Text Segmentation by Language Using Minimum Description Length

Hiroshi Yamaguchi
Graduate School of
Information Science and Technology,
University of Tokyo
yamaguchi.hiroshi@ci.i.u-tokyo.ac.jp

Kumiko Tanaka-Ishii
Faculty and Graduate School of Information
Science and Electrical Engineering,
Kyushu University
kumiko@ait.kyushu-u.ac.jp

Abstract

The problem addressed in this paper is to segment a given multilingual document into segments for each language and then identify the language of each segment. The problem was motivated by an attempt to collect a large amount of linguistic data for non-major languages from the web. The problem is formulated in terms of obtaining the minimum description length of a text, and the proposed solution finds the segments and their languages through dynamic programming. Empirical results demonstrating the potential of this approach are presented for experiments using texts taken from the Universal Declaration of Human Rights and Wikipedia, covering more than 200 languages.

1 Introduction

For the purposes of this paper, a multilingual text means one containing text segments, limited to those longer than a clause, written in different languages. We can often find such texts in linguistic resources collected from the World Wide Web for many non-major languages, which tend to also contain portions of text in a major language. In automatic processing of such multilingual texts, they must first be segmented by language, and the language of each segment must be identified, since many state-of-the-art NLP applications are built by learning a gold standard for one specific language. Moreover, segmentation is useful for other objectives such as collecting linguistic resources for non-major languages and automatically removing portions written in major languages, as noted above. The study reported here was motivated by this objective. The problem addressed in this article is thus to segment a multilingual text by language and identify the language of each segment. In addition, for our objective, the set of target languages consists of not only major languages but also many non-major languages: more than 200 languages in total.

Previous work that directly concerns the problem addressed in this paper is rare. The most similar previous work that we know of comes from two sources and can be summarized as follows. First, (Teahan, 2000) attempted to segment multilingual texts by using text segmentation methods used for non-segmented languages. For this purpose, he used a gold standard of multilingual texts annotated by borders and languages. This segmentation approach is similar to that of word segmentation for non-segmented texts, and he tested it on six different European languages. Although the problem setting is similar to ours, the formulation and solution are different, particularly in that our method uses only a monolingual gold standard, not a multilingual one as in Teahan's study. Second, (Alex, 2005) (Alex et al., 2007) solved the problem of detecting words and phrases in languages other than the principal language of a given text. They used statistical language modeling and heuristics to detect foreign words and tested the case of English embedded in German texts. They also reported that such processing would raise the performance of German parsers. Here again, the problem setting is similar to ours but not exactly the same, since the embedded text portions were assumed to be words. Moreover, the authors only tested for the specific language pair of English embedded in German texts. In contrast, our work considers more than 200 languages, and the portions of embedded text are larger: up to the paragraph level to accommodate the reality of multilingual texts. The extension of our work to address the foreign word detection problem would be an interesting future work.

From a broader view, the problem addressed in this paper is further related to two genres of previous work. The first genre is text segmentation. Our problem can be situated as a sub-problem from the viewpoint of language change. A more common setting in the NLP context is segmentation into semantically coherent text portions, of which a representative method is *text tiling* as reported by (Hearst, 1997). There could be other possible bases for text segmentation, and our study, in a way, could lead to generalizing the problem. The second genre is classification, and the specific problem of text classification by language has drawn substantial attention (Grefenstette, 1995) (Kruengkrai et al., 2005) (Kikui, 1996). Current state-of-the-art solutions use machine learning methods for languages with abundant supervision, and the performance is usually high enough for practical use. This article concerns that problem together with segmentation but has another particularity in aiming at classification into a substantial number of categories, i.e., more than 200 languages. This means that the amount of training data has to remain small, so the methods to be adopted must take this point into consideration. Among works on text classification into languages, our proposal is based on previous studies using cross-entropy such as (Teahan, 2000) and (Juola, 1997). We explain these works in further detail in §3.

This article presents one way to formulate the segmentation and identification problem as a combinatorial optimization problem; specifically, to find the set of segments and their languages that minimizes the description length of a given multilingual text. In the following, we describe the problem formulation and a solution to the problem, and then discuss the performance of our method.

2 Problem Formulation

In our setting, we assume that a small amount (up to kilobytes) of monolingual plain text sample data is available for every language, e.g., the Universal Declaration of Human Rights, which serves to generate the language model used for language identification. This entails two sub-assumptions.

First, we assume that for all multilingual text, every text portion is written in one of the given languages; there is no input text of an unknown language without learning data. In other words, we use supervised learning. In line with recent trends in unsupervised segmentation, the problem of finding segments without supervision could be solved through approaches such as Bayesian methods; however, we report our result for the supervised setting since we believe that every segment must be labeled by language to undergo further processing.

Second, we cannot assume a large amount of learning data, since our objective requires us to consider segmentation by both major and non-major languages. For most non-major languages, only a limited amount of corpus data is available.[1]

This constraint suggests the difficulty of applying certain state-of the art machine learning methods requiring a large learning corpus. Hence, our formulation is based on the minimum description length (MDL), which works with relatively small amounts of learning data.

