

# Unsupervised Creation of Normalisation Dictionaries for Micro-Blogs in Arabic, French and English

Amal Htait<sup>1,2</sup>, Sébastien Fournier<sup>1,2</sup>, and Patrice Bellot<sup>1,2</sup>

<sup>1</sup> Aix Marseille Univ, Université de Toulon, CNRS, LIS, Marseille, France.  
{firstname.lastname}@lis-lab.fr

<sup>2</sup> Aix Marseille Univ, Avignon Université, CNRS, EHES, OpenEdition Center,  
Marseille, France  
{firstname.lastname}@openedition.org

**Abstract.** Text normalisation is a necessity to correct and make more sense of the micro-blogs messages, for information retrieval purposes. Unfortunately, tools and resources of text normalisation are rarely shared. In this paper, an approach is presented based on an unsupervised method for text normalisation using distributed representations of words, known also as "word embedding", applied on Arabic, French and English Languages. In addition, a tool will be supplied to create dictionaries for micro-blogs normalisation, in a form of pairs of misspelled word with its standard-form word, in the languages: Arabic, French and English. The tool will be available as open source<sup>3</sup> including the resources: word embedding's models (with vocabulary size of 9 million words for Arabic language model, 5 million words for English language model and 683 thousand words for French language model), and also three normalisation dictionaries of 10 thousand pairs in Arabic language, 3 thousand pairs in French language and 18 thousand pairs in English language. The evaluation of the tool shows an average in *Normalisation* success of 96% for English language, 89.5% for Arabic Language and 85% for French Language. Also, the results of using an English language normalisation dictionary with a sentiment analysis tool for micro-blog's messages, show an increase in f-measure from 58.15 to 59.56.

**Keywords:** normalisation, dictionaries, word embedding, micro-blogs, unsupervised, multilingual, arabic, french.

## 1 Introduction

Twitter and other micro-blogging services are considered as a source of large-volume real-time data, which make them highly attractive for information extraction and text mining. Unfortunately, the quality of micro-blogs' text, with

---

<sup>3</sup> <https://github.com/amalhtait/NormAFE>  
<https://github.com/OpenEdition/NormAFE>  
<http://amalhtait.com/tools.html>

the typos, misspellings, phonetic substitutions and ad hoc abbreviations creates huge obstacles in the way of text processing. Therefore, normalisation techniques are a necessity to correct and make more sense of the micro-blogs messages.

This work is inspired by Sridhar et al. [1], an unsupervised method for text normalisation using distributed representations of words, known also as "word embedding". The method was not applied on Arabic language, nor French. Also the resources of the previous work were never publicly shared. Therefore, in addition to this work, a tool will be supplied to create dictionaries for micro-blogs normalisation, in a form of pairs of misspelled word with its standard-form word, in the languages: Arabic, French and English. The tool will be available as open source<sup>3</sup> including: three word embedding's models, with vocabulary size of 9 million words for Arabic language model, 5 million words for English language model and 683 thousand words for French language model. And three normalisation dictionaries of 10 thousand pairs in Arabic language, 3 thousand pairs in French language and 18 thousand pairs in English language.

This paper is presented as below:

1. The word embedding's models training: strategy, parameters and datasets.
2. The exemplary list of standard-form words (spelled correctly) used as seeds for the creation of the dictionaries.
3. The method applied using the models and the seeds list (with their antonyms), to extract for each seed its list of misspelled words. Followed by a post-processing, or filtering, for more accurate results.
4. The evaluation of the tool and dictionaries.

## 2 Related Work

The primary approach in text normalisation was the noisy channel model [2], the approach aims to find  $\mathit{argmax}P(S|T)$  where the misspelled text is  $T$  and its corresponding standard form is  $S$ , and that's by computing  $\mathit{argmax}P(T|S)P(S)$ , in which  $P(S)$  is a language model and  $P(T|S)$  is an error model. For many applications, there was a considerable energy to improve both models, with a result of improvement in overall system accuracy. For example, some researchers worked on a new error model for spelling correction, based on generic string to string edits [3]. And others expanded the error model by analyzing a sample of texting forms to define frequent word formation processes in creative texting language [3]. The noisy channel model in text normalisation showed effectiveness, but its methods are based on the assumption that a token  $t_i \in T$  only depends on  $s_i \in S$ , ignoring the context around the token, which can cause ambiguity between words (e.g. *goood* was meant to be *good* or *God?*).

