University of Tübingen

International Studies in Computational Linguistics

Bachelor of Arts Thesis

Course: Computational Morphology and Phonology

Lecturer: Dale Gerdemann

Evaluation of Language Identification Methods

Simon Kranig

Bogentorstr. 4

72070 Tübingen

mismo@web.de

Abstract

Language identification plays a major role in several Natural Language Processing applications. It is mostly used as an important preprocessing step. Various approaches have been made to master the task. Recognition rates have tremendously increased. Today identification rates of up to 99 % can be reached even for small input. The following paper will give an overview about the approaches, explain how they work, and comment on their accuracy. In the remainder of the paper, three freely available language identification programs are tested and evaluated.

1 Introduction

This thesis paper is concerned with different language identification methods. As international communication, business, as well as private, has grown rapidly over the last years, systems are needed which can correctly identify languages of documents, such as e-mails, letter, webpages etc. The task of language identification serves in various areas of NLP. One active company in the field (Lextek) sells a professional language identification program and provides a few possible applications for it:

- E-mail routing and filtering engines
- Text mining applications
- Identification of the language or encoding of WWW pages
- Information retrieval systems
- Content based and language specific web crawlers and search engines

. (Resnik 99) used language identification for the aquisition of a bilinguagal corpus from the internet. They developed a system called *Strand*, which is able to look for parallel translated pages and helps to construct a bilingual corpus. (Langer 01) developed a web crawler to get representative numbers on the distribution of the language on webpages. He parsed around three million webpages to find out their primary language. The main language was English with 65% , followed by Japanese (6%) and German (5%). At the fast pace the economy is growing in Asia, these figures will probably have changed over the last few years. With the increasing number of different languages the internet, the the importance of language identification also grows. In his paper (Grefenstette 95) points out that language identification is needed for further morphologial processing of data. The language of a document needs to be known before techniques such as stemming or parsing can be applied. Also if a lexicon is used for spell checking, the program needs to know which language specific rules to apply.

As one can see, there are various tasks which need language identification programs. After the presentation of possible applications ,I will now talk about the different approaches made within the last decade. During this time the task for written documents has been mostly solved and identification rates are up to 99%.

In Section 2 the small words and unique letter combination approaches by (Grefenstette 95) will be introduced. Section 3 will provide a presentation of an statistical approach introduced by (Dunning 94). In Section 4 the N-gram approach by (Cavnar et al. 94) is explained. Section 5 will present a compression based approach using PPM (partial pattern matching) by (Teahan et al. 01). The N-gram approach as well as the statistical one and PPM compression based are closely related. Therefore I will refer to previous sections if a concept has already been explained before. Section 6 will provide an overview about an extension to the statistical approach which can handle East Asian languages by (Kikui 95). As practical part of this paper in Section 7 I will introduce two freely available Perl modules (LING IDNT), (LING IDFY) and the Text_Cat program by (van Noord) for language identification. Lingua::Ident is an implementation of the approach by (Dunning 94), Lingua::Identify combines various methods and Text_Cat is an implementation of the approach by (Cavnar et al. 94)

2 Common Words and Unique Letter Combinations

The main idea of the common words approach is to store highly frequent words for each language in a database. The document to be classified is then compared to all the word lists in question. Via a scoring system the word list with the most occurrences in that document indicates the language of the document. This seems to be a good approach for large documents. For testing (Grefenstette 95) used the ECI CD-ROM (Corpus from European Corpus Initiative), which is a corpus containing newspaper articles from 27

	Identified as ...									
Language	Danish	Dutch	English	French	German	Italian	Norwegian	Portuguese	Spanish	???
Danish	6032	3	1	25	5	1	286	2	1	275
Dutch	2	6914	8	5	9		2	1	17	61
English		2	7808	2	6		39	8		6
French	4	2		5023	3	24	2	7	14	13
German	56	7	5	92	5590	14	5	33	1	125
Italian	35	3	1	8	11	5742	15	18	9	97
Norwegian	8	1	1	2	2	2	13102		1	2
Portuguese	2	8	9	12	14	26	3	8505	69	140
Spanish				3	1	16		23	5150	198

Table 1: Results for common words method (Grefenstette 95)

European languages. The first 100,000 words were extracted and small words up to 5 characters which occurred more than three times were kept for testing. This was done for 10 languages. For classification results see Table 1. The column "*???*" denotes the number of documents, which had an equal probability for all languages. Several drawbacks have to be mentioned however. Especially for short texts, i.e. input of less than ten words, the probability is quite low that one of the common words occurs in the input. This leads to misclassification of the language. Also for languages as Chinese, where tokenization is not easily possible, this approach won't work in the desired way. An advantage of the common words approach is that it is easy to implement and that it requires less computational power than a tri-gram approach as there are fewer calculations to be done. The common words approach is reviewed in (Dunning 94) and discussed in (Grefenstette 95)".

A variant to the common words are unique letter combinations. (Churcher 94) proposed a list of unique letter combinations. In Table 2 on page 4 there are some language specific strings. The problem with this approach is again, if there is not enough input and the letter combinations do not occur. Also these strings are not as unique as they seem, as many words can also occur in different languages. The previously mentioned approaches are predecessors of the N-gram approach by (Cavnar et al. 94) discussed in section 4. The difference to the N-gram approach is that the authors take all N-grams into account and not only special words like common words or language specific strings.

3 Statistical Approach by Ted Dunning

One of the most important papers on statistical language identification is by (Dunning 94). In his technique he uses Markov Models to calculate the probability that a document orig-

Language	String
Dutch	"vnd"
English	"ery"
French	"eux"
Gaelic	"mh"
German	"der"
Italian	"cchi"
Portuguese	"seu"
Serbo-croat	"lj"
Spanish	"ir"

Table 2: Language specific character sequences

inated from a given language model. In the following paragraphs I will explain how his approach works, and later on comment on its efficiency.

