Linguini: Language Identification for Multilingual Documents

John M. Prager
IBM T.J. Watson Research Center, P.O. Box 704, Yorktown Heights, NY 10598
jprager@us.ibm.com

Abstract

Given the vast and still-growing availability of electronic documents from around the world, it is becoming increasingly important for managers of the information systems on which these documents are stored to sort or tag these documents so that their end-users can most readily access those documents that are of most interest and use to them, which in our context means in a language they can understand.

We present in this paper Linguini, a vector-space based categorizer tailored for high-precision language identification. We determine the functional dependencies of Linguini's performance, and demonstrate that it can identify the language of documents as short as 5-10% of the size of average Web documents with 100% accuracy.

We also present how to determine if a document is in two or more languages, without incurring any appreciable extra computational overhead. We argue that this approach can be applied equally to subject-categorization systems to distinguish between cases where, when the system recommends two or more categories, the document belongs strongly to all or really to none.

1. Introduction

Thanks largely to the Internet, computer users have access to documents from all around the world, and written in a variety of languages. For purposes of organizing these documents, filtering them, and possibly directing automatic translation processes at them, it is very desirable to be able to automatically determine the language(s) in which they are written. We present in this paper Linguini, a vector-space based categorizer tailored for high-precision language identification. We show how the accuracy depends on the size of the input document and the set of languages under consideration, and what we discovered were the best features for use in this effort. In our tests, we found that Linguini could identify the language of documents as short as 5-10% of the size of average Web documents with 100% accuracy. The major contributions of these tests are a determination of the relative rank-ordering of these feature-sets, and an expectation of identification accuracy as a function of input text length.

We also present a discovery of how to determine if a document is in two or more languages, without incurring any appreciable computational overhead beyond the monolingual analysis, which is linear in the number of languages considered. By comparison, a naive approach would be quadratic in the number of languages under consideration. Our algorithm finds not only the component languages of a document, but the relative proportions of each. We also argue that this approach can be applied equally to subject-categorization systems to distinguish between cases where, when the system recommends two or more categories, the document belongs strongly to all or really to none.

1.1 Importance of Language Identification

The Internet is a vast source of textual material and it continues to grow rapidly. Current measurements show the number of hosts to be about 60,000,000 increasing at about 2,000,000 a month (Hobbes' Internet Timeline at http://www.isoc.org/zakon/Internet/History/HIT.html). The World Wide Web consists of about 9,500,000 domains of which about 1,700,000 are in non-English-speaking countries (http://www.domainstats.com/). Estimates of the amount of textual material vary widely, but there is no doubt that it is large, growing, and a fair proportion of it in languages other than ones primary language, whatever that may be. There are many systems that mediate the transfer of information between author and reader, from word-processor to browser. These involve the storage, recall, selection and transmission of information, and can all potentially include the operations of filtering or tagging the information encountered. It is one of the principal goals of the managers of information systems installations to ensure that their end-users have timely access to that information which they need, and if possible, if not to make inaccessible then at least to hide information that is irrelevant to them. To act in this way on the documents' subject-matter requires knowledge both of the user's interests and the semantic contents of the text. This is the domain of *text categorization*, which is not directly the subject

of this paper. However, another document property whose identification can aid in the management of information is the language in which it is written; as it happens, language identification (LI) is a very specific application of the techniques of categorization.

Being able to identify the language(s) of a document can aid the administration of an information system in a variety of ways. Let us assume that the end-user can specify in which one or several languages he or she wishes to see documents. Given an LI system, the administration can sort the documents under their control into separate databases or repositories based on language (or tag by language documents in a common repository), so that on search or browsing the end-user only sees documents in the chosen language set. Alternatively, the documents' language(s) can be identified on the fly and undesirable documents suppressed from view. A variation of this scenario does not involve suppression of the documents but merely flagging them with the language name to let the user make the final choice, as is done by the AltaVista search engine (http://www.altavista.com).

To perform a reasonable job of searching for documents, search engines need to know the languages of both the query and the document text, the latter at indexing time. First, the stopwords, a collection of the 200 or so most common words (such as "the", "and", of", not generally considered useful for purposes of search), are removed. The stopword list of course changes from language to language. Maybe more important, since stopword removal is more a benefit to speed than accuracy, is the treatment of morphology. This involves the normalization to a common form of all of the variations of the form of a word due to number, case or conjugation. This is done either by stemming, in which word endings are lopped off (so that "save", "saver", "saves", "saving", "saved" all get represented by the stem "sav"), or converted to a lemma form (so that "sing", "singer", "sings", "singing", "sang", "sung" all get represented by "sing") [11], [14], [22]. These processes are language-dependent. Additionally, any higher-order processing such as phrase approximation [20], proper name finding [23], extraction of relations [4], [13], use of thesauri [2, p. 170], or question-answering [16] is language-dependent and can only proceed following an accurate identification of the language in which a document is written.

Summarization of documents is an increasingly important functionality of text-based information systems. It is used by search engines to provide a condensed view of the documents on hit-lists, it can be employed in document catalogs, and in any document notification service to which users have subscribed, amongst other uses. While there are some techniques that are language independent (such as extracting the first so-many bytes), many leading techniques, including extraction of cue-phrases such as “The aim of this paper is to …” [26], and determination of topicality [3], are predicated upon knowing the language of the texts concerned. Other application areas such as *message understanding* and *topic detection* (see the latest proceedings of the DARPA- and NIST-sponsored MUC [21] and TDT [25] conferences) rely heavily on at least partial natural language understanding.

One function that some information systems managers might be interested in supporting is that of automatic machine translation of foreign-language texts. This too requires LI to proceed. Moreover, a more complete service for the end-user would be to identify and translate embedded foreign-language texts, anything from large passages down to sentences or even phrases. Not only does this require LI, but the alternative use of metadata, which we discuss in the next section, would not be remotely helpful.

1.2 Use of Metadata

A natural question that arises in the context of LI is why not have the document author tag the document with a language descriptor, so that any system which processes the document in the future might know unambiguously the language of the text. One way to do this is by means of the World Wide Web Consortium’s *Resource Description Framework* (RDF) (http://www.w3.org/RDF/). It turns out there are several problems in general with the idea of relying on the existence of a document’s language tag. First, as mentioned above, such tagging would not accommodate the use of embedded foreign-language phrases. Second, it would not help with identifying the language of a query issued to a search engine. Third, it would not address the hundreds of millions of text documents on the Web and elsewhere that currently do not have such descriptors affixed – indeed plain-text (non-HTML) documents stored in file-systems (as opposed to databases) do not have a mechanism for attaching metadata. Fourth, it is not always the case that authors can be persuaded to take the time, minimal though it may be, to make the appropriate annotations. From conversations with the IBM Webmaster [6], who desired page owners to supply descriptors to help direct customers to the appropriate pages, this author learned that trying to get owners to voluntarily supply extra information, even if it was to their ultimate advantage, was often a painful and largely fruitless task. Finally, it is arguably the case that even if such annotations existed, they would introduce errors more often than by automatic determination. We show in this paper that Linguini has 100% accuracy for paragraph- and above sized texts. However, it is a useful and frequently-used practice when

authoring a document to take as a starting point another document, especially one that has some common content or structure, and to modify it. Sometimes this modification entails leaving just the HTML skeleton, or a JavaScript script, or a set of out-going links, or a diagram, and removing the text contents entirely. If the new text contents were in a different language, then this procedure leaves open the possibility that the original language tag would go unnoticed and unmodified.

1.3 Background

Many of the techniques of text classification are described in [8]. The classical vector-space model of Salton & McGill [24] is the basis of several approaches, including ours. In this model, the basic idea is to compute a similarity measure, based on cosine distance in feature-space; in text retrieval, queries are compared with documents, while in categorization documents are compared with categories, or with other documents in a k-Nearest-Neighbor algorithm [19], [27]. The features used might typically be all of the non-stopwords in the domain of interest. The item of interest **D** is represented by $\mathbf{d} = (d_1, d_2, \ldots d_n)$, where the d_i represent a weighted count of the number of times feature i occurs in **D**. An item **F** (a document or a category) with which **D** is to be compared is represented by a vector $\mathbf{f} = (f_1, f_2, \ldots f_n)$. The f_i are the weighted occurrences of the feature i in **F**. The weighting is by some function of the *rarity* of the feature in the domain; the *inverse document frequency* (*idf*) is commonly used [24]. Lewis et al. [18] describe a variation with *idf* for the f_i but binary values for the d_i. In all such cases, though, the cosine distance between **D** and a given **F** is simply calculated from the formula for the scalar product of **d** and **f.** This value is used as a measure of the quality of the match, where a score of 0 corresponds to orthogonality rising to 1 for exact correspondence. Typically, the **F** with the highest score is taken to be the best match for **D**.

Some classifiers are trained by a learning or optimization procedure; see Lewis et al. [18] for a comparison of three of these, or Yang [28] for an evaluation of these plus some other systems. Another approach to categorization is through the use of decision rules [1], also evaluated in [28].

There is not a great deal of research currently into classification of text documents by language as opposed to subject-matter - indeed, literature searches for "language identification" mainly retrieve either articles on identification of language in speech, or of determining what formal grammar generates certain sets of strings. Some recent works on LI include [5] and [7]. As with the present paper, these articles also examine the use of N-grams, but do not consider the use of words, as we do.

Now, in all such systems, the matching score is determined by the kind of features chosen, the weighting (if any) and the formula used. The best-matching **F** for a given **D** is determined not only by its matching score, but by the matching score for other **F**'s. This means, for example, that it is generally difficult to compare the relative accuracy of different classifiers if they are not using exactly the same set of categories. Cavnar and Trenkle [5] seek to identify the 8 European languages found in 14 Usenet newsgroups. They use an ad-hoc method to compare the rank order of feature occurrence. They examine the effects not of different feature types, but of truncating feature-sets to lose the less-useful features. They do not specifically examine the effect of input length on results, as they just divide their input texts according to whether they are greater than 300 bytes or not; their average text size is 1700 bytes. Indeed, the TEXTCAT Language Guesser demo (http://odur.let.rug.nl/~vannoord/TextCat/Demo/textcat.html), based on their system, advises users to supply inputs of at least 3 or 4 lines of text. Damashek [7] does not do LI, but uses N-grams to perform language-independent document retrieval.