Statistical machine translation (SMT) has been also used as a method for text normalisation, by treating the misspelled text as the source language, and the standard form as the target language. Similar work is found on phrase-based

SMT model for text normalisation with bootstrapping the phrase alignment [5]. Unfortunately, SMT approaches tend to face a lack of training data.

Some researchers used speech recognition to solve text normalisation issue [6]. They converted the input text tokens into phonetic tokens, then restored them to words using phonetic dictionary. Others used a classifier to detect misspelled words, and generates correction possibilities based on morphophonemic similarity [7]. But these methods need large-scale of annotated training data, which limits their adaptability to new domains and languages.

To overcome the limitations of previously cited methods, a technique is applied to learn distributed representation of words (also called word embedding or continuous space representation of words), and to capture distributional similarity between words in a unsupervised manner. As a result, each word will be represented by a numeric vector of high-dimensionality, encoding many linguistic regularities and patterns, also syntactic and semantic word relationships. Due to this representation, words with semantic similarity are represented by similar vectors. Therefore, a misspelled word is represented by a similar vector as its standard-form word.

Sridhar et al. [1] were first to propose that method with a training dataset of 27356 English SMS phrases. His research was the base of several similar work in Portuguese [8], Turkish [9] and Chinese [10], but never in Arabic nor French. In addition, none of these work is open source, and they didn't share the word embedding models, nor the lexicons or dictionaries. Also, all their work was based on relatively small datasets. For example, Bertaglia's work [8] was focused on products reviews, that are slightly effected by the misspelling errors, the slang words and the typo errors, compared to the tweets, which leads to a much more effective work in micro-blogs' normalisation. Also Bertaglia's work [8] was based on a dataset of only 86 thousand products reviews and an unknown small amount of tweets in Portuguese. And like the rest of the previously cited researchers, the datasets and word embedding models were not publicly shared. This paper focuses on tweets as a dataset resource, with their richness in misspellings and slang words. As a language, it is not limited with one language, but it presents the work with three languages: Arabic, French and English. And as a dataset size, a large corpora of tweets is used to create the word embedding models: one billion tweets in English language, 238 million tweets in Arabic language and 48 million tweets in French language.

### 3 Word Embedding Training

Word embedding, or distributed representations of words in a vector space, are currently considered to be among a small number of successful applications of unsupervised learning. Also, they are capable of capturing lexical, semantic, syntactic, and contextual similarity between words. In the following subsections,

a discription of the word embedding’s models training dataset, strategy and parameters.

### 3.1 Training Dataset

The word embedding’s training datasets was extracted from the archived twitter streams<sup>4</sup>, which is a collection of JSON<sup>5</sup> format data from the general twitter stream, available for the purposes of research, history, testing and memory. This collection contains tweets in many languages what allowed the extraction of tweets in the three languages: Arabic , French and English. Randomly, we extracted files of archived twitter streams dated between 2012 and 2017.

A pre-processing is applied on the three corpora, to improve their usefulness:

1. The tweets’ corpora is tokenized.
2. The user names, hyperlinks and emoticons are replaced by *uuser*, *http* and *sentiment\_emoticon*.
3. Some characters and punctuations were removed.
4. And also, the duplicated tweets were eliminated.

As a result one billion tweets in English language were extracted, in addition to 238 million tweets in Arabic language and 48 million tweets in French language.

### 3.2 Strategy and Parameters

For the purpose of learning word embedding from the previously prepared corpora (which is raw text), we use Word2Vec [11].

Word2Vec is a widely used method in natural language processing for generating word embedding, and it has two training strategies:

1. Continuous Bag-of-Words(CBOW), in which the model is given a sequence of words with a missing one, and it attempts to predict this omitted word.
2. Skip-Gram, in which the model is given a word and it attempts to predict its neighboring words.

According to Mikolov et al. [11], Skip-Gram is more efficient in presenting infrequent words than CBOW. And since the purpose is to seek misspelled words, which are relatively infrequent words, therefore, the Skip-gram architecture is chosen to train the models.

To train word embedding and create the models, Gensim<sup>6</sup> framework for Python is used. And for the parameters, the models are trained with word representations of dimensionality 400, a context window of one and negative sampling for five iterations ( $k = 5$ ).

By applying the previously mentioned strategy to the datasets, three models were created with a vocabulary size of 9 million words for Arabic model, 5 million words for English model and 683 thousand words for French model.