For statistical language identification a set of character level language models is prepared from training data, i.e. sample texts in different languages as a first step. This is done by segmenting the strings and entering them into a transition matrix which contains the probabilites of the occurrence of all character sequences.

3.1 Language Identification with Markov Models

In the second step the probability is calculated that a document derives from one of the existing language models, i.e. the probability that a String S occurs being from an alphabet X. In the example below the probability of string S is defined by the probability of the sequence of characters $s_1 \dots s_n$.

$$p(S) = p(s_1 \dots s_n) = p(s_1) \prod_{i=2}^{n} p(s_i \mid s_{i-1}) \tag{1}$$

where we have an intitial state distribution $p(s_i)$ and transition probabilities s_{i-1}. Changing the formula, introducing a fixed context of k states, we get a Markov model with limited history. Now only the last k previous characters are taken into account. The new initial states are now $p(s_1 \dots s_k)$ and the transition probabilities $p(s_{i+k} \mid s_i \dots s_{i+k-1})$.

For language identification we can now create a language model, where the occuring strings show the distribution in that language. The alphabet X is the set of characters in that language. As (Dunning 94) shows a higher order Markov model is needed for the process as it must be capable to grasp enough information about the language.

Now given our observation X, one needs to find out which phenomenon caused it, the phenomenon being the language to be found (the language model). (Dunning 94) provides an example for that. Given two phenomena A and B and an observation X the task is to find out which of them caused the observation. Using Bayes Theorem, we get:

$$p(A, X) = p(A \mid X)p(X) = p(X \mid A)p(A) \tag{2}$$

which can be reduced to

$$p(A \mid X) = p(X \mid A)p(A) \tag{3}$$

as the observation p(X) = 1 occured
As we do not know the prior probability of A, we assume a uniform probability for A and B. Now the probability that a string S has been produced by the language model A can be computed as follows:

$$p(S \mid A) = p(s_1 \dots s_k \mid A) \prod_{i=k+1}^{N} p(s_i \mid s_{i-k} \dots s_{i-k} \mid A) \tag{4}$$

(Dunning 94) reports that it is better to compare logarithms of the conditional probabilities in order to avoid numeric underflow. Rearranging equation 4 gives us the following formula to calculate the probability of string S given the language model A.

$$\log p(S \mid A) = \sum_{w_1 \dots w_{k+1} \epsilon S} T(w_1 \dots w_{k+1}, S) logp(w_{k+1} \mid w_1 \dots w_k \mid A) \tag{5}$$

where $T(w_1 \dots w_{k+1}, S)$ stands for the number of times that the k+1 gram $w_1 \dots w_{k+1}$ occurs in S.
The probability $p(S \mid languagemodel)$ is computed for all models. The highest probability indicates the model that produced the string. The discussion about the estimation of the model parameters is left out here and can be read in (Dunning 94).

3.2 Evaluation of Dunnings approach

For training his algorithm (Dunning 94) used 50 training texts containing 1000, 2000, 5000, 10000 and 50000 bytes. The testing involved 100 texts with lengths 20, 50, 100, 200, and 500 bytes. To prevent the results from being falsified by the topic of the training data, Dunning used a parallel translated corpus of English and Spanish. After extraction of the text from the corpus, the documents were manually checked to filter out any unwanted data like English occurring in the Spanish texts to provide ideal testing conditions. Dunning compared different training and testing sizes. His findings for small training data are that they work better with bi-grams and tri-grams. This means that *k=1* or *k=2*, so that only the previous or the two previous characters are taken into account. He also conducted a test with DNA sequences, stating that it is similar to language identification. Here the task was to extract sequences of yeast from human sequences. Dunning showed that his approach could also be used extract foreign sequences.

A detailed list of his results can be seen in Figure 1. As with all statistical methods, results can improve as more training data is available, which in this case are just sample texts from a language. The input size is also of concern. (Dunning 94) reports an accuracy of 92% with 20 bytes of text and 500K of training data. As the amount of text to be examined grows, he reports 99,9% accuracy for 500 bytes of text to be classified. The advantages of statistical methods are manifold. No linguistic knowledge is needed to perform identification and the input can be character based. This way the approach can also be used for documents written in languages which cannot be easily tokenized and no preprocessing is needed. I will come back to this point in Section 6.

