Machine learning algorithm
for automatic labeling and its application in text-to-speech conversion

GÉZA KISS, GÉZA NÉMETH

BME, Department of Telecommunications and Media Informatics
{kgeza, nemeth}@tmit.bme.hu

Reviewed

Keywords: machine learning, language identification, LID, TTS

In this paper we present a novel machine learning approach usable for text labeling problems. We illustrate the importance of the problem for Text-to-Speech systems and through that for telecommunication applications. We introduce the proposed method, and demonstrate its effectiveness on the problem of language identification, using three different training sets and large test corpora.

1. Introduction

The number of speech-based telecommunication applications, which accept incoming calls through an IVR (Interactive Voice Response) system, is continuously increasing, both in Hungary and abroad. In the most modern systems the user can express his wishes not just by typing, but also through an ASR (Automatic Speech Recognition) system, and in response s/he will receive good quality speech. In the most simple case the response will contain prerecorded messages (so called "prompts") or messages assembled from a few separate items (limited vocabulary speech synthesis).

If the content of the message to utter is unpredictable or shows great variability, then we produce the speech using a Text-to-Speech system (TTS). A few examples for the latter from the services available in Hungary are the e-mail system of T-Mobile Hungary [1], its reverse directory system [2], or the loud SMS service of T-Com.

We expect TTS systems to generate good quality, understandable and correctly intonated speech using only the written form. But we can claim that the writing systems in use (whether it be Hungarian or of another language) contain only a fraction of the information content of speech and only hints at prosody using some punctuation marks. The human reader completes the text in his mind with the missing pieces of information using his world knowledge and the context, which helps him to read it out also (if necessary). For just determining the proper pronunciation and stress a system needs to correctly find out information such as the language of the text and of intruding foreign words and the role of the individual words in the sentence (part-of-speech, grammatical structure). In *Figure 1* you can see a few examples for Hungarian where this is not trivial because a different decision is needed even in the case of identical word forms or because the sentence contains a foreign language part.

In this paper we review the methods used for language identification from text, then describe a machine learning algorithm that learns from labeled text and can be used for various automatic labeling tasks. We explain it on the exemplar of language labeling, referring to the possibility of part-of-speech tagging; this kind of information is important for TTS programs, which are increasingly used in telecommunication also. Besides these, the method can probably be used in diverse other areas.

2. Methods for language identification from text

The notion of "Language Identification" (LID) can refer to the methods used for identifying the language of speech before ASR or to those used for identifying the language of texts. Language identification can be viewed as a special case of classification (or labeling) problems, so the lessons learned here can be applied for other similar problems, e.g. for part-of-speech tagging.

Figure 1. Examples of non-trivial language tagging and part-of-speech tagging tasks

***"a test"*– Hungarian or English expression:**	A lélek és *a test*. [The soul and *the body*.]	This is *a test*.
foreign word in Hungarian text:	A "Sok hűhó semmiért" *Shakespeare* műve. ["Much ado about nothing" is *Shakespeare*'s comedy.]	
***"egy"* – determiner or indefinite article:**	*Egy* vagy két alma. [*One* or two apples.]	*Egy* alma esett le a fáról. [*An* apple fell from the tree.]

Although at first glance most people who are interested in the topic will have ideas for automatically determining the language of texts, several issues harden the problem and make it far from trivial. While we can assign the correct language to a longer text with high probability using fairly simple techniques, the correct identification of the words in a mixed-language text with word forms occurring in more than one language is a much harder task. In addition, the effective solution, besides being precise, must be fast and need relatively small storage place.

2.1. Morphological analysis

Some of the systems, endeavoring to have correct identification on the word level, or even morpheme-level, use detailed morphological analysis, e.g. using DCG's (Definite Clause Grammar) [3], or indirectly using a spell checker [4]. In an intermediate solution no real morphological analysis takes place; instead they infer the language of words by matching items of a dictionary (made up of words and sub-word units) against the text, augmenting this with statistical methods [5].

The Humor [6] and Hunmorph [7] morphological analyzers can be used for Hungarian. However, if the decision to be made is not simply "Is it Hungarian or not?", but you have to decide on one of several potential languages, then you need a morphological analyzer for each one. This is hard to accomplish, moreover the necessary processing power may be problematic in certain applications.