<sup>4</sup> <https://archive.org/details/twitterstream>

<sup>5</sup> JavaScript Object Notation is an open-standard file format

<sup>6</sup> <https://radimrehurek.com/gensim/index.html>

## 4 Dictionaries/Resources Creation

Word embedding models allow to capture the nearest neighbors of a certain word  $X$  using the cosine distance between the dimensional vector of that word  $X$  and the dimensional vector of each word in the model. The example, in Fig.1, shows that most of the nearest neighbors of the word *alors* (*then* in French) are not real French words but the misspellings of the word *alors*, such as: *alrs*, *allrs*, *alr*, *alord*, *alirs*, *allors* and *alorq*, in addition to some other words, such as: *sachant* (*knowing* in French) and *parce* (*because* in French). And another example, in Arabic language, the word بارد (*cold* in Arabic) has the nearest neighbors as its misspellings and not real Arabic words, like: باررد , بآرد , بارذ , بارذ and باردد.

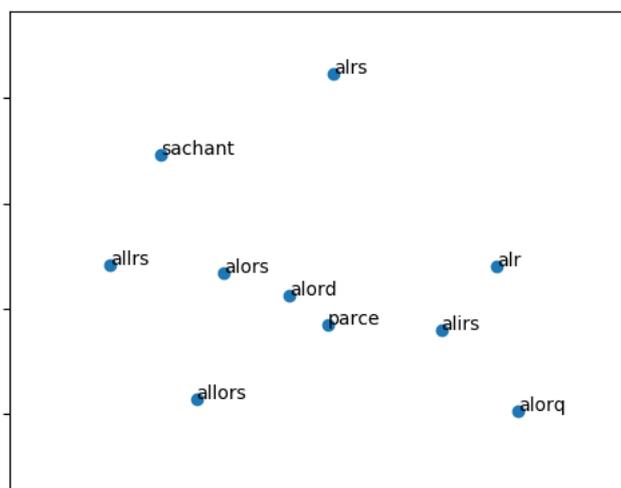


Fig. 1: The French word *alors* (*then*) with its nearest neighbors.

Therefore, as a first step in the dictionaries creation, list of standard-form words is needed to capture their misspelled neighbors. In the following subsections, the discription of the collected standard-form seed-words lists and the procedure followed for dictionaries creation.

### 4.1 Lists of Standard-form Seed-words

Most common words in Arabic, French and English were collected for the purpose of creating the standard-form seed-words lists, as below:

1. For the Arabic language, the list of 1000 most common Arabic words<sup>7</sup>, the list of 480 Arabic stop-words<sup>8</sup> and the list of 230 Arabic words highly positive

<sup>7</sup> <http://1000mostcommonwords.com/tag/arabic-words/>

<sup>8</sup> <https://github.com/stopwords-iso/stopwords-ar/blob/master/stopwords-ar.txt>

- or negative [12] were combined. A list of 1492 Arabic words is the result of the previous lists combination, after removing duplications. The collected list is Standard Arabic words, but dictionaries of any Arabic language dialect can be created (using the shared word embedding Arabic model and tool).
2. For the French language, the most common French adjectives and nouns<sup>9</sup> and the list of 689 French stop-word<sup>10</sup> were combined. A list of 986 French words is the result of the previous lists combination, after removing duplications.
  3. For the English language, the 3429 English words from Oxford dictionary<sup>11</sup> which English stop-words are included, the 500 most frequently used words on twitter<sup>12</sup> and a list of 90 frequent sentiment words<sup>13</sup> in tweets [13] were combined. A list of 3501 English words is the result of the previous lists combination, after removing duplications.

To note that the tool and the word embedding models will be publicly available, and more lists of standard-form seed-words (in Arabic, French and English languages) can be added by users to enrich the current dictionaries.

## 4.2 Procedure of Dictionaries Creation

To create a list of possible misspellings (noisy versions) for every standard-form word, two steps are applied:

1. First, to determine the similarity between two word embedding, the measure of cosine distance is used between the vectors of standard-form words and the vectors of every other word in the word embedding model. The class *most\_similar* (based on cosine distance measure), of Gensim framework, is used to give a list of 10 most similar words to the standard-form word. To refine the results, *most\_similar* class is used with the antonym of the standard-form word (by Natural Language Toolkit<sup>14</sup>) as in the following example:

