

# LANGUAGE IDENTIFICATION: A SOLVED PROBLEM

## SUITABLE FOR UNDERGRADUATE INSTRUCTION\*

*Paul McNamee*  
*Johns Hopkins University Applied Physics Laboratory*  
*11100 Johns Hopkins Road, Laurel MD 20723-6099*  
*paul\_mcnamee@jhuapl.edu*

### ABSTRACT

Automatic determination of the language of an electronic text is an important problem, which arises when processing natural language. This paper describes the main methods used in attacking this problem and demonstrates how even the most simple of these methods using data obtained from the World Wide Web achieve accuracy approaching 100% on a test suite comprised of ten European languages. The language identification problem and its solution illustrate many fundamental issues in computer science and the processing of human language; accordingly it is well suited for classroom instruction.

### 1 INTRODUCTION

Automated language identification is the problem of determining the language that a passage of text is written in; though a text may in fact be written in more than one language, in this paper we make the typical assumption that only a single answer is correct. Correct determination of language is an important first step for a variety of language processing tasks. For example, information retrieval systems often perform language-specific processing to eliminate common words (stopwords) or to generate a canonical term for a set of morphologically related variants (*e.g.*, mapping kayaker, kayak, and kayaking to a single base form); they need to have prior knowledge about the language of documents or possess an ability to infer it. Email sent to *customer-support@large-multinational.com* could be written in many languages and should be routed to a staff member who can understand the message and respond in the appropriate language. In countries where there are diverse multilingual populations or having multiple official languages (*e.g.*, Canada, Switzerland, and India) this problem is

---

\* Copyright © 2004 by the Consortium for Computing Sciences in Colleges. Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the CCSC copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Consortium for Computing Sciences in Colleges. To copy otherwise, or to republish, requires a fee and/or specific permission.

more common than in the United States. As some documents are multilingual, it would be nice if word processing programs could automatically identify when a sentence or paragraph is written in a different language and apply the proper rules for checking grammar and spelling. Optical Character Recognition (OCR) software needs to know which language to generate for a scanned page. Other examples abound.

Related problems include correct identification of language from speech, organization of languages by language family, determination of dialect, and identifying the character encoding of text. Language identification from speech is receiving a lot of attention at present - it remains a significantly more challenging problem compared to text. Language similarity has typically been established by linguists, though automated approaches are known [1]. Determination of dialect or character encoding can be viewed as a sub-language identification problem and the same methods may be employed.

Language identification is clearly a multi-class classification problem - choose exactly one of  $n$  languages that a text should be assigned to. Despite this fact, traditional machine-learning algorithms such as Decision Trees or K-Nearest-Neighbor are not typically relied upon. Numerous approaches to automated identification of language have been developed; however two in particular are most widely used in practice: the use of statistical language modeling to estimate the probability that a sample input is written in a given language, and comparisons between the frequency of usage of common words (or other terms) in a textual sample to frequencies extracted from a statistical analysis of a large reference corpus.

Statistical approaches to human language processing have experienced a renaissance since the mid 1990s and often outperform knowledge-based approaches that rely on automated reasoning. This is exactly the case with language identification and accuracies approaching 100% are possible given relatively little training data. Dunning [2] and Grefenstette [3] present nice studies of this problem using European languages. A web site listing commercial and research tools that predict language is available online at: <http://odur.let.rug.nl/~vannoord/TextCat/competitors.html>

In the remainder of this paper we first discuss the principle approaches in greater detail. This presentation is followed by a description of an experiment using data, which is publicly available from the Internet.

## 2 RELATED WORK

Anyone that has taken even a short course in a foreign language quickly learns the distinct orthography and morphology of that language. Given a small sample of a text one can recognize common words, particularly closed-class words (*i.e.*, determiners, conjunctions, prepositions). In English text one expects to see *the*, *and*, and *of*; in Spanish, *de*, *les*, *los*, *el*, *la*, and *y*. Similarly the occurrence of *-ious* and *-ism* is suggestive of English, whereas *-ueux* and *-isme* support the belief that the writing is French; unfortunately *-ation* is found frequently in both. There are other features that might be used, such as distinctive punctuation (*e.g.*, *¿* in Spanish), use of diacritical marks (*e.g.*, *á* and *ñ*), and mean word length (Finnish and German are notably longer than English); however, accurate discrimination is possible using character or word-level features alone and these attributes are more numerous and less error-prone. In cases where short

passages must be assigned a language, distinctive accents or punctuation might not be found.

The first major approach is based on statistical language modeling. Markov models are used to estimate the probability of a sentence. By training different language models, one for each language, the conditional probability of a new sentence being generated from the language model can be computed. These conditional probabilities can be compared and the largest one used to make a prediction. Almost all of language modeling research is based on the fact that it is difficult to estimate probabilities for large order Markov models; this is called the data sparseness problem. To overcome this difficulty, typically trigram models are used and probabilities are smoothed to overcome issues with unseen terms. This processing can be done on the word or character level; characters seem best. While Grefenstette compared common words and common trigrams he did not fully describe his methodology; however from context it would appear to have been a zeroth-order Markov model based on either words or trigrams. Dunning made a more exhaustive comparison using models of order zero-through six on characters (*i.e.*, from single letters to sequences of 7 letters); he also found that trigrams work well.

In this paper we focus on simpler methods that do not require an understanding of statistical language modeling. This class of algorithms is based on counting the common words or character sequences (called n-grams) and comparing the frequencies that these are seen in the input text to frequencies obtained through analysis of a large reference corpus. How then should these frequencies be compared? One means is to represent a language profile and novel documents as vectors in a space where each dimension corresponds to a different common word or n-gram. Different vector comparisons could be made, including comparing the cosine of the angle between the two vectors or the inner product. The language that produces the highest score is deemed closest to the to-be-classified text.

A recent study by Takei and Sogukpinar used n-grams of length 1 (*i.e.*, the frequencies of single letters) to discriminate between 4 languages with about 98% accuracy [7]. They used a cosine vector comparison. It was not clear what length of documents they considered; it would be surprising if they obtained that level of precision on single sentences and relatively close languages. Their technique is efficient and probably intended for indexing HTML documents, which tend to be roughly 10 KB in size on average.

Grefenstette compared the efficacy of common words and common trigrams to determine language among ten European languages. His test data was based on individual sentences from the European Corpus Initiative (ECI) collection. The n-gram method performed better than the use of common words when sentence length was small; however both methods worked well when the number of words in a sentence was greater than 15. Danish appeared to be the language that presented the greatest difficulty.

Other approaches based on information theoretic measures (*e.g.*, relative entropy) have been proposed by Benedetto et al. [1] and Sibun and Reynar [6]. Relative entropy was also used by Jones and Ghani to automatically obtain large corpora for minority languages by searching the World Wide Web [4].

### 3 EXPERIMENTATION

To explore how easily an accurate language identification system can be developed we searched for sources of data from the Internet. Particularly useful for this experiment is data from the Europa web site; Europa is the online resource where official information from the European Union is published in all of the official languages [8]. At present this includes: Danish, Dutch, English, Finnish, French, German, Greek, Italian, Portuguese, Spanish, and Swedish; however, EU enlargement will grow this list to about twenty languages. Philipp Koehn has produced the Europarl corpus that consists of EU parliamentary oration, which he has applied to the study of phrase-based statistical machine translation [5]. The data is easily transformed into individual sentences. I decided to use a small sample of the Europarl data for testing since it was split up into sentences. To avoid overtraining on this data, I decided to seek different training data. Training data - data on which to compute word frequencies - was obtained from Project Gutenberg, a web site that publishes electronic texts, typically literature [9]. Though the vast amount of data from the Gutenberg site is in English, a little data exists for about twenty languages. I chose a single document for each of 10 languages - all of the languages mentioned above except Greek, which does not use the ISO-8859-1 encoding. These data were rather different in length, period, and content. The training data are summarized in Table 1.

**Table 1. Training data obtained from Project Gutenberg site**

Language	Author	Title	PG ID#	Length (KB)
Danish	Herman Bang	Faetra: Brudstykker af et Livs Historie	11396-8.txt	325
Dutch	Constantijn Huygens	Vitaulium: Hofwyck en Spaansche Wijsheit	10975-8.txt	339
English	Beatrix Potter	The Great Big Treasury of Beatrix Potter	gbtbp11.txt	180
Finnish	Maiju Lassila	Tulitikkuja lainaamassa	10927-8.txt	326
French	Jules Verne	20000 Lieues Sous Les Mers	820kc10.txt	914
German	Johann Wolfgang von Goethe	Die Wahlverwandtschaften	8wahl10.txt	542
Italian	Dante Alighieri	Divina Commedia di Dante	0ddc809a.txt	566
Portuguese	Luis Vaz de Camoes	Os Lusíadas	lusds10.txt	349
Spanish	Miguel de Cervantes	Don Quijote	2donq10.txt	2143
Swedish	Viktor Rydberg	Den siste atenaren	10117-8.txt	980

All documents were encoded in ISO-8859-1. Each training document contained a lengthy English legal disclaimer in addition to the language-specific narrative - this was left in as a realistic problem in obtaining pure training data, though its removal would not have been prohibitively difficult.

Documents were processed as follows. All text was case-normalized to lower-case letters and accent marks were retained. Tokens were obtained by splitting on whitespace and by removing any punctuation marks at the beginning or end of a token. This left a series of "words", possibly containing interior punctuation. For each language the number of times each word occurred was computed and this profile was saved on disk. Each profile consisted of the percentage of the training data attributed to each observed word; the values were written out in order of decreasing frequency. Samples for Danish, Dutch, English, German, and Spanish are shown in Table 2.

After each profile was generated from the training data, a separate program was written that loaded profiles for each language and scored sentences according to which profile best fit a given sentence. Sentences were chosen from a small set of the Europarl collection. The following procedure was followed to generate 1000 sentences per language for testing purposes:

1. The data was aligned (to English) using the Perl script provided with the Europarl data
2. Blank lines were removed
3. The first 500 lines were taken from two files in the Europarl corpus: ep-00-01-17.txt and ep-02-07-02.txt

**Table 2. The ten words most observed in the training data for five languages.**

%	Danish	%	Dutch	%	English	%	German	%	Spanish
4.35	og	4.19	en	6.16	the	3.14	und	5.38	que
3.71	hun	3.11	t	4.31	and	2.31	die	4.75	de
1.82	i	2.34	de	2.87	a	2.62	sie	4.74	y
1.70	de	2.25	van	2.06	of	1.92	zu	2.70	la
1.59	det	1.84	een	2.02	to	1.91	der	2.57	a
1.45	var	1.62	in	1.61	was	1.59	sich	2.15	en
1.38	han	1.56	die	1.57	he	1.30	er	2.14	el
1.38	at	1.50	is	1.50	in	1.24	in	1.65	no
1.25	saa	1.23	niet	1.04	it	1.19	nicht	1.23	los
1.24	en	1.22	ick	0.86	his	1.12	das	1.22	se

To compare the two vectors I used the inner product based on the words in the sentence and the 1000 most common words in each language. To illustrate the procedure, consider the nonsensical sentence, "in die" and suppose that only the top ten words in each language were used. The dot produce for Danish would be 0 as neither of these two words is one of the ten most common Danish words. The score for Dutch would be:  $1.62 * 1 + 1.56 * 1 = 3.18$ . Similarly, English would be  $1.50 * 1 + 0 * 0 = 1.50$ ; German would be  $2.31 + 1.34 = 3.65$ ; and, Spanish would be 0. Accordingly, German would be

considered the most likely language for this short snippet of text with Dutch as a close second choice.

A confusion matrix, which shows the mistakes made on sentences in the test set, is shown in Table 3. There were 1000 sentences in total and correct predictions show up in the diagonal of the matrix. Performance varies from 80.0% for Portuguese to 99.5% with German. German is seldom mistaken for anything else and little else is mistaken for German. On the other hand, Spanish is usually correctly identified (97.9%), but French, Dutch, and Portuguese are often mistaken for it. The Spanish training data was twice as large as the next largest training sample; this could have an effect.

**Table 3. Confusion matrix for 1000 sentences. Rows denote the true language and columns represent predictions.**

	da	de	en	es	fi	fr	it	nl	pt	sv
da	866	16	2	17	1	12		27	1	58
de	2	995			1			1		1
en	2		984	3	2			8		1
es	4		1	979	1	2	3	5	5	
fi	34	1	1	8	934	6	1	13	2	
fr	2			162		829	1	5	1	
it	5		1	19		19	939	3	11	3
nl	8	5	4	166	1	3		813		
pt	3			190		6	1		800	
sv	26	3		7	3	2		16		943

Short sentences pose the greatest challenge. To simulate a situation where a longer snippet is available, such as an email message, the prediction program, which calculates the dot products for each sentence was changed to group sentences together. When taken 2 or 4 sentences at a time, precision dramatically increases, as can be seen in Figure 1.

This high level of performance would be adequate for applications such as predicting the language of email messages, but would not be sufficient for predicting the query language of queries submitted to a search engine, which are usually only a few words in length.

#### 4 OBSERVATIONS

The exercise of computing the frequency of occurrence of words can be quite interesting to students. It is well known that word frequency observes Zipf's law, which can be interpreted as saying, "the  $i^{\text{th}}$  most frequent word will occur roughly  $1/i$  times as often as the most frequent word". Modern information retrieval systems use this fact to consider some words as more important for document ranking than others. An entertaining exercise is to present students with a sample of text in a language unknown to them and see if they can identify the most common words (via limited visual

inspection). Students can then determine (or can be shown) the actual frequencies and translations. In this way they learn that function words (*e.g.*, prepositions) are easy to identify in language simply by observing word frequency.

A reasonably high level of performance was attained using a very simple algorithm

