Gergely Pethő & Eszter Mózes:
An n-gram-based language identification algorithm for variable-length and variable-language texts
Argumentum 10 (2014), 56-82
Debreceni Egyetemi Kiadó

Tanulmány

Gergely Pethő & Eszter Mózes

An n-gram-based language identification algorithm for variable-length and variable-language texts

Abstract

The aim of this paper is to describe a new language identification method that uses language models based on character statistics, or more specifically, character n-gram frequency tables or Markov chains. An important advantage of this method is that it uses a very simple and straightforward algorithm, which is similar to those that have been used for the past 20 years for this purpose. In addition, it can also handle input such as target texts in an unknown language or more than one language, which the traditional approaches inherently classify incorrectly. We systematically compare and contrast our method with others that have been proposed in the literature, and measure its accuracy using a series of experiments. These experiments demonstrate that our solution works not only for whole documents but also delivers usable results for input strings as short as a single word, and the identification rate reaches 99.9 % for strings that are about 100 characters, i.e. a short sentence, in length.
Keywords: character statistics, language identification, Markov chain, n-gram

0 Introduction

In this paper our goal is to present the outlines of a new algorithm that can recognise the language of texts by using very simple language models based on statistics of chains of characters, i.e. character n-grams. Although language identification algorithms using n-gram models have been known and used for a long time (the seminal papers of the field date back to 1994), our algorithm has some attractive characteristics that are not self-evident: it can recognise the language of very short sections of text, consisting of only a few words; it can detect multiple languages and output the ratio of these languages within a single text, or even divide multi-language texts into single-language segments (e.g. sentences or paragraphs); and it is reliable in determining if a section of text does not belong to any of the languages it has been trained for.

In Section 1 we present some background, including a summary of important criteria by which specific language identification methods can be characterised or classified. In Section 2 we review four papers on language identification that are most relevant to our approach, and briefly refer to other earlier work as well. In Section 3 we describe our algorithm itself, whereas in Section 4 we present the results of the experiments that we have used to establish the reliability of this method, and compare these results to others published in the literature.

Gergely Pethő & Eszter Mózes:
An n-gram-based language identification algorithm for variable-length and variable-language texts
Argumentum 10 (2014), 56-82
Debreceni Egyetemi Kiadó

1 Background

A language identification algorithm can tell us automatically about any written or spoken text which language it belongs to. When identifying the language of a written text manually, people usually rely on characters unique to or very typical of a certain language; common and peculiar combinations of letters, beginnings or endings of words; and the most frequent words (function words).[1] Computational algorithms, on the other hand, are usually based on dictionary comparisons, various statistical approaches, or a combination of these two.

Language recognition software is necessary for a variety of tasks. For instance, it can be used to categorise by language certain electronic text resources found on the World Wide Web. This can be necessary for various reasons, e.g. in order to compile a linguistic corpus for a given language. Whether such a corpus is used for linguistic research or to build a language model for natural language processing, it is generally problematic when it contains texts or even long parts of texts in other languages. Language identification software can be used for filtering out the 'foreign-language' or mixed-language documents from very large quantities of texts mass-downloaded from the internet, where manual classification would be impossible because of the amount of work required. Another ubiquitous use of these tools is to recognise automatically the source language for machine translation, which is used on machine translation sites (particularly Google Translate and Bing Translator) and is also integrated in web browsers (Google Chrome). A possible further application is to filter texts returned by web search engines, e.g. to display only Hungarian-language hits for a search.[2]

In our paper we will concentrate on non-dictionary-based statistical approaches to language identification for several reasons: Firstly, creating a dictionary requires a lot of effort. A complete lexicon containing all current words of a certain language would be neither necessary nor feasible. However, it is not obvious how large a dictionary should be in order to guarantee recognition of a language with a high degree of reliability, i.e. how frequent words should be added to it. For languages like Hungarian, where most lexemes can have a large number of inflected forms, putting together a dictionary that contains all possible or at least common forms of the frequent words would also be highly impractical. Thus complementing the dictionary by a module the handles morphology would also be advisable, which would in turn complicate this process even further. Regardless of whether such a dictionary is combined with a morphological module or not, it tends to require a large amount of storage space (although this can be reduced using techniques like hashing in actual implementations). Finally, using a dictionary-based method means that we need to examine relatively long stretches of text to recognise a language, since a short sequence often does not contain the necessary amount of sufficiently common words, thereby making identification impossible. By contrast, a method based on character statistics instead of a vocabulary list does not have these limitations.

Language identification methods can be characterised along various criteria, such as the following:

---

1 For a nice example, see Wikipedia's Language recognition chart at http://en.wikipedia.org/wiki/Wikipedia: Language_ recognition_chart

2 These are, of course, the raison d'être of the use of ISO language codes in HTML via the lang attribute as well, but since the latter are specified by the author of the page in question manually, they can be set incorrectly or left out and are thus not reliable as a general solution for the classification of documents on the web.

a) **The language model:** What kinds of sources are necessary? Are they available? How costly are they in terms of money or other resources (e.g. time)?
b) **What is the (length of the) unit that we want to recognise the language of?** Whole texts, blocks, sentences or words?
c) **The main characteristics of the algorithm:** statistics-based vs. rule-based; character-based vs. word-based, etc.
d) **Accuracy or reliability of results**
e) **How much time is needed for the identification?** E.g. what is the average running time per million words? Is it possible to recognise the language online (while typing) or only offline (for finished texts)?
f) **How are multilingual texts treated?** Does the method identify one of the languages that appear in the text as the language of the whole text? Can the method determine the ratios of the languages appearing in a multilingual text? Can it divide multilingual texts into sections belonging to each language?
g) How does the method **classify texts that belong to a language that is not 'known' to the system**[3] or that could in principle **belong to more than one 'known' language** (e.g. sports tables, numerical data)? Does the method still assign one of the languages to the text in this case or classify it as 'unknown' or 'uncertain'/ 'ambiguous'?

In the next section we will review part of the earlier literature on language identification and see how various approaches can be characterised in terms of these questions.

2 A survey of the literature

A number of language identification methods have been developed and published in the past two decades. Their aim has always been to achieve maximum accuracy.

All of the methods that are discussed in the literature share a common background. The first step is creating a model of the languages that we want to train the system for. In very general terms, this means that the characteristic features of each relevant language are established, collected and then stored. Some approaches also build a model for the input text in the same way as a second step.[4] Finally the language model is compared either to the input text model (as shown in Figure 1, which we have reproduced from Poutsma 2001), or directly to the input text. Based on the degree of similarity between the input text (or its model) and the language models, it is decided to which language the input text belongs.

[3] Depending on the algorithm used, this can mean for example that the system in question has not been trained to recognise the language in question (in the case of a statistical method) or no rules have been specified for that language (in case of a rule-based algorithm).

[4] Poutsma (2001) and other papers that cite it (e.g. Grothe, De Luca & Nürnberger 2008) claim that all statistical approaches generate input document models. This is evidently not true; for example, Dunning (1994), Sibun & Reynar (1996) as well as Řehůřek and Kolkus (2009) compare the language model directly to the target document instead of its model, as does our approach.

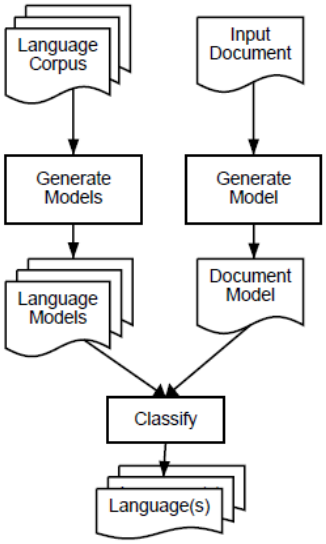

Figure 1: The structure of identification methods using language and document models

In the literature on language identification, Cavnar & Trenkle (1994) and Dunning (1994) are unanimously considered as the seminal papers of this field.[5]

2.1 Cavnar & Trenkle (1994)

Cavnar & Trenkle's (1994) method is based on character n-grams. Their objective is to identify the language of whole monolingual texts. For this reason no other sources are needed but descending lists of the most frequent n-grams appearing, on the one hand, in the input document (i.e. a document model) and, on the other hand, in the training corpus for each recognition language (i.e. language models). The ordered list for the input document is then compared to each language model. As illustrated in Figure 2, an out-of-place measure is computed by comparing the n-grams in the lists (in this particular case, 3-grams which may contain spaces) based on their ranking. If an n-gram has the same ranking in both lists, its out-of-place score is zero. If, for example, one n-gram is ranked second in the document's list and fourth in the language model, then this n-gram's out-of-place score is equivalent to the distance of the two numbers, i.e. 2. If an n-gram appearing in the document's list is not matched in the language model, the out-of-place score is a default maximum value. The out-of-place scores computed for each n-gram are then added up. The lower this sum is for a certain language, the more likely it is that the document belongs to that language; thus the algorithm assigns the language with the lowest total score to the input document. According to the authors, this method yields the best performance if the descending lists contain the 400 most frequent n-grams, and achieves an accuracy of 99.8 per cent in language classification. They note that whereas the number of n-grams considered has a bearing on accuracy, the length of the text to be categorised does not (i.e. the method works well with short texts).

[5] These papers are also summarised briefly in Grothe, De Luca & Nürnberger (2008). The sections below are loosely based on their discussion.

