Language Guessers

Haixia Liu

School Of Computer Science, University of Nottingham Malaysia Campus, Jalan Broga, 43500 Semenyih, Selangor Darul Ehsan.
khyx3lhi@nottingham.edu.my
Horizon Digital Economy Research, School of Computer Science, University of Nottingham, NG7 2TU, UK.
psxhl2@nottingham.ac.uk

Abstract. Three algorithms (IDme-byTfIdf, IDme-byLRC1 and IDme-byLRC2) are presented to identify the language that a piece of text is written in. The IDme-byTfIdf algorithm constructs TfIdf vectors from the entire text corpus, which contains documents that have ground truth (labeled with the language the text is written in, we call them *docsgtr*), and the document that needs to be tested(*doctest*). The prediction is based on the similarity (cosine measurements) between the *doctest*-TfIdf-vector and the *docsgtr*-TfIdf-vectors. The algorithm IDme-byLRC (LRC stands for Language Recognition Chart) is based on the observation that the language of a text can often be identified by looking up characters specific to that language [1]. IDme-byLRC1 analyzes the entire specific characters (which are recorded in the LRC list) occurrences in the input text, whereas IDme-byLRC2 focuses on characters that belong to only one language (e.g.: ß belongs to German only). By accumulating the characters occurrences (grouped by language), the result was determined by the highest occurrences. The experiments were conducted on a multi-language corpus that contains eight European languages. Two major evaluations were performed, one of which is focusing on detecting longer text (with 400+ lines) and the other is aiming at testing short text (one sentence). The results (with 100-percent accuracy) of testing long text detected by IDme-byLRC1 (tested on 80 files) and IDme-byLRC2 (tested on 8 files) indicated that using LRC is an effective technique to detect language. However, the performance got worse when testing short text using LRC. The simple evaluation on IDme-byTfIdf showed that TfIdf could be a promising technique for identifying languages, which also indicated that information theory can be applied to language detection. IDme-byTfIdf also has difficulty on identifying short text.

Keywords: Language guessing, TfIdf, Language recognition chart

1 Introduction

Automatic language detection is an important task in NLP and IR. *Cybozu.labs* has developed a practical Java library that can detect 49 languages with high

[1] $https : //en.wikipedia.org/wiki/Wikipedia : Language_recognition_chart$

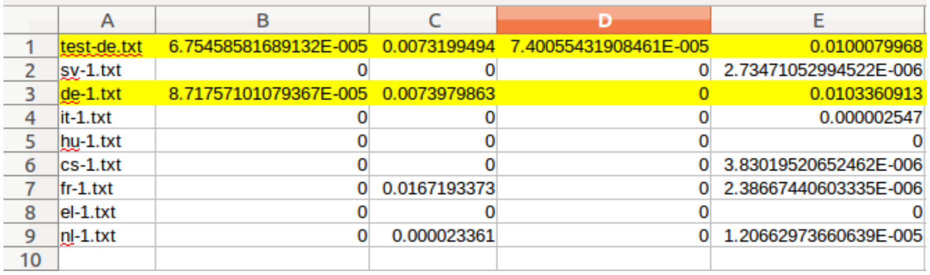
precisions [2]. Their method calculates language probabilities from features of spelling, using Naive Bayes with character n-gram. It is an effective method, but it requires a lot of work to train the profiles. In comparison, the simple method of word-dictionary-matching needs a huge dictionary. In this work, alternatives to detection language are explored.

2 Methodology

2.1 IDme-byTfIdf

This method is inspired by the paper [1]: the relatedness between the two pieces of text can be measured by TfIdf scores. The implementation is based on a Java package (computergodzilla) [3].

The term-pool is built from the full dataset, including documents labeled with ground truth and the text document. The TfIdf vectors are computed based on this term-pool. Every document is represented by a TfIdf vector. The cosine scores between The TfIdf vectors indicate the relatedness between the corresponding documents. The idea of this method is demonstrated in Fig. 1 and 2. We expect that the highlighted vectors (derived from texts written in the same language) are the most similar ones.



	A	B	C	D	E
1	test-de.txt	6.75458581689132E-005	0.0073199494	7.40055431908461E-005	0.0100079968
2	sv-1.txt	0	0	0	2.73471052994522E-006
3	de-1.txt	8.71757101079367E-005	0.0073979863	0	0.0103360913
4	it-1.txt	0	0	0	0.000002547
5	hu-1.txt	0	0	0	0
6	cs-1.txt	0	0	0	3.83019520652462E-006
7	fr-1.txt	0	0.0167193373	0	2.38667440603335E-006
8	el-1.txt	0	0	0	0
9	nl-1.txt	0	0.000023361	0	1.20662973660639E-005
10					

Fig. 1. TfIdf vectors between test file (written in German) and other files that are labeled with ground truth

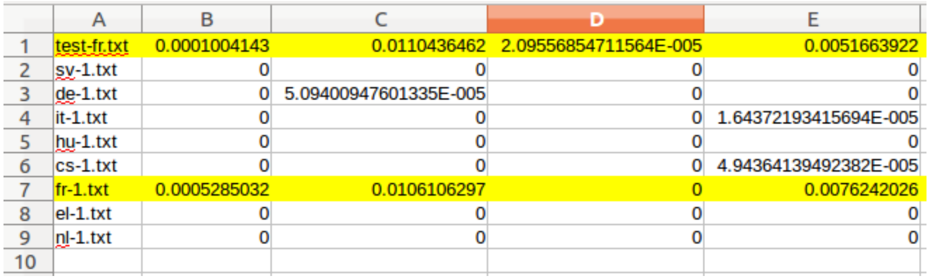
2.2 IDme-byLRC

This method is based on the observation that the language of a text can often be identified by looking up characters specific to that language [4] As wikipedia

[2] http://code.google.com/p/language-detection/

[3] Credits to its original author Mubin Shrestha. http://computergodzilla.blogspot.co.uk/2013/07/how-to-calculate-tf-idf-of-document.html

[4] Credits to Wikipedia:Language recognition chart (LRC).

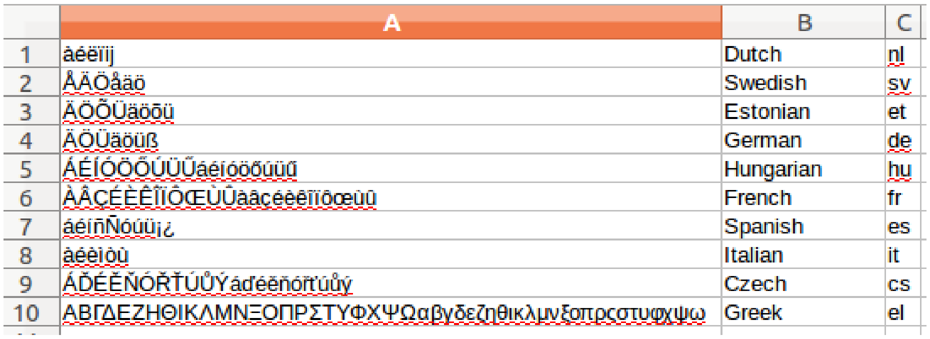


	A	B	C	D	E
1	test-fr.txt	0.0001004143	0.0110436462	2.09556854711564E-005	0.0051663922
2	sv-1.txt	0	0	0	0
3	de-1.txt	0	5.09400947601335E-005	0	0
4	it-1.txt	0	0	0	1.64372193415694E-005
5	hu-1.txt	0	0	0	0
6	cs-1.txt	0	0	0	4.94364139492382E-005
7	fr-1.txt	0.0005285032	0.0106106297	0	0.0076242026
8	el-1.txt	0	0	0	0
9	nl-1.txt	0	0	0	0
10					

Fig. 2. TfIdf vectors between test file (written in French) and other files that are labeled with ground truth

stated: Language recognition chart describes a variety of simple clues one can use to determine what language a document is written in with high accuracy. IDme-byLRC algorithm works as a combination of statistical and rule based classifier.

The list of LRC is constructed by parsing the html file provided by wikipedia. Fig. 3 shows part of the characters together with the language that the characters belong to specifically. We can see that some characters belong to more than one languages. We excluded Latin characters due to their high frequency in most of the western languages. The strategy to detect English is: if there is no specific character detected in the input text, it is probably written in English.



	A	B	C
1	àéëïij	Dutch	nl
2	ÅÄÖåäö	Swedish	sv
3	ÄÖÕÜäöõü	Estonian	et
4	ÄÖÜäöüß	German	de
5	ÁÉÍÓÖŐÚÜŰáéíóöőúüű	Hungarian	hu
6	ÀÂÇÉÈÊÎÏÔŒÙÛàâçéèêîïôœùû	French	fr
7	áéíñÑóúü¡¿	Spanish	es
8	àéèìòù	Italian	it
9	ÁĎÉĚŇÓŘŤÚŮÝáďéěňóřťúůý	Czech	cs
10	ΑΒΓΔΕΖΗΘΙΚΛΜΝΞΟΠΡΣΤΥΦΧΨΩαβγδεζηθικλμνξοπρςστυφχψω	Greek	el

Fig. 3. Part of the characters together with the language that the characters belong to specifically.

1. IDme-byLRC1. Fig. 4 shows the idea of IDme-byLRC1. After obtaining the character distribution (only consider language, not count by specific character), the language with the highest proportion would be judged as the language that the input text is written in. The algorithm is detailed in the method *idmebyonehot* in the java file *IDme.java*.

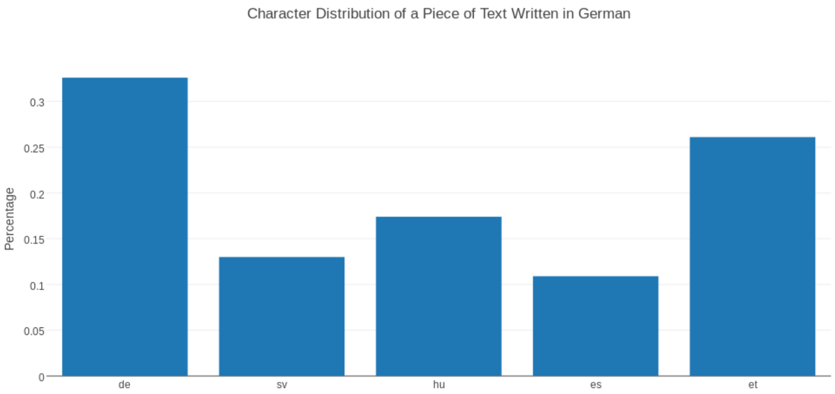


Fig. 4. Character occurrences from a piece of text written in German. It shows that the percentage of German characters is higher than others.

2. IDme-byLRC2. This algorithm differs from IDme-byLRC1 in such a way that it only takes into account the characters that belong to a unique language. The output differences between IDme-byLRC1 and IDme-byLRC2 were shown in Fig. 10 and 11 respectively. The algorithm is detailed in the method *idmebydistribution* in the java file *IDme.java*.

3 Evaluation

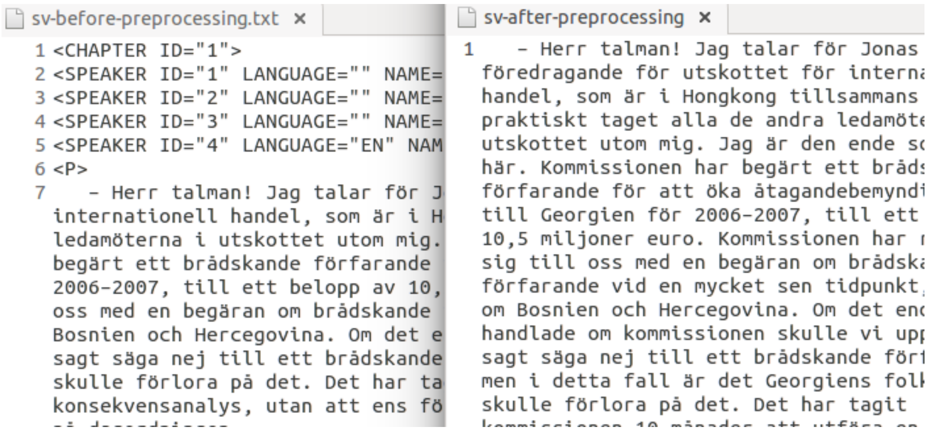
3.1 Dataset

In general, the multi-language text files from *Europarl* are used in this experiment [5]. Eight categories of languages are targeted: sv (Swedish), de(German), it (Italian), hu (Hungarian), cs (Czech), fr (French), el (Greek), nl (Dutch). The pre-processing step removes files that contains less than 400 lines as well as excludes lines with tags, because there are Latin characters inside the tags. Fig. 5 shows the differences before and after pre-processing. In this step, files were renamed according to the following standard: language-filename.txt, which makes it easier to compare the detected language name and the ground truth (e.g.: de-1.txt).

1. Sub-dataset1. Ten files from each category (there are 8 categories in total) are selected randomly to form the test set ($t80$).
2. Sub-dataset2. One file from each category is chosen randomly to form a smaller dataset ($t8$) [6]

[5] http://www.statmt.org/europarl/

[6] The processing time of IDme-byTfIdf and IDme-byLRC2 is about one hour on the smaller dataset. It is not practical to perform experiment on the larger dataset using low performance computer at the moment.



```
<CHAPTER ID="1">
<SPEAKER ID="1" LANGUAGE="" NAME=
<SPEAKER ID="2" LANGUAGE="" NAME=
<SPEAKER ID="3" LANGUAGE="" NAME=
<SPEAKER ID="4" LANGUAGE="EN" NAM
<P>
  - Herr talman! Jag talar för J
internationell handel, som är i H
ledamöterna i utskottet utom mig.
begärt ett brådskande förfarande
2006-2007, till ett belopp av 10,
oss med en begäran om brådskande
Bosnien och Hercegovina. Om det e
sagt säga nej till ett brådskande
skulle förlora på det. Det har ta
konsekvensanalys, utan att ens fö
```

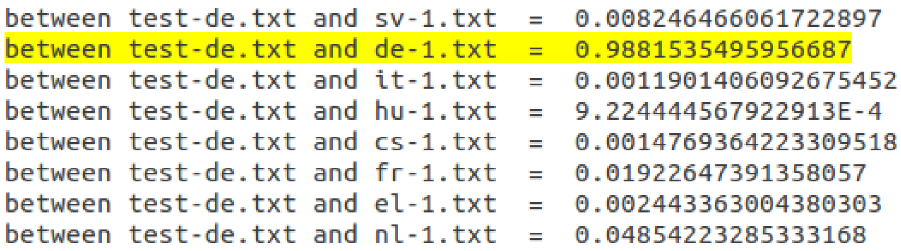
```
  - Herr talman! Jag talar för Jonas
föredragande för utskottet för interna
handel, som är i Hongkong tillsammans
praktiskt taget alla de andra ledamöte
utskottet utom mig. Jag är den ende sc
här. Kommissionen har begärt ett bråds
förfarande för att öka åtagandebemyndi
till Georgien för 2006-2007, till ett
10,5 miljoner euro. Kommissionen har r
sig till oss med en begäran om brådska
förfarande vid en mycket sen tidpunkt,
om Bosnien och Hercegovina. Om det enc
handlade om kommissionen skulle vi upp
sagt säga nej till ett brådskande fört
men i detta fall är det Georgiens folk
skulle förlora på det. Det har tagit
kommissionen 10 månader att utföra en
```

Fig. 5. Examples from the dataset: before pre-processing on the left and after preprocessing on the right.

3. Sub-dataset3. Sub-dataset3 is the sub-dataset of sub-dataset2. There is only one sentence in every file. Sub-dataset3 is used for testing the algorithm performance on short text.
4. Sub-dataset3plus (For testing IDme-byTfIdf on detecting short text). Sub-dataset3plus is constructed by doubling dataset3plus in such a way that there is no overlap lines between two files with same language, which guarantees *doctest*-TfIdf-vector does not appear in the *docsgtr*-TfIdf-vectors.

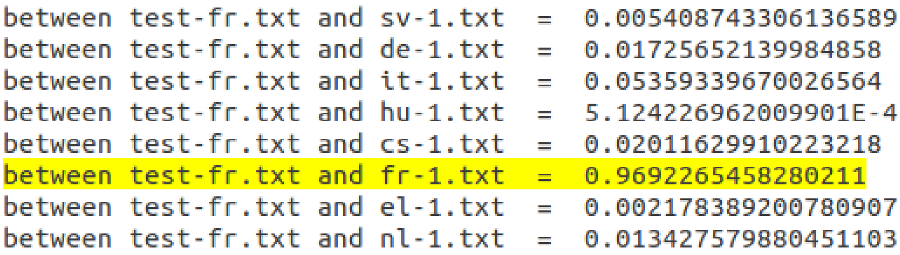
3.2 Results of IDme-byTfIdf

TfIdf vectors are computed from the randomly select 10 files(one file from each category), together with the test file.



```
between test-de.txt and sv-1.txt  =  0.008246466061722897
between test-de.txt and de-1.txt  =  0.9881535495956687
between test-de.txt and it-1.txt  =  0.0011901406092675452
between test-de.txt and hu-1.txt  =  9.224444567922913E-4
between test-de.txt and cs-1.txt  =  0.0014769364223309518
between test-de.txt and fr-1.txt  =  0.01922647391358057
between test-de.txt and el-1.txt  =  0.002443363004380303
between test-de.txt and nl-1.txt  =  0.04854223285333168
```

Fig. 6. Cosine scores between *doctest*-TfIdf-vector and *docsgtr*-TfIdf-vectors. The *doctest* is test-de.txt, which is written in German.



```
between test-fr.txt and sv-1.txt  =  0.005408743306136589
between test-fr.txt and de-1.txt  =  0.01725652139984858
between test-fr.txt and it-1.txt  =  0.05359339670026564
between test-fr.txt and hu-1.txt  =  5.124226962009901E-4
between test-fr.txt and cs-1.txt  =  0.02011629910223218
between test-fr.txt and fr-1.txt  =  0.9692265458280211
between test-fr.txt and el-1.txt  =  0.002178389200780907
between test-fr.txt and nl-1.txt  =  0.013427579880451103
```

Fig. 7. Cosine scores between *doctest*-TfIdf-vector and *docsgtr*-TfIdf-vectors. The *doctest* is test-fr.txt, which is written in French.

3.3 Results of IDme-byLRC

Both IDme-byLRC1 and IDme-byLRC2 gave 100-percent accuracy score on the tests conducted on datasets $t80$ and $t8$ respectively.

4 Discussion

4.1 When the language recognition chart is getting rich

Fig. 8 in the Appendix showed the enriched LRC list, which contains more characters from more languages. We can see that there are more overlapping between several languages. In the Appendix, two figures demonstrated a false detection occurred due to the overlapping characters between Dutch and Occitan.

4.2 Computational complexity

Table. 1 showed the complexity by analyzing the processing time that it costs for finished the experiment on the same dataset $t8$.

Table 1. Computational complexity of the three algorithms

Algorithm	Processing time
IDme-ByTfIdf	60 minutes
byLRC1	5 seconds
IDme-byLRC2	40 minutes

4.3 Detecting short text

Expectedly, the language detection accuracy based on LRC is depending on the length of the input text: the shorter the lower accuracy. It is because, the proportion of specific characters in the input text depends on the characters distribution, which is relatively fixed by nature. The IDme-byTfIdf also showed its weakness on detecting short text. Table. 2 showed the detection results based on Sub-dataset3plus. Note that the dataset is not big enough to produce results that are statistically reliable, but we can have a general view about the detection ability of the algorithms.

Table 2. results of detecting short text

Algorithm	accuracy
IDme-ByTfIdf	0.63
byLRC1	0.56
IDme-byLRC2	0.56

Table 3. Examples of Failed Detections on Short Text

Algorithm	GTR	Predict [7]
IDme-ByTfIdf	cs	sv
IDme-ByTfIdf	fr	nl
IDme-ByTfIdf	el	-
byLRC1	cs	-
byLRC1	de	et
byLRC1	de	-
byLRC1	fr	nl
byLRC1	nl	-
byLRC2	de	-
byLRC2	fr	-
byLRC2	it	-
byLRC2	nl	-

4.4 Limitations

In this work focuses on European languages. How well the algorithms work on other languages needs further investigation.

5 Conclusion

IDme-byLRC1 is a practical method to detect longer text considering the time costs. The more shared-characters (multi-languages share the same specific characters) are in the LRC list, the more difficult it is to detect the language. IDme-byTfIdf maybe more robust given enough data to build the TfIdf matrix.

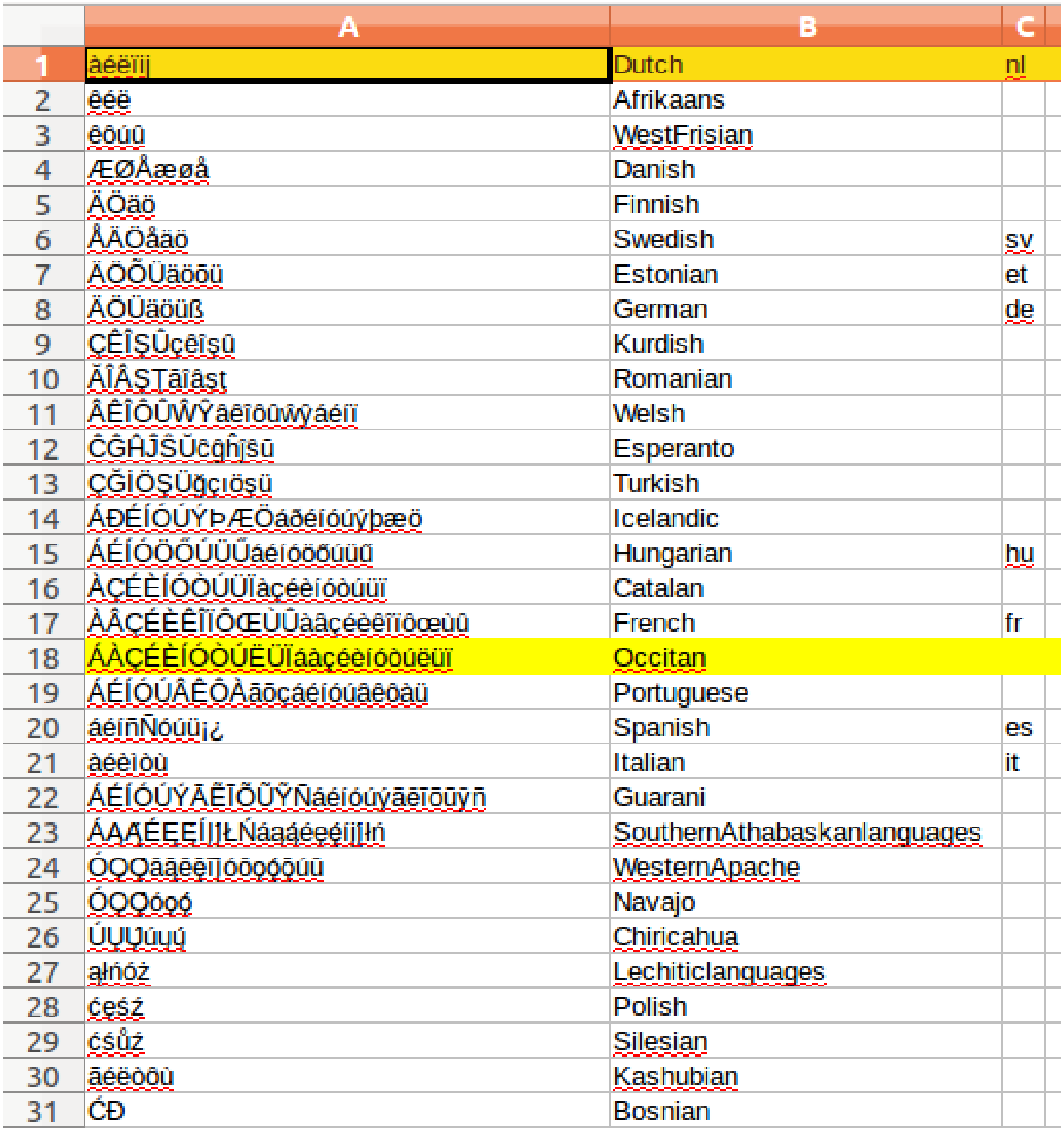
6 Acknowledgement

Thanks for UKP for giving me this interesting task.

References

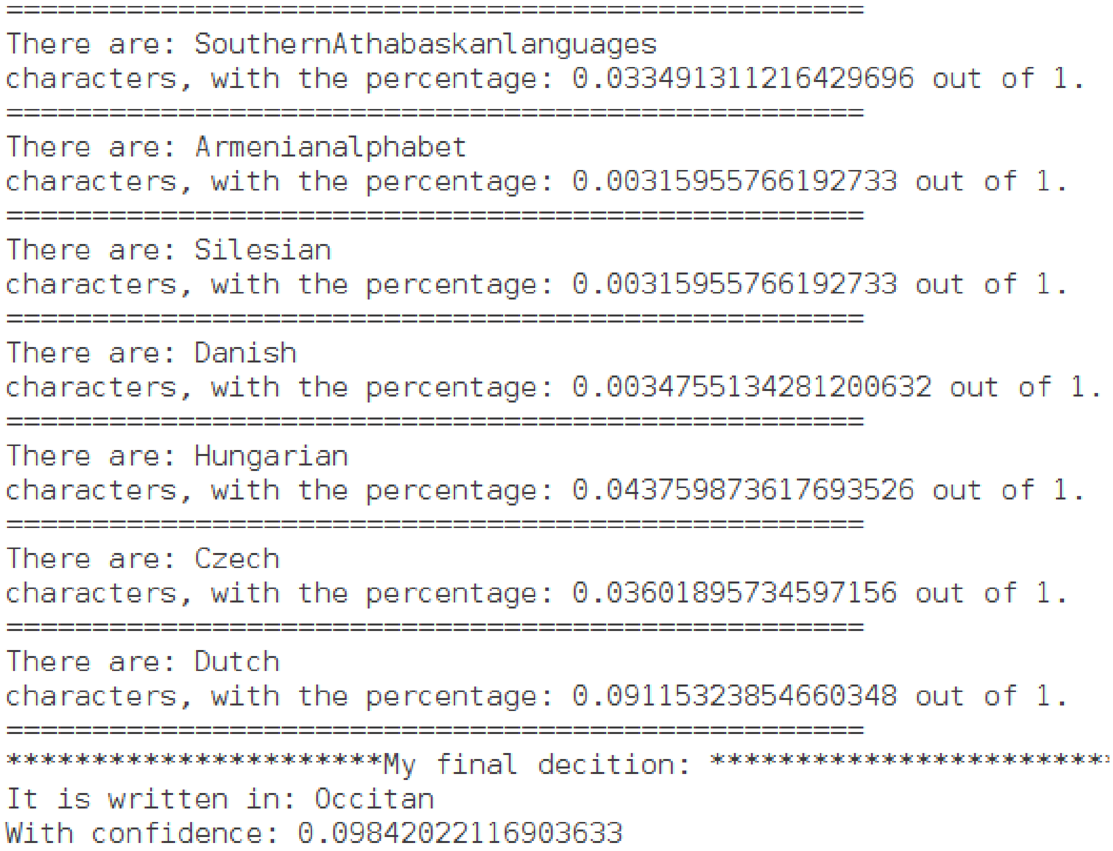
1. E. Gabrilovich and S. Markovitch, “Computing semantic relatedness using wikipedia-based explicit semantic analysis.,” in *IJCAI*, **7**, 1606–1611 (2007).

A Appendix



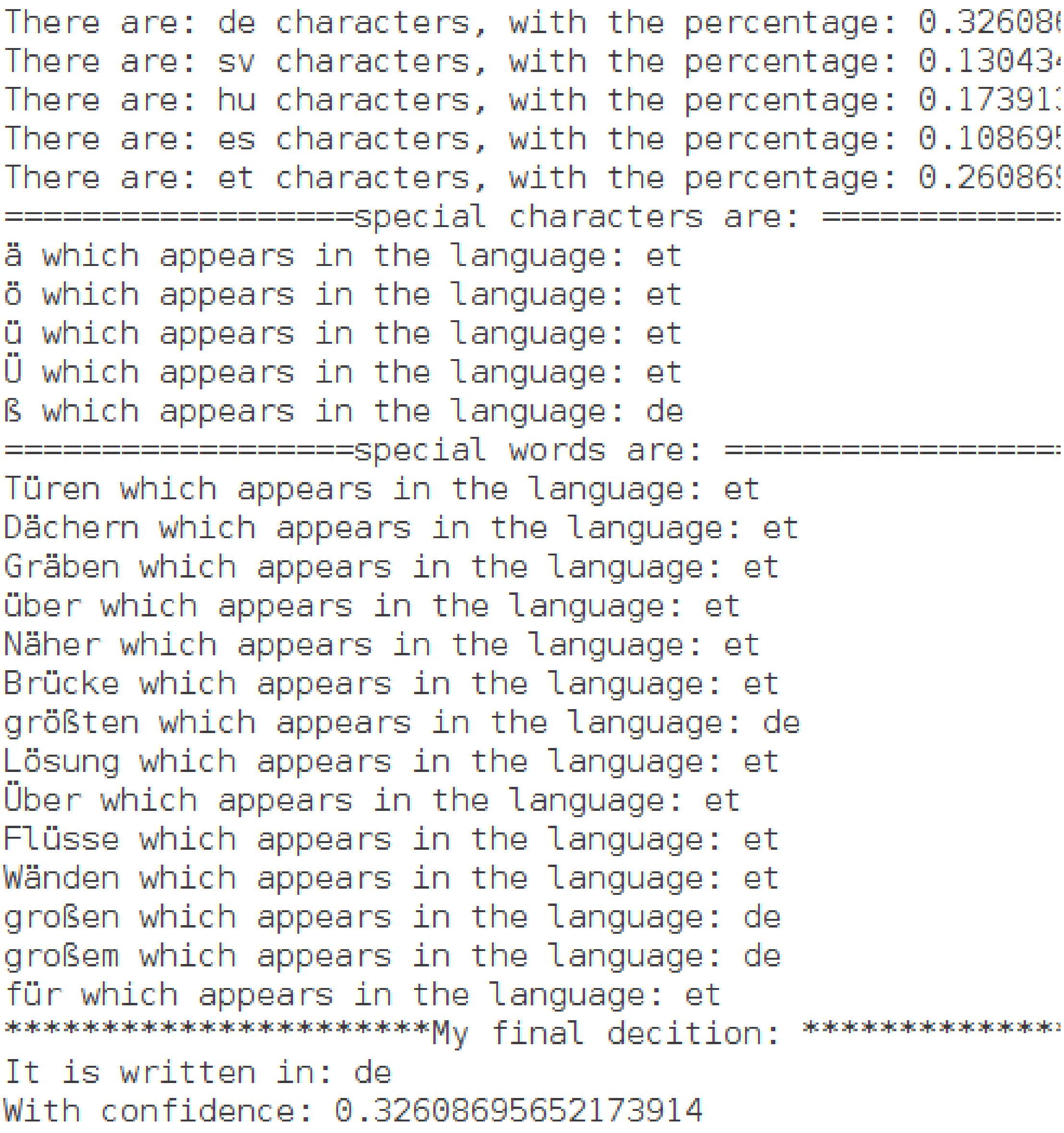
	A	B	C
1	àéëïĳ	Dutch	nl
2	êéë	Afrikaans	
3	êôúû	WestFrisian	
4	ÆØÅæøå	Danish	
5	ÄÖäö	Finnish	
6	ÅÄÖåäö	Swedish	sv
7	ÄÖÕÜäöõü	Estonian	et
8	ÄÖÜäöüß	German	de
9	ÇÊÎŞÛçêîşû	Kurdish	
10	ĂÎÂŞŢăîâşţ	Romanian	
11	ÂÊÎÔÛŴŶâêîôûŵŷáéíï	Welsh	
12	ĈĜĤĴŜŬĉĝĥĵŝŭ	Esperanto	
13	ÇĞİÖŞÜğçıöşü	Turkish	
14	ÁÐÉÍÓÚÝÞÆÖáðéíóúýþæö	Icelandic	
15	ÁÉÍÓÖŐÚÜŰáéíóöőúüű	Hungarian	hu
16	ÀÇÉÈÍÓÒÚÜÏàçéèíóòúüï	Catalan	
17	ÀÂÇÉÈÊÎÏÔŒÙÛàâçéèêîïôœùû	French	fr
18	ÁÀÇÉÈÍÓÒÚËÜÏáàçéèíóòúëüï	Occitan	
19	ÁÉÍÓÚÂÊÔÀãõçáéíóúâêôàü	Portuguese	
20	áéíñÑóúü¡¿	Spanish	es
21	àéèìòù	Italian	it
22	ÁÉÍÓÚÝÃẼĨÕŨỸÑáéíóúýãẽĩõũỹñ	Guarani	
23	ÁĄĄ́ÉĘĘ́ÍĮĮ́ŁŃáąą́éęę́íįį́łń	SouthernAthabaskanlanguages	
24	ÓǪǬāą̄ēę̄īį̄óōǫǫ́ǭúū	WesternApache	
25	ÓǪǪ́óǫǫ́	Navajo	
26	ÚŲŲ́úųų́	Chiricahua	
27	ąłńóż	Lechiticlanguages	
28	ćęśź	Polish	
29	ćśůż	Silesian	
30	ãéëòôù	Kashubian	
31	ĆĐ	Bosnian	

Fig. 8. Enriched-LRC list, with Dutch and Occitan categories highlighted.



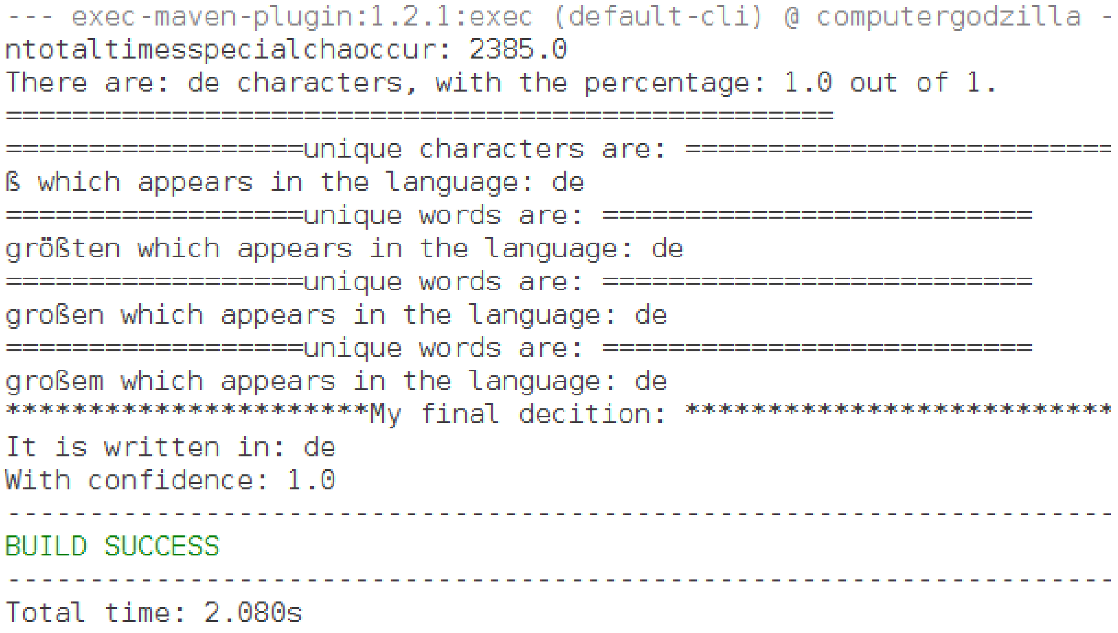
```
==================================================
There are: SouthernAthabaskanlanguages
characters, with the percentage: 0.033491311216429696 out of 1.
==================================================
There are: Armenianalphabet
characters, with the percentage: 0.00315955766192733 out of 1.
==================================================
There are: Silesian
characters, with the percentage: 0.00315955766192733 out of 1.
==================================================
There are: Danish
characters, with the percentage: 0.0034755134281200632 out of 1.
==================================================
There are: Hungarian
characters, with the percentage: 0.043759873617693526 out of 1.
==================================================
There are: Czech
characters, with the percentage: 0.03601895734597156 out of 1.
==================================================
There are: Dutch
characters, with the percentage: 0.09115323854660348 out of 1.
==================================================
**********************My final decition: ***********************
It is written in: Occitan
With confidence: 0.09842022116903633
```

Fig. 9. The algorithm is confused between Dutch and Occitan. The ground truth is Dutch, but Occitan detected.



```
There are: de characters, with the percentage: 0.326086
There are: sv characters, with the percentage: 0.130434
There are: hu characters, with the percentage: 0.173913
There are: es characters, with the percentage: 0.108695
There are: et characters, with the percentage: 0.260869
==================special characters are: ============
ä which appears in the language: et
ö which appears in the language: et
ü which appears in the language: et
Ü which appears in the language: et
ß which appears in the language: de
==================special words are: =================
Türen which appears in the language: et
Dächern which appears in the language: et
Gräben which appears in the language: et
über which appears in the language: et
Näher which appears in the language: et
Brücke which appears in the language: et
größten which appears in the language: de
Lösung which appears in the language: et
Über which appears in the language: et
Flüsse which appears in the language: et
Wänden which appears in the language: et
großen which appears in the language: de
großem which appears in the language: de
für which appears in the language: et
**********************My final decition: *************
It is written in: de
With confidence: 0.32608695652173914
```

Fig. 10. The detection result using IDme-ByLRC1



```
--- exec-maven-plugin:1.2.1:exec (default-cli) @ computergodzilla -
ntotaltimesspecialchaoccur: 2385.0
There are: de characters, with the percentage: 1.0 out of 1.
==================================================
==================unique characters are: ==========================
ß which appears in the language: de
==================unique words are: ==========================
größten which appears in the language: de
==================unique words are: ==========================
großen which appears in the language: de
==================unique words are: ==========================
großem which appears in the language: de
**********************My final decition: **************************
It is written in: de
With confidence: 1.0
-------------------------------------------------------------------
BUILD SUCCESS
-------------------------------------------------------------------
Total time: 2.080s
```

Fig. 11. The detection result using IDme-ByLRC2. The input text is the same as in Fig. 10