Evaluation of language identification methods using 285 languages

Tommi Jauhiainen
University of Helsinki
@helsinki.fi

Krister Lindén
University of Helsinki
@helsinki.fi

Heidi Jauhiainen
University of Helsinki
@helsinki.fi

Abstract

Language identification is the task of giving a language label to a text. It is an important preprocessing step in many automatic systems operating with written text. In this paper, we present the evaluation of seven language identification methods that was done in tests between 285 languages with an out-of-domain test set. The evaluated methods are, furthermore, described using unified notation. We show that a method performing well with a small number of languages does not necessarily scale to a large number of languages. The HeLI method performs best on test lengths of over 25 characters, obtaining an F_1-score of 99.5 already at 60 characters.

1 Introduction

Automatic language identification of text has been researched since the 1960s. Language identification is an important preprocessing step in many automatic systems operating with written text. State of the art language identifiers obtain high rates in both recall and precision. However, even the best language identifiers do not give perfect results when dealing with a large number of languages, out-of-domain texts, or short texts. In this paper seven language identification methods are evaluated in tests incorporating all three of these hard contexts. The evaluations were done as part of the Finno-Ugric Languages and The Internet project (Jauhiainen et al., 2015) funded by the Kone Foundation Language Programme (Kone Foundation, 2012). One of the major goals of the project is creating text corpora for the minority languages within the Uralic group.

In Section 2, we describe the methods chosen for this evaluation. In Section 3, we present the corpora used for training and testing the methods and in Section 4 we discuss and present the results of the evaluations of the methods using these corpora.

2 Previous work

There are not many previously published articles which provide language identification results for more than 100 languages. Results for such evaluations were provided by King and Dehdari (2008), Jauhiainen (2010), Vatanen et al. (2010), Rodrigues (2012), and Brown (2012). King and Dehdari (2008) achieved 99% accuracy with 500 bytes of input for over 300 languages. Vatanen et al. (2010) created a language identifier which included 281 languages and obtained an in-domain identification accuracy of 62.8% for extremely short samples (5-9 characters). Rodrigues (2012) presents a boosting method using the method of Vatanen et al. (2010) for language identification. His method could possibly also be used with other language identification methods and we leave the evaluation of the boosting method to future work. The language identifier created by Brown (2012), "whatlang", obtains 99.2% classification accuracy with smoothing for 65 character test strings when distinguishing between 1,100 languages (Brown, 2013; Brown, 2014).

The HeLI method described in Jauhiainen et al. (2016) was used successfully with 103 languages by Jauhiainen (2010). Some of the more detailed results concerning the Uralic languages for the evaluations presented in this paper were previously published by Jauhiainen et al. (2015).

In this section, we also include the original method of Cavnar and Trenkle (1994), as it is the most frequently used baseline in the language identification literature. As baselines, we have also included the methods presented by Tromp and Pechenizkiy (2011) and Vogel and Tresner-Kirsch (2012), which provided promising results when used with 6 languages.

Proceedings of the 21st Nordic Conference of Computational Linguistics, pages 183–191,
Gothenburg, Sweden, 23-24 May 2017.

2.1 On notation (Jauhiainen et al., 2016)

A corpus C consists of individual tokens u which may be words or characters. A corpus C is a finite sequence of individual tokens, $u_1, ..., u_l$. The total count of all individual tokens u in the corpus C is denoted by l_C. A set of unique tokens in a corpus C is denoted by $U(C)$. The number of unique tokens is referred to as $|U(C)|$. A feature f is some countable characteristic of the corpus C. When referring to all features F in a corpus C, we use C^F and the count of all features is denoted by l_{C^F}. The count of a feature f in the corpus C is referred to as $c(C, f)$. An n-gram is a feature which consists of a sequence of n individual tokens. An n-gram starting at position i in a corpus is denoted $u_{i,...,i-1+n}$. If $n = 1$, u is an individual token. When referring to all n-grams of length n in a corpus C, we use C^n and the count of all such n-grams is denoted by l_{C^n}. The count of an n-gram u in a corpus C is referred to as $c(C, u)$ and is defined by Equation 1.

$$c(C,u) = \sum_{i=1}^{l_C+1-n} \begin{cases} 1 & \text{, if } u = u_{i,...,i-1+n} \\ 0 & \text{, otherwise} \end{cases} \quad (1)$$

The set of languages is G, and l_G denotes the number of languages. A corpus C in language g is denoted by C_g. A language model O based on C_g is denoted by $O(C_g)$. The features given values by the model $O(C_g)$ are the domain $dom(O(C_g))$ of the model. In a language model, a value v for the feature f is denoted by $v_{C_g}(f)$. A corpus in an unknown language is referred to as a mystery text M. For each potential language g of a corpus M, a resulting score $R(g, M)$ is calculated.

