STATISTICAL LANGUAGE IDENTIFICATION OF SHORT TEXTS

Fela Winkelmolen and Viviana Mascardi
Department of Computer Science (DISI), University of Genova, Genova, Italy

Keywords: Language identification, Short text, Naive Bayes classifier.

Abstract: Although correctly identifying the language of short texts should prove useful in a large number of applications, few satisfactory attemps are reported in the literature.
In this paper we describe a Naive Bayes Classifier that performs well on very short texts, as well as the corpus that we created from movie subtitles for training it.
Both the corpus and the algorithm are available under the GNU Lesser General Public License.

1 INTRODUCTION

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 may support 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 texts** the performance degrades quickly. The literature devotes little attention to the accuracy of state-of-the-art algorithms when very short texts are tested. On documents that are 25 characters long, one of the most widely used algorithms (Cavnar and Trenkle, 1994) identifies the language correctly only about 80% of the times (Figure 1). 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 have already been sent.

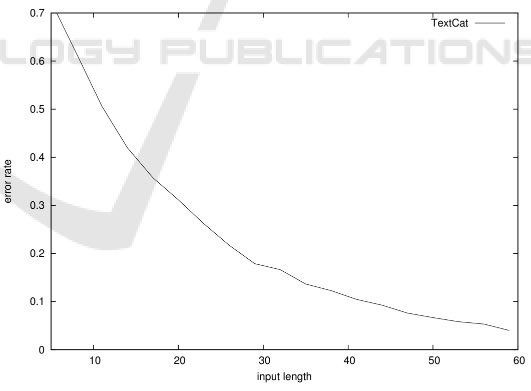

Figure 1: Identification accuracy.

An even more striking example is provided by SMS texts, which are usually very short. Most phones use T9 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.

Considering these situations, where the language of short text strings needs to be identified with a high accuracy, in this paper we discuss a Naive Bayes Classifier we implemented and tested, obtaining promising results.

The paper is organized as follows: Section 2 surveys the related work, Section 3 describes the corpus we built by exploiting movie subtitles, Section 4 formalizes the problem we face and introduces the features we use, Section 5 discusses our Naive Bayes Classifier, and Section 6 concludes and highlights some future research directions.

2 RELATED WORK

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.

Most free programs providing language identification of text are based on the work of Cavnar and Trenkle (Cavnar and Trenkle, 1994). These authors described 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 paper 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 (Prager, 1999) 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 distance used was the cosine distance of the vectors normalized in feature space.

Ahmed et al. (Ahmed et al., 2004) 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 (Hakkinen et al., 2001) and vector quantization (Pham and Tran, 2003) 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. (MacNamara et al., 1998) 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 (Elworthy, 1998) proposes to avoid processing the whole document, managing 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.

3 LANGUAGE CORPORA

All algorithms using statistical comparison of language features need some amount of text in the languages of interest to be trained. The bigger and the more representative of the language the data is, the better the algorithm will perform.

Big existing corpora are freely available in some languages (http://corpus.byu.edu/, http://www.clres.com/corp.html), 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. *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 Wikipedia exists. *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. *Movie 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). In the sequel we describe how subtitles were used to build our own multilingual corpus. *BBC Worldservice* contains articles in a wide range of languages (http://www.bbc.co.uk/worldservice/languages/index.shtml) which might be easily scraped. Combining some of the sources above, in a clever and statistically sound way, may provide even better results.

Since none of the corpora described above met our needs, we built our own corpus by using movie subtitles. To obtain subtitles opensubtitles.org's XMLRPC public API was used (http://trac.opensubtitles.org/projects/opensubtitles/wiki/XMLRPC).

We downloaded subtitles from 22 languages. 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.

To evaluate the impact of the use of different corpora we compared the trigrams provided by TextCat (http://odur.let.rug.nl/∼vannoord/TextCat/), 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 2. Lastly, as seen in the same figure, when English news articles were tested, the results were inverted and our 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 our 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.

4 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

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

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.

In order to solve the problem, we need to model and select features we will exploit.

Features Modeling. 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 we 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

$$P(L|F_1, F_2, ..., F_n) = \frac{P(F_1, F_2, ..., F_n|L)P(L)}{P(F_1, F_2, ..., F_n)} \quad (1)$$

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, which have been proved to work very well in practice.