Further work on information retrieval in the context of multiple languages can be found in [10].

Grefenstette [9] uses a probabilistic method to compare the use of trigrams with short words (words of 5 characters or fewer). This method gives good results, but it is unclear if it is possible to extend it to deal with multilingual documents efficiently or effectively.

The lack of much activity in written natural language identification is probably because it is not considered a difficult problem, which is certainly true if the document is long enough and computational resources are not constrained. However, we are interested in performing language identification in domains with possibly severe constraints. Not only are documents in many languages readily available over the Internet, but users are speakers of these languages, and when using search engines would prefer to use their own language to express queries. For a search engine to operate properly (i.e. use the appropriate rules of morphology and "de-agglutinization"), it must know the user's language. To this end, it is desirable to identify the language of a query, which can be as short as a few words - or even one word. Furthermore, an Internet search engine whose index is kept up-to-date will be indexing and language-identifying at the very least just those pages that change daily. Lawrence & Giles [17] estimate the World Wide Web to be 320 million documents, and the present author has observed through analysis of last-modified dates of documents returned by a crawler on the IBM

internet domain that approximately 1-2% of documents change daily. These figures suggest the need to process several million documents per day; to operate on just one million documents per day allows an average of less than 1/10th of a second per document.

1.4 Organization of this paper

The rest of this paper is in three parts. In section 2 we look at the performance of Linguini for different kinds of feature-sets and for different lengths of test text data. From these results an approximate rank-ordering of the different feature-sets can be derived. While it is impossible to prove, it is thought likely that the relative merits of these feature-sets would pertain to any language identification system operating on similar principles. As argued above, it is difficult to perform a direct comparison with other systems, but the demonstrated accuracy of Linguini appears to be at least as good as the alternative systems. The experiments with length of input data provide a user or system administrator an expectation of how much text is necessary to perform language identification at a given accuracy level with a state-of-the-art system.

In section 3 we look at the underlying vector-space model to explain some of the observed phenomena and to derive an efficient algorithm for identification of multilingual documents. We also briefly discuss the treatment of short embedded foreign-language texts such as quotations, and possible application of the multilingual analysis to subject categorization. In section 4 we summarize our findings.

2 Determining Linguini's Performance for Monolingual Documents

Linguini is a vector-space based categorizer used for language identification. It uses dictionaries generated from features extracted from training texts, and compares these against feature vectors generated from test inputs. Features used are N-grams (sequences of N consecutive characters) and words, and combinations of both. Experiments were performed to determine the accuracy of Linguini for input texts of different lengths, for different features and combinations thereof. Dictionaries and test inputs were generated from texts in 13 different European languages. A subset of the six most “common” of these languages was also tested.

2.1 Overview of the Approach

Although training sets (texts) were available for over 20 languages, including oriental ones, a 13-language European subset was used for these particular tests. For each of these 13 languages, a collection of about 100 Kbytes worth of text was gathered from the Internet. Due to well-known problems of pollution, these texts were scanned manually and any obvious inclusions of foreign-language text strings were deleted. The resulting texts were tested as described below.

Our objective was to determine the accuracy of Linguini for a variety of input text sizes, for the best feature dictionaries (used in the computer-science sense of "hash-tables"). For doing language identification, the obvious candidate features to use are words. For short documents, however, this puts a very strong requirement on the training data to include all words likely to be encountered, and also fails to capture the observed human ability to identify languages with high confidence just from the appearance of words, without having seen the particular words before. Clearly, letter sequences (N-grams) are important too. This was verified with an version of Linguini trained on about 25 languages, including some rarer European languages as well as some Oriental ones.

We investigated which of words or N-grams would be better to use, and what would be the best values for N and word-length. Intuitively, short N-grams have an advantage over longer N-grams in that, due to there being fewer possible permutations of them, their individual occurrence rates would be much higher and so relatively little training data would give distributions matching large corpora. On the other hand, longer N-grams, though sparser, would have the advantage of being individually more definitive of one or a small subset of languages.

In the field of document retrieval, the most common words in a language (called *stop-words*) are filtered out when doing search or categorization; these words are typically very short (at least for Western European languages). In contrast, these same words show a great deal of variation between languages, even between linguistically very close languages. In some cases it appears that there is more variation amongst these short words than longer ones (compare “There is an administrative organization” with the French “Il y a une organization administrative”. This would suggest the use of short words as features. However, an argument can be made for longer words similar to that made for longer N-grams.

Consequently, we needed to determine empirically the optimal features for our purposes. The features we chose to examine were N-grams (sequences of N consecutive characters, not spanning words but possibly including word-ending spaces) with N ranging from 2 to 5, and words of two different size groups: either words of 4 characters or less, or words of any length. Also, testing was done of dictionaries using both N-grams and word features together.

Chunk sizes of 20, 50, 100, 200, 500 and 1000 bytes were tested. For each chunk-size in turn, each language text was divided up into chunks of approximately that size (using word-boundaries), for testing with Linguini. The chunks were generated by starting at the beginning of the text and counting the appropriate number of characters, then advancing to the next word boundary. Consequently, what is reported as a test of 20-byte chunks is really a test of a range of chunk sizes from 20 to about 32 bytes; likewise 50 to about 62 and so on, but heavily skewed to the low end of each range. The word-boundary marking the end of one chunk becomes the beginning of the next chunk to be extracted. Obviously, using this method, there were more tests of the smaller-sized chunks than larger, but even with the largest size (1000 bytes) there were approximately 100 tests for most languages.

Two important consequences of this chunk extraction method should be noted. The chunks of different sizes were extracted from the same text pool. Thus for a given language, the approx. 5,000 tests of 20-byte chunks used the same material as the 100 tests of 1,000-byte chunks. Therefore, if the text was in any way unrepresentative of the language (due to, say, pollution by foreign-language fragments), the effect will be present in tests of all chunk sizes. Similarly, the tests using different dictionaries used the same sample texts with the same chunking methodology. Because of these properties of the testing method, it is appropriate to directly compare dictionaries and chunk size results.

Since the vector space method computes the angle between a vector representing the test document and a vector representing the training data of a category (language), it is necessary to determine what is used for the numerical value in each position in these vectors. In our application, it is the number of times the feature (N-gram or word) occurs in the training set, times an *inverse document frequency (idf)* weight. The term 'idf' is from the field of document retrieval; in applying it here we count each language training-set as a 'document'. Thus if feature *i* occurred (any number of times) in n_i of the different language training sets, its associated weight by literal interpretation of 'idf' would be $1/n_i$.

We performed experiments with this value, with a more aggressive weighting, $1/n_i^2$, and a less-aggressive weighting, $1/\log(1+n_i)$, and also constant *idf*=1. The first-mentioned, namely $1/n_i$ worked the best and was used in all feature-set evaluations reported here.

If feature *i* occurred m_i times in a language training set, the value we stored was the integral part of $k.m_i/n_i$. This allowed us to suppress infrequently occurring words, especially if they occurred in many other languages, and varying *k* allowed us to control this filter. Specifically, a word would not be stored if its occurrence count $m_i < n_i k$. We experimented with values of *k* from .1 to 10, and found values in the region of .3 to .5 to work best.

Running Linguini with a given dictionary on a given input text produces a *hit-list*, which is an ordered list of the languages represented in the dictionary with their scores with respect to this input. A test is considered to give the correct result if the language at the top of the hit-list is indeed the language of the test text. For a given chunk size, the percentage correct for each language was calculated. Since there was approximately the same number of tests performed for each language, at any chunk size, it was meaningful to aggregate across languages, to obtain an average percentage of correct identifications.

The 13 languages used in the training and testing of Linguini were all Western European and were chosen because they were of particular interest to IBM for inclusion in one of its text products [15]. Since these languages share etymological roots and have largely overlapping character sets, they provide a more difficult test of Linguini's classification powers than languages taken from a wider set would do. Indeed, when Oriental languages were included in the training and test sets (not reported in detail here), it was found that Linguini's average performance *improved.* (Despite the fact that some of these languages were DBCS (double-byte character set) based, Linguini used the same approach of taking sequences of bytes as N-grams. In some cases this would mean N-grams did not represent integral characters, but this did not seem to matter.) This effect occurred because these languages used largely *non*-overlapping code-points, to the extent that when Chinese text was tested, Chinese would top the hit-list and Japanese would appear at the bottom, and vice-versa. Correct identification of these languages was therefore easier than for the European set. The virtual orthogonality of the Oriental languages' character-coding representations, both from those of European languages and of each other, is illustrated in Table 1. In this Table is shown the pair-wise scalar product between the training sets in Chinese (Big 5 encoding), Japanese (Shift-JIS), Korean (KR-EUC), English, German, Danish and Norwegian. The feature-set used was that of 3-grams with unity *idf* weighting.

Table 1. Pair-wise scalar products of selected training sets.