Gergely Pethő & Eszter Mózes:
An n-gram-based language identification algorithm for variable-length and variable-language texts
Argumentum 10 (2014), 56-82
Debreceni Egyetemi Kiadó

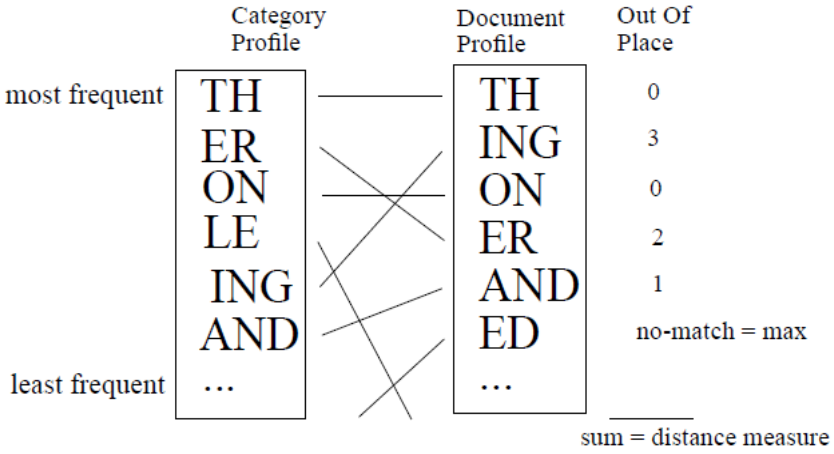

Figure 2: Determining out-of-place measures

2.2 Dunning (1994)

The basic idea underlying Dunning (1994)'s approach is the computation of the probability of the string of characters appearing in the input document given each language model that the system has been trained for. The probabilities of the strings are calculated using Markov chains, and language identification is carried out with a Bayesian decision rule. The essence of these Markov chains is that given *k* characters we have already seen (as the initial state), we calculate for each language model the conditional probability of the event that the next character will be *c* (the character that actually follows). This calculation of conditional probabilities is carried out for the whole string of characters that the input document consists of, yielding an overall probability for the whole text for each language model. The constant *k* mentioned above plus 1, i.e. the length of the whole sequence of characters under consideration at each point in this process, is called the order of the Markov model. Normally a relatively small order (2 or 3) is chosen for the purposes of language identification: Although a higher order model captures the structure of the language better, "the amount of training data needed to accurately estimate the parameters of the Markov model is roughly exponential in the order of the model" (Řehůřek & Kolkus 2009: 361). In practice, this means that higher-order models tend to exhibit very serious overtraining effects and are thus less accurate than lower-order models when identifying the language of texts that are not very similar to those in the training corpus. Like Cavnar & Trenkle's method (where a default maximum out-of-place value is used for this purpose), Dunning's approach also has to handle cases where a string of length *k* or *k* + 1 appears in the input document that is missing from a training corpus simply because the size of the latter is necessarily restricted. Given that assigning the probability 0 in such cases would give rise to misleading results, Dunning suggests a sophisticated mathematical technique for smoothing the probabilities in the Markov models. After the probabilities have been established for each language model, these can finally be weighted by the expected (prior) probability of the occurrence of each language, in case the languages are not considered equally likely. Language L is identified as the language of the input text if the probability of the input text assuming the language model for L is higher than the probability assigned to this text for all other language models. With this

method, test documents that contain just 20 characters are identified with 92 % reliability, and documents longer than 500 characters with 99.9 % (Grothe, De Luca & Nürnberger 2008: 2).

2.3 Grothe, De Luca & Nürnberger (2008)

Grothe, De Luca & Nürnberger (2008) compare the performance of language identification methods that are based on three different types of language model, namely, the short word-based model, the frequent word-based model and the character n-gram-based model. For the short word-based model, words of no longer than a set number of characters (for example, 5) are collected. This reflects the idea that the most typical words of a language, e.g. conjunctions, are only a few characters long. In this case frequency does not play a role directly, although there happens to be a significant overlap between high-frequency and short words across languages, of course. The frequent word-based model also collects words, but is based only on frequency. The idea behind this approach is that frequent words are so typical of a language that they enable us to identify the language of an unknown text with over 90 per cent accuracy (op. cit. 980). The character n-gram-based approach uses strings of characters. In all three cases the resulting language model is a list or table (of words or character n-grams, respectively) ordered by frequency.

Once a model of some type has been compiled for the languages to be recognised (and additionally, in the case of some methods, also for the document to be identified), the language classification process must be carried out, which consists in a comparison between the language model and the document, or its model. There are numerous mathematical methods that have been applied for this comparison, such as computing the probability of the target document based on each language model (see Dunning 1994's method above), similarly calculating the relative entropy (or cross-entropy) of the target document based on each language model (Sibun & Reynar 1996), computing the distance in terms of vector spaces between the language and the document models (Prager 1999), and the 'ad-hoc' distance measure between the language and document models proposed by Cavnar & Trenkle (1994), as outlined above. After a brief discussion of these approaches, the authors decide to work with the ad-hoc ranking method of Cavnar & Trenkle (1994) because of its simplicity and convincing results.[6]

As already mentioned, the main goal of Grothe, De Luca & Nürnberger (2008) is to compare the performance of language identification methods based on different types of language model. The authors carry out two experiments based on each of the three types of language model. In order to get the best results, they set different parameters for word length, n-gram length and word frequency.

Experiment materials are derived from two different sources: the Leipzig Corpora Collection (which contains subcorpora for different languages) and Wikipedia. To build the

[6] Obviously, the underlying assumption of the authors at this point is that the decisive factor for the performance of a language identification procedure is the choice of language model (i.e. short or frequent words or character n-grams, etc.) rather than the choice of algorithm or the interaction of these two factors. In other words, they seem to assume without further discussion that by combining any of the language models with the Cavnar & Trenkle (1994) method, the accuracy of language identification will be about as good relative to other language models as if they had combined the same model with a different algorithm (such as the cross-entropy method). Note that this is not a trivial assumption.

language model they use for both experiments a single document from the Leipzig Corpora for each language to be recognised.

In the first experiment they work with nine different languages. The test data set is composed of 15 randomly chosen Wikipedia documents for each language. The goal of this experiment is to establish how the language identification accuracies for the various language models compare to each other. For the frequent word-based approach they try building the models on the most frequent 10 %, 25 % and 50 % of all word forms appearing in the training and target documents, respectively. For the short word-based and n-gram-based language models they try three different lengths: 3, 4 and 5.

Table 1 shows the results of this experiment. In the first column we see the language classification method (**F**requent **W**ord, **S**hort **W**ord and character-**N**-**G**ram) and the different parameters for these models as explained in the previous paragraph. The second and third columns show the number of incorrectly and correctly identified documents. The fourth column lists the number of documents not classified by the method at all (which the authors, unfortunately, do not elaborate on). The last column shows the percentage of the correctly identified documents. As is apparent, the frequent word-based approach performs best, and the n-gram-based approach worst. In the last two lines the best performing variants are combined so that it can be checked if they can achieve a higher accuracy together, which turns out not to be the case. Unfortunately the authors again do not explain the method of combining the two models, but presumably they simply compute the total out-of-place scores for both types of model for each language, add these up, and select the language with the lowest sum.

LM	incor.	cor.	unkn.	cor. %
FW(10%)	2	133	0	98,5
FW(25%)	1	134	0	99,2
FW(50%)	2	133	0	98,5
SW(3)	9	126	0	93,3
SW(4)	8	127	0	94,1
SW(5)	11	124	0	91,8
NG(3)	3	107	25	79,2
NG(4)	7	41	87	30,3
NG(5)	36	2	97	1,5
FW(25%), SW(4)	8	127	0	94,1
FW(25%), NG(3)	18	116	1	85,9

Table 1: Results of Grothe, De Luca & Nürnberger's experiments, quoted from op. cit. 5

We should note at this point that the results for the n-gram models are surprisingly low and, in fact, do not seem correct. As we have mentioned above, Cavnar & Trenkle claim that they reach 99.8 per cent accuracy with the NG(3) type model in particular, so less than 80 % seems to indicate some methodological problem. It is also very surprising that accuracy decreases drastically for longer n-grams, which is neither indicated by the literature nor confirmed by our own results to be discussed in Section 4.2 below. We believe that these poor results might have to do with the fact that only a single short text was used to build the model, and that the authors used tokenised texts rather than raw character sequences, filling up the n-grams for the left and right edges of each word with up to $n - 1$ spaces. In any event, it would be a grave

mistake to conclude from this table that NG-based methods are inherently worse than FW or SW-based ones, and furthermore that NG(3)-based methods are far superior to NG(4) or NG(5)-based ones in terms of accuracy.

The authors' second experiment is based on the recognition that a crucial factor of the accuracy of Cavnar & Trenkle's method is the specific default maximum value of the out-of-place measure. Recall that this value is applied when an item appears in the document model that is not present in the language model, because it is either of low frequency or does not appear at all in the language corpus used to build the model. For the first experiment Grothe, De Luca & Nürnberger had chosen a fixed arbitrary maximum out-of-place value, as seems to be standard in the literature. In the second experiment they examine how accuracy can be improved if this value is chosen 'dynamically' for each model. The authors do not explain how this dynamic value is set, but we assume that they experiment with various choices until they get the best results. For the second experiment the pool of languages is increased to 13, and the number of documents to be recognised is 250 per language from the Leipzig Corpora. For SW(4), FW(25 %) and NG(3) (which had proven to be the best-performing versions of each model type during the first experiment) they are able to improve the accuracy to 100 %.[7] So Grothe, De Luca & Nürnberger (2008) conclude that the choice of the out-of-place measure plays a much more important role than that of the language model.

