International Journal of Computer Applications (0975 – 8887)
Volume 178 – No.3, November 2017

Classification of Tweets based on Emotions using Word Embedding and Random Forest Classifiers

Parth Vora
K.J Somaiya College of Engineering, Mumbai

Mansi Khara
K.J Somaiya College of Engineering, Mumbai

Kavita Kelkar
K.J Somaiya College of Engineering, Mumbai

ABSTRACT
With the large-scale penetration of social media into our daily lives, it has become a platform for individuals to share and express their views, feelings, opinions, and thoughts. Identifying emotions has many applications ranging from personalized marketing to behavior study. Individuals express their feelings in a language that is frequently accompanied by ambiguity and figure of speech, which makes it difficult even for humans to comprehend. In this paper, we propose a new approach to classify text into emotion categories. We use Twitter data as labeled input, this data is labeled using hashtags and addresses features like emoticons, emoji, apostrophes, Twitter slang and spelling variations which are a part of informal language on social media. Our model uses word vectors generated by architecture like Word2vec, Glove, and Fasttext to generate word representations of the text. We then investigate the utility of these models on random forest classifier. Ultimately we compare the results to find the best model for text classification based on emotions. We achieve an overall 91% precision for four emotional classes on a mined dataset of more than 100,000 tweets. This is a very useful tool to understand human behavior and a natural step beyond the positive/negative polarity.

General Terms
Machine Learning, Text Classification, Word Vectors, Natural Language Processing

Keywords
Keywords Word vectors, random forests, Word2vec, Glove, emotion, text classification

1. INTRODUCTION
Advancement of smartphone technology and widespread availability of the internet, has changed people’s lifestyles. This has given rise to the single most influential factor in our life – the social media. Impacts of social media on society as a whole is just astounding. It has made sharing information so easy and hassle-free. From news to personal incidents, it takes seconds to spread across the globe. It can be said that the social media is the reflection of a society. Across continents and communities, social media has a very strong footprint. We choose Twitter for our classification specifically because Twitter is a very accurate account of emotions. With any event occurring around the world, tweets are the first thing that start floating on the internet. By limiting its characters to only 140, it is a perfect candidate for emotion classification because of how concisely and briefly the information is expressed. Also, the use of hashtags makes it easier to automatically label tweets. And these tweets provide us with a rich and diverse collection of emotions. They include a multitude of emotions ranging from fear, happiness, sadness, anger, love, disgust, surprise, excitement, etc.

The rest of the paper is organized as follows, section 2 gives the readers an idea about our motivation to create this model. Section 3 introduces the concept of word vectors, about the vectors we are going to use and why do we use word vectors. Section 4 addresses the challenges and how our model will overcome them. In section 5, the actual implementation of the model is explained and in section 6 we discuss the results obtained during experimentation. Section 7 concludes our paper and in section 8 we put forth our ideas for the future.

2. MOTIVATION
In some cases, the two classes of emotions (positive and negative) are not adequate to comprehend the nuances of the underlying tone of the sentence. In this paper, we will explore a model that efficiently and effectively classifies text into emotion categories. This model can be used for marketing purposes, community study, human behavior study, to target individuals to track their mental health, track customer emotion over time and much more.

Emotions drive users to purchase. Brands and companies realize this potential and are shifting to more emotionally connecting advertisements. These advertisements can be targeted towards a particular set of customers to get the most out of advertising revenues. They can also be used to gauge the success or failure of a product or a service by analyzing emotions. The stock market also depends on variations of mood across the public and this model will help with stock market predictions [1]. Demographic well-being and quality of life across various cultures and areas can be studied by analyzing tweets of that area. This normally would be done by tedious and time consuming door-to-door surveys and questionnaires. Our method will eliminate such manual effort and make it easier to draw conclusions. Mental health care experts and counselors can use this classification to keep track of patient’s mental health and effectively combat issues like depression and chronic anxiety.

Although some work has been done to classify text, most of it is focused on sentiment analysis.

3. WORD VECTORS: BASIC CONCEPT
Sometimes a single word with the same pronunciation and even the same spelling can have multiple meanings. Like the word ‘apple’ is a company name as well as a ‘fruit’. A solution to these problems is word embedding. When it comes to humans, we can tackle such ambiguities very intuitively but humans cannot process millions of sentences and documents that are generated every single day. So to make computers

perform this task effectively and scale tasks like classification, clustering on textual data we need to be able to convert textual information which is generally in the form of strings to numerical representations. Word vectors help to convert text to numbers. Machine learning algorithms are not capable of dealing with plaintext in their raw form, they require numbers for any sort of job. Word vectors extract knowledge from large text corpuses and build models which have words effectively represented as continuous vectors of numbers. Applications of which include language translation, google search optimization, text classification etc.