---

```
1 model.most_similar(positive=['active'], negative=['inactive'])
```

---

- The antonyms exclusion eliminates the possibility of extracting the word *inactive* as a similar word to the word *active*, since the list of the 5 most similar words of *active* is: *inactive*, *acitve*, *avtive*, *actuve* and *innactive*.
2. Second, Python's class *SequenceMatcher* is applied to compare the previously collected similar words to the standard-form word, for the purpose of eliminating the errors in the list of possible misspellings (e.g. for similar words to the word *good* -> *good*, *goid*, *great* and *goos*, the word *great* will

<sup>9</sup> <http://www.encyclopedie-incomplete.com/?Les-600-Mots-Francais-Les-Plus>

<sup>10</sup> <https://github.com/stopwords-iso/stopwords-fr>

<sup>11</sup> <https://www.oxfordlearnersdictionaries.com/wordlist/english/oxford3000/>

<sup>12</sup> <http://techland.time.com/2009/06/08/the-500-most-frequently-used-words-on-twitter/>

<sup>13</sup> Sentiment words are words highly positive (e.g. *Happy*) or negative (e.g. *Sad*)

<sup>14</sup> <http://www.nltk.org/>

be eliminated in this step). The idea of this method is to find the longest contiguous matching subsequence that contains no junk elements (or different elements). The same is then applied recursively to the pieces of the sequences to the left and to the right of the matching subsequence. This method tend to give matches that “look right” to people<sup>15</sup>.

As a result for that procedure, three dictionaries were extracted in the form of pairs of misspelled words with their standard-form word:

1. Arabic language dictionary with 10 thousand pairs.
2. French language dictionary with 3 thousand pairs.
3. English language dictionary with 18 thousand pairs.

## 5 Evaluation

### 5.1 Evaluation of Dictionaries’ Content

To evaluate the tool by dictionary’s content, a manual annotation is applied by checking the correct pairing between the misspelled words and their assigned standard-form words, and the annotation differentiate between two types of evaluation: *Correction* (e.g. *graet* and *great*) and *Normalisation* which includes the correction and the lemmatization (e.g. *shows* and *show*). Table 1 shows an example of the method of annotation, where the check-mark is a right correction or normalisation, and the x-mark is a wrong one.

Table 1: An example of dictionaries annotation, where three examples from each language is selected (English, French and Arabic), and where the check-mark is a right correction or normalisation, and the x-mark is a wrong one.

<i>Misspelled</i>	<i>Standard – word</i>	<i>Correction</i>	<i>Normalisation</i>
gladd	glad	✓	✓
hates	hate	✗	✓
horrific	horrible	✗	✗
aiiiiime (loooooove)	aime (love)	✓	✓
decevera (will disappoint)	decevoir (disappoint)	✗	✓
deballer (unpack)	deprimer (depress)	✗	✗
ممتاز ( misspelled excellent)	ممتاز (excellent)	✓	✓
اكرهه (hate him)	اكره (hate)	✗	✓
اهبل (dump)	غبي (stupid)	✗	✗

<sup>15</sup> <https://docs.python.org/2/library/diffib.html>

To have as much similar evaluation as possible between the three languages, a sample of 50 standard-form words highly positive or negative is selected, for each language, since sentiment words are usually used in micro-blogs at same frequencies in most languages.

Cosine similarity measure allows to find a list of most similar words of the standard-form word. And the class *most\_similar* of Gensim framework, which is based on cosine similarity measure, has by default 10 as a size of that list. Changing the size of that list can lead to creating larger dictionaries, like in the below example:

1. The list of 5 similar words of *good* is: good, goid, goood, gooooo, gud.
2. The list of 15 similar words of *good* is: good, goid, goood, gooooo, gud, goooooo, goos, gpod, great, gopd, giod, gooooooo, cargood, gooooooo, g00d.

The second list is larger and richer in words, but it includes unwanted words like *great* and *cargood* (since these words are not corrections of the word *good*). Therefore, and as part of the evaluation, the number of similar words extracted is added as a parameter, and the evaluation is applied by varying its value as 5, 25, 50 and 100.

First, a briefing of the evaluation results are presented below:

1. For English language, an average of 96% in *Normalisation* success, and of 86% in *Correction* success.
2. For Arabic language, an average of 89.5% in *Normalisation* success, and of 83.7% in *Correction* success.
3. For French language, an average of 85% in *Normalisation* success, and of 73.6% in *Correction* success.

