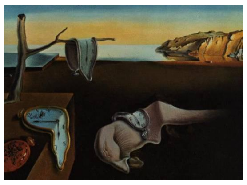

Journal of Statistical Software

January 2013, Volume 52, Issue 6. *http://www.jstatsoft.org/*

The textcat Package for n-Gram Based Text Categorization in R

Kurt Hornik
WU Wirtschafts-universität Wien

Patrick Mair
WU Wirtschafts-universität Wien

Johannes Rauch
WU Wirtschafts-universität Wien

Wilhelm Geiger
WU Wirtschafts-universität Wien

Christian Buchta
WU Wirtschafts-universität Wien

Ingo Feinerer
Technische Universität Wien

Abstract

Identifying the language used will typically be the first step in most natural language processing tasks. Among the wide variety of language identification methods discussed in the literature, the ones employing the Cavnar and Trenkle (1994) approach to text categorization based on character n-gram frequencies have been particularly successful. This paper presents the R extension package **textcat** for n-gram based text categorization which implements both the Cavnar and Trenkle approach as well as a reduced n-gram approach designed to remove redundancies of the original approach. A multi-lingual corpus obtained from the Wikipedia pages available on a selection of topics is used to illustrate the functionality of the package and the performance of the provided language identification methods.

Keywords: text mining, text categorization, language identification, n-grams, **textcat**, R.

1. Introduction

Working with written text usually requires knowledge about the language used. For example, typical text mining workflows remove stop words from texts or transform words into their stems, which clearly cannot be performed without knowing the underlying language. Therefore, modern text processing tools heavily rely on highly effective algorithms for language identification.

The first language identification methods developed were very simple: one had to know a language in order to recognize it. Humans read documents and classified only the ones they understood. To improve the process, methods were developed which enabled a reader to identify the language of a document without actually knowing the language. This was done using lists (see Ingle 1976; Keesan 1987; Newman 1987) with peculiarities of the candidate languages, such as unique letters or combinations thereof and unique words or combinations thereof. This approach, however, had many shortcomings. One would often have to work through a considerable amount of text before finding one of the unique characteristics. In addition, the method was quite vulnerable to the presence of foreign or misspelled words, or the use of highly specialized vocabulary. Thus, these methods performed rather poorly, both in terms of speed and precision.

Then came the age of information technology and with it the pioneers in the field of automated language identification, such as Mustonen (1965), Beesley (1988), Henrich (1989), and Souter, Churcher, Hayes, Hughes, and Johnson (1994). The main idea was to create distributions of specific "elements" for a number of languages and, subsequently, to compare these to the distribution of the same elements obtained from a given text. Over the years, the elements chosen to represent a certain language were altered and the methods to create and compare the distributions improved. The key contribution was certainly the Cavnar and Trenkle (1994) paper on "*n*-Gram-Based Text Categorization", which suggested character *n*-gram frequencies as elements and the so-called out-of-place measure for comparing these.

Language identification has continued to be a subject of high interest until nowadays and many further approaches have been suggested. Batchelder (1992) built a neural net, Dunning (1994) classified using a Markov Chain model, Sibun and Reynar (1996) and Singh (2006) worked with *mutual cross entropy*, and Ahmed, Cha, and Tappert (2004) used *cumulative frequency addition* to increase accuracy and efficiency. Murray (2002) employed hidden Markov models that allow for segmentation of text into unknown languages and the extraction of foreign words in known languages from English text. Combinations of different approaches were tested in, e.g., Ljubešić, Mikelić, and Boras (2007).

Clearly, all general-purpose methods for text classification can also be applied to the specific problem of language identification. In this paper, however, we will focus on *n*-gram based approaches as these have been amazingly successful. In Section 2, we present the original Cavnar and Trenkle (1994) approach and a reduced version designed to eliminate redundancies of the original approach. The implementation in the R (R Core Team 2012) extension package **textcat** (Hornik, Rauch, Buchta, and Feinerer 2013) is discussed in Section 3. In Section 4, we use the `Wikipedia_multi` multi-lingual corpus obtained from the Wikipedia pages available on a selection of topics to illustrate the functionality of the package and the performance of the provided language identification methods. Section 5 concludes.

2. Methodology

2.1. The Cavnar and Trenkle approach

An *n*-gram as defined by Cavnar and Trenkle (1994, p. 163) – hereafter CT – is a contiguous *n*-"character" slice of a longer word string. (In general, *n*-grams are subsequences of *n* items computed from a given sequence, e.g., Wikipedia (2013a). *n*-grams of lengths 1, 2 and 3

$n = 1$	$n = 2$	$n = 3$	$n = 4$	$n = 5$
`_`	`_c`	`_co`	`_cor`	`_corp`
`c`	`co`	`cor`	`corp`	`corpu`
`o`	`or`	`orp`	`orpu`	`orpus`
`r`	`rp`	`rpu`	`rpus`	`rpus_`
`p`	`pu`	`pus`	`pus_`	`pus__`
`u`	`us`	`us_`	`us__`	`us___`
`s`	`s_`	`s__`	`s___`	`s____`

Table 1: Classical n-gram representation of the word '`corpus`'.

are typically referred to as unigrams, bigrams and trigrams, respectively; for lengths $n \geq 4$, one simply uses the generic term "n-gram"). Depending on whether word strings are taken as sequences of characters or sequences of bytes (where the distinction matters if multi-byte character systems are used for encoding text), CT n-grams are thus *character n-grams* or *byte n-grams*, and to be distinguished from word n-grams (sequences of consecutive words) commonly employed in a variety of natural language processing tasks. In this section, we will follow common practice to refer to the word tokens as "characters".

Typically, underscores are used to identify whether an n-gram includes the first character (e.g, '`_fi`'), characters from the middle (e.g., '`dd`', no underscore here), or the end (e.g., '`nd_`') of a word, i.e., to indicate word boundaries. Cavnar and Trenkle (1994) use the word '`text`' to show how n-grams (n varying from 1 to 5) should be generated. In Table 1, this "classical" method is applied to the word '`corpus`'. Note that words of length k yield $k + 1$ n-grams. (A useful tool for visualizing n-grams is provided by the WolframAlpha computational knowledge engine and can be found at `http://www.wolframalpha.com/input/?i=n-gram` when selecting "n-gram" as *a general topic*.)