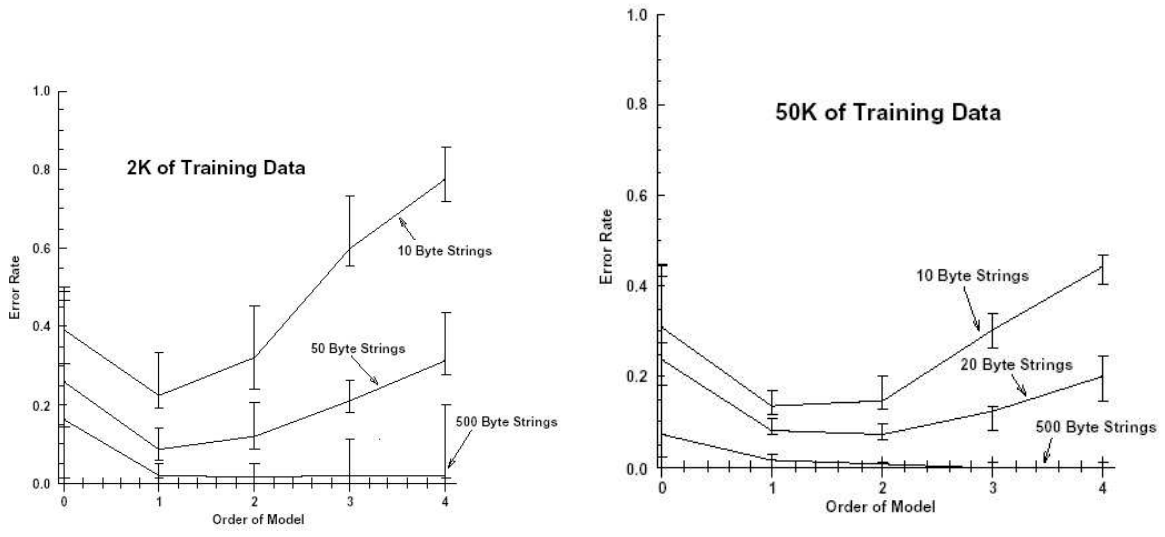

Figure 1: Testing results in (Dunning 94)

4 The N-gram Approach by Cavnar

One of the most successful approaches is the N-gram approach, introduced by (Cavnar et al. 94). They were more concerned with text categorization, but they found out that their method also performed very well on the task of language identification. The main idea of using n-grams for language identification is that every language uses certain n-grams more frequently than others, hence providing a clue about the language. This can be derived from Zipf's Law restated here:

Zipf's Law: The size of the r'th largest occurrence of the event is inversely proportional to it's rank r (WIKI.org)

Figure 2 depicts this relation between frequency and rank.

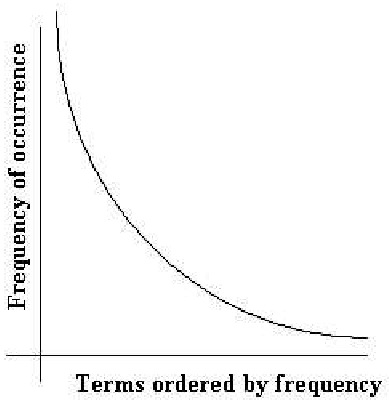

Figure 2: Distribution by Zipf's Law (WIKI.org)

4.1 Introduction of the N-gram Approach

In our case the events are words occurring in a text. N-grams can be seen as substrings of words and respectively words depending on the size of N. For language identification one calculates the N-gram profile of a document to be identified and compares it to language specific N-gram profiles. The language profile which has the smallest distance to our sample text N-gram profile then indicates the language.

(Cavnar et al. 94) use overlapping N-grams, as can be seen in the following example. For N-grams at word initial and final positions, they add an underscore marker which forms part of the actual N-gram. In the example below all possible N-grams with *N=2-4* for the word "GARDEN" are listed. Uni-grams, i.e. $N = 1$ are just the letters of the word themselves.

bi-grams: _G, GA, AR, RD, DE , EN , N_

tri-grams: _GA, GAR, ARD, RDE, DEN, EN_

quad-grams: _GAR, GARD, ARDE, RDEN, DEN_

Their algorithm works the following way. In the first step the sample texts in serveral languages read in ony by one and all punctuation marks are deleted. Each word becomes a token delimited by white space before and after. All tokens are scanned and N-grams with *N=1-5* are produced from these tokens. The N-grams are stored in a hash and for each occurence the counter for the N-gram in question is increased. After that the hash is ordered starting with the most frequent N-grams. Usually uni-grams are at the top of the list, but are discarded as they just reveal information about alphabetical distribution of a language. They do not account too much for the information we are intested in. This procedure is repeated for each language. The N-gram hashes constitute our N-gram profiles for each language.
In order to identify the language of a new document, (Cavnar et al. 94) repeat the above mentioned procedure for that document, yielding an N-gram profile for that document. Now they compare the document profile to the existing language profiles by calculating the distance between the N-gram profiles.

The distance is calculated in the following way. For each N-gram in our test document, there can be a corresponding one in the current language profile we are comparing it to. N-grams having the same rank in both profiles receive a zero distance. If the respective ranks for an N-gram vary, they are assigned the number of ranks between the two with a maximum distance of 3, see example in Figure 3 on page 9. Finally all individual N-gram rank distances are added up. This number is now the distance between the sample document and the current language profile. This step is repeated until the sample document has been compared to all language profiles in question. The smallest distance then indicates the language. After explaining the technique a report on the evaluation of the n-gram approach will be given.

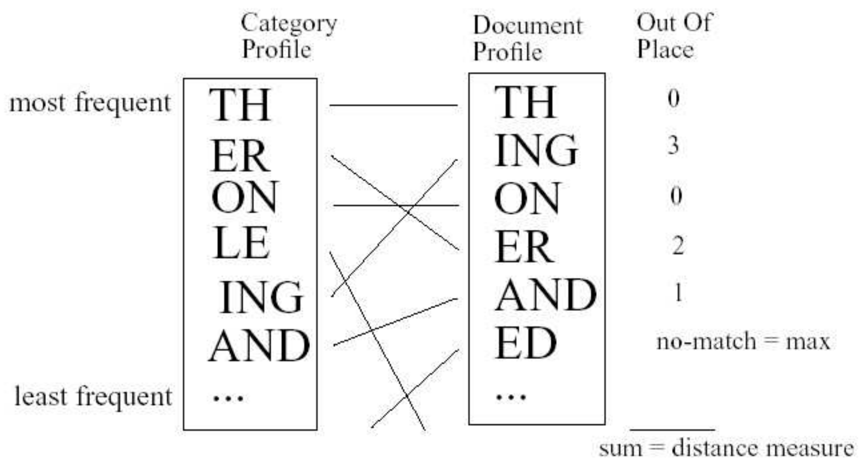

Figure 3: Comparison of n-gram profiles (Cavnar et al. 94)

4.2 Evaluation of the N-gram Approach

As stated in (Cavnar et al. 94), one of the most important advantages of the N-gram approach is that it is robust and does not require linguistic knowledge. Spelling- and similar errors in texts do not prevent correct identification as all words are split up into N-grams, producing only a minimal offset for each mispelling or foreign word occurring in a text. (Cavnar et al. 94) found out that the 300 most frequent N-grams stand in close connection to the language a text is written in. Starting from rank 300 the N-grams become more topic specific. Therefore it should be enough to only use those 300 first N-grams and omit the remaining for language identification. For their evaluation (Cavnar et al. 94) used 3713 language samples from the soc.culture newsgroup hierarchy of the Usenet considering only samples for one language when training a language model, i.e. discarding all noise, i.e. words not being from that language.

