Statistical Language Identification for NLP-Supported Full Text Retrieval

Master's Thesis – Status Report 5

Michael Piotrowski, October 30, 1997

1 Introduction

IRF/1 is an experimental full-text retrieval system. Its main goal is to evaluate whether the use of NLP techniques, especially morphology, can improve the retrieval performance and/or user-friendliness of a text retrieval system. Another important aspect is to test the suitability of the NLP components for use in the text retrieval environment, especially with regard to their ability to process large amounts of unrestricted text. Because it is based on linguistic principles which are applicable to almost all languages, it will be possible to implement support for multiple languages in a uniform way.

Linguistic methods, however, are inherently language-dependent. This makes it necessary to know in which language a text is writen. For example, analyzing German text with an English grammar will probably lead to undesirable effects. For small, homogenous collections the text language can be specified manually, but this is not possible for large text collections like the WWW, which contains documents written in many different languages, most of them without a formal indication of the language.

Furthermore, automatic identification of language is not only necessary to ensure that the correct grammar is applied, but also allows to annotate the indexed documents so that it is possible to restrict the search to documents in specific languages, as it is offered by Digital Equipment Corporation's *AltaVista* search engine, for example.

While most of the work is done for spoken language, automatic language identification for written texts also has many other possible uses, which might explain why there's research going on at Sun Labs [5] and Xerox [7]. There are at least two commercial language identification tools by large companies [6, 7].

Consequently, a module to automatically identify the language of a text was developed for IRF/1.

2 Design and Implementation[1]

When presented with the following 20 character text samples:

```
den anforderungen ih
r being a successful
nous republions le d
messaggi chimici che
```

most people would possibly be able to identify them as German, English, French, and Italian, respectively, even if they didn't understand these languages.

This ability can be modeled on the computer by low-order character-level Markov chains (for an in-depth treatment of statistical methods in linguistics see for example [2]). Of course, this doesn't capture the structure of a language very well, but this is obviously not necessary for language identification; it's more the "look" that's important here.

A Markov chain is a random (stochastic) process in which the probability of the next state depends only on the current state. More formally, a Markov chain defines a stochastic variable whose values are strings from an alphabet Ω, and where the probability of a particular string S is

$$P(S) = P(s_1 \ldots s_n) = P(s_1) \prod_{i=2}^{n} P(s_i|s_{i-1}) \tag{1}$$

The conditional probabilities $P(s_i|s_{i-1})$ are called *transition probabilities.*

A simple example of a Markov chain might be used to model the British weather according to [4]. We assume that the weather is observed at intervals of half an hour and that it can be in one of three states (making up the alphabet Ω): s_0, it's neither raining nor foggy (which is extremely unlikely), s_1, it's foggy, and s_2, it's raining. The weather at a point of time t is characterized by a single one of these three states. We can now build a transition matrix:

$$\boldsymbol{W} = (w_{ij}) = \begin{pmatrix} 0.1 & 0.45 & 0.45 \\ 0.25 & 0.5 & 0.25 \\ 0.25 & 0.25 & 0.5 \end{pmatrix} \tag{2}$$

Using this model, we can now ask and answer questions about British weather patterns over time. For example, what is the probability (according to the model) that, if it's currently neither raining nor foggy, at the next two checks it's raining, and then foggy?

$$\begin{aligned} P(s_0, s_2, s_2, s_1|\boldsymbol{W}) =& P(s_0)P(s_2|s_0)P(s_2|s_2)P(s_1|s_2) \\ =& \pi_0 w_{02} w_{22} w_{21} \\ =& 1.0 \cdot 0.45 \cdot 0.5 \cdot 0.25 \\ =& 0.05625 \end{aligned} \tag{3}$$

[1] This section closely follows [1], which you should consult for details. For a very brief overview of two identification methods see [3].

where π_0 denotes the initial state probability for s_0, which is 1 in this case because it was actually observed.

A random process where the probability of the next state depends only on the last k states can also be described by Markov chains by using the last k states as the current state. These Markov chains are called Markov chains of order k or n-gram models (where $n = k+1$; an n-gram is a sequence of characters of length n).