2.4 Řehůřek & Kolkus (2009)

In their paper, Řehůřek and Kolkus (2009) identify a number of weaknesses of the 'traditional' language identification methods that have been discussed above. Trying to apply the cross-entropy-based decision algorithm with character 3-grams to the task of real-life web page language classification, they note that the main problem lies not in an insufficient accuracy of the classification but rather in the fact that (op. cit. 359f):

1. all traditional methods return "the nearest, best-fitting language" for a given target document even if its real language is not included in the training set of the system at all, i.e. documents **whose language is unknown** to the system are mistakenly classified;
2. frequently, target documents contain **parts in more than one language** for various reasons (e.g. structure of the page including navigational or automatically generated elements); even in such cases the usual methods identify **a single language for the mixed document**, which "may even be […] a completely unrelated language not present in the input text at all";
3. "languages with considerable grammatical as well as lexical overlap", especially those belonging to the same language family, can be hard to distinguish for these methods, which is exacerbated by the fact that documents on the web are often missing the language-specific diacritics.

Trying to solve these issues in the framework of the cross-entropy-based method, they find that the results achieved through various possible solutions (such as introducing minimum thresholds for the calculated document scores, below which the target document is not assigned to any of the known languages) are not satisfactory, and furthermore these

[7] It is unclear whether the authors consider the danger of overtraining their recogniser to the specific set of target documents, and what steps they take to avert this.

complicate the already complex algorithm (op. cit. 360). For this reason, they decide to apply a completely different classification method that is based on words rather than character n-grams. To construct their language models, they do not apply any of the well-known dictionary-based solutions (such as the short word or frequent word strategy outlined above), but rather **calculate** so-called **'relevance values'** for all word forms present in the training data (op. cit. 361f). Positive relevance typically means that the word in question is indicative of the language, whereas a negative value indicates that it cannot belong to that language. The authors propose that the language of a text should be determined by adding up the positive relevance values for each word, dividing the total by the number of words in the text to get an average per-word positive value, and finally comparing this result to a threshold value to exclude texts in unknown languages (op. cit. 363). The resulting method is perfectly applicable in practice as it is very fast and the memory requirements are reasonable.

The authors tested the proposed system on a corpus of "texts" in 9 target languages (English, German, 3 Romance and 4 Slavic languages) extracted from Wikipedia, each target text being apparently a single sentence. The subcorpora for each language consisted of three types of text: short (2 to 5 words, or 10 to 30 characters), medium (6 to 50 words, or 30 to 300 characters) and long (more than 50 words or 300 characters); and they used roughly 500 short texts, 900 medium-length texts and 800 long texts from each language for testing. Table 2 shows the results for this experiment. Each cell in the table contains two values separated by a slash. The first value is the precision percentage (i.e. of all cases where the system returned that the text is, for instance, in Polish, in how many cases was this label correct, or in other words, was the text in fact in Polish?), whereas the second value is the recall percentage (of all cases where the system should have labeled a text as Italian because it is in fact in that language, in how many cases did the system assign that label to the text?). The algorithm used by Řehůřek and Kolkus – as it becomes clear in the discussion of their results (op. cit. 365f.), although not when they describe the actual classification method – permits the system to assign more than one language label to a single text. Note that labeling a sentence as both Czech and Slovak does not necessarily mean that the system found both Czech and Slovak parts in it, but rather that it cannot decide on the correct answer.

	n-gram method			dictionary method		
language	**text size**			**text size**		
code	small	medium	large	small	medium	large
cs	81.9/75.2	96.8/96.0	100.0/100.0	64.9/84.0	85.2/96.9	97.8/99.6
pl	84.1/67.6	97.4/96.5	97.9/97.2	82.9/90.2	95.5/97.0	96.9/97.5
sk	81.6/77.7	97.7/96.9	99.3/99.0	57.8/82.9	71.4/96.6	87.6/96.7
sl	89.3/79.7	97.8/97.2	99.3/99.2	68.6/88.2	91.9/97.2	98.8/99.0
it	81.9/58.4	98.7/96.4	99.9/99,8	78.6/88.1	95.8/98.0	99.4/99.7
fr	80.1/52.9	98.3/97.3	99.8/99.6	82.7/88.7	98.4/99.0	99.5/99.6
de	85.2/73.6	98.8/98.1	99.0/98.4	85.7/91.8	98.2/99.6	98.8/99.2
es	81.5/61.6	99.0/98.1	100.0/99.9	73.2/86.4	94.3/98.9	99.3/99.8
en	81.4/51.7	99.4/98.2	99.8/99.1	86.1/91.6	99.2/99.7	99.8/99.4

Table 2: Results of Řehůřek & Kolkus (2009)'s experiment, quoted from op. cit. 365.

The authors also considered the question how their system can recognise documents that are in fact multilingual and segment them into single-language parts.[8] They refer (op. cit. 366) to an earlier work (Teahan 2000), which seems to be the only paper that has described a method for dealing with this problem and reported excellent results in terms of correctness of classification, but at an extremely slow runtime performance (tens of seconds necessary to process a document, which is clearly not practicable for real-life classification tasks related to the Web or very large corpora). Their extension of the dictionary-based method to this task, however, both leads to acceptable results and performs at a speed that they describe (without providing further details) as 'impressive'.

3 Description of a new language identification method based on character statistics

3.1 General Features

The method to be introduced in the following section can be characterised as follows in terms of the criteria proposed above in Section 1:

a) The language model: The language model is based on texts retrieved from the internet. Ideally these texts cover a wide range of topics and styles for each language. Since such texts are easily accessible, construction of the necessary training corpora is very economical both in terms of money and time. Word models could probably be used in principle, but the method was developed with character n-gram models in mind.

b) What is the (length of the) unit that we want to recognise the language of? The unit of recognition is a string of target text characters of a freely specified length (e.g. 50 characters). The algorithm is extremely flexible and, in principle, works for units of any length. Recognition rates are already useful on the word level (segments of around 10 characters in length), and essentially perfect results are achieved at sentence level (50 characters and above).

c) The main characteristics of the algorithm: The algorithm is based on the calculation of the average of estimated probabilities for all character n-grams appearing in the input.

d) Accuracy or reliability of results: In practice we found that the precision of the algorithm's output is already convincing for extremely short units (above 80 % for segments of 10 characters, i.e. one or two words, and around 95 % for 20 characters, i.e. 3 or 4 words). Recognition rate reaches 99 % for segments that are around 60 characters long, and 99.9 % at around 100 characters.

[8] At this point we should mention the recent work by Pataki & Vajna (2011), which also sets out to identify the languages of multilingual documents. Their method is inspired by Cavnar & Trenkle's out-of-place-measure-based procedure. We believe that the algorithm they outline is not very convincing and the results it leads to are essentially useless in terms of practical application: Apparently it is only able to identify more or less correctly what the languages of a multilingual document are, assuming that the second language constitutes at least 30 % of the target text. However, crucially, it is completely unable to determine whether a text is multilingual in the first place, which seems to us to defeat the idea of this method regardless of its performance characteristics.

e) How much time is needed for the identification? The running time of the algorithm depends on various factors, especially the number of training languages (to which running time is linearly proportional) and the amount of output generated. On a modern commercial desktop PC machine (as of 2012), a large corpus of 100 million words is processed with 6 recognition languages in about an hour. It would be trivial to implement a version of our method that can recognise the language while typing, but we have only used and tested its offline application.

f) How are multilingual texts treated? Normally, longer texts are processed in a way that they are automatically divided into shorter segments of e.g. 100 characters and then the language for each such segment is calculated. Accordingly, if a multilingual text contains linguistically homogeneous parts in each of the known languages, the algorithm can calculate correctly how long (in % of the whole text's length) the parts belonging to each language are in total, and also output the language of each segment if necessary. If a multilingual text were handled as a single long segment, the algorithm would normally not be able to assign any of the constituent languages to the text as a whole, which is an intentional consequence of the method of evaluation. If the target document does not contain longer contiguous parts (consisting of at least a few words or a short sentence each) in each of the languages, but is instead e.g. a mixed list of individual words from these languages, like in a simple bilingual or multilingual dictionary, then the algorithm will normally classify the whole text as belonging to an unknown language. However, determining the language of the input text word by word is also a viable option in this case, see Section 4.3 below.

g) How does the method **classify texts that belong to a language that is not 'known' to it** or that could **belong to more than one 'known' language?** Our algorithm assigns a known language to a target segment just in case the calculated probability measure of the segment belonging to that language is relatively high. A segment that does not reach this probability threshold is classified as 'other', which can mean that it is in a language that the system has not been trained for, or that the linguistic material in the segment is not characteristic of any particular language in the training set. This will be the case for mixed-language target segments (e.g. a German sentence that mentions a longer English movie title) or ones comprised mainly of numbers, program code, international words, many proper names, etc.

In the section below we will outline the structure of our method and the process of how a language is assigned to a target text.

3.2 Language Models

Like in all approaches seen above, our first step is the construction of language models based on training corpora. The approaches proposed in the literature often use very small corpora for training. As mentioned above, Grothe, De Luca & Nürnberger (2008) use a single short document for each language for this purpose, and (2009) apparently use training corpora (based on the number of training sentences stated in their Table 1 in op. cit. 364) that range very roughly from 50 thousand to 2 million words. Since we assume that a bigger training corpus leads to better results unless the data are anomalous, we used very large corpora (in the order of magnitude of 100 million words) for each of the recognition languages. We should also note that the training corpus for each language was filtered for texts in 'foreign'

languages[9], and close duplicate documents were also discarded. Furthermore all texts in the training corpus were cleaned of other types of junk, e.g. accidentally included HTML code, special non-linguistic characters such as decorative character patterns, recurring navigational and structural elements of pages, etc. Thus the resulting training corpus was of a very high quality. For the experiment described further below we built 6 different subcorpora for the following languages: Hungarian, German, English, Polish, French and Italian.

On the basis of these six training corpora two different character statistics were built: simple n-gram frequency lists and Markov models. We considered not only letters as characters but also all other symbols including numbers, punctuation or whitespace. As opposed to e.g. Grothe, De Luca & Nürnberger (2008), we did not use tokenised texts for building the models or for recognition since we do not see how tokenisation could improve performance in any way; in fact, we strongly believe that it has a detrimental effect[10]. We experimented with ignoring the distinction between upper and lower case both in the models and during recognition, but found that distinguishing them does in fact improve accuracy, so in the experiments discussed below we counted upper and lower case versions as different characters.

In order to construct the simple n-gram models, the whole training corpus is scanned for n-grams of a given length, and their frequency is counted and stored as a table data type. On the basis of the n-gram counts in this table, the relative frequency and its (base 10) logarithm is calculated for each n-gram. Since the training corpora were relatively large, relative frequencies for most n-grams were very small, and thus logarithms are much easier to handle and to understand. The following table illustrates this process with the ten most frequent 3-grams of the English corpus. The first row specifies the number of all 3-grams in the corpus. The first column lists the 3-grams, the number in the second column shows how many times that 3-gram occurred in the whole corpus, and the third column is the logarithm of the relative frequency of that 3-gram, e.g. of 13,841,335 divided by 820,567,509 in the first row. The underscore characters in the first column stand for whitespace.

---

9 We used the same algorithm trained on the raw, 'dirty' versions of these corpora for filtering out the foreign-language texts. As the latter constituted a very small proportion of each language corpus, they were very atypical of the given language, and therefore their language was recognised correctly as foreign (for example, a German text appearing mistakenly in the English corpus would be correctly identified as German even if the English language model had been trained on a corpus containing this very text). The reason for this is that with such a huge training corpus the impact of single documents or even small groups of documents on the whole language model is so infinitesimal that it is unnoticeable in practice, and thus there is essentially no overtraining effect for individual documents contained in the corpus.

10 The longer the n-gram length of the model, the stronger the distortion that is caused by whitespace arbitrarily inserted at the beginning and end of the n-grams. As already mentioned in Section 2.3, we believe that this is one of the reasons why accuracy drops drastically between NG(3) and NG(5) in Table 1.

Number of 3-grams: 820,567,509		
_th	**13,841,335**	**-1.772**
the	**12,139,679**	**-1.829**
_he	**11,513,082**	**-1.852**
ion	**7,000,256**	**-2.069**
_of	**6,637,249**	**-2.092**
of_	**6,524,666**	**-2.099**
_on	**5,609,863**	**-2.165**
_in	**5,329,659**	**-2.187**
tio	**5,179,465**	**-2.199**
ed	**4,576,072**	**-2.253**

Table 3: Excerpt from a trigram model

We also tried a slightly different n-gram model which is based on the idea of Markov chains. Essentially an nth order (character-based) Markov model for a language tells us the estimated likelihood of various possible next characters, given the directly preceding string of n–1 characters. To construct this model we have to count not only all n-grams for a given value of n, but also the corresponding n–1-grams, leaving off the rightmost character. The following table presents an excerpt from a 3rd order model. The first row states the total number of 3-grams beginning with the 2-gram *wo* (563,255). The first column shows some characters that in fact follow this string in the corpus, whereas the second column states the number of 3-grams that consist of the 2-gram and the current character; for the trigram *wor* shown in the second row this number is 328,789. The estimated probability that the bigram *wo* is followed by an *r* is thus 328,789 divided by 563,255. As in the previous table, the third column lists the logarithm of the latter value, i.e. of the likelihood of the given bigram being followed by the character in question. According to these figures *wo* is followed by *r* more than half of the time and by *u* about 1 in 4 times. The other listed options are far less frequent.

Number of 3-grams beginning with *wo:* 563,255		
r	**328,789**	**-0.233**
:	**116**	**-3.686**
0	**356**	**-3.199**
j	**45**	**-4.097**
0	**3811**	**-2.169**
u	**131,672**	**-0.631**
h	**64**	**-3.944**
.	**338**	**-3.221**
f	**448**	**-3.099**
t	**137**	**-3.613**

Table 4: Excerpt from a 3rd order Markov model

Gergely Pethő & Eszter Mózes:
An n-gram-based language identification algorithm for variable-length and variable-language texts
Argumentum 10 (2014), 56-82
Debreceni Egyetemi Kiadó

Note that although there is an obvious similarity between these two types of model, the specific probabilities they list can be very different. For example, the trigram *wor* is classified as extremely frequent given the prefix *wo* according to the Markov model, but this is independent of the question how frequent the trigram *wor* is compared to all other trigrams, which is what the simple n-gram model in Table 3 expresses.

3.3 The Algorithm

After having introduced the language models, let us now turn to the algorithm that serves to establish the language of the target document. Similarly to the n-gram-based method Řehůřek & Kolkus (2009) experiment with (and then reject in favour of their proposed dictionary method), our approach is based on the idea that, on average, the probability of the n-grams appearing in a target document will be highest if we assign the probabilities listed in the correct language model to each n-gram.

The following example illustrates how the recognition of a segment would proceed based on a simple trigram model. Let us take a segment that is 9 characters long*: _korpusz_*, i.e. the Hungarian word meaning 'corpus' in a sentence-internal position between two spaces. First we have to split up this segment into trigrams, moving from the left to the right edge of the segment one character at a time. The following table lists the logarithmic probabilities for each trigram as they appear in the Hungarian, German and English language models:

	Hungarian	**German**	**English**
_ko	-3.038276676	-3.64560781	-5.984184757
kor	-2.892040054	-4.765485141	-6.083525648
orp	-5.093781399	-5.001671238	-4.44175286
rpu	-5.904191352	-4.946926072	-6.688805035
pus	-4.070700543	-6.09553898	-5.575857087
usz	-3.641693544	-4.16765841	-6.723782619
sz_	-3.258777435	-6.400193151	-6.380088211
Average	**-3.985637286**	**-5.003297257**	**-5.982570888**

Table 5: Determining the score of the segment _korpusz_ *using a simple trigram model*

The last row of the table lists the average of the logarithms for each language. The numbers mean that the average order of magnitude of the probabilities of the trigrams appearing in this segment is 10^{-4} based on the Hungarian language model, 10^{-5} based on the German model and 10^{-6} for the English model. In other words this means that, on average, it is a whole order of magnitude more likely that the n-grams constituting this string of characters form a Hungarian word than a German word, and that would in turn be another order of magnitude more likely than the n-grams constituting an English word. Thus even for such a short segment containing the adapted version of an international word, the algorithm correctly indicates that the best candidate language is Hungarian.

Using a Markov model works in a similar way and yields similar results:

	Hungarian	**German**	**English**
_k|o	-0.983255575	-0.71226489	-2.176747393
ko|r	-0.449770836	-1.643145679	-1.284555683
or|p	-2.77579409	-2.613464593	-2.364356321
rp|u	-1.793534138	-1.47833708	-3.170808082
pu|s	-0.355680973	-2.466871726	-2.436972269
us|z	-0.657383725	-1.700298274	-4.006099294
sz|_	-1.384270964	-2.760161249	-0.997708203
Average	**-1.199955757**	**-1.91064907**	**-2.348178178**

Table 6: Determining the score of the segment _korpusz_ *using a 3rd order Markov model*

The average values again correctly state that Hungarian is the most probable candidate, being 0.7 orders of magnitude more likely than German and more than an order of magnitude more likely than English. Note that the point of these examples is not to demonstrate how well the algorithm works: The character combinations appearing in international words are generally equally unusual or uncommon in all languages, and features that are (counter)indicative of a certain language (e.g. the fact that words commonly begin with *ko* in Hungarian and German but not in English, or that words ending in *sz* are frequent in Hungarian but not in the other two languages) often cannot offset this fact. Thus it is relatively likely for foreign or international words to be classified incorrectly or not assigned to a language at all by our algorithm. What the examples above are really meant to demonstrate is simply how the average orders of magnitude are calculated.

So far our method does not differ substantially[11] from others proposed in the literature, especially Dunning (1994)'s and others based on n-gram probabilities (which are sometimes also referred to as 'Monte Carlo methods', although in our view this is a misnomer). This means that we face the same problem that was highlighted in Řehůřek & Kolkus (2009: 360): if we simply assign the language with the best average logarithm, i.e. the one closest to zero, to a text, this means that texts in languages that the system has not been trained for will always be misclassified as belonging to some random known language.

A second, superficially unrelated problem is that of sparse data: some n-grams occur extremely rarely in the training corpus, e.g. the trigram *5°;* appearing in the English language model. Like in this case, these extremely rare n-grams are not typical of the language in question at all: They might be parts of foreign expressions mentioned very rarely in English texts, rare misspellings, or – like in this particular case – non-linguistic entities that could appear in any language. Therefore, sparse data (whether they do occur in the training corpus, albeit in small numbers, or do not occur there at all) are very unlikely to be of any help in deciding in favour of a language. This is why we all but ignored the issue of smoothing the probability values in the language models, to which end authors have often employed very sophisticated solutions, see e.g. Dunning (1994: 8). When experimenting with various versions of our algorithm, we found that disregarding low-frequency items in all language models actually improved recognition accuracy significantly, in addition to having the further

[11] There are, of course, small differences. Other approaches work with probabilities rather than their logarithms, and some calculate sums of probabilities rather than averages per n-gram.

obvious beneficial effect that smaller-sized models (with the rare items removed) require less memory to store.