The CT n-gram based approach to language identification uses two steps. First, one collects training corpora of texts all written in the same known language, and builds *language profiles* from these. For every corpus, one computes the frequency distribution of the n-grams of the texts in the corpus for $n = 1, \ldots, n_{\max}$. Typically, to improve performance, only a maximal number of words is used from each text. One then sorts the n-grams from the most to the least frequent, and retains the s most frequent ones, for a prescribed profile size s. Cavnar and Trenkle (1994, p. 162) argue that n-gram distributions follow Zipf's law (Zipf 1949, "The nth most common word in a human language text occurs with a frequency inverse proportional to n.") and hence a rather small value of s can be taken, recommending taking $s = 300$. Note, however, that the quoted empirical law really refers to word and not character n-grams: we will take a closer look at frequency distributions of the latter in Section 4.1, using the `Wikipedia_multi` multilingual corpus.

In a second step, to identify the (unknown) language of a given text "document", one computes a *document profile* from this text in the same manner as previously having computed the language profiles, and classifies the text according to the language of the (language) profile which best fits the document profile, in the sense of minimizing a suitable distance measure for n-gram profiles. (Optionally, if the fit is not good enough, one classifies as "unknown".) Cavnar and Trenkle (1994) suggest the so-called "out-of-place" distance measure, which counts the number of rank inversions between the profiles (how to handle n-grams not present in both profiles is not precisely specified).

We note that the CT approach is applicable to arbitrary text classification tasks, employing document profiles and category profiles obtained from corpora of texts belonging to the same known category. See, e.g., Cavnar and Trenkle (1994, Section 5) for an illustration of n-gram based text categorization on a computer newsgroup classification task, and Khreisat (2009) for an application to identifying topics in Arabic newspaper articles. However, the superb performance of the approach is typically only achieved for language identification tasks.

2.2. A reduced n-gram approach

The basic CT approach can be customized via several "options", such as the size s of the profiles, the number of words in a text used for computing the profiles, or the distance measure employed when comparing profiles. Interestingly, whereas many authors have explored how the performance of the approach varies with the choices for these options (e.g., Grefenstette 1995; Ahmed *et al.* 2004; Singh 2006), it seems that the method used for computing n-grams was never questioned. Our initial language identification task was based on very short (SMS-style) texts, prompting us to explore the possibilities of deriving more efficient n-gram representations.

The following observations can be made for the "classical" CT method for computing n-grams.

- The unigram '`_`' indicating the beginning of a word is always included. If a fixed number of words is taken to generate n-grams, this will lead to precisely the same number of '`_`' for each document.
- Both n-grams with and without the extra information about their position (i.e., the leading and trailing underscores) are included. In our example in Table 1, one can find '`corp`' as well as '`_corp`'. Using the former would mean that the n-gram '`corp`' can be found in the middle of the word '`corpus`', which, of course, is not the case. It thus should be preferable to use n-grams which include the first or last character of a word only if the additional positional information is part of the n-gram.
- Including '`pus_`' as well as '`pus__`' is redundant. In fact, only the information that there is a '`pus`' at the end of the word '`corpus`' is of importance.
- Short words result in n-grams with omitted position information. E.g., the word '`is`' yields the trigrams '`_is`' and '`is_`' (and '`s__`'). Similar to the above, it might be preferable to drop these and only use the n-grams which include the position indicators (i.e., '`_i`', '`s_`', and '`_is_`').

Our new "reduced" method for computing n-grams thus adds the following set of rules to the classical method, in order to possibly improve performance.

- '`_`' is excluded.
- n-grams containing the first or last, respectively, character of a word, without the additional information about this position (i.e., without the underscores), are not used.
- n-grams containing more than one word boundary indicator at the end are excluded.
- Words with $k > 1$ characters only yield n-grams of lengths $n \leq k$ or $n = k + 2$ (the word plus both word boundary indicators).

$n = 1$	$n = 2$	$n = 3$	$n = 4$	$n = 5$
	`_c`	`_co`	`_cor`	`_corp`
`o`	`or`	`orp`	`orpu`	
`r`	`rp`	`rpu`		`rpus_`
`p`	`pu`		`pus_`	
`u`		`us_`		
	`s_`			

Table 2: Reduced n-gram representation of the word '`corpus`'.

- Words with a single character c only yield the trigram '`_`c`_`'.

The "reduced" n-gram representation of '`corpus`' obtained by applying these rules is shown in Table 2.

These rules are aimed at improving the "information" quantity and quality within the language profiles. As already mentioned, only the 300 most frequent n-grams were used by Cavnar and Trenkle (1994) to form a language profile. In this case, if a language consists of many words ending with, e.g., '`s`', then with $n_{\max} = 5$, 4 out of 300 places within the language profile will be used to record this same piece of information (by including '`s_`', '`s__`', '`s___`' and '`s____`'), so that 1% (3 out of 300) places are wasted. Similarly, excluding n-grams containing the first and/or last character of a word without the additional word boundary indicators avoids redundancies and improves consistency of the representation employed.

3. Implementation

The first implementation of the Cavnar & Trenkle approach which was publicly made available is Gertjan van Noord's Perl-based **TextCat** (van Noord 1997), which also provides language profiles for 74 "languages" (more precisely, language/encoding combinations). The code was subsequently integrated into the **SpamAssassin** spam filter software (Apache 2010), **TextCat** itself is no longer actively maintained. Van Noord provides a web page listing "competitors" (`http://odur.let.rug.nl/~vannoord/TextCat/competitors.html`), pointing in particular to **TextCat**-style implementations in Java and Python as well as **libTextCat** (WiseGuys 2003), a lightweight C library re-implementation, which is included in most Linux distributions (e.g., `libtextcat0` on Debian-based systems).

The implementation in the R extension package **textcat** aims at both flexibility and convenience. An n-gram profile really is a frequency table, so that it seems natural to represent the profile as a numeric vector of frequencies named by the corresponding n-grams. As names in R should really be "valid" character strings, when using character n-grams texts containing non-ASCII characters must declare their encoding, and will be re-encoded to UTF-8. For byte n-grams, we take advantage of the 'bytes' encoding for character strings added in R 2.13.0, motivated by our work on **textcat**. This new encoding allows representing a sequence of bytes as a single character string, rather than a sequence of individual *raw* bytes (which would result in a substantially more complex representation of byte n-gram profiles). Where necessary, functions `readBytes()` and `readChars()` from package **tau** (Buchta, Hornik, Feinerer, and Meyer 2012) can be used to read texts in files into byte strings and UTF-8 encoded character strings, respectively.

