Multilingual Document Language Recognition for Creating Corpora

Yevgeny Ludovik **Ron Zacharski**

New Mexico State University
USA

Abstract

In this paper we describe a language recognition algorithm for multilingual documents that is based on mixed-order n-grams, Markov chains, maximum likelihood, and dynamic programming. We present the results of an experimental study that showed that the performance of this algorithm has practical value.

1 Introduction

Language recognition algorithms are an essential component of many natural language processing systems. For example, a system that translates web-documents must first determine the language of that document. Moreover, in multilingual documents, the system must determine monolingual segments and the language of each segment. Language recognition algorithms are also essential in the *development* of many natural language processing systems. For example, very large corpora are often required in the development of machine translation systems and a language identification system can be combined with a web spider to automatically collect such corpora.

These recognition algorithms are not just concerned with the language of electronic texts, but also with how the characters of that language are encoded. This specification is important because the characters of a given language may be represented by several distinct encoding methods or code sets. For example, Japanese text may be in J1S-X-0208-1990 or JIS-X-0212-1990, among others, and Chinese text might be in GB-2312-80 or Big Five. Thus, throughout this paper we will always be concerned with identifying a specific language/code set pair. However, to ease exposition we use the term *language* to mean this language/code set pair.

In the following sections we describe a language recognition algorithm for multilingual documents based on mixed order n-grams, Markov chains, maximum likelihood, and dynamic programming. We present the results of a study that examined the performance of this algorithm on multilingual documents, which consisted of fragments in 30 languages. We discuss how the algorithm can be incorporated into a web spider that collects multilingual documents in specified languages.

2 Recognition Algorithm for Monolingual Documents

A common sense and frequently used approach to language identification has been to use lists of words that commonly appear in the languages of interest (Ingle 76).That is, for every language of interest one constructs a list containing the most frequent words in that language. For example, *the, of,* and *and* are common words in English, and *el, de, que,* and *y* are common words in Spanish. If you are trying to identify the language of a text and you see a lot of *the*'s *of*'s and *and*'s in the text a good guess would be that the text is in English. Similarly, if you see *el, que,* and *de,* a good guess would be Spanish. While this approach is simple, intuitive, and performs adequately in many cases, it does have several drawbacks. First, while the approach works well for identifying texts of moderate length, its performance significantly degrades as the text to be identified gets smaller. Second, since it is a word-based approach it crucially depends on the ability to segment the text into words. This is easy if the task is to recognize languages like English and Spanish where words are bordered by spaces, but is significantly more challenging if we include languages that do not use spaces as word delimiters such as Chinese and Japanese.

The approach we propose here is not dependent on identifying the words of a text. Rather, it is a statistically based approach that learns to distinguish between languages by building an n-gram model.[1] It is similar to the work of Cavnar and Trenkle (Cavnar & Trenkle 94), among others, in that it makes use of

[1] The n-grams of a text are all the character sequences of length n contained in that text. For example, *unmarked helicopters,* contains 20 unigrams *(u, n, m, a ,r, …),* 19 bigrams *(un, nm, ma, …), 1*8 trigrams *(unm, nma, mar, …)* and so on.

mixed-order n-grams.[2] In contrast to the Cavnar and Trenkle algorithm, which uses an ad hoc rank-order distance measure, the algorithm we propose here is theoretically well motivated based on a maximum likelihood approach.

The algorithm has two main phases: training and classification.

2.1 Training

The algorithm is a variable length n-gram approach. We start with an empty common n-gram pool, *CNP,* which we will fill with n-grams from each training text—m_1 unigrams, m_2 bigrams, ... m_n n-grams. For every language we extract the n-grams (n = 1...N) from the training text. (In our comparative experiments we used an N of 4.) First, we create separate lists for unigrams, bigrams, and so on. Next, we add a portion of these n-grams to *CNP* (the common pool); first we add unigrams, then bigrams, and so on.