Thus to solve the problem of sparse data we essentially use a version of the 'frequent word' method as characterised above in section 2.3, only with n-grams instead of words: we include n-grams only above a certain minimum threshold in the runtime models, e.g. those with a logarithm value > -6. If the target document contains an n-gram that is not included in this model of a recognition language, this n-gram is assigned a default value that is identical to or slightly less than the threshold, e.g. -6 or -7.[12] We proceed analogously for the Markov models, although the thresholds and defaults are much higher there, as is evident in Tables 4 and 6.

In their paper, Řehůřek & Kolkus (2009) conclude that distinguishing known languages from untrained-for ones in the n-gram-based paradigm is not viable. Applying their reasoning to our data above for the sake of exposition, the question is basically how we can tell whether the best average probability value is good enough to qualify the target segment as belonging to that language. For example, given that we got the score of -4 for Hungarian in Table 5, and this was the highest of the three scores, how can we tell whether this average probability is high enough for the target segment to qualify as Hungarian, instead of belonging to a completely different language other than the three we are looking at? Řehůřek and Kolkus try to solve this issue by assigning various minimum probability threshold values to each language. Assuming, for example, that this threshold were -4 for Hungarian, this would mean that the segment examined in Table 5 (barely) passed it and this language would be assigned to the segment. If none of the average scores for the known languages passed their respective threshold, the system would output that the target segment was in an unknown language. However, Řehůřek & Kolkus find that such thresholds simply do not exist. Based on our own research we can confirm that this is indeed the case and thus solving the problem of unknown languages in this way does in fact appear impossible.

However, this does not entail that n-gram-based methods are intrinsically unsuited to distinguishing known from unknown languages. One way of solving this problem, which we have incorporated in our algorithm, is based on the idea that the fundamental distinction is to be drawn not between known and unknown languages, but between input segments ('texts' in the terminology of Řehůřek & Kolkus) that clearly belong to exactly one of the known languages and ones that do not. The latter 'negative' category encompasses segments in unknown languages, but also mixed-language segments (containing words from more than one known language), ones consisting mostly of non-linguistic material, or ambiguous ones. The recognition that is crucial to dealing with such segments – and, to our knowledge, has not been noticed in the literature so far – is that all of these categories will be assigned relatively close probability scores by the models of all or at least several 'known' languages. A segment entirely in an unknown language will be assigned similarly low scores based on several models.[13] One that consists of words from two known languages will receive partly high and

[12] We could of course choose defaults that are far below the threshold, e.g. -1000, but this would be counter-productive as the recognition rate is extremely poor in that case.

[13] It might of course be the case that the unknown language in question is very similar to a 'known' language, especially one that it is historically closely related to. In this case the model of the latter language will assign a relatively high score to a segment in this unknown language, whereas it is assigned low scores by all other models. Under these circumstances the unknown language would likely be misclassified as the known language, which can only be prevented by training the system for this unknown language as well. In such a

partly low scores from both relevant models, which results in a mediocre average score for both languages. Finally, non-linguistic material, especially numbers will receive relatively high scores from all models. The same will characterize segments that can equally well belong to more than one language, e.g. the word *international* could be either German or English and does not belong to exactly one language. Thus what we need in order to distinguish segments that are clearly in a known language and those that belong to this negative umbrella category is not an absolute minimum threshold that Řehůřek & Kolkus were looking for, but rather a minimum threshold for the distance between the best-scoring language model (regardless of how high its score is in absolute terms) and the second best. If the distance (i.e. absolute value of the difference) between the best and the second best score is less than this minimum threshold, then the language of the target segment is 'other', meaning 'unknown/uncertain/ mixed/ambiguous or otherwise not identifiable'.[14]

Although it is not at all self-evident that using such a minimum distance threshold should indeed work, we have found that extending the basic algorithm in this way leads to remarkably good results in practice while representing an extremely small overhead: it consists in finding the best and second best average value, a subtraction and a comparison, all of this just once per segment.

3.4 Parameters

As is evident in the description of our method above, the functioning of the algorithm depends on the value of a number of parameters. These parameters are the following:

- length of the n-grams in the language model or order of the Markov model;
- the estimated probability value (see the third column in Tables 3 and 4) below which an n-gram is excluded from the language model, i.e. the minimum probability threshold;
- the fixed default value that all n-grams not listed in the model, i.e. those below this minimum threshold, are assigned;
- length of the segments that are analysed, e.g. 9 characters in case of the example in Tables 5 and 6;
- given the average probability scores for the segment under examination for all language models, the minimum threshold value that the distance between the best and the second best score must surpass for the segment to be assigned the best-scoring language (if this condition is not met, the segment is not assigned any language but classified as 'other').

situation the problem arguably lies in the fact that we are not dealing with separate languages at all but rather dialects, and the algorithm performs as expected.

[14] This raises the question whether it is possible in any way to distinguish more specific types within this umbrella category, e.g. to identify automatically whether a segment is mixed-language or unknown-language. This is certainly possible, although only to a very limited degree: Texts that are not only in an unknown language but are written in an untrained-for script or are not linguistically coded in a known writing system – but are instead e.g. binary files, ciphers, random junk, etc. – can be identified as a separate category since their average probability per n-gram will be very close (e.g. within the minimum threshold) to the default probability assigned to unlisted n-grams. Apart from this, we have not examined the question of automatic subclassification within the 'other' category in detail, but we have not seen anything indicating that this could be done, nor do we see a practical point in pursuing this question further.

Setting these parameters correctly is absolutely crucial for the recognition to work. Different settings are needed for different numbers of training languages, i.e. a system that is trained to differentiate between six languages works with very different values from one that should distinguish just two. The choice of languages, or more specifically the degree to which their models overlap also has a profound effect on parameter choice. Very different settings are needed in a system that is trained to distinguish between Hungarian with and without diacritics (as two separate recognition 'languages') versus a system that is supposed to tell apart Hungarian and English. The choice of the length of the n-grams or order of the Markov model does not depend on extrinsic factors in the same sense, and can be chosen freely (sensible values are 2 to 5); however, this choice in turn has a very profound effect on the ideal value of the other parameters, as well as on the quality of the results (see Section 4.2).
The ranges of these parameters are clearly predetermined to a certain extent. For instance, the minimum probability threshold will obviously not be larger than the most frequent n-gram (i.e. -1.8 in Table 3) and no smaller than the estimated probability of the hapax n-grams in the training corpus (i.e. around -8.5 for the same table). The fixed default value will not be larger than this threshold, but should probably not receive a value that is much lower than what the hapax n-grams would receive based on their observed frequency. We would normally prefer segment length to be as short as possible, but there is an inherent low boundary below which the idea of language identification as such does not make much sense, around 5 or 10 characters. Since a language identification method that is unable to assign the correct language to a long sentence reliably is not much good, we can assume about 150 to 200 characters for the upper boundary on segment length. We should note at this point that because the algorithm is based on calculating averages, the choice of the other parameters very strongly depends on the chosen segment length. Finally the distance threshold will necessarily be greater than zero and smaller than the distance between the value of the most frequent n-gram and the minimum probability threshold.

Apart from these obvious boundaries we do not see a way to calculate good or ideal values for the parameters. A feasible alternative, however, is choosing parameters by trial and error that work well enough for a given combination of language models and segment length. To automate this process of guesswork we used a small set of 24 texts, each in a different language and about 500 words long on average. These 24 languages included those that we have trained our system for, plus a number of other languages partly with different scripts (Japanese, Greek, Bulgarian), partly related to some of the recognition languages (Dutch, Spanish, Portuguese, Romanian; as well as Latin and Esperanto), and partly unrelated or only remotely related (Finnish, Irish, Latvian, Kurdish, Turkish, etc.). After a set of specific parameter values has been chosen, the language recogniser is run on this small corpus of roughly 10,000 words. Each segment that is either assigned the correct 'known' language, or 'other' in case of the unknown languages, counts as a success. The whole pass receives a score based on the number of successfully identified segments. The recogniser is run some 10,000 times (each time with a different combination of parameter values), which takes a couple of hours. The parameter values yielding the best scores are recorded in the process and output. Since this set of documents is relatively small, the risk of overtraining the recogniser, i.e. optimising the settings so that they work extremely well with this specific small set of documents but nothing else, is relatively high. Thus we have found that what leads to the best practical results is choosing parameters which receive very good, but not the absolutely highest scores during these optimisation passes.

4 Evaluating the Performance of Our Method

4.1 Setup of the Experiment

In order to test and quantify the performance of our method, as well as compare it to other approaches in the literature, we compiled three test corpora. The Hungarian, German and English corpora contain slightly more than 50 thousand words or about 300,000 to 370,000 characters each. While compiling these corpora we took special care to only include texts that belonged homogeneously to the language in question, i.e. without long stretches of foreign-language texts (quotations, references, etc.), lists of foreign names (like in sports texts), long symbolic or numeric sections (e.g. scientific, business or technical texts containing calculations, formulae, price lists, etc.). There was no overlap whatsoever between the training corpora that the language model was based on (cf. Section 3.2), the optimization set that was used to determine good choices for the algorithm's parameters (cf. the end of Section 3.4), and the test corpora that the experiments below were run on.

We tested our method both with simple n-gram and Markov models, as explained in Section 3.2, in order to examine whether one performed better than the other. We tried both model types with n-gram lengths/orders between 2 and 5.[15] In all cases the system was trained for 6 languages (Hungarian, German, English, French, Italian, Polish), each on a training corpus containing an amount of text at the order of magnitude of 100 million words. We tested each of these 8 model versions on target segments that were 10 to 150 characters in length. Each of the 8 models requires separate sets of parameter values. Since very short and very long segments also demand different parameters, we worked with two parameter sets for each model: one set for the short segments (10 to 50, optimised for a length of 30), and another for the longer segments (60 and above, optimised for a length between 70 and 90).

4.2 Results

The following figures show the results of this experiment. The horizontal axis shows the length of the segments, whereas the vertical axis indicates accuracy. The accuracy is calculated as the mean of the results of the three languages: 100 per cent means that all segments in the Hungarian, English and German texts were assigned the respective language.[16] Figure 3 shows the results for the simple n-gram models (n-gram length 2 to 5), and Figure 4 for the four Markov models.[17]

---

[15] We also considered using n-grams of length 6, but since memory requirements for the models grew to a huge size at this point and there was no observable improvement in the accuracy of the system as compared to length 5, we did not pursue this option any further.

[16] Note that (as opposed to e.g. the experiment reported in Řehůřek & Kolkus 2009) it would not make sense in our case to directly differentiate between separate precision and recall scores since our algorithm always assigns exactly one label – either the name of a trained-for language or 'other' – to every segment. However, assuming that we consider 1) cases when the system assigns a specific language to a segment, and that language is correct, as true positives, 2) cases when the system assigns an incorrect language to a segment as false positives, and 3) cases when the system assigns 'other' as negatives, then in this sense our precision rate is always extremely high (above 97 % on average even for the shortest segments for the 4- and 5-gram models, as we will see in connection with word recognition scores at the end of this section below), and the graphs in Figures 3 and 4 essentially approximate the recall rates.

[17] A slight drop (or slower than expected growth) in performance can be observed in both diagrams for length 40. The reason for this phenomenon may be that, as we indicated above, the parameter settings we used for

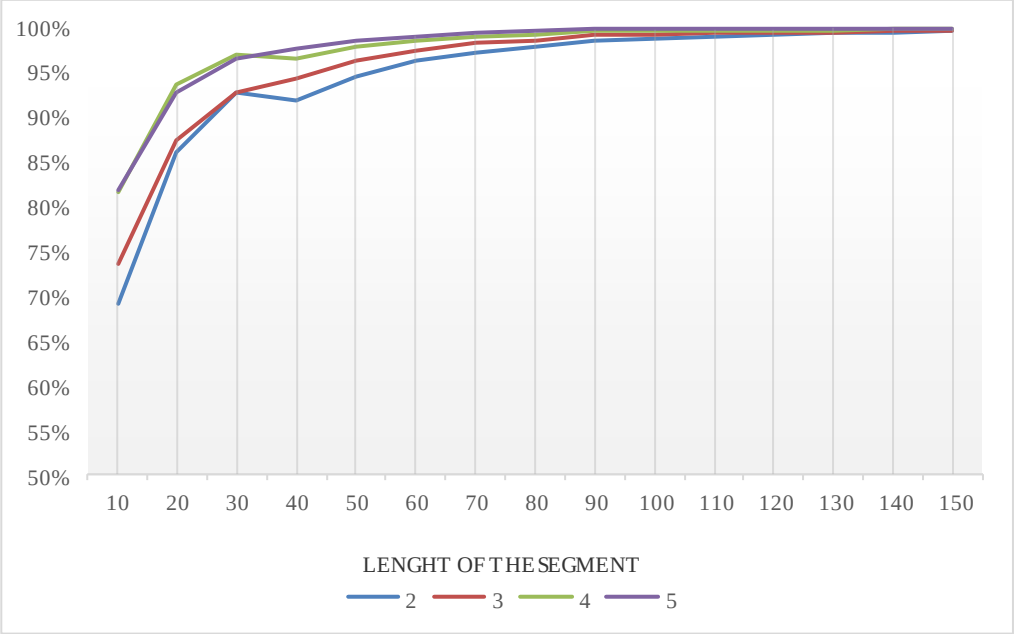

Figure 3: Recognition accuracy for simple n-gram models

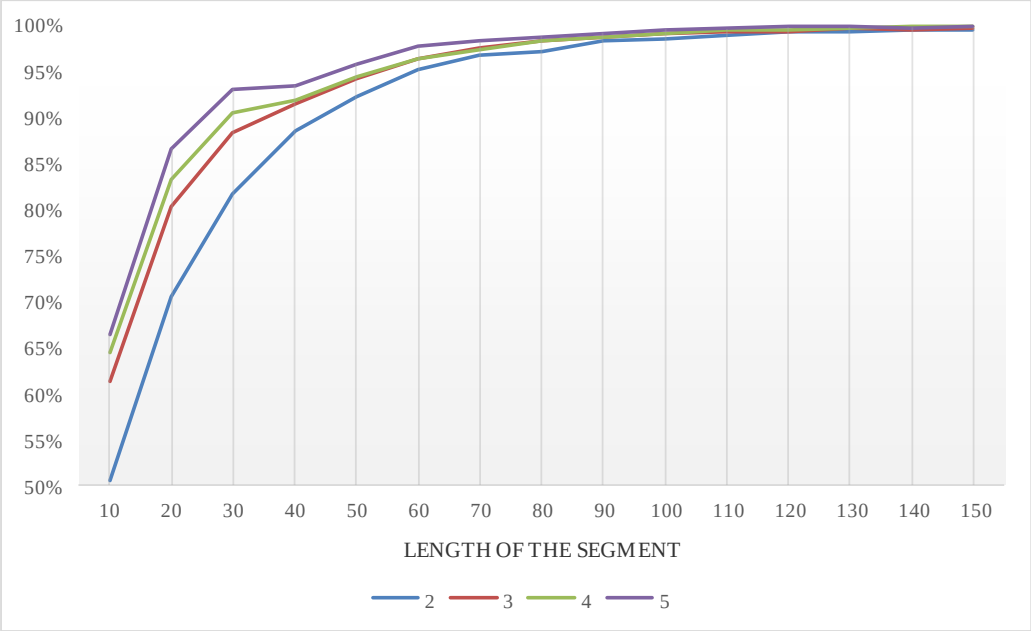

Figure 4: Recognition accuracy for Markov models

short segments, including length 40, were optimised for segments that were 30 characters long. Thus the scores at length 40, as well as 10 and 20, are slightly worse than what could be optimally achieved if we had adjusted the parameters for these lengths specifically.

Based on these two diagrams we can summarise the conclusions of our experiment as follows:

- Raising the length of the n-grams in the model results in improved accuracy with both types of model and all segment lengths. Bigrams are much worse than all other choices, trigrams are clearly better, and we get best results with 4- and 5-gram. If we also take the memory requirements of these models into consideration, 4-grams appear to represent the ideal trade-off between accuracy and resource use, yielding scores that are far better than those for trigrams, and only barely noticeably worse than for 5-grams, but with significantly lower memory use. As we have already mentioned in connection with Table 1, this directly contradicts Grothe, De Luca & Nürnberger's results, who claim that 4-grams and 5-grams work drastically worse than 3-grams.
- As we increase the length of the segments to be recognised, the accuracy measurably improves up to a length of about 100 characters (110 for n-grams, 120 for Markov models). Improvement is very steep between 10 and 30 characters and very slow above that. Accuracy for segments longer than 100 characters is, to all intents and purposes, perfect.
- Simple n-gram models work better than Markov chains across the board. This is especially apparent for shorter segments: for 10 characters the worst result using 2-gram models is nearly 70 per cent accuracy, whereas 2nd order Markov chains give rise to only 50 per cent.
- Comparing our results to those reported by Řehůřek & Kolkus (see Table 2 above), our n-gram-based method seems to perform markedly better. The average of their precision rates for small texts (10 to 30 characters) is 75.61 %, whereas their recall rate is 87.99 %. By comparison, our best accuracy rates (achieved with a 4-gram or a 5-gram model) are 84.84 %, 93.66 % and 97.09 % for lengths 10, 20 and 30, respectively. In particular, assuming that their small texts are about 20 characters long on average, and that their recall score is closest to what our accuracy rate expresses (see footnote 16 above), we see a difference of more than 5 percentage points (87.99 vs. 93.66) in favour of our n-gram-based solution. Precision, as explicated in the same footnote, is drastically better, by more than 20 percentage points, although this is not directly visible in the diagram.[18]
- For medium-length texts, i.e. those between 30 and 300 characters in length, Řehůřek & Kolkus report an average precision of 92.2 % and recall of 98.1 %. By contrast, our accuracy is no worse or in fact better than this even at the lowest end of this range (97.65 % at length 40, 98.49 % at length 50 for the 5-gram model). At segment length 60 our results reliably surpass 99 %, and above length 110 even 99.9 %, and are thus again far better than the rates that the authors' method manages to achieve for long texts of more than 300 characters (average precision 97.54 %, recall 98.94 %).
- Comparing our results to those reported in Dunning (1994), as cited in Section 2 above, our solution also fares relatively well. Not only is it able to handle unknown-language and mixed-language target segments, which Dunning's method would always classify incorrectly, but it also performs better (Dunning reporting 92 %

[18] Note that the scores they report for their version of the n-gram method, i.e. 83 % precision and 66.5 % recall for short texts, are also strikingly worse than our results.

accuracy for 'segments' of length 20 and apparently reaching 99.9 % only for texts that are 500 characters long).

Although the results for segment lengths above 70 as displayed in Figure 3 are already nearly perfect, they even underestimate the true level of accuracy. Almost all segments to which the recogniser does not assign the language of the subcorpus in which they appear are classified as 'other' rather than specific incorrect languages, i.e. the overwhelming majority of the mistakes are false negatives instead of false positives. This classification is expected to be incorrect since the texts in the test corpora are supposed to be linguistically homogeneous, but if we examine specific segments classified as 'other', it becomes clear that this label is in fact accurate in many cases. The following examples show typical segments (70 characters long) from the Hungarian test corpus which are classified as 'other' by the 5-gram model:

- Political Warfare Executive (PWE) - felügyelte a BBC Világszolgálatá
- lődést mutat. Zoubeir Chaieb, a tunéziai Advanced e-Technologies elnök
- jci Neue Zürcher Zeitung (NZZ) nemzetközi kiadása Tar Pál volt washing
- látható A Bigger Splash: Painting After Performance című tárlaton a Ta

All four segments are mixed-language, the first and fourth being Hungarian and English, the second containing a Tunisian name in addition, and the third Hungarian and German with part of an English name at the end. In all four cases at least half of the segments is not Hungarian. Thus these segments in fact belong to the category 'other' by its definition, i.e. we are dealing with 'true' rather than 'false negatives'. Although the tested system does make real mistakes, their number is in fact lower than the figures above indicate.

In addition to this experiment that was carried out on a relatively large test corpus, we also examined two specific performance characteristics of the algorithm using smaller, more ad hoc test corpora: how our method performs when the task is to recognise the language of individual words, and how reliably languages that have not been trained for are recognised as 'other'. These are discussed in the next two sections.

4.3 Word Classification Performance

In this small experiment we examined how our method fares if it is used to classify individual words in tokenised target documents. For the purposes of this experiment we defined a word as a string of characters which begins and ends with a single whitespace character, does not contain any other whitespace, and contains at least one alphabetic character. This naturally means that punctuation directly preceding or following an alphabetic character is considered part of the word. For this experiment we used simple 5-gram models, and the test corpus was the same as the optimization corpus from Section 3.4. The following figure shows the results.

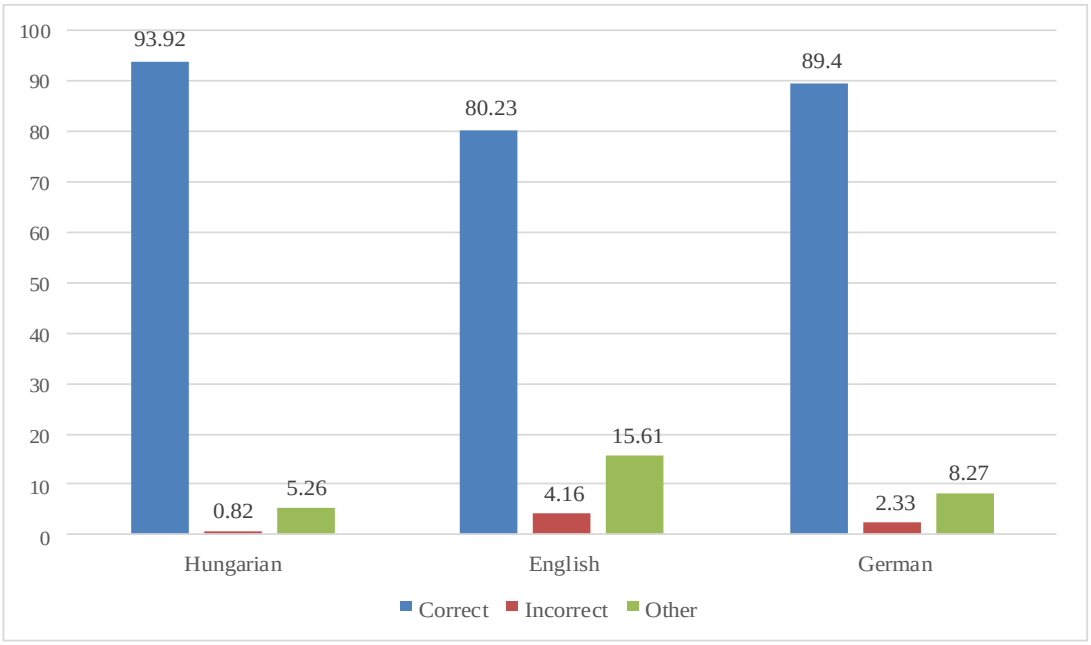

Figure 5: Recognition accuracy for word segments using 5-gram models

Since average segment length for words (adding two whitespace characters at the left and right end) is slightly below 10 characters, it is not surprising that the rate of accurately identified items is close to the score for segment length 10 as seen in Figure 3. It is also expected that the recognition rate for English, which shares a large part of its vocabulary with three of the trained-for languages (German, French and Italian), is far lower at approximately 80 % than for Hungarian (at almost 94 %), the vocabulary of which is much less international and which has a large number of specific characters with diacritics.

What is interesting, however, is the distribution of the mistakes. As we have mentioned earlier, the number of cases where a language is incorrectly assigned is extremely low, just around 2.5 % on average, although with a large variance: about 1 % for Hungarian and 4 % for English. Compared to the total number of unsuccessful classifications (i.e. the sum of incorrect language assignments and 'others') the proportion of these 'false positives' is around 20 %, the 'others' (i.e. 'false negatives') making up the remaining 80 %. Taking into consideration the fact that the words classified as 'other' could often really belong to more than one trained-for language, and thus many of them are in fact 'true negatives' rather than mistakes, we believe that the performance characteristics are very convincing even at the word level. In particular, the n-gram-based solution is an excellent alternative to the dictionary method for languages like Hungarian, where using (at least full) vocabulary lists is impractical because of the huge number of potential inflected forms for each word.

4.4 Classification of languages that are unknown to the system

Although we have claimed that one important benefit of our method is that it can identify correctly that a language is 'unknown', rather than classifying it as one of the training languages like traditional n-gram-based approaches do, we have yet to see specific figures

supporting this claim. For the next small experiment we have used another small corpus consisting of the same 24 languages as the optimisation corpus, but different texts (taken from Wikipedia). The average size of the texts in this corpus was approximately 2000 words. We used exactly the same parameters for the algorithm as in the experiment detailed in Section 4.2, and again we employed our best-performing model, i.e. the simple 5-gram model.

The following diagram presents the results for all 15 'unknown' languages with Latin-based scripts that were included in this corpus.[19] The percentage represents the ratio of segments that were classified by the system correctly as 'other', compared to the total number of segments that made up a text belonging to an untrained-for language in the corpus. For example, a value of 90 % for an 'unknown' language (e.g. Czech) means that 90 % of the segments constituting the Czech test document were identified correctly as 'other', whereas the remaining 10 % were incorrectly assigned one of the six 'known' languages.

In order to keep the figure as simple and informative as possible, we have only included two sets of data in this diagram instead of showing all 15 languages separately: the average accuracy for these 15 languages, and the worst value among them for the given segment length.

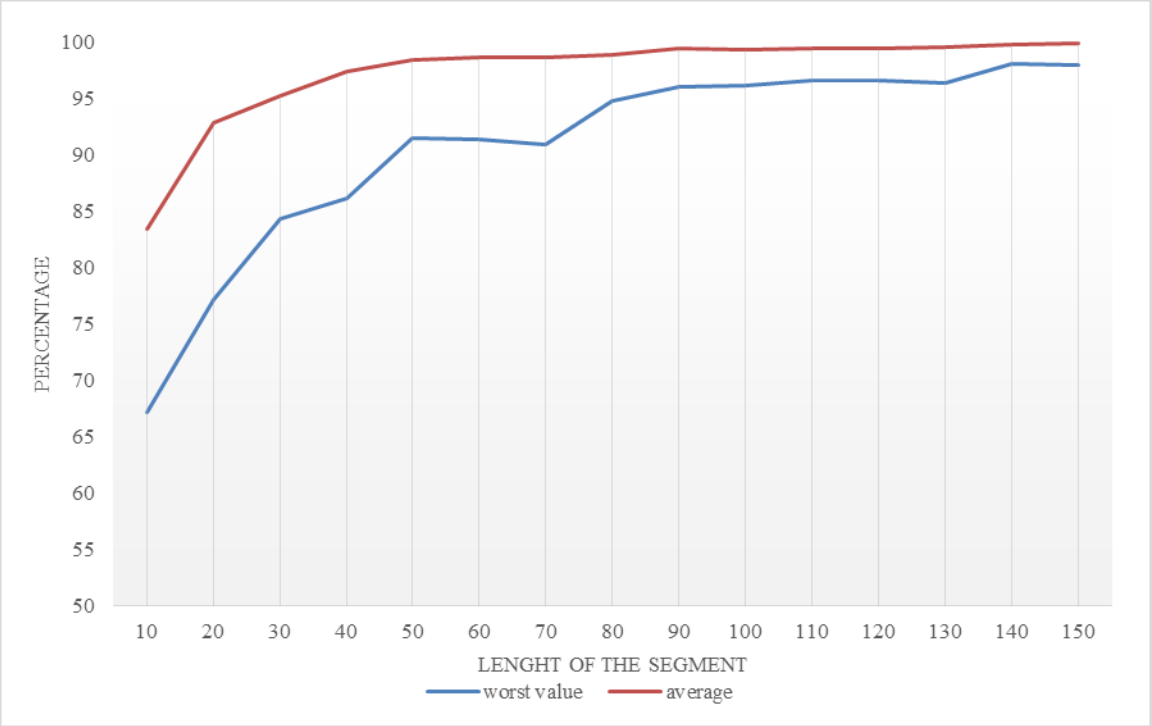

Figure 6: Recognition accuracy for segments in 'unknown' languages using 5-gram models

[19] We did not take into consideration the results for the Japanese, Greek and Bulgarian texts in our test corpus since these were recognised with 100 per cent accuracy for all segment lengths as 'other' because of their non-Latin script. This is an excellent result, of course, and it is reassuring that the system can reliably solve the trivial task of distinguishing between scripts, but what we are mainly interested in is how it fares with the non-trivial task of telling apart languages that use the same script.

For comparison, we have also included the corresponding average and worst accuracy scores for the six known languages in this test corpus:

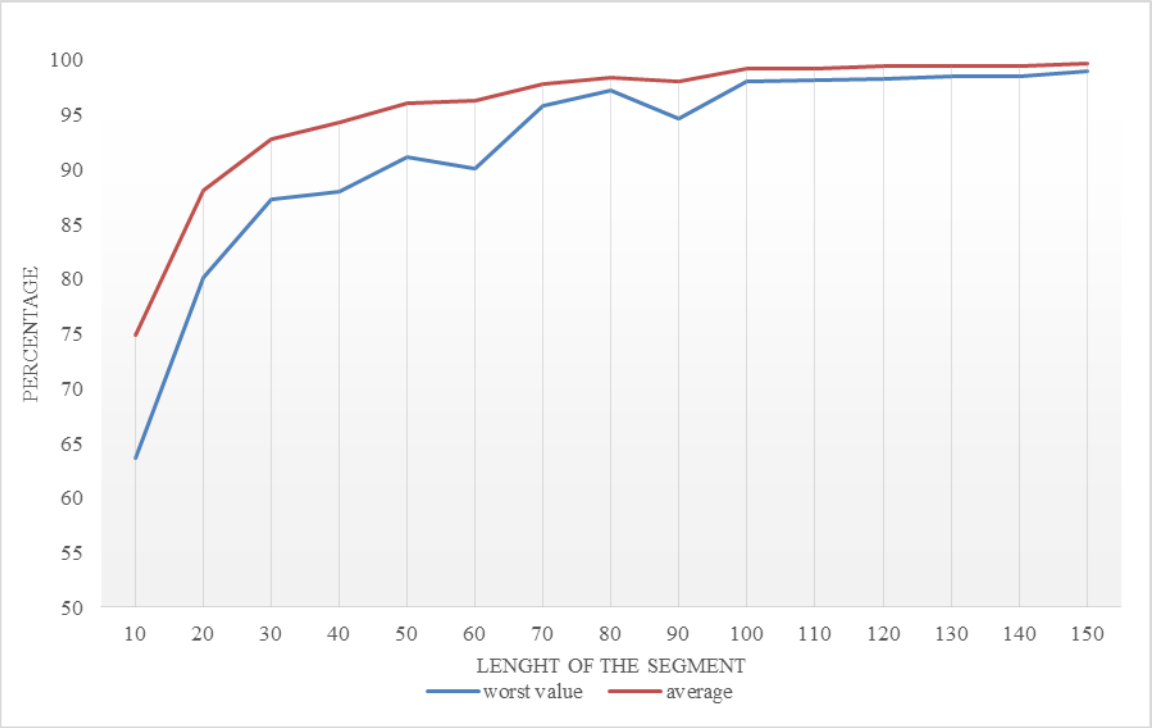

Figure 7: Recognition accuracy for segments of known languages using 5-gram models

According to Figure 6, the worst value for segment length 10 is about 67 per cent, which is slightly higher than the worst value for the known languages. This value of 67 per cent accuracy belongs to the Spanish language, which is explained by the fact that two other Romance languages have been trained for, and therefore many segments are mistakenly classified by the system as either Italian or French. For related reasons the results for Portuguese (72 %) and Dutch (70 %) are far below average at segment length 10, and remain worse than average for longer segments as well. Note that these low values do not mean that our method is inherently unable to recognise the language of Spanish, Portuguese or Dutch texts; after all, the accuracy of the recognition of the known languages French and Italian is slightly higher than average. Instead, what these numbers demonstrate is that if we want the system to be able to tell apart an 'unknown' language from one or more trained-for languages that are very similar to it, and we want reliable results even for very short segments, then the only way to ensure this is to train the system for that unknown language as well.[20]

The average score for the shortest segments is 83.41 per cent, i.e. better than the scores we see in Figures 3 and 7 for known languages. At segment length 20 the average surpasses 90 per cent, and at length 50 even the worst value reaches this mark. The average climbs to

[20] Another option would be to tweak the parameters of the algorithm, in particular the minimum distance threshold value, so that the system becomes more likely to assign 'other' rather than the name of a known language to a segment. This does indeed improve accuracy for unknown languages, but in turn reduces accuracy for the known languages. This trade-off is an essential characteristic of our algorithm.

approximately 99.4 % around segment length 90 and only improves very slightly for longer segments.

As Figure 7 shows, among the six known languages at the length of 10 the worst value (English) in this corpus is about 63 per cent, and the average is about 74 per cent, nearly 10 per cent lower than in the case of the unknown languages. The average reaches 90 per cent at length 30, the worst value at length 50. The 95 % correct mark is surpassed at length 50 (average) and 70 (worst), respectively. The average reaches 99 per cent at length 100, whereas the worst value misses this level slightly even for the longest segments. As explained at the end of Section 4.2, the real accuracy is certainly better than what these scores show, since the Wikipedia articles used for the experiment contain foreign names, titles, citations, etc., and thus part of the segments classified as foreign or 'other' do in fact belong to a different language or contain mixed language. Indeed the reason why the scores are lower in this experiment for the known languages than in Figure 3 is presumably that the test corpus was carefully filtered for texts containing mixed-language material in that case, whereas we are dealing with more 'natural' encyclopaedia articles here. In other words, it is probably not the performance of the system that is worse for this corpus, but the linguistic material in the corpus is simply less homogeneous, which is correctly registered by the system.

Thus we can conclude that there is no striking difference between the accuracy of the recognition of known and unknown languages: even if very short segments are being identified, the algorithm can determine with a high accuracy that part of a text in an unknown language does not belong to any of the trained-for languages.

5 Conclusion

In this paper our main goal was to present a new language identification method based on traditional n-gram language models. It is straightforward to implement, very simple and fast, scales well with the number of recognition languages, and is able to handle tasks that similar algorithms proposed in the literature fail at. In particular, it can deal with input texts in unknown languages, in more than one language, and those containing non-linguistic material (like program code, binary code, numbers, etc.). To construct the models on which this method is based, what is needed is simply raw corpora in the recognition languages; no other resources such as dictionaries, morphologies, etc. are necessary, nor any degree of human supervision. The method already works very reliably for target strings that are 40 or 50 characters in length and reaches almost perfect accuracy around 100 characters. This permits a very easy and natural solution to handling multilingual texts: They are divided up into short segments, and these homogeneous single-language segments are each assigned the correct language. In addition to the experiment presented in this paper, the same implementation of our algorithm has also been used in a 'production' environment to classify texts in corpora the total volume of which was at the order of magnitude of a billion words. We can therefore confirm that this method also works reliably in practice, and it is fast enough to be used in contexts where this is critical, being able to process even such huge corpora in a matter of hours, or at most one or two days using a simple desktop PC.

References

Cavnar, W. B. & Trenkle, J. M. (1994): N-Gram-Based Text Categorization. *Proceedings of the Third Annual Conference on Document Analysis and Information Retrieval (SDAIR).* Las Vegas, 161–175.

Dunning, T. (1994): *Statistical identification of language. Technical Report MCCS.* New Mexico State University.

Grefenstette, G. (1995): Comparing two Language Identification Schemes. *JADT 1995, 3rd International conference on Statistical Analysis of Textual Data*. Rome.

Grothe, L., De Luca, E. W. & Nürnberger, A. (2008): A comparative study on language identification methods. *Proceedings of the 6th International Conference on Language Resources and Evaluation.* Marrakech, 980–985.

Pataki, M. & Vajna, M. (2011): Többnyelvű dokumentum nyelvének megállapítása. *VIII. Magyar Számítógépes Nyelvészeti Konferencia*. Szeged, 3–11.

Poutsma, A. (2001): Applying Monte Carlo techniques to language identification. *Proceedings of Computational Linguistics in the Netherlands*, 179–189.

Prager, J. M. (1999): Linguini: Language identification for multilingual documents. *Proceedings of the 32nd Hawaii International Conference on System Sciences*.

Řehůřek, R. & Kolkus, M. (2009): Language Identification on the Web: Extending the Dictionary Method. *Proceedings of the 10th International Conference on Intelligent Text Processing and Computational Linguistics*, 357–368.

Sibun, P. & Reynar, J. C. (1996): Language identification: Examining the issues. 5th *Symposium on Document Analysis and Information Retrieval*. Las Vegas, 125–135.

Teahan, W. (2000): Text classification and segmentation using minimum cross-entropy. *Proceeding of RIAO 2000, 6th International Conference Recherche d'Information Assistee par Ordinateur*. Paris, 943–961.

Dr. Gergely Pethő
Institute of German Linguistics
University of Debrecen
Pf. 47
H-4010 Debrecen
pethog@inf.unideb.hu

Eszter Mózes
University of Debrecen
Doctoral School of Linguistics
Pf. 47
H-4010 Debrecen
dtotheszter@gmail.com