The basic data structure in package **textcat** is the S3 class `"textcat_profile_db"` for categorized collections of n-gram profiles, implemented as lists of frequency tables as discussed above, with the category IDs as names and the options employed for creating the profile data base (DB) as attributes. Provided that they use the same options, such profile DBs can be combined via `c()`.

Profile DBs can be created using function `textcat_profile_db()`, with synopsis

```
  textcat_profile_db(x, id = NULL, ...)
```

where `id` gives the category IDs (suitable language IDs in the case of language identification) and `x` the corresponding texts, either directly as character vectors or as R objects from which texts can be extracted using `as.character()`, such as text corpora obtained via function `Corpus()` in package **tm** (Feinerer, Hornik, and Meyer 2008). The further '`...`' arguments allow specifying the options to employ for creating the n-gram profiles, including:

`n`: A vector containing the numbers of characters or bytes in the n-gram profiles (default: `1:5`).

`size`: The maximal number of n-grams used for a profile (default: `1000L`).

`reduce`: A logical indicating whether reduced n-gram representations as discussed in Section 2.2 should be employed (default: `TRUE`).

`useBytes`: A logical indicating whether to use byte n-grams rather than character n-grams (default: `FALSE`).

In `textcat_profile_db()`, texts are split according to the given categories (the default corresponds to taking each text separately), and n-gram profiles are computed via efficient C code for counting n-gram frequencies provided by function `textcnt()` in package **tau** (Buchta *et al.* 2012).

Categorization is performed by function `textcat()`, with synopsis

```
  textcat(x, p = TC_char_profiles, method = "CT")
```

where `x` is a character vector of texts (or coercible to such using `as.character()`), `p` is a profile DB, and `method` is a character string specifying a built-in method, or a user-defined function for computing distances between n-gram profiles. By default, categorization uses the **TextCat** character profiles and the Cavner-Trenkle out-of-place measure. To provide a simple example:

```
R> library("textcat")
R> textcat(c(
+    "This is an English sentence.",
+    "Das ist ein deutscher Satz.",
+    "Esta es una frase en español."))

[1] "english" "german"  "spanish"
```

As we see, all three sentences are classified correctly.

The `TC_char_profiles` DB provides a subset of 56 character profiles obtained by converting the **TextCat** byte profiles to UTF-8 strings where possible. (Actually, the byte profiles are taken from **libTextCat** rather than **TextCat**, which contains one additional non-empty profile). The full set of byte profiles is available as `TC_byte_profiles`. Both profiles use a size of 400 and the classical method for computing n-grams.

Alternatively, **textcat** provides the `ECIMCI_profiles` DB for 26 mostly European languages built by one of us (JR) from the European Corpus Initiative Multilingual Corpus I (Armstrong-Warwick, Thompson, McKelvie, and Petitpierre 1994), using a size of 1000 and reduced n-grams. Traditionally, high-quality low-cost large-scale multilingual text corpora were rather scarce, with ECI's MC I a major step forward. In Section 4, we will show how nowadays Wikipedia can very conveniently be used for building domain specific multilingual corpora.

We are planning to make additional textcat profile data packages available at `http://datacube.wu.ac.at/` (currently, this provides the character trigram profiles from the "An Crúbadán" project, Scannell 2007).

Pairwise distances between collections of n-gram profiles or text documents can be computed via `textcat_xdist()`. Currently, the following distance methods for n-gram profiles are available and can be specified through the `method` argument to `textcat_xdist()` (and `textcat()`):

`"CT"`: The out-of-place measure of Cavnar and Trenkle (1994) (default).

`"ranks"`: A variant of the Cavnar-Trenkle measure based on the aggregated absolute difference of the ranks of the combined n-grams in the two profiles.

`"ALPD"`: The sum of the absolute differences in n-gram log frequencies.

`"KLI"`: The Kullback-Leibler I-divergence $I(p,q) = \sum_i p_i \log(p_i/q_i)$ of the n-gram frequency distributions p and q of the two profiles.

`"KLJ"`: The Kullback-Leibler J-divergence $J(p,q) = \sum_i (p_i - q_i)\log(p_i/q_i)$, the symmetrized variant $I(p,q) + I(q,p)$ of the I-divergences.

`"JS"`: The Jensen-Shannon divergence between the n-gram frequency distributions as a symmetrized and smoothed version of the I-divergence.

`"cosine"`: The cosine dissimilarity between the profiles, i.e., one minus the inner product of the frequency vectors normalized to Euclidean length one (and filled with zeros for entries missing in one of the vectors).

`"Dice"`: The Dice dissimilarity, i.e., the fraction of n-grams present in one of the profiles only.

For the measures based on distances of frequency distributions, n-grams of the two profiles are combined, and missing n-grams are given a small positive absolute frequency which can be controlled by option `eps`, and defaults to `1e-6`).

The options used for building n-gram profiles ('`...`' arguments to `textcat_profile_db()`) and categorization based on these ('`method`' argument to `textcat()`) can also be manipulated as dynamic variables via `textcat_options()`.

4. Applications

In order to study the performance of the classical and the reduced n-gram approach we scraped Wikipedia entries for Philosophy, Mathematics, Statistics, France, USA, Religion, Wikipedia, Internet, Medicine, and Rice in all available languages. Technically, we start with the English pages, use the MediaWiki API (`action=query&prop=langlinks`) to get the language links of these, and XPath (e.g., Wikipedia 2013c) to extract the page "texts" as their content inside `<p>` tags. As of 2010-10-17, this leads to a collection of 1641 text documents in 254 different languages. One should note how easily the same method can be used to build large-scale, possibly domain-specific multi-lingual corpora.