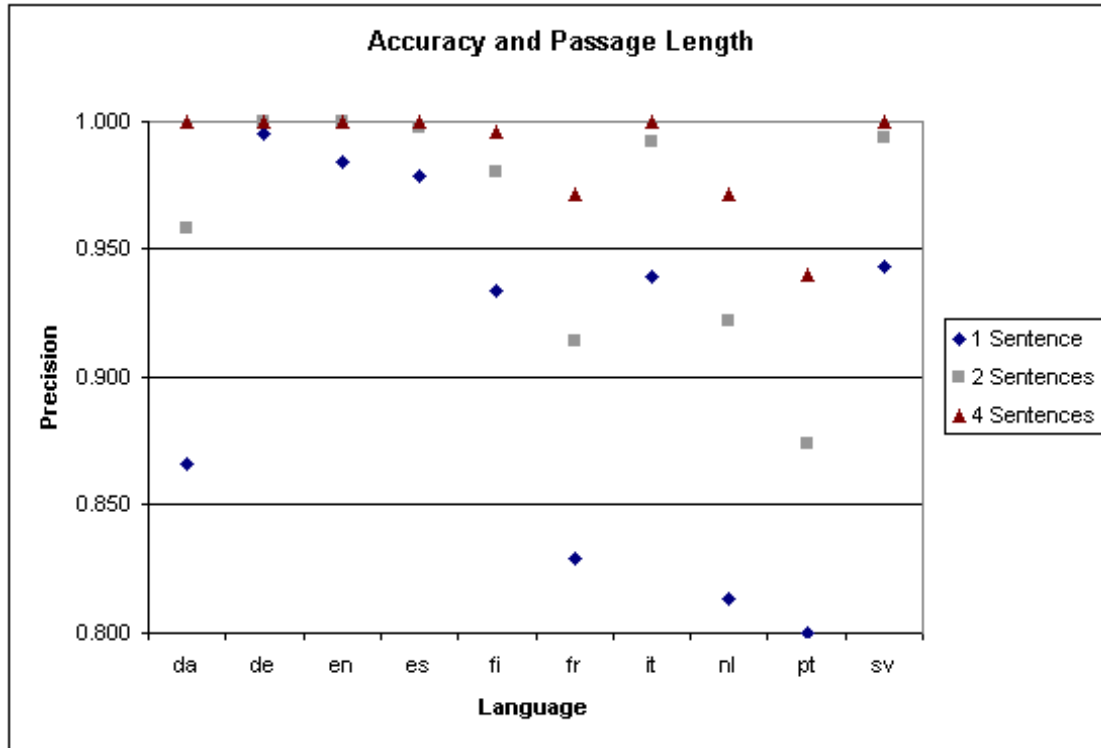


Figure 1. Accuracy of predictions for each of ten languages. Performance is shown for passages consisting of 1, 2, or 4 sentences.

and easily obtainable data. This is a reasonable laboratory project for beginning computer science students and it assumes the following: the ability to processing textual data by reading files; some understanding of data structures (hash tables or binary trees) to store word frequency information in; and, the ability to sort data by frequency.

The code used in this experiment was based on two Java programs totaling less than 300 lines of code. A one-line awk script was used to measure performance. Students can easily investigate variables that influence performance such as passage length, disparate training data, larger amounts of training data, training data drawn from the same source as testing data, and algorithmic improvements. The results are relatively easy to interpret and the computer performance probably exceeds the performance of the average student in predicting language. To encourage experimentation a classroom competition can be run using held-out data.

## 5 CONCLUSIONS

With the decrease in the cost of hard disk space, an experiment once infeasible for students is now quite easily done. Each student does not necessarily need to have a copy of the training and testing data, only the ability to read the data from a server. Though some basic skills are required, automated language identification is an interesting problem

that can demonstrate to students the importance of data structures and the application of mathematics to real-world problems. Students can build a program that does a useful task that they probably cannot and they can gain an appreciation for some of the issues faced in applications in computational linguistics.

## 6 REFERENCES

1. D. Benedetto, E. Caglioti, and V. Loreto, 'Language Trees and Zipping.' *Physical Review Letters* 88(4), 2002.
2. T. Dunning, 'Statistical Identification of Language.' In Technical report CRL MCCS-94-273, Computing Research Lab, New Mexico State University, March 1994.
3. G. Grefenstette, 'Comparing Two Language Identification Schemes.' In *3rd International Conference on Statistical Analysis of Textual Data*, 1995.
4. R. Jones and R. Ghani, 'Automatically Building a Corpus for a Minority Language from the Web.' *Proceedings of the Student Workshop at the 38th Annual Meeting of the Association for Computational Linguistics*, 2000.
5. P. Koehn, 'Europarl: A Multilingual Corpus for Evaluation of Machine Translation' Available on-line at: <http://www.isi.edu/~koehn/europarl/>
6. P. Sibun and J. Reynar, 'Language Identification: Examining the Issues.' *Proceedings of the Fifth Conference on Document Analysis and Information Retrieval*, pp 125-135, 1996.
7. H. Takei and I. Sogukpinar, 'Centroid-Based Language Identification Using Letter Feature Set.' In the *Proceedings of CICLING 2004*, LNCS 2945, pp. 640-648, 2004.
8. <http://europa.eu.int/>, Europa web site
9. <http://www.gutenberg.org/>, Project Gutenberg