In this article, we use the following terms and notations. A multilingual text to be segmented is denoted as $X = x_1, \ldots, x_{|X|}$, where x_i denotes the i-th character of X and $|X|$ denotes the text's length. *Text segmentation by language* refers here to the process of segmenting X by a set of borders $\mathbb{B} = [B_1, \ldots, B_{|\mathbb{B}|}]$, where $|\mathbb{B}|$ denotes the number of borders, and each B_i indicates the location of a language border as an offset number of characters from the beginning. Note that a pair of square brackets indicates a list. Segmentation in this paper is character-based, i.e., a B_i may refer to a position inside a word. The list of *segments* obtained from $\mathbb{B}$ is denoted as $\mathbb{X} = [X_0, \ldots, X_{|\mathbb{B}|}]$, where the concatenation of the segments equals X. The language of each segment X_i is denoted as L_i, where $L_i \in \mathcal{L}$, the set of languages. Finally, $\mathbb{L} = [L_0, \ldots, L_{|\mathbb{B}|}]$ denotes the sequence of languages corresponding to each segment X_i. The elements in each adjacent pair in $\mathbb{L}$ must be different.

We formulate the problem of segmenting a multilingual text by language as follows. Given a multilingual text X, the segments $\mathbb{X}$ for a list of borders $\mathbb{B}$ are obtained with the corresponding languages $\mathbb{L}$. Then, the total description length is obtained by calculating each description length of a segment X_i for the language L_i:

$$(\hat{\mathbb{X}}, \hat{\mathbb{L}}) = \underset{\mathbb{X},\mathbb{L}}{\arg\min} \sum_{i=0}^{|\mathbb{B}|} dl_{L_i}(X_i). \tag{1}$$

The function $dl_{L_i}(X_i)$ calculates the description length of a text segment X_i through the use of a language model for L_i. Note that the actual total description length must also include an additional term, $\log_2 |X|$, giving information on the number of segments (with the maximum to be segmented

[1] In fact, our first motivation was to collect a certain amount of corpus data for non-major languages from Wikipedia.

by each character). Since this term is a common constant for all possible segmentations and the minimization of formula (1) is not affected by this term, we will ignore it.

The model defined by (1) is additive for X_i, so the following formula can be applied to search for language L_i given a segment X_i :

$$\hat{L}_i = \underset{L_i \in \mathcal{L}}{\arg\min}\, dl_{L_i}(X_i), \tag{2}$$

under the constraint that $L_i \neq L_{i-1}$ for $i \in \{1, \ldots |\mathbb{B}|\}$. The function dl can be further decomposed as follows to give the description length in an information-theoretic manner:

$$\begin{aligned} dl_{L_i}(X_i) = &- \log_2 P_{L_i}(X_i) \\ &+ \log_2 |X| + \log_2 |\mathcal{L}| + \gamma. \end{aligned} \tag{3}$$

Here, the first term corresponds to the code length of the text chunk X_i given a language model for L_i, which in fact corresponds to the cross-entropy of X_i for L_i multiplied by $|X_i|$. The remaining terms give the code lengths of the parameters used to describe the length of the first term: the second term corresponds to the segment location; the third term, to the identified language; and the fourth term, to the language model of language L_i. This fourth term will differ according to the language model type; moreover, its value can be further minimized through formula (2). Nevertheless, since we use a uniform amount of training data for every language, and since varying γ would prevent us from improving the efficiency of dynamic programming, as explained in §4, in this article we set γ to a constant obtained empirically.

Under this formulation, therefore, when detecting the language of a segment as in formula (2), the terms of formula (3) other than the first term will be constant: what counts is only the first term, similarly to much of the previous work explained in the following section. We thus perform language detection itself by minimizing the cross-entropy rather than the MDL. For segmentation, however, the constant terms function as overhead and also serve to prohibit excessive decomposition.

Next, after briefly introducing methods to calculate the first term of formula (3), we explain the solution to optimize the combinatorial problem of formula (1).

3 Calculation of Cross-Entropy

The first term of (3), $-\log_2 P_{L_i}(X_i)$, is the cross-entropy of X_i for L_i multiplied by $|X_i|$. Various methods for computing cross-entropy have been proposed, and these can be roughly classified into two types based on different methods of universal coding and the language model. For example, (Benedetto et al., 2002) and (Cilibrasi and Vitányi, 2005) used the universal coding approach, whereas (Teahan and Harper, 2001) and (Sibun and Reynar, 1996) were based on language modeling using PPM and Kullback-Leibler divergence, respectively.

In this section, we briefly introduce two methods previously studied by (Juola, 1997) and (Teahan, 2000) as representative of the two types, and we further explain a modification that we integrate into the final optimization problem. We tested several other coding methods, but they did not perform as well as these two methods.

3.1 Mean of Matching Statistics

(Farach et al., 1994) proposed a method to estimate the entropy, through a simplified version of the LZ algorithm (Ziv and Lempel, 1977), as follows. Given a text $X = x_1 x_2 \ldots x_i x_{i+1} \ldots$, Len_i is defined as the longest match length for two substrings $x_1 x_2 \ldots x_i$ and $x_{i+1} x_{i+2} \ldots$. In this article, we define the longest match for two strings A and B as the shortest prefix of string B that is not a substring of A. Letting the average of Len_i be $E\,[Len]$, Farach proved that $|E\,[Len] - \frac{\log_2 i}{H(X)}|$ probabilistically converges to zero as $i \to \infty$, where $H(X)$ indicates the entropy of X. Then, $H(X)$ is estimated as

$$\hat{H}(X) = \frac{\log_2 i}{E\,[Len]}.$$

(Juola, 1997) applied this method to estimate the cross-entropy of two given texts. For two strings $Y = y_1 y_2 \ldots y_{|Y|}$ and $X = x_1 x_2 \ldots x_{|X|}$, let $Len_i(Y)$ be the match length starting from x_i of X for Y[2]. Based on this formulation, the cross-entropy is approximately estimated as

$$\hat{J}_Y(X) = \frac{\log_2 |Y|}{E\,[Len_i(Y)]}.$$

[2]This is called a *matching statistics* value, which explains the subsection title.

Since formula (1) of §2 is based on adding the description length, it is important that the whole value be additive to enable efficient optimization (as will be explained in §4). We thus modified Juola's method as follows to make the length additive:

$$\hat{J}'_Y(X) = E\left[\frac{\log_2 |Y|}{Len_i(Y)}\right].$$

Although there is no mathematical guarantee that $\hat{J}_Y(X)$ or $\hat{J}'_Y(X)$ actually converges to the cross-entropy, our empirical tests showed a good estimate for both cases[3]. In this article, we use $\hat{J}'_Y(X)$ as a function to obtain the cross-entropy and for multiplication by $|X|$ in formula (3).

3.2 PPM

As a representative method for calculating the cross-entropy through statistical language modeling, we adopt prediction by partial matching (PPM), a language-based encoding method devised by (Cleary and Witten, 1984). It has the particular characteristic of using a variable n-gram length, unlike ordinary n-gram models[4]. It models the probability of a text X with a learning corpus Y as follows:

$$\begin{aligned} P_Y(X) &= P_Y(x_1 \dots x_{|X|}) \\ &= \prod_{t=1}^{|X|} P_Y(x_t | x_{t-1} \dots x_{\max(1,t-n)}), \end{aligned}$$

where n is a parameter of PPM, denoting the maximum length of the n-grams considered in the model[5]. The probability $P_Y(X)$ is estimated by *escape probabilities* favoring the longer sequences appearing in the learning corpus (Bell et al., 1990). The total code length of X is then estimated as $-\log P_Y(X)$. Since this value is additive and gives the total code length of X for language Y, we adopt this value in our approach.

4 Segmentation by Dynamic Programming

By applying the above methods, we propose a solution to formula (1) through dynamic programming. Considering the additive characteristic of the description length formulated previously as formula (1), we denote the minimized description length for a given text X simply as $DP(X)$, which can be decomposed recursively as follows[6]:

$$\begin{aligned} DP(X) = \min_{t\in\{0,\dots,|X|\}, L\in\mathcal{L}} &\{DP(x_0 \dots x_{t-1}) \\ &+ dl_L(x_t \dots x_{|X|})\}, \end{aligned} \quad (4)$$

In other words, the computation of $DP(X)$ is decomposed into obtaining the addition of two terms by searching through $t \in \{0, \dots, |X|\}$ and $L \in \mathcal{L}$. The first term gives the MDL for the first t characters of text X, while the second term, $dl_L(x_{t+1} \dots x_{|X|})$, gives the description length of the remaining characters under the language model for L.

We can straightforwardly implement this recursive computation through dynamic programming, by managing a table of size $|X| \times |\mathcal{L}|$. To fill a cell of this table, formula (4) suggests referring to $t \times |\mathcal{L}|$ cells and calculating the description length of the rest of the text for $\mathrm{O}(|X|-t)$ cells for each language. Since t ranges up to $|X|$, the brute-force computational complexity is $\mathrm{O}(|X|^3 \times |\mathcal{L}|^2)$.

The complexity can be greatly reduced, however, when the function dl is additive. First, the description length can be calculated from the previous result, decreasing $\mathrm{O}(|X| - t)$ to $\mathrm{O}(1)$ (to obtain the code length of an additional character). Second, the referred number of cells $t \times |\mathcal{L}|$ is in fact $U \times |\mathcal{L}|$, with $U \ll |X|$: for MMS, U can be proven to be $\mathrm{O}(\log |Y|)$, where $|Y|$ is the maximum length among the learning corpora; and for PPM, U corresponds to the maximum length of an n-gram. Third, this factor $U \times |\mathcal{L}|$ can be further decreased to $U \times 2$, since it suffices to possess the results for the two[7] best languages in computing the first term of (4). Consequently, the complexity decreases to $\mathrm{O}(U \times |X| \times |\mathcal{L}|)$.

[3]This modification means that the original $\hat{J}_Y(X)$ is obtained through the harmonic mean, with Len obtained through the arithmetic mean, whereas $\hat{J}'_Y(X)$ is obtained through the arithmetic mean with Len as the harmonic mean.

[4]In the context of NLP, this is known as Witten-Bell smoothing.

[5]In the experiments reported here, n is set to 5 throughout.

[6]This formula can be used directly to generate a set $\mathbb{L}$ in which all adjacent elements differ. The formula can also be used to generate segments for which some adjacent languages coincide and then further to generate $\mathbb{L}$ through post-processing by concatenating segments of the same language.

[7]This number means the two best scores for different languages, which is required to obtain $\mathbb{L}$ directly: in addition to the best score, if the language of the best coincides with L in formula (4), then the second best is also needed. If segments are subjected to post-processing, this value can be one.

Table 1: Number of languages for each writing system

character kinds	UDHR	Wiki
Latin	260	158
Cyrillic	12	20
Devanagari	0	8
Arabic	1	6
Other	4	30

5 Experimental Setting

5.1 Monolingual Texts (Training / Test Data)

In this work, monolingual texts were used both for training the cross-entropy computation and as test data for cross-validation: the training data does not contain any test data at all. Monolingual texts were also used to build multilingual texts, as explained in the following subsection.

Texts were collected from the World Wide Web and consisted of two sets. The first data set consisted of texts from the Universal Declaration of Human Rights (UDHR)[8]. We consider UDHR the most suitable text source for our purpose, since the content of every monolingual text in the declaration is unique. Moreover, each text has the tendency to maximally use its own language and avoid vocabulary from other languages. Therefore, UDHR-derived results can be considered to provide an empirical upper bound on our formulation. The set $\mathcal{L}$ consists of 277 languages , and the texts consist of around 10,000 characters on average.

The second data set was Wikipedia data from Wikipedia Downloads[9], denoted as "Wiki" in the following discussion. We automatically assembled the data through the following steps. First, tags in the Wikipedia texts were removed. Second, short lines were removed since they typically are not sentences. Third, the amount of data was set to 10,000 characters for every language, in correspondence with the size of the UDHR texts. Note that there is a limit to the complete cleansing of data. After these steps, the set $\mathcal{L}$ contained 222 languages with sufficient data for the experiments.

Many languages adopt writing systems other than the Latin alphabet. The numbers of languages for various representative writing systems are listed in Table 1 for both UDHR and Wiki, while the Appendix at the end of the article lists the actual languages. Note that in this article, a character means a Unicode character throughout, which differs from a character rendered in block form for some writing systems.

To evaluate language identification for monolingual texts, as will be reported in §6.1, we conducted five-times cross-validation separately for both data sets. We present the results in terms of the average accuracy $A_{\mathbb{L}}$, the ratio of the number of texts with a correctly identified language to $|\mathcal{L}|$.

5.2 Multilingual Texts (Test Data)

Multilingual texts were needed only to test the performance of the proposed method. In other words, we trained the model only through monolingual data, as mentioned above. This differs from the most similar previous study (Teahan, 2000), which required multilingual learning data.

The multilingual texts were generated artificially, since multilingual texts taken directly from the web have other issues besides segmentation. First, proper nouns in multilingual texts complicate the final judgment of language and segment borders. In practical application, therefore, texts for segmentation must be preprocessed by named entity recognition, which is beyond the scope of this work. Second, the sizes of text portions in multilingual web texts differ greatly, which would make it difficult to evaluate the overall performance of the proposed method in a uniform manner.

Consequently, we artificially generated two kinds of test sets from a monolingual corpus. The first is a set of multilingual texts, denoted as $Test_1$, such that each text is the conjunction of two portions in different languages. Here, the experiment is focused on segment border detection, which must segment the text into two parts, provided that there are two languages. $Test_1$ includes test data for all language pairs, obtained by five-times cross-validation, giving $25 \times |\mathcal{L}| \times (|\mathcal{L}| - 1)$ multilingual texts. Each portion of text for a single language consists of 100 characters taken from a random location within the test data.

The second kind of test set is a set of multilingual texts, denoted as $Test_2$, each consisting of k segments in different languages. For the experiment, k is not given to the procedure, and the task is to obtain k as well as $\mathbb{B}$ and $\mathbb{L}$ through recursion. $Test_2$

[8] http://www.ohchr.org/EN/UDHR/Pages/Introduction.aspx

[9] http://download.wikimedia.org/

was generated through the following steps:

1. Choose k from among 1,...,5.
2. Choose k languages randomly from $\mathcal{L}$, where some of the k languages can overlap.
3. Perform five-times cross-validation on the texts of all languages. Choose a text length randomly from {40,80,120,160}, and randomly select this many characters from the test data.
4. Shuffle the k languages and concatenate the text portions in the resultant order.

For this $Test_2$ data set, every plot in the graphs shown in §6.2 was obtained by randomly averaging 1,000 tests.

By default, the possibility of segmentation is considered at every character offset in a text, which provides a *lower bound* for the proposed method. Although language change within the middle of a word does occur in real multilingual documents, it might seem more realistic to consider language change at word borders. Therefore, in addition to choosing $\mathbb{B}$ from $\{1, \ldots, |X|\}$, we also tested our approach under the constraint of choosing borders from *bordering locations*, which are the locations of spaces. In this case, $\mathbb{B}$ is chosen from this subset of $\{1, \ldots, |X|\}$, and, in step 3 above, text portions are generated so as to end at these bordering locations.

Given a multilingual text, we evaluate the outputs $\mathbb{B}$ and $\mathbb{L}$ through the following scores:

$P_{\mathbb{B}}$/$R_{\mathbb{B}}$: Precision/recall of the borders detected (i.e., the correct borders detected, divided by the detected/correct border).

$P_{\mathbb{L}}$/$R_{\mathbb{L}}$: Precision/recall of the languages detected (i.e., the correct languages detected, divided by the detected/correct language).

Ps and Rs are obtained by changing the parameter γ given in formula (3), which ranges over 1,2,4,...,256 bits. In addition, we verify the speed, i.e., the average time required for processing a text.

Although there are web pages consisting of texts in more than 2 languages, we rarely see a web page containing 5 languages at the same time. Therefore, $Test_1$ reflects the most important case of 2 languages only, whereas $Test_2$ reflects the case of multiple languages to demonstrate the general potential of the proposed approach.

The experiment reported here might seem like a case of over-specification, since all languages are considered equally likely to appear. Since our motivation has been to eliminate a portion in a major language from the text, there could be a formulation specific to the problem. We consider it trivial, however, to specify such a narrow problem within our formulation, and it will lead to higher performance than that of the reported results, in any case. Therefore, we believe that our general formulation and experiment show the broadest potential of our approach to solving this problem.

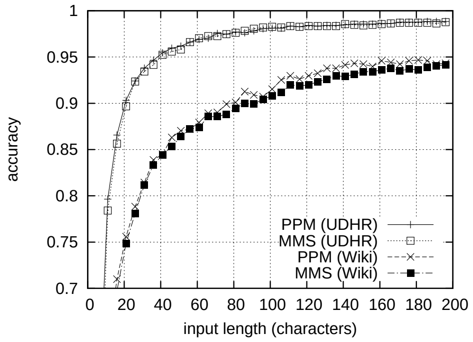

Figure 1: Accuracy of language identification for monolingual texts

6 Experimental Results

6.1 Language Identification Performance

We first show the performance of language identification using formula (2), which is used as the component of the text segmentation by language. Figure 1 shows the results for language identification of monolingual texts with the UDHR and Wiki test data. The horizontal axis indicates the size of the input text in characters, the vertical axis indicates the accuracy $A_{\mathbb{L}}$, and the graph contains four plots[10] for MMS and PPM for each set of data.

Overall, all plots rise quickly despite the severe conditions of a large number of languages (over 200), a small amount of input data, and a small amount of learning data. The results show that language identification through cross-entropy is promising.

Two further global tendencies can be seen. First, the performance was higher for UDHR than for Wiki. This is natural, since the content of Wikipedia is far broader than that of UDHR. In the case of UDHR, when the test data had a length of 40 characters, the accuracy was over 95% for both the PPM and the MMS methods. Second, PPM achieved

[10]The results for PPM and MMS for UDHR are almost the same, so the graph appears to contain only three plots.

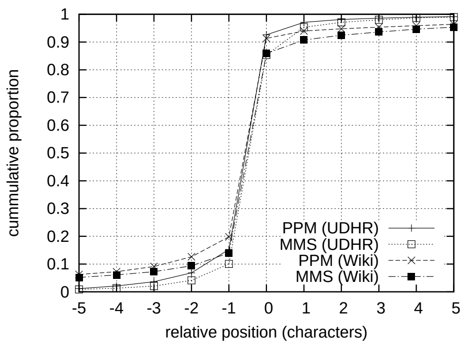

Figure 2: Cumulative distribution of segment borders

slightly better performance than did MMS. When the test data amounted to 100 characters, PPM achieved language identification with accuracy of about 91.4%. For MMS, the identification accuracy was a little less significant and was about 90.9% even with 100 characters of test data.

The amount of learning data seemed sufficient for both cases, with around 8,000 characters. In fact, we conducted tests with larger amounts of learning data and found a faster rise with respect to the input length, but the maximum possible accuracy did not show any significant increase.

Errors resulted from either noise or mistakes due to the language family. The Wikipedia test data was noisy, as mentioned in §5.1. As for language family errors, the test data includes many similar languages that are difficult even for humans to correctly judge. For example, Indonesian and Malay, Picard and Walloon, and Norwegian Bokmål and Nynorsk are all pairs representative of such confusion.

Overall, the language identification performance seems sufficient to justify its application to our main problem of text segmentation by language.

6.2 Text Segmentation by Language

First, we report the results obtained using the $Test_1$ data set. Figure 2 shows the cumulative distribution obtained for segment border detection. The horizontal axis indicates the relative location by character with respect to the correct border at zero, and the vertical axis indicates the cumulative proportion of texts whose border is detected at that relative point. The figure shows four plots for all combinations of the two data sets and the two methods. Note that segment borders are judged by characters and *not* by bordering locations, as explained in §5.2.

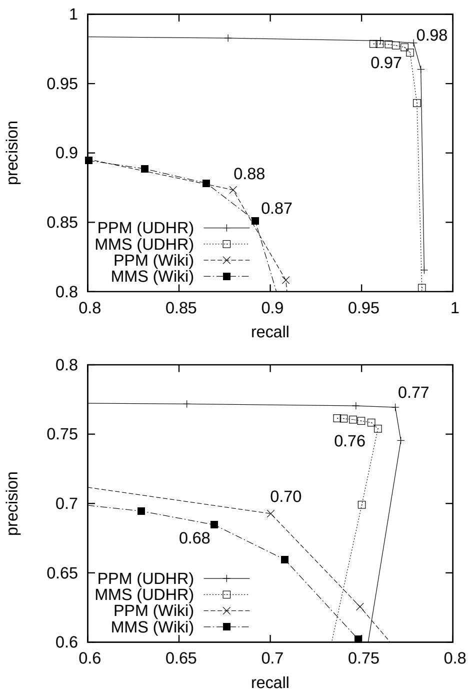

Figure 3: $P_{\mathbb{L}}/R_{\mathbb{L}}$ (language, upper graph) and $P_{\mathbb{B}}/R_{\mathbb{B}}$ (border, lower graph) results, where borders were taken from *any* character offset

Since the plots rise sharply at the middle of the horizontal axis, the borders were detected at or very near the correct place in many cases.

Next, we examine the results for $Test_2$. Figure 3 shows the two precision/recall graphs for language identification (upper graph) and segment border detection (lower graph), where borders were taken from any character offset. In each graph, the horizontal axis indicates precision and the vertical axis indicates recall. The numbers appearing in each figure are the maximum F-score values for each method and data set combination. As can be seen from these numbers, the language identification performance was high. Since the text portion size was chosen from among the values 40, 80, 120, or 160, the performance is comprehensible from the results shown in §6.1. Note also that PPM performed slightly better than did MMS.

For segment border performance (lower graph), however, the results were limited. The main reason for this is that both MMS and PPM tend to detect a border one character earlier than the correct location, as was seen in Figure 2. At the same time, much of the test data contains unrealistic borders

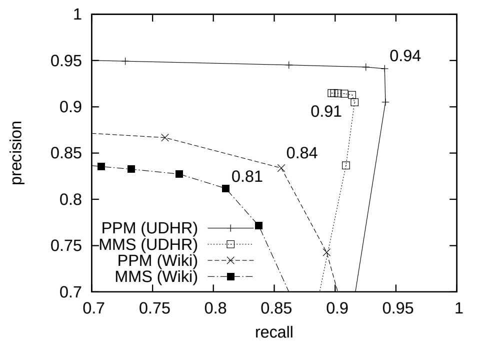

Figure 4: $P_{\mathbb{B}}/R_{\mathbb{B}}$, where borders were limited to spaces

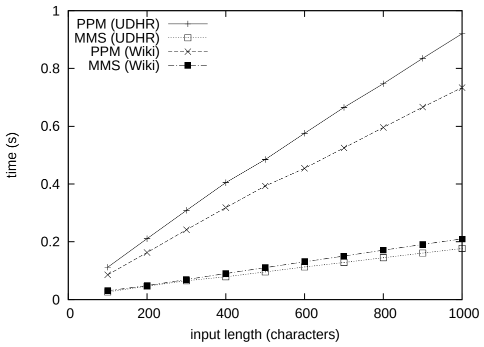

Figure 5: Average processing speed for a text

within a word, since the data was generated by concatenating two text portions with random borders. Therefore, we repeated the experiment with $Test_2$ under the constraint that a segment border could occur only at a bordering location, as explained in §5.2. The results with this constraint were significantly better, as shown in Figure 4. The best result was for UDHR with PPM at 0.94[11]. We could also observe how PPM performed better at detecting borders in this case. In actual application, it would be possible to improve performance by relaxing the procedural conditions, such as by decreasing the number of language possibilities.

In this experiment for $Test_2$, k ranged from 1 to 5, but the performance was not affected by the size of k. When the F-score was examined with respect to k, it remained almost equal to k in all cases. This shows how each recursion of formula (4) works almost independently, having segmentation and language identification functions that are both robust.

Lastly, we examine the speed of our method. Since $|\mathcal{L}|$ is constant throughout the comparison, the time should increase linearly with respect to the input length $|X|$, with increasing k having no effect. Figure 5 shows the speed for $Test_2$ processing, with the horizontal axis indicating the input length and the vertical axis indicating the processing time. Here, all character offsets were taken into consideration, and the processing was done on a machine with a Xeon5650 2.66-GHz CPU. The results confirm that the complexity increased linearly with respect to the input length. When the text size became as large as several thousand characters, the processing time became as long as a second. This time could be significantly decreased by introducing constraints on the bordering locations and languages.

7 Conclusion

This article has presented a method for segmenting a multilingual text into segments, each in a different language. This task could serve for preprocessing of multilingual texts before applying language-specific analysis to each text. Moreover, the proposed method could be used to generate corpora in a variety of languages, since many texts in minor languages tend to contain chunks in a major language.

The segmentation task was modeled as an optimization problem of finding the best segment and language sequences to minimize the description length of a given text. An actual procedure for obtaining an optimal result through dynamic programming was proposed. Furthermore, we showed a way to decrease the computational complexity substantially, with each of our two methods having linear complexity in the input length.

Various empirical results were shown for language identification and segmentation. Overall, when segmenting a text with up to five random portions of different languages, where each portion consisted of 40 to 120 characters, the best F-scores for language identification and segmentation were 0.98 and 0.94, respectively.

For our future work, details of the methods must be worked out. In general, the proposed approach could be further applied to the actual needs of preprocessing and to generating corpora of minor languages.

[11] The language identification accuracy slightly increased as well, by 0.002.

References

Beatrice Alex, Amit Dubey, and Frank Keller. 2007. Using foreign inclusion detection to improve parsing performance. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 151–160.

Beatrice Alex. 2005. An unsupervised system for identifying english inclusions in german text. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics, Student Research Workshop*, pages 133–138.

T.C. Bell, J.G. Cleary, and I. H. Witten. 1990. *Text Compression*. Prentice Hall.

Dario Benedetto, Emanuele Caglioti, and Vittorio Loreto. 2002. Language trees and zipping. *Physical Review Letters*, 88(4).

Rudi Cilibrasi and Paul Vitányi. 2005. Clustering by compression. *IEEE Transactions on Information Theory*, 51(4):1523–1545.

John G. Cleary and Ian H. Witten. 1984. Data compression using adaptive coding and partial string matching. *IEEE Transactions on Communications*, 32:396–402.

Martin Farach, Michiel Noordewier, Serap Savari, Larry Shepp, Abraham J. Wyner, and Jacob Ziv. 1994. On the entropy of dna: Algorithms and measurements based on memory and rapid convergence. In *Proceedings of the Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 48–57.

Gregory Grefenstette. 1995. Comparing two language identification schemes. In *Proceedings of 3rd International Conference on Statistical Analysis of Textual Data*, pages 263–268.

Marti A. Hearst. 1997. Texttiling: Segmenting text into multi-paragraph subtopic passages. *Computational Linguistics*, 23(1):33–64.

Patrick Juola. 1997. What can we do with small corpora? document categorization via cross-entropy. In *Proceedings of an Interdisciplinary Workshop on Similarity and Categorization*.

Gen-itiro Kikui. 1996. Identifying the coding system and language of on-line documents on the internet. In *Proceedings of 16th International Conference on Computational Linguistics*, pages 652–657.

Casanai Kruengkrai, Prapass Srichaivattana, Virach Sornlertlamvanich, and Hitoshi Isahara. 2005. Language identification based on string kernels. In *Proceedings of the 5th International Symposium on Communications and Information Technologies*, pages 926–929.

Penelope Sibun and Jeffrey C. Reynar. 1996. Language identification: Examining the issues. In *Proceedings of 5th Symposium on Document Analysis and Information Retrieval*, pages 125–135.

William J. Teahan and David J. Harper. 2001. Using compression-based language models for text categorization. In *Proceedings of the Workshop on Language Modeling and Information Retrieval*, pages 83–88.

William John Teahan. 2000. Text classification and segmentation using minimum cross-entropy. In *RIAO*, pages 943–961.

Jacob Ziv and Abraham Lempel. 1977. A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory*, 23(3):337–343.

Appendix

This Appendix lists all the languages contained in our data sets, as summarized in Table 1.

For UDHR

Latin

Achinese, Achuar-Shiwiar, Adangme, Afrikaans, Aguaruna, Aja, Akuapem Akan, Akurio, Amahuaca, Amarakaeri, Ambo-Pasco Quechua, Arabela, Arequipa-La Unión Quechua, Arpitan, Asante Akan, Asháninka, Ashéninka Pajonal, Asturian, Auvergnat Occitan, Ayacucho Quechua, Aymara, Baatonum, Balinese, Bambara, Baoulé, Basque, Bemba, Beti, Bikol, Bini, Bislama, Bokmål Norwegian, Bora, Bosnian, Breton, Buginese, Cajamarca Quechua, Calderón Highland Quichua, Candoshi-Shapra, Caquinte, Cashibo-Cacataibo, Cashinahua, Catalan, Cebuano, Central Kanuri, Central Mazahua, Central Nahuatl, Chamorro, Chamula Tzotzil, Chayahuita, Chickasaw, Chiga, Chokwe, Chuanqiandian Cluster Miao, Chuukese, Corsican, Cusco Quechua, Czech, Dagbani, Danish, Dendi, Ditammari, Dutch, Eastern Maninkakan, Emiliano-Romagnolo, English, Esperanto, Estonian, Ewe, Falam Chin, Fanti, Faroese, Fijian, Filipino, Finnish, Fon, French, Friulian, Ga, Gagauz, Galician, Ganda, Garifuna, Gen, German, Gheg Albanian, Gonja, Guarani, Güilá Zapotec, Haitian Creole, Haitian Creole (popular), Haka Chin, Hani, Hausa, Hawaiian, Hiligaynon, Huamalíes-Dos de Mayo Huánuco Quechua, Huautla Mazatec, Huaylas Ancash Quechua, Hungarian, Ibibio, Icelandic, Ido, Igbo, Iloko, Indonesian, Interlingua, Irish, Italian, Javanese, Jola-Fonyi, K'iche', Kabiyè, Kabuverdianu, Kalaallisut, Kaonde, Kaqchikel, Kasem, Kekchí, Kimbundu, Kinyarwanda, Kituba, Konzo, Kpelle, Krio, Kurdish, Lamnso', Languedocien Occitan, Latin, Latvian, Lingala, Lithuanian, Lozi, Luba-Lulua, Lunda, Luvale, Luxembourgish, Madurese, Makhuwa, Makonde, Malagasy, Maltese, Mam, Maori, Mapudungun, Margos-Yarowilca-Lauricocha Quechua, Marshallese, Mba, Mende, Metlatónoc Mixtec, Mezquital Otomi, Mi'kmaq, Miahuatlán Zapotec, Minangkabau, Mossi, Mozarabic, Murui Huitoto, Mískito, Ndonga, Nigerian Pidgin, Nomatsiguenga, North Junín Quechua, Northeastern Dinka, Northern Conchucos Ancash Quechua, Northern Qiandong Miao, Northern Sami, Northern Kurdish, Nyamwezi, Nyanja, Nyemba, Nynorsk Norwegian, Nzima, Ojitláan Chinantec, Oromo, Palauan, Pampanga, Papantla Totonac, Pedi, Picard, Pichis Ashéninka, Pijin, Pipil, Pohnpeian, Polish, Portuguese, Pulaar, Purepecha, Páez, Quechua, Rarotongan, Romanian, Romansh, Romany, Rundi, Salinan, Samoan, San Luís Potosí Huastec, Sango, Sardinian, Scots, Scottish Gaelic, Serbian, Serer, Seselwa Creole French, Sharanahua, Shipibo-Conibo, Shona, Slovak, Somali, Soninke, South Ndebele, Southern Dagaare, Southern Qiandong Miao, Southern Sotho, Spanish, Standard Malay, Sukuma, Sundanese, Susu, Swahili, Swati, Swedish, Sãotomense, Tahitian, Tedim Chin, Tetum, Tidikelt Tamazight, Timne, Tiv, Toba, Tojolabal, Tok Pisin, Tonga (Tonga Islands), Tonga (Zambia), Tsonga, Tswana, Turkish, Tzeltal, Umbundu, Upper Sorbian, Urarina, Uzbek, Veracruz Huastec, Vili, Vlax Romani, Walloon, Waray, Wayuu, Welsh, Western Frisian, Wolof, Xhosa, Yagua, Yanesha', Yao, Yapese, Yoruba, Yucateco, Zhuang, Zulu

Cyrillic

Abkhazian, Belarusian, Bosnian, Bulgarian, Kazakh, Macedonian, Ossetian, Russian, Serbian, Tuvinian, Ukrainian, Yakut

Arabic

Standard Arabic

Other

Japanese, Korean, Mandarin Chinese, Modern Greek

For Wiki

Latin

Afrikaans, Albanian, Aragonese, Aromanian, Arpitan, Asturian, Aymara, Azerbaijani, Bambara, Banyumasan, Basque, Bavarian, Bislama, Bosnian, Breton, Català, Cebuano, Central Bikol, Chavacano, Cornish, Corsican, Crimean Tatar, Croatian, Czech, Danish, Dimli, Dutch, Dutch Low Saxon, Emiliano-Romagnolo, English, Esperanto, Estonian, Ewe, Extremaduran, Faroese, Fiji Hindi, Finnish, French, Friulian, Galician, German, Gilaki, Gothic, Guarani, Hai//om, Haitian, Hakka Chinese, Hawaiian, Hungarian, Icelandic, Ido, Igbo, Iloko, Indonesian, Interlingua, Interlingue, Irish, Italian, Javanese, Kabyle, Kalaallisut, Kara-Kalpak, Kashmiri, Kashubian, Kongo, Korean, Kurdish, Ladino, Latin, Latvian, Ligurian, Limburgan, Lingala, Lithuanian, Lojban, Lombard, Low German, Lower Sorbian, Luxembourgish, Malagasy, Malay, Maltese, Manx, Maori, Mazanderani, Min Dong Chinese, Min Nan Chinese, Nahuatl, Narom, Navajo, Neapolitan, Northern Sami, Norwegian, Norwegian Nynorsk, Novial, Occitan, Old English, Pampanga, Pangasinan, Panjabi, Papiamento, Pennsylvania German, Piemontese, Pitcairn-Norfolk, Polish, Portuguese, Pushto, Quechua, Romanian, Romansh, Samoan, Samogitian Lithuanian, Sardinian, Saterfriesisch, Scots, Scottish Gaelic, Serbo-Croatian, Sicilian, Silesian, Slovak, Slovenian, Somali, Spanish, Sranan Tongo, Sundanese, Swahili, Swati, Swedish, Tagalog, Tahitian, Tarantino Sicilian, Tatar, Tetum, Tok Pisin, Tonga (Tonga Islands), Tosk Albanian, Tsonga, Tswana, Turkish, Turkmen, Uighur, Upper Sorbian, Uzbek, Venda, Venetian, Vietnamese, Vlaams, Vlax Romani, Volapük, Võro, Walloon, Waray, Welsh, Western Frisian, Wolof, Yoruba, Zeeuws, Zulu

Cyrillic

Abkhazian, Bashkir, Belarusian, Bulgarian, Chuvash, Erzya, Kazakh, Kirghiz, Macedonian, Moksha, Moldovan, Mongolian, Old Belarusian, Ossetian, Russian, Serbian, Tajik, Udmurt, Ukrainian, Yakut

Arabic

Arabic, Egyptian Arabic, Gilaki, Mazanderani, Persian, Pushto, Uighur, Urdu

Devanagari

Bihari, Hindi, Marathi, Nepali, Newari, Sanskrit

Other

Amharic, Armenian, Assamese, Bengali, Bishnupriya, Burmese, Central Khmer, Chinese, Classical Chinese, Dhivehi, Gan Chinese, Georgian, Gothic, Gujarati, Hebrew, Japanese, Kannada, Lao, Malayalam, Modern Greek, Official Aramaic, Panjabi, Sinhala, Tamil, Telugu, Thai, Tibetan, Wu Chinese, Yiddish, Yue Chinese