We select the unigrams to add to *CNP* as follows. For every unigram, a_1 observed in the training data, we compute the following training weight:[3]

$$W(a_1) = -p(a_1)\log p(a_1)$$

We then order the list of all unigrams in descending order on this value and place the top m_1 unigrams in the common pool, *CNP*.

Then we use the following recursive procedure. Suppose we have already picked up n-grams of length 1 … (k-1). For every k-gram $(a_1a_2...a_k)$ we compute a training weight. If the (k-l)-gram $(a_2...a_k)$ has not been included in *CNP* the training weight value is[4]

$$W(a_1a_2\ldots a_k) = -p(a_1a_2\ldots a_k)\log p(a_k|\, a_1a_2\ldots a_{k-1})$$

If $(a_2...a_k)$ is in *CNP,* then we use a slightly different formula:

$$W(a_1a_2\ldots a_k) = -p(a_1a_2\ldots a_k)(\log p(a_k|\, a_1a_2\ldots a_{k-1}) - \log p(a_k|\, a_2a_3\ldots a_{k-1})).$$

Next we sort the k-gram list in descending order on this weight and place the m_k top k-grams in *CNP*. The training weights describe the reduction of the cross entropy between training data and the model if we include the k-gram $(a_1a_2...a_k)$ in our pool. We repeat the procedure for k = 2 ... N. For the comparative experiments described in §3 we used an m_1 of 170, an m_2 of 200, an m_3 of 400, and an m_4 of 230. The number of unigrams chosen was enough to include all the unigrams for all languages in the training data. These numbers have been determined through experimentation.

At the end of this process *CNP,* the common n-gram pool, contains the union of the selected n-grams for all L languages in our training set. For every n-gram (n = 1 ... N) in *CNP* we compute a primary recognition weight L-dimension vector, *PRW,* as follows. Each *i*-th component of the vector, *PRW,* is associated with the *i*-th language, and contains a recognition weight. For unigrams, the recognition weight is

$$PRW_i(a_1) = -\log p_i(a_1)$$

For all other n-grams, the recognition weight is:

$$PRW_i(a_1a_2\ldots a_k) = -\log p_i(a_k|\, a_1a_2\ldots a_{k-1})$$

That is, the anti-log of the conditional probability of a_k given $a_1a_2...a_{k-1}$ in the training set associated with the *i*-th language. If some n-gram in *CNP* has not been encountered in some language training data, its recognition weight for this language is defined to be some maximum value MAX (in our experiments, MAX = 20.) At the end of this process we have a primary recognition weight vector, $PRW(a_1a_2...a_k)$ for every n-gram, $a_1a_2...a_k$, in our n-gram pool, *CNP*. Now we define the recognition weight L-dimension vector of any N-gram (encountered or not encountered in our training data) as

$$RW(a_1a_2\ldots a_N) = \begin{cases} PRW(a_1a_2\ldots a_N) & a_1a_2\ldots a_N \in CNP \\ PRW(a_2a_3\ldots a_N) & a_2a_3\ldots a_N \in CNP \\ \ldots & \ldots \\ PRW(a_N) & a_N \in CNP \\ \textit{vector of MAX} & \textit{otherwise} \end{cases}$$

In addition to developing the *CNP* containing recognition weight vectors, we also compute a weight average and weight dispersion for each language (these are used in the verification step of the recognition phase described in the following section). This is done as follows. We divide the training text of a given language, *i*, into K 500 byte segments $(x_1x_2...x_{500})$. For every segment *k* we compute:

[2] See also. Churcher, Hayes, Johnson, and Souter (Churcher et al. 94) and Dunning (Dunning 94).

[3] *p(a)* represents the probability of a.

[4] $p(a_n|a_1..\ .a_{n-1})$ represents the conditional probability. That is, the probability of character a_n given that the immediately previous characters were $a_1...a_{n-1}$. For example, p('e'|'th') is the probability that the third character is 'e' given that the first two characters are 'th'.

$$d_{i,k} = \sum_{s=1}^{500} \frac{RW_i(x_{s-N+1}x_{s-N+2}\ldots x_s)}{500}$$

The weight average for language *i* is defined as

$$WA_i = \frac{\sum_{k=1}^{K} d_{i,k}}{K}$$

The dispersion for language *i* is defined as:

$$D_i^2 = \frac{\sum_{k=1}^{K} (d_{i,k} - WA_i)^2}{K}$$

2.2 The Recognition Phase

The recognition process consists of two steps. First, in the classification step, we tentatively classify the text to be recognized as being in one of the languages specified in the training phase. Next, in a verification phase, we determine how well the text to be recognized fits the proposed language. If the fit is not good enough, we classify the language of the text as unknown.

Classification Step: Let $x_0x_1\ldots x_{s-1}$ be the byte sequence of the text to be classified. We define a result recognition weight vector as:

$$RRW = \frac{\sum_{s=0}^{S-1} RW(x_{s-N+1}x_{s-N+2}\ldots x_s)}{S}$$

The recognition result (the proposed language of the text) is

$$i^* = \arg\min_i RRW_i$$

(That is, the text is classified being in the language, *i**, associated with the result recognition weight vector component with the lowest value.)

Verification Step: If [5]

$$\frac{RRW_{i^*} - WA_{i^*}}{D_{i^*}} \le VER\_THR$$

then we recognize the text as belonging to language *i**. If this does not hold then the language of the text is classified as "unknown".

3 Evaluation of Monolingual Recognizer

We evaluated this monolingual recognizer by comparing its performance to the Cavnar and Trenkle n-gram algorithm (Cavnar and Trenkle 1994).

3.1 The Method

The training data for both algorithms consisted of 50k samples from the following 34 languages:

Afrikaans	Italian
Albanian	Japanese
Arabic	Korean
Bulgarian	Latin
Chinese	Lithuanian
Croatian	Malay
Czech	Norwegian
Danish	Persian
Dutch	Polish
English	Portuguese
Estonian	Russian
French	Serbian
German	Slovak
Greek	Spanish
Haitian Creole	Swedish
Hawaiian	Thai
Icelandic	Turkish

We evaluated each algorithm under five conditions that varied as to the size of the text to be identified: 1k, 500, 100, 50, and 20 byte samples. The samples were distinct from the training text and were drawn from 200k texts from each language. 200 samples for each language were used to evaluate the algorithms in the 1k condition, 400 samples for the 500 byte condition, 2000 samples for the 100 byte condition, 4000 samples for the 50 byte condition, and 10,000 samples for the 20 byte condition.

[5] The constant, VER_THR, is used to rule out documents, which are not in the pre-trained languages.

3.2 Results

The comparison results for the algorithms (the Cavnar & Trenkle (C&T) algorithm, and the monolingual algorithm described above (mono) are shown in the following table.

Percent Error Rate of Algorithms at Different Sample Sizes

Sample Size	Algorithms	
	mono	C&T
1000	0.27	3.03
500	0.52	3.10
100	2.02	4.23
50	4.01	6.86
20	11.92	16.56

As this table shows, the performance of the algorithms is good when the text to be classified is relatively large (500-1,000 bytes). However, even under this condition it should be noted that the algorithm proposed here has less than one sixth the error rate of the Cavnar and Trenkle algorithm. A significant percentage of the error rate in the 1,000 and 500 byte conditions is due to misidentification of the test samples from Haitian Creole and its lexifier, French.[6] This is probably due to the fact that Creoles borrow much of their vocabularies from their respective lexifiers (Romaine 88). It is interesting to note that the algorithms performed relatively well on Afrikaans and Dutch, although many creolists regard Afrikaans as a semi-creole of Dutch.

