PROVA FINALE

Statistical Language Identification of Short Texts

candidato
Fela Winkelmolen

relatore
Viviana Mascardi

UNIVERSITÀ DEGLI STUDI DI GENOVA
Facoltà di Scienze Matematiche Fisiche e Naturali
Corso di studio di Informatica

Marzo 2010

Contents

Chapter 1

Introduction

1.1 Motivation

Being able to identify the language of a given text proves useful in a large number of applications. It can be used by indexing tools, such as those used in search engines and in desktop search software, to provide the ability of making language specific queries. Text-to-speech applications capable of reading multiple languages need to first identify the language correctly. It might prove useful to be able to use automatic translation tools in those cases where the source language is not known. Finally, OCR digitalization of written text can make use of language identification, both to classify the output document and to improve the OCR process itself (as language specific knowledge might be used to improve the accuracy of the conversion).

Many of the difficulties posed by language identification have therefore already been studied, and a variety of methods that given enough text, are able to quickly and accurately identify a language, have been proposed. Those methods are largely resistant to spelling errors.

However, for short strings the performance degrades quickly. The literature devotes little attention to the accuracy of state-of-the-art algorithms when very short texts are tested. Results obtained with one of the most widely used algorithms [3] are shown in Figure 1.1. It can be seen how for documents that are 25 characters long the language is identified correctly only about 80% of the times. There are many situations where a higher accuracy is required.

For example, we might want to develop a multilingual spellchecker used by an instant messaging (chat) client, able to guess the language that is being used. With the existing techniques, there will be a non negligible chance of the language not being identified correctly until *after* a number of messages

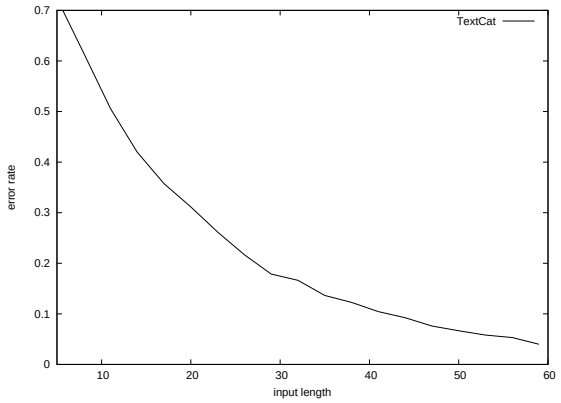

Figure 1.1: identification accuracy

have already been sent.

An even more striking example is provided by SMS texts, which are usually very short. Most phones use T9[1] to allow for fast typing of alphabetic messages using only the 9 number keys. To work properly such a system needs to know the language being used. If a user writes regularly in two or more languages it would be convenient to initially propose words in all those languages, and switch to only one language as soon as it has been correctly identified.

Another scenario that has never been addressed in the literature is one where we want to exactly know *which* parts of a multilingual document are written in which language. In this case it becomes necessary to have statistical meaningful data for very short pieces of text. This information can then be combined using the knowledge that nearby text is likely to be in the same language.

Considering these situations, where the language of short text strings needs to be identified with a high accuracy, the object of this thesis is to analyze some of the existing algorithm, with particular attention to their performance on short texts, and to propose and implement a number of improvements.

1.2 Approach

While there is a wide range of published work on the subject of language identification of written texts [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 13, 14], it is not easy to compare the various algorithms. The comparisons made in

[1] http://en.wikipedia.org/wiki/T9_(predictive_text)

the literature often only deal with very specific use cases, and are not easy to reproduce and to generalize. Also, almost all conclusions drawn by the authors of the algorithms are empirical in nature.

Hence, the thesis tries to shed some light on the theoretical foundations that make the algorithms work. Additionally, a number of comparisons have been made, using existing algorithms and variations thereof, those were carried out with particular attention towards the performance on very short strings of text.

A multilingual text corpus was needed to train the various algorithms. As shown, the quality of the corpus can greatly influence the results of the identification process. Two methods were developed to harvest language specific textual data, using subtitles downloaded from the Internet proved to work best.

In many of the situations where the existing approaches still lack in accuracy, the use of external information can provide an estimate of the prior probability of the languages (the probability before even having looked at the contents of the text): a clever use of this information might dramatically reduce the likelihood of wrong guesses. An example can be given looking at a language aware spell-checker. If the operation system is installed in a certain language, that will be the most likely language for the user to use. This preliminary guess can be much improved by looking at the language in which the user usually writes. If we are checking text in an instant messaging application or an email client, an even more sophisticated guess can be made looking at the language usually used when writing to the given person.

There is not a straightforward way to combine this information with the results of the thus far used classification methods, as they do not give accurate information about the confidence of the result. A naive Bayes classifier was therefore developed, it proved to be able to give a good approximation of the confidence of the results. Then it was proved how combining these results with prior knowledge about the language probability distribution can dramatically improve the identification accuracy.

1.3 Structure of the Thesis

In Chapter 2 an overview of previous research dealing with language identification is given, also related fields are briefly explored.