They found out, that their method works well for very short input like one or two words, too. As soon as the input contains more than 20 characters, their method reaches a very high identification rate. Please see Table 3 showing the results for their testing suite. One disadvantage of their method has to be mentioned here. The N-gram approach will not be able to work with Asian documents as it relies on correct tokenization. Tokenization for Asian Languages, as stated before, is a hard task. To overcome this problem (Kikui 95) introduced a statistical method which works bytebased and does not need tokenization beforehand.

Article Length	**≤300**	**≤ 300**	**≤ 300**	**≤ 300**	**300**	**300**	**300**	**300**
Profile Length	**100**	**200**	**300**	**400**	**100**	**200**	**300**	**400**
Newsgroup								
australia	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
brazil	70.0	80.0	90.0	90.0	91.3	91.3	95.6	95.7
britain	96.9	100.0	100.0	100.0	100.0	100.0	100.0	100.0
canada	96.9	100.0	100.0	100.0	100.0	99.6*	100.0	100.0
celtic	100.0	100.0	100.0	100.0	99.7	100.0	100.0	100.0
france	90.0	95.0	100.0	*95.0	99.6	99.6	*99.2	99.6
germany	100.0	100.0	100.0	100.0	98.9	100.0	100.0	100.0
italy	88.2	100.0	100.0	100.0	91.6	99.3	99.6	100.0
latinamerica	91.3	95.7	*91.3	95.7	97.5	100.0	*99.5	*99.0
mexico	90.6	100.0	100.0	100.0	94.9	99.1	100.0	*99.5
netherlands	92.3	96.2	96.2	96.2	96.2	99.0	100.0	100.0
poland	93.3	93.3	100.0	100.0	100.0	100.0	100.0	100.0
portugal	100.0	100.0	100.0	100.0	86.8	97.6	100	100
spain	81.5	96.3	100.0	100.0	90.7	98.9	98.9	99.45
Overall	**92.9**	**97.6**	**98.6**	**98.3**	**97.2**	**99.5**	**99.8**	**99.8**

Table 3: N-gram approach results in (Cavnar et al. 94)

After discussing the N-gram approach the Language identification approach using compression based models will be introduced in the section below.

5 Compression based approach with PPM by Teahan

Another interesting approach is the Compression Based approach by (Teahan et al. 01), parts of it already introduced in (Teahan 00). PPM stands for "prediction by partial matching". Sample text from various languages are compressed using the PPM algorithm described below to produce language models. A document to be identified is then also compressed and the number of bits needed to encode this document are compared to the bits in the existing language models. Before explaining the usage of the PPM algorithm I will introduce the notion of entropy and cross-entropy. For a detailed explanation please see (Shannon 48). (Shannon 48) described in the fundamental coding theorem that the lower bound on the average number of bits per symbol needed to encode messages of the language is given by the entropy of the probability distribution, often referred to as the "Entropy of Language"

$$H(Language) = H(p) = \sum_{i=1}^{k} p(s_i) \log_2 p(s_i) \tag{6}$$

where k is the size of the alphabet and s_i the symbols with their probability distribution $p(s)$.

(Teahan 00) states that $-\log_2 p(s_1)$ is the information content of a particular symbol, .i.e the number of bits required to encode that symbol. The individual symbol probabilities are independent and summed up. $H(Langage)$ can be seen as the average number of bits needed to encode a symbol for a given language. To get the highest compression rate, the symbols have to be encoded with the actual probability distribution (Teahan 00). But as we do not have the actual probability distribution for a language, we need to create a language model estimating a probability distribution for that language. How such a model is created, will be explained after the introduction of cross entropy. The *cross entropy* states to what degree a language model ($p_m(x)$) deviates from the actual distribution and is calculated for a given language model p_M by:

$$H(L, p_m) = -\lim_{m \to \infty} \frac{1}{n} \sum_{x_{1n}} p(x_{1n}) \log_2 p_m(x_{1n}) \tag{7}$$

With these formulae at hand we can calculate the cross entropy of a document D by:

$$H(L, p_m, D) = -\frac{1}{n} \log_2 p_M(D), whereD = x_{1n} \tag{8}$$

For the complete derivation of entropy please see (Teahan 00). We can now turn to the actual point, namely using PPM for language identification. A detailed information of the algorithm can be found in (Bell et al. 90).

