Automatic Bilingual Language Identification in Short, Noisy Texts

Clarissa Castella Xavier
Meedan
clarissa@meedan.com

Karim Ratib
Meedan
karim@meedan.com

Abstract

Language identification is the task of determining the language in which a text is written. Besides the task being very well addressed for monolingual long texts, it is an open issue for noisy and short texts, as microblog posts. This work introduces a language identification approach designed to work with noisy and short texts, that identifies one or two language per input. Our evaluation demonstrates high accuracy over monolingual and bilingual microblog texts.

1 Introduction

Noisy text are found in informal settings such as microblogs, online chat, email, social message boards, news-group postings, blogs, wikis and web pages. Such text may contain spelling errors, abbreviations, non-standard terminology, missing punctuation, misleading case information, as well as false starts, repetitions, and special characters (Contractor et al., 2010). The Tweet *Much lov 4 u tmorrow* is noisy because some of the words are contracted, misspelled or replaced by symbols.

According Lui, Lau and Baldwin (Lui et al., 2014) “language identification is the task of automatically detecting the language(s) present in a document based on the content of the document”. Language identification task has been considered solved for long documents written in a single language, among others conditions (McNamee, 2005). However, according (Zubiaga et al., 2014) “the emergence of social media and the chatspeak employed by its users has brought about new previously unseen issues that need to be studied in order to deal with these kinds of texts”.

The literature points out three issues that still need to be addressed in language identification: multilingual documents (Lui et al., 2014); short texts (Carter et al., 2013) (Laboreiro et al., 2013); distinction of similar languages (Zampieri, 2013). Here, we work with the first two issues.

We introduce an approach to identify single languages or pairs of languages from a short, noisy text, like a Twitter[1] post. In case of pairs, our system currently assumes one of the languages is English. The method builds a model with monolingual training data trained over a vector space model of character 3-grams using TF-IDF, a well-know method to convert the text into VSM (Rajaraman and Ullman, 2011). The language identification is made combining heuristics with TF-IDF similarity calculation.

The experiment empirically demonstrates the effectiveness of our method, by evaluating it over a monolingual and a bilingual datasets of Tweets. The evaluation has shown that our system, XX’s LangId, achieves accuracy higher than comparable systems, Langid.py (Lui et al., 2012) (Lui and Baldwin, 2011) and Polyglot (Lui et al., 2014). The rest of the paper is organized as follows. In Section 2 we include an overall description language identification, the work regarding short texts and present the two state-of-the-art identifiers used in the evaluation. Section 3 outlines our method and describes its implementation. Section 4 presents evaluation, results and carries out a short analysis of the issues encountered in the assessment and their possible solutions.

[1]www.twitter.com

2 Language Identification

Language identification is the task of determining the language in which a text is written. It has been formulated as a supervised machine learning task (Lui and Baldwin, 2011) and defined as a text classification problem (Zubiaga et al., 2014). Currently the dominant approach, based in (Cavnarand and Trenkle, 1994), is using of n-grams to learn one model per language and to represent the document to be classified.

2.1 Short Texts Language Identification

The input document size has been considered quite important for the identification results. As larger the input document, higher the accuracy (Lui et al., 2012) (Lui and Baldwin, 2011) (Cavnarand and Trenkle, 1994). The rise of worldwide social networks like Twitter brough relevancy to the language identification of short strings (Lui et al., 2012), reappearing the interest for the subject.

The global growth and adoption of the social media platform in recent years has increased the diversity in the use of different languages in Twitter. This brought to light the interest in automatically identifying the languages of tweets. According (Zubiaga et al., 2014) "the identification of the language of a tweet is crucial for the subsequent application of NLP tools such as machine translation, sentiment analysis, or information extraction". Therefore, it is important that a tweet language is properly identified to facilitate the application of NLP resources suitable to it (Zubiaga et al., 2014).

2.2 Language Identifiers

Following we briefly present two state-of-the-art language identifiers. First, a monolingual tool whose authors report experiments with short, noisy texts from Twitter. The second one is a a multilanguage identifier.

2.2.1 Langid.py

Langid.py (Lui et al., 2012) (Lui and Baldwin, 2011) is a monolingual language identifier. It is trained over a Naive Bayes classifier with a multinomial event model, over a mixture of byte n-grams. It implements a supervised classifier that contain a pre-trained multi-domain model.