Chapter 3 describes how a multilingual corpus was built, using subtitles downloaded from the Internet, and some information is given about corpora building by web page crawling.

Chapter 4 provides a formal description of the problem at hand, showing

how feature comparison can be used for language identification of written texts. Some of the existing algorithms are then analyzed and the implementation of a naive Bayes classifier is described. The last section of this chapter shows how to combine the results of this algorithm with prior knowledge about the language distribution.

Chapter 5 gives an overview of the results obtained and provides some directions for further work.

Chapter 2

Background

The most widely used methods to programmatically identify the language of a given text compare the characteristics (usually called features) of the text with those most common in the various languages. The features most often compared are *n-grams*. Given a string, an n-gram is defined as any sequence composed by n consecutive characters. 3-grams are commonly called *trigrams* and are the most widely used n-grams.

2.1 Review of the literature

Most free programs providing language identification of text are based on the work of Cavnar and Trenkle [3]. They have been the first writing about the subject, describing a simple algorithm comparing the 300 most frequent trigrams in every language with the top 300 trigrams of the text given in input. The trigrams are ordered from the most frequent to the least frequent, without keeping any additional frequency information. The language whose profile is most similar to the profile of the input text is then chosen. This method has been proved to work well in practice given long enough input strings. Accuracy is nearly 100% for texts more than 300 characters long. For simplicity in the rest of the thesis this algorithm will be called ROC*NNN*, where ROC stands for Rank Order Classifier, and *NNN* is the number of features stored for each language (thus the original version will be called ROC300).

A number of improvements have since been proposed. Prager [14] has compared the results obtained using n-grams of various length, words, and combinations thereof. Those features were weighted using their inverse document frequency (features found in less languages were weighted higher; the terminology is a remainder from the document retrieval field), while the dis-

tance used was the cosine distance of the vectors normalized in feature space. Viewing the problem as a geometric one can prove useful and can provide mathematical foundations and intuitions for further improvements.

Ahmed et al. [1] calculate the distance from a given feature model using a custom "cumulative frequency addition" measure in comparison with an algorithm similar to naive Bayes. A database is used to store the frequencies of *all* trigrams encountered in the training set.

Other classification methods such as decision trees [8] and vector quantization [13] have been proposed. While looking promising, it is hard to reach final conclusions, as not much information is provided about the exact methodologies by which the results were obtained.

MacNamara et al. [10] explore the application of specific architecture of Recurrent Neural Networks to the problem, showing they perform worse than trigrams methods. It must however be noted that a wide variety of Neural Networks exists, and other variations might give better results.

Elworthy [7] proposes to not process the whole document but only as many characters as needed to reach the required confidence level. This speeds up the text categorization, specially in the case of very long texts, under the assumption that all documents are monolingual.

Some research focuses on specific languages, scripts, or geographical locations. Murthy and Kumar [12] focus on Indian languages using multiple linear regression [11] to make pairwise comparisons. They tell us Indian scripts are not alphabetic in nature so not characters or bytes but "aksharas" (similar to our syllables) should be used as atomic units. Ljubešić et al. [9] make an even higher use of language specific knowledge to build a classifier able to distinguish between Croatian, Serbian and Slovenian, languages which have proved to be very hard to distinguish from each other. Botha [2] compared various methods to distinguish among the 11 official South African languages.

Information about computational intensity is almost never given, which is unfortunate, as for example choosing the language having the shortest Hamming distance from the spell checked text is likely to get very accurate results, but this approach would be very inefficient in most cases.

2.2 Related work

Language identification can be seen as a special case of the the more general problem of classification, and more specifically in the case where a wide number features can be observed. Many of the techniques used are also widely studied in other fields such as bio-informatics, computer vision and

artificial learning.

Spoken language identification, used for example to route phone calls to the appropriate operator, can be solved using much the same techniques as those described here. The main difference being that, instead of characters, phonemes are used, with additional difficulties like the need of extracting the phonemes from spoken words.

Language identification is a subset of text categorization, which has been extensively studied (mostly in relation to document retrieval).

2.2.1 Character encoding detection

When identifying the language of a text sometimes a sequence of bytes is given, without any prior knowledge about it's encoding. In this case, before the language can be guessed the encoding should be identified. One way to work around this problem is by making the classifier work directly on the bytes, ignoring the encoding altogether. This however assumes the classifier has been trained on all possible encodings in which a language can be found, this is therefore not the optimal solution.

Given any sequence of bytes, usually most encodings are not valid, this drastically reduces the possibilities. At this point the encoding can be guessed using the fact that encoding a text incorrectly results in a large number of very rarely used characters. Thus the correct encoding can be accurately identified by a simple comparison of bigram or even unigram (character) frequencies. Guessing the encoding of a text is sometimes done in web browsers and text editors.

The approach followed in this thesis is to make sure all text is converted to UTF-8[1] beforehand. At which point all strings can be viewed simply as sequences of characters.

