Abstract**

The purpose of this project is to develop a program that is using N-Grams, in order to identify the language of the given file.

In summary, the program compiles with several embedded files, in order to develop a profile for each language, in the current case, is English, French and Spanish. Then, the program will try to compare the input profile with each of the pre-made profiles, calculate the distance between the two compared profiles and find the smallest distance. The language profile which have the smallest distance, means that the probability of this language to be the input file is the maximum.

In order to create the profiles, there was needed to split each file into tokens, and from tokens to N-Grams. The N-Grams take from 1 to the length of the token and spit it into sub-strings. The data structure is used is at first Linked Lists for test and then Hash Table, for better runtime. The programming language used is C#, because, there are a lot of helpful ready-made functions to use and an “improved” Hash Table, called the Dictionary.

**Acknowledgements**

I would like to thank my parents for their encouragement and financial support during my studies. Also my project advisor for his critical questions, which forced me to read more, think critically and write with better clarity and for teaching me that we learn by doing. Special thanks also go to my friend John Smith, who read my project report and helped me with the grammar and style. I would also like to express my gratitude to my Manager in work, Mr YiannisIoannou and my co – workers in Computer Center in University of Nicosia, which helped me a lot to the understanding and solving the project.

**Table of Contents**

**Chapter 1:**

* 1. **Text classification**

Text classification is a method used in Computer Science, most used in Artificial Intelligence and Machine Learning areas. The idea of text classification is to make results out of a lot amount of data. For example, we can extract data from Facebook or Twitter and based on each post, that each person posted, we can make predictions, identify the feeling of each post (if that person is happy, sad etc). We can also use it in emails, for categorization between the spam emails and the emails sent from real persons, or storing each email based on the topic. For example, if that email contains details about sellings or advertisements, we can store it in different folders.

John Graham said :

*“A text classifier is an automated means of determining some metadata about a document. Text classifiers are used for such diverse needs as spam filtering, suggesting categories for indexing a document created in a content management system, or automatically sorting help desk requests”. --- John Graham-Cumming,*[*Naive Bayesian Text Classification*](http://aitopics.org/link/naive-bayesian-text-classification)*.*

* 1. **Brief History of Text Classification**

Text classification was first introduced back in the beginning of 1960. Due to the high amount of data nowadays, it has become very popular, in order to solve specific tasks based of what the needs want. From its first appearance until today, many techniques has been introduced. Each technique serves differently, based on the data that are being processed. These techniques differs on accuracy, speed and precision. Some techniques that were used up to these days are:

1. Naïve Bays
2. Nearest Neighbour
3. Support Vector Machine
4. N grams
5. Decision Trees
6. Artificial neural network

In this project, I have decided to use the N –Gram technique, because it seemed more likely to be implemented in my choice of programming language, C#.

**Chapter 2:**

* 1. **Identify the problem**

The problem in this project was to identify the language using N-Grams. In order to do so, Dr Athena Stassopoulou, my advisor, suggested some papers for reading to understand the differences between the different solutions that problem has. Some solutions were the Naive Bayes approach and the N-Grams approach. For this project, the method used was N-Grams.  
  
After reading the (enter text here) paper, I decided to use the N-Grams method, because in my opinion, it was more programmatically efficient and practical to be implemented.   
  
The idea behind N-Grams with a text, for example “beer” is that the text will be broken into: (consider where ‘\_’ is the space character)

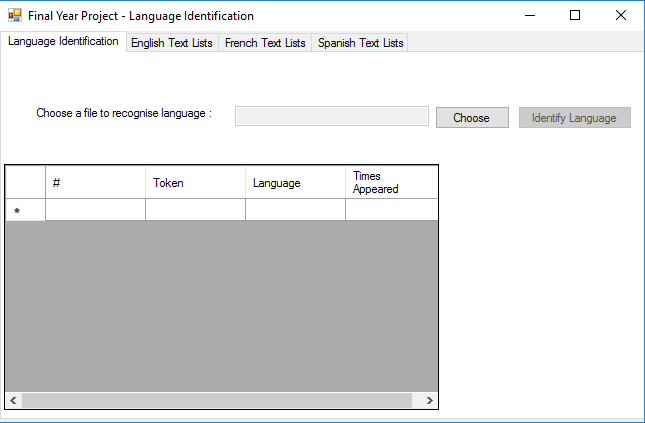
1. Bi – Grams : \_b, be, ee, er, r\_
2. Tri – Grams : \_\_b, \_be, bee, eer, er\_, r\_\_
3. Quad – Grams : beer

That means that the N-Gram algorithm should be applied from number 1 to the length n of the current token. This will be done with every token.  
Now, let’s talk about the tokens. A token is basically every single world individually in all the text documents. Every text document will be spitted into tokens and stored in a data structure. In this case, Linked Lists are used.