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

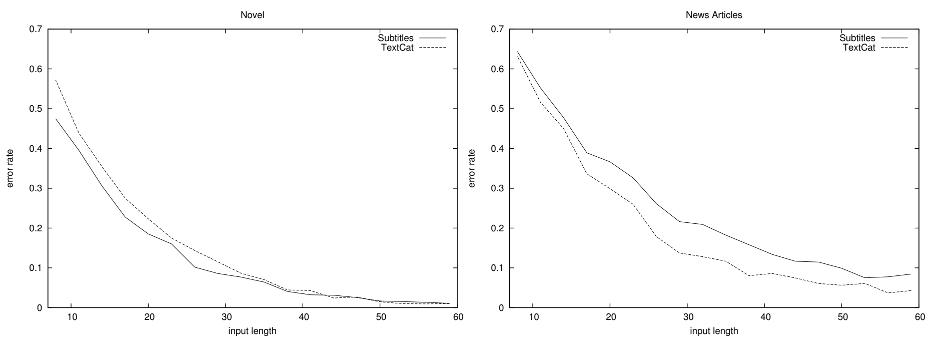

Figure 2: A comparison of TextCat versus the same algorithm trained with the subtitle corpus.

number of features might *worsen* the performance: it has been observed (Cavnar and Trenkle, 1994) 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.

In the implementations discussed in this paper 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 our experiments those places and everywhere a paragraph ends and the next one starts, have been replaced by the two characters '——'.

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 (Prager, 1999) for example has proposed the use of cosine distance: 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.

5 OUR 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.

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.

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[1].

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 3) to that of other algorithms that have so far been considered.

Using prior probabilities can improve these results.

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 our work, we attempted to get an as accurate approximation as possible.

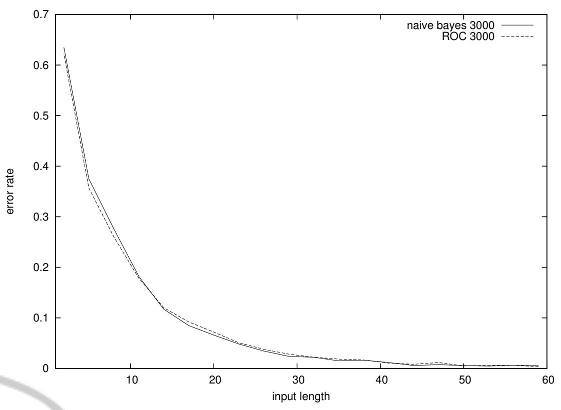

Figure 3: Naive Bayes compared to ROC.

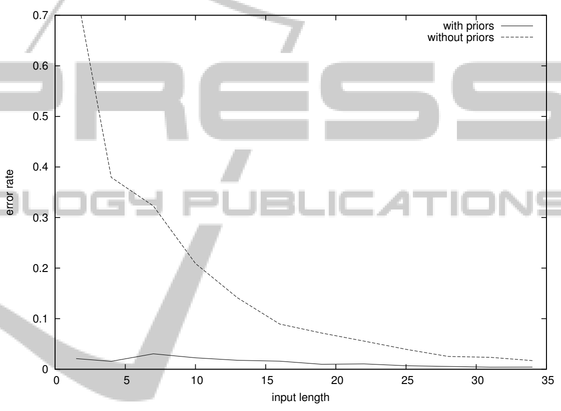

Figure 4: Experiments where prior knowledge about the language distribution is exploited.

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 (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 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%.

[1]Ahmed et al. (Ahmed et al., 2004) 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.

6 CONCLUSIONS AND FUTURE WORK

In this paper 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 suitable multilingual corpus is needed, thus we built our own corpus from movie subtitles. All source code written for this paper can be found online[2].

The naive Bayes classifier could be improved in two ways. First, 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. Second, 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.

Another research direction that we plan to pursue in the close future involves the identification of the language of a given text obtained by running the text through a spell checker in the different target languages and using 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.

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 showing an application of this idea has previously been developed (Winkelmolen, 2010). Further improvements in accuracy and speed could be made, using some of the results that have been described here.

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

REFERENCES

Ahmed, B., Cha, S., and Tappert, C. (2004). Language identification from text using n-gram based cumulative frequency addition. In *Proc. of CSIS Research Day*.

Cavnar, W. and Trenkle, J. (1994). N-Gram-Based Text Categorization. In *Proc. of SDAIR-94*, pages 161–169.

Elworthy, D. (1998). Language identification with confidence limits. In *Proc. of WVLC-98*.

Hakkinen, J., Tian, J., Phones, N., and Tampere, F. (2001). N-gram and decision tree based language identification for written words. In *Proc. of IEEE ASRU-01*, pages 335–338.

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

Pham, T. and Tran, D. (2003). VQ-based written language identification. In *Proc. of ISSPA-03*.

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

Winkelmolen, F. (2010). Statistical language identification of short texts. Bachelor's Thesis, University of Genova, http://www.disi.unige.it/person/MascardiV/Download/Winkelmolen-Fela.pdf.