[1] http://en.wikipedia.org/wiki/UTF-8

Chapter 3

Language corpora

All algorithms using statistical comparison of language features need some amount of text in the languages of interest to be trained. It is easy to see how the bigger and the more representative of the language the data is, the better the algorithm will perform. How much exactly will be explored later.

Unfortunately not many corpora are freely available, and none was found satisfactory for the thesis' purpose. A new corpus was thus built, as described in the next sections.

3.1 Existing corpora, and ways to build them

This section provides an overview of the corpora available and the sources from which a corpus can relatively easily be built.

Big existing corpora are freely available in some languages[1][2], but homogeneous corpora for a relatively high number of languages were needed.

The Universal Declaration of Human Rights has been translated into at least 375 languages and dialects, and is therefore much used when text in a large number of idioms is needed. The human rights declaration and Wikipedia are the most used sources in the language identification tools that have been analyzed. It was not used because it is too short (the English version is less than ten thousand characters long) and because it covers a very specific topic.

Wikipedia is also available in a wide range of languages, it is therefore quite easy to harvest a large number of data in any of the languages

[1] http://corpus.byu.edu/

[2] http://www.clres.com/corp.html

Wikipedia exists using, for example, it's "random article" function. But once again the text obtained is not very representative of the average use of a language. In most languages the majority of the pages are articles about people, places, or biological species.

The Internet may be the most obvious place to look for data, as it contains huge amounts of text in almost every language, spanning every field. In the next section this possibility is explored in more depth.

Film subtitles in various languages can be downloaded from a number of websites. They contain a nice compromise between formal and informal language (whereas most other sources only provide text written in formal language). A flaw, perhaps, is that it is mostly direct discourse, so again not really representative of the average text. Section 3.3 describes how subtitles were used to build a multilingual corpus.

BBC Worldservice contains articles in a wide range of languages[3] which might be easily scraped.

XNLRDF [15] describes itself as an Open Source Natural Language Resource Description Framework, containing, among other useful resources, a number of heterogeneous corpora, free from any kind of copyright restriction.

Project Gutenberg contains hundreds of public domain books, mostly in English.

Combining some of the sources above, in a clever and statistically sound way, may provide even better results.

It may be noted that the notion of being representative of a language has not been defined with much precision. It can be said, however, that we want a corpus to have the same features as, on average, the text we expect to match with it. So the meaning of "good" and "representative" may change depending on context.

3.2 The Internet and Google

A script was written which tries to build language corpora downloading random pages from the Internet. First a Google search in the target language

[3] http://www.bbc.co.uk/worldservice/languages/index.shtml

is performed, using queries that are not specific to any topic, such as numbers or the most frequent words in a given language (such as "the" or "in" in English). The results given by Google are extracted from the HTML output using standard website scraping techniques, as the only public API for the search engine available can be used only inside web applications. A few searches allowed to conclude that most pages found this way, even discarding the first few hundred entries returned, are homepages of news websites. News articles are an acceptable (and often used) basis for a language corpus. The homepages themselves mostly contain summaries of articles with links, thus a random link taken from each page is selected as possible target.

To convert the selected page into plain-text a number of textual browsers were tried, all of which have a `-dump` option, to dump the text of a given website to standard output. *w3m*[4] gives the cleanest output, and more importantly has the best character encoding detection.

The data obtained this way contains a lot of noise and unusable characters: the majority of the text comes from image captions, menus, navigation bars, links, etc. The fact that *w3m* formats the pages also made most text very hard to use, as in the following example:

```
this is a menu   this is the main content, unfortunately
> click here <   in  the  resulting   string  this  will
>  a link    <   be mixed  with the  content of the menu
```

In an attempt to clean up the text and only select the most promising data, the script first strips all empty space at the start and end of every line. Then the text is separated into blocks using empty lines as separators. Thus only the blocks of a given size and containing a low percentage of consecutive non-alphanumeric characters are selected. This makes the program discard most of the websites. Testing done on a few dozen of pages in English and Italian showed that still most of the results contain unwanted noise. Looking at these results it was decided to also try another approach, which proved more promising, as shown in the text section.

A better way to extract text from web pages might be to parse the HTML data directly, and work on the DOM tree to select the text. Research done on web crawlers for search engines will provide useful guidance.

An Crúbadán[5] is a web crawling software based on some of the same ideas and has already built corpora for 447 languages. Unfortunately neither the software nor the collected data are available to the public.

[4] http://w3m.sourceforge.net/

[5] http://borel.slu.edu/crubadan/

3.3 Subtitles

To obtain subtitles opensubtitles.org's XMLRPC public API was used[6]. Film subtitles can by searched for, among others, by title, language and Internet Movie Database ID. Given the inability to find films available in all languages needed, data was collected for every language independently. The keywords used were common words (like "the" and "and"; all film titles are in English) and year numbers (2008, 2009, ...). It was noticed that not all results that should match are returned, instead, the number of results of a query are quite non deterministic, but given enough queries eventually enough data gets collected.