* 1. **Decision of the Programming language**

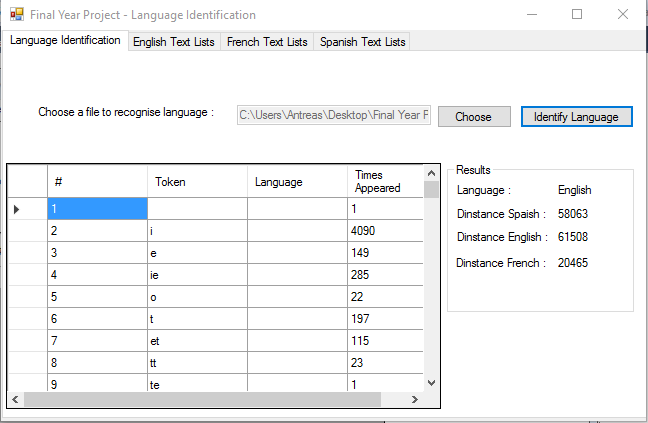
Based on my current knowledge on programming languages, I decided to use C# for the implementation of this project. This programming language provides many flexibilities both with GUI (Graphical User Interface) and with libraries. It has many build-in ready-made functions that will be very helpful for the implementation of this project.

For a better review of what is going on with the tokens and the subtokens, I decided to include in the project 4 datagrids, objects of C#. This object will help us see the actual counter of every subtoken that occurs. There are 4 datagrids, one for each language that are included in our project, and a 4th one for the input language. The final Graphical Interface looks like this :



With the datagrid filled up with the input text and the language identified, the Interface looks like this :

* 1. **Breaking the problem into sub-problems**

****

In order to achieve such a project goal, first it is needed for the goal to be broken up in a few and more easy sub-goals or tasks. With this method, the time, the effort and the complication of the problem will be greatly reduced. Finally, the small tasks will be assembled and end up with the final product / project.

The idea is to read all .txt files as resources, using the *System.IO.StreamReader*library, that is available in this programming language. This library has a build in function called *“ReadAllText()”*, that will read all the txt file and store it in a string variable.

Now that we have out text files into string variables we can proceed further in our project. We can take a string variable came from our text file and split it into words (or tokens).

For example, let our text be :

*“This is a text for being used in text classification”*

The first step of achieving the desire goal is to split the test text into tokens, in our case, into words. Therefore, our list will contain the following tokens :*“This”, “is”, “a”, “text”, “for”, “being”, “used”, “in”, “text”, “classification”* . This subgoal is achieved by using a ready made function called *“Text.Split(‘ ’);* . This function will split the text whenever it finds the character ‘ ’ (space character). This is done for every document and stored in the appropriate dictionary (hash table). For example, the *English* documents will be stored in the *English* list, the *Spanish* documents in the *Spanish* list and so on.

The second step is to take each token from the list and examine it. For example, if the token “being” is taken, it will be checked for its length. The length of this token is 5. That means that in this token it can be applied up to the 5-Gram. Therefore, our subtokensare :

For 5-gram: being

For 4-gram: bein, eing

For 3-gram: bei, ein, ing

For 2-gram: be, ei, in, ng

For 1-gram: b,e,i,n,g

After this process, the subtokens are stored in the appropriate dictionary/hash table. If in the process a token already exists, then there is a counter that counts how many times the current subtoken occurred and it is increased by 1. This will help us keep track of how many times in each language a subtoken occurred, so that we would be able to compare it with the dictionary of the input document.

The code I used to implement this is the following :

bool found = false;

foreach (var key in dictionary)

{

if (key.Value.getToken() == ngram)

{

key.Value.increaseTimesOccured();

data.Rows[key.Key].Cells[3].Value =key.Value.getTimeOcc();

found = true;

}

}

if (!found)

{

node temp = new node();

temp.setToken(ngram);

index++;

data.Rows.Add(index, ngram, "", 1);

dictionary.Add(index, temp);

}

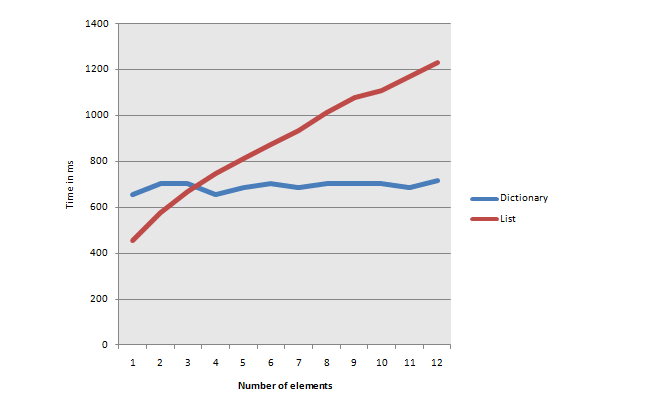
* 1. **Why use Dictionary instead of Lists**

For this project to be implemented, I have used a class (or a struct). If this struct was stored in a list, that would require a lot of memory. In our case, we have quite a lot of data, if we take into account that a simple word of length 5 has 15 different subtokens. That means that for just a word we have already filled 15 memory allocations (not to mention that this is a struct, so the memory allocation is increasing) . Dictionary, on the other hand, is lighter in terms of speed and memory. It is used widely in the area of text classification.

It is worth mentioning that what data structure is widely used in text classification is the Hash Table. Dictionary is just a light improvement of a Hash table in C#, as far as I have searched.

How does a Hash Table or a Dictionary works? It actually use a key in order to keep track all of the data it has been given. For example, lets assume that we have a Hash Table with string values. If someone is searching for a specific string, the Hash Table will not search for the string itself, but for the key containing the specific string. This is very helpful in our project, because if we are storing whole stucts of data in a List and trying to search for it, that will take time. I have made some tests. The time to load of this specific program using Lists took approximately 10 - 12 minutes to load. Instead, the Hash Table (in our case, the Dictionary) took only 20 seconds. That is a lot of difference, if we are storing a lot of data.

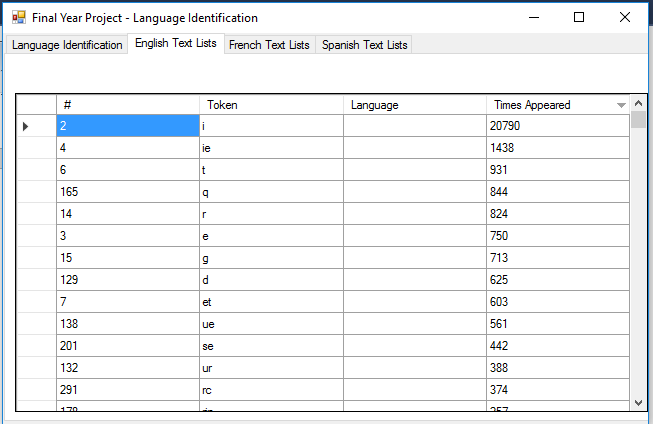
Here I am showing a picture comparing the Dictionary and the List in terms of data and time.



Based on that graph, for a few elements, Lists can be used without causing any runtime problems, but with a lot amount of data, a Hash table or a Dictionary is required.

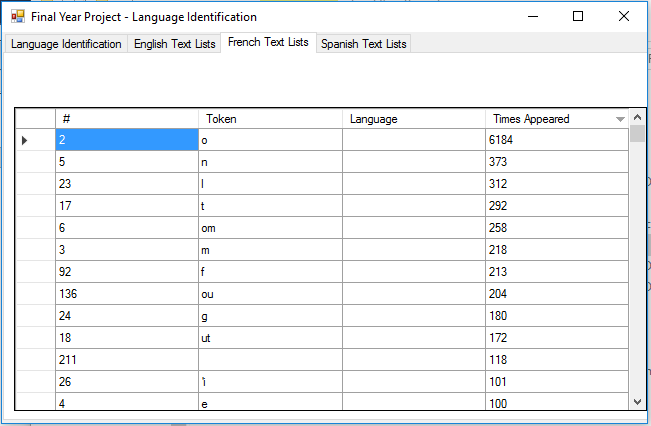
**Chapter 3:**

* 1. **Results**

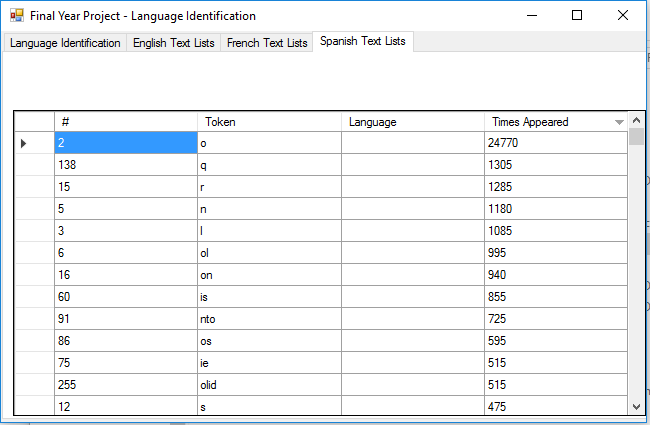
When the process of creating the profiles and the form is loaded, the Data Grid Views are showing all the tokens, along with their time that appeared in all of the texts. The Dictionary itself has no sorting method, as we mentioned before, it is using the key value in order to identify the objects. So, I have decided to take advantage of the Data Grid Views that are hosting all of the tokens and the time that appears and use the method“*<DataGridViewName>.Sort(*“*<DataGridViewName>..Columns[<Column number or column name>], ListSortDirection.Descending);”* to show the dictionary sorted.