Language	Chinese	Japanese	Korean	Danish	English	German	Norwegian	Spanish
Chinese	1	0.001	0.008	0.000	0.000	0.000	0.000	0.000
Japanese	0.001	1	0.000	0.000	0.002	0.000	0.000	0.000
Korean	0.008	0.000	1	0.000	0.000	0.000	0.000	0.000
Danish	0.000	0.000	0.000	1	0.399	0.630	0.916	0.384
English	0.000	0.002	0.000	0.399	1	0.362	0.380	0.337
German	0.000	0.000	0.000	0.630	0.362	1	0.601	0..363
Norwegian	0.000	0.000	0.000	0.916	0.380	0.601	1	0.357
Spanish	0.000	0.000	0.000	0.384	0.337	0.363	0.357	1

The training materials used in the work reported in this paper were collections of documents fetched from the Web, specifically from sources pointed to by the Human Languages page (http://www.june29.com/HLP). Commonly these were news articles, but an attempt was made to include samples of informal speech and literature too. For both English and Swedish, word frequency information was available, from the same source. In these two cases, half of the training collection was artificially constructed by generating texts with words in proportion to their normally occurring frequency, as given by the frequency tables. This approach did not generate meaningful text, but since the features examined in this exercise were words and N-grams (the N-grams not spanning words), the semantics of the texts were irrelevant. The sizes of the training sets are given in Table 2.

Table 2. Training set sizes

Language	Size (bytes)
Catalan	27,101
Danish	76,099
Dutch	58,190
English	99,743
Finnish	24,419
French	98,138
German	61,622
Icelandic	19,759
Italian	99,186
Norwegian	98,435
Portuguese	46,510
Spanish	86,041
Swedish	98,326

Most of these languages employ diacritical marks. These turn out to be useful for identification, but Linguini does not do anything special to take advantage of them. These marks are generally represented by different code-points than un-accented characters - for example, in the ISO-Latin-1 character set, the lower-case “a” is at code-point 97, while an a-circumflex “â” is at 226. Hence the trigram “bat” is automatically different from “bât”, thus helping to enhance the separation between English and French, amongst others. Linguini scans for and converts any SGML entities (such as “â” for “â”) in the input text prior to parsing into N-grams.

No specific experiments were performed to determine the effect of training set size on performance, since it is clear that, up to a point, the more training material the better; given the capacities of today’s computers, it was felt more important to optimize accuracy at the cost of one or two megabytes for a hash-table dictionary. We did find, though, that relatively short training texts are sufficient if any included languages are not related to any others in the collection. In a separate experiment, Linguini was additionally trained with a half page of Maori (the Treaty of Waitangi, about 2,300 bytes). This was sufficient to identify an average of 9 out of 10 native New Zealand place-names as Maori. However, for the set of languages of interest to us, larger training texts were used due to the overlapping nature of the vocabularies involved.

2.2. Measuring Performance against Single Features

For N-grams, we needed to determine which feature-size (i.e. value for "N") is best. Intuitively, the larger N is, the more discriminating the feature, but at the cost of a larger dictionary (since the number of N-grams is A^N, where A is the size of the alphabet). Also, using a larger value of N produces difficulties with short texts since there is an increased risk of a lack of match due to insufficient training material. Weighing these considerations, experiments were done with N taking values of 2, 3, 4 and 5.

It was also of interest to see how words would fare as features. Since the function words in a language (pronouns, prepositions, articles, auxiliaries) tend to be quite distinctive, it was felt to be desirable to perform an experiment with these too. However, since actual lists of such words were not available to the author in all of the languages studied, the heuristic of using short words (words of length 4 characters or fewer) was followed. As will be seen, the results show good performance and it is expected that using explicit lists of function words would not make any substantial difference. It might be noted, though, that these function words are very much the words that are commonly *excluded* from feature sets in the indexers of most search engines

Additional experiments were performed using both N-grams and words together as features. While it is true that N-grams would seem to subsume short words (for suitable values of N and "short"), it was decided that since words are privileged units in language texts, being coherent linguistic units which convey information above and beyond that of character sequences, they should be treated as privileged features in Linguini. However, so as not to give words too much weight in the overall process, if a character sequence is recognized both as a word and an N-gram, then it is treated solely as a word in both the indexing and matching processes (Linguini's dictionary flags whether a feature is a word or an N-gram).

Table 3. Performance averaged over all languages (percentage correct); rows correspond to feature-type, columns to chunk size in bytes.

	Chunk Size					
Feature-set	20	50	100	200	500	1,000
2-grams	68.8	86.2	93.5	97.7	98.8	100.0
3-grams	79.5	93.0	97.7	99.3	100.0	100.0
4-grams	83.6	94.3	98.2	99.6	99.9	100.0
5-grams	81.4	93.1	97.8	99.4	99.9	99.9
Words	69.7	86.6	94.7	98.1	99.9	100.0
SWords	61.3	81.5	92.1	97.1	99.6	100.0
SW+3grams	83.8	94.9	98.5	99.7	100.0	100.0
SW+4grams	84.9	95.3	98.6	99.7	99.9	100.0
W+4grams	85.4	95.6	98.7	99.7	99.9	100.0

Table 4. Performance (percentage correct) for Words and 4grams as features.

	Chunk Size					
Language	20	50	100	200	500	1,000
Catalan	71.5	91.1	97.6	100.0	100.0	100.0
Danish	66.6	85.5	95.5	99.4.0	100.0	100.0
Dutch	80.4	94.2	98.9	99.6	100.0	100.0
English	92.7	99.6	100.0	100.0	100.0	100.0
Finnish	96.6	99.6	100.0	100.0	100.0	100.0
French	94.6	99.2	99.8	100.0	100.0	100.0
German	95.3	99.5	100.0	100.0	100.0	100.0
Icelandic	94.4	99.0	99.8	99.7	100.0	100.0
Italian	94.5	99.4	100.0	100.0	100.0	100.0
Norwegian	80.2	92.9	96.8	97.9	99.1	100.0
Portuguese	90.5	99.7	100.0	100.0	100.0	100.0
Spanish	79.4	94.4	98.9	100.0	100.0	100.0
Swedish	73.6	88.7	96.1	99.6	100.0	100.0
Average	85.4	95.6	98.7	99.7	99.9	100.0

The results of these runs are presented in Table 3. It should be noted that in these and subsequent tables and graphs of performance by chunk-size, the horizontal coordinates are presented in an approximately logarithmic scale, which has the effect of appearing to soften the actual sharp rise of performance with increased chunk-size. Combining words and N-grams is shown to give better results than either feature alone. The best performance is with 4-grams and words of unrestricted length. The breakdown of results by language for these features is given in Table 4. The results in Table 3 are presented graphically in Figure 1.

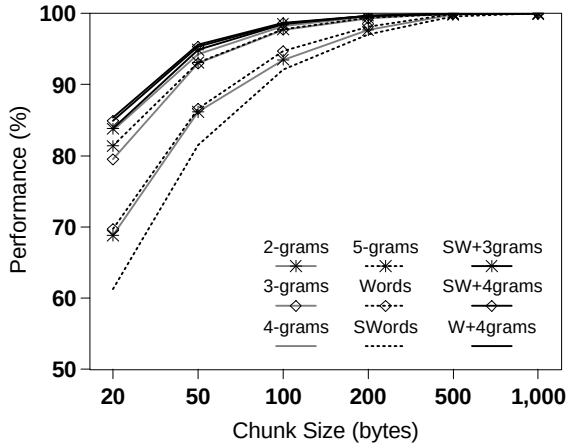

Figure 1. Plot of values in Table 3

2.3. Reduced Language Set

It is clear that the longer the input text, the better Linguini performs. This is a consistent trend, for every language tested, for every dictionary type. With the language set examined here, the bulk of the errors are from chunks in Catalan and the Scandinavian languages. The problem with Catalan is twofold: a much smaller training set (fewer documents available in Catalan than the other languages at the time the dictionary was built) and the fact that Catalan is very close to both Spanish and French. The three Scandinavian languages also suffer from a similarity problem, namely that Danish, Norwegian and Swedish are all very close to each other.

These problems influence the chunk average totals in an unfortunate manner, especially since (outside of the corresponding geographic regions) relatively few documents in those languages are generally encountered. This

highlights a difficult problem in presenting performance results for a language identifier, namely that the effective performance for an individual user should be calculated with respect to the occurrence frequencies of texts in different languages that this user experiences. If the average were to be computed by weighting the languages relative to actual occurrences, the figures might be very different. This distribution is of course unknown at this time, obviously, since the user is unspecified.

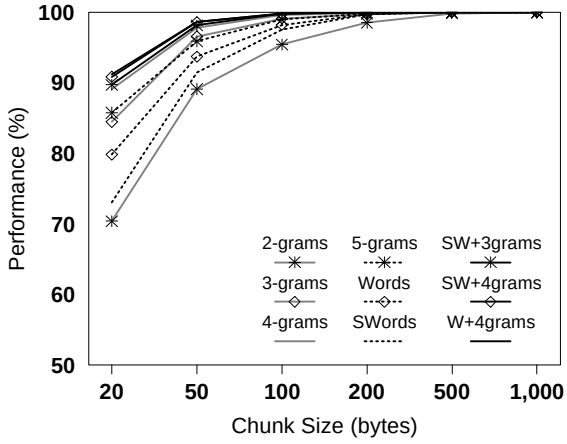

Figure 2. Reduced language set performance from values in Table 5

In order to approximate a possibly more realistic test, we considered the subset of the languages in our set with the largest speaking populations, namely English, French, German, Italian, Portuguese and Spanish. We averaged these scores for each chunk size for each dictionary type. The results are shown in Table 5 and Figure 2. These results are better than the averages reported earlier - on average, a 7.5% improvement is shown. Note that in generating this table and chart, we used the previously-reported results of testing texts in these six languages against dictionaries trained with all 13 languages. If we had generated new dictionaries with just these six languages, then the results would be better still.

Table 5. Average performance using just English, French, German, Italian, Portuguese and Spanish. Rows correspond to feature-type, columns to chunk size in bytes.

	Chunk Size					
Feature-set	20	50	100	200	500	1,000
2-grams	70.4	89.1	95.4	98.5	99.9	100.0
3-grams	84.4	96.6	99.1	99.7	100.0	100.0
4-grams	89.1	97.8	99.6	99.9	100.0	100.0
5-grams	85.8	95.9	99.0	99.8	100.0	100.0
Words	79.8	93.7	98.2	99.7	100.0	100.0
SWords	73.0	91.5	97.5	99.7	100.0	100.0
SW+3grams	89.7	98.2	99.8	100.0	100.0	100.0
SW+4grams	90.9	98.6	99.8	100.0	100.0	100.0
W+4grams	91.2	98.6	99.8	100.0	100.0	100.0

Table 6 shows, for each feature set or feature set combination, what the derived minimum input text size must be to achieve accuracy at rates of 90%, 95% and 99% or better. Interpolation was used where necessary.