Word vector models generally map a word using a dictionary to a vector. Consider this example, "Word vectors are words represented as numbers". Words in this sentence are – 'word', 'vectors', 'are', 'words', 'represented', 'as', and 'numbers'. A dictionary is a list of all unique words in the text corpus. The dictionary for this sentence would look something like ('word', 'vector', 'represented', 'numbers'). For simplicity, consider the one-hot encoded form. This process uses the binary values to mark the presence and absence of words. The number 1 is used to mark the presence and the number zero to mark the absence. Consequently, for the word 'numbers' the vector generated in this format, based on dictionary defined above would be [0,0,0,1]. There is a one in the fourth position because the word 'numbers' are present in that position and similarly the vector for the word 'vector' would be [0,1,0,0]. This is just a simple illustration of how word vectors are created from text corpuses. There are different types of word embeddings which could be classified broadly into Frequency based Embedding and Prediction based Embedding. Frequency-based methods were limited in their ability to effectively generate word representations. Mikolov et al. introduced Word2vec, which was prediction based embedding in the sense that they provided probabilities [2]. This model was capable of word similarities and analogies. For example, in the Google's pre-trained model if we add the vectors of the word 'man' to 'woman' and then subtract the vector of word 'king' we get a vector which is closest to the vector of the word 'queen'. Word embedding use shallow neural networks that map words to words. These shallow neural networks learn weights which act as word vectors. Basic functioning includes taking a text corpus as input and subsequent vectors are generated as output. It learns a vocabulary and its consequent word vector through unsupervised learning. In simple terms, a word vector model can be seen as a matrix where each column is a feature and each row is a distinct word in the vocabulary, if you have 10,000 unique words with 400 features, the model will be a 10,000 * 400 sized matrix.

Word vectors is a combination of two techniques – Skip-gram model and Continuous bag of words model. While the former, predicts the context given the word, the latter predicts the word given the context. We will use the Skip-gram model as it can capture the two semantics of the same word. For example, it will have two vectors for the word 'orange'. One for the color orange and other for the fruit. Skip-gram model with negative sub-sampling performs far superior to other methods [2]. And these are the models that will be used to create vectors.

This paragraph briefly explains the three types of word-embedding models used in the architecture. Word2vec by Mikolov et al. was published in 2013 [2]. It was one of the very first models to learn word representations from trillions of words with relatively low computational costs. It has significantly outperformed various n-gram models [3,4,5]. Later after a year, Glove was released by natural language processing lab at Stanford [6]. Glove stands for global vectors for word representations. It was an improvement over Word2vec as it trains on global co-occurrence counts instead of separate local context windows in Word2vec. Lastly, the Fasttext library was created by the Research Team at Facebook for classification and learning of word representations [7,8]. Above two mentioned models i.e. Word2vec and Glove, treat words as smallest atomic units. However, Fasttext uses a different approach where it treats each individual word as being made of n-gram characters. For example, 'happy' is made of 'h', 'ha', 'hap', 'happ' and 'happy'. Here n could range from one individual character to the length of the entire word. This renders it more powerful than other two models as it can effectively handle rare words which are not present in the dictionary. For example, the word 'marvellouslyfantastic' would have word vector that is close to vectors of marvelous and fantastic. This kind of representations is useful especially when there are a lot of rare words in the text corpus. Like a text corpus with different names of companies.

4. PROPOSED METHODOLOGY

This section presents a model which uses unsupervised learning and shallow neural networks to learn word representations and ensemble classifiers to classify these vectors that are aggregated over each tweet, into emotion category. The solution is proposed to address the following challenges.

1) Automatic Class labeling

In most cases, emotion class labeling is done with human annotators. However as studied by M Hasan et al. hashtags, which are keywords preceded by the hash symbol can be used to collect labeled data [9]. Hashtags are used to tag text to make it more accessible or to label it with a particular theme. These hashtags were originally created to make content more search friendly - by using these keywords as search parameters. Usage of hashtags is very common and almost 15% of tweets include at least one hashtag [9]. Hence, hashtags are used as labels for collecting and labeling tweets.

2) Handling casual Twitter language, emojis, and emoticons

The language used over social media in most cases is informal. Given the 140-character limit on tweets, they are almost always replete with apostrophes, slang words, spelling mistakes, improper grammar, use of rare words, emoji (graphics used to describe objects and feelings) and emoticons (punctuations used to express facial gestures). This paper employs different pre-processing techniques which include the use of regular expressions, removal of stop words and spell correction to make tweets uniform and easy to translate into vectors leaving no information untouched.

3) Capturing semantic and syntactic information

In text classification, semantic and syntactic information plays a huge role. We use relatively high dimensional vectors along with a window size of 10 to ensure that words that appear in context with each other are represented accurately in the word vectors. Given the small character count of each tweet, such measures effectively capture all the relevant information. For example, in the tweet "#happy finally a long holiday after a tiring week", word vectors are able to relate "holiday" and "happy" in context of each other.

In brief, this model makes the following contributions:

- Design and implement a model that effectively classifies text into emotion category.
- To build word vectors from Twitter corpuses and aggregate them over tweets to represent the tweet as a feature set of n-dimensions.
- And, to finally train a classifier over the extracted feature set to create a model that detects emotion in tweets with the highest accuracy.

5. IMPLEMENTATION

Emotion recognition in tweets can be simply considered a classification problem of text according to pre-labeled emotions. In this case, it requires collecting tweets, labeling them, cleaning the tweets to extract most information possible, generating word vector models from tweet data corpuses and then ultimately training the ensemble classifiers to classify tweets. Figure 1 shows the architecture of the model. Steps are as follows:

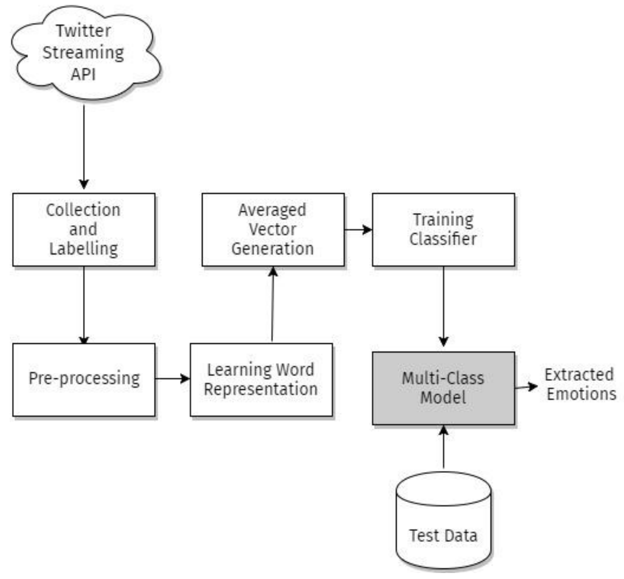

Fig 1: Architecture of the model

5.1 Collection and Labelling of Tweets

Availability of data is the primary requirement to train any machine learning algorithm. For a classification problem, one needs a proper label for each entry in the dataset. This experiment requires data labeled on four emotions – happiness, sadness, anger, and surprise. One approach to this problem would be to use manual annotations, where each tweet is analyzed by a person and then labeled. This is very time consuming and ineffective for a relatively large dataset. A possible solution could be to crowdsourcing and paying groups of people to do the task. Again, this is not an ideal solution as it is costly, slow and does not maintain consistency throughout the datasets because of subjective nature of how people interpret tweets. A novel and effective solution were offered by M Hasan et al. [9]. They used hashtags as keywords for labeling tweets. This paper adopts this approach to collect tweets for each category. For each emotion category, a set of hashtag keywords is shortlisted which are most likely to appear in those tweets and then use them with the Twitter API to collect tweets. Some examples are shown in Table 1.

Table 1. Examples of hashtags used to mine tweets

Category	Hashtags
Happy	#happy, #veryhappy, #awesome, #amazing, #blessed, #delighted
Sad	#bored, #sad, #depressed, #depress, #lonely, #rip, #missyou, #tragedy, #miserable, #hopeless, #unhappy
Angry	#angry, #furious, #bothered, #annoying, #mad, #irritated, #annoyed, #infuriated, #enraged, #fuming, #raging
Surprise	#surprised, #shocked, #confused, #baffled, #astonished, #startled, #flabbergasted, #disbelief, #numb, #shaken, #stunned

Common English words like 'and', 'the', 'a', etc. are used to collect large textual data to train word vector models. Since these are a common part of any sentence in English, most of the tweets mined would be in English. This reduces tweets that are eliminated during preprocessing.