Then, the results of the evaluation are presented with more details in the graphs of Fig.2, consequentially from left to right in English, French and Arabic languages, where the percentage of successful *Correction* is the line in blue and the percentage of successful *Normalisation* is the line in red, both calculated relatively to the variation of most similar words number (as 5, 25, 50 and 100) and the variation of the dictionaries' size (the bars in grey). To note that the created dictionaries for the Arabic language reached the size of 2053 pairs when selecting 100 most similar words, for the French language the size of 500 pairs, and for the English language the size of 2776 pairs.

The results in Fig.2 shows that the percentage of successful *Normalisation* is always higher than the percentage of successful *Correction*. Also, for English (left graph) and French (middle graph) languages, an increase in the percentage of successful *Correction* and *Normalisation* appears when the number of similar words extracted is between 5 and 25, followed by a continuous decrease. And for Arabic language (right graph), a sharp decrease with the percentage of successful *Correction* and *Normalisation* is observed, with the increase of similar words extracted number.

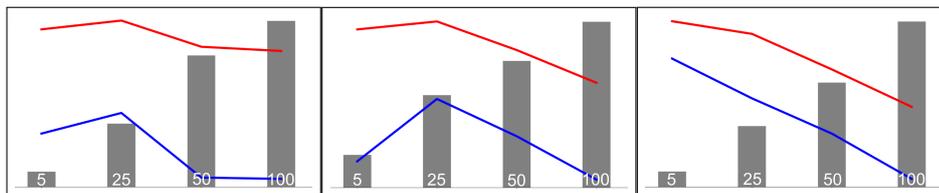


Fig. 2: The percentage of successful *Correction* (blue) and *Normalisation* (red) in the test dictionaries (dictionaries created based on 50 sentiment standard-words), both calculated relatively to the variation of most similar words number (as 5, 25, 50 and 100) and the variation of the dictionaries' size (the bars in grey), consequentially from left to right, in English, French and Arabic languages.

## 5.2 English Dictionary Evaluation with Sentiment Analysis Tool

The evaluation of the dictionaries by their content, in the previous section, shows promising results. But to prove the usefulness of the dictionaries, the effect of the English language normalisation dictionary is evaluated by predicting sentiment polarity (positive, negative or neutral) in micro-blogs' messages.

Echo<sup>16</sup>, an open source software for sentiment analysis based on supervised machine learning algorithm, is used as a test tool. And as training and testing datasets, SemEval2014's datasets [14] are used: for training, the annotated (by sentiment polarity) training dataset of almost 10000 Tweets, and for testing the 1000 Live-Journal from 2014, the 2000 SMS from 2013 and the 3800 Tweets from 2013.

The results<sup>17</sup> of Echo, predicting sentiment polarity of the testing data, are presented in Table 2. The first row is the baseline, where Echo runs without normalisation. Then, for the rest of the rows, the normalisation was applied using four dictionaries, all based on the same list of 50 English sentiment standard-words, but differ in the number of most similar words chosen at the level of dictionary creation (5, 25, 50 or 100), and as a result, these dictionaries differ in their size (since the size of a dictionary increases with the increasing number of "most similar words"). The results, in Table 2, show an increase in the capability of Echo to successfully predict sentiment polarity in micro-blogs's messages, using the normalisation dictionaries. Also, they show that Echo achieves better results when increasing the dictionary size, and best results in this evaluation is achieved with dictionary size equals to 2776 pairs.

<sup>16</sup> <https://github.com/OpenEdition/echo>

<sup>17</sup> The results are displayed with the f-measure value, a measure of a test's accuracy [15].

Table 2: Results of Echo with SemEval2014’s data [14], with a baseline of no normalisation, then with a normalisation applied using four dictionaries that differ in the number of most similar words and in their size.

<i>Echo</i>	<i>#SimilarWords</i>	<i>DictSize</i>	<i>LiveJournal2014</i>	<i>SMS2013</i>	<i>Twitter2013</i>
baseline	-	-	58.15	55.95	55.64
+Dict_1	5	371	58.50	55.97	55.94
+Dict_2	25	1449	58.67	56.16	56.16
+Dict_3	50	2337	58.90	56.42	56.21
+Dict_4	100	2776	<b>59.56</b>	<b>56.61</b>	<b>56.22</b>

## 6 Conclusion

This paper presents an approach based on an unsupervised method for text normalisation using word embedding, applied on Arabic, French and English languages. In addition, a tool will be supplied to create dictionaries for micro-blogs normalisation, and will be available as open source including the resources: three word embedding models, and three normalisation dictionaries, for the three languages: Arabic, French and English. The evaluation of the tool shows an average in *Normalisation* success of 96% for English language, 89.5% for Arabic Language and 85% for French Language. Also the results of using an English language normalisation dictionary on a sentiment analysis tool for micro-blog’s messages, show an increase in the tool’s ability to predict the sentiment polarity of the messages.