2.2. Word-based method

Word-based methods [8] rely on the observation that in every language there is a fairly small set of words that are used very frequently. Therefore the presence of such words from a language indicates with high reliability that the text was written in that language. The most frequent 1000 words can make up 50 to 70% of all occurring word forms [9].

A disadvantage of the method is that the shorter the text, the more likely it is that no words from the list of its language will occur therein. Additionally you need considerable effort to assemble the wordlist, partly because some of the word forms will occur in several of the most frequent word lists of languages, as you can see from the examples in *Figure 2.*

Figure 2.
Examples of frequent word forms occurring in several of five European languages

word form	English	German	Spanish	Polish	Hungarian
de			X		X
a	X		X		X
na				X	X
to	X			X	
in	X	X			
do	X			X	
el			X		X
is	X				X
es		X	X		
mit		X			X
was	X	X			
ha			X		X
so	X	X			
on	X			X	
mi			X	X	X
most	X				X
ja		X		X	
be	X				X
ma				X	X

2.3. Vectorspace methods

The basic idea of vectorspace methods is to assign feature vectors to the document being examined and to the possible classification categories, which contain the numeric value of some properties, weighted by the importance of the feature. The feature vectors for the classification categories are created from belonging sample documents. The similarity of the document to a category is characterized by the scalar product of the feature vectors, where 0 means ortogonality and 1 means identity. In the approach described in [10], the features are the number of n-grams in the texts with N=2...5 and the number of short words or words with unlimited length.

2.4. Neural networks

In the approach used in [11] a Multi-Layer Perceptron (MLP) is trained. The input of the network are the characters within a window placed at each character position of the words, the output is a language probability at the position. A language decision is made for each word by combining the probabilities for each letter.

2.5. Decision-tree-based methods

A decision tree (different from the one in our proposal in section 3. is used in [12]: they train a separate decision tree for each different character, where the branches have questions about the identity of neighboring characters, and the leaves contain the most probable language. They make a word level decision, choosing the language candidate that was voted most often.

2.6. N-gram-based methods

N-gram based methods use sub-word items as the basic units for identification. These can be letter sequences of two, three or more characters or sequences of different lengths simultaneously. N-gram frequency statistics can be created from training corpora prepared for the languages.

N-grams are good at handling several problems that word-based solutions handle poorly. One of these is the frequent presence of spelling errors in electronic texts (coming from mistyping or OCR errors), as these have such great variability that they cannot be handled by storing all

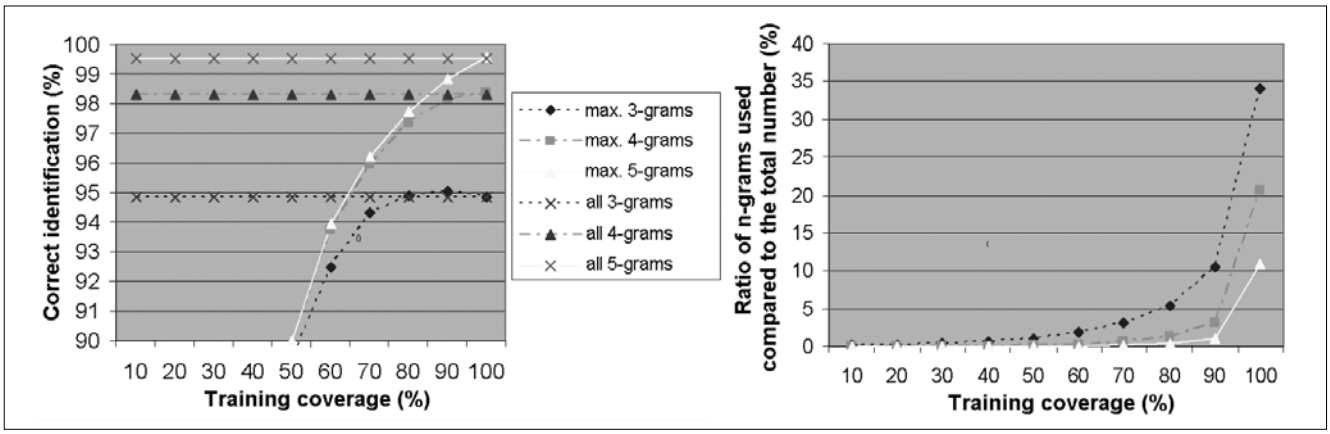


Figure 3.
Comparison of the performance and database size of the methods on the training set using ML estimation and either fixed or variable length character context

the incorrect word forms, but do not spoil the n-gram statistics very much. Another problem is data sparseness (i.e. you practically cannot collect sufficient data to have distribution information about all items that can occur; you will always encounter some that was not seen in the training corpus). This problem will be present to a much lesser degree than when working with words as the number of n-grams in a word is proportional to the square of the word length. A characteristic example of this approach is the method of Canvar and Trenkle [13].

2.7. Markov model

We know that we can get the exact word probability for a word of *l* characters using the chain rule of probability:

$$\mathrm{P}(\mathrm{word}\,|\,\mathrm{language}) = \prod_{i=1}^{l+1} \mathrm{P}(c_i \,|\, c_0 \ldots c_{i-1}, \mathrm{language}) \quad (1)$$

where $c_1,\ldots,c_l$ are the characters of the word, $c_i, i \leq 0$ and c_{l+1} are special word-starting and word-ending characters.

This probability can be approximated using a Markov model, i.e. assuming this is a random process where the probability distribution of the next character depends on the current state only. Traditionally the state is defined to consist of the previous *n-1* characters for LID (2):

$$\mathrm{P}(\mathrm{word}\,|\,\mathrm{language}) \approx \prod_{i=1}^{l+1} \mathrm{P}(c_i \,|\, c_{i-n+1} \ldots c_{i-1}, \mathrm{language})$$

We can approximate the conditional probability of characters after a given context using the ML (Maximum Likelihood) estimation:

$$\mathrm{P}(c_i \,|\, c_{i-n+1} \ldots c_{i-1}) \approx \frac{\#\, c_{i-n+1} \ldots c_{i-1} c_i}{\#\, c_{i-n+1} \ldots c_{i-1}} \quad (3)$$

Since we can expect that previously unseen n-grams will occur no matter what the size of the training set is, the use of some kind of smoothing technique is inevitable. The choice of this considerably determines the quality of the approximation.

The probability of the word in the language can in turn be used to estimate probability of language:

$$\mathrm{P}(\mathrm{language}\,|\,\mathrm{word}) = \frac{\mathrm{P}(\mathrm{word}\,|\,\mathrm{language}) \cdot \mathrm{P}(\mathrm{language})}{\mathrm{P}(\mathrm{word})} \quad (4)$$

When determining the most probable language, we can ignore the denominator (as this is characteristic of the input text and not of the language). We can regard the probability of the language as a constant value or one changing dynamically based on the context (5):

$$\mathrm{language} = \arg\max_{\mathrm{language}} \mathrm{P}(\mathrm{word}\,|\,\mathrm{language}) \cdot \mathrm{P}(\mathrm{language})$$

In theory the Markov model estimation in combination with a good smoothing technique can give arbitrarily good estimation if *n* is large enough. (If *n* is the maximum word length, we get the chain rule probabilities, i.e. the exact word probabilities for the training set.) But in practice generally *n*=2 (bigrams) or *n*=3 (trigrams) are used for two main reasons: because of the data sparseness problem and because larger *n*'s would need significant storage capacity.

Unfortunately such lengths do not allow for correct classification when deciding on multiple languages, as shown in *Figure 3* for the training set described in the first line of *Table 1*; these values are the theoretical limit for correct identification. As we can see, when training and testing on the same set, the correct identification rate does not reach 100% even for 5-grams but this would need a rather large database, furthermore it does not yet contain the smoothing inevitable for unknown texts. Using the method proposed in section 3., the size of the database is 10% to 35% of the previous with similar identification rates, plus it gives slightly better results when training for 100% of the training corpus.

Training (words)	Languages	Database	Training, word	Test, word	Test, sentence
2-9 million words	3	54 Kbytes	99,6%	94,2%	98,5-99,5%
600-700 words	3	7,4 Kbytes	95,5-97,8%	79,6-87,4%	91,7-97,2%
500-1700 words	77	5,4 Mbytes	70,0-99,8%	30,1-59,6%	71,0-84,0%

Table 1.
Results for different training sets with an independent test set for three languages, for word and sentence level identification

2.8. Summary

In this section we overviewed some of the methods used for language identification from text. The two main groups thereof are detailed analysis and statistical methods.

We can conclude that the purely statistical methods at present do not yield precise language identification on short texts in general; therefore they are not trustworthy enough for word level classification. Moreover, those that compare the document to so-called "language profiles" previously generated from training text corpora for the languages often need significant computing power in the identification phase also. In contrast, detailed morphological analysis is hard to accomplish, especially for a large number of languages, and the necessary processing power may be too high for some applications.

3. The proposed method

The method described below was first devised for the task of language identification from text; therefore we shall discuss it from this perspective. But by substituting "class" for "language" and "text unit" for "word", you can read it as the description of a general text labeling method.

3.1. Basic principle

Our aim was to create a method that has the ability to identify correctly even short sentences (down to one word), and can be held in hand in the sense that the system can be trained to correctly identify any required input, while it retains its ability to generalize, i.e. to identify unseen words correctly based on similarity to known training words. We also wanted to restrain the size of the resulting database.

A suitable way for this is to approximate P(word | language) with a precision set by a predefined criterion - e.g. with enough precision for the calculated probabilities to be the greatest for the language whose probability is greatest based on the training set. Another element of our method is to calculate language probability for every word based on its context and to decide on the language according to equation (5).

This approach makes it theoretically possible that we get correct language identification on word level, even in the case of homomorphs (in our case word forms belonging to several languages at the same time), furthermore that inserted foreign words can be labeled with their real language, rather than with the naive approach that deterministically decides based on the context. If one decision is needed for the whole piece of text (e.g. for a sentence), we can decide using the language labels determined for words, e.g. using the principle of majority voting.

This approach retains the advantage of the word-based method, i.e. it can be fully controlled, and extends it with generalizing abilities, allowing for correct word level identification. Using an appropriate probability estimation technique we may be able to estimate this probability for previously unseen words based on the spelling of known words. The key for the success of this method is the estimation of the conditional probabilities and language probabilities with enough precision.

3.2. Estimating conditional probabilities

The method we worked out for estimating P(word | language) relies on n-grams of variable length. While traditional LID solutions with Markov-model use a fixed length character context for estimating the conditional probability of the next character, in the proposed method we use variable length character context. (6)

$$\mathrm{P}(\mathrm{word}\,|\,\mathrm{language}) \approx \prod_{i=1}^{l+1} \mathrm{P}(c_i \,|\, c_{i-n_i+1}\ldots c_{i-1}, \mathrm{language}), n_i \geq 0$$

We determine the n_i lengths in the course of a training process. Training starts out with a 0 length character context for every character (this is the occurrence probability of the character), then increases this length in certain contexts for attaining the predefined probability estimation criterion, which can be e.g. the correct identification of a certain percentage of the most frequent words of the training set. Continuing the process without a limit yields the chain rule, and through this the word-probability (assuming an appropriate smoothing technique), therefore the training process converges to the correct identification for any training set. Using a larger database with longer n-grams will give more accurate identification results. Therefore the method is scalable, as one of the database size-desired identification rate-pair can be chosen freely.

Storing the conditional probabilities belonging to the n-gram contexts in a tree, we can view the method as training a kind of decision tree for the purpose of word probability estimation. The direction for expanding the tree is decided based on the "usefulness" of the expansion in regard to the estimation criterion. We worked with several such usefulness functions, examining them with regard to the correct identification rate on the training set, the results on an independent test set (i.e. generalizing ability), and how concise the resulting database is. We did not characterize conciseness simply with size – as it is not indifferent what identification rate a smaller database produces – but with the identification rate/size quotient, which we called the performance of the LID-database. We can see the best usefulness functions and the graphs that describe them in *Figure 4*. As we can see, different usefulness functions are needed for the best generalization ability and for the most concise database.

Another novelty is that instead of observing languages separately, we endeavor to estimate the conditional probabilities difference between languages with enough precision. From this we expect smaller database size, since this way we make the algorithm "concentrate" on the features that distinguish the languages.

In so doing the algorithm is not only scalable, but automatically scales itself to the complexity of the problem. E.g. if two languages have different character sets, then even if we aim at 100% correct identification, training stops at using unigrams (1 character long n-grams).

3.3. Using language probabilities

3.3.1 The notion of "language probabilities"

By "language probabilities" we mean the probability that a word with a certain language will occur in a given context; we use this in (5) in the place of P(language). The features that can be used in the estimation of this probability in our solution are the language of the surrounding words and the punctuation between them. We have chosen to include these among possible questions to ask because these seem to play a role in forming the human interpretation, but a different, smaller or broader set of questions is also possible.

We can view the modeling of language probabilities as training a decision tree for storing conditional probabilities similarly to the type given in 3.2., but it is different from that (and the generally used methods [15,16]) in that we do not only use the previous words, but the following ones also.

The difficulty in this approach is partly that we presume to know the language of the words around the word in question, which is not fulfilled in practice, and partly that, because of the large number of possible questions that guide the expansion of the decision tree during training, the number of possible alternatives is huge. We shall discuss the solution of these in the next section.

3.3.2 Finding the most probable label sequence

The first difficulty is a mathematically solvable problem, although the efficient realization of the calculation is not trivial. The task is to find the label sequence that maximizes the probability on the sentence made up of N words.

$$\{\mathrm{lang}_{l_i} \mid i \in [1..N]\} = \arg\max \prod_{i=1}^{N} \mathrm{P}(\mathrm{lang}_{l_i} \mid \mathrm{word}_i) = \tag{7}$$

$$= \arg\max \prod_{i=1}^{N} \frac{\mathrm{P}(\mathrm{word}_i \mid \mathrm{lang}_{l_i}) \cdot \mathrm{P}(\mathrm{lang}_{l_i})}{\mathrm{P}(\mathrm{word}_i)}$$

We can disregard the $\mathrm{P}(\mathrm{word}_i)$ coefficient here also, since it does not depend on the language, therefore it does not affect the result.

Finding the most probable label sequence with exhaustive search with L possible labels would mean L^N calculation steps, which defines a search space that grows exponentially with the length of the sentence; there are not many practical applications that can allow this. Therefore we need to find a method that approximates the optimal solution in some way.

The approximation used currently in our system is the following: first we calculate a label sequence with uniform language probability, and then using the language context received this way we recalculate the labels for the whole sequence from left to right. If a label is modified, then we recalculate the labels to its left that could be affected by the change, but (in order to avoid iteration) modify them only if the probability calculated for the new language label is higher than for the previous one. The algorithm can be improved, e.g. by using simulated annealing (allowing label modifications that do not immediately result in probability growth to a lesser and lesser degree).

3.3.3 Using rule templates

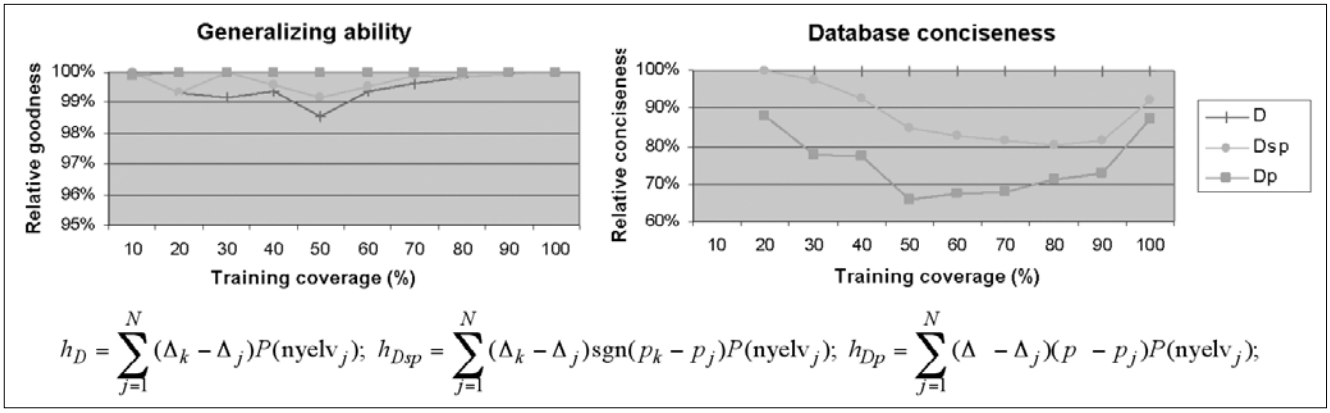
To avoid the second difficulty, in our sample system we use so-called rule templates, which gives a way for using linguistic knowledge and heuristics, and largely diminishes the number of alternatives to examine.

The rule templates can have the following form (in BNF notation):

<template> ::= { <label-description> [<separator-description>] }
<label-description> ::= L{[<label ID >][?|*] | "<label name>"}
<separator-description> ::= S{[<separator ID>][*] | "<separator>"}

The label is language in this case. Labels with the same label ID must be equal; the same is true for separators. The descriptions marked with an asterisk indicate items whose different values do not create independent rules, while those that are not marked create a separate rule for every different value of the label or separator.

Figure 4. Characterizing usefulness functions
(values are compared to the best result, training for diverse percentages of most frequent words);
N is the number of languages, k is the index of the real language of the word, p is the occurrence probability of the n-gram to add, Δ is the change in the probability value arising because of adding the n-gram.



$$h_D = \sum_{j=1}^{N} (\Delta_k - \Delta_j) P(\mathrm{nyelv}_j);\ h_{Dsp} = \sum_{j=1}^{N} (\Delta_k - \Delta_j) \mathrm{sgn}(p_k - p_j) P(\mathrm{nyelv}_j);\ h_{Dp} = \sum_{j=1}^{N} (\Delta_k - \Delta_j)(p_k - p_j) P(\mathrm{nyelv}_j);$$

The label-description marked with a question mark ("?") is the one for which we are observing the occurrence probability of the different labels. When creating the rules to use, we match the rule templates onto each word of the training set at this point, and where the template can be matched against it, we create a rule that is filled with the specific values found in the text. These rules will contain the occurrence probability of the labels at the end of the process (calculated from the number of places the rule could be used, and number of occurrences of the label types).

If several rules can be applied at a position with different number of conditions, then we use the one that has more conditions (following the principle of the decision tree). You can see examples for rule templates and the rules created from them later in *Figure 5*, for the topic of language identification.

4. Application to language identification from text

4.1. Assembling a training corpus

We can say that at present no large text corpus is available that is labeled with the correct language label on word level, moreover it is hard to assemble corpora that are actually monolingual (because of the foreign names, expressions and loanwords that are present in every language, plus foreign language parts or complete foreign texts find their way even into well-known monolingual corpora such as Project Gutenberg). Therefore assembling training sets for LID is also difficult despite the fact that vast amounts of text can be downloaded from the internet for practically any language, but with the aforementioned mixed nature.

A way to solve the problem can be to train the LID system for several languages assuming the collected texts to be monolingual (or at least to have the nominal language in majority), and then to automatically label the texts using the resulting LID system. After omitting complete texts or sentences that are identified as clearly differing from the language of the training corpus, we can repeat the training process, this time with a cleaner text that better approaches monolinguality, until no improvement can be realized.

In case of a corpus that contains relatively short texts (e.g. the archive of a newspaper), the headers and footers, which are usually present in texts and are basically the same in each one, can notably distort word and n-gram statistics, as these multiply the number of occurrences of some (perhaps otherwise rare) words or expressions. For the theoretically correct operation it is worth removing these from longer pieces of text also. This also helps in cleaning the corpus from foreign language parts, as the header and footer is normally written in the language of the corpus even for incidentally retained foreign language documents.

We need to pay attention to the fact that the texts collected for a language may be coded with diverse character sets. In order to handle these correctly, we need to know the encoding of these texts. If we can expect to know the character set of the input to be identified, then we just need to convert the texts to a common coding (e.g. Unicode). If it is not known, we can train the system with texts in different encodings at the same time, or we can label the texts of the training sets with the same language but different encodings with a name that includes language and character set, so that we can identify these two features in one step.

Another problem is that in certain application areas (e.g. when working with SMS or e-mail messages) the diacritics, which are used in Hungarian and many other languages, may be missing from the letters, which can hinder language identification if we do not cope with this. Possible approaches are to use a training set that contains both kinds of text (i.e. with and without diacritics); or to convert the training set and the text to be identified to a smaller common character set (the one without diacritics) and during identification to modify the language probability based on the original character set of the word [11].

4.2. The test corpus used

We performed several test with training and test corpora of different sizes. First we trained the system for three languages (English, German, Hungarian) on large corpora (British National Corpus, Project Gutenberg DE, Hungarian Electronic Library), without cleaning them of the foreign language parts, to correctly identify 90% of the most frequent words. We did the testing on independent test sets (Project Gutenberg, online Hungarian Newspapers).

We also did the training on small texts (5 kilobyte) belonging to 77 languages that are used in an implementation (http://odur.let.rug.nl/~vannoord/TextCat/Demo/) of the method introduced in [13].