Our Wikipedia multilingual corpus can be installed and loaded as follows:

```
R> install.packages("tm.corpus.Wikipedia.multi",
+    repos = "http://datacube.wu.ac.at", type = "source")
R> library("tm.corpus.Wikipedia.multi")
R> data("Wikipedia_multi", package = "tm.corpus.Wikipedia.multi")
```

The following lines extract the language information and the texts.

```
R> langs <- meta(Wikipedia_multi, "Language", type = "local")
R> langs <- unlist(langs)
R> texts <- lapply(Wikipedia_multi, paste, collapse = "\n")
R> texts <- unlist(texts)
```

The languages and the texts are now in a structure suited for further analyses. First, we start with an examination of the Zipf approximation for several languages. Second, we carry out simulation experiments where we study the classification performance of the classical and the reduced n-gram approaches for various scenarios (i.e., different numbers of words and different languages).

4.1. Examining the n-gram distributions

As mentioned above, Cavnar and Trenkle (1994) imply that character n-gram distributions follow Zipf's law. Recent works on word n-grams point out corresponding systematic deviations. Baayen (2008, p. 226) elaborates the problem of sample independence of Zipf's law. In fact, Ha, Hanna, Ming, and Smith (2009) propose an extension of Zipf's law for large corpora. Egghe (2000) shows that the rank-frequency distribution follows Zipf's law with an additional exponent.

In order to visualize the Zipf approximation for different languages, out of the 254 languages we pick the ones with the highest numbers of n-grams. These nine languages with corresponding language ID in parentheses are German (de), English (en), French (fr), Italian (it), Spanish (es), Russian (ru), Portuguese (pt), Catalan (ca), and Tamil (ta). We create the n-gram profiles for these languages using the `textcat_profile_db()` function (`size = NA` indicates to include all n-grams in the profiles).

```
R> ind <- langs %in% c("de", "en", "fr", "it", "es", "ru", "pt", "ca", "ta")
R> profiles <- textcat_profile_db(texts[ind], langs[ind], size = NA)
```

	Densities			Cumulative Frequencies		
Language	300	400	1000	300	400	1000
Catalan	4.26e-04	3.13e-04	1.22e-04	0.559	0.595	0.708
German	3.77e-04	2.91e-04	1.18e-04	0.534	0.567	0.676
English	3.92e-04	2.93e-04	1.24e-04	0.561	0.594	0.708
Spanish	4.13e-04	3.09e-04	1.21e-04	0.566	0.600	0.712
French	4.21e-04	3.20e-04	1.24e-04	0.561	0.597	0.711
Italian	4.62e-04	3.29e-04	1.24e-04	0.573	0.611	0.727
Portuguese	4.22e-04	3.07e-04	1.24e-04	0.550	0.586	0.699
Russian	4.00e-04	3.06e-04	1.32e-04	0.482	0.516	0.634
Tamil	4.17e-04	3.27e-04	1.31e-04	0.535	0.572	0.692

Table 3: Densities and cumulative frequencies of the s-most frequent n-gram, for sizes s of 300, 400 and 1000.

For each language we create a Zipf plot with the logarithm of the n-gram ranks on the x-axis and the log-frequencies of the n-grams on the y-axis. A regression line is added that reflects the expected trajectory under the Zipf distribution.

The results in Figure 1 show rather marked deviations from the Zipf distribution, indicating that unlike for word n-grams, Zipf's law does not hold for character n-grams, which seem to yield frequency distributions with heavier tails. Table 3 shows the densities and cumulative frequencies of the s-most frequent n-grams, for sizes s of 300 (the Cavnar-Trenkle suggestion), 400 (used for the **TextCat** profiles), and 1000 (the default profile size used by **textcat**). Using $s = 1000$ substantially increases coverage, suggesting that employing larger profile sizes than originally suggested might be more appropriate.

Character and byte n-gram distributions for many texts (e.g., obtained from Project Gutenberg, http://www.gutenberg.org/) typically look "rather similar" to the ones displayed in Figure 1. It should be both interesting and useful to find simple parametric families for representing such distributions.

4.2. Simulation study: Classical versus reduced n-gram approach

In order to study the behavior of both n-gram approaches for different languages and different number of words considered for n-gram generation, we carry out an extensive simulation experiment. First, let us select the languages that have Wikipedia entries for each of the 10 search terms mentioned at the beginning of Section 4. Then we extract the corresponding texts.

```
R> tab <- table(langs)
R> languages <- names(tab)[tab == 10]
R> languages <- languages[languages != "simple"]
R> ind <- langs %in% languages
R> langs <- langs[ind]
R> texts <- texts[ind]
```

Note that we eliminate texts from "Simple English Wikipedia", a Wikipedia platform for people whose first language is not English (including children and adults who are learning