The authors report experiments with several corpus. We consider relevant the results using Twitter datasets: 0.941 of accuracy for T-BE (Tromp and Pechenizkiy, 2011) and 0.886 for T-SC (Carter et al., 2013).

2.2.2 Polyglot

Polyglot (Lui et al., 2014) aims to detect multilingual documents, identify each language and estimate the proportion of the document written in that language. It uses an approach based in a probabilistic mixture model, using Langid.py monolingual document representation. The model assumes that each document is generated as samples from a mixture of languages from the training set. It uses a sampler to map samples to languages for any given set of languages and use this to select the languages that maximizes the probabilities of the document.

The authors report an evaluation of the method performance under Wikipedia Multi dataset (a mixture of monolingual and multilingual documents from Wikipedia). The results were 0.963 of Precision, 0.955 of Recall and 0.959 F-measure.

Other evaluation was performed in order to measure the ability of identifying documents that contain text in English in addition to another target language. Here, it used a collection of 149 documents manually-annotated data in 42 languages from the web. Polyglot was re-trained for each target language, using only training data for English and the target language. It achieved 100% accuracy in the target language detection. The results for English detection were 0.907 of Precision, 0.916 of Recall and 0.912 F-measure.

3 Solution

Our goal is to identify the language or pair of languages in which a input text is written. In case of pair, one of the languages need to be English. Specifically we work with short and noisy texts, like

```
1.  Model generation
2.  Generate input text similarities array
3.  If multi-language input verification is needed
4.    Normalize and generate and words 3-gram
5.    For each word 3-gram:  calculate similarities array
6.    Generate probability pairs array
7.  If single probabilities array generation is needed
8.    Generate single probabilities array
```

Figure 1: Language identification process.

Twitter posts. Like (Lui et al., 2014), our solution works with monolingual training data. Monolingual documents corpora are widely available on the web, which facilitates the inclusion of a larger number of languages according to the necessity.

For practical purposes, in order to ensure reliable information to the user, we assume that in case of a bilingual post, is more important to identify correctly one of the languages of the input, than identifying a pair containing a wrong language.

Figure 1 presents the outline of our solution, giving a broad understanding of the process as a whole. Next we detail each line.

3.1 Model generation

The product of this step is a VSM representing the training set of documents as vectors in a high-dimensional space (McNamee, 2005). VSM is an algebraic model for representing text documents as vectors of identifiers (Salton, 1975). We consider that each language is represented by one document. Each vector dimension corresponds to a separate term. If a term occurs in the document, its value is non-zero. Several different ways of computing these values, have been developed. We use TF-IDF.

In the TF-IDF weighting each component of a vector corresponds to the TF-IDF value of a particular term in the corpus dictionary, a character 3-gram in our case. TF-IDF term weighting assigns a high weight to a term, if it occurs frequently in the document but rarely in the whole document collection (Manning et al., 2008).

3.1.1 Generate input text similarities array

Calculate the similarity metric between the input text and each language (document) in the TF-IDF VSM generated in 3.1 (Section 3.2.1). With this result, a series of language heuristics are applied (Section 3.2.2) generating similarities array for the input text.

3.1.2 Normalization and characters 3-Gram generation

Here, a string of normalized character 3-grams is generated. First, all text is converted to lowercase. Then, URLs, hashtags, users references, emojis, typical noise expressions (as *hahaha* and *kkk*) and punctuation are removed. It is important to note that not all punctuation is removed. For example, exclamation and question points are removed, but hyphens are not, since they are important for detecting languages as Portuguese.

Chang and Lin (Chang and Lin, 2014) argues that “character ngrams prove to be valuable features because languages often have distinct character combinations” and that these features are widely used by existing programs. Table 1, column 2 shows the input text in column 1 after normalization. Then the normalized text is splited into character 3-grams, as shown in column 3.

3.1.3 Model generation

The product of this step is a VSM representing the training set of documents as vectors in a high-dimensional space (Rehurek and Sojka, 2010). VSM is an algebraic model for representing text documents as vectors of identifiers [15]. We consider that each language is represented by one document.

Input text	Normalized text	Character 3-grams
Eh di kayo na	eh di kayo na	[eh] [h d] [di] [di]
nasa isang tent! #	nasa isang tent	[i k] [ka] [kay] [ayo]
ALDUBSummerAdventure		[yo] [o n] [na] [a n]
https://t.co/sE7UXCt2aq		[na] [nas] [asa] [sa]
		[a i] [is] [isa] [san]
		[ang] [ng] [g t] [te]
		[ten] [ent]

Table 1: Example of normalization and character 3-gram generation.

```
(1, 0.040080883) (13, 0.038130838) (8, 0.030514058),
(4, 0.025788601) (5, 0.018582165) (9, 0.018380802),
(7, 0.0095378431) (10, 0.0050673038) (6, 0.0013473346),
(12, 0.00067303592) (14, 0.0002794051) (15, 0.00015746937),
(11, 0.0001403786) (0, 6.7040637e-05) (2, 0.0)
(3, 0.0)
```

Figure 2: Similarities of the input *I love u 4 ever* against each language in VSM. Each pair inside the vector contains (document id, similarity measure).

Each vector dimension corresponds to a separate term. If a term occurs in the document, its value is non-zero. Several different ways of computing these values, have been developed. We use TF-IDF.

In the TF-IDF weighting each component of a vector corresponds to the TF-IDF value of a particular term in the corpus dictionary, a character 3-gram in our case. TF-IDF term weighting assigns a high weight to a term, if it occurs frequently in the document but rarely in the whole document collection (Manning et al., 2008).

3.2 Generate input text similarities array

Calculate the similarity metric between the input text and each language (document) in the TF-IDF VSM generated in 3.1 (Section 3.2.1). With this result, a series of language heuristics are applied (Section 3.2.2) generating similarities array for the input text.

3.2.1 TF-IDF VSM similarity measure

Our goal is to calculate the similarity between the input text and each language represented in the documents in the TF-IDF VSM described in Section 3.1.1.

The first step is to transform the input text into a string of character 3-grams, using the method described in 3.1.2, to generate an input document. Then, the similarity measure between the input document and each language (being each language one document in the VSM) is calculated. The result is an array with the following information per language: document id and similarity measure (Figure 2).

3.2.2 Language heuristics

In order to improve the similarity results, we apply several heuristics over the input text and the similarity result. If one of the requirements described in the heuristics are fulfilled, the similarity measure of the language is updated in the similarities array.

In some cases, it is used a Word2Vec with the heuristics, as shown in Annex 1. Word2vec is a group of related models used for learning vector representations of words, called word embeddings (Mikolov et al., 2013). We use it by generating a Word2Vec model from most common words per language list and calculating similarity over the input text to identify its language.

3.2.3 If multi-language input verification is needed

Steps to verify if the input text contains more than one language (line 3) are:

Check if the input text contains symbols in more than one alphabet. In this case, the input text may be multi-language. Giving the following data:

Let $\mathcal{A}$ be the similarities array

Let $\mathcal{H}\infty$ be the higher similarity measure in a

Let $\mathcal{H}\in$ be the second highest similarity measure in a

Let $\mathcal{C}$ be

$$H1 > \frac{H2}{3}$$

If $\mathcal{C}$ is true then the input text may be multi-language.

3.3 Normalize and generate and words 3-gram

Split the input text into groups of normalized words 3-grams. First, the same normalization process described in 3.1.2 is performed. Then, the normalized text is split into groups of words 3-Grams. For instance the normalized eh di kayo na nasa isang tent is split into the following words 3-grams: *[eh di kayo] [kayo na nasa] [na nasa isang] [nasa isang tent].*

3.4 For each word 3-gram: calculate similarities array

For each word 3-gram, a similarities array is calculated using the same process described in Section 3.2. Next, the language with highest similarity score is stored in a new array of languages with a counter initialized to 1. The counter is increased each time its correspondent language has the highest score. Array of languages example: *[[4, EN], [4, ES], [1, PT]].*

3.5 Generate probability pairs array

If the array generated in 3.5 contains English (EN), an unified probabilities array is generated. Each item of the array is calculated as follow:

Let $\mathcal{A}$ be the array of languages

Let $\mathcal{I}$ be an input text

Let $\mathcal{EN}$ be English

Let $\mathcal{O}$ be a language other than English

Let $\mathcal{EC}$ be $\mathcal{EN}$ counter in $\mathcal{A}$

Let $\mathcal{OC}$ be $\mathcal{O}$ counter in $\mathcal{A}$

Let $\mathcal{C}$ be the sum of all counters in $\mathcal{A}$

Let $\mathcal{S}$ be $\mathcal{EC} + \mathcal{OC}$

Let $\mathcal{P}(\mathcal{EN}, \mathcal{O})$ be the the probability of $\mathcal{I}$ being writing in $\mathcal{O}$ and $\mathcal{EN}$

$$P(e, o) = \frac{S}{C}$$

For instance, from the languages array presented in 3.5, the following probabilities array would be generated: *[[EN, ES, 0.88], [EN,PT, 0.55]*, which means 88% probability of the input text written in English and Spanish and 55% probability of the input text written in English and Portuguese.

3.6 If single probabilities array generation is needed

The conditions to generate the single probabilities array are:

- The input text does not contains more than one language (condition described in 3.3 is false)
- The array of languages (Section 3.5) doesn't contain English (*EN*)

```
[(EN, 0.2123522896495467), (TR, 0.202020480731943)
 (ES, 0.16166650453445985), (IT, 0.13663037149505872),
 (PT, 0.09845020238382388), (FR, 0.097383166685139),
 (AZ, 0.05053235237509718), (ID, 0.026847020833892152),
 (TL, 0.007138278092895213), (RU, 0.00356580654667366),
 (ZH-CHT, 0.0014803133534808746),(ZH-CHS,0.000834285225601832),
 (KA, 0.000743740835877281), (AR, 0.00035518725651069205),
 (FA, 0.0), (HI, 0.0)]
```

Figure 3: Probabilities array of the input *I love u 4 ever*. Each pair inside the vector contains (language code[3] , probability).

3.7 Generate single probabilities array

In line 8, the input text similarity array, generated in 3.2, is transformed in a probabilities array, i.e., an array that inform the likely percentage of the input text being in each language. Each item of the probabilities vector is calculated as follow:

Let $\mathcal{L}$ in $\mathcal{S}$ be a language in the set of languages $\mathcal{S}$
Let $\mathcal{I}$ be an input text
Let $\mathcal{S}$ be the similarity value for one language
Let $\mathcal{A}$ be the sum of all similarities
Let $\mathcal{P}(\mathcal{L})$ be the probability of $\mathcal{I}$ being written in $\mathcal{A}$

$$P(l) = \frac{S}{A}$$

For instance, the similarity in Figure 2 generates the single probabilities array in Figure 3, wherein the item *[EN, 0.2123522896495467]* describes the probability of the input text being written in English.

3.8 Implementation

The solution described in this paper is implemented as part of Meedan linguistic server[4] and it is named Alegre LangId. It was coded in Python and it is freely available. We used Gensim[5] [4] to generate the TF-IDF VSM (Section 3.1.2) and to calculate similarity (Section 3.2.1).

The model (Section 3.1.2) is available with the implementation works with the following languages: Arabic, Azerbaijani, English, Farsi, Georgian, Hindi, Italian, Portuguese, Tagalog, Traditional-Chinese, Simplified-Chinese, Turkish and Russian.

This model is not static. It was designed for being easily being changed (adding or removing languages, for instance). For that, it is only necessary to update the languages directories.

4 Evaluation

In this section, we compare Alegre LangId performance with Polyglot (Lui et al., 2014) and Langid.py (Pawar and Gawande, 2012). We have performed three evaluation rounds: 1st using a subset of TwitUser (Lui and Baldwin, 14) dataset, 2nd using a test set of monolingual tweets and 3rd using a test set of bilingual tweets. Table 2 summarizes the results of the three rounds for all the tested tools.

The formula used to calculate accuracy is:
Let $\mathcal{C}$ be the number of correct classifications.
Let $\mathcal{N}$ be the total number of messages.

$$accuracy = \frac{C}{N}$$

[4]`https://github.com/meedan/alegre`
[5]`https://radimrehurek.com/gensim`

System	TwitUser Subset	Monolingual Test Set	Bilingual Test Set
Alegre LangId	0.8265	0.85	0.35
Polyglot	0.2233	0.32	0.3
Langid.py	0.7901	0.83	N/A

Table 2: Accuracy of the 3 compared language identifiers system in the 3 rounds.

4.1 First Round

TwitUser (Lui and Baldwin, 14) dataset is a tweets dataset built using a mostly-automated method that leverages user identity. The complete dataset has 14178 messages across 65 languages. Here we report the results of Alegre LangID, Langid.py and Polyglot over a subset of 5309 messages. The subset consists of all tweets in the 13 languages from Alegre LangId model (Section 3.9).

Alegre Langid correctly classified 4388 tweets, resulting in accuracy of 0.8265. From the 921 incorrect classification, 226 were incorrectly classified as bilingual. Langid.py correctly classified 4195 tweets, resulting in accuracy of 0.7901. Polyglot correctly classified 1186 tweets, resulting in accuracy of 0.2233.

It is interesting to point that out Polyglot identified 853 messages as multilingual and Alegre LangId 226 messages. We assume that this difference occurs due Alegre LangId premisse that is better to correctly identify an unique language than identify an incorrect pair (Section3).

4.2 Second Round

In this round we executed Alegre LangID, Langid.py and Polyglot over a test set of 100 monolingual tweets. Alegre Langid classified 94 tweets as monolingual and 6 as bilingual. From the tweets classified as monolingual, 85 were correctly classified and 9 incorrectly classified. In that way, the system accuracy is 0.85. Polyglot classified 82 tweets as monolingual and 18 as multilingual. From the tweets classified as monolingual, 32 were correctly classified and 50 incorrectly classified. In that way, the system accuracy is 0.32. Langid.py only works with one language. So, 83 were correctly classified and 17 incorrectly classified. In that way, the system accuracy is 0.83.

If we compare Alegre LangId and Polyglot, the two multilingual systems, 0.94 of the tweets were correctly identified as monolingual by Alegre LangId against 0.82 from Polyglot. We believe that this result come from our reliability focus (Section 3), where we assume as requirement that that in case of bilingual posts, is more important to identify correctly one of the languages than identifying a pair containing a wrong language. Also, even being a bilingual system and incorrectly identifying 6 posts as bilingual, Alegre LangId performance was slightly better than Langid.py, which is designed for monolingual inputs and is reported to work well with tweets [8].

4.3 Third Round

In this round we executed Alegre LangID and Polyglot over a test set of 100 bilingual tweets.

In order to illustrate the test and its results, Table 3 shows examples of Alegre LangID and Polyglot work for the same input. Row 2 shows a case where Alegre LangID correctly identified the pair of languages and Polyglot identified only one language. In row 3 Alegre LangID any language and Polyglot correctly identified the pair. In row 4 both tools identified only one language, not the pair.

Polyglot classified 38 tweets as bilingual and 62 as monolingual. From the tweets classified as bilingual, 3 were correctly classified and 35 incorrectly classified, that is 0.789 of accuracy. From the tweets classified as monolingual, 18 were correctly classified and 44 incorrectly classified, that is 0.2903 of accuracy. To determine accuracy, we consider the tweets classified as bilingual as incorrect. In that way, the system accuracy is 0.32.

Alegre LangID classified 48 tweets as multilingual and 52 as monolingual. From the tweets classified as multilingual, 35 correctly and 13 incorrectly, that is 0.7291 of accuracy. From the tweets classified as monolingual, 51 correctly and 1 incorrectly, that is 0.9807 of accuracy. For strictly evaluating of the multilingual detection we need to consider the tweets classified as monolingual as incorrect. In that case, Polyglot accuracy is 0.3 and Alegre LangID 0.35.

Post	Alegre LangID	Polyglot
El Chavo 26 Book Class Set English Spanish Partido de Futbol Soccer Match New http://buy-books-online.info/bybk/snln/?query=http://rover.ebay.com/rover/1/711-53200-19255-0/1?ff3=2&toolid=10039&campid=5337797091&item=401069720492&vectorid=229466&lgeo=1	[['EN,ES', 1.0]]	"es": 1.0
Com 2 hematomas gigantes depois da partida de futebol de hoje/with 2 big bruises from today's soccer match =/	[]	"en": 0.34782608695652173, "pt": 0.65217391304347827
Mi amigo el much gundo Gryffin I lav jou you are best for me and frind #ULTRALIVE	[('EN', 0.7119265507394346), ('ES', 0.06630115012912295), ('PT', 0.0539414644743907),...]	"en": 1.0

Table 3: Alegre LangID and Polyglot results for 3 tweets.

Alegre LangId identifying as many tweets as monolingual was expected, due the reliability focus (Session 3). From that, it is clear that Alegre LangId need to improve the performance of bilingual identification. Meanwhile, Polyglot, identified even less tweets as multilingual. So, we consider that this is a challenge for both systems.