As it shown below, the English profile is showing the highest number of occurrences first, in this case, is the letter “i”.

This picture shows the French Dictionary. As shown in the picture, the highest frequency of token occurrence is the letter “o”.

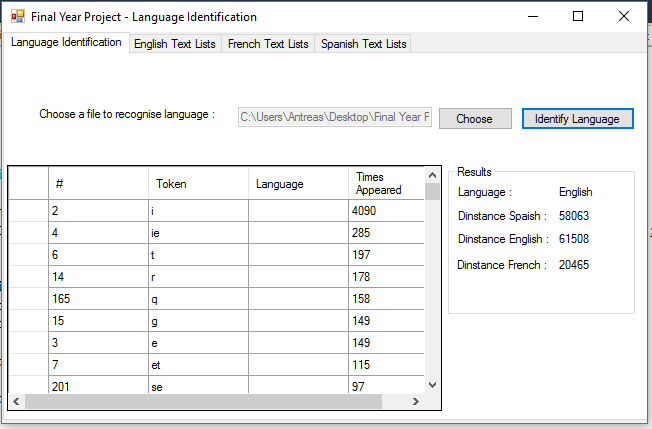
****

This picture shows the Spanish Dictionary. As shown in the picture, the highest frequency of token occurrence is the letter “o”.

****

1. **Results worth mentioning**

After making some experiments with the final software, there is a remarkable result that is occurred. As it is known, Spanish, English and French are using the same characters, Latin. But although the Spanish Language and the English Language are a bit different, the distance of an English document as an input is very close to each other, both for English and Spanish Language. French is very far away from the others. But, in some cases, although it was an English document that was given as input, the result pop up as Spanish.

As we can see in the picture below, the Spanish Distance is 58063 and the English is 61508. These are very close numbers.

**References :**

**English Documents :**

1. <http://www.scientificamerican.com/article/how-nuclear-power-can-stop-global-warming>  
2. <http://www.britannica.com/technology/tank-military-vehicle>  
3. <http://www.everydayhealth.com/diet-nutrition/101/benefits-of-healthy-eating/eating-for-energy.aspx>  
4. <http://www.weeklystandard.com/article/1001589>  
5. <http://www.theguardian.com/technology/2012/oct/21/john-naughton-foursquare-social-networking>

**French Documents :**

1. <http://www.huffingtonpost.fr/2015/09/13/futur-intelligence-artificielle-humanite-immortalite-25-ans-2040_n_8123014.html>  
  
2. <http://www.topsante.com/minceur/nutrition-minceur/conseils-minceur/minceur-on-apprend-a-manger-equilibre-10853>

3. <http://energie-nucleaire.net/qu-est-ce-que-l-energie-nucleaire>

4. <http://www.stop-tabac.ch/fr/les-effets-du-tabagisme-sur-la-sante/quels-effets-pour-les-qpetitsq-fumeurs>  
  
5. <http://www.lemonde.fr/pixels/article/2015/12/22/apprentissage-l-intelligence-artificielle-une-eleve-de-plus-en-plus-douee_4836339_4408996.html>

**Spanish Documents**

1. <https://www.des.umd.edu/os/lead/spanish_factsheet.html>
2. <http://www.magodeoz.com/biografia>
3. <http://www.cafelanacional.com/nosotros/proceso-de-produccion-del-cafe>
4. <http://vidaverde.about.com/od/Tecnologia-y-arquitectura/tp/Como-Funciona-Un-Molino-De-Viento.htm>
5. <http://www.mineducacion.gov.co/1621/article-183910.html>

**Helpful sites for C# :**

1. <https://msdn.microsoft.com/en-us/library/5ycd1034.aspx?f=255&MSPPError=-2147217396>
2. <http://stackoverflow.com/questions/18113278/populate-a-datagridview-with-sql-query-results>

3. <http://10tec.com/articles/datagridview-filter.aspx>

4. <http://stackoverflow.com/questions/3309188/how-to-sort-a-listt-by-a-property-in-the-object>

5. <http://www.dotnetperls.com/dictionary-time> (List - Dictionary comparison)

**History, webpages and e-books :**

1.<http://aitopics.org/topic/text-classification>