Table 6. For each feature-set, the minimum input text size to generate the given accuracy is given.

Language Set	All			Reduced		
Features	90%	95%	99%	90%	95%	99%
2-grams	76	136	386	57	97	307
3-grams	43	71	181	34	46	98
4-grams	38	59	157	23	40	83
5-grams	42	70	175	32	47	100
Words	71	109	350	42	64	153
SWords	90	158	428	48	79	168
SW+3grams	37	51	142	21	39	75
SW+4grams	35	49	136	<20	36	67
W+4grams	34	48	130	<20	35	67

2.4. Results Summary

It is clear that N-grams performance is best with N=4, followed by 5 and 3, with N=2 trailing badly. Words of unrestricted length perform better than short words alone. All of the word+N-gram combination dictionaries did better than any of the single-features, with words+4grams performing the best of all.

In the following section we attempt to improve on the best results so far by taking advantage of the different circumstances when words and N-grams fare better.

2.5. Weighted Average of Two Dictionaries

In Section 2.1 we discussed the different merits of words and N-grams of different lengths as features, and the foregoing experiments have been aimed at determining the best features to use. As we can see from Tables 4 and 6, if a feature or feature combination is better than another at one particular input chunk size, it is almost certainly better at all chunk sizes. What these results do not reveal, though, is that the set of particular input chunks that contribute to a given feature's success-rate at a selected input chunk size are not always the same as those that contribute to a different feature's success rate.

What that means is that a given chunk C might generate a hit-list topped by language $\mathbf{F}_1$ followed by $\mathbf{F}_2$ for a particular feature set S_i, but its hit-list for feature set S_j might have $\mathbf{F}_3$ in first place followed by $\mathbf{F}_2$. If $\mathbf{F}_3$ is not particularly close to $\mathbf{F}_1$ and $\mathbf{F}_2$ in the first hit-list, and likewise $\mathbf{F}_1$ to $\mathbf{F}_2$ and $\mathbf{F}_3$ in the second, a case could be made that $\mathbf{F}_2$ is the best language match for C *on average*. We performed some experiments and discovered that in some circumstances an improvement in performance can be gained by judging the input against weighted averages of different dictionaries.

We decided to measure performance on a sample input text with a words-based and an N-grams-based dictionary, and average the scores in the two hit-lists. (If a language showed up in just one hit-list, the resulting score would be half of the single hit-list score.) The resulting hit-list in some instances did indeed have a different top language than both original hit-lists. The overall performance was found to improve. The first experiment was done for the short words and 3-grams dictionaries. These were chosen because they have been shown to give good results with relatively small dictionary sizes. The second experiment was done with unrestricted words and 4-grams, the best-performing dictionaries.

In both of these cases, we initially dropped the idf weighting factor. This was done because we had observed before in experiments not reported here with *idf* uniformly set to 1 that for small text chunk sizes (less than about 200 bytes), N-grams generally did better than words, while for larger chunk sizes, words did as well as or better than N-grams. [The experiments with constant *idf* were not reported because, as can be seen later in Table 10, the baseline performance was inferior.] We were keen to initially explore the use of averaging in cases where it was most likely to succeed, to get an idea of its marginal value.

Performance was measured for all of the test data previously examined, but broken down by chunk sizes from 10 to 320 bytes in the input text. The weighted average was computed as

$$\text{new hit-list} = \mu * \text{word-hit-list} + (1-\mu) * \text{N-gram-hit-list}$$

where μ was varied between 0 and 1 in steps of .1. The results of the run with short words and 3-grams are shown in Table 7 (restricted, for purposes of reducing the table size, to showing chunk sizes which are multiples of 10. However, all chunk sizes were used in the optimization calculation). The rows correspond to the input chunk size, the columns to different values of μ, and the cell values the fraction of correct results, averaged over all languages. It is clear that for

short inputs the optimum value of μ is close to 0, rising to about .5 for the longer inputs. For inputs greater than 300 words in length, it didn't seem to matter what value of μ was used, since both dictionaries usually gave perfect performance. A value of μ = .5 is suggested for all chunks >300 bytes.

By using a hill-climbing optimization procedure, it was found that a good fit with the data was obtained with

$$\mu = \frac{1}{2}\left(\frac{W}{300}\right)^{.85}, \qquad W <= 300$$

where W is the input chunk size.

The results of the run with unrestricted words and 4-grams are shown in Table 8 (the display of which is restricted again to chunk sizes which are multiples of 10). The best fitting formula for this data was found to be very similar to that for the short words/3-grams combination, but with a different exponent. The formula was:

$$\mu = \frac{1}{2}\left(\frac{W}{300}\right)^{.7}, \qquad W <= 300$$

The existence of non-trivial formulas for μ follows from the presence of a maximum value in the rows of Tables 7 and 8 at other than one of the end-points of the range. The variation across any given row is often small, but a persistent trend does stand out. Values of μ for the commonly-used chunk sizes in our experiments are given in Table 9.

Table 7. Weighted average performance of short words and 3-grams. The left-hand column gives the chunk sizes, the top row the weight μ. μ= 0 gives 3-grams exclusively, μ=1 gives short words.

	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
10	0.71	0.73	0.74	0.75	0.75	0.76	0.76	0.77	0.77	0.78	0.78
20	0.76	0.78	0.78	0.79	0.81	0.8	0.78	0.77	0.77	0.77	0.76
30	0.84	0.85	0.85	0.85	0.85	0.84	0.83	0.82	0.81	0.8	0.79
40	0.89	0.89	0.89	0.89	0.88	0.87	0.86	0.85	0.85	0.82	0.81
50	0.89	0.9	0.9	0.9	0.89	0.89	0.89	0.88	0.87	0.85	0.84
60	0.89	0.9	0.9	0.9	0.9	0.9	0.89	0.88	0.86	0.85	0.82
70	0.93	0.93	0.93	0.92	0.92	0.92	0.91	0.91	0.9	0.89	0.87
80	0.9	0.93	0.93	0.94	0.93	0.92	0.92	0.93	0.92	0.9	0.86
90	0.95	0.95	0.96	0.95	0.95	0.95	0.95	0.94	0.94	0.92	0.91
100	0.94	0.95	0.95	0.95	0.95	0.97	0.97	0.96	0.95	0.93	0.91
110	0.96	0.96	0.96	0.96	0.95	0.95	0.95	0.95	0.95	0.94	0.93
120	0.99	0.99	0.99	0.99	0.99	0.99	0.97	0.96	0.96	0.95	0.92
130	0.95	0.95	0.96	0.96	0.95	0.95	0.95	0.95	0.95	0.95	0.94
140	0.9	0.91	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.91	0.87
150	0.96	0.97	0.97	0.97	0.97	0.96	0.96	0.96	0.96	0.95	0.95
160	0.95	0.96	0.96	0.96	0.96	0.95	0.97	0.96	0.96	0.96	0.96
170	0.96	0.96	0.96	0.97	0.96	0.97	0.97	0.97	0.97	0.97	0.96
180	1	1	1	1	1	1	1	1	1	1	1
190	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.96
200	1	1	1	1	1	1	1	1	1	1	0.98
210	0.97	0.98	0.98	0.98	0.97	0.97	0.97	0.97	0.96	0.95	0.95
220	1	1	1	1	1	1	1	1	1	1	1
230	0.98	0.98	0.99	0.98	0.98	0.98	0.99	0.98	0.98	0.97	0.97
240	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97
250	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98
260	0.95	0.95	0.95	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98
270	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.99	0.98	0.98	0.98
280	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.95
290	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97
300	0.91	0.91	0.91	0.91	0.91	0.91	0.91	0.91	0.91	0.91	0.91
310	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
320	0.96	1	1	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96

Table 8. Weighted average performance of words and 4-grams. The left-hand column gives the chunk sizes, the top row the weight μ. μ= 0 gives 4-grams exclusively, μ=1 gives words.

	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
10	0.8	0.81	0.81	0.81	0.81	0.8	0.8	0.8	0.81	0.81	0.82
20	0.82	0.82	0.81	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.79
30	0.89	0.89	0.89	0.89	0.88	0.87	0.86	0.85	0.84	0.83	0.82
40	0.9	0.91	0.91	0.9	0.9	0.88	0.86	0.85	0.84	0.83	0.83
50	0.92	0.93	0.92	0.92	0.92	0.91	0.9	0.9	0.89	0.88	0.86
60	0.94	0.94	0.95	0.94	0.91	0.91	0.91	0.9	0.9	0.87	0.84
70	0.94	0.95	0.95	0.95	0.95	0.94	0.93	0.92	0.91	0.9	0.89
80	0.95	0.98	0.98	0.98	0.97	0.97	0.96	0.95	0.93	0.9	0.87
90	0.96	0.97	0.97	0.97	0.96	0.96	0.96	0.95	0.94	0.94	0.93
100	0.98	0.97	0.98	0.97	0.97	0.96	0.96	0.96	0.95	0.94	0.91
110	0.98	0.98	0.98	0.97	0.97	0.97	0.96	0.96	0.96	0.95	0.94
120	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.94	0.94	0.92	0.91
130	0.98	0.98	0.98	0.98	0.97	0.97	0.97	0.96	0.96	0.95	0.95
140	0.94	0.94	0.93	0.93	0.93	0.92	0.92	0.92	0.92	0.92	0.89
150	0.98	0.98	0.98	0.98	0.98	0.97	0.97	0.97	0.97	0.96	0.95
160	1	1	1	1	1	1	1	0.99	0.98	0.98	0.98
170	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.97	0.97
180	1	1	1	1	1	1	1	1	1	1	1
190	0.99	0.99	0.99	0.98	0.98	0.98	0.98	0.98	0.98	0.97	0.97
200	1	1	1	1	1	1	1	1	0.97	0.97	0.97
210	0.99	0.99	0.99	0.99	0.99	0.98	0.98	0.97	0.97	0.97	0.96
220	1	1	1	1	1	1	1	1	1	1	1
230	1	1	1	1	0.99	0.99	0.99	0.99	0.99	0.98	0.97
240	1	1	1	1	1	1	1	1	1	1	1
250	0.99	0.99	0.99	0.99	0.99	0.98	0.98	0.98	0.98	0.98	0.98
260	0.98	0.98	0.98	1	1	1	1	0.98	0.98	0.98	0.98
270	1	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
280	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.95
290	0.98	0.99	0.99	0.99	0.99	0.99	0.98	0.98	0.98	0.98	0.97
300	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96
310	1	1	1	1	1	1	1	1	1	0.99	0.99
320	1	1	1	1	1	1	1	1	1	1	1

Table 9. Values of weight μ for selected chunk sizes

Chunk Size	20	50	100	200	500	1000
μ for Short Words/3grams	0.05	0.11	0.20	0.35	0.50	0.50
μ for Words/4grams	0.08	0.14	0.23	0.38	0.50	0.50

Given the values of μ in Table 9, we can interpolate Table 7 to find the scores for weighted mixtures of the 3-grams and short words dictionaries, and Table 8 to find the scores for weighted mixtures of the 4-grams and words dictionaries. These computations are done for the complete set of languages considered, so extend the data presented in Table 4 and Figure 1.

These results are presented, along with non-averaged scores using *idf*=1, in Table 10. We see that at the 20-byte input level, performance was shown to increase by about 9% over the best results with single dictionaries.

Table 10. Scores for weighted average and component dictionaries, along with dictionary sizes. In the left column, a '+' indicates a dictionary with two kinds of features, a '/' indicates a weighted average of two different dictionaries

Features	20	50	100	200	500	1000
3-grams	71.28	87.28	93.82	96.68	98.74	99.16
4-grams	74.38	89.05	95.51	98.18	99.66	100.00
Words	62.72	78.10	88.87	94.80	98.29	99.50
SWords	56.26	75.12	87.38	94.25	97.85	99.59
SW+3grams	73.16	88.03	93.83	96.38	98.32	98.74
W+4grams	75.25	89.89	95.96	98.41	99.66	99.86
SW/3grams	76.65	89.72	94.80	100.00	100.00	100.00
W/4grams	81.96	92.52	97.70	100.00	100.00	100.00

Since the performance results with our standard *idf* of $1/n_i$ were universally better than the baseline of *idf*=1, we were hopeful that applying the averaging technique to the superior results shown earlier would give even better performance. However, we found no benefit to using the averaging procedure. The entire advantage went away since the performance was always monotonic in μ. [On the positive side, the optimum averaged performance was never less than the better of N-grams or words, by setting μ=0 or 1 as appropriate.]

We think that the reason for this result is that whatever benefit is introduced by averaging is already captured in some fashion by using *idf* term weights. To continue the example from the beginning of this Section, if language $\mathbf{F}_2$ beats $\mathbf{F}_3$ considerably when using S_i and beats $\mathbf{F}_1$ when using S_j, for a given text chunk, then it may not have a great overlap in features with $\mathbf{F}_1$ and $\mathbf{F}_3$ (especially considering that the feature-sets we are considering are by no means mutually independent). Hence it is possible that $\mathbf{F}_2$ matches with the test input in a larger number of features that are absent from other languages, than are $\mathbf{F}_1$ and $\mathbf{F}_3$. In other words, $\mathbf{F}_2$'s features that are present in the test input will have relatively high *idf* weights. This suggests a general correlation between a language doing well with the averaging technique and doing well with *idf* weighting.

Since no gain was noted with averaging and variable *idf* combined, and since using variable *idf* alone gives better performance than averaging alone, we conclude that the averaging technique is not generally useful.

3. Multilingual Documents

3.1. Vector-Space treatment of Monolingual Documents

Let us initially assume that document **D** is written in a single language $\mathbf{F}_j$, which is one of a set $\{\mathbf{F}_i\}$ of *n* reference languages, represented by feature vectors $\{\mathbf{f}_i\}$. **D** can be represented by a feature vector **d**. The objective of identifying $\mathbf{F}_j$ is achieved by determining the $\mathbf{f}_j$ which is closest to **d**. The similarity measure of choice is the cosine of the angle between vectors, and this is computed via the scalar product. Thus

$$\cos\theta_i = \frac{\mathbf{d}.\mathbf{f}_i}{|\mathbf{d}||\mathbf{f}_i|} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (1)$$

$\cos\theta_i$ is computed for every $\mathbf{f}_i$, and the $\mathbf{F}_i$ corresponding to the smallest θ_i is selected as the system's estimate of the language of **D**. $\cos\theta_i$ is the *score* for language $\mathbf{F}_i$.

It is common practice to weight the vector elements using a formula such as *tfxidf* [12]. as explained in Section 2.1. An element in feature vector $\mathbf{f}_i$ is the number of times the element occurs in the training set (*tf*) multiplied by a function of the number of feature sets the element occurs in (*idf*). In the worked example later in this Section, though, in order to make the mechanics of finding the correct category in mixed-language cases more clear, the *idf* term is dropped. This does not affect the analysis; in fact, in the particular domain we have been working in, we found that a large number of features made at least one appearance in most of the training sets (part of this effect was due to languages being historically related, part was due to borrowing and pollution, and part to chance). This meant that the *idf* component was

fairly close to constant anyway. The numerical values in the feature vectors, then, were simply the *tf* terms, namely the number of times the feature occurred in the training data for the respective category.

The identification procedure for monolingual documents is straightforward. For a test document **D** and language set $\{\mathbf{F}_i\}$, the scores $\{\cos\theta_i\}$ are computed, sorted and presented to the user in the form of a *hit-list*, analogous to the hit-lists returned by search engines. The language at the top is the one which the system proposes for **D**.

The analysis in the previous section is the standard vector-space or centroid-based approach to categorization. It is sometimes called the centroid-based approach since the feature vector representing a category (language) is effectively the centroid of the set of sample vectors that represent the training documents. The question arises of what to do if a category consists of two or more distinct but similar sub-categories, such as language dialects: should the training data be merged to generate a single feature vector that represents the average document, but maybe doesn't closely resemble any, or should a separate feature vector be generated for each centroid?

Norwegian is a particularly interesting language to study in this regard, for two reasons. It has two major dialects (Bokmal and Nynorsk), but Bokmal is closer to Danish than to Nynorsk, in cosine-distance. These facts make it quite difficult to get a categorizer to achieve high precision with short Norwegian texts.

Initially, a single Norwegian category was established with an approximately equal number of Bokmal and Nynorsk documents. In the testing procedures reported earlier, Norwegian texts were found to be correctly identified consistently less often than texts in any of the other languages under consideration. To fix this, separate categories were established for Bokmal and Nynorsk, but both were labelled Norwegian. It was found that performance improved considerably. The comparative performances for the "words + 4-grams" feature-set are shown below in Table 11, but all feature-sets tested exhibited such improvement. The performance numbers in Table 3 and Figure 1 shown earlier all were generated with the two separate Norwegian categories.

The typical error case that was fixed by taking this approach is shown in Figure 4. This Figure illustrates how a document **d** is found to be closer to Danish than a 50-50 mixture of Bokmal and Nynorsk, but is closest of all pure Bokmal. (The Danish and **d** vectors are depicted in the plane of Bokmal-Nynorsk for illustrative purposes only.)

Table 11. Comparison of performance by treating Norwegian dialects as Combined or Separate.

	20 bytes	50 bytes	100 bytes	200 bytes	500 bytes	1000 bytes
Combined	71.1	84.3	92.3	95.4	98.3	100
Separate	80.2	92.9	96.8	97.9	99.1	100

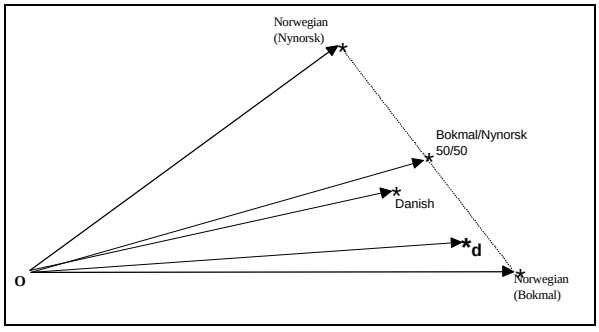

Figure 4. Illustration of how, if the two major Norwegian dialects are combined into a single category, document d is incorrectly identified as Danish, yet if the dialects are separated d is correctly identified as Bokmal.

Documents often include words in more than one language. When a second language makes up a significant proportion of the text in the document, it would be desirable to identify it. The same also goes for third and other languages, but these cases will not be dealt with explicitly here since the treatment is a simple extension of that for documents in two languages.

At first sight, all one need do for languages in two documents is to select the top two languages in the hit-list. This is not the right thing to do, for two reasons. Firstly, suppose a document **D** is written in Norwegian and English, with

Norwegian predominating. Presumably, Norwegian will top the hit-list, but the second place will more than likely be occupied by Danish. This is because the second place in the hit-list is occupied by the language which has the second-best overall match with **D**, not the language which best matches the features 'left over' from the first-place match. [There is in fact no concept of 'left over' features in vector-space based categorizers.] In other words, the system does not know *a priori* that the document is in two languages.

