Yet Another Language Identifier

Martin Majliš
Charles University in Prague
Institute of Formal and Applied Linguistics
Faculty of Mathematics and Physics
majlis@ufal.mff.cuni.cz

Abstract

Language identification of written text has been studied for several decades. Despite this fact, most of the research is focused on a few most spoken languages, whereas the minor ones are ignored. The identification of a larger number of languages brings new difficulties that do not occur for a few languages. These difficulties are causing decreased accuracy. The objective of this paper is to investigate the sources of such degradation. In order to isolate the impact of individual factors, 5 different algorithms and 3 different number of languages are used. The Support Vector Machine algorithm achieved an accuracy of 98% for 90 languages and the YALI algorithm based on a scoring function had an accuracy of 95.4%. The YALI algorithm has slightly lower accuracy but classifies around 17 times faster and its training is more than 4000 times faster.

Three different data sets with various number of languages and sample sizes were prepared to overcome the lack of standardized data sets. These data sets are now publicly available.

1 Introduction

The task of language identification has been studied for several decades, but most of the literature is about identifying spoken language[1]. This is mainly because language identification of written form is considered an easier task, because it does not contain such variability as the spoken form, such as dialects or emotions.

Language identification is used in many NLP tasks and in some of them simple rules[2] are often good enough. But for many other applications, such as web crawling, question answering or multilingual documents processing, more sophisticated approaches need to be used.

This paper first discusses previous work in Section 2, and then presents possible hypothesis for decreased accuracy when a larger number of languages is identified in Section 3. Data used for experiments is described in Section 4, along with methods used in experiments for language identification in Section 5. Results for all methods as well as comparison with other systems is presented in Section 6.

2 Related Work

The methods used in language identification have changed significantly during the last decades. In the late sixties, Gold (1967) examined language identification as a task in automata theory. In the seventies, Leonard and Doddington (1974) was able to recognize five different languages, and in the eighties, Beesley (1988) suggested using cryptoanalytic techniques.

Later on, Cavnar and Trenkle (1994) introduced their algorithm with a sliding window over a set of characters. A list of the 300 most common n-grams for n in 1..5 is created during training for each training document. To classify a new document, they constructed a list of the 300 most common n-grams and compared n-grams position with the testing lists. The list with the least differences is the most similar one and new document is likely to be written in same language.

[1] http://speech.inesc.pt/~dcaseiro/html/bibliografia.html

[2] http://en.wikipedia.org/wiki/Wikipedia:Language_recognition_chart

Proceedings of the EACL 2012 Student Research Workshop, pages 46–54,
Avignon, France, 26 April 2012.

They classified 3478 samples in 14 languages from a newsgroup and reported an achieved accuracy of 99.8%. This influenced many researches that were trying different heuristics for selecting n-grams, such as Martins and Silva (2005) which achieved an accuracy of 91.25% for 12 languages, or Hayati (2004) with 93.9% for 11 languages.

Sibun and Reynar (1996) introduced a method for language detection based on relative entropy, a popular measure also known as Kullback-Leibler distance. Relative entropy is a useful measure of the similarity between probability distributions. She used texts in 18 languages from the European Corpus Initiative CD-ROM. She achieved a 100% accuracy for bigrams.

In recent years, standard classification techniques such as support vector machines also became popular and many researchers used them Kruengkrai et al. (2005) or Baldwin and Lui (2010) for identifying languages.

Nowadays, language recognition is considered as an elementary NLP task[3] which can be used for educational purposes. McNamee (2005) used single documents for each language from project Gutenberg in 10 European languages. He preprocessed the training documents – the texts were lower-cased, accent marks were retained. Then, he computed a so-called profile of each language. Each profile consisted of a percentage of the training data attributed to each observed word. For testing, he used 1000 sentences per language from the Euro-parliament collection. To classify a new document, the same preprocessing was done and inner product based on the words in the document and the 1000 most common words in each language was computed. Performance varied from 80.0% for Portuguese to 99.5% for German.

Some researches such as Hughes et al. (2006) or Grothe et al. (2008) focused in their papers on the comparison of different approaches to language identification and also proposed new goals in that field, such as as minority languages or languages written non-Roman script.

Most of the researches in the past identified mostly up to twenty languages but in recent years, language identification of minority languages became the focus of Baldwin and Lui (2010), Choong et al. (2011), and Majliš (2012). All of them observed that the task became much harder for larger numbers of languages and accuracy of the system dropped.

