International Journal of Computer Applications (0975 - 8887)
Volume 180 - No.32, April 2018

A New Recommender System for Hashtags

Varun Ranganathan
Student at PES University

Tanya Mehrotra
Student at PES University

ABSTRACT

Hashtags are one of the trendiest methods to label content on the internet. They are used to cluster data which in turn, results in its easier retrieval. Twitter is the main source from where the use of hashtags rocketed [1]. With the generation of large amounts of microposts, there is a need for effective categorization and search of the data. In this paper, we aim to propose a new method of recommending hashtags for a given message in a tweet so that hashtags can be effectively used by both data analysts and common users of the twitter platform. We will use various machine learning and deep learning concepts, and later combine them all together in one model to obtain a set of relevant hashtags for tweets fetched at real time. The results obtained by this new model resulted in far better recommendation than when each of the models were implemented separately.

General Terms

Twitter, Tweets

Keywords

Naive Bayes, Support Vector Machine, Artificial Neural Network, Machine Learning

1. INTRODUCTION

The use of hashtags has accelerated in the recent times. Everyday, a topic which is of pivotal importance to the society, becomes a trending hashtag. The use of hashtags has not only taken over Twitter, but now is also expanding over to other social media outlets like Facebook and Instagram. Of the above mentioned microblogging services, Twitter has emerged out to provide a dominant service with estimates of 250 million registered users who post up to 500 million Tweets per day [11]. With the increase in the popularity of Twitter, the need for an organizational strategy to manage its contents is also increased, hence the use of hashtags. They give a conglomeration of numerous text, pictures and videos about the topic to which it is related. There is no limit to the type of hashtag a user can create, unless and until it is a single alphanumeric string of text.

Now, we shall briefly introduce the three different classification techniques used by our model. The task of recommendation is converted into a task of classification over three classifiers, namely, Naive Bayes Classifier, Support Vector Machines, and Deep Neural Networks.

A Deep Neural Network (DNN) is an artificial neural network (ANN) with multiple hidden layers between the input and output layers.[6] [7] DNNs can model complex non-linear relationships. DNN architectures generate compositional models where the object is expressed as a layered composition of primitives. The extra layers enable composition of features from lower layers, potentially modeling complex data with fewer units than a similarly performing shallow network [8]. The deep neural network is a great learning algorithm which offers a great deal of accuracy with non-linear data, but, it does have its own set of limitations. The amount of data needed to train, so that it can classify its input correctly, is enormous, and for the task at hand, which comprises of a classification task involving more than 1000 classes, the enormity of the amount of data required becomes ten-fold. With the drawback of the requirement of data comes another drawback of training time. If we would have to recommend hashtags which have been created recently due to a particular real-time event, we must not wait for the neural network to train and then suggest. Therefore, we have to introduce another classifier which can work with less amount of data but can be trained in a short period of time.

The Naive Bayes classifier is a straightforward and powerful algorithm for the classification task. Based on the Bayes Theorem. it predicts membership probabilities for each class such as the probability that given record or data point belongs to a particular class. This algorithm is fast and highly scalable, as it's computations are based on a bunch of counts. Therefore, this algorithm would perfectly introduce real-time capabilities to our model. But, due to the Bayes Theorem, the naive bayes classifier assumes that all the features are unrelated to each other. Presence or absence of a feature does not influence the presence or absence of any other feature. This poses a lot of problems when it comes with text data, as a few words may be clubbed together in a text to represent a feature. By not considering the relationship between features in the input (words in our case), we will lose out on accuracy. The second disadvantage of the naive bayes classifier is that it does not work efficiently on a large amount of data with several classes. The resulting accuracy in mapping tweet messages to the hashtag used reduces dramatically when the number of tweets enter the model. Therefore, we need to find another classifier which can bridge the gap between the naive bayes classifier and the deep neural network, so that while the neural network trains, the load of incoming tweets need not be applied on the naives bayes classifier.

The Support Vector Machine acts like this bridge for us. Support vector machines are supervised learning models that analyze data used for classification. It can be coupled with various kernels which offer us the non-linearity we require. Due to this non-linearity in

the required hypothesis function, the amount of data needed to train the SVM would be on the higher side, but not as much as the deep neural networks require. The classification task by the SVM is done by find a curve which can divide the data points between the classes of data given to it. Now, this is slightly more computationally expensive than the naives bayes classifier, and therefore it would take more time to train. But the advantage the SVM offers is a more accurate classification of hashtags for more number of hashtags.

The Naive Bayes Classifier is analogous to the L1 Cache on the CPU and the Support Vector Machine is analogous to the L2 Cache on the CPU. If we were to compare the Deep Neural Networks to the other classifiers in terms of memory units in a computer, the Deep Neural Network is the hard disk, or a solid state disk at best. Capable of forming very complex functions, but time consuming.

2. PROBLEM STATEMENT

The main advantage of using hashtags is enhancing Twitter search. There is a vast pool into which information can be grouped and accessed. But it is seen that only 10% of the tweets are tagged with hashtags, which somehow destroys their purpose [3]. Thus, if there is a system in which hashtags can be suggested to the users based upon his tweet, more data will end up getting tagged and clustered together. With this dramatic increase of the usage of hashtags, the audience of the content that is put on the web also increases, resulting in better content delivery systems.

We seek to address the problem of making the data more available to the users of Twitter, by increasing the scope of the content provided to them. This should be achieved effortlessly and be as accurate as possible. The simplest way to bring about this modification is to use relatable hashtags for a tweet, such that it can be associated with as many topics as possible, without losing its sense. In this project we provide a platform for the users to get a list of suggested hashtags for the tweets entered by them. Hence, we have build a 'New Recommendation System for Hashtags'. The implementation is done in real time, using Tweepy, the Twitter's API, to fetch real time tweets.

3. LITERATURE SURVEY

3.1 Godin, Frderic, Viktor Slavkovikj, Wesley De Neve, Benjamin Schrauwen, and Rik Van de Walle. Using topic models for twitter hashtag recommendation. In Proceedings of the 22nd International Conference on World Wide Web, pp. 593-596. ACM, 2013

3.1.1 Approach. Using the Twitter streaming API, 18 million tweets were collected. The dataset was preprocessed by removing URLs, special HTML entities, digits etc. Slang words were converted into proper english words. An unsupervised language classifier was used because the language of the tweet may or may not be the same as the locale. For Hashtag Recommendation, a binary classifier, based on Naive Bayes technique was implemented, which discriminated between English and non-English language tweets. Latent Dirichlet Allocation (LDA), which is a hidden topic model, was used for retrieving information from a document by finding out the general topics in it. Gibbs Sampling Algorithm was used to find the undelying topic of a new tweet. From this topic distribution, keywords were selected and later used as hashtags.

3.1.2 Results. The language classification algorithm converges to a local maxima and therefore the training set depends upon the starting conditions. Thus, while training, multiple versions of the classifier were fed with different starting parameters. The language classifier gave an acccuracy of 97.4%. For 80% of the tweets, at least one suitable hashtag could be suggested out of five recommended hashtags.

3.2 Kywe, Su, Tuan-Anh Hoang, Ee-Peng Lim, and Feida Zhu. On recommending hashtags in twitter networks. Social Informatics (2012): 337-350

3.2.1 Approach. The dataset was collected on Singapore users over a period of thee months, comprising of a total of 44M tweets. Users were represented by their preference weights for each hashtag. The hashtag weightage is given by TF-IDF method. Tweets were also represented as a weighted vector of words using the same technique. The probable hashtag that can be recommended for a user u and tweet t was obtained by taking the union of hashtags from top-X similar users and top-Y similar tweets. This new list of suggested hashtags was ranked by frequency and the top ranked ones were recommended.

3.2.2 Results. The results depend upon the value of X and Y. 40% of the hashtags generated per day are fresh and stable usage patterns were observed over the time span of 3 months.

4. MODEL ARCHITECTURE

4.1 Data Source

NLTK tweets are taken as our dataset. The NLTK dataset contains tweets of General Elections 2015, which are only used for training purposes because its specific to a particular topic. Since we aim to generate hashtags in real time, Tweepy, the Twitter's API is used for testing purposes. Tweets in their raw form becoming very difficult to analyze, and therefore they need to be preprocessed and then given to our models. The metadata of the tweet consists of many fields, like location, timestamp, user details, URLs mentioned in the tweet etc. However, all this is not required for our model. We are only concerned with the text of the tweet written by the user and no other details. Therefore, we take care that only the tweet, i.e., the text in the tweet is recorded in the dataset and nothing more, just to reduce the complexity.

4.2 Preprocessing

Various techniques of Natural Language Processing have been used to preprocess the data for the contents in the tweet. Post using the word tokenizer and tweet tokenizer, a lot of changes are applied to the tweets in order to standardize them so that the models can easily classify to give an appropriate result. Contraction words like cant, dont, etc have been replaced by their original words like can not and do not respectively. The words are then lemmatized, aiming to remove inflectional endings and to return the base or dictionary form of a word. Next, stopwords are removed because they are not at all relevant and should not be taken into consideration while recommending hashtags. Any forms of URLs present in the tweet are also removed. Lastly, slang language, which mostly consists of abbreviated words, is replaced by complete, meaningful words. Another of the methods used in preprocessing, is correcting the misspelled words. Spelling corrector, made by Peter Norvig has been adopted for the same

[9] - an edit to a word is carried out by deletion, transposition (swapping), replacement or insertion. Words are restricted to a limited vocabulary defined for this method and the number of edits made to a misspelled word, to convert it into a meaningful word from the vocabulary is taken into account. Corrections that require two simple edits are also taken into consideration, generating a much bigger set of probabilities, but not necessary all words formed are known words. P(word) is estimated by its occurance in a text file containing a million words, big.txt. It is a concatenation of public domain book excerpts from Project Gutenberg and lists of most frequent words from Wiktionary and the British National Corpus. This model gives around 68-75% accuracy.

Additionally for the deep neural network, there is a requirement of converting the message into a vector. For this purpose, we used a one-hot encoded vector where each vector is the size of the number of words in the GloVe dictionary, where the position of '1' represented that the word, relative to the position in the GloVe dictionary, is present in the message. We chose the GloVe dictionary as there is a partial ordering of words, based on their semantics [10].

4.3 Naive Bayes Model

The input to the Naive Bayes classifier is a file containing a dictionary, wherein the keys are the hashtags and the values are words of the corresponding cleaned tweet (stored in the form of a list). Tweets that do not contain a hashtag in the training set are omitted.

Naive Bayes Theorem states : $P(A|B) = \frac{P(B|A)P(A)}{P(B)}$

Where,

1)P(A) represents the prior probability, or initial belief of A

2)$P(A|B)$ represents the conditonal probability of A given B.

3)$\frac{P(B|A)}{P(B)}$ represents the support B provides for A

We are examining the following conditional probability :
$P(hashtag/tweet) = \frac{P(hashtag)*P(tweets/hashtags)}{P(tweets)}$

The algorithm makes a naive assumption that all features are independent, given the hashtag :

$$P(hashtag|tweets) = \frac{P(hashtag)*P(t1|hashtag)*...*P(tn|hashtag)}{P(tweets)}$$

Rather than computing P(tweet) explicitly, the algorithm just calculates the numerator for each label, and normalizes them so they sum to one :

$$P(hashtag|tweet) = \frac{P(hashtag)*P(t1|hashtag)*..P(tn|hashtag)}{SUM[h](P(h)*P(t1|h)*..P(tn|h)}$$

If the classifier encounters an input with a feature that has never been seen with any label, then rather than assigning a probability of 0 to all labels, it will ignore that feature. Hence, all tweets having no hashtags, are ignored.

Training is performed on the feature set and while testing, the most probable hashtag is returned.

4.4 Support Vector Machine

The text is converted into vectors for SVM predictions. Its implementation is somewhat similar to that of Naive-Bayes. The input again, is taken in the form of a dictionary, where the keys are the hashtags and the values is a list containing the words of the corresponding cleaned tweet. While forming the dictionary from the tweets and their hashtags, if hashtag(s) is absent from a tweet, that tweet is ignored, for simplicity purposes. SVM was trained by Linear as well as Radial Basis Functions(RBF). It turns out that Linear SVM gave an accuracy of 35%. On the contrary, RBF resulted into an accuracy of 13%. Thus, Linear SVM is adapted for further computations.

4.5 Combined Model

We know that a deep neural network takes a lot of time to train, and for every new hashtag, a new class is to be created, and all the tweets need to be trained yet again. This is not feasible computationally and is very time-consuming. There, we decided to use smaller classifiers to train for fewer tweets which can be trained repeated with very little cost of time. Our idea involves combining three models of classifiers and running them simultaneously. The classifiers are Naive Bayes, Support Vector Machines, and a Deep Neural Network. Each model has a buffer which indicates as to how many tweets it can train at a time. The buffers are set for all the three models : Naive Bayes - 100 tweets, Support Vector Machine - 1,000 tweets, Deep Neural Network - 100,000 tweets.

The number of tweets chosen in accordance to the time they take to train, corresponding to the accuracy given by each model. Naive Bayes takes the minimum time for training so it is executed first, with the least buffer. It also gives us a higher accuracy for smaller amount of tweets. Support Vector Machine, which takes more time than Naive Bayes to train, is kept second, having medium buffer. Lastly, Deep Neural Network, being the most accurate model, takes the most time to train, and hence is executed at the last, having a huge buffer size. The Deep Neural Network does give us a much higher accuracy, but it does require a lot of data.

A tweet enters the listeners process and it is added to the naive bayes, support vector machine, and deep neural network buffers. When a threshold for training the naive bayes model is reached (currently set to 100) ,the tweets in the naive bayes buffer are trained. The training process ends, but the buffer is not cleared, yet. It is theoretically not possible to retrain a classifier for new classes, therefore this imposes a restriction. Another 100 tweets come into the listener process, and the naive bayes classifier is re-trained for all the 200 tweets. The threshold kept for the support vector machine is 1,000 tweets, therefore, when the number of tweets received reaches 1,000, the support vector machine starts to train, and after its training process, it will be able to predict hashtags that were being predicted by the naive bayes model (since it already consisted of the tweets that were previously been trained by the naive bayes classifier). Therefore, the buffer of the naive bayes (the first 1,000 tweets in the buffer) can now be deleted. Now, the naive bayes model will be trained for the next 900 tweets. The same logic is extended for the deep neural network. The threshold set for the deep neural network model is 100,000, because it takes

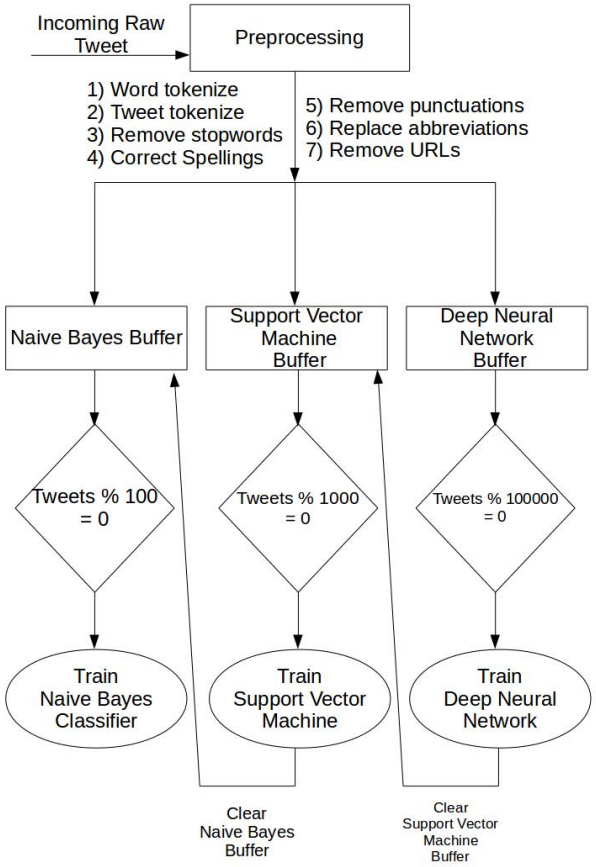

Fig. 1. Flowchart for the Listener Process

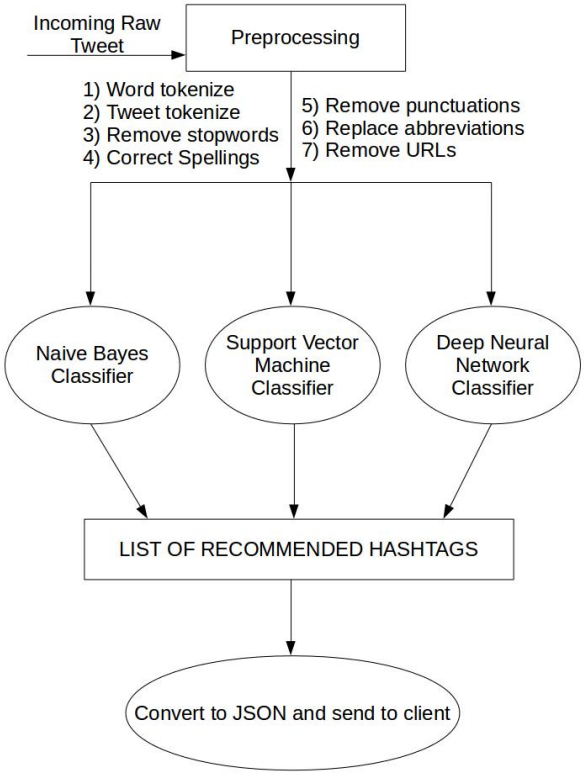

Fig. 2. Flowchart for the Advisor Process

a lot of time to train it, therefore adequate time should be given to it to train and be constantly used taking into the accuracy decrease of support vector machines when the number of classes increase. When the number of tweets increase to 100,000, the tweets in the buffer for the deep neural network are used to train the neural network, whilst the naive bayes model and the support vector machine model continue to work as before. After the completion of the training process of the neural network, the buffer of the support vector machine (first 100,000) and the buffer of the naive bayes (first 1,000) are cleared, since the deep neural network model is now able to predict the hashtags for the tweets contained in both naive bayes and SVM model. Therefore, after clearing their respective buffers, they can work on newer tweets more efficiently.

5. RESULTS

The results obtained are on random tweets taken from the NLTK Tweets dataset. The accuracy shown is based on the testing data chosen using the K-Fold splitting of tweets. The split value taken for K-Fold is 3, which would mean that, 2 folds (67% of the tweets) will be used for training and 1 fold (33% of the tweets) will be used for testing.

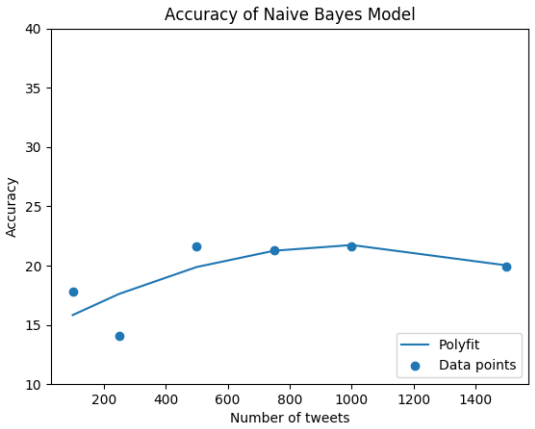

Fig. 3. Naive Bayes

The graph shown in the figure 3 shows the accuracy of the naive bayes model on testing data. We can observe an increase in the accuracy till 1000 tweets. This can be reasoned with the availability of hashtags for the model to chose from. With only

100 tweets, the number of hashtags available will be limited, and this would fair rather poorly with the test tweets. Now, with the availability of more number of tweets, the number of hashtags will also increase giving the classifier more number of classes to choose from. At about 1000 tweets, the naive bayes classifier has a relatively higher accuracy. But at this point of time, the number of tweets that must be used for training the naive bayes classifier is high. Computationally, this becomes very expensive, as naive bayes has a Big-O complexity O(Nd), where 'N' is the number of training examples and 'd' is the dimensionality of the features which is the number of words in a tweet (maximum 140 words due the character limit). The computation increase amounts to increase in training time, and at the 1000 tweets mark, the classifier takes too much time to train (about 15 minutes), resulting in loss of real-time capabilities. We can also observe a steep decrease in accuracy past 1500 tweets. Therefore, by setting the threshold as 1000 for the SVM model, the naive bayes classifier will train only for 900 tweets.

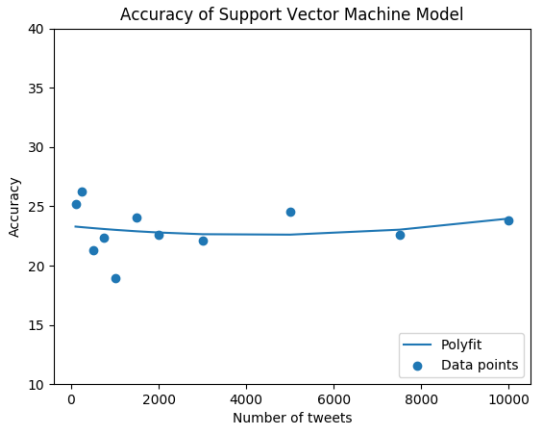

Fig. 4. Support Vector Machine

Figure 4 shows the accuracy of the support vector machine classifier on testing data. It can be observed that, from the 2000 tweets mark, the accuracy steadily increases. The complexity of the training of support vector machine is O(Nr), where 'N' is the number of training examples and 'r' is the number of iterations (sklearn set 1000 as default). Since the SVM classifier involves linear vector multipliers, it can be highly vectorized, leading to lower training time. Taking the relative performance of Naive Bayes and SVM, it can be observed that the SVM classifier, even after 1000 tweets, is able to train much faster than the Naive Bayes Classifier.

Therefore, we can conclude that SVM makes a perfect bridge between the Naive Bayes Classifier and the Deep Neural Network. Accuracies shown in the graphs are relatively low due to the fact that tweets can contain more than one hashtag, whereas the classifiers are tested for only one hashtag.

Just like the support vector machine outperforms the naive bayes at when the dataset is higher, similar logic can be employed for the training of the deep neural network. The deep neural network requires a massive amount of data before it can be trained efficiently, but once trained it performs very well.

The results achieved are above satisfactory, as all the models work well with each other, and the architecture of the overall machine learning model is complex but enables us to give results in real time, i.e., it helps us identify the events that are happening around the world in real time. The accuracy achieved by each model is quite satisfactory, as we are trying to classify over 5000 hashtags. If we were to randomly pick one hashtag, the probability of picking a correct recommendation would be 1/5000 = 0.0002, or 0.02%. But we are achieving an accuracy in the high 70s with the neural network. Similarly, we are able to achieve accuracies of 40% with the naive bayes model for about 50 classes, and about 50% for over 200 classes with the support vector machine.

6. CONCLUSION

Three