There is no easy way to identify the character encoding of the subtitles. Additionally, as the contents are being contributed by users of the site, often the language information is not accurate. Various heuristics to determine the character encoding of the downloaded files were tried. It turned out almost all files are either encoded in UTF-8 or in a local encoding varying according to the language.

A script was developed that for every file presents the user with the first 500 non ASCII characters and a few lines of text (containing those characters if any), giving the user a choice between accepting the file, trying another encoding or discarding the subtitle. This way all subtitles were manually checked, and those not being in the supposed language or having other deficiencies were discarded.

A lot of the subtitles contain the character I (uppercase i) instead of l (lowercase L), as a consequence of being extracted from movies using OCR technology, which sometimes has difficulty distinguishing between the two characters. All subtitles containing an uppercase I preceded by a lowercase character were thus removed. Sometimes the only wrongly spelled word was 'Il', showing up as 'II'. It is a common word in both French and Italian, thus in the subtitles for those languages all occurrences of 'II' were replaced by 'Il'.

Some subtitles have no diacritics in them, where their text, spelled correctly, should contain them. For every language a few such subtitles were kept, as a percentage of the text in a language is actually stored that way.

Subtitle lines can not be very long, instead they are split into multiple lines. This has been undone, using '.', '?' or '!', followed by a newline, as sentence separators. When lines started or ended with ellipsis ('...'), those were removed. Lines starting and ending with characters like '#' and '~' usually contain song lyrics which have a high chance of being in the wrong

[6] http://trac.opensubtitles.org/projects/opensubtitles/wiki/XMLRPC

language and were therefore discarded.

This way subtitles from 22 languages were downloaded. For each language a corpus of approximately 500 thousand characters was generated, another 50 thousand characters of text were kept in a separate file for testing purposes. All languages chosen use the Latin script, the only exception being Greek. Having one non Latin script makes sure the results are easy to generalize to other languages, including those which use non Latin scripts composed of a small number of characters. Scripts like Chinese have not been looked at.

3.4 Importance of the training corpus

To evaluate the impact of the use of different corpora I compared the trigrams provided by TextCat[7] (and used by most other free language identification tools) with those extracted from the subtitle corpus. It does not come as a surprise that, when using the testing data obtained from the subtitles, the trigrams obtained from the same kind of text provide a more accurate identification. The gap closes considerably when testing the first two chapters of the novel "The Picture of Dorian Gray", a comparison is shown in Figure 3.1. Lastly, as seen in the same figure, when English news articles were tested, the results were inverted and the subtitle corpus performed worse. This makes it likely TextCat has been trained using a corpus composed mostly by news articles or text using a similar vocabulary, while it is reasonable to say that a novel uses terminology more similar to the one used in movie scripts.

While there is no evidence of my corpus being better that the one which has been used to train TextCat, it is clear that the choice of the corpus does have a big influence on the results. Knowing the context of the application developed can thus be helpful to improve it's performance, by using this information to appropriately select the training corpus.

One final observation can be made, looking at Figure 3.1. It can be observed how text from news articles is generally harder to identify. Most likely this is due to the wider vocabulary used and the presence of many proper names and foreign words.

From here on, unless differently specified, both the corpus used to train the algorithms and the testing data are those obtained from the subtitles. In the comparisons, input data in the languages Dutch, English, Italian, French, German, and Portuguese is used.

[7] http://odur.let.rug.nl/~vannoord/TextCat/

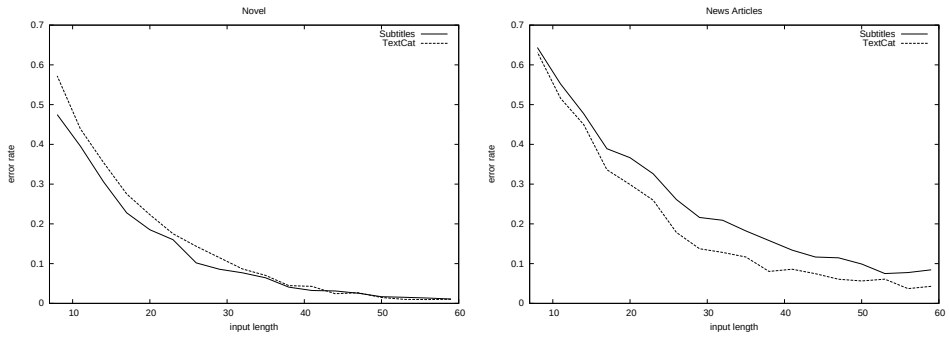

Figure 3.1: A comparison of TextCat versus the same algorithm trained with the subtitle corpus. The leftmost graph shows the accuracy with which text from a novel was identified, while the graph to the right shows the performance identifying text from English news articles.

Chapter 4

The modeling of features

Most approaches studied, in a particular way those based on n-grams distribution, can be viewed as composed by two steps. First the peculiarities that make the languages differ (the features) need to be elicited, and models or profiles of the various languages need to be built. Then a method is devised to test how well a certain text fits with these models. Given any text the language having the best fitting model will be chosen.