4 Recognition Algorithm for Multilingual Documents

While this algorithm has important uses in many applications it does have some limitations in the current form. For example, suppose we have a multilanguage machine translation system that converts web pages into an "English-only" form that is displayed to users. If the original web pages are in a single language then the task is simply to identify the language of the page using the current algorithm "as is" and then applying the appropriate machine translation. However, if the web pages are multilingual then the process is more complex. In this case we need to segment the page into single-language chunks, identify the language of each chunk, and then perform the correct translation. This task is illustrated in the following example.[7]

His example is essentially this (taken from Chechen): if in a language geminates occur only inter-syllabically, and the non-geminate version appears in those intersyllabic positions plus word-initially and word-finally, then if we know the average number of syllables per word, we can make a prediction of the relative frequency of the single and geminate versions of a consonant. If our prediction does not match the reality, then we can infer there is something that remains to be accounted for.

> "Le chiffre absolu de la fréquence réelle d'un phoneme n'a qu'une importance accessoire. Seul le rapport entre ce chiffre et le chiffre de fréquence attendu théoriquement possède une valeur veritable. (284)'Le calcul des probabilités théoriques n'est pas toujours aussi simple que dans les exemples ci-dessus. Mais on ne doit pas se laisser rebuter par les difficultés d'un tel calcul, car c'est seulement par comparaison avec les chiffres de fréquence possible obtenus au moyen de ces calculs que les chiffres de fréquence effective acquièrent une valeur, en montrant si un phonème, dans la langue en question, est beaucoup ou peu utilisé. (285)."

This is a powerful notion that remains to be fully explored. What Troubetzkoy (and others since) have seen is that a study of frequency can often be tantamount to a search for lurking generalizations.

As described above, the system needs to segment this document, identify the languages of the chunks (in this case English and French) and translate the French segment into English. Fortunately, an important advantage of the approach presented above is that it can be generalized for multilingual documents. This generalization is accomplished by adding a dynamic programming algorithm based on a simple Markov model of multilingual documents. This algorithm is described in the following section.

[6] Without these two languages the results for the 1,000 and 500 byte tests are as follows:

Sample Size	Algorithms	
	mono	C&T
1000	0.05	2.05
500	0.22	1.91

[7] from http://humantities.uchicago.edu/faculty/goldsmith/Royaumont98/InfoTheory.html.

4.1 Segmentation algorithm

The algorithm is based on dynamic programming that had been first used for segmentation in (Vintsiuk 70) and is a further development of a computationally effective algorithm presented in (Ludovik 82). It consists of two steps:

- a recognition step where we apply a dynamic programming algorithm to find the segmentation maximizing the likelihood of the total character sequence of the document being processed,
- a verification step, which determines how well every segment fits the language assigned to it.

Recognition step

The segmentation algorithm is based on the following Markov model of a multilingual document. The model has one state per language, $1 \le i \le L$ plus one additional 0-state accounting for segments that are not in any of pre-trained languages, in particular, such segments may be in no language at all (e.g., a table of numbers).

If a system is in a state i it generates characters in the i-th language depending on the left context according to the n-gram model described above in §2. Switching from language i_1 to language i_2 is defined by transition probabilities $p(i_2/i_1)$ and probability distribution of segment length r $p(r)$, $r_{min} \le r \le r_{max}$, ranging from minimal segment length r_{min} to maximal r_{max}. The algorithm presented below finds the segments and segment languages assuring the maximum likelihood of the observed text given the Markov model. If k-th segment starts at s_{k-1} and ends at s_k-1, then using antilog, the criterion to minimize is as follows:

$$Q(\{x_s, 0 \le s < S\}, \{s_k, i_k, 0 \le k \le K\}) = \sum_{k=1}^{K} TSW(i_{k-1}, i_k, s_{k-1}, s_k)$$

where *TSW* is a segment weight:

$$TSW(i_{k-1}, i_k, s_{k-1}, s_k) = \sum_{s=s_{k-1}}^{s_k-1} RW_{i_k}(x_{s-N+1}x_{s-N+2}\ldots x_s) - \log p(i_k \mid i_{k-1}) - \log p(s_k - s_{k-1})$$

if i_k is not 0. Otherwise:[8]

$$TSW(i_{k-1}, i_k, s_{k-1}, s_k) = \text{JUNK\_THR}*(s_k - s_{k-1}) - \min_{i:1\le i\le L}\left[\sum_{s=s_{k-1}}^{s_k-1} RW_i(x_{s-N+1}\ldots x_s) - \log p(i \mid i_{k-1})\right] - \log p(s_k - s_{k-1})$$

The dynamic programming algorithm that finds optimal values of $\{s_k, i_k, 0<k<K\}$ consists in the following iterative step for

$$Ind(i_k, s_k) = (i_{k-1}^*, s_{k-1}^*) = \underset{\substack{(i_{k-1}, s_{k-1}):\\ i_{k-1}:0\le i_{k-1}\le L;\\ s_{k-1}:r_{min}\le(s_k - s_{k-1})\le r_{max}, s_{k-1}\ge 0}}{\arg\min} \left(F(s_{k-1}, i_{k-1}) + TSW(i_{k-1}, i_k, s_{k-1}, s_k)\right)$$

$$F(i_k, s_k) = F(i_{k-1}^*, s_{k-1}^*) + TSW(i_{k-1}^*, i_k, s_{k-1}^*, s_k)),$$

with initial values:

$$F(i, 0) = 0,\ 0<=i<=L,$$
$$F(i, s) = \text{INFINITY}, s:\ 0 < s < r_{min}.,\ 0<=i<=L.$$

After the algorithm has finished with the recursive steps, we find the language of the last segment in the optimal set of segments:

$$i^* = \underset{i:0\le i\le L}{\arg\min} F(i, S)$$

where S is the total length of the text The value i^* together with information stored in array *Ind(i, s)* will allow us to get all segments with corresponding language labels from the optimal set of segments. During this process, the optimal number of segments is automatically determined.

[8] The constant JUNK_THR ("junk threshold") is used to rule out document segments that are not in any of the pre-trained languages.

Verification step

The verification step is executed exactly as in the monolingual algorithm regarding every segment as a separate monolingual document.

4.2 Evaluation

We evaluated the performance of the segmentation algorithm on segmenting documents that contain multilingual text. The test data contained text in the following languages:

Afrikaans	Japanese
Albanian	Korean
Arabic	Latin
Chinese	Lithuanian
Croatian	Malay
Czech	Norwegian
Danish	Persian
Dutch	Portuguese
English	Russian
Estonian	Serbian
French	Slovak
German	Spanish
Greek	Thai
Italian	Turkish

The test documents were created by concatenating randomly selected segments from documents in the languages listed above. In this way we created six documents; each document contained 1000 segments having approximately the same length (20, 50, 100, 200. 500, or 1000 bytes).[9] Thus, all bytes in a document are labeled with a language. We compared this known labeling to the labeling based on the optimal segmentation produced by the algorithm. The error rate was computed by dividing the number of mislabeled bytes by the length of the document. The following table shows the segmentation error rate as a factor of segment byte size.

Percent Error Rate of Multilingual Algorithm at Different Segment Sizes

Avg. Segment Size	% Error Rate
1,034	0.47
540	0.69
202	1.40
101	2.08
49	4.70
20	12.88

[9] The length ranges were 17-23, 45-55, 90-110, 190-210, 500-550, 1000-1060.

As this chart shows, this algorithm works extremely well for moderate-sized segments and performs adequately for short (approximately 3 word) segments.

5 Spider for collecting multilingual corpora

A variety of natural language processing tasks (for example, creating lexicons, and proper noun dictionaries) could make use of multilingual corpora. While pre-existing multilingual corpora are available for some languages (often in the form of aligned texts), they are not available for all languages that are of interest to researchers. The algorithm we have just described is a key element in a web spider, which collects such texts.