A further problem is that it is possible that neither of the two languages in a document occupies the first position in the hit-list. The following real example illustrates this problem. The document in Figure 5 is a section of a Dutch web document polluted with some English nursery-rhyme fragments. When analyzed by Linguini, the hit-list in Figure 6 was generated. It is seen that the top-nominated language is neither Dutch nor English, but Danish. How can this occur?

titel Jan den Hollander NIEUW Waarom krijg ik met maar halve plaatjes Tim van paulusma Waarom mag ik het caffee niet in Robert Derksen nieuw Reply to Robert NIEUW ONZIN wat hier verteld wordt Ebenezer Kakka Ouwe cafe Sander van Drooge of dit adres Wouter Voortman mij lukt het ook niet Marike Maijers
went up the hill to fetch a pail of water
fell down and broke his crown and jill came tumbling after

Figure 5. Example of document in Dutch and English which gets classified as Danish in the absence of mixed-language functionality.

0.314	Danish
0.307	English
0.306	Dutch
0.276	Norwegian
0.262	German
...	

Figure 6 Top of hit-list generated by processing document in Figure 5. Numbers are cosine values

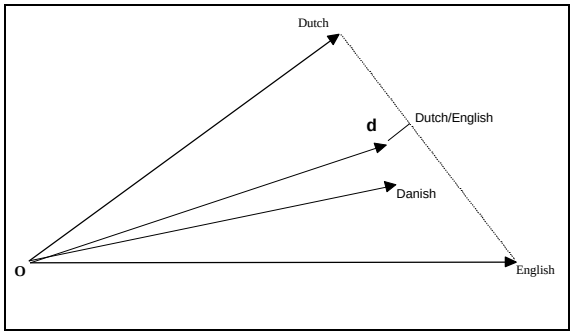

Figure 7. Illustration of how a document d in Dutch and English can appear to be closest to Danish if the mixed-language functionality is missing.

Figure 7 illustrates this situation, again via a 2-d projection. In this case the document **d** is closer to Danish than to either Dutch or English, but is closer yet to a particular mixture of Dutch and English. It is clear that it is an objective of a successful system to identify this mixture.

To demonstrate the mechanics of this phenomenon more clearly, the following very simple example was contrived. *idf* values of 1 are used throughout to enhance the clarity of the worked example. Suppose the three languages French, Italian and Spanish are of interest, and the feature sets just consist of two or three common words. The sets, given in Table 12, intentionally overlap.

Table 12. Complete feature-sets for three languages in an illustrative example.

Language	Features	English Equivalent
French	le	the (masc. sing.)
	mes	my (plural)
	son	his (masc. sing)
Italian	il	the (masc. sing.)
	le	the (fem. plural)
Spanish	mes	month
	son	are (3rd person)

Using the arbitrary ordering (il, le, mes, son) we generate the following feature vectors:

French: (0, 1, 1, 1)
Italian: (1, 1, 0, 0)
Spanish: (0, 0, 1, 1)

Suppose now that the document "il le mes son" is processed. It will have the feature vector (1,1,1,1). The following cosine values are calculated:

French: $\frac{3}{\sqrt{12}} = \frac{\sqrt{3}}{2} = 0.866$

Italian: $\frac{2}{\sqrt{8}} = \frac{1}{\sqrt{2}} = 0.707$

Spanish: $\frac{2}{\sqrt{8}} = \frac{1}{\sqrt{2}} = 0.707$

French is at this time the leading candidate language for the document. However, if we allow that the document is an equal mixture of two languages, we consider the three "virtual" mixed languages, whose feature vectors and scores are as follows:

French/Italian: (½, 1, ½, ½) $\frac{2\frac{1}{2}}{\sqrt{4 \times 1\frac{3}{4}}} = \frac{5}{2\sqrt{7}} = 0.945$

French/Spanish: (0, ½, 1, 1) $\frac{2\frac{1}{2}}{\sqrt{4 \times 2\frac{1}{4}}} = \frac{5}{6} = 0.833$

Italian/Spanish: (½, ½, ½, ½) $\frac{2}{\sqrt{4 \times 1}} = 1$

While the French/Italian mixture is an improvement on French, we see that Italian/Spanish is an even better match, indeed a perfect one. This small and contrived example demonstrates that the best combination of languages is not guaranteed to contain the best language found in the monolingual context. However, the calculation of the best virtual mixed language here assumed that if the document was in a mixture of two languages, the proportions of the component languages were equal. It is shown in the derivations below that no such assumption is necessary for correct identification

of the best language or language mixture, and in fact, if a language mixture is found, then the relative proportion of the components is discovered too.

3.2. Bilingual Documents

A document may be written in a mixture of languages from $\{\mathbf{F}_i\}$, in which case it is desired to find the component languages. A complicating factor is that it is not known what are the relative proportions of the component languages, nor how many components there are. We will initially just consider bilingual documents, but extend the analysis to the general case later. Now, in the case of bilingual documents, if the component proportions were known, it would be straightforward to create a feature vector from the appropriate weighted mean of the components - this is what we call a *virtual mixed language* - and measure the cosine of the angle between it and the document. However, we don't know the proportion in advance.

In the following, we will denote the two languages being examined as $\mathbf{F}_i$ and $\mathbf{F}_j$ and we will use $\mathbf{f}_i$ and $\mathbf{f}_j$ to denote their respective feature vectors. A document **D,** which is in a mixture of $\mathbf{F}_i$ and $\mathbf{F}_j$, is modelled as a vector **d** which approximates $\mathbf{k} = \alpha\mathbf{f}_i + (1-\alpha)\mathbf{f}_j$ for some *mixing weights* α and $1-\alpha$ to be determined. **k** represents the virtual mixed language **K** (which is, of course, not in $\{\mathbf{F}_i\}$). The problem, then, is to find the **k** (in other words the *i*, *j* and α) that minimizes the angle between **k** and **d**. We will show first how to determine α for a given pair of $\mathbf{f}_i$ and $\mathbf{f}_j$ (where $i \neq j$).

Without loss of generality, $\mathbf{f}_i$ and $\mathbf{f}_j$ are unit vectors. By reference to Figure 8, we see that **k** is the projection of some multiple β of **d** on the plane containing $\mathbf{f}_i$ and $\mathbf{f}_j$. (Any other vector in the plane will have a greater angle with **d**.)

In other words $\mathbf{p} = \beta\mathbf{d} - \mathbf{k}$ is perpendicular to the plane, and hence to $\mathbf{f}_i$ and $\mathbf{f}_j$ individually. Hence we get:

$$\mathbf{f}_i.\mathbf{p} = \mathbf{f}_i.(\beta\mathbf{d}-\mathbf{k}) = \beta\,\mathbf{f}_i.\mathbf{d} - \alpha\,\mathbf{f}_i.\mathbf{f}_i - (1-\alpha)\mathbf{f}_i.\mathbf{f}_j = \mathbf{0}$$
$$\mathbf{f}_j.\mathbf{p} = \mathbf{f}_j.(\beta\mathbf{d}-\mathbf{k}) = \beta\,\mathbf{f}_j.\mathbf{d} - \alpha\,\mathbf{f}_j.\mathbf{f}_i - (1-\alpha)\mathbf{f}_j.\mathbf{f}_j = 0$$

Eliminating β, we get

$$\alpha = \frac{\mathbf{f}_i.\mathbf{d} - \mathbf{f}_j.\mathbf{d}*\mathbf{f}_i.\mathbf{f}_j}{(1 - \mathbf{f}_i.\mathbf{f}_j)*(\mathbf{f}_i.\mathbf{d} + \mathbf{f}_j.\mathbf{d})} \quad \text{..... (2)}$$

From this, it can readily be verified as a check that if **d** is a multiple of $\mathbf{f}_i$, then $\alpha = 1$, or if **d** is a multiple of $\mathbf{f}_j$, then $\alpha = 0$, as expected. We discuss the development of this formula for three of more languages in the next section.

For a given pair $\mathbf{f}_i$ and $\mathbf{f}_j$ we have seen how to compute the α which generates the optimal virtual mixture relative to **d**. We now need to compute the cosine similarity measure in order to rank this virtual mixed language with the other languages. This value is

$$\frac{\mathbf{d}.\mathbf{k}}{|\mathbf{d}|\,|\mathbf{k}|} \quad \text{..... (3)}$$

where

$$\mathbf{k} = \alpha\,\mathbf{f}_i + (1-\alpha)\mathbf{f}_j,$$

so

$$\mathbf{d}.\mathbf{k} = \alpha\,\mathbf{f}_i.\mathbf{d} + (1-\alpha)\mathbf{f}_j.\mathbf{d} \quad \text{..... (4)}$$

and

$$|\mathbf{k}|^2 = \alpha^2\,\mathbf{f}_i.\mathbf{f}_i + 2\alpha(1-\alpha)\mathbf{f}_i.\mathbf{f}_j + (1-\alpha)^2\,\mathbf{f}_j.\mathbf{f}_j$$

$$= \alpha^2\,|\mathbf{f}_i|^2 + 2\alpha(1-\alpha)\mathbf{f}_i.\mathbf{f}_j + (1-\alpha)^2\,|\mathbf{f}_j|^2 \quad \text{..... (5)}$$

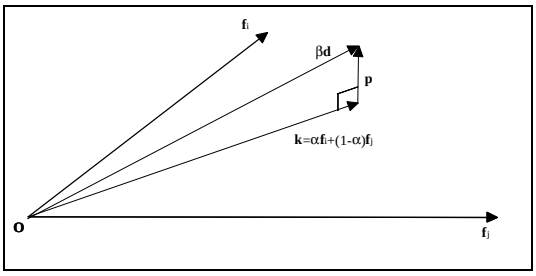

Figure 8. Bilingual document modelled as an approximation to the weighted mean of two language vectors

3.3 Trilingual Documents and Beyond

The extension of the preceding development to three or more languages is straightforward. For any three languages $\mathbf{F}_i$, $\mathbf{F}_j$ and $\mathbf{F}_l$ we need three mixing weights from two parameters, α and γ, say, and posit

$\mathbf{k} = \alpha\mathbf{f}_i + \gamma\mathbf{f}_l + (1-\alpha-\gamma)\mathbf{f}_j$ .....(6)

As before, setting $\beta\mathbf{d} - \mathbf{k} = 0$ and forming the scalar product in turn with $\mathbf{f}_i$, $\mathbf{f}_j$ and $\mathbf{f}_k$ gives us three simultaneous equations. Eliminating β we find that

$$\alpha = \frac{(-\mathbf{f}_i\mathbf{d}+\mathbf{f}_i\mathbf{f}_l*\mathbf{f}_i\mathbf{d}-\mathbf{f}_i\mathbf{f}_j*\mathbf{f}_l\mathbf{d}*\mathbf{f}_j\mathbf{f}_l+\mathbf{f}_i\mathbf{f}_j*\mathbf{f}_j\mathbf{d}+\mathbf{f}_i\mathbf{d}*\mathbf{f}_j\mathbf{f}_l^2-\mathbf{f}_i\mathbf{f}_l*\mathbf{f}_j\mathbf{d}*\mathbf{f}_j\mathbf{f}_l}{X}$$

and

$$\gamma = \frac{(-\mathbf{f}_l\mathbf{d}+\mathbf{f}_l\mathbf{f}_i*\mathbf{f}_i\mathbf{d}-\mathbf{f}_l\mathbf{f}_j*\mathbf{f}_i\mathbf{d}*\mathbf{f}_j\mathbf{f}_i+\mathbf{f}_l\mathbf{f}_j*\mathbf{f}_j\mathbf{d}+\mathbf{f}_l\mathbf{d}*\mathbf{f}_j\mathbf{f}_i^2-\mathbf{f}_l\mathbf{f}_i*\mathbf{f}_j\mathbf{d}*\mathbf{f}_j\mathbf{f}_i}{X}$$

where the common denominator X is

$$\begin{aligned}&\mathrm{f}_i.\mathrm{d}(1-\mathrm{f}_j.\mathrm{f}_l)(\mathrm{f}_i.\mathrm{f}_l+\mathrm{f}_i.\mathrm{f}_j+\mathrm{f}_j.\mathrm{f}_l-1)+\\&\mathrm{f}_j.\mathrm{d}(1-\mathrm{f}_i.\mathrm{f}_l)(\mathrm{f}_i.\mathrm{f}_j+\mathrm{f}_j.\mathrm{f}_l+\mathrm{f}_i.\mathrm{f}_l-1)+\\&\mathrm{f}_l.\mathrm{d}(1-\mathrm{f}_i.\mathrm{f}_j)(\mathrm{f}_i.\mathrm{f}_l+\mathrm{f}_j.\mathrm{f}_l+\mathrm{f}_i.\mathrm{f}_j-1)\end{aligned}$$

[It can be verified that if $\mathbf{d}$ is a multiple of $\mathbf{f}_i$, then $\alpha = 1$, $\gamma = 0$; if $\mathbf{d}$ is a multiple of $\mathbf{f}_j$, then $\alpha = 0$, $\gamma = 0$; if $\mathbf{d}$ is a multiple of $\mathbf{f}_l$, then $\alpha = 0$, $\gamma = 1$.]

From this, we can easily recalculate equations (4) and (5) from (6) and hence the cosine similarity measure in (3).

The extension to four or more languages can be done exactly analogously, but this may not be desirable, for the following reasons. In the bilingual case, α and $1-\alpha$ give us the relative proportions of the two component languages. A value of α very close to 0 or 1 might indicate a true but lop-sided mixture, but in our experience we have found that almost always it is a situation either of pollution by the less-well-represented language or a confluence of effects of imperfect training sets and variations in writing styles. A typical example of pollution occurs when a continental European newspaper reviews an British or American movie or music CD, and inescapably includes many instances of names and titles that conform to English lexical patterns. The computer/Internet field is another one where English words abound in non-English articles.

We claim that it is generally not useful to identify such documents as mixed. The approach we take to avoid such "false alarms" is to discard tentative mixtures wherever α is in the range 0-.1 or .9-1.0. This still leaves a wide range (80%) of possible values for α for bilingual mixtures to be considered. However, if we keep the (arbitrarily chosen) .1 safety margin then as the number of languages that might be in the mixture increases, the "wriggle-room" for the mixing weights drops dramatically (a more quantitative description depends on having the mixture's probability distribution, which we don't know). That, in combination with the a priori unlikelihood of documents with substantial sections in several languages, leads us to not test for such documents in practice.

3.4. Computational Considerations

Returning to the bilingual case, we have shown how to compute α for a given $\mathbf{f}_i$ and $\mathbf{f}_j$, but not how to select the best $\mathbf{f}_i$ and $\mathbf{f}_j$. In theory, every pair of languages could be selected in turn, the corresponding α computed as above, and the associated cosine value computed of $\mathbf{d}$ with the virtual mixed feature vector $\mathbf{k}$. In practice, a simple heuristic can be used to cut down on the computation required.

It is observed, and it is intuitively obvious, that if a document **D** is in a mixture of two languages, the individual component languages will be close to **D** - probably closer than most or all other languages. Additionally, even if it is suspected that **D** is in two languages, it must still be tested for monolinguality. Thus it is still necessary to compare **D** against all of the $\{\mathbf{F}_i\}$ individually. If this is done first, and a tentative *hit-list* is generated of the sorted $\mathbf{F}_i$ with corresponding cosine similarity values, then we can select the top *m* hit-list entries as candidates for components of the putative mixture. Then each pair from this set is considered in turn. The score for each such pair is computed. Practically speaking, only the best such mixed score is usually of interest, and then only if it beats the best monolingual score. In such cases, the bilingual pair and its score can be inserted at the top of the hit-list.

A value for *m* needs to be selected. It has been found in practice with 25 languages in the set $\{\mathbf{F}_i\}$, that when bilingual documents are tested, the individual component languages always show up in the top 5 of the single-language hit-list. Most of the time, the components are #1 and #2 on the list, not surprisingly, or otherwise #1 and #3. However, this is not always the case, as was discovered for the document in Figure 5 with monolingual hit-list in Figure 6. When the bilingual processing described above is performed on this document, the Dutch-English virtual mixture is found to have an even better score than the previous best (Danish) - the resulting hit-list is shown in Figure 9.

When Linguini is trained, hash-tables are generated containing all of the features found in the corresponding training set, along with the weighted feature-counts. In order to facilitate the evaluation of equation (1), the sum of the squares of the weighted feature-counts for each feature vector $\mathbf{f}_i$ is calculated and stored at training time too. This value is the square of the factor $|\mathbf{f}_i|$. To identify a test text **D** with feature-vector **d**, then, **D** is parsed into features and each one is looked up for each trained language, and (1) is computed directly. Thus the computational complexity of a monolingual analysis is proportional to the number of features in the test text **D** multiplied by *n*, the number of languages trained for.

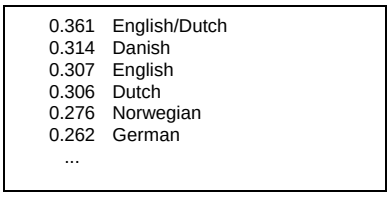

Figure 9. When multi-lingual processing is performed, the hit-list in Figure 6 gets amended by the appearance of the virtual English/Dutch combination in top position.

Training is performed serially by language. To train on a language, several texts in the language are concatenated together and presented to Linguini, which not only computes and stores all of the features with counts as described above, but also computes the scalar product of the training text as if it were a test text with all previously-trained languages, and stores this too (i.e. $\mathbf{f}_i.\mathbf{f}_j$ for all pairs of *i* and *j*). Now, on examination of Equations (2) and (5), it is seen that the optimal α used to weight a pair of language vectors $\mathbf{f}_i$ and $\mathbf{f}_j$, and the distance of **d** from the corresponding optimal virtual mixed language are computed from the scalar product of **d** with each of these ($\mathbf{f}_i.\mathbf{d}$ and $\mathbf{f}_j.\mathbf{d}$) and the scalar product of $\mathbf{f}_i$ and $\mathbf{f}_j$ themselves ($\mathbf{f}_i.\mathbf{f}_j$). All of these quantities are either computed during the monolingual phase of identifying **D**, or are precomputed during training. Hence the extra computation necessary for bilingual processing is insignificant.

3.5 Short Embedded Foreign Texts

A document that has a relatively small amount of material in a second language – a single quotation, for example – will not be determined by Linguini to be a bilingual document, which is technically the correct behavior. It is also the correct way to configure a LI system since lowering thresholds to detect a very small presence of foreign material will likely give false alarms when foreign names are present, or possibly if an author uses a particular writing style that has word and N-gram statistics that vary substantially from the training-set centroid. However, it might still be desirable to detect such passages, in order to automatically translate them, for instance.

Linguini can be used to detect such short sequences, but the exact parameters of such an algorithm cannot be determined in advance, since they depend on the average size of the anticipated embedded foreign-language texts, and the computational resources available.

Ideally, every word could be examined to determine its language, but since Linguini uses statistics rather than complete dictionaries for each language under consideration, this process will be too noisy. Examining each sentence will be much

more accurate, at the expense of computation time. [N.B. If Linguini is run as a *hot process*, then processing each sentence individually will not take much longer than processing an entire document; otherwise a system initialization cost will be incurred for each sentence.]

A compromise would have intermediate-sized chunks of text, such as paragraphs, processed in turn. If the chunk is entirely in one language, then Linguini will identify it so. If it is clearly in two languages, then Linguini will detect that too. If there is a small amount of embedded foreign language text, then the degree of match with the majority language centroid will be less than typically found for a chunk of that size. In the latter two cases, the chunk can be broken into smaller chunks such as sentences and Linguini run on each of them individually. If the smaller chunk is determined to be not entirely in one language then it too can be subdivided (into clauses, say, or possibly words), but with the understanding that at this granularity the ambiguity can be quite high. On the other hand, the automatic translation system might be able to resolve this ambiguity; this is an area for further investigation.