[3] http://alias-i.com/lingpipe/demos/tutorial/langid/read-me.html

3 Hypothesis

The accuracy degradation with a larger number of languages in the language identification system may have many reasons. This section discusses these reasons and suggests how to isolate them. In some hypotheses, charts involving data from the W2C Wiki Corpus are used, which are introduced in Section 4.

3.1 Training Data Size

In many NLP applications, size of the available training data influences overall performance of the system, as was shown by Halevy et al. (2009).

To investigate the influence of training data size, we decided to use two different sizes of training data – 1 MB and 4 MB. If the drop in accuracy is caused by the lack of training data, then all methods used on 4 MB should outperform the same methods used on 1 MB of data.

3.2 Language Diversity

The increasing number of languages recognised by the system decreases language diversity. This may be another reason for the observed drop in the accuracy. We used information about language classes from the Ethnologue website (Lewis, 2009). The number of different language classes is depicted in Figure 1. *Class 1* represents the most distinguishable classes, such as Indo-European vs. Japonic, while *Class 2* represents finer classification, such as Indo-European, Germanic vs. Indo-European, Italic.

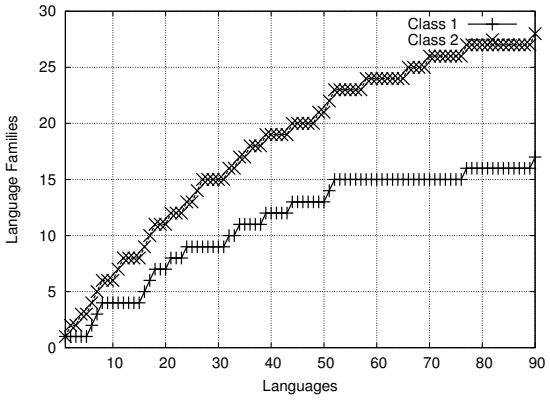


Figure 1: Language diversity on Wikipedia. Languages are sorted according to their text corpus size.

The first 52 languages belong to 15 different *Class 1* classes and the number of classes does not

change until the 77th language, when the Swahili language from class Niger-Congo appears.

3.3 Scalability

Another issue with increasing number of languages is the scalability of used methods. There are several pitfalls for machine learning algorithms – a) many languages may require many features which may lead to failures caused by curse-of-dimensionality, b) differences in languages may shrink, so the classifier will be forced to learn minor differences and will lose its ability to generalise, and become overfitted, and c) the classifier may internally use only binary classifiers which may lead up to quadratic complexity (Dimitriadou et al., 2011).

4 Data Sets

For our experiments, we decided to use the W2C Wiki Corpus (Majliš, 2012) which contains articles from Wikipedia. The total size of all texts was 8 GB and available material for various languages differed significantly, as is displayed in Figure 2.

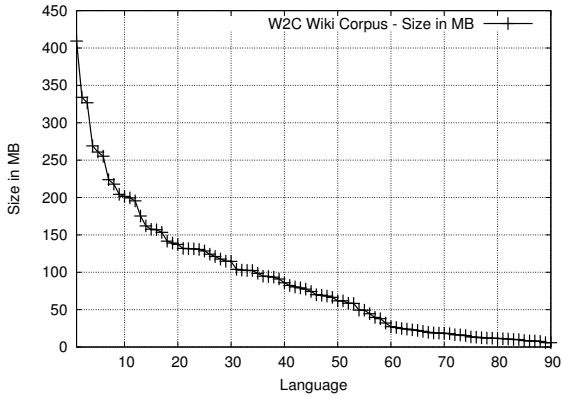


Figure 2: Available data in the W2C Wiki Corpus. Languages are sorted according to their size in the corpus.

We used this corpus to prepare 3 different data sets. We used one of them for testing hypothesis presented in the previous section and the remaining two for comparison with other systems. These data sets contain samples of length approximately 30, 140, and 1000 bytes. The sample of length 30 represents image caption or book title, the sample of length 140 represents tweet or user comment, and sample of length 1000 represents newspaper article.

All datasets are available at `http://ufal.mff.cuni.cz/~majlis/yali/`.

4.1 Long

The main purpose of this data set (*yali-dataset-long*) was testing hypothesis described in the previous section.

To investigate the drop, we intended to cover around 100 languages, but the amount of available data limited us. For example, the 80th language has 12 MB, whereas the 90th has 6 MB and tbe 100th has only 1 MB of text. To investigate the hypothesis of the influence of training data size, we decided to build a 1 MB and 4 MB corpus for each language, where the 1 MB corpus is a subset of the 4 MB one.

Then, we divided the corpus for each language into chunks with 1000 bytes of text, so we gained 1000 and 4000 chunks respectively. These chunks were divided into training and testing sets in a 90:10 ratio, thus we had 900 and 3600 training chunks, respectively, and 100 and 400 testing chunks respectively.

To reduce the risk that the training and testing are influenced by the position from which they were taken (the beginning or the end of the corpus), we decided to use every 10th sentence as a testing one and use the remaining ones for training.

Then, we created an n-gram for n in 1..4 frequency list for each language, each corpus size. From each frequency list, we preserved only the first m = 100 most frequent n-grams. For example, from the raw frequency list – a: 5, b: 3, c: 1, d: 1, and m = 2, frequency list a: 5, b: 3 would be created. We used this n-grams as features for testing classifiers.

4.2 Small

The second data set (*yali-dataset-small*) was prepared for comparison with Google Translate[4] (GT). The GT is paid service capable of recognizing 50 different languages. This data set contains 50 samples of lengths 30 and 140 for 48 languages, so it contains 4,800 samples in total.

4.3 Standard

The purpose of the third data sets is comparison with other systems for language identification. This data set contains 700 samples of length 30, 140, and 1000 for 90 languages, so it contains in total 189,000 samples.

[4] `http://translate.google.com`

Size	L\N	1	2	3	4
1MB	30	177	1361	2075	2422
	60	182	1741	3183	4145
	90	186	1964	3943	5682
4MB	30	176	1359	2079	2418
	60	182	1755	3184	4125
	90	187	1998	3977	5719

Table 1: The number of unique *N*-grams in corpus *Size* with *L* languages. ($D^{(Size,L,n)}$)

5 Methods

To investigate the influence of the language diversity, we decided to use 3 different language counts – 30, 60, and 90 languages sorted according to their raw text size. For each corpus size ($cS \in \{1000, 4000\}$), language count ($lC \in \{30, 60, 90\}$), and n-gram size ($n \in \{1, 2, 3, 4\}$) we constructed a separate dictionary $D^{(cS,lC,n)}$ containing the first 100 most frequent n-grams for each language. The number of items in each dictionary is displayed in Table 1 and visualised for 1 MB corpus in Figure 3.

The dictionary sizes for 4 MB corpora were slightly higher when compared to 1 MB corpora, but surprisingly for 30 languages it was mostly opposite.

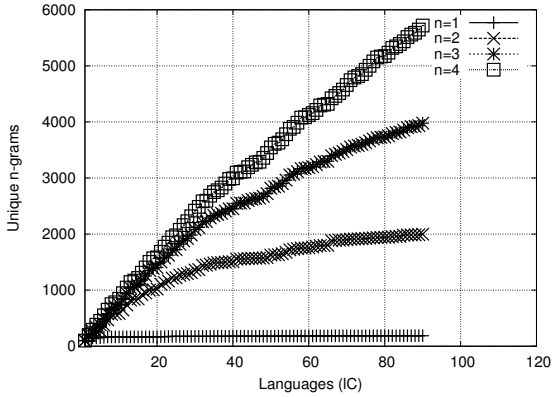


Figure 3: The number of unique n-grams in the dictionary $D^{(1000,lC,n)}$. Languages are sorted according to their text corpus size.

Then, we converted all texts into matrices in the following way. For each corpus size ($cS \in \{1000, 4000\}$), language count ($lC \in \{30, 60, 90\}$), and n-gram size ($n \in \{1, 2, 3, 4\}$) we constructed a training matrix $Tr^{(cS,lC,n)}$ and a testing matrix $Te^{(cS,lC,n)}$, where element on $Tr^{(cS,lC,n)}_{i,j}$ represents the number of occurrences of j-th n-gram from dictionary $D^{(cS,lC,n)}$ in training sample i, and $Tr^{(cS,lC,n)}_{i,0}$ represents language of that sample. The training matrix $Tr^{(cS,lC,n)}$ has dimension $(0.9 \cdot cS \cdot lC) \times (1 + \mid D^{(cS,lC,n)} \mid)$ and the testing matrix $Te^{(cS,lC,n)}$ has dimension $(0.1 \cdot cS \cdot lC) \times (1 + \mid D^{(cS,lC,n)} \mid)$.

For investigating the scalability of the different approaches to language identification, we decided to use five different methods. Three of them were based on standard classification algorithms and two of them were based on scoring function. For experimenting with the classification algorithms, we used R (2009) environment which contains many packages with machine learning algorithms[5], and for scoring functions we used Perl.

5.1 Support Vector Machine

The Suport Vector Machine (SVM) is a state of the art algorithm for classification. Hornik et al. (2006) compared four different implementations and concluded that Dimitriadou et al. (2011) implementation available in the package e1071 is the fastest one. We used SVM with sigmoid kernel, cost of constraints violation set to 10, and termination criterion set to 0.01.

5.2 Naive Bayes

The Naive Bayes classifier (NB) is a simple probabilistic classifier. We used Dimitriadou et al. (2011) implementation from the package e1071 with default arguments.

5.3 Regression Tree

Regression trees are implemented by Therneau et al. (2010) in the package rpart. We used it with default arguments.

5.4 W2C

The W2C algorithm is the same as was used by Majliš (2011). From the frequency list, probability is computed for each n-gram, which is used as a score in classification. The language with the highest score is the winning one. For example, from the raw frequency list – a: 5, b: 3, c: 1, d: 1, and m=2, the frequency list a: 5; b: 3, and computed scores – a: 0.5, b: 0.3 would be created.

[5] `http://cran.r-project.org/web/views/MachineLearning.html`

5.5 Yet Another Language Identifier

The Yet Another Language Identifier (YALI) algorithm is based on the W2C algorithm with two small modifications. The first is modification in n-gram score computation. The n-gram score is not based on its probability in raw data, but rather on its probability in the preserved frequency list. So for the numbers used in the W2C example, we would receive scores – a: 0.625, b: 0.375. The second modification is using rather byte n-grams instead of character n-grams.

6 Results & Discussion

At the beginning we used only data set *yali-dataset-long* to investigate the influence of various set-ups.

The accuracy of all experiments is presented in Table 2, and visualised in Figure 4 and Figure 5. These experiments also revealed that algorithms are strong in different situations. All classification techniques outperform all scoring functions on short n-grams and small amount of languages. However, with increasing n-gram length, their accuracy stagnated or even dropped. The increased number of languages is unmanageable for NB a RPART classifiers and their accuracy significantly decreased. On the other hand, the accuracy of scoring functions does not decrease so much with additional languages. The accuracy of the W2C algorithm decreased when greater training corpora was used or more languages were classified, whereas the YALI algorithm did not have these problems, but moreover its accuracy increased with greater training corpus.

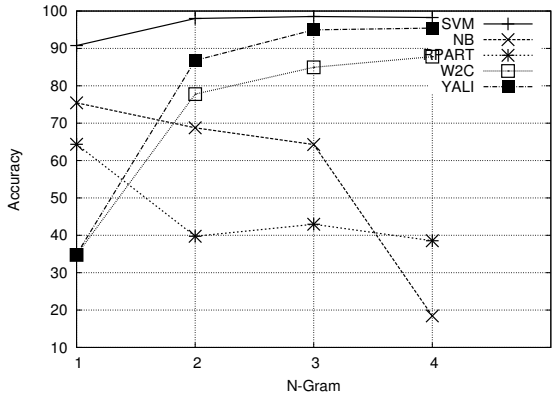


Figure 4: Accuracy for 90 languages and 1 MB corpus with respect to n-gram length.

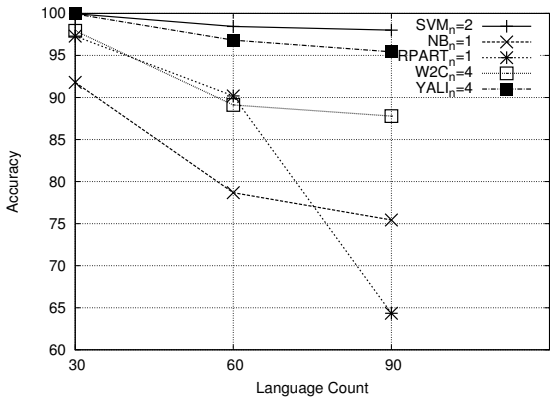


Figure 5: Accuracy for 1 MB corpus and the best n-gram length with respect to the number of languages.

The highest accuracy for all language amounts – 30, 60, 90 was achieved by the SVM with accuracies of 100%, 99%, and 98.5%, respectively, followed by the YALI algorithm with accuracies of 99.9%, 96.8%, and 95.4% respectively.

From the obtained results, it is possible to notice that 1 MB of text is sufficient for training language identifiers, but some algorithms achieved higher accuracy with more training material.

Our next focus was on the scalability of the used algorithms. Time required for training is presented in Table 3, and visualised in Figures 6 and 7.

The training of scoring functions required only loading dictionaries and therefore is extremely fast, whereas training classifiers required complicated computations. The scoring functions did not have any advantages, because all algorithms had to load all training examples, segment them, extract the most common n-grams, build dictionaries, and convert text to matrices as was described in Section 5.

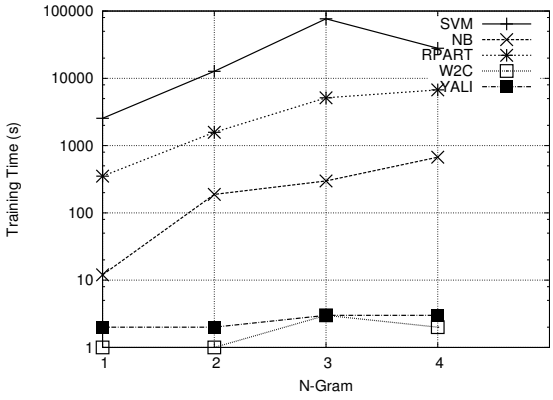


Figure 6: Training time for 90 languages and 1 MB corpus with respect to n-gram length.

N-Gram	L	1		2		3		4	
Method	S	1MB	4MB	1MB	4MB	1MB	4MB	1MB	4MB
SVM	30	96.3%	96.7%	100.0%	99.9%	100.0%	99.9%	99.9%	99.9%
	60	91.5%	92.3%	98.5%	98.5%	99.0%	99.0%	98.6%	98.5%
	90	90.8%	91.6%	98.0%	98.0%	98.5%	-	98.3%	-
NB	30	91.8%	94.2%	91.3%	90.9%	82.2%	93.3%	32.1%	59.9%
	60	78.7%	84.8%	70.6%	68.2%	71.7%	77.6%	25.7%	34.0%
	90	75.4%	82.7%	68.8%	66.5%	64.3%	71.0%	18.4%	17.5%
RPART	30	97.3%	96.7%	98.8%	98.6%	98.4%	97.8%	97.7%	97.4%
	60	90.2%	91.2%	67.3%	72.0%	67.2%	68.8%	65.5%	74.6%
	90	64.3%	55.9%	39.7%	39.6%	43.0%	44.0%	38.5%	39.6%
W2C	30	38.0%	38.6%	89.9%	91.0%	96.2%	96.5%	97.9%	98.1%
	60	34.7%	30.9%	83.0%	81.7%	86.0%	84.9%	89.1%	82.0%
	90	34.7%	30.9%	77.8%	77.6%	84.9%	83.4%	87.8%	82.7%
YALI	30	38.0%	38.6%	96.7%	96.2%	99.6%	99.5%	99.9%	99.8%
	60	35.0%	31.2%	86.1%	86.1%	95.7%	96.4%	96.8%	97.4%
	90	34.9%	31.1%	86.8%	87.8%	95.0%	95.6%	95.4%	96.1%

Table 2: Accuracy of classifiers for various corpora sizes, n-gram lengths, and language counts.

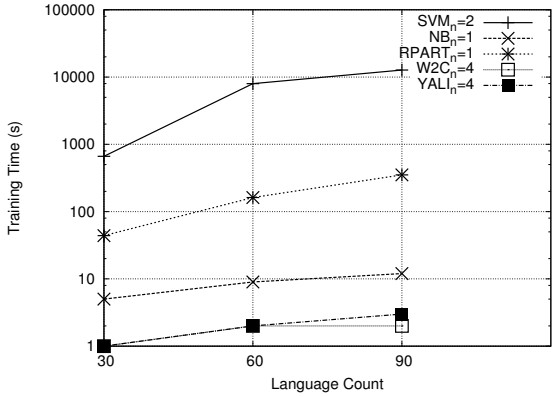


Figure 7: Training time for 1 MB corpus and the best n-gram length with respect to the number of languages.

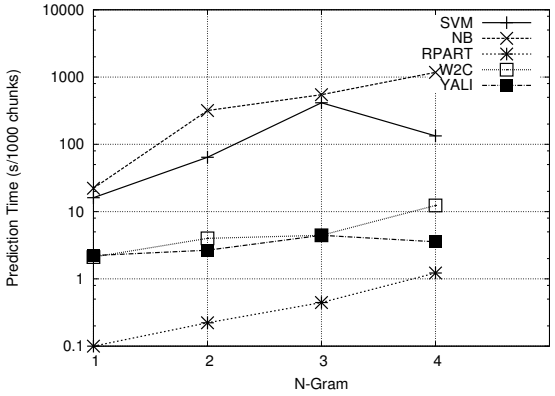
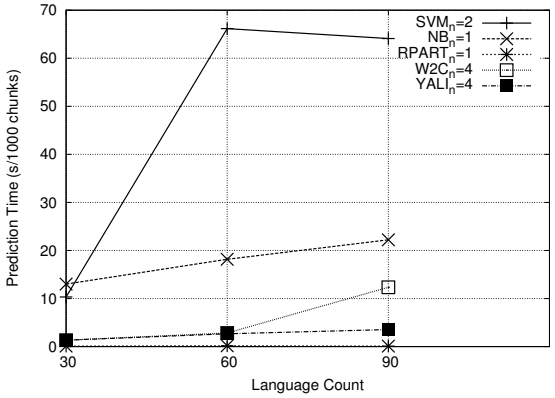


Figure 8: Prediction time for 90 languages and 1 MB corpus with respect to n-gram length.

Time required for training increased dramatically for SVM and RPART algorithms when the number of languages or the corpora size increased. It is possible to use the SVM only with unigrams or bigrams, because training on trigrams required 12 times more time for 60 languages compared with 30 languages. The SVM also had problems with increasing corpora sizes, because it took almost 10-times more time when the corpus size increased 4 times. Scoring functions scaled well and were by far the fastest ones. We terminated training the SVM on trigrams and quadgrams for 90 languages after 5 days of computation.

Finally, we also measured time required for classifying all testing examples. The results are in Table 4, and visualised in Figure 8 and Figure 6. Times displayed in the table and charts represents the number of seconds needed for classifying 1000 chunks.



Prediction time for 1 MB corpus and the best n-gram length with respect to the number of languages.

The RPART algorithm was the fastest classifier followed by both scoring functions, whereas NB was the slowest one. All algorithms with 4 times more data achieved slightly higher accuracy, but their training took 4 times longer, with the exception of the SVM which took at least 10 times longer. The SVM algorithm is the least scalable

N-Gram	L	1		2		3		4	
Method	S	1MB	4MB	1MB	4MB	1MB	4MB	1MB	4MB
SVM	30	215	1858	663	1774	627	7976	655	3587
	60	1499	13653	7981	87260	7512	44288	26943	207123
	90	2544	24841	12698	267824	76693	-	27964	-
NB	30	5	19	27	83	40	144	54	394
	60	9	32	76	255	142	515	363	1187
	90	12	56	188	683	298	1061	672	2245
RPART	30	44	189	144	946	267	1275	369	1360
	60	162	1332	736	3447	1270	11114	2583	7493
	90	351	1810	1578	7647	5139	23413	6736	17659
W2C	30	1	1	0	1	0	1	1	1
	60	1	2	1	2	2	1	2	2
	90	1	1	1	1	3	1	2	1
YALI	30	1	1	1	1	1	1	1	1
	60	2	2	2	2	2	2	2	0
	90	2	1	2	1	3	1	3	2

Table 3: Training Time

Method		30	60	90
SVM n=2	Acc	100.0%	98.5%	98.0%
	Tre	663	7981	12698
	Pre	10.3	66.2	64.1
NB n=1	Acc	91.8%	78.7%	75.4%
	Tre	5	9	12
	Pre	13.0	18.2	22.2
RPART n=1	Acc	97.3%	90.2%	64.3%
	Tre	44	162	351
	Pre	0.1	0.2	0.1
W2C n=4	Acc	97.9%	89.1%	87.8%
	Tre	1	2	2
	Pre	1.3	2.8	12.3
YALI n=4	Acc	99.9%	96.8%	95.4%
	Tre	1	2	3
	Pre	1.3	2.7	3.6

Table 5: Comparison of classifiers with best parameters. Label *Acc* represents accuracy, *Tre* represents training time in seconds, and *Pre* represents prediction time for 1000 chunks in seconds.

algorithm of all the examined – all the rest required proportionally more time for training and prediction when the greater training corpus was used or more languages were classified.

The comparison of all methods is presented in Table 5. For each model we selected the n-grams size with the best trade-off between accuracy and time required for training and prediction. The two most accurate algorithms are SVM and YALI. The SVM achieved the highest accuracy for all languages but its training took around 4000 times longer and classification was around 17 times slower than the YALI.

In the next step we evaluated the YALI algorithm for various size of selected n-grams. These

	Languages		
Size	30	140	1000
100	64.9%	85.7 %	93.8 %
200	68.7%	87.3 %	93.9 %
400	71.7%	88.0 %	94.0 %
800	73.7%	88.5 %	94.0 %
1600	75.0%	88.8%	94.0%

Table 6: Effect of the number of selected 4-grams on accuracy.

experiments were evaluated on the data set *yali-dataset-standard*. Achieved results are presented in Table 6. The number of used n-grams increased the accuracy for short samples from 64.9% to 75.0% but it had no effect on long samples.

As the last step in evaluation we decided to compare the YALI with Google Translate (GT), which also provides language identification for 50 languages through their API.[6] For comparison we used data set *yali-dataset-small* which contains 50 samples of length 30 and 140 for each language (4800 samples in total). Achieved results are presented in Table 7. The GT and the YALI perform comparably well on samples of length 30 on which they achieved accuracy 93.6% and 93.1% respectively, but on samples of length 140 GT with accuracy 97.3% outperformed YALI with accuracy 94.8%.

7 Conclusions & Future Work

In this paper we compared 5 different algorithms for language identification – three based on the

[6] `http://code.google.com/apis/language/translate/v2/using_rest.html`

N-Gram	L	1		2		3		4	
Method	S	1MB	4MB	1MB	4MB	1MB	4MB	1MB	4MB
SVM	30	3.7	7.3	10.3	6.8	9.0	31.8	9.3	13.8
	60	13.3	30.1	66.2	189.7	59.8	92.8	236.7	375.2
	90	16.1	36.7	64.1	381.4	414.9	-	133.4	-
NB	30	13.0	13.6	75.3	77.1	132.7	147.9	186.0	349.7
	60	18.2	18.8	155.3	162.0	291.5	297.4	860.3	676.0
	90	22.2	24.7	318.1	251.9	546.3	469.3	1172.8	1177.8
RPART	30	0.1	0.1	0.3	0.1	0.1	0.2	0.7	0.2
	60	0.2	0.1	0.2	0.0	0.2	0.4	0.8	0.2
	90	0.1	0.1	0.2	0.1	0.4	0.3	1.2	0.3
W2C	30	0.7	0.8	1.7	1.6	3.3	1.5	1.3	2.2
	60	1.3	1.3	2.2	2.4	2.7	2.5	2.8	2.9
	90	2.1	1.8	4.0	3.2	4.4	3.8	12.3	5.8
YALI	30	0.7	0.8	1.0	1.2	2.0	1.9	1.3	2.2
	60	1.3	1.5	1.8	2.2	2.5	2.2	2.7	2.5
	90	2.2	1.8	2.7	2.9	4.4	3.5	3.6	3.7

Table 4: Prediction Time

		Text Length	
		30	140
System	Google	93.6%	97.3%
	YALI	93.1%	94.8%

Table 7: Comparison of Google Translate and YALI on 48 languages.

standard classification algorithms (Support Vector Machine (SVM), Naive Bayes (NB), and Regression Tree (RPART)) and two based on scoring functions. For investigating the influence of the amount of training data we constructed two corpora from the Wikipedia with 90 languages. To investigate the influence of number if identified languages we created three sets with 30, 60, and 90 languages. We also measured time required for training and classification.

Our experiments revealed that the standard classification algorithms requires at most bigrams while the scoring ones required quadgrams. We also showed that Regression Trees and Naive Bayes are not suitable for language identification because they achieved accuracy 64.3% and 75.4% respectively.

The best classifier for language identification was the SVM algorithm which achieved accuracy 98% for 90 languages but its training took 4200 times more and its classification was 16 times slower than the YALI algorithm with accuracy 95.4%. This YALI algorithm has also potential for increasing accuracy and number of recognized languages because it scales well.

We also showed that the YALI algorithm is comparable with the Google Translate system. Both systems achieved accuracy 93% for samples of length 30. On samples of length 140 Google Translate with accuracy 97.3% outperformed YALI with accuracy 94.8%.

All data sets as well as source codes are available at `http://ufal.mff.cuni.cz/~majlis/yali/`.

In the future we would like to focus on using described techniques not only on recognizing languages but also on recognizing character encodings which is directly applicable for web crawling.

Acknowledgments

The research has been supported by the grant Khresmoi (FP7-ICT-2010-6-257528 of the EU and 7E11042 of the Czech Republic).

References

[Baldwin and Lui2010] Timothy Baldwin and Marco Lui. 2010. *Language identification: the long and the short of the matter*. Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics, pp. 229–237.

[Beesley1988] Kenneth R. Beesley. 1988. *Language identifier: A computer program for automatic natural-language identification of on-line text*. Languages at Crossroads: Proceedings of the 29th Annual Conferenceof the American Translators Association, 12-16 October 1988, pp. 47-54.

[Cavnar and Trenkle1994] William B. Cavnar and John M. Trenkle. 1994. *N-gram-based text categoriza-*

tion. In Proceedings of Symposium on Document Analysis and Information Retrieval.

[Choong et al.2011] Chew Yew Choong, Yoshiki Mikami, and Robin Lee Nagano. 2011. *Language Identification of Web Pages Based on Improved N gram Algorithm*. IJCSI, issue 8, volume 3.

[Dimitriadou et al.2011] Evgenia Dimitriadou, Kurt Hornik, Friedrich Leisch, David Meyer, and and Andreas Weingessel 2011. *e1071: Misc Functions of the Department of Statistics (e1071), TU Wien*. R package version 1.5-27. `http://CRAN.R-project.org/package=e1071`.

[Gold1967] E. Mark Gold. 1967. *Language identification in the limit*. Information and Control, 5:447474.

[Grothe et al.2008] Lena Grothe, Ernesto William De Luca, and Andreas Nrnberger. 2008. *A Comparative Study on Language Identification Methods*. Proceedings of the Sixth International Language Resources and Evaluation (LREC'08). Marrakech, 980-985.

[Halevy et al.2009] Alon Halevy, Peter Norvig, and Fernando Pereira. 2009. *The unreasonable effectiveness of data*. IEEE Intelligent Systems, 24:8–12.

[Hayati 2004] Katia Hayati. 2004. Language Identification on the World Wide Web. Master Thesis, University of California, Santa Cruz. `http://lily-field.net/work/masters.pdf`.

[Hornik et al.2006] Kurt Hornik, Alexandros Karatzoglou, and David Meyer. 2006. *Support Vector Machines in R*. Journal of Statistical Software 2006., 15.

[Hughes et al.2006] Baden Hughes, Timothy Baldwin, Steven Bird, Jeremy Nicholson, and Andrew Mackinlay. 2006. *Reconsidering language identification for written language resources*. Proceedings of LREC2006, 485–488.

[Kruengkrai et al.2005] Canasai 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 (ISCIT2005), pages 896899, Beijing, China.

[Leonard and Doddington1974] Gary R. Leonard and George R. Doddington. 1974. *Automatic language identification*. Technical report RADC-TR-74-200, Air Force Rome Air Development Center.

[Lewis2009] M. Paul Lewis. 2009. *Ethnologue: Languages of the World, Sixteenth edition*. Dallas, Tex.: SIL International. Online version: `http://www.ethnologue.com/`

[McNamee2005] Paul McNamee. 2005. *Language identification: a solved problem suitable for undergraduate instruction*. J. Comput. Small Coll, volume: 20, issue: 3, February 2005, 94–101. Consortium for Computing Sciences in Colleges, USA.

[Majliš2012] Martin Majliš, Zdeněk Žabokrtský. 2012. *Language Richness of the Web*. In Proceedings of the Eight International Language Resources and Evaluation (LREC'12), Istanbul, Turkey, May 2012.

[Majliš2011] Martin Majliš. 2011. *Large Multilingual Corpus*. Mater Thesis, Charles University in Prague.

[Martins and Silva2005] Bruno Martins and Mário J. Silva. 2005. Language identification in web pages. Proceedings of the 2005 ACM symposium on Applied computing, SAC '05, 764–768. ACM, New York, NY, USA. `http://doi.acm.org/10.1145/1066677.1066852`.

[R2009] R Development Core Team. 2009. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. ISBN 3-900051-07-0. `http://www.R-project.org`,

[Sibun and Reynar1996] Penelope Sibun and Jeffrey C. Reynar. 1996. *Language identification: Examining the issues*. In Proceedings of the 5th Symposium on Document Analysis and Information Retrieval.

[Therneau et al.2010] Terry M. Therneau, Beth Atkinson, and R port by Brian Ripley. 2010. *rpart: Recursive Partitioning*. R package version 3.1-48. `http://CRAN.R-project.org/package=rpart`.