4.3. Results without using language probabilities

The results for correct identification can be seen in *Table 1*. A manual overview of the classified texts showed that in the case given in the first line the texts classified to a different language often really did not belong to the language of their groups, or had mixed language; additionally we found that for precise word level operation we need to recognize certain expressions beforehand that can be viewed as language-independent (regarding their format, not their pronunciation), for example Roman numbers, internet and e-mail addresses, dates, international words (e.g."tel.","fax."), abbreviations, expressions containing measurements (e.g."2 cal").

4.4. Results using language probabilities

We also examined the effect of taking into account language probability calculated from the environment of words. For this we first labeled the training set using the LID database created for the third line of Table 1. We used this one because the small training set and the resulting relatively poorer identification rate means

rule templates
L1* S* L? S* L1*
L1* S* L? S* L2*

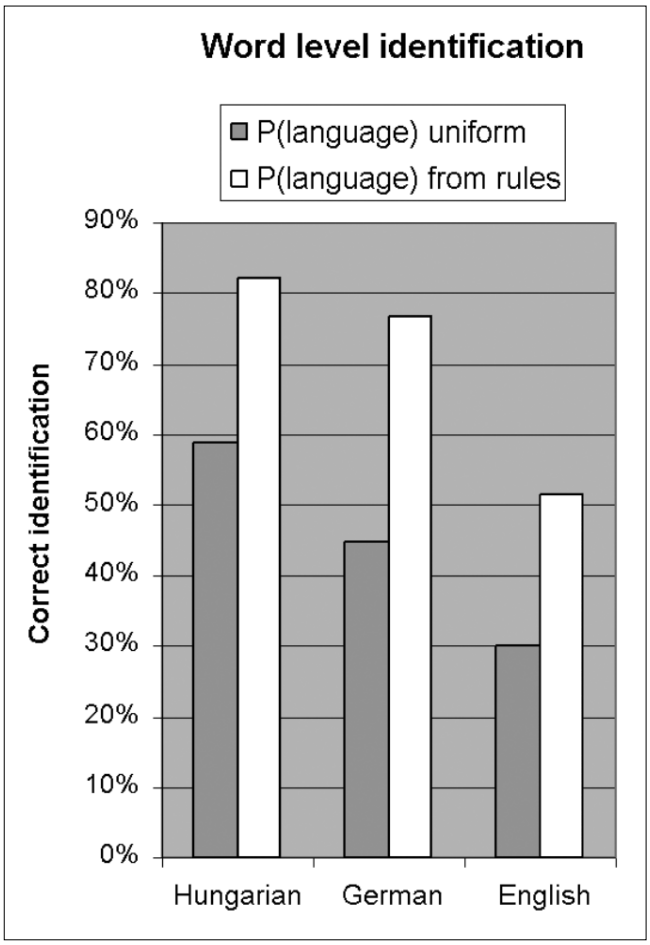
rules generated for the training corpus using the LID database for Table 1 Line 3
("": number of times the template could be applied;
"L": number of other labels – neither L1, nor L2)
L1* S* L? S* L1*; { "": 13300305, "L1":13230575, "L":69730 }
L1* S* L? S* L2*; { "": 20188476, "L1":16625250, "L2":16625298, "L":168503 }

Figure 5. The rule templates used and the rules generated by applying them

a greater challenge for our method. Then we applied the rule templates shown in *Figure 5* onto the labeled text and we arrived at the rules shown below the templates. Using these rules to estimate P(language), we relabeled the texts with the technique described in 3.3.2. With this, the correct identification rate on the German test corpus increased from the former 45% to 65% (assuming the text to be purely German), then iterating the rule generation – labeling steps it increased to 70%, and further on to 72%.

The improvement is shown in *Figure 6* for three languages. The bulk of errors for English (10% and 14%) resulted from erroneously deciding on the very similar Scottish language. It is noteworthy that such improvement was attained despite that we did not use the actual probability of each language but a rough approximation of the probability of neighboring words having the same language. Based on the figures in Table 1, such word level identification rate allows for 80% to 90% correct sentence-level identification using majority voting. Because of the availability of a great amount of text in various languages, e.g. through the internet, we are not obliged to use such small training sets. Therefore in practical applications we can expect an even better rate of correct identification that the one shown in the first line of Table 1, very close to 100%, by using language probabilities in the described way.

Figure 6. Examples for the improvement attained using language probabilities



5. Conclusions

In this article we sketched the significance of TTS systems in telecommunications and pointed out the importance of automatic labeling methods, like e.g. language identification and part-of-speech tagging.