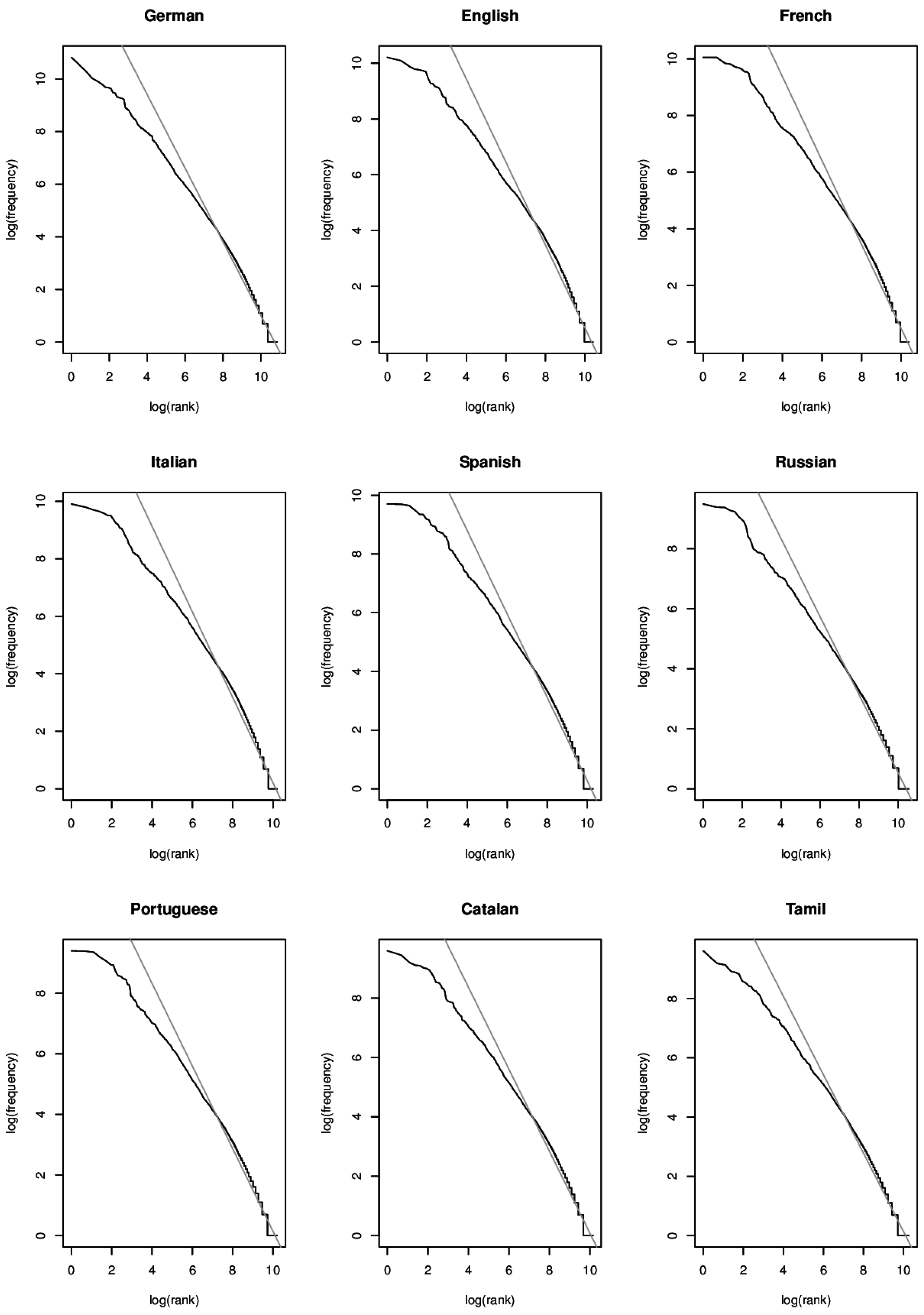

Figure 1: Zipf plots of character n-gram frequencies for nine different languages.

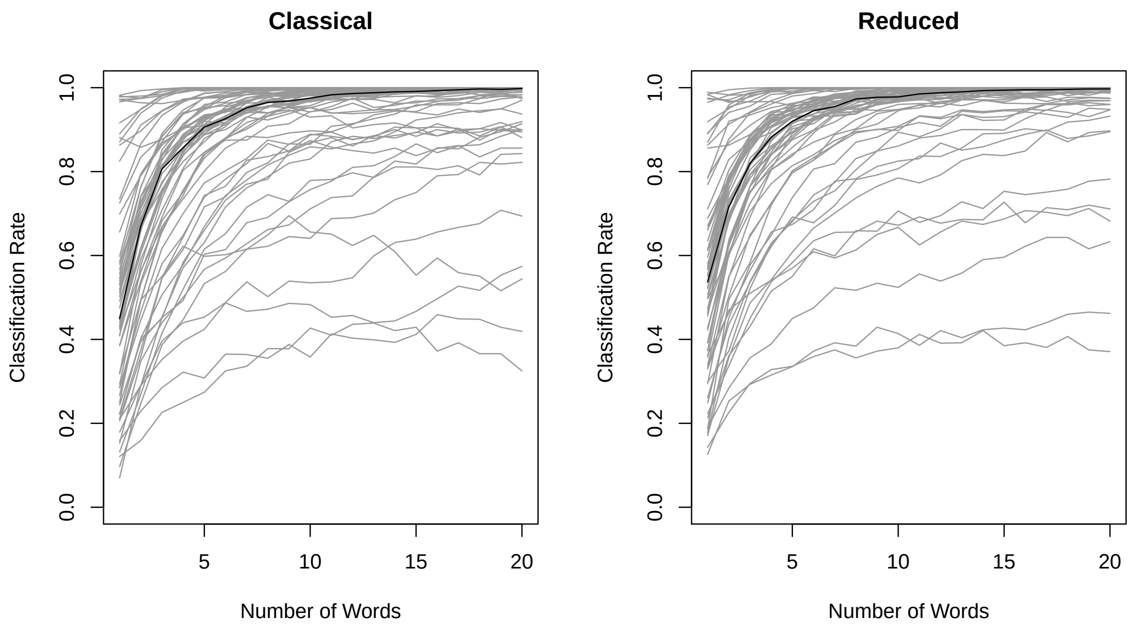

Figure 2: Trajectories plots for classical and reduced n-gram approach.

English), see http://simple.wikipedia.org/. Not excluding Simple English in our simulation study would drastically affect the misclassification rate for English texts. After these pre-selections, 63 languages are left for our simulation experiment.

Now we create the n-gram profiles for the reduced approach (JR) and the classical Cavnar-Trenkle approach (CT). According to the suggestions by Cavnar and Trenkle (1994), we set the maximal number of n-grams to 300.

```
R> TC_Wiki_profiles_db_a_la_JR <- textcat_profile_db(texts, langs,
+    reduce = TRUE, size = 300)
R> TC_Wiki_profiles_db_a_la_CT <- textcat_profile_db(texts, langs,
+    reduce = FALSE, size = 300)
```

For each language we build a pool of the words used. Then, for every w from 1 to 20, we generate 1000 texts by randomly drawing w words from the word pool, and perform text categorization using `textcat()`. Finally, for each language/number-of-words-scenario we count the number of correctly classified texts. The R code for reproducing the whole simulation experiment as well as our simulation results, stored in the R data file loaded below, are given in the supplementary materials.

```
R> load("simTCWiki.rda")
```

As a first tool to explore the results we create trajectories plots. The panels in Figure 2 show the classification trajectories for both approaches. A single trajectory refers to a particular language. The black trajectory displays the median of the classification rates. Those languages that are classified badly are examined below in more detail. For the moment let us focus on