2.2 N-Gram-Based Text Categorization

The method of Cavnar and Trenkle (1994) uses overlapping character n-grams of varying size calculated from words. The language models are created by tokenizing the training texts for each language g into words and then padding each word with spaces, one before and four after. Each padded word is then divided into overlapping character n-grams of sizes from 1 to 5 and the counts of every unique n-gram are calculated over the whole corpus. The n-grams are ordered by frequency and k of the most frequent n-grams, $u_1, ..., u_k$, are used as the domain of the language model $O(C_g)$ for the language g. The rank of an n-gram u in language g is determined by the n-gram frequency in the training corpus C_g and denoted $rank_{C_g}(u)$.

During language identification, the mystery text is treated in a similar way and a corresponding model $O(M)$ of the k most frequent n-grams is created. Then a distance score is calculated between the model of the mystery text and each of the language models. The value $v_{C_g}(u)$ is calculated as the difference in ranks between $rank_M(u)$ and $rank_{C_g}(u)$ of the n-gram u in the domain $dom(O(M))$ of the model of the mystery text. If an n-gram is not found in a language model, a special penalty value p is added to the total score of the language for each missing n-gram. The penalty value should be higher than the maximum possible distance between ranks. We use $p = k + 1$, as the penalty value.

$$v_{C_g}(u) = \begin{cases} |rank_M(u) - rank_{C_g}(u)|, \text{ if } u \in dom(O(C_g)) \\ p, \text{ if } u \notin dom(O(C_g)) \end{cases} \quad (2)$$

The score $R_{sum}(g, M)$ for each language g is the sum of values as in Equation 3.

$$R_{sum}(g,M) = \sum_{i=1}^{l_{M^F}} v_{C_g}(f_i) \quad (3)$$

The language having the lowest score $R_{sum}(g, M)$ is selected as the identified language.

2.3 LIGA-algorithm

The graph-based n-gram approach called LIGA was first described in (Tromp, 2011). The method is here reproduced as explained in (Vogel and Tresner-Kirsch, 2012). The language models consist of relative frequencies of character trigrams and the relative frequencies of two consecutive overlapping trigrams. The frequency of two consecutive overlapping trigrams is exactly the same as the 4-gram starting from the beginning of the first trigram. So the language models consist of the relative frequencies $v_{C_g}(u)$ of 3- and 4-grams as in Equation 4.

$$v_{C_g}(u) = \frac{c(C_g, u)}{l_{C_g^n}} \quad (4)$$

where $c(C_g, u)$, is the number of 3- or 4-grams u and $l_{C_g^n}$, is the total number of 3- and 4-grams in the training corpus.

The mystery text M is scanned for the 3- and 4-grams u. For each 3- and 4-gram found in the model of a language g, the relative frequencies are added to the score $R_{sum}(g, M)$ of the language g, as in Equation 3. The winner is the language with the

highest score as opposed to the lowest score with the previous method.

In the logLIGA variation of the method, introduced by Vogel and Tresner-Kirsch (2012), the natural logarithm of the frequencies is used when calculating the relative frequencies, as in Equation 5.

$$v_{C_g}(u) = \frac{ln(c(C_g,u))}{ln(l_{C_g^n})} \quad (5)$$

Otherwise the method is identical to the original LIGA algorithm.

2.4 The method of King and Dehdari (2008)

King and Dehdari (2008) tested the use of the relative frequencies of byte n-grams with Laplace and Lidstone smoothings in distinguishing between 312 languages. They separately tested overlapping 2-, 3-, and 4-grams with both smoothing techniques. They used the Universal Declaration of Human Rights corpus, which is accessible using NLTK (Bird, 2006), separating the testing material before training. The values for each n-gram are calculated as in Equation 6,

$$v_{C_g}(u) = \frac{c(C_g,u)+\lambda}{l_{C_g^n} + |U(C_g^n)|\lambda} \quad (6)$$

where $v_{C_g}(u)$ is the probability estimate of n-gram u in the model and $c(C_g,u)$ its frequency in the training corpus. $l_{C_g^n}$ is the total number of n-grams of length n and $|U(C_g^n)|$ the number of distinct n-grams in the training corpus. λ is the Lidstone smoothing parameter. When using Laplace smoothing, the λ is equal to 1 and with Lidstone smoothing, the λ is usually set between 0 and 1. King and Dehdari (2008) found that Laplace smoothing with the bigram model turned out to be the most accurate on two of their longer test sets and that Lidstone smoothing (with λ set to 0.5) was better with the shortest test set. King and Dehdari (2008) used the *entropy(model, text)* function of NLTK, which evaluates the entropy between *text* and *model* by summing up the log probabilities of words found in the text.[1]