These two steps will be analyzed in detail. But first it will be useful to provide an accurate description of the exact problem we are trying to solve.

4.1 Formal description of the problem

There are a number of Languages (or classes) in which a text can be written. Given any text in input it will have, for each language in the given set, a certain known prior probability of being in that language, and a posterior probability based on the characters of which it is composed, this latter probability is the one we are interested in. The prior probability can be fixed or may depend on external factors; most methods covered by the literature implicitly suppose input text has the same chance of being in every language. Most often the programs identifying a language only return the one having the highest posterior probability.

Bayes' theorem states that for any given language[1]

[1]For the sake of simplification I've used a relaxed (and slightly ambiguous) notation where $P(L)$ indicates the prior probability of any random text being in the language L, where the traditional notation would require L to mean the *event* of a text being in the language (and not the language itself). Similarly, $P(F_1, F_2, ..., F_n)$ will be defined as the probability of the features of a text being equal to $F_1, F_2, ..., F_n$, etc.

$$P(L|C) = \frac{P(C|L)P(L)}{P(C)}$$

where $P(L)$ is the prior probability of a text being in the chosen language and $P(C)$ is the probability of the characters of any random text being exactly as they are in the input string; this latter value is constant for any fixed input. We are trying to determine $P(L|C)$, the probability of the language being L given the characters C.

It thus becomes clear that if we knew the probability $P(C|L)$ of a random text in a given language being exactly equal to the input text, we could easily derive the probability $P(L|C)$ of our text being in that language (if all languages are presumed to have equal prior probability it will be the only value that determines our choice). This statement may seem a bit nonsensical. After all if, say, I'm testing an email I wrote, what is the chance of a random English text being exactly equal to my email? Not much, but it can not be said to be zero, and this is proved by the existence of the text itself. What is important is that the probability of an Italian text being equal to my English email is even smaller (still not zero if we take typing errors into account).

Herein lies the key to the language identification problem. Trying to solve it means finding a way to extract the probability distribution of text in a given language. This information is usually extracted from an amount of training data.

The single characters obviously have a high interdependence. In a particular way, what character appears at a given position depends very highly on the characters nearby, and the way this dependence work is what makes languages differ. Those dependencies is what we are interested in eliciting, and those dependencies are what we call features.

Features can be anything from n-grams frequencies to average word length, from vowel/consonant ratio to the number of two letter words. There is some ambiguity as to the meaning of the word "feature", as often it is used to mean those characteristics whose frequency are actually being used as features. In the literature usually an n-gram is called a feature, whereas the actual feature used is the n-gram frequency. From now on I will follow this convention, and will call n-grams features.

The most useful and easy to use features are n-grams and words. A useful property they have is that any text can be easily subdivided into a finite number of such features. Suppose now that we know the frequency of every possible feature in every language. From Bayes' theorem we can conclude that, a text having n features $F_1, F_2, ..., F_n$, has a probability of being in a certain language L, equal to

$$P(L|F_1, F_2, ..., F_n) = \frac{P(F_1, F_2, ..., F_n|L)P(L)}{P(F_1, F_2, ..., F_n)} \tag{4.1}$$

Unfortunately the conditional probability $P(F_1, F_2, ..., F_n|L)$ depends on the interdependence between the various features, which is much harder to elicit from the training data and to store that the single feature distributions. The problem can be simplified by assuming all features are conditionally independent (even if they are not). This is exactly the (naive) assumption made by naive Bayes classifiers[2], which have been proved to work very well in practice.

It is clear that how not every possible feature of every language can be stored, but this does not have to invalidate our model. We can keep only the most meaningful features and group the others together as they were a single feature, with it's own frequency. Actually, any kind of grouping can be done (and often is done) without altering the validity of the considerations made. An example of this is ignoring the difference between lower and uppercase characters, or removing all punctuation.

It is a lot harder to also be able to identify the case in which a text is in none of the trained languages. To do that we should be able so somehow approximate the probability of $P(F_1, F_2, ..., F_n)$. This is not easy once we leave the assumption that any text is in one of the languages for which we have training data.

4.2 Feature selection

In order to improve accuracy, increasing the number of features stored is the first thing that springs into mind. With one caveat, if the training data is too specific, storing a too high number of features might *worsen* the performance: it has been observed [3] that the most frequent n-grams characterize the language while other n-grams tend to be more specific to the topic (and have sometimes been used to distinguish documents by subject). However this can not be true if the training corpus is chosen appropriately, that is, sufficiently generic, or specifically chosen for the given context. Figure 4.1 shows how even using 4000 trigrams, more then 10 times as much as those used by Cavnar and Trenkle, accuracy improvements can be noticed. This happens also when testing news articles, suggesting the training data used is sufficiently generic.

Given the fact that not an indefinite number of features can be memorized, a selection must somehow be made. It follows from common sense that

[2] http://en.wikipedia.org/wiki/Naive_Bayes_classifier

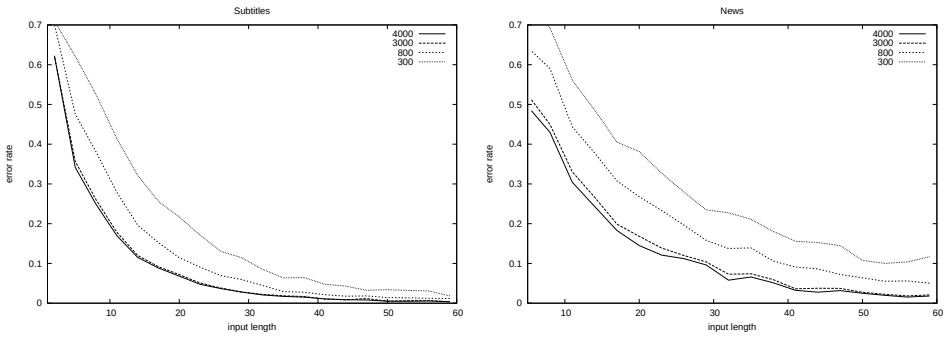

Figure 4.1: Comparison of the accuracy obtained when storing a different number of trigrams. The use of subtitles and news articles as input text were both tested.

the more a feature is specific to a language the more useful it will be. On the other hand, if a feature is very unlikely to occur it will not be of much use no matter how specific to a language it is. Thus we also want the chosen features to be widely used, even if in more than one language, so that it has a high change to appear in the input text.

In the implementations used for this thesis the selection is based only on the frequency of the features in the various languages. But a more sophisticated selection could weight this factor against the frequencies in the languages most similar.

Features can be arbitrarily combined. This throws away some information but saves space and allows for more features to be stored. It is best to combine features that are likely to have a similar influence on the language selection. In the comparisons made here all characters were converted to lowercase, effectively combining trigrams like 'Bob' and 'bob'. Furthermore all non alphabetic characters were stripped (with the exception of the apostrophe) which means combining features such as 'n,_a' with 'n_a' (where '_' is used to represent spaces).

When texts are short the start and end of the strings become more influential. Thus in this thesis those places and everywhere a paragraph ends and the next one starts, have been replaced by the two characters '||'.

Thus for example the string:

> Hi Bob.
> Hey!

generates the following trigrams:

```
||h, |hi, hi_, i_b, bob, ob|, b||, ||h, |he, hey, ey|, y||
```

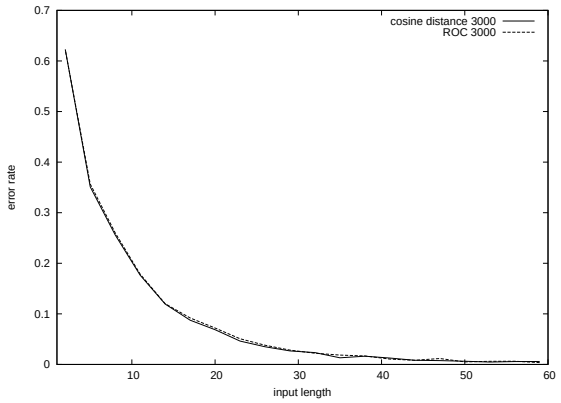

Figure 4.2: Similar performance of cosine distance and ROC.

Once the features have been stored a way needs to be devised to decide, given a text in input, which of these feature sets is most similar (or most "nearby").

All algorithms analyzed make use of some kind of distance measure, but most do not use the statistical model described in the previous chapter directly.

Prager [14] for example has proposed the use of cosine distance, which was here compared to ROC (Figure 4.2), the implementation developed for the test did not use the *inverse document frequency* of the original algorithm, and instead of using the frequencies directly their square root was used, as this proved to give far better results.

An optimal distance measure would be the probability of the input text being in a given language, knowing nothing more that the feature distribution of the various languages. Next section describes how to calculate this probability.

4.3 Building a Naive Bayes Classifier

A naive Bayes classifier has been built as such a classifier uses the exposed formalization directly, and it thus becomes more easy to draw further statistically sound conclusions. More specifically this method provides a confidence level that can be used as an estimate of the accuracy of the result. Next section shows how this proves useful.

The naive assumption made is that all trigrams are statistically independent. No other algorithm copes directly with the interdependence between trigram, so this assumption should not be in any way penalizing.

Under this assumption

$$P(L|F_1, F_2, ..., F_n) = \frac{1}{K} P(L) \prod_{i=1}^{n} P(F_i|L)$$

where K is equal to $P(F_1, F_2, ..., F_n)$, which is constant once the features are fixed, and can thus be ignored leaving the classification task unaffected. $P(L)$ has also been supposed constant (next sections shows how to use different prior probabilities).

As all possible trigrams could not be realistically stored, a special feature "others" (O), has been used to denote any of the least frequent trigrams. Thus first the 3000 most frequent trigrams for each language were selected. Then their frequency was stored for every language, while the cumulative frequency of the remaining characters was stored as the frequency of O.

Sometimes a trigram would have no occurrences in a language corpus. This obviously is because the training corpus has to be finite. We can not suppose that a feature has frequency zero, as that clearly would break the algorithm: $P(F_i|L)$ would be zero for some i, voiding the influence of all other features. The workaround is to use a default value for the trigrams never encountered. It was experimentally found that using any value between 0.01 and 5 as the number of occurrences in the whole language corpus worked well. The value of 1 was chosen[3].

The results obtained were not in the $[0, 1]$ range as the constant $P(L)/K$ has been ignored. This can be easily overcome by normalizing the resulting confidence levels.

However, because of the naive assumption, the probability values obtained this way were not realistic. Any trigram influences those nearby (as they have two characters in common), in a particular way for long strings the confidence was overestimated. It was experimentally found that very realistic results for short strings could be obtained by using the square root of the confidence levels and then normalizing them again. Some more empirical tests showed that even better result are given using $con^{1/log(1+size)}$, where *con* is the initial confidence level, and *size* is the number of characters in the given text. It is important to note that, for long strings, the default value used for not encountered trigrams, while not influencing the classification, *does* have a big influence on the resulting confidence levels.

The performance of naive Bayes proved to be similar (Figure 4.3) to that of other algorithms that have so far been considered. Next section will explore

[3] Ahmed et al. [1] tried to solve the same problem normalizing the values in the [1, 2] range. This however very much alters the behaviour of the classifier, even when no values equal to zero are encountered. This is easily proved: for example, 0.01×0.01 is equal to 0.1×0.001, whereas 1.01×1.01 is not equal to 1.1×1.001.

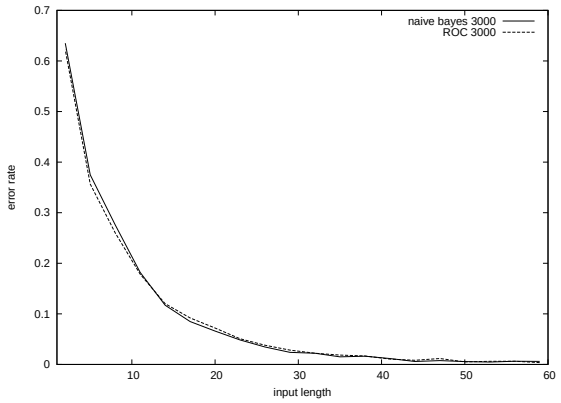

Figure 4.3: Naive Bayes compared to ROC.

how to improve it's accuracy in the case prior knowledge about the language probabilities is available.

4.4 Using prior probabilities

Most language identification methods provide some kind of confidence values (related to the distance measure), unfortunately those values have no real statistical meaning; the only useful property they all have is that, *for any fixed text*, languages having a higher confidence are more likely to be the one searched for (exactly the supposition on which the classification is based). Usually the values are not in the [0..1] range, but even after normalizing the values to make their sum equal to 1, the value obtained is not of much use. The biggest flaw is that there is absolutely no guarantee that the same confidence means the same thing for different texts. That being said, having an approximate probability is still better than having none at all. In this thesis it has been attempted to get an as accurate approximation as possible.

As shown in the previous section, if $C(L|F)$ is the probability of the language being L given the features F, under the assumption of having no prior knowledge, this value will be directly proportional to $P(F|L)$. At the same time the following proportionality follows from (4.1):

$$P(L|F) \propto P(F|L)P(L)$$

where some prior knowledge *is* assumed (i.e. $P(L)$ may change when L changes). Thus we can obtain $P(L|F)$ by calculating the combined confidence $C(L|F) = C(F|L)P(L)$ and normalizing the results.

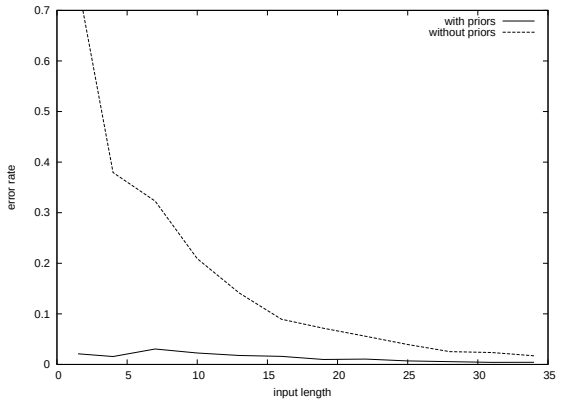

Figure 4.4: Using prior knowledge about the language distribution can greatly improve the accuracy of the naive Bayes algorithm.

Figure 4.4 shows an example of the accuracy improvement that can be provided by the use of prior knowledge. In this example the prior probability of the most likely language was set to 80%, meaning 20% of the times a wrong language was intentionally suggested. Similarly, the prior probability of the next four languages was equal and cumulatively spanning 18%. All other languages covered the remaining 2%.

Chapter 5

Conclusions

5.1 Results

It has been proved how good accuracy identifying short text can be archived using the statistical properties of trigrams, combined with some prior information about the language probabilities. To train the algorithm a multilingual corpus is needed, thus a way to build such a corpus was provided.

All source code written for this thesis can be found on-line[1]. The repository includes: the corpus built and the scripts that were used for it's creation; implementations of the various algorithms tested and of the naive Bayes classifier; the scripts to make the comparisons and draw the graphs shown in this thesis.

5.2 Directions for further work

There are two places were the naive Bayes classifier could be improved. The feature selection could be made more sophisticated, by trying to select features that are more likely to discriminate between similar languages, and by including features other than trigrams. Additionally, while the confidence levels proved to give good results in practice, improvements could be made either by a theoretical study of the effect of the naive assumption, or by rigorous experimental tests showing exactly *how* accurate the confidence levels are and perhaps improving their approximation.

It might also be tried to apply to the problem some of the techniques used in language identification of spoken words, such as Hidden Markov Models or Support Vector Machines.

[1] http://github.com/fela/rlid

The subtitle corpus sometimes contained text in a wrong language. Those that were discovered were removed by hand, but a way to automatically detect and remove the incriminating subtitles (or parts of them) would further improve the robustness of the resulting corpus.

5.2.1 Language identification using a spellchecker

Another way to identify the language of a given text would be to run the text through a spell checker in the different target languages and to use the number of errors in each language (or the Hamming distance from the corrected text) as a distance estimate. This is obviously very inefficient but it might provide a very high accuracy. It would be interesting to see what degree of accuracy is possible.

This method could be also used as a fallback after other algorithms fail to identify a given text with the required confidence, or only in the case the input text is very short. Not all languages would need to be tested, only those between which faster algorithms are in doubt.

When detecting the languages of multilingual documents this method might be used to reliably determine the exact point where the language switches.

5.2.2 Multilingual documents

A use case that has never been addressed in the literature is the case of a multilingual document for which it is required to know which parts are written in which language. The easiest way to solve this problem is by splitting the text into sentences or paragraph and then applying one of the existing algorithms to each piece thus obtained. As shown, however, if the pieces are too short it becomes likely to get a number of inaccurate results.

To solve this the locality principle can be used: nearby words and sentences are likely to be written in the same language. A proof of concept implementation[2] showing an application of this idea has previously been developed (not as part of this thesis). Further improvements in accuracy and speed could be made, using some of the results that have been described here.

[2] http://github.com/fela/multi-lid

Bibliography

[1] B. Ahmed, S.H. Cha, and C. Tappert. Language identification from text using n-gram based cumulative frequency addition. In *Proceedings of CSIS Research Day*, 2004.

[2] G. Botha, V. Zimu, and E. Barnard. Text-based language identification for the South African languages. In *Proceedings of the 17th Annual Symposium of Pattern Recognition Association of South Africa, Parys, South Africa*, 2006.

[3] W.B. Cavnar and J.M. Trenkle. N-Gram-Based Text Categorization. In *Proceedings of the 3rd Symposium on Document Analysis and Information Retrieval in Las Vegas*, pages 161–169, 1994.

[4] HP Combrinck and EC Botha. Text-based automatic language identification. In *Proceedings of the 6th Annual Symposium of the Pattern Recognition Association of South Africa, Gauteng, South-Africa*, November 1995.

[5] M. Damashek. Gauging similarity with n-grams: Language-independent categorization of text. *Science*, 267(5199):843, 1995.

[6] Ted Dunning. Statistical identification of language. Technical report, 1994.

[7] D. Elworthy. Language identification with confidence limits. In *Proceedings of the 6th Annual Workshop on Very Large Corpora*, 1998.

[8] J. Hakkinen, J. Tian, N.M. Phones, and F. Tampere. N-gram and decision tree based language identification for written words. In *Proceedings of the IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 335–338, 2001.

[9] N. Ljubešić, N. Mikelić, and D. Boras. Language identification: how to distinguish similar languages? In *Proceedings of the 29th International Conference on Information Technology Interfaces*, 2007.

[10] S. MacNamara, P. Cunningham, and J. Byrne. Neural networks for language identification: a comparative study. *Information Processing and Management*, 34(4):395–403, 1998.

[11] D.C. Montgomery, E.A. Peck, and G.G. Vining. *Introduction to linear regression analysis*. Wiley New York, 1982.

[12] K.N. Murthy and G.B. Kumar. Language identification from small text samples. *Journal of Quantitative Linguistics*, 13(1):57–80, 2006.

[13] T. Pham and D. Tran. VQ-based written language identification. In *Proceedings of the 7th International Symposium on Signal Processing and Its Applications*, 2003.

[14] J.M. Prager. Linguini: Language identification for multilingual documents. *Journal of Management Information Systems*, 16(3):71–101, 1999.

[15] O. Streiter and M. Stuflesser. XNLRDF, the open source framework for multilingual computing. In *Proceedings of the Conference of Lesser Used Languages & Computer Linguistics*, 2005.