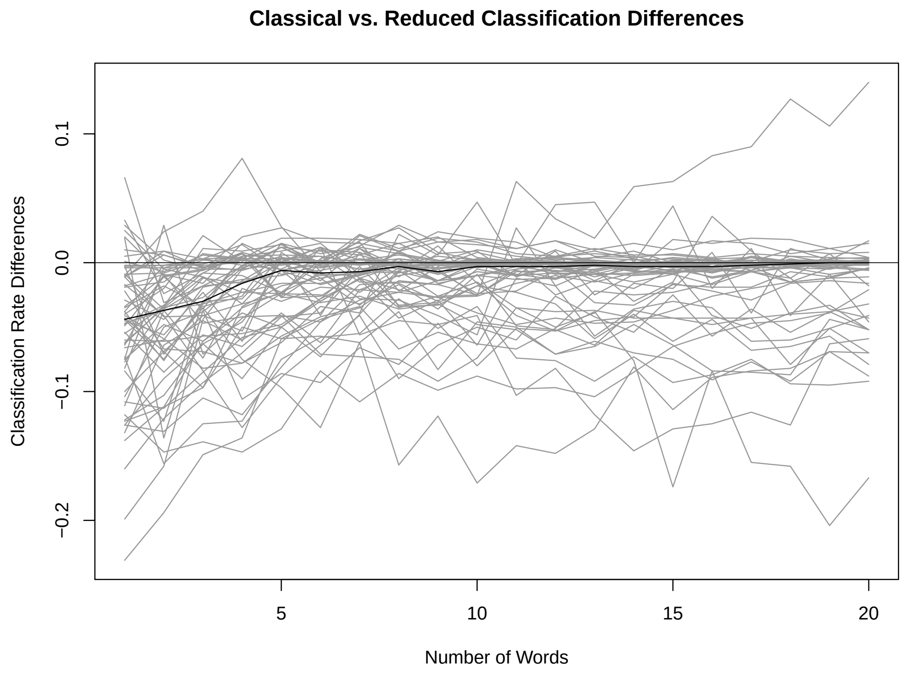

Figure 3: Trajectories plots for classification differences between classical and reduced n-gram approach.

the performance differences between the two approaches. To do so, we create the same type of plot, except that this time we put the differences in the classification rates between the classical and the reduced approach on the y-axis.

In Figure 3, trajectories below the zero line indicate the cases for which the reduced n-gram approach performs better than the classical approach. Especially for short texts (i.e., numbers of words smaller than five) we see that the reduced approach outperforms the classical one.

Let us have a closer look at languages that are classified badly. A table with the "worst" 10 languages (in terms of classification rates using texts of $w = 20$ words) for the reduced n-gram approach is given in Table 4. These languages are: Bosnian (bs; 72.47% correctly classified), Danish (da; 87.55%), Spanish (es; 95.04%), Galician (gl; 92.81%), Croatian (hr; 57.55%), Indonesian (id; 79.33%), Malay (ms; 78.51%), New Norwegian (nn; 86.74%), Norwegian (no; 76.02%), and Serbo-Croatian (sh; 39.58%).

The percentage values in Table 4 are conditional on the row margins. For Bosnian (br) we see that in 14.63% percent of the cases it is classified as Croatian (hr) and in 12.16% as Serbo-Croatian (sh). This is not surprising since Bosnian is one of the three "Serbo-Croatian" standards, with the latter term resulting from the time before the dissolution of former SFR Yugoslavia (see also Wikipedia 2013b, for how Wikipedia handles this rather delicate matter). In fact, Standard Croatian, Serbian and Bosnian are almost completely mutually intelligible. This also explains the bad classification rate for Croatian which in 23.66% is "misclassified" as Serbo-Croatian. Note that Croatian is not confounded with Serbian since Serbian uses the Cyrillic script.

	bs	hr	sh	es	gl	id	ms	da	nn	no
bs	72.469	14.635	12.162	0.043	0.054	0.070	0.091	0.059	0.337	0.080
hr	18.039	57.546	23.663	0.070	0.081	0.103	0.070	0.038	0.281	0.108
sh	23.018	36.562	39.584	0.054	0.071	0.087	0.065	0.049	0.408	0.103
es	0.079	0.011	0.006	95.043	4.549	0.062	0.028	0.091	0.062	0.068
gl	0.035	0.006	0.018	6.967	92.805	0.059	0.035	0.018	0.035	0.023
id	0.047	0.021	0.063	0.047	0.042	79.332	20.048	0.089	0.136	0.173
ms	0.052	0.036	0.047	0.042	0.036	21.075	78.509	0.021	0.088	0.094
da	0.047	0.047	0.016	0.073	0.058	0.047	0.031	87.545	4.641	7.495
nn	0.079	0.026	0.031	0.042	0.052	0.079	0.016	3.356	86.743	9.576
no	0.047	0.016	0.016	0.057	0.063	0.052	0.052	12.042	11.634	76.021

Table 4: Confusion matrix (in %) for the "worst" languages. The rows represent the correct languages, the columns represent the classified languages.

The number of misclassifications between Spanish and Galician, spoken in Galicia, an autonomous community located in northwestern Spain, are rather low. Note that Galician is actually more similar to the Portuguese language than to Spanish.

Considering Scandinavian languages, the situation is quite interesting. There are two official forms of written Norwegian: One is Bokmål (Book Norwegian; no), the other one Nynorsk (New Norwegian; nn). The misclassification rates between these two languages are around 10% (in both directions). Furthermore, in 12.05% of the cases Book Norwegian texts are classified as Danish, whereas in 7.50% Danish texts are classified as Norwegian. This is not surprising since together with Swedish and Danish, Norwegian forms the family of Scandinavian languages which are more or less mutually intelligible.

Finally, with respect to language similarities addressed above, we visualize the n-gram distances between various languages by means of hierarchical clustering. The resulting dendrogram is given in Figure 4. On the one hand it substantiates the misclassifications from Table 4, on the other hand, interesting similarities between languages are visualized; not necessarily related to misclassifications.

Croatian, Serbo-Croatian and Bosnian are merged into a cluster at a very early stage. Subsequently, they are merged with Czech, Slowak, and Slowenian such that this cluster represents the group of Slavic languages (Latin script). The other cluster of Slavic languages (Cyrillic script), placed on the left hand side of the dendrogram, is formed by Russian, Bulgarian, Serbian, and Ukrainian.

Malay and Indonesian on the one hand, and Norwegian, Nynorsk, and Danish (and Swedish) on the other hand, are also clustered at an early stage. The latter languages belong to the family of Germanic languages and are subsequently clustered with German and Dutch. In the middle of the dendrogram we have the cluster with Romance (or Latin) languages such as Spanish, Portuguese, French, Italian, Galician, Romanian, and Catalan.

5. Conclusion

The n-gram based approach to text categorization introduced in Cavnar and Trenkle (1994) provides a popular, high-performance methodology for language identification. We discuss

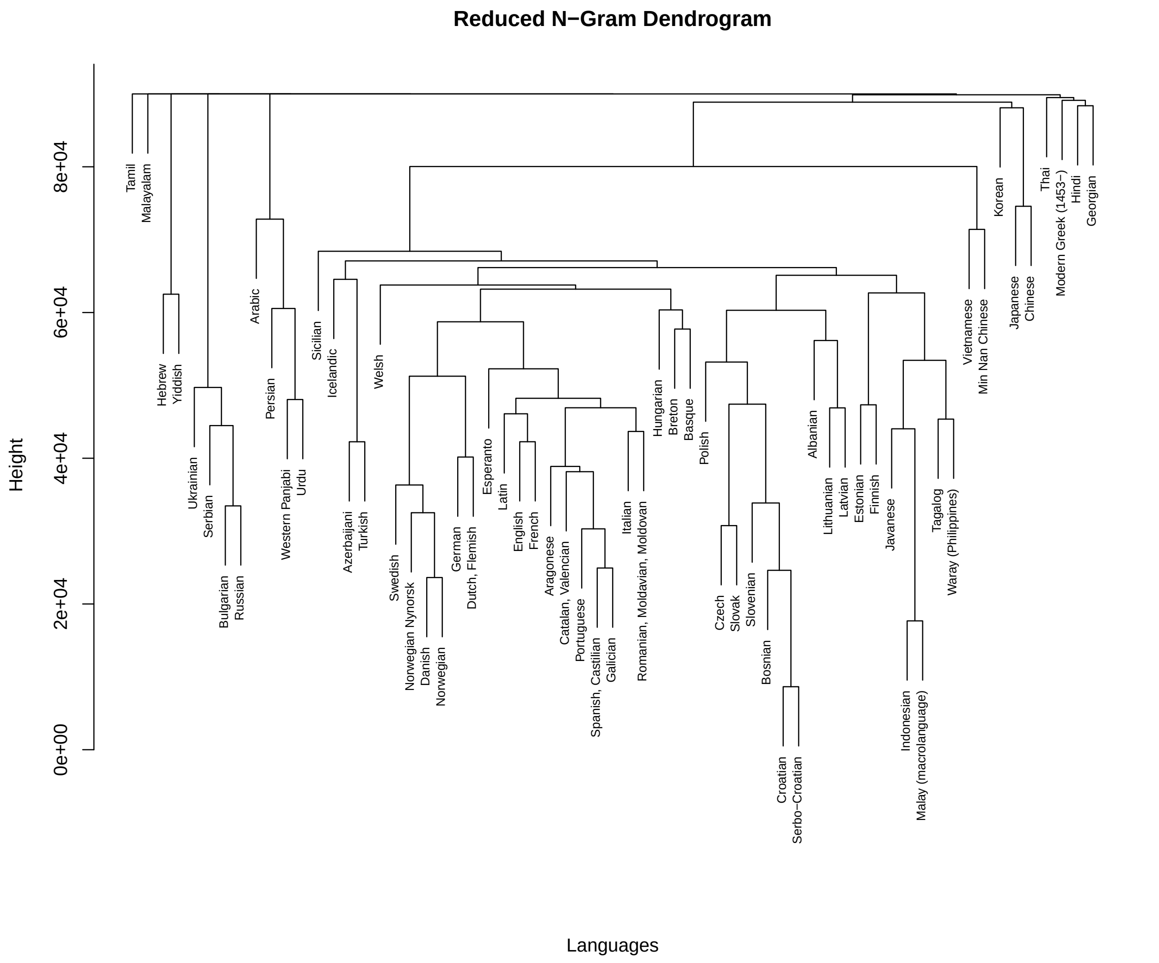

Figure 4: Dendrogram for n-gram distances between languages. Labels are obtained by expanding the Wikipedia language IDs according to the IANA Language Subtag Registry, using `parse_IETF_language_tag()` in package **tau**.

a possible enhancement based on "reduced" n-gram representations and show that this can outperform the standard approach in small data situations. Note that in our simulation study we used a profile size of 300. Increasing this size to, for instance, $s = 1000$, differences in terms of classification rates between the two n-gram approaches are getting less marked.

We present the R extension package **textcat** which provides an easy-to-use and flexible implementation of the CT approach. The package provides an important addition to R's facilities for natural language processing, in particular by providing the infrastructure for general statistical analyses of frequency distributions of (character or byte) n-grams. This should allow for more systematically investigating the effects of truncating these distributions to the most frequent n-grams, and how to appropriately measure dissimilarity between such n-gram profiles.

For many languages, several encodings are widely employed (e.g., for most Western European languages, UTF-8 and Latin1/CP1252 and possibly even transliteration to ASCII), and encodings are not necessarily known in applications. Thus, developing collections of byte profiles with more language/encoding combinations than currently provided in the **TextCat** byte profile data base would certainly be very useful. Hopefully, such collections can be made available in the future.

References

Ahmed B, Cha SH, Tappert C (2004). "Language Identification from Text Using n-Gram Based Cumulative Frequency Addition." In *Proceedings of Student/Faculty Research Day, CSIS, Pace University, May 7th, 2004*. URL http://www.csis.pace.edu/~ctappert/srd2004/paper12.pdf.

Apache (2010). "The Apache **SpamAssassin** Project." URL http://spamassassin.apache.org/.

Armstrong-Warwick S, Thompson HS, McKelvie D, Petitpierre D (1994). "Data in Your Language: The ECI Multilingual Corpus 1." In *Proceedings of the International Workshop on Sharable Natural Language Resources*, pp. 97–106. Nara, Japan. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.44.950.

Baayen RH (2008). *Analyzing Linguistic Data: A Practical Introduction to Statistics Using R*. Cambridge University Press, Cambridge. URL http://www.ualberta.ca/~baayen/publications/baayenCUPstats.pdf.

Batchelder EO (1992). "A Learning Experience: Training an Artificial Neural Network to Discriminate Languages." *Technical report*, City University of New York.

Beesley KR (1988). "Language Identifier: A Computer Program for Automatic Natural-Language Identification of On-Line Text." In *Languages at Crossroads: Proceedings of the 29th Annual Conference of the American Translators Association*, pp. 47–54. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.22.9868.

Buchta C, Hornik K, Feinerer I, Meyer D (2012). ***tau**: Text Analysis Utilities.* R package version 0.0-15, URL http://CRAN.R-project.org/package=tau.

Cavnar WB, Trenkle JM (1994). "n-Gram-Based Text Categorization." In *Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval*, pp. 161–175. Las Vegas, US. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.53.9367.

Dunning T (1994). "Statistical Identification of Language." *Technical Report MCCS 94-273*, Computing Research Lab (CRL), New Mexico State University. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.48.1958.

Egghe L (2000). "The Distribution of N-Grams." *Scientometrics*, **47**(2), 237–252.

Feinerer I, Hornik K, Meyer D (2008). "Text Mining Infrastructure in R." *Journal of Statistical Software*, **25**(5), 1–54. URL http://www.jstatsoft.org/v25/i05/.

Grefenstette G (1995). "Comparing Two Language Identification Schemes." In *Proceedings of the 3rd International Conference on Statistical Analysis of Textual Data (JADT 1995)*, volume II, pp. 263–268. Rome. URL http://www.xrce.xerox.com/content/download/20524/147563/file/Gref---Comparing-two-language-identification-schemes.pdf.

Ha LQ, Hanna P, Ming J, Smith FJ (2009). "Extending Zipf's Law to *n*-Grams for Large Corpora." *Artificial Intelligence Review*, **32**(1–4), 101–113.

Henrich P (1989). "Language Identification for the Automatic Grapheme-to-Phoneme Conversion of Foreign Words in a German Text-to-Speech System." In *EUROSPEECH-1989 (First European Conference on Speech Communication and Technology)*, pp. 2220–2223. URL http://www.isca-speech.org/archive/eurospeech_1989/e89_2220.html.

Hornik K, Rauch J, Buchta C, Feinerer I (2013). ***textcat***: *n-Gram Based Text Categorization.* R package version 1.0-0, URL http://CRAN.R-project.org/package=textcat.

Ingle NC (1976). "A Language Identification Table." *The Incorporated Linguist*, **15**(4), 98–101.

Keesan C (1987). "Identification of Written Slavic Languages." In K Kummer (ed.), *Proceedings of the 28th Annual Conference of the American Translators Association*, pp. 517–528.

Khreisat L (2009). "A Machine Learning Approach for Arabic Text Classification Using *n*-Gram Frequency Statistics." *Journal of Informetrics*, **3**(1), 72–77. doi:10.1016/j.joi.2008.11.005.

Ljubešić N, Mikelić N, Boras D (2007). "Language Identification: How to Distinguish Similar Languages." In V Lužar-Stifter, VH Dobrić (eds.), *Proceedings of the 29th International Conference on Information Technology Interfaces*, pp. 541–546. SRCE University Computing Centre, Zagreb. URL http://www.nljubesic.net/main/publications_files/ljubesic07-language.pdf.

Murray IA (2002). "Probabilistic Language Modelling." *Technical report*, Cavendish Laboratory, Cambridge, The Inference Group. URL http://www.inference.phy.cam.ac.uk/is/papers/langreport.pdf.

Mustonen S (1965). "Multiple Discriminant Analysis in Linguistic Problems." *Statistical Methods in Linguistics*, **4**, 37–44. Stockholm.

Newman P (1987). "Foreign Language Identification: First Step in the Translation Process." In K Kummer (ed.), *Proceedings of the 28th Annual Conference of the American Translators Association*, pp. 509–516.

R Core Team (2012). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL http://www.R-project.org/.

Scannell KP (2007). "The Crúbadán Project: Corpus building for under-resourced languages." In C Fairon, H Naets, A Kilgarriff, GM de Schryver (eds.), *Building and Exploring Web Corpora: Proceedings of the 3rd Web as Corpus Workshop*, volume 4 of *Cahiers du Cental*, pp. 5–15. Presses universitaires de Louvain, Louvain-la-Neuve, Belgium. URL http://borel.slu.edu/pub/wac3.pdf.

Sibun P, Reynar JC (1996). "Language Identification: Examining the Issues." In *5th Symposium on Document Analysis and Information Retrieval*, pp. 125–135. Las Vegas, Nevada, U.S.A. URL `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.52.4524`.

Singh AK (2006). "Study of Some Distance Measures for Language and Encoding Identification." In *Proceedings in the Workshop on Linguistic Distances, Sydney, July 2006*, pp. 63–72. URL `http://acl.ldc.upenn.edu/W/W06/W06-1109.pdf`.

Souter C, Churcher G, Hayes J, Hughes J, Johnson S (1994). "Natural Language Identification Using Corpus-Based Models." *Hermes Journal of Linguistics*, **13**, 183–203. URL `http://download2.hermes.asb.dk/archive/FreeH/H13_15.pdf`.

van Noord G (1997). "**TextCat**." URL `http://odur.let.rug.nl/~vannoord/TextCat`.

Wikipedia (2013a). "n-Gram – Wikipedia, The Free Encyclopedia." URL `http://en.wikipedia.org/wiki/N-gram`, accessed 2013-01-15.

Wikipedia (2013b). "Serbo-Croatian – Wikipedia, The Free Encyclopedia." URL `http://en.wikipedia.org/wiki/Serbo-Croatian`, accessed 2013-01-15.

Wikipedia (2013c). "XPath – Wikipedia, The Free Encyclopedia." URL `http://en.wikipedia.org/wiki/XPath`, accessed 2013-01-15.

WiseGuys (2003). "**libTextCat**: Lightweight Text Categorization." URL `http://software.wise-guys.nl/libtextcat/`.

Zipf GK (1949). *Human Behavior and the Principle of Least Effort: An Introduction to Human Ecology*. Addison-Wesley, Reading, MA.

Affiliation:

Kurt Hornik
Institute for Statistics and Mathematics
Department of Finance, Accounting and Statistics
WU Wirtschaftsuniversität Wien
Augasse 2–6
1090 Wien, Austria
E-mail: `Kurt.Hornik@wu.ac.at`
URL: `http://statmath.wu.ac.at/~hornik/`

Journal of Statistical Software `http://www.jstatsoft.org/`
published by the American Statistical Association `http://www.amstat.org/`
Volume 52, Issue 6 *Submitted:* 2011-07-11
January 2013 *Accepted:* 2012-10-01