2.5 "Whatlang" program (Brown, 2013)

The "Whatlang" program uses variable length byte n-grams from 3 to 12 bytes as its language model. K of the most frequent n-grams are extracted from training corpora for each language and their relative frequencies are calculated. In the tests reported in (Brown, 2013), K varied from 200 to 3,500 n-grams. After the initial models are generated, n-grams, which are substrings of longer n-grams in the same model, are filtered out, if the frequency of the longer n-gram is at least 62% of the shorter n-grams frequency. The value $v_{C_g}(u)$ of an n-gram u in the model of the corpus C_g is calculated as in Equation 7

$$v_{C_g}(u) = \Big(\frac{c(C_g,u)}{l_{C_g^n}}\Big)^{0.27} n^{0.09} \quad (7)$$

where $c(C_g,u)$ is the frequency of the n-gram u and $l_{C_g^n}$ is the number of all n-grams of the length n in the training corpus C_g. The weights in the model are calculated so that the longer n-grams have greater weights than short ones with the same relative frequency. Baseline language models $O_{base}(C_g)$ are formed for each language g using the values $v_{C_g}(u)$.

For each language model $O_{base}(C_g)$, the cosine similarity between it and every other language model is calculated. A union of n-grams is formed by taking all of the models for which the similarity is higher than an empirically determined threshold. The corpus C_g is scanned for occurrences of the n-grams in the union. If some of the n-grams are not found at all, these n-grams are then appended with negative weights to the base model. The negative weight used for an n-gram u in the model $O(C_g)$ is the maximum cosine similarity between $O_{base}(C_g)$ and the models containing an n-gram u times the maximum $v_C(u)$ within those models. These negative weighted n-grams are called stop-grams. If the size of the training corpus for a certain model is less than 2 million bytes, the weights of the stop-grams are discounted as a function of the corpus size.

The score $R_{whatlang}(g,M)$ for the language g is calculated as in Equation 8

$$R_{whatlang}(g,M) = \frac{\sum_i v_{C_g}(u_i)}{l_{M^1}} \quad (8)$$

where u_i are the n-grams found in the mystery text M. The score is also normalized by dividing it with the length (in characters) of the mystery text l_{M^1}. The language with the highest score is identified as the language of the mystery text.

Brown (2013) tested "Whatlang" with 1,100 languages as well as a smaller subset of 184 languages. The reported average of classification ac-

[1] Jon Dehdari recently uploaded the instructions to: `https://github.com/jonsafari/witch-language`

curacy with 1,100 languages for lines up to 65 characters is 98.68%, which is extremely good.

2.6 VariKN toolkit (Vatanen et al., 2010)

The problem with short text samples was considered by Vatanen et al. (2010). Several smoothing techniques with a naive Bayes classifier were compared in tests of 281 languages. Absolute discounting (Ney et al., 1994) smoothing with a maximum n-gram length of 5 turned out to be their best method. When calculating the Markovian probabilities in absolute discounting, a constant D is subtracted from the counts $c(C,u_{i-n+1}^n)$ of all observed n-grams u_{i-n+1}^n and the left-out probability mass is distributed between the unseen n-grams in relation to the probabilities of lower order n-grams $P_g(u_i|u_{i-n+2}^{n-1})$, as in Equation 9.

$$P_g(u_i|u_{i-n+1}^{n-1}) = \frac{c(C,u_{i-n+1}^n) - D}{c(C,u_{i-n+1}^{n-1})} + \lambda_{u_{i-n+1}^{n-1}} P_g(u_i|u_{i-n+2}^{n-1}) \tag{9}$$

The language identification is performed using the "perplexity" program provided with the toolkit.[2] Perplexity is calculated from the Markovian probability $P(M|C_g) = \prod_i P_g(u_i|u_{i-n+1}^{n-1})$ for the mystery text M given the training data C_g as in Equations 10 and 11.

$$H_g(M) = -\frac{1}{c(M,u)} \prod_i log_2 P_g(u_i|u_{i-n+1}^{n-1}) \tag{10}$$

$$R_{perplexity}(g,M) = 2^{H_g(M)} \tag{11}$$

2.7 The HeLI method

The HeLI[3] method (Jauhiainen, 2010) is described in Jauhiainen et al. (2016) using the same notation as in this article. In the method, each language is represented by several different language models only one of which is used for every word found in the mystery text. The language models for each language are: a model based on words and one or more models based on character n-grams from one to n_{max}. When a word not included in the model based on words is encountered in the mystery text M, the method backs off to using the n-grams of the size n_{max}. If it is not possible to apply the n-grams of the size n_{max}, the method backs off to lower order n-grams and continues backing off until character unigrams, if needed. A development set is used for finding the best values for the parameters of the method. The three parameters are the maximum length of the used character n-grams (n_{max}), the maximum number of features to be included in the language models (cut-off c), and the penalty value for those languages where the features being used are absent (penalty p). Because of the large differences between the sizes of the training corpora, we used a slightly modified implementation of the method, where we used relative frequencies as cut-offs c. The values in the models are 10-based logarithms of the relative frequencies of the features u, calculated using only the frequencies of the retained features, as in Equation 12

$$v_{C_g}(u) = \begin{cases} -\log_{10}\left(\frac{c(C_g,u)}{l_{C_g}}\right) & \text{, if } c(C_g,u) > 0 \\ p & \text{, if } c(C_g,u) = 0 \end{cases} \tag{12}$$

where $c(C_g,u)$ is the number of features u and l_{C_g} is the total number of all features in language g. If $c(C_g,u)$ is zero, then $v_{C_g}(u)$ gets the penalty value p.

A score $v_g(t)$ is calculated for each word t in the mystery text for each language g, as shown in Equation 13.

$$v_g(t) = \begin{cases} v_{C_g}(t) & \text{, if } t \in dom(O(C_g)) \\ v_g(t, min(n_{max}, l_t+2)) & \text{, if } t \notin dom(O(C_g)) \end{cases} \tag{13}$$

The whole mystery text M gets the score $R_{HeLI}(g,M)$ equal to the average of the scores of the words $v_g(t)$ for each language g, as in Equation 14

$$R_{HeLI}(g,M) = \frac{\sum_{i=1}^{l_{T(M)}} v_g(t_i)}{l_{T(M)}} \tag{14}$$

where $T(M)$ is the sequence of words and $l_{T(M)}$ is the number of words in the mystery text M. The language having the lowest score is assigned to the mystery text.

3 Test setting

In addition to the Uralic languages relevant to the project (Jauhiainen et al., 2015), the languages for the evaluation of the language identification methods were chosen so that we were able to train and test with texts from different sources, preferably also from different domains. We were able to gather suitable corpora for a set of 285 languages.[4]

[2] https://github.com/vsiivola/variKN

[3] https://github.com/tosaja/HeLI

[4] The list of all the languages and most of the sources can be found at http://suki.ling.helsinki.fi/LILanguages.html.

In our project we are interested in gathering as much of the very rare Uralic texts as possible, so we need a high recall. On the other hand, if our precision is bad, we end up with a high percentage of incorrect language labels for the rare languages. For these reasons we use the F_1-score as the main performance measure when evaluating the language identifiers. We calculate the language-level averages of recall, precision and the F_1-score. Language-level averages are referred to as *macro-averages* by Lui et al. (Lui et al., 2014). As the number of mystery texts for each language were identical, the macro-averaged recall equals the commonly used classification accuracy[5]. The F_β-score is based on the effectiveness measure introduced by van Rijsbergen (1979) and is calculated from the precision p and recall r, as in Equation 15

$$F_\beta = (1+\beta^2)\left(\frac{pr}{(\beta^2 p)+r}\right) \quad (15)$$

where $\beta = 1$ gives equal weight to precision and recall.

3.1 Training Corpora

The biggest bulk of the training corpora is formed from various Wikipedias.[6] The collection sizes range from a few articles for the very small languages to over a million articles in the English, German, French, Dutch and Italian collections. The sheer amount of linguistic material contained in the article collections makes using them as text corpora an appealing thought. The article collections had to be cleaned as they contained lots of non-lingual metadata and links as well as text in non-native languages. In addition to the text from Wikipedia, there is material from bible translations[7], other religious texts[8], the Leipzig Corpora Collection (Quasthoff et al., 2006), the AKU project[9], Sámi giellatekno[10], and generic web pages. Even with these additions, the amount of training material differs greatly between languages.

Each language has one file including all the training texts for that language. Some of the texts are copyrighted, so they cannot be published as such. The amount of training material differs drastically between languages: they span from 2,710 words of Tahitian to 29 million words of English. Some of the corpora were manually examined to remove text obviously written in foreign languages. Even after all the cleaning, the training corpora must be considered rather unclean.

3.2 Testing Corpora

The test corpora are mostly derived from the translations of the universal declaration of human rights.[11] However, the test set includes languages for which no translation of the declaration is available and for these languages texts were collected from some of the same sources as for the training corpora, but also from Tatoeba.[12] Most of the test texts have been examined manually for purity, so that obvious inclusions of foreign languages were removed.

The aim was to have the mystery texts from different domains than the training texts. Wikipedia refers to the declaration of human rights in several languages and in many places. In order to deal with possible inclusion of test material in training corpora, every test corpus was divided into 30 character chunks and any lines including these chunks in the corresponding training corpus were removed. Also, if long sequences of numbers were noticed, they were removed from both corpora. There are still numbers in the test set and for example some of the 5 character or even 10 character sequences in the test set consist only or mostly of numbers.

The test set has been randomly generated from the test corpora. A test sample always begins at the beginning of a word, but it might end anywhere, including in the middle of a word. An extra blank was inserted in the beginning of each line when testing those language identifiers, which did not automatically expect the text to begin with a word. The test samples are of 19 different sizes ranging from 5 to 150 characters. Each language and size pair has 1,000 random (some can be identical) test samples. The full test set comprises of around 5.4

[5] The evaluation results for all the languages and identifiers can be found at `http://suki.ling.helsinki.fi/NodaEvalResults.xlsx`.

[6] `http://www.wikipedia.org`

[7] `http://gospelgo.com/biblespage.html`, `http://worldbibles.org`, `http://ibt.org.ru`, `http://gochristianhelps.com`

[8] `http://www.christusrex.org`, `https://www.lds.org`

[9] `http://www.ling.helsinki.fi/~rueter/aku-index.shtml`

[10] `http://giellatekno.uit.no`

[11] `http://www.unicode.org/udhr/`

[12] `http://tatoeba.org/eng/`

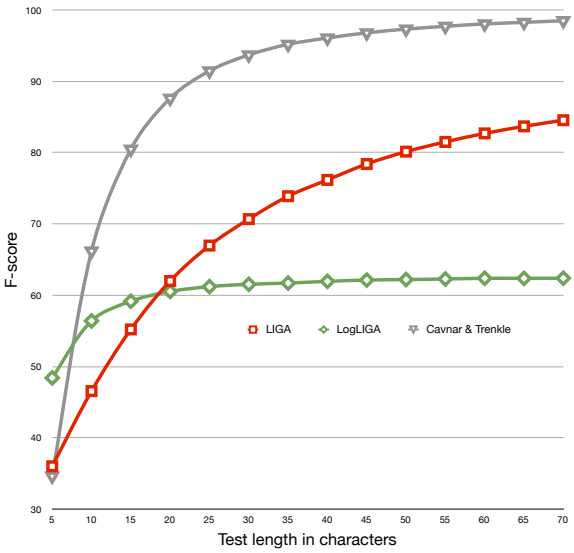

Figure 1: Comparison between the F_1-scores of the method of Cavnar and Trenkle (1994), LIGA, and logLIGA.

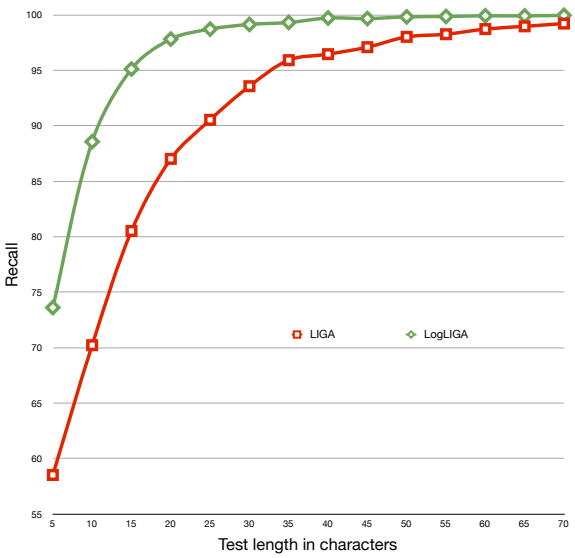

Figure 2: The recall of LIGA and logLIGA algorithms with 6 languages.

million samples to be identified.

4 The results and discussion

After training the aforementioned language identifiers with our own training corpora, we tested them against all the languages in our test suite.

4.1 The baselines

Cavnar and Trenkle (1994) included the 300 most frequent n-grams in the language models. In our tests the best results were attained using 20,000 n-grams with their method. From the LIGA variations introduced by Vogel and Tresner-Kirsch (2012), we chose to test the logLIGA as it performed the best in their evaluations in addition to the original LIGA algorithm. For these three methods, the averaged results of the evaluations for 285 languages can be seen in Figure 1.

The results of both the LIGA and logLIGA algorithms are clearly outperformed by the method of Cavnar and Trenkle (1994). Especially the poor results of logLIGA were surprising, as it was clearly better than the original LIGA algorithm in the tests presented by Vogel and Tresner-Kirsch (2012). To verify the performance of our implementations, we tested them with the same set of languages which were tested in (Vogel and Tresner-Kirsch, 2012), where the baseline LIGA had an average recall of 97.9% and logLIGA 99.8% over 6 languages. The tweets in their dataset average around 80 characters. The results of our tests can be seen in Figure 2. The logLIGA clearly outperforms the LIGA algorithm and obtains 99.8% recall already at 50 characters even for our cross-domain test set. From these results we believe that especially the logLIGA algorithm does not scale to a situation with a large number of languages.

4.2 The evaluations

For the evaluation of the method of King and Dehdari (2008) we created Laplace and Lidstone smoothed language models from our training corpora and programmed a language identifier, which used the sum of log probabilities (we did not use NLTK) to measure the distance between the models and the mystery text. We tested n-grams from 1 to 6 with several different values of λ. King and Dehdari (2008) used byte n-grams, but as our corpus is completely UTF-8 encoded, we use n-grams of characters instead.

The best results (Figure 3) in our tests were achieved with 5-grams and a λ of 0.00000001. These findings are not exactly in line with those of King and Dehdari (2008). The number of languages used in both language identifiers is comparable, but the amount of training data in our corpus varies considerably between languages when compared with the corpus used by King and Dehdari (2008), where each language had about the same amount of material. The smallest test set they used was 2%, which corresponds to around 100 - 200 characters, which is comparable to the

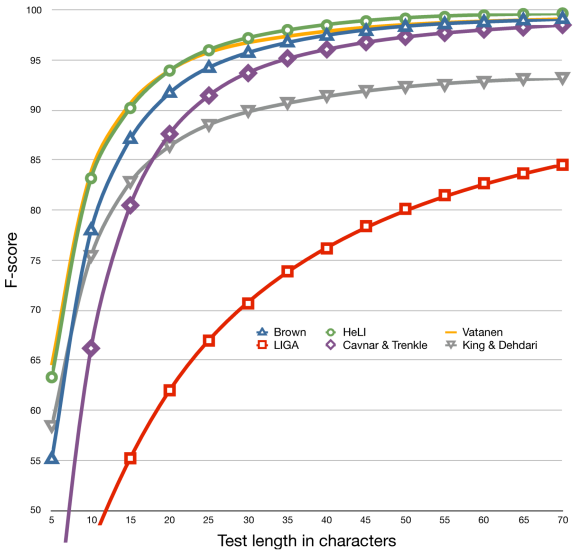

Figure 3: The F_1-scores of the six best evaluated methods.

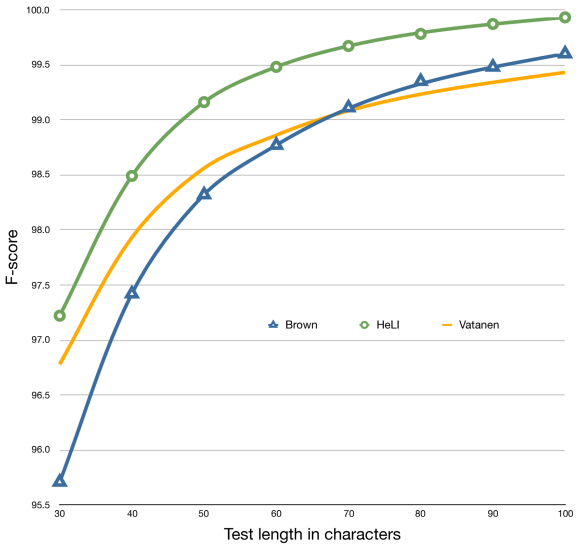

Figure 4: The F_1-scores of the HeLI method compared with the methods of Brown (2012) and Vatanen et al. (2010).

longest test sequences used in this article. We believe that these two dissimilarities in test setting could be the reason for the differing results, but we decided that investigating this further was not within the scope of this article.

In the evaluation of the method of Brown (2013), we used the "mklangid" program provided with the Brown's package[13] to create new language models for the 285 languages of our test suite. The best results with the "whatlang" were obtained using up to 10-byte n-grams, 40,000 n-grams in the models, and 160 million bytes of training data as well as stop-grams. Stop-grams were calculated for languages with a similarity score of 0.4 or higher. The average recall obtained for 65 character samples was 98.9% with an F_1-score of 99.0%. Brown's method clearly outperforms the results of the algorithm of Cavnar and Trenkle (1994), as can be seen in Figure 3. One thing to note is also the running time. Running the tests using the algorithm of Cavnar and Trenkle (1994) with 20,000 n-grams took over two days, as opposed to the less than an hour with Brown's "Whatlang" program.

In order to evaluate the method used by Vatanen et al. (2010), we utilized the VariKN toolkit (Siivola et al., 2007) to create language models from our training data with the same settings: absolute discounting smoothing with a character n-gram length of 5. When compared with the Browns identifier the results are clearly in favor of the VariKN toolkit for short test lengths and almost equal at test lengths of 70 characters, after which Brown's language identifier performs better.

For the evaluation of the HeLI method we used a slightly modified Python based implementation of the method. In our implementation, we used relative frequencies as cut-offs c instead of just the frequencies. In order to find the best possible parameters using the training corpora, we applied a simple form of the greedy algorithm using the last 10% of the training corpus for each language as a development set. We started with the same n-gram length n_{max} and the penalty value p, which were found to provide the best results in (Jauhiainen, 2010). Then we proceeded using the greedy algorithm and found at least a local optimum with the values $n_{max} = 6$, $c = 0.0000005$, and $p = 7$. The HeLI method obtains high recall and precision clearly sooner than the methods of Brown (2013) or Vatanen et al. (2010). The F_1-score of 99.5 is achieved at 60 characters, while Brown's method achieved it at 90 characters and the method of Vatanen et al. (2010) at more than 100 characters, which can be seen in Figure 4. The method of Vatanen et al. (2010) performs better than the HeLI method when the length of the mystery text is 20 characters or less.

The HeLI method was also tested without using the language models composed of words. It was found that in addition to obtaining slightly

[13] https://sourceforge.net/projects/la-strings/

Lang.	ISO	HL	LG	LL	VK	WL	CT	KG
N.-Aram.	aii	5	5	5	5	5	10	80
Amharic	amh	5	5	20	5	10	5	-
Tibetan	bod	5	10	10	5	5	25	20
Cherokee	chr	5	10	15	5	5	20	-
Greek	ell	5	5	5	5	5	10	40
Gujarati	guj	5	5	5	5	5	10	15
Armenian	hye	5	5	5	5	5	10	65
Inuktitut	iku	5	10	15	5	5	10	-
Kannada	kan	5	5	5	10	5	10	30
Korean	kor	5	5	5	5	5	15	70
Malayal.	mal	5	5	5	5	5	15	15
Thai	tha	5	5	5	20	5	15	25

Table 1: Some of the easiest languages to identify showing how many characters were needed for 100.0% recall by each method.

Lang.	ISO	HL	LG	LL	VK	WL	CT	KG
Achinese	ace	120	-	-	-	-	-	-
Bislama	bis	100	-	-	-	-	-	-
Chayah.	cbt	-	70	-	-	80	90	90
Danish	dan	150	-	-	-	-	100	-
T. Enets	enh	150	80	-	70	-	-	45
Evenki	evn	150	-	-	-	-	-	150
Erzya	myv	-	-	-	-	-	-	-
Newari	new	-	-	-	90	-	-	-
Tumbuka	tum	-	-	-	90	150	150	-
Votic	vot	-	-	-	150	-	100	-

Table 2: Some of the most difficult languages to identify showing how many characters were needed for 100.0% recall by each method.

lower F_1-scores, the language identifier was also much slower when the words were not used. We also tested using Lidstone smoothing instead of the penalty values. The best results were acquired with the Lidstone value of 0.0001, almost reaching the same F_1-scores as the language identifier with the penalty value p of 7. The largest differences in F_1-scores were at the lower mid-range of test lengths, being 0.5 with 25-character samples from the development set.

Some of the languages in the test set had such unique writing systems that their average recall was 100% already at 5 characters by many of the methods as can be seen in Table 1. Some of the most difficult languages can be seen in Table 2. In both of the Tables HL stands for HeLI, LG for LIGA, LL for LogLIGA, VK for VariKN, WL for Whatlang, CT for Cavnar and Trenkle, and KG for King and Dehdari.

5 Conclusions and Future Work

The purpose of the research was to test methods capable of producing good identification results in a general domain with a large number of languages. The methods of Vatanen et al. (2010) and Brown (2012) outperformed the other methods, even though the original method of Cavnar and Trenkle (1994) also obtained very good results. The recently published HeLI method outperforms previous methods and considerably reduces the identification error rate for texts over 60 characters in length.

There still exists several interesting language identification methods and implementations that we have not evaluated using the test setting described in this article. These methods and implementations include, for example, those of Lui and Baldwin (2012), Majli[Pleaseinsertintopreamble] (2012), and Zampieri and Gebre (2014).

Acknowledgments

We would like to thank Kimmo Koskenniemi for many valuable discussions and comments. This research was made possible by funding from the Kone Foundation Language Programme.

References

Steven Bird. 2006. Nltk: the natural language toolkit. In *COLING-ACL '06 Proceedings of the COLING/ACL on Interactive presentation sessions*, pages 69–72, Sydney.

Ralf D. Brown. 2012. Finding and identifying text in 900+ languages. *Digital Investigation*, 9:S34–S43.

Ralf D. Brown. 2013. Selecting and weighting n-grams to identify 1100 languages. In *Text, Speech, and Dialogue 16th International Conference, TSD 2013 Pilsen, Czech Republic, September 2013 Proceedings*, pages 475–483, Pilsen.

Ralf D. Brown. 2014. Non-linear mapping for improved identification of 1300+ languages. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*, pages 627–632, Doha, Qatar.

William B. Cavnar and John M. Trenkle. 1994. N-gram-based text categorization. In *Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval*, pages 161–175, Las Vegas.

Heidi Jauhiainen, Tommi Jauhiainen, and Krister Lindén. 2015. The finno-ugric languages and the internet project. *Septentrio Conference Series*, 0(2):87–98.

Tommi Jauhiainen, Krister Lindén, and Heidi Jauhiainen. 2016. Heli, a word-based backoff method for language identification. In *Proceedings of the 3rd Workshop on Language Technology for Closely Related Languages, Varieties and Dialects (VarDial)*, pages 153–162, Osaka, Japan.

Tommi Jauhiainen. 2010. Tekstin kielen automaattinen tunnistaminen. Master's thesis, University of Helsinki, Helsinki.

Josh King and Jon Dehdari. 2008. An n-gram based language identification system. The Ohio State University.

Kone Foundation. 2012. The language programme 2012-2016. http://www.koneensaatio.fi/en.

Marco Lui and Timothy Baldwin. 2012. langid.py: an off-the-shelf language identification tool. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, pages 25–30, Jeju.

Marco Lui, Jey Han Lau, and Timothy Baldwin. 2014. Automatic detection and language identification of multilingual documents. *Transactions of the Association for Computational Linguistics*, 2:27–40.

Martin Majliš. 2012. Yet another language identifier. In *Proceedings of the Student Research Workshop at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 46–54, Avignon.

Hermann Ney, Ute Essen, and Reinhard Kneser. 1994. On structuring probabilistic dependences in stochastic language modelling. *Computer Speech and Language*, 8(1):1–38.

Uwe Quasthoff, Matthias Richter, and Christian Biemann. 2006. Corpus portal for search in monolingual corpora. In *Proceedings of the fifth international conference on Language Resources and Evaluation, LREC 2006*, pages 1799–1802, Genoa.

Paul Rodrigues. 2012. *Processing Highly Variant Language Using Incremental Model Selection*. Ph.D. thesis, Indiana University.

Vesa Siivola, Teemu Hirsimäki, and Sami Virpioja. 2007. On growing and pruning kneser-ney smoothed n-gram models. *IEEE Transactions on Audio, Speech and Language Processing*, 15(5):1617–1624.

Erik Tromp and Mykola Pechenizkiy. 2011. Graph-based n-gram language identification on short texts. In *Benelearn 2011 - Proceedings of the Twentieth Belgian Dutch Conference on Machine Learning*, pages 27–34, The Hague.

Erik Tromp. 2011. Multilingual sentiment analysis on social media. Master's thesis, Eindhoven University of Technology, Eindhoven.

C. J. van Rijsbergen. 1979. *Information Retrieval*. Butterworths.

Tommi Vatanen, Jaakko J. Väyrynen, and Sami Virpioja. 2010. Language identification of short text segments with n-gram models. In *LREC 2010, Seventh International Conference on Language Resources and Evaluation*, pages 3423–3430, Malta.

John Vogel and David Tresner-Kirsch. 2012. Robust language identification in short, noisy texts: Improvements to liga. In *The Third International Workshop on Mining Ubiquitous and Social Environments*, pages 43–50, Bristol.

Marcos Zampieri and Binyam Gebrekidan Gebre. 2014. Varclass: An open source language identification tool for language varieties. In *Proceedings of Language Resources and Evaluation (LREC*.