3.6 Application to Subject-based Categorization

The phenomenon of mixed categories occurs too in content-based text categorization. If the top categories of the content-based hit list have relatively good scores, it may be because the document indeed belongs to several categories, or a mixed-category situation analogous to the mixed-language case analyzed here might apply. We might conclude, then, that better performance will be achieved with centroid-based categorizers if broad categories are split into several smaller sub-categories, even if this is done entirely within the system and users are not made aware of the finer internal detail. This may also explain the good performance shown by k-Nearest-Neighbor categorizers (see for example Yang, 1994). Such categorizers work by first assigning categories to a set of known documents in the training phase. When an unknown document is to be categorized, its distance to all training documents is computed according to a matching function typically similar to the one presented here. Each document then "votes" for its own category with a weight proportional to its similarity to the test document; the category with the most votes wins.

A desirable feature of such a system would be to categorize a training document correctly if presented as a test document. Thus the scoring algorithm should combine the votes non-linearly, so giving an edge to a small number of really good matches over a larger number of less-good matches.

Thus a category labelled "Physics & Chemistry", say, might in practice be a location for documents about either science individually, rather than about both. Centroid-based categorizers would construct a virtual category mid-way in feature space between the locations of a Physics and a Chemistry category, and documents about either subject would have a fairly good match with this centroid, but not necessarily any better than "Astronomy", say, or "Biochemistry". On the other hand, with a kNN categorizer Physics documents would likely match strongly with other Physics documents, and likewise Chemistry, thus giving better accuracy.

4. Summary

The first half of this paper reported the results of examining the performance of a vector-space based language classifier on different text input sizes, using different features and combinations of features. The major contributions of these tests are a determination of the relative rank-ordering of these feature-sets, and an expectation of identification accuracy as a function of input text length.

It is intuitively true that the longer the input text, the better the performance should be, and this was clearly borne out. A more interesting question was which of a predefined set of features was the best for Linguini to use. The features which were available were N-grams, where N ranged from 2 to 5, and words, either short (<=4 characters) or of unrestricted length. Certain combinations of these features were also tested.

For the N-grams alone, it was shown that 4-grams gave the best results. For words alone, it was shown that words of unrestricted length did better than short words. Perhaps unsurprisingly, when dictionaries were generated by using both N-grams and words as features, the best performance was with the combination of 4-grams and words of unrestricted length. The average performance was 85.4% for 20-byte inputs, rising to over 99% for 130 bytes and up. By restricting the languages involved in the test, even better performance was shown: 91.2% for 20-byte inputs rising to over 99% for inputs of size 67 bytes and up, and similar improvements shown for the majority of the feature-sets investigated. This

language restriction was not done to try to artificially boost Linguini's scores, but rather to show what the performance measurements would likely be in a realistic setting.

One problem with the testing procedure used for a system whose identification performance varies by language is that the average derived performance is a function of the relative mix of test strings used, and that what constitutes a realistic or typical mix will vary from user to user. Another problem lies with the interpretation of the results with very short text strings. It is not completely clear to what extent short text chunks extracted from documents mirror the feature distributions to be found in short utterances, such as queries. More investigation in this area needs to be carried out.

An attempt was made to improve on the best results achieved by noting that in some circumstances N-grams fare better than words, and vice versa, and that a dynamic weighted average might improve performance. While it was shown that there was indeed a gain in performance when *idf* was constant, these results were inferior to those with variable *idf* with no averaging. Averaging made no difference to the variable-*idf* results, and so is not thought to be useful in practice.

The second half of the paper was devoted to the issue of multiple languages. Firstly, it was shown, through the example of Norwegian, that if a language exists as two (or more) major dialects, improved performance can be achieved by generating separate categories for each dialect (even though they may be labelled the same) over generating a single combined category. Secondly, it was shown that treating bilingual documents as monolingual can give counter-intuitive results. An algorithm is presented for detecting and determining the nature of multilingual documents, including the relative proportions of the component languages. This algorithm is shown not to have any higher computational complexity than that for single language identification; indeed, in practice the extra required processing time was not measurable. Finally, a discussion was given of the applicability of these results to subject-based document categorization.

One area that has not been explored in the work reported here is that of determining the minimum feature set size necessary to give good results. The weighting formula that we used does a little truncation of the feature set, but no determined effort was undertaken to see how far we could go. Yang [29] reports success with removal of up to 98% of features using a corpus of Reuters documents for text categorization. We suspect that we may be able to achieve similar feature-set reduction for language identification.

5. Acknowledgments

I would like to thank Bart Emanuel for inspiring this work and Rong Chang, Bob Mack and Alan Marwick for supporting it. I am grateful to Herb Chong for supplying dictionary support code, Zhihao Zhang for running many of the evaluations, and Ed Costello, Mary Neff, Bran Boguraev and Yael Ravin for valuable discussions. Finally, I would like to thank the editors and anonymous referees for their very helpful comments.

6. References

[1] Apte, C.; Damerau, F. and Weiss, S. Automated learning of decision rules for text categorization, *ACM Trans. Inf. Sys.*, 12, 3, (1994) 233-251.

[2] Baeza-Yates, R. and Ribeiro-Neto, B. *Modern Information Retrieval*. New York: ACM Press, 1999.

[3] Boguraev, B. and Kennedy, C. Salience-based content characterization of text documents. In *Advances in Automatic Text Summarization*, Mani, I. and Maybury. M. (eds.), Cambridge: MIT Press, 1999.

[4] Byrd, R.J. and Ravin, Y. Identifying and extracting relations in text. *Proceedings of 4th International Conference on Applications of Natural Language to Information Systems (NLDB 99)* Klagenfurt, Austria, (1999).

[5] Cavnar, W.B. and Trenkle., J.M. N-gram-based text categorization, *Symposium On Document Analysis and Information Retrieval*, University of Nevada, Las Vegas (1994), 161-176.

[6] Costello, E.P. Personal communication.

[7] Damashek, M. Gauging similarity with n-grams: language-independent categorization of text. *Science 267* (1995), 843-848.

[8] Duda, R.O. and Hart, P.E. *Pattern Classification and Scene Analysis*, New York: John Wiley & Sons, 1973.

[9] Grefenstette, G. Comparing two language identification schemes, http://www.rxrc.xerox.com/publis/mltt/jadt/jadt.html (1996)

[10] Grefenstette G. (ed.). *Cross-Language Information Retrieval*. Boston: Kluwer Academic Publishers, 1998.

[11] Harmon, D. How effective is suffixing? *Journal of the American Society for Information Science*, 42(1) (1991) 7-15.

[12] Harmon, D. Ranking algorithms. In *Information Retrieval: Data Structures and Algorithms*, Frakes, W. and Baeza-Yates, R. (eds.), Prentice-Hall, Upper Saddle River NJ, (1992) 363-392.

[13] Hearst, M.A. Automated discovery of WordNet relations, in *WordNet: an Electronic Lexical Database*, Fellbaum, C. (ed.), Cambridge: MIT Press, 1998.

[14] Hull, D.A. Stemming algorithms: a case study for detailed evaluation, *Journal of the American Society for Information Science*, 47(1) (1996) 70-84.

[15] IBM Intelligent Miner for Text. http://www.software.ibm.com/data/iminer/fortext (1997).

[16] Kupiec. J. MURAX: a robust linguistic approach for question answering using an on-line encyclopedia. In *Proceedings of the 16th International ACM SIGIR Conference on Research and Development in Information Retrieval*, Pittsburgh, PA, (1993) 181-190.

[17] Lawrence S. and Giles, C.L. Searching the World Wide Web, *Science 280* (1998) 98-100.

[18] Lewis, D.D.; Shapire, R.E.; Callan J.P. and Papka R. Training algorithms for linear text classifiers, *Proceedings of the 19th International ACM SIGIR Conference on Research and Development in Information Retrieval* (1996) 298-306.

[19] Ling C.X. and Wang, H. Computing optimal attribute weight settings for Nearest Neighbor algorithms, *Artificial Intelligence Review Vol. 11* (1997), 255-272.

[20] Maarek Y. and Smadja, F. Full text indexing based on lexical relations. an application: Software libraries. *Proceedings of the 17th International ACM SIGIR Conference on Research and Development in Information Retrieval* (1989), 198-206.

[21] MUC. Proceedings of the Seventh Message Understanding Conference (MUC-7). DARPA Software and Intelligent Systems Technology Office. 1998.

[22] Porter, M. F. An algorithm for suffix stripping, *Program*, 14(3) (1980), 130-137.

[23] Ravin, Y.; Wacholder, N. and Choi, M. Disambiguation of names in text , *Proceedings of the Fifth Conference on Applied Natural Language Processing*, Washington D.C. (1997) 202-208.

[24] Salton G. and McGill, M. *Introduction to Modern Information Retrieval*. New York: McGraw-Hill, 1983.

[25] TDT. Proceedings of DARPA Broadcast News Workshop (TDT-2), Washington, D.C., February-March 1999.

[26] Teufel, S. and Moens, M. Argumentative classification of extracted sentences as a first step towards flexible abstracting. In *Advances in Automatic Text Summarization*, I. Mani and M. Maybury (eds.), Cambridge: MIT Press, 1999.

[27] Yang, Y. Expert Network: effective and efficient learning from human decisions in text categorization and retrieval, *Proceedings of the 17th International ACM SIGIR Conference on Research and Development in Information Retrieval*, Dublin, Ireland (1994), 13-22.

[28] Yang, Y. An evaluation of statistical approaches to text categorization, *Carnegie Mellon University School of Computer Science Technical Report* CMU-CS-97-127, Pittsburgh, PA, 1997.

[29] Yang, Y. A comparative study on feature selection in text categorization, *Proc. 14th International Conference on Machine Learning*, Vanderbilt University, Nashville, TN, 1997.