5 Conclusion

In this paper we presented language identification solution designed to deal with noisy and short texts. It works with one language or pairs of languages. In case of pairs, one of the languages is assumed to be English. The presented solution combines TF-IDF VSM similarity of character 3-grams with heuristics for single language identification and uses words 3-grams windows for bilingual identification.

We conducted three test rounds. One using a subset of TwitUser dataset [8] , one with 100 monolingual tweets and other with 100 bilingual tweets. Our evaluation shows that our solution outperforms Langid.py and Polyglot in accuracy in all test rounds. However, 52% of the bilingual tweets from the third round were identified as monolingual, a performance which requires improvement, but is still higher than Polyglot. As future work,

- Improve the performance of bilingual identification, increasing the number of correctly identified pairs.
- Work with any combination of languages, not just with English and other one.
- Work with any number of languages in each identification, increasing the two languages limit.
- Include more languages in the identification scope, updating the model available with Alegre LangId.
- Replace heuristic rules (Section 3.2.2) with better and learning from samples.
- Allow users to submit corrections, in a way that user corrects a detected language. This would update the model such that next time, this input and others like it are better classified.

References

Carter, S., Weerkamp, W. and Tsagkias, M. 2013. Microblog Language Identification: Overcoming the Limitations of Short, Unedited and Idiomatic Text. *Lang. Resour. Eval.*, 47(1):195–215.

Cavnar WB, Trenkle JM. 1994. *N-gram-based text categorization.* Ann Arbor MI , 48113(2):161-75.

Contractor, D., Faruquie, T.A. and Subramaniam, L.V. 2010. *Unsupervised cleansing of noisy text*. *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, :189–196.

Laboreiro, G., Bonjak, M., Sarmento, L., Rodrigues, EM, and Oliveira, E. 2013. Determining language variant in microblog messages. *roceedings of the 28th Annual ACM Symposium on Applied Computing 2013*, :902–907.

Lui, M. and Baldwin, T. 2011. Cross-domain Feature Selection for Language Identification. *Proceedings of the Fifth International Joint Conference on Natural Language Processing* :553–561.

Lui, M., Lau, J.H. and Baldwin, T. 2012. langid.py: An Off-the-shelf Language Identification Tool. *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics, Demo Session*

Lui, M. and Baldwin, T. 2014. Accurate Language Identification of Twitter Messages. *Proceedings of the 5th Workshop on Language Analysis for Social Media* :17–25.

Lui, M., Lau, J.H. and Baldwin, T. 2014. Automatic detection and language identification of multilingual documents. *Transactions of the Association for Computational Linguistics*, 2:27–40.

Manning, C., Raghavan, P., Schtze, H. 2008. *Introduction to Information Retrieval.* Cambridge University Press.

McNamee, P. 2005. *Language identi*

cation: A solved problem suitable for undergraduate instruction. J. Comput. Sci. Coll. , 20(3):94–101.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. 2013. Efficient Estimation of Word Representations in Vector Space *Proceedings of Workshop at ICLR*,

Chang, JC. and Lin, CC. 2014. Recurrent-Neural-Network for Language Detection on Twitter Code-Switching Corpus *arXiv preprint arXiv:1412.4314*

Pawar, P. Y., and S. H. Gawande. 2012. A comparative study on different types of approaches to text categorization. *International Journal of Machine Learning and Computing*, 2.4:423–426.

Rajaraman, A. and Ullman, J. D. 2011. Data Mining *Mining of Massive Datasets*, :1–17

Rehurek, R. and Sojka, P. 2010. *Software Framework for Topic Modelling with Large Corpora*. *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, :45–50.

Salton, G., Wong, A. and Yang C. S. 1975. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620.

Tromp, E. and Pechenizkiy, M. 2011. Graph-based n-gram language identification on short texts. *Proc. 20th Machine Learning conference of Belgium and The Netherlands*

Zampieri M. 2013. Using bag-of-words to distinguish similar languages: How efficient are they?. *Computational Intelligence and Informatics*, :37–41.

Zubiaga, A., San Vicente, I., Gamallo, P., Campos, J.R.P., Loinaz, I.A., Aranberri, N., Ezeiza, A. and Fresno-Fernndez, V. 2014. Overview of TweetLID: Tweet Language Identification. *SEPLN 2014*, :1–11.