5.1 PPM - Prediction by Partial Matching

The main idea of PPM is to predict an upcoming character using the set of previous characters with a fixed context. Each upcoming character is assigned a probability. The amount depends on whether the character has appeared before in a sequence with the previous characters. If a certain sequence has not been observed so far, it uses the escape probability and the context size is reduced by one. The escape probability for a sequence receives the number of times the sequence has been observed being followed by a character. In the lower order model again, a check is conducted to see, if a sequence with that character at the end exists. If the character has never been seen in a sequence the context size is set to -1. At this level all characters are predicted equally. This means the character receives a probability, which is calculated by 1 divided by (the total numbers of characters - the already occured characters). To get an optimal compression rate for a model p_M, (Teahan 00) provide the following derivation:

$$
\begin{aligned}
H(L, p_M, D) &= -\frac{1}{n}\log_2 p_M(D) \\
&= -\frac{1}{n}\log_2 \prod_{i=n}^{n} p_M(x_i \mid context_i) \\
&= \frac{1}{n}\sum_{i=n}^{n} -\log_2 p_M(x_i \mid context_i)
\end{aligned}
$$

where our $context_i = x_1, x_2, \ldots, x_{i-1}$

In Table 4 the sample string *abracadabra* is processed using a maximum context of *k=2*. In real life applications a context of *k=5* seemed to give the best results according to (Teahan et al. 01). The model starts out with the maximum context, using 2 letters and showing the letter that follows. We get a count c of 2 for an r following the string *ab*, as this sequence occurs two times in the string *abracadabra*. Therefore the probability will be $\frac{2}{3}$ for an r following the string *ab*. The remaining $\frac{1}{3}$ is the escape probability.

Order k=2			Order k=1			Order k=0			Order k=-1		
Predictions	c	p	Predictions	c	p	Predictions	c	p	Predictions	c	p
ab $\rightarrow$ r	2	$\frac{2}{3}$	a $\rightarrow$ b	2	$\frac{2}{7}$	$\rightarrow$ a	5	$\frac{5}{16}$	$\rightarrow$ A	1	$\frac{1}{\|A\|}$
$\rightarrow$ Esc	1	$\frac{1}{3}$	$\rightarrow$ c	1	$\frac{1}{7}$	$\rightarrow$ b	2	$\frac{2}{16}$			
			$\rightarrow$ d	1	$\frac{1}{7}$	$\rightarrow$ c	1	$\frac{1}{16}$			
ac $\rightarrow$ a	1	$\frac{1}{2}$	$\rightarrow$ Esc	3	$\frac{3}{7}$	$\rightarrow$ d	1	$\frac{1}{16}$			
$\rightarrow$ Esc	1	$\frac{1}{2}$				$\rightarrow$ r	2	$\frac{2}{16}$			
			b $\rightarrow$ r	2	$\frac{2}{3}$	$\rightarrow$ Esc	5	$\frac{5}{16}$			
ad $\rightarrow$ a	1	$\frac{1}{2}$	$\rightarrow$ Esc	1	$\frac{1}{3}$						
$\rightarrow$ Esc	1	$\frac{1}{2}$									
			c $\rightarrow$a	1	$\frac{1}{2}$						
br $\rightarrow$ a	2	$\frac{2}{3}$	$\rightarrow$ Esc	1	$\frac{1}{2}$						
$\rightarrow$ Esc	1	$\frac{1}{3}$									
			d $\rightarrow$a	1	$\frac{1}{2}$						
ca $\rightarrow$ d	1	$\frac{1}{2}$	$\rightarrow$ Esc	1	$\frac{1}{2}$						
$\rightarrow$ Esc	1	$\frac{1}{2}$									
			r $\rightarrow$ a	2	$\frac{1}{3}$						
da $\rightarrow$ b	1	$\frac{1}{2}$	$\rightarrow$ Esc	1	$\frac{1}{3}$						
$\rightarrow$ Esc	1	$\frac{1}{2}$									
ra $\rightarrow$ c	1	$\frac{1}{2}$									
$\rightarrow$ Esc	1	$\frac{1}{2}$									

Table 4: PPMC model processing the string *abracadabra* (Teahan 00)

In Table 5 on page 13 one can see the calculation for the characters c, d and t following the string *ra*. For c the probability will be $\frac{1}{2}$, as it occurs exactly one time after the string ra. So the needed space for encoding will be $-\log_2 \frac{1}{2} = 1bit$. If the character d follows the string *ra*, we jump down to context k = 1, as this sequence has still not occured, storing the escape probability of $\frac{1}{2}$. In the smaller context of k = 1 the sequence $a\rightarrow$ d has

character	probabilities encoded (without exclusions)	probabilities encoded (with exclusions)	codespace occupied
c	$\frac{1}{2}$	$\frac{1}{2}$	$-\log_2 = 1$ bit
d	$\frac{1}{2}, \frac{1}{7}$	$\frac{1}{2}, \frac{1}{6}$	$-\log_2(\frac{1}{2} \cdot \frac{1}{6}) = 3.6$ bits
t	$\frac{1}{2}, \frac{3}{7}, \frac{5}{16}, \frac{1}{\lvert A\rvert}$	$\frac{1}{2}, \frac{3}{6}, \frac{5}{12}, \frac{1}{\lvert A\rvert-5}$	$-\log_2(\frac{1}{2}, \frac{3}{6}, \frac{5}{12}, \frac{1}{251}) = 11.2$ bits

Table 5: Sample Encodings using PPM model in Table 4 (Teahan 00)

occured with a probability of $\frac{1}{6}$. Actually it would be $\frac{1}{7}$, but a mechanism called *exclusion* deletes the occurrence of a *c* following the a. A *c* following the sequence *ab* would have been encoded already at the context k = 2. By using *exclusion* lower order predictions will not be included in the final probability. This leaves us with a needed encoding space of $-\log_2(\frac{1}{2} \cdot \frac{1}{6}) = 3.6 bits$.

5.2 Evaluation of the Compression Based Approach

(Teahan 00) claims equal performance to (Dunning 94)'s approach using compression based models. For testing he used parts of the Bible's *Book of Genesis* in English, French, German, Italian, Latin and Spanish. He compressed the first 10 000 characters of the Book of Genesis, using order 5 PPM character models which were trained on the remaining text. The results can be seen in Table 6. The lowest compression rates indicate the correct language (bold face numbers). A closely related topic to Language identification is dialect identification. Tests conducted by (Teahan 00) showed that the compression based approach could also be used for this. He compressed parts containing 100,000 characters each from the Brown Corpus for American English and the LOB Corpus for British English and trained a order 5 PPMD model on the remaining parts. The compression rates differed by 3.6% for the Brown Corpus and 3.1% for the LOB Corpus, and in this way identfied from which Corpus the samples were taken.

Original Text	Untrained	English	French	German	Italian	Latin	Spanish
English	1.97	**1.56**	2.30	2.38	2.30	2.33	2.29
French	2.13	2.54	**1.71**	2.56	2.54	2.59	2.56
German	2.14	2.64	2.60	**1.75**	2.55	2.59	2.59
Italian	2.26	2.67	2.66	2.65	**1.83**	2.61	2.68
Latin	2.45	2.87	2.92	2.91	2.82	**1.97**	2.84
Spanish	2.19	2.57	2.61	2.61	2.57	2.57	**1.75**
Size of training text (chars.)		180,359	175,331	191,840	170,923	149,121	169,461

Table 6: Results testing on Bible text (Teahan 00)

For a second test one year later, (Teahan et al. 01) collected web documents in sixteen different languages using www.google.com. The minimum for each language was about 100,000 bytes. The documents for each language were split in two halfes, one used for training the language models and the other for testing. The testing documents were again split into segments from 50 to 500 bytes. (Teahan et al. 01) reported that as much as 2500 bytes were enough training data to achieve an identification rate above 95%. In Table 7 you can see the complete listing of their results.

Language	**Precision at first rank**			
testing size in bytes	50	100	200	500
Czech	0.976	0.994	0.996	1.000
Danish	0.905	0.950	0.972	0.990
Dutch	0.988	0.994	1.000	1.000
English	0.933	0.980	0.992	1.000
Finnish	0.935	0.952	0.960	0.970
French	0.937	0.966	0.976	0.990
German	0.959	0.984	0.992	1.000
Hungarian	0.936	0.960	0.976	0.990
Icelandic	0.939	0.960	0.968	0.970
Italian	0.984	0.996	0.996	1.000
Norwegian	0.829	0.910	0.936	0.920
Polish	0.999	1.000	1.000	1.000
Portuguese	0.945	0.974	0.984	1.000
Romanian	1.000	1.000	1.000	1.000
Spanish	0.796	0.876	0.920	0.960
Swedish	0.959	0.978	0.988	0.990
Average	0.939	0.967	0.979	0.986

Table 7: Results for 16 different languages by (Teahan et al. 01)

Another interesing point is mentioned in (Teahan 00),namely classification of different languages within one document. Instead of using one PPM model at a time, various models are used. For every character the models are switched, and after a sequence of grouped characters a terminating character is set to limit the recognized sequence. By applying the Viterbi Algorithm the highest ranking model for a certain region within the text is found.

(Teahan 00) tried this on the same Bible data used before. For testing, 1000 words of each Bible text were extracted, split into a set of 50 samples containing 120 words. Each of the sample text contained blocks of 20 words from each of the languages. The result was that out of 34049 characters, there were only 180 misclassified characters . This corresponds to a correct identification rate of 99.5%. Teahan also tried word based instead of character based compression, which reduced the errors to 108. A consequence of the word based approach was also the reduction of the search space, which improved calculation speed. As can be seen the PPM compression based approach can also be used for language identification. It can be used either word- or character -based. Though the character-based approach does not score as high as the word-based approach, its use is more universial. No preprocessing like tokenization is needed for the characterbased approach and it can handle e.g Chinese documents and documents encoded in unicode. After discussing the compression based approach, the next section will deal with a variant of (Dunning 94)'s approach.

6 Language Identification and Character Sets by Kikui

A variant of Ted Dunnings statistical approach can be found in a paper by Gen-itiro Kikui (Kikui 95). He developed an algorithm to overcome the problem of tokenization. Tokenization of Asian languages is hardly possible as they do not have wordboundaries similar to European Languages. Kikui therefore tried to look into the character sets to first discriminate East Asian text from European text. Today there exist many different character coding systems, e.g. ISO 8859-1 or ASCII. Each coding system includes a character set and an encoding scheme, which defines how a byte sequence is to be represented. Depending on which coding system is used, the representation of a byte sequence can have various visual representations. If one does not know the applicable coding system, the bytestring cannot be represented in the desired way. A solution of this problem is the use of Unicode which combines all so far known characters in one coding system. As the other coding systems are still widely used, Unicode is not yet the standard encoding system. Their algorithm supports the following coding systems:

- **7bit Coding** ISO 646 USA (ASCII), JIS Code (ISO-2022-jp), KS C 5601-1992, GB 2312-80, ISO-2022-int
- **8bit Coding** ISO-8859-1, EUC-GB(Simplified Chinese), EUC-KS, EUC-JIS(UJIS), BIG5 (Traditional Chinese), Shift JIS(MS Kanji Code)
- **Entitiy Reference with ASCI** *entitiy references* (e.g. ö is represented as "&Ouml") for the ISO-8859 specifications.

They included the following languages:

- **European:** English, French, German, Spanish, Italian, Portugese
- **East Asian:** Chinese, Korean, Japanese

The algorithm works in two steps. In the first step the algorithm tries to find out which coding system was used in a document, i.e. either East-Asian or European. This can be easy if certain escape sequences occur. The encoding system ISO-2022 can include different character sets in one encoding scheme. To switch character sets, a special escape sequence is put into the text. In the coding system JIS (ISO-2022-jp) there exist such escape sequences which indicate the change of the character set. The escape sequence for example **"ESC $ B"** marks the beginning of the JIS character set and **"ESC (B"** indicates the beginning of the ASCII character set. If such escape sequences occur the algorithm uses two loops to find out the coding system and the language. In the first loop the algorithm tries to extract East-Asian characters trying each applicable coding system. (Kikui 95) provides an example. The first loop presupposes that a string has been encoded with EUC-JIS. EUC stands for "Extended Unix Code". It has one primary code set and can have three additional ones. EUC-JIS is the combination of ASCII, JIS X0201 and JIS X0208 and is used on Unix machines in Japan. If two adjacent bytes now match the pattern [A1H-FEH]{2}, i.e. a two byte sequence from A1 to FE, then this must be a JIS character. It is stored in a string list.

After the first loop has collected all applicable strings, the second loop of the algorithm tries to map the strings in the string list to a language. They are compared to the statistical language models for the East-Asian languages. If they pass a certain threshold, the document is classified according to the language models. If the threshold is not reached, the authors suppose that there are no East Asian strings, and the document is then passed on to the European language models. To find the probabilities for European languages they use the algorithm described by (Dunning 94). East-Asian languages have special properties, hence Cavnars N-gram strategy is not applicable here. Asian language dispose of a large number of characters and usually do not have clear word boundaries. Therefore, in contrast to the European languages, character based unigrams are used to create language models.

6.1 Evaluation of Kikui's Approach

For testing (Kikui 95) used a total of 1340 documents found on the internet, dividing them up into 700 for training and 640 for testing purposes. They reported an error rate of 4.8% for European documents and 4.6% for East-Asian documents. This can be seen as a progress as Cavnar's method relied on correctly encoded input. Most of the errors seem to be due to indiviual Asian characters, mostly proper names, occurring in documents that are written in a European language. In Table 8 their results are shown.

This concludes the part about the different language identification approaches. In the remaining section I will report on three Perl programs for language identification and show test results.

	Germ	Engl	Span	Fren	Ital	Port	Other
German	51						
English		178		1			6
Spanish		1	23				
French				90			1
Italian					3		
Portuguese						6	
Other	1	8	1				70

	Jap	Kor	chin	Other
Jap	123			4
Kor	1	7	0	0
Chin	2	0	47	0
Other	1	0	2	8

Table 8: Testing results (Kikui 95)

7 Open source Language Identification Programs

After the description of the different language identification approaches, I will now report on two Perl language identification modules ((LING IDNT) and (LING IDFY)) and (Resnik 99) , which I have tested and compared. First I will introduce Lingua::Ident.

Language::Ident is a Perl module implemented by Michael Piotrowski. It is an implementation of the algorithm described in (Dunning 94). In order to use it, a Perl script called *trainlid* has to be run on sample texts in different languages. The script creates transition matrices, i.e. bi- and tri-grams with their respective probability of occurrence for each of the sample documents. For the testing suite I have created language models for the following languages: German, Englisch, Spanish, Swedish, Portuguese, Finnish, Italian, French, Dutch and Danish. Training was done with papers of about 3.5 MB size containing political speeches. For testing I used part one of the European Constitution which is available in all the languages I needed. The document can be found on (EU Constitution 04).

The second module (LING IDFY) has been written by Jose Castro. The program already comes with trained language models and so far supports 26 languages. It has been developed to work with several language identification methods. Supported idenfication methods are N-grams, common words and affixes. All the approaches can be tried separately or combined. When used in combined mode, one can indicate the percentage of how much each method accounts for the process of identification. In order for Lingua::Identify to function correctly the following modules have to be installed first Class::Factory::Util , Text::Affixes , Text::ExtractWords and Text::N-gram. They are all available on (CPAN).

The Text_Cat program by (van Noord) is an implementation of (Cavnar et al. 94)'s N-gram approach. It currently supports 77 languages. Text_Cat can also create new language profiles. For testing the existing language models of the program were used.

7.1 Evaluation of Open Source Identification Programs

Lingua::Ident was tested against three different methods for Lingua::Identify and Text_Cat. I have created a Perl tester script, which reads one document at a time and splits it into 13 samples each of 3, 5, 10, 20, 30, 50, 100, 200. The remaining text was split into samples of 500 tokens each and also tested. One should note that only the needed languages in Lingua::Ident and Text_Cat are set to active, as to provide fairer testing conditions. After creation of the samples, all of them are tested with both modules. At the end a summary is printed out, providing the number of samples tested, the number of correctly and incorrectly classified samples and the identification percentage. The misclassified samples are also printed out with the assigned language tag. Usage of the program is:

```
mismo@myubuntu:$ ./tester language.txt language-specific-abbreviation
```

where *language.txt* is the path to the input file containing part one of the European Constitution or any other file and *"language specific abbreviation"* is the abbreviation for the language, e.g "de" for a German text. In Table 9 the results are shown for the ten languages tested. Column (Ident) provides the results for testing with Lingua

Ident. The remaining columns provide results for testing the Identify module using various methods. In the left part of Table 9 results are shown for testing without preprocessing, i.e. punctuation characters remained. In the right part the data was preprocessed before testing. Lingua::Identify was tested in mixed mode, with a combination of N-grams and small words and with N-grams. The default mixed mode, has the following ranking for the methods used:

- small words = 0.5
- prefixes2 =1
- suffixes3 = 1
- ngrams = 1.3

The next column shows the result for testing with N-grams (weight 1.5) and small words (weight 0.5). The column "n-gram" contains the results testing Lingua::Identify only with N-grams. As one can notice, the results are slightly better when preprocessing was done before the testing for all of the programs. Lingua::Ident and Text_Cat have reached very reliable identification rates in comparison to Lingua::Identify. A reason for this could be that I used the already existing language models for Lingua::Identify. In the N-gram columns identification Italian, Swedish and Finnish are very low, although reliability grew with preprocessing.