The person using this spider specifies a set of languages (for example, English, and Russian) and approximate percentage desired for each language. For example, if we are trying to collect potentially parallel texts, the user would specify that the document be approximately 50% English and 50% Russian. The person using the spider also specifies a set of starting URLs. These URLs are placed on a queue of places to visit. The user can optionally specify an "only get pages modified since" date.

The spider is of standard design and conforms to general spider behavior conventions. It examines the robots.txt file at each site specified to see if it is excluded from that site. It checks for meta robot tags in the URL text itself to see if it is either excluded from reading the text or excluded from following the links contained on the page. It identifies itself by using the user-agent and from fields supported in HTTP header requests. Finally, it waits several minutes before visiting the same site again to avoid overloading the server.

If the spider is not excluded from reading the URL text, it gets the text, strips out the HTML tags and passes the text to the language recognizer. If the recognizer identifies the text as matching the specified languages in the specified percentages, the text is saved in a local file. If the spider is not excluded from following the links on a page the links are collected. This link list is filtered to exclude a variety of links including those to JPEG and MIDI files, links to CGI scripts and links to common sites like www.altavista.digital.com and www.hotbot.com. URLs of sites the robot has previously visited are also filtered out. If the page containing these links matched the target language the links are placed on a priority queue. If the page does not match the target the links are placed on the regular queue. URLs on the priority queue are visited before those of the

regular queue. The algorithm continues until there are no URLs in either queue.

6 Conclusions

In this paper we have described a language recognition algorithm for multilingual documents. This algorithm is based on mixed order n-grams, Markov chains, maximum likelihood, and dynamic programming. In §3, we described the results of an experimental study that shows that the proposed algorithm significantly outperforms the Cavnar and Trenkle algorithm, one of the most popular language recognition algorithms. Moreover, the proposed algorithm has an additional benefit. The Cavnar and Trenkle algorithm finds the closest language that matches the text to be classified, but give no guarantee that this language is actually the language of the text. The proposed algorithm has a separate verification step that assures, with a controllable degree of certainty, that the text to be classified is actually *in* the closest language. Because of these advantages, the algorithm can be successfully used in a variety of applications.

In §4 we described a language recognition algorithm for multilingual documents, and presented the results of an experimental study which showed that the performance of this algorithm was extremely good. For example, for 20 byte segments (roughly three words) the error rate was only 13%, which means that only that the initial and final bytes are misrecognized. These misrecognized initial and final bytes could be spaces or punctuation and not be serious errors for applications. These results suggest that the algorithm has practical value; this algorithm offers a new reliable tool for natural language engineering, which allows for more sophisticated processing of documents including web pages. In §5 we presented one such application for this algorithm: a web spider that collects multilingual documents.

References

Cavnar W. and Trenkle J. (1994). "N-gram-based text categorization". Symposium on Document Analysis and Information Retrieval.

Churcher G., Hayes J., Johnson S., and Souter C. (1994). "Bigram and trigram models for language identification and character recognition". In Proceedings of the 1994 AISB Workshop on Computational Linguistics for Speech and Handwriting Recognition.

Ted Dunning T. (1994). "Statistical identification of language". Computing Research Laboratory Technical Report MCCS-94-273. New Mexico State University.

Ingle N. C. (1976). "A language identification table". The incorporated linguist 15(4).98-101.

Romaine S. (1988). "Pidgin and Creole Languages". Longman.

Ludovik Ye. (1982). "An Algorithm for optimal quasi linear compression of speech signals", in Proceedings of "Automatic Speech Recognition-12". Odessa, USSR.

Vintsiuk T.K. (1970), "Optimal splitting of a sequence of elements into subsequences". Cybernetics, v.4,pp. 128-133, Kiev, USSR.