We overviewed some techniques used for language identification. To address some of their weaknesses we introduced a new method that work with two kinds of conditional probabilities, estimating their values using decision trees. We demonstrated its effectiveness on the task of language identification from text. The results justify the viability of the approach. We expect it to be applicable for other topics also, e.g. for an approximation of part-of-speech tagging without using a morphological analyzer.

References

[1] Németh, G., Zainkó, Cs., Fekete, L., Olaszy, G., Endrédi, G., Olaszi, P., Kiss, G., Kis, P.: "The Design, Implementation and Operation of a Hungarian E-mail Reader", International Journal of Speech Technology, Volume 3, Numbers 3/4, December 2000, pp.217–236.

[2] G. Németh, Cs. Zainkó, G. Kiss, M. Fék, G. Olaszy and G. Gordos: "Language Processing for Name and Address Reading in Hungarian", Proc. of IEEE Natural Language Processing and Knowledge Engineering Workshop 2003, Oct. 26-29, Beijing, China. pp.238–243.

[3] Pfister, B., Romsdorfer, H.:
"Mixed-lingual text analysis for polyglot TTS synthesis",
Proc. of Eurospeech 2003,
pp.2037–2040.
[4] Halácsy, P., Kornai, A., Németh, L., Rung, L.,
Szakadát, I., Trón, V.:
"Creating open language resources for Hungarian",
Proc. of LREC 2004,
pp.203–210.
[5] Marcadet, J. C., Fischer, V., Waast-Richard, C.:
"A Transformation-based learning approach to
language identification for mixed-lingual
text-to-speech synthesis", Proc. of Eurospeech 2005,
pp.2249–2252.
[6] Prószéky, G.:
"Humor: a Morphological System for Corpus Analysis.
Language Resources for Language Technology",
Proc. of the First European TELRI Seminar,
Tihany, Hungary, 1995.
pp.149–158.
[7] Németh, L., Halácsy, P., Kornai, A., Trón, V.:
"Nyílt forráskódú morfológiai elemző"
(Open-source Morphological Analyser)
In: Csendes D, Alexin Z. (eds.):
II. Magyar Számítógépes Nyelvészeti Konferencia
(Hungarian Conference on Computer Linguistics),
Szeged, 2004.
pp.163–171.
[8] Ted Dunning:
"Statistical Identification of Languages",
Computing Research Laboratory,
New Mexico State University, 1994.
[9] G. Németh, Cs. Zainko:
"Multilingual statistical text analysis,
Zipf's law and Hungarian Speech Generation",
Acta Linguistica Hungarica, Vol. 49 (3-4), 2002.
pp.385–405.
[10] Prager, J. M. Linguini:
"Language Identification for Multilingual Documents",
Proc. of the 32nd Annual Hawaii International
Conference on System Sciences, Vol. 1., 1999.
pp.2035.
[11] Tian, J., Suontausta, J.:
"Scalable neural network based language
identification from written text",
Proc. of IEEE International Conference on Acoustics,
Speech, and Signal Processing, Vol. 1, 2003.
pp.48–51.
[12] Häkkinen, J., Tian, J.:
"N-gram and Decision Tree-based Language
Identification for Written Words",
Proc. of IEEE Workshop on Automatic Speech
Recognition and Understanding,
Madonna di Campiglio Trento, Italy, 2001.
[13] W. B. Canvar, J. M. Trenkle:
"N-gram based Text Categorization",
Symposium on Document Analysis and Information
Retrieval, University of Nevada, Las Vegas, 1994.
pp.161–176.
[14] Sproat, R., Riley, M.:
"Compilation of weighted finite state transducers
from decision trees"
In: Association for Computational Linguistics,
34th Annual Meeting, Santa Cruz, Canada, 1996.
pp.215–222.
[15] Suendermann, D., Ney, H.:
"Synther – a New M-Gram POS Tagger",
Proc. of the NLP-KE 2003,
International Conference on Natural Language
Processing and Knowledge Engineering,
Beijing, China, 2003.
[16] Halácsy P, Kornai A., Varga D.:
"Morfológiai egyértelműsítés
maximum entrópia módszerrel"
(Morphological Disambiguation with
Maximum Entropy Method),
Magyar Számítógépes Nyelvészeti Konferencia
(Hungarian Conference on Computer Linguistics),
2005.