Language	Ident	mixed	n-gram, sw	n-gram	text cat	Language	Ident	mixed	n-gram, sw	n-gram	text cat
German	97.32	95.54	95.53	88.39	99.11	German	99.08	96.33	96.33	90.83	99.1
English	100.00	93.86	96.49	92.98	98.25	English	100.00	96.43	99.11	96.43	99.11
French	98.25	90.35	81.58	85.96	97.37	French	98.20	91	75.68	82.88	96.4
Spanish	99.13	90.43	93.91	93.91	93.04	Spanish	98.23	94.69	94.69	92.04	96.46
Italian	100.00	84.82	83.93	40.18	93.75	Italian	100.00	83.64	77.27	48.18	92.73
Dutch	98.25	92.98	83.33	88.60	94.74	Dutch	98.21	92.86	85.71	90.18	96.43
Danish	100.00	90.00	87.27	80.91	95.45	Danish	100.00	87.96	89.81	81.48	96.3
Swedish	99.10	82.88	81.98	14.41	98.2	Swedish	100.00	84.40	85.32	24.77	98.17
Portuguese	97.35	91.15	92.92	88.50	93.81	Portuguese	99.1	88.18	88.18	81.81	93.64
Finnish	100.00	93.40	83.01	43.40	98.11	Finnish	99.03	91.34	88.46	47.12	98.08
Average:	**98.94**	**90.54**	**~88**	**71.72**	**96.18**	**Average**	**99.18**	**90.63**	**88.05**	**73.57**	**96.64**

Table 9: Results Testing Perl Programs

8 Summary

In this paper the task and evaluation of language identification has been presented. In Section 2 the small word approach was explained and discussed. Section 3 contained information about (Dunning 94)' statistical approach using Markov Models. In section 4 I discussed the N-gram approach. This approach only works for languages which can be tokenized. In section 5 the use of the compression-based approach with PPM by (Teahan et al. 01) was explained. In the last section I have introduced three language identification programs for Perl, which were tested in ten different languages. After all we can see that the language identification task has been mastered with high accuracy as far as recognition rates are concerned. The topic will continue to be of importance for years as international communication keeps growing in importance for business and private use. A related field is the language recognition of speech, which uses similar methods to the one described here, but for audio data instead of written documents. It is a field of active research and in the years to come, we will see if it reaches the same identification rates as the ones for written documents.

Contents

List of Tables

References

[Bell et al. 90] Bell T.C., Cleary J.G and Witten I.H. 1990 *"Text Compression"*, Prentice Hall,New Jersey, 1990

[Cavnar et al. 94] Cavnar, W.B. and Trenkle, J. M. *" N-gram-based text categorization"* In 1994 Symposium on Document Analysis and Information Retrieval in Las Vegas, pp.161-175, 1994,http://citeseer.ist.psu.edu/68861.html

[Churcher 94] Churcher,G *Distinctive character sequences*, 1994, personal communication by Ted Dunning

[CPAN] http://www.cpan.org

[Dunning 94] Dunning T.*"Statistical Identification of language"* ,Computing Research Laboratory, New Mexico State University: Technical Report Pages 94-273, 1994,http://citeseer.ist.psu.edu/dunning94statistical.html

[EU Constitution 04] http://europa.eu.int/eur-lex/lex/ last visited on 25.August 2005

[Grefenstette 95] Grefenstette, G.*"Comparing two language identification schemes"*, In: Proceedings of JADT 1995, 3rd International Conference on Statistical Analysis of Textual Data, 1995, http://www.xrce.xerox.com/competencies/content-analysis/tools/publis/jadt.ps

[Kikui 95] Kikui G. "Identifying, the coding system and language, of on-line documents on the Internet" . In Proceedings of the 16th conference on Computational Linguistics - Volume 2,pp: 652 - 657,1996, http://citeseer.ist.psu.edu/588447.html

[Langer 01] Langer, St. *"Sprachen auf dem WWW"* In: Proceedings der GLDV-Jahrestagung 2001, pp. 85-91 , 2001 http://www.uni-giessen.de/fb09/ascl/gldv2001/proceedings/pdf/GLDV2001-langer.pdf

[Lextek] http://www.lextek.com/langid/ last visited on 25.August 2005

[LING IDNT] http://search.cpan.org/∼mpiotr/Lingua-Ident-1.5/Ident.pm by Michael Piotrowski

[LING IDFY] http://search.cpan.org/∼cog/Lingua-Identify-0.16/lib/Lingua/Identify.pm by Jose Castro

[Resnik 99] Resnik, P. *"Mining the Web for Bilingual Text"* In: Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL99), 1999, http://myweb.ncku.edu.tw/ p7693177/Bilingual/5.pdf

[Shannon 48] SHANNON, C.E. *"A Mathematical Theory of Communication"*, Bell Syst. Tech. J., 27, 379-423,1948

[Teahan 00] Teahan, W. *"Text classification and segmentation using minimum cross entropy"* Proceedings RIAO 2000, 6th International Conference Recherche d'Information Assistee par Ordinateur, pp. 943-961, 2000,http://citeseer.ist.psu.edu/410024.html

[Teahan et al. 01] Teahan, W. and Harper D.*"Using Compression-Based Language Models for Text Categorization"* In: Proceedings of 2001 Workshop on Language Modeling and Information Retrieval, 2001,http://citeseer.ist.psu.edu/435756.html

[van Noord] Gertjan van Noord, Text_Cat http://odur.let.rug.nl/ vannoord/TextCat/ last visited on 25 August 2005

[WIKI.org] http://en.wikipedia.org/wiki/Zipf's_law visited 25.August 2005