5.2 Pre-processing Text

Pre-processing is a very important step in producing text that is free from noise and hence does not create problems for the word representation learning models. Language over social media is clouded by a lot of shorthand notations and spelling errors. This architecture processes the raw tweet data through a variety of steps to get the most features out of them. Following is the sequence of steps:

5.2.1 Keeping only English Tweets

The architecture is restricted only to the English domain. Tweets mined with a particular hashtag as search query at times include tweets in other languages. This language is encoded in a two-letter form as 'en' for English in the tweet JSON object that is mined. Only those tweets which have 'en' in their language field are kept. This helps remove any ambiguity introduced by non-English words.

5.2.2 Removing duplicates
Many times a tweet is mined multiple numbers of times. This is because people retweet. This can be the case when someone influential on Twitter posts something, and people simply repost their tweets. These repeated tweets need to be eliminated to ensure that word representations are not biased towards words that are repeated because of duplicate tweets.

5.2.3 Removing URL's
Tweets often contain links to other websites. This is the case when these websites contain content, referencing which the tweet was posted. These are just unwanted noise and contribute in no positive way to classification and hence must be removed. This architecture uses regular expressions to remove any URL that is present in tweets.

5.2.4 Removing reserved word
Retweets are often preceded by a reserved keyword "RT" and this adds unnecessary noise to the textual data and must be filtered out using regular expressions.

5.2.5 Replacing mentions
Twitter allows users to mention other users in their tweets. Like "@carljones very happy to hear your story!", includes a mention to the user with handle "carljones". These mentions need to be handled but they do have semantic meaning and contribute to the structure of the sentence as subjects. But to make the text more uniform, we replace such mentions with the text "Username" so the ambiguity is removed and the structure is maintained.

5.2.6 Remove hashtags
Sometimes people use normal words preceded by hashtags in sentences. For example, "Finally time to eat some #cake!". These sentences when tokenized are broken down along with the hashtag, and the words "cake" and "#cake" have different word representations. One must remove the hashtag punctuation symbol and interpret it as a plain word. This again is done by using regular expressions.

5.2.7 Replace Apostrophes
Because of the limited character count on Twitter, people usually sought to use apostrophes extensively. On tokenizing words like "we're" are taken as a single entity and hence these apostrophe words need to be uncoupled and explicitly replaced. These are replaced using regular expressions. Some examples of which are shown in Table 2.

Table 2. Examples of apostrophe replacements

Before Pre-processing	After Pre-processing
isn't	is not
he's	he is
we'll	we will
I've	I have

5.2.8 Replace Emoticons
When it comes to expressing emotions on Twitter, emoticons are a huge sign of such expression. Emoticons are combinations of punctuations arranged to express a facial expression. They cannot be interpreted by word representation learning models. M Hasan et al. used this as emotion features [10]. This architecture adopts the same approach and replaces emoticons with emotion classes they portray. Some of which are shown in Table 3.

Table 3. Class-wise emoticons

Class	Emoticons
Happy	:) :-) :-] :] :p :-p :D :-D :-> :> =) ;) ;-) ;^) ;-D
Sad	:(:-(=(:^(;'-(:< :-< :'(
Angry	>:-(>:(>:S x-@ ;@ :/
Surprise	:O :-O O_O :$

5.2.9 Replace Emoji
A picture can speak a thousand words. Emoji are small graphics that are extensively used on social media to depict a variety of things. From a facial expression to some food item, there are hundreds of emojis each portraying a different idea or an object. F Barbieri et al. developed a vector skip-gram model for emoji [11]. Tweets that are mined are in UTF-8 encoding. These emoji also appear in the text in the form of these encodings. We use Unicode Consortium's emoji definitions and replace these UTF-8 codes with their textual meaning. This will help capture more details when creating word vectors as tweets become more verbose. A few emojis and their interpretations are shown in Table 4.

Table 4. Emoji Unicode and short names

Emoji	Unicode	Short Name
😁	U0001F601	Smiling face
🤔	U0001F914	Thinking face
😲	U0001F632	Astonished face
😡	U0001F621	Angry face
❤️	U00002764	Love

5.2.10 Handling misspelling and repeated characters
Tweets are filled with misspellings like - “todau” is used instead of “today” and “saddddd” is used instead of “sad”. These arise due to small cellular keyboards and the repeated characters are just a way of showing stronger emotion. This architecture replaces characters that repeat more than twice with a single occurrence. For example, the word “happyyyyy” will be replaced by “happy” and the word “tedddyyy” with “tedy”. Peter Norvig’s spell corrector is used on these replaced words to get the correct spelling of the word [12]. In the above example, the word “happy” remains the same, however, the word “tedy” becomes “teddy”. This ensures that the implied meaning, in spite of incorrect spelling, is captured.