The evaluations’ results in Sect. 5.1 show that while the dictionary’s size increases, the percentage of *Normalisation* and *Correction* success decreases. But, on the other hand, and based on the evaluation in Sect. 5.2, the effectiveness of the dictionary (in sentiment analysis) increases with its size, independently from the percentage of success in *Normalisation* and *Correction*.

Finally, this work can be a resource for many domains in Natural Language Processing. For example, by observing the Arabic language dictionary, many pairs of dialect word with its standard-form word were found, some examples are in Table 3. Also, in the creation process of the word embedding models, a large number of emoticons and emojis (expressing sentiment, like ☺) were replaced by expressions: *positive\_emoticon*, *negative\_emoticon* and *neutral\_emoticon*, for the purpose of a futur use of these models in sentiment analysis tasks.

Table 3: An example of Arabic language pairs of dialect word with its standard-form word in the normalisation dictionary.

<i>DialectWord - DialectSource - StandardWord</i>		
عيب	Egypt Arabic	غبي (Stupid)
ضايق	Gulf Arabic	ضايقك (bothered you)

**Acknowledgments** This work has been supported by the French State, managed by the National Research Agency under the «Investissements d’avenir» program under the EquipEx DILOH projects (ANR-11-EQPX-0013)

## References

1. Sridhar, V. K. R. (2015) Unsupervised text normalization using distributed representations of words and phrases. In Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing, pp. 8-16.
2. Shannon, C. E. (1948) A Mathematical Theory of Communication. The Bell System Technical Journal, 27(3):379-423.
3. Brill, E. , Moore, R. C. (2000) An improved error model for noisy channel spelling correction. In : Proceedings of the 38th Annual Meeting on Association for Computational Linguistics p. 286-293.
4. Cook, P. , Stevenson, S. : An unsupervised model for text message normalization. In : Proceedings of the workshop on computational approaches to linguistic creativity. Association for Computational Linguistics, p. 71-78.
5. Aw, A., Zhang, M., Xiao, J., et al. (2006) A phrase-based statistical model for SMS text normalization. In : Proceedings of the COLING/ACL on Main conference poster sessions. Association for Computational Linguistics p. 33-40.
6. Kobus, C., Yvon, F., Damnati, G. (2008) Transcrire les SMS comme on reconnaît la parole. In : Actes de la Conférence sur le Traitement Automatique des Langues (TALN’08), p. 128-138.
7. Han, B. , Baldwin, T. (2011) Lexical normalisation of short text messages: Makn sens a# twitter. In : Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1. Association for Computational Linguistics, p. 368-378.
8. Bertaglia, T. F. C., Nunes, M. d. G. V. (2016) Exploring Word Embeddings for Unsupervised Textual User-Generated Content. Normalization. Proc. 26th Int’l Conf. Computational Linguistics (COLING 16), pp. 112-120.
9. Eryiğit, G., Torunoğlu-Selamet, D. (2017) Social media text normalization for Turkish. Natural Language Engineering p. 1-41.
10. Yan, X. , Li, Y., Fan, W. (2017) Identifying domain relevant user generated content through noise reduction: a test in a Chinese stock discussion forum. Information Discovery and Delivery, 45(4), pp.181-193.

11. Mikolov, T., Chen, K., Corrado, G., et al. (2013) Efficient estimation of word representations in vector space. In ICLR Workshop Papers.
12. Salameh, M., Mohammad, S., Kiritchenko, S. (2015) Sentiment after Translation: A Case-Study on Arabic Social Media Posts. In Proceedings of the 2015 conference of the North American chapter of the association for computational linguistics: Human language technologies (pp. 767-777).
13. Htait, A., Fournier, S., Bellot, P. (2017) LSIS at SemEval-2017 Task 4: Using Adapted Sentiment Similarity Seed Words For English and Arabic Tweet Polarity Classification. In Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017) (pp. 718-722).
14. Rosenthal, S., Nakov, P., Ritter, A., Stoyanov, V. (2014) SemEval-2014 Task 9: Sentiment analysis in Twitter. In Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014), pages 73–80, Dublin, IE.
15. Powers, D. M. (2011) Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation.