langdetect: A Language Detection Library

Federico López

1 Method

The method implemented for language detection was taken from the article N-Gram-Based Text Categorization [1]. It is a supervised method, therefore it requires texts of a known language beforehand.

The algorithm builds the languages profiles by counting the occurrences of N-grams of different length in every text. In the training phase it creates the models for every available idiom. This means that it counts the frequencies along many texts of the same language. The model only keeps a fixed amount of the most frequent N-grams. Usually this quantity is between 300 and 700.

The measure used to compare two profiles is the difference of the positions of the same N-gram in both models, after they are sorted by frequency. If we have profiles A and B, the distance is given by the formula:

$$dist(A, B) = \sum_{ngram \varepsilon A} |pos_A(ngram) - pos_B(ngram)| \tag{1}$$

In case that the N-gram in profile A is not included in profile B, a penalization distance is added. Finally, the profile of the given text is compared with all the languages profiles, the closest one is chosen as the detected language.

To sum up, the steps followed are:

1. Build N-gram frequencies profiles for every available language.
2. Build N-gram frequencies profiles for the given text.
3. Compare the profile of the text with all the languages profiles according to the distance measure.
4. Choose as detected language the one corresponding with the profile to the minimum distance.

2 Implementation

The library was implemented in Python and the code is available on Github[1].

Currently the algorithm can detect among 23 languages, but many more can be easily added. The available languages are: Arabic, Czech, Danish, English, Estonian, Finnish, French, German, Greek, Hebrew, Hungarian, Italian, Latvian, Lithuanian, Norwegian, Persian, Polish, Portuguese, Romanian, Russian, Slovak, Spanish and Swedish. Note that not only latin alphabets are tolerated.

[1] `https://github.com/fedelopez77/langdetect`

3 Usage

Given a text, it returns a list of three tuples, sorted according to the probabilites of that text belonging to each language. The tuples are `(language_code, probability)`. The language codes follow the ISO 639-1 Standard [2].

- Command-line usage:

```
cd path/to/folder/langdetect/
python langdetect.py -f FILE
```

- Library usage:

```
>>> text = """Automatic summarization is the
process of reducing a text document with a
computer program in order to create a
summary that retains the most important
points of the original document."""

>>> import langdetect as ld
>>> print ld.detect_language(text)
[('en', 0.9179458640640037),
('ro', 0.04370950743932613),
('pt', 0.03834462849667017)]
```

Since building the models for every language it is a time consuming operation, to perform many detections it is better to use:

```
>>> profiles = ld.create_languages_profiles()
>>> ld.detect_language(text, profiles)
```

4 Hyperparameters optimization

The texts used to train, validate and test the algorithm were randomly taken from Wikipedia articles in all the available languages. There are texts from 50 characters up to many paragraphs. The datasets were defined as:

– Train dataset: 1,530 articles. 120 Kb of texts per language approximately.
– Validation dataset: 3,674 articles. 250 Kb of texts per language approximately.
– Test dataset: 6,598 articles. 500 Kb of texts per language approximately.

Considering the language with higher probability as the detected one, the main measure used to evaluate the performance of the algorithm was precision. This is:

$$Precision = \frac{Correctly\ detected\ texts}{Total\ texts} \tag{2}$$

The configurable parameters of the algorithm are:

[2] `https://www.w3schools.com/TAgs/ref_language_codes.asp`

– Minimum length of n-grams.
– Maximum length of n-grams.
– Amount of n-grams used for every language profile.
– Penalization distance.

A grid search was conducted by adjusting all this variables to a different set of values, training the algorithm with the train dataset, and checking results on the validation dataset. The best parameters obtained were:

– Minimum length of n-grams: 1
– Maximum length of n-grams: 3
– Amount of n-grams used for every language profile: 400
– Penalization distance: 700

5 Results

With the parameters obtained by the grid search, the following results were gathered on the test dataset:

Table 1. Evaluation results.

Language	**Texts**	**Precision** (%)	**Language**	**Texts**	**Precision** (%)
Arabic	206	99.03	Latvian	275	99.27
Czech	211	94.79	Lithuanian	392	99.74
Danish	346	93.35	Norwegian	340	97.65
English	205	97.56	Persian	472	99.58
Estonian	353	97.73	Polish	333	97.6
Finnish	255	99.22	Portuguese	332	98.8
French	208	98.56	Romanian	453	97.35
German	153	98.69	Russian	102	100
Greek	80	100	Slovak	408	98.77
Hebrew	104	100	Spanish	197	99.49
Hungarian	245	100	Swedish	704	98.44
Italian	224	96.88	**Total**	**6598**	**98.22**

6 Conclusion

It can be noticed that these numbers match the results collected by Cavnar and Trenkle [1] as well as by Řehůřek and Kolkus [2] using the same method for profile lengths of 400 N-grams and texts of more than 50 characters.

References

1. Cavnar, W.B., Trenkle, J.M.: N-gram-based text categorization. In: Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval. pp. 161–175 (1994)
2. Radim Řehůřek, M.K.: Language identification on the web: Extending the dictionary method. In: 10th International Conference in Computational Linguistics and Intelligent Text Processing. pp. 357–368. Mexico (March 2009)