5.2.11 Handling punctuations
Emotion-laden text is always coupled with exclamation marks and question marks. Exclamation marks, often are used to display strong emotion and feeling. They are used to make the statements more impactful. Question marks when used atomically are just an indication of questions, however, when used with exclamation mark they indicate surprise and even sometimes are used interchangeably with exclamation marks. These punctuations are replaced with their literal meaning using regular expressions. Exclamation marks get replaced by “exclamation” and question marks get replaces by “question”. This helps in learning word representations and preserves the semantic meaning of the text.

5.2.12 Lowercase
The final step in pre-processing is to convert all the text to lowercase. This ensures that the words that are in capitals, or even with the first letter capitalized are interpreted like their lowercase counterparts. Basically, it ensures, the word vector of each word is not affected by their case. This helps maintain uniformity throughout.

5.3 Learning Word Representations
There are two ways to create word representations, one is to custom create a model based on text corpus as input. Second is to use pre-trained models which have been trained on very large text corpus which have been derived from various sources like news, Twitter, Wikipedia etc. Advantage of using a custom trained model is that they learn words in the domain of application. For example, if one trains a model based on text corpus from Twitter, words will be learned in the domain of Twitter. However, a downside to this is that it is very time consuming to mine enough data and then train vectors on it. This paper discusses the use of both custom and pre-trained data to create word representations. These models differ by a variety of factors like dimension of word vectors, size of context window, number of negative samples and loss function. Experiments with different dimensions and different models are also discussed later.

Once the models are generated, they are loaded using the genism module [13]. Next task is to represent each tweet in the training dataset, which is cleaned by several pre-processing steps, as a vector of features. This is done as follows:

5.3.1 Tokenizing Each Tweet into Words
Each tweet is made up of sentences and each sentence of multiple words. Sentences are broken down into word tokens to process each word separately. NLTK tokenizer is used to break sentences into individual words [14].

5.3.2 Removing Stop Words
Stop words are a set of commonly occurring words in any language. They typically are prepositions, conjunctions and determiners like “the”, “a”, “an”, etc. The basic intuition for using stop words is that once they are removed, one can focus on other words that hold more information instead.

5.3.3 Removing Numbers
Numbers hold no value when detecting emotions in text. In most cases, they simply convey some time or date or quantity and are totally irrelevant for in the current domain. They simply add noise and hence must be removed. Again regular expressions are employed to replace them with blanks.

5.3.4 Vector Averaging
This is the most important step, after tokenizing and removing stop words, what remains is each tweet broken into words. Vector for each word in the list is added and vector operations are used to average the resultant vector over the number of words. The result is a vector of a particular dimension (based on the model) for each tweet and a label for each vector. This will be used to train the classifiers.

For custom trained vectors, the models are trained using 1 million tweets collected with random queries and no specific emotion tags, this ensures all kind of Twitter language is captured by the models. They will be tested against pre-trained vectors. Models experimented with include Word2vec model trained over google news data with 300 dimensions, Glove model with 200 dimensions trained over 2 billion tweets and Fasttext model with 300 dimensions [2,6,7].

5.4 Classification
Many statistical classification algorithms have been used for text classification including regression models, decision trees, Bayesian classifiers, neural networks, support vector machines, etc. But with an increase in computational capacity one can use more compute intensive classifiers. One such classifier is Random Forest Classifier. It can be seen as a bootstrapping algorithm which uses Decision Tree (CART: classification and regression tree) model. It will create CART model with random samples and random initial variables. Final classifier is a function of all predictions. Random forests are used because they give more accurate results when compared to pure CART and CHAID (Chi-Squared Automated Interaction Detection) algorithms, especially with a large number of variables and huge dataset in text categorization [15]. The classifier is trained with resultant average vectors explained in the previous section and uses the trained model to make predictions on test samples using Scikit-learn’s ensemble package [16]. Out of 160138 samples, 112096 samples as training data and 48042 samples as test data with a varying number of estimators, the results of which are discussed in the next section.

6. RESULTS

A total of 160138 tweets were mined. Table 5 shows the distribution of these tweets in each category.

Table 5. Number of Tweets of each category

Class	Number of tweets
Happy	119858
Sad	67694
Angry	74328
Surprise	58396

Classification results of Random Forest Classifiers by changing estimators for the Fasttext model are shown in Table 6.

Table 6. Variations in Precision and Recall

No of Estimators	Precision	Recall
5	0.78	0.75
50	0.89	0.86
100	0.88	0.88
200	**0.91**	**0.90**

From the results, 200 estimators give the best training time and highest precision and accuracy. The same settings are used to train classifiers for labeled vectors. Table 7 shows precision and recall achieved after training on vectors generated by different word embedding models.

Table 7. Precision and Recall results across all models

Model	Dimension	Precision	Recall
Word2vec (pre-trained)	300	0.90	0.89
Glove (pre-trained)	200	0.88	0.88
Fasttext (pre-trained)	200	0.88	0.87
Fasttext	300	0.91	0.90

From the results, it is clear that out of all the models, the Fasttext model performs best with 300-dimension model and 200 estimators. Table 8 shows the confusion matrix for that model.

Table 8. Confusion Matrix Fasttext model

		Actual Class			
		Happy	Surprise	Angry	Sad
Predicted Class	Happy	**17573**	65	234	99
	Surprise	780	**7077**	578	284
	Angry	540	96	**10111**	381
	Sad	600	94	924	**8606**

7. CONCLUSION

The proposed model successfully classifies tweets based on emotions expressed by their authors with a highest possible precision of 91%. It can also handle variations in linguistic styles like sarcasm, slang, emoticons, emoji, incorrect spellings and even figures of speech. And it does so with relatively high computational efficiency. Processing textual data at almost million words per second.

8. FUTURE SCOPE

The proposed model can detect up to four emotions in the text with high accuracy, however, this is just a very broad classification. In future, we aim to classify text into more fine-grained categories and capture nuanced differences with the power of word embedding. We also, plan to apply other concepts of machine learning to optimize the model to an even greater accuracy using concepts from unsupervised learning. We had discussed various applications of emotion classification and are interested in applying them to various tasks like stock market prediction and demographic mental health study with the help of text and content from various sources including data from other social media.

9. REFERENCES

[1] Bollen, Johan, Huina Mao, and Xiaojun Zeng. "Twitter mood predicts the stock market." Journal of computational science 2.1 (2011): 1-8.

[2] Mikolov, Tomas, et al. "Efficient estimation of word representations in vector space." arXiv preprint arXiv:1301.3781 (2013).

[3] Bengio, Yoshua, et al. "A neural probabilistic language model." Journal of machine learning research 3.Feb (2003): 1137-1155.

[4] Schwenk, Holger. "Continuous space language models." Computer Speech & Language 21.3 (2007): 492-518.

[5] Mikolov, Tomáš, et al. "Empirical evaluation and combination of advanced language modeling techniques." Twelfth Annual Conference of the International Speech Communication Association. 2011.

[6] Pennington, Jeffrey, Richard Socher, and Christopher Manning. "Glove: Global vectors for word representation." Proceedings of the 2014 conference on

empirical methods in natural language processing (EMNLP). 2014.

[7] Bojanowski, Piotr, et al. "Enriching word vectors with subword information." arXiv preprint arXiv:1607.04606 (2016).

[8] Joulin, Armand, et al. "Bag of tricks for efficient text classification." arXiv preprint arXiv:1607.01759 (2016).

[9] Hasan, Maryam, Emmanuel Agu, and Elke Rundensteiner. "Using hashtags as labels for supervised learning of emotions in Twitter messages." Proceedings of the Health Informatics Workshop (HI-KDD). 2014

[10] Hasan, Maryam, Elke Rundensteiner, and Emmanuel Agu. "Emotex: Detecting emotions in twitter messages." (2014).

[11] Barbieri, Francesco, Francesco Ronzano, and Horacio Saggion. "What does this Emoji Mean? A Vector Space Skip-Gram Model for Twitter Emojis." LREC. 2016.

[12] Norvig, Peter. "How to write a spelling corrector." De: http://norvig. com/spell-correct. HTML (2007).

[13] Rehurek, Radim, and Petr Sojka. "Software framework for topic modelling with large corpora." In Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks. 2010.

[14] Bird, Steven, Ewan Klein, and Edward Loper. Natural language processing with Python: analyzing text with the natural language toolkit. " O'Reilly Media, Inc.", 2009.

[15] Xu, Baoxun, et al. "An Improved Random Forest Classifier for Text Categorization." JCP 7.12 (2012): 2913-2920.

[16] Pedregosa, Fabian, et al. "Scikit-learn: Machine learning in Python." Journal of Machine Learning Research 12.Oct (2011): 2825-2830.