Language Identification of Written Text: The R Package **langid**

Johannes Rauch

Abstract

This vignette gives a short overview over available features in the **langid** package for *n*-gram-based language detection purposes in R.

Loading the package

First, we need to load the package:

```
> options(width = 42)
> library("langid")
```

Since **langid** heavily relies on **tm** (Feinerer, 2008), this package must also be installed.

Data Import

Two different types of data are required for running **langid**: text documents and language-profiles. All text documents will have to be made available in the form of a *corpus*—a collection of text documents generated using **tm**. In the following examples, we will generate a *corpus* from three english .txt-files in the two most typical ways[1]:

First, we consider the case, where all .txt-files are saved in one file:

```
> options(width = 42)
> tdAdr <- system.file("traindata",
+     "tdeng", package = "langid")
> (icorpus <- Corpus(DirSource(tdAdr),
+     readerControl = list(reader = readPlain)))

A text document collection with 3 text documents
```

Next, we want to create a corpus from elements of a vector:

```
> (icorpus2 <- Corpus(VectorSource(letters)))
```

[1] For detailed information on how to generate a *corpus* see Feinerer et al. (2008)

Table 1: n-grams of the word ”‘manual”’

n=1	n=2	n=3	n=4	n=5
_	_m	_ma	_man	_manu
m	ma	man	manu	manua
a	an	anu	anua	anual
n	nu	nua	nual	nual_
u	ua	ual	ual_	ual_ _
a	al	al _	al_ _	al_ _ _
l	l_	l_ _	l_ _ _	l_ _ _ _

```
A text document collection with 26 text documents
```

All text which is to be used either to train language models or for classification, must be encoded in UTF-8. If this is not guarateed, the algorithm will inevitably produce errors.

The second type of data used is a matrix with language-profiles. Usually such a matrix must first be generated. Once this tedious process is completed, the profiles should be saved and thereafter loaded, whenever needed.

Generating Language-Profiles

To identify a language, first a document-profile must be generated. It depicts certain criteria of the text which is to be classified. In a second step, this document-profile is compared to different language-profiles. The best fit is understood to identify the language. The algorithm in **langid** takes *n-grams* as the criteria. n-grams are substrings of a longer string with length n. For example, take the word ‘manual’. In a first step, the word is padded with blanks[2] to become ‘_ manual _ _ _ _’. Then the n-grams are generated, n varying from 1 to 5. Of all the possible n-grams, only certain ones will be used further on. Table 1 shows the result—the n-grams in grey fields will be deleted.

The result can be obtained through:

```
> (generate_ngrams(x = "corpus", splits = 5))
```

```
 [1] "o"      "r"      "p"      "u"
 [5] " c"     "or"     "rp"     "pu"
 [9] "s "     " co"    "orp"    "rpu"
[13] "us "    " cor"   "orpu"   "pus "
[17] " corp" "orpus" "rpus "
```

Where x is the word from which the n-grams are to be generated and *splits* is the upper boundary of the range of n. Once, n-grams have been generated from many words, a frequency table can be used to obtain a n-gram-distribution of the text. This reveals, which n-grams occur often and which

[2] Represented trough underscores.

ones appear only rarely. Taking into consideration, that each language has a very characteristic n-gram-profile, the distribution of the n-grams from the text can be compared with those of different languages to measure, which fits best.

To train a language-model, a certain amount of training data will have to be supplied. Composing it can be a very time consuming and tedious process, which has to be conducted with greatest care, as the quality of the training data will highly influence the quality of the results[3]. As an example, we use training data that consits of three pages of englisch text[4] which allready has been read in as *icorpus* (initial corpus) to generate the english-language-profile:

```
> profX <- make_profX(icorpus, div = 0.5,
+     profX = "new", profXlen = 1000,
+     splits = 5, langcode = "eng")
```

The arguments stand for:

icorpus the name of the corpus with the trainingdata for one language, generated using **tm**.

div specifies how many percent of the most frequent n-grams generated from each of the documents within the corpus, should be included in the language-profile. The reason for this is, that an extremely high number of n-grams (up to 50%) occure only once or twice. So they are not significant and hence can be deleted.

profX if a new language profile should be created *profX* should be set to *"new"*. If an existing profile language matrix should be augmented by a new language, *profX* should be given the name of this existing *profX*.

profXlen only the top *profXlen* n-grams of all the trainingdate in *icorpus* will be included in the *profX*.The significance of a n-gram decreases with its rank. Hence, a language-profile will only need to include a small number of the most frequent n-grams—for example the top 1000. This value should be the same across all language-profiles within one *profX*

splits the upper boundary of the range of n

langcode the ISO 639-3-code of the language for which the profile is being generated[5].

[3] For detailed information on how to build text-corpora see for example Wynne (2005)

[4] Both, the english and the german trainingdata are composed of juridical text, newspaper articles and a novel—one page each.

[5] See http://www.sil.org/ISO639-3/codes.asp for all the codes.

One only can identify languages, for which a language-profile exists. As an example, we will continue by augmenting the *profX* by another language: german. The procedure is the same as before:

```
> tdAdr <- system.file("traindata",
+     "tdger", package = "langid")
> (icorp <- Corpus(DirSource(tdAdr),
+     readerControl = list(reader = readPlain)))

A text document collection with 3 text documents

> profX <- make_profX(icorpus, div = 0.5,
+     profX = profX, profXlen = 1000,
+     splits = 5, langcode = "ger")
```

The result is a 2 × 1695 matrix. Each line represents one language, each column one n-gram. The values reveal, how often a specific n-gram has been counted in the trainingdata—if it did belong to the top 1000. This process should be repeated for all the languages required.

The 26-Language profX

A *profX* with 26 language-profiles has been trained and can be loaded via:

```
%> profX <- profXcompl
> data(profX)
> dim(profX)

[1]        26 11022
```

The training-data stems from the European Language Resources Association (2008), Gutenberg (2008) and private sources. Table 2 gives an overview over some characteristica of this language-profiles-collection. If a language belongs to the ISO 639-3 makro-languages set[6], its name is given in the third column. The total number of words used to train each language profile is displayed in the last column.

Although it might prove useful for rather general purposes, specialised tasks will require language-profiles that are generated from tailor-made training-data.

Language Identification

In the preceding chapter it was explained, how language profiles can be trained and combined in one single matrix. Now it shall be explained, how

[6] See SIL-International (2008) for more information on macro-languages.

Table 2: Language Profiles

language	ISO 639-3		cyrillic	Words
	language	macro		
Albanian	alb			54202
Bosnian	bos	hbs		8578
Bulgarian	bul		✓	58058
Danish	dan			52130
German	ger			67475
English	eng			55916
Finish	fin			35859
French	fre			80609
Greek	gre			48274
Italian	ita			43970
Croatian	cro	hbs		49664
Latin	lat			38540
Dutch	dut			50427
Norwegian	nor			35275
Polish	pol			50954
Portuguese	por			46804
Romanian	rum			54694
Russian	rus		✓	45113
Swedish	swe			50351
Serbian	scc	hbs	✓	37027
Serbian	scc	hbs		42514
Slovak	slo			56647
Slovenian	slv			69826
Spanish	spa			53690
Czech	cze			25727
Hungarian	hun			38992

this *profX* can be used to identify the language of text documents. First, one should generate or load the *profX*—just, what we have done above. The next step is, to generate a corpus with all the text documents which should be classified. As an example, ten documents are provided—5 are written in english, five in german.

```
> tdAdr <- system.file("testdata",
+     "td1", package = "langid")
> (icorpus <- Corpus(DirSource(tdAdr),
+     readerControl = list(reader = readPlain)))
```

```
A text document collection with 10 text documents
```

Now we will identify the language of each document:

```
> (lang_ids <- check_langs(icorpus,
+     profX = profX, omw = 20, numw = 100,
+     numng = 100, splits = 5, method = "re"))

 [1] "eng" "eng" "eng" "eng" "eng" "ger"
 [7] "ger" "ger" "ger" "ger"
```

The arguments stand for:

icorpus the name of the corpus with the text documents to be tested, generated using **tm**.

profX the name of the language profile matrix, generated using *make_profX*.

omw number of words to be ommited at the beginning of the text document.

numw number of words to be used for identifying a document's language.

numng number of the most frequent n-grams that are to be used for building the n-gram distribution of the text document.

splits the upper boundary of the range of n

method method used for testing document profile against language profiles—can be *"re"*,*"mce"* or a customized function.

Only the amount of *numw* words is being used for each document to identify its language. They are cut out of the text as a block, starting with the word at position *omw* +1 . This gives the opportunity to skip abstracts or other features of a document, which might not be best suited for identifying the language. From those selected words, n-grams will be generated. Of those n-grams, only the top *numng* n-grams will be used for building the frequency distribution needed for language identification.

The result is the vector *lang_ids*. He contains the ISO 639-3-codes of the identified languages[7]

Comparing the Distributions

There are many ways to compare one distribution to another one, in order to determine, which one fits best. Two methods have allready been implemented: relative entropy("'re"') and mutual cross entropy ("'mce"'). If

[7] Two special ISO 639-2-codes might show up as well: "zxx" means, the text-document is either empty or the text is not suitable for language identification. "und" means the result is ambiguous—more than one language have been identified.

necessary, different measures can be implemented as well. We will use the function ***fun_re*** as an example. Relative entropie is defined as:

$$I(p;q) = \sum_{i=1}^{n} p_i \log \left(\frac{p_i}{q_i} \right) \tag{1}$$

The implemented function looks as follows:

```
> fun_re <- function(langsubset, docprof) {
+     langsubset[which(langsubset ==
+         0)] <- 1e-06
+     matchres <- apply(langsubset,
+         1, function(x, y = docprof) {
+             sum(y * log2(y/x))
+         })
+     lang <- rownames(langsubset)[which(matchres ==
+         min(matchres))]
+     if (length(lang) > 1)
+         lang <- "und"
+     lang
+ }
```

The function requires two arguments:

docprof the frequency distribution of the document's n-grams—p_i.

langsubset the frequency distribution of the same n-grams as in *docprof* within the language-profiles. This is a matrix with one line for each language and one column for each of the n-grams—q_i.

The function must return the argument *lang* wich is either the ISO 639-3-code of the identified language or the ISO 639-2-code ”und”.

References

European Language Resources Association. *Catalog Reference: W0004*, 2008. URL http://catalog.elra.info.

I Feinerer. tm, 2008. URL http://cran.r-project.org/. R package Version 0.3-1.

I Feinerer, K Hornik, and D Meyer. Text mining infrastructure in R. *Journal of Statistical Software*, 25(5), 2008. URL http://www.jstatsoft.org/v25/i05.

Project Gutenberg. *About Project Gutenberg*, 2008. URL http://www.gutenberg.org.

SIL-International. Scope of denotation for language identifiers, 2008. URL `http://www.sil.org/iso639-3/scope.asp#I`. Firmenwebsite, Stand 8. April 2008.

M Wynne. *Developing Linguistic Corpora: a Guide to Good Practice.* Oxbow Books, 2005. Online erhältlich unter: http://ahds.ac.uk/linguistic-corpora/.