The transition probabilites look like this:

$$P(s_{i+1} \dots s_{i+k} | s_i \dots s_{i+k-1}) = P(s_{i+k} | s_i \dots s_{i+k-1}) \tag{4}$$

If we assume that the distributions of strings in the language satisfy the distribution for a k-order Markov chain, we can now use a k-order Markov chain, where k is relatively small (1 or 2, 2 in our implementation), to produce a simple model of language. In this case, the alphabet Ω is the set of characters in the language (e. g. a subset of ISO 8859-1).

We do this by building a transition matrix similar to the one for the weather example above, only that it has more states since the alphabet is larger, and that it is $k+1$-dimensional. However, it can be collapsed into a 2-dimensional table (see equation 4). Table 1 shows a part of the transition matrix this implementation uses for German.

Table 1: Trigram transition matrix

	l	m	n	o	p	q	r	s
de	.009	.089	.287	.032	.032	.032	.352	.071
di	.032	.032	.014	.032	.032	.032	.040	.008

If we have a table like this for every language we want to be able to identify, we can calculate the probability for a string S to be generated by a particular Markov chain A like this:

$$P(S|A) = \prod_{s_1 \dots s_{k+1} \in S} T(s_1 \dots s_k, S) P(s_{k+1} | s_1 \dots s_k | A) \tag{5}$$

where $T(s_1 \dots s_k, S)$ is the number of times the $k+1$ gram $s_1 \dots s_{k+1}$ occurs in the string S.

To avoid problems of numeric underflow, we compare logarithms of these conditional probabilites. This gives us:

$$\log P(S|A) = \sum_{s_1 \dots s_{k+1} \in S} T(s_1 \dots s_k, S) \log P(s_{k+1} | s_1 \dots s_k | A) \tag{6}$$

By computing this value for each of our language models and selecting the largest, we can pick the language model which is most likely to have generated the observed string.

How do we get the transition probabilities (also known as *model parameters*) for a language? We can learn them from sample texts (this is also referred to as training). By counting the number of occurrences of n-grams in the text and then estimating the probabilities for the n-grams we have observed, i. e. $P(s_{k+1} | s_1 \dots s_k | A)$ (from above).

The most obvious estimation method is the maximum likelihood estimator

$$P(s_{k+1}|s_1 \dots s_k|A) = \frac{T(s_1 \dots s_{k+1}, S_A)}{T(s_1 \dots s_k, S_A)} \tag{7}$$

where S_A is the training string for language A. This is however not completely satisfactory for our application because our training data is relatively limited, so that it will necessarily happen that n-grams which were not in the training data appear in the data to classify. The maximum likelihood estimator gives n-grams which don't appear in the sample a probability of 0, which results in $P(S|A) = 0$. This would result in a completely wrong classification if this n-gram actually appeared in the training data of one language because every string containing this n-gram would then be automatically judged to be from this language.

What we use instead is the Bayesian estimator which minimizes the mean squared error for $P(S|A)$ instead of maximizing the probability. The details are described in [1]; the final form is the expression

$$\hat{p} = \frac{T(s_1 \dots s_{k+1}, S_A) + 1}{T(s_1 \dots s_k, S_A) + m} \tag{8}$$

also known as Laplace's sample size correction. This method is used in the implementation described in this report.

3 Performance

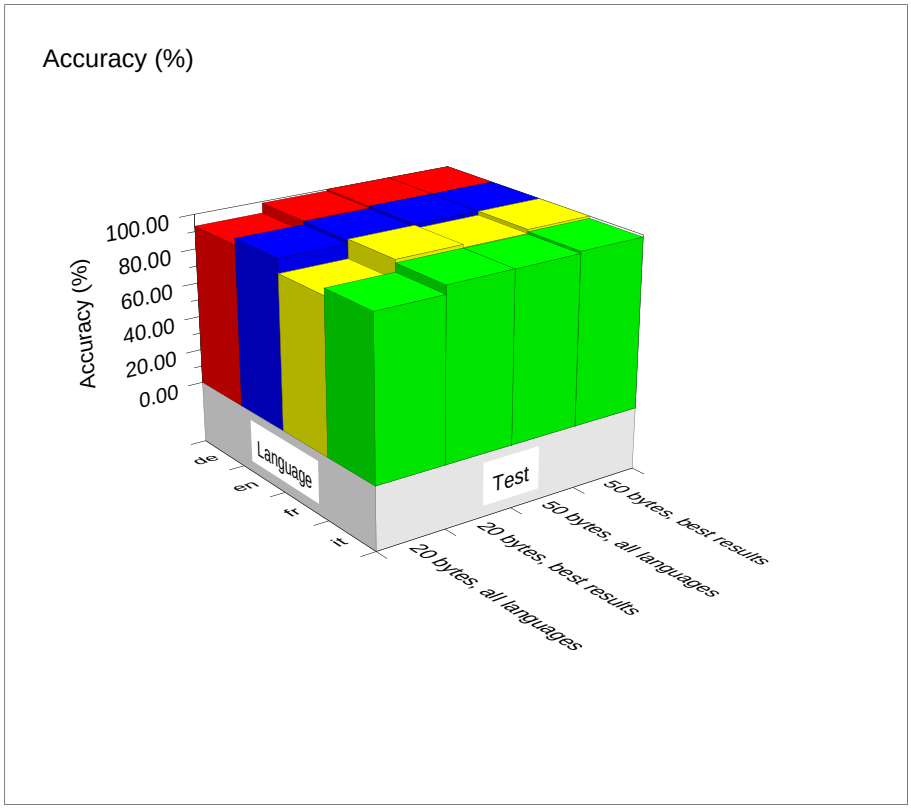
In [1] the following conditions are identified as relevant to the performance of a language classifier:

1. Selection of test strings
2. Amount of training data
3. Length of strings to be identified
4. Number of languages to be identified
5. Correlation between domain and language (quality of training data for the intended identification task)

We can confirm the relevance of these points. The quality and size of the training data becomes more important with more languages; this problem didn't occur in [1] since selections from a parallel English-Spanish corpus for training and testing were used and English and Spanish were the only languages.

In [1] both the training and testing data were carefully selected and manually checked; there wasn't enough time to do the same for this implementation, so the data was selected in a rather ad-hoc manner from corpora available at CLUE, and from on-line sources like newspapers and press releases. Nevertheless, this implementation compares quite favorably to the original one as well as to the commercial ones. Up to now, training has been performed for German, English, French, and Italian with 50K of training data for each language.

Figure 1: Identification Accuracy



For 20 byte strings and optimal language pairs, an average accuracy of 98.73 % is achieved, which rises to 99.69 % for 50 byte strings. For strings longer than 60 bytes no errors could be observed for any of the languages, thus achieving 100 % accuracy (see figure 1). Classification speed is approximately 340 20 byte strings per seconds when deciding between two languages, but the current implementation wasn't optimized for speed.

4 Possible Enhancements

To avoid false classifications due to insufficient training or test data a threshold for accepting a classification could be introduced. No work has been done in this direction yet since language identification is only one component of IRF/1.

References

[1] Dunning, Ted (1994). *Statistical Identification of Language.* Technical report CRL MCCS-94-273. Computing Research Lab, New Mexico State University.

[2] Krenn, Brigitte and Christer Samuelsson (1997). *The Linguists Guide to Statistics.* Available online at `http://coli.uni-sb.de/~christer/stat_cl.ps`.

[3] Grefenstette, Gregory (1995). "Comparing Two Language Identification Schemes". In: *Proceedings of the 3rd International Conference on the*

the Statistical Analysis of Textual Data (JADT'95). Available online at http://www.xerox.fr/publis/mltt/jadt/jadt.html.

[4] Goscinny, René and Albert Uderzo (1966). *Astérix chez les Bretons*. Paris, Dargaud.

[5] Sun Labs International Linguistic Application Group Language Identification Demo.
http://www.sunlabs.com/research/ila/demo/.

[6] Novell ATD Collexion Language Identifier.
http://www.novell.com/atd/colx/alt/lanf/lanrec.html.

[7] RXRC Language Identifier.
http://www.xerox.fr/research/mltt/Tools/guesser.html
(Commercial version: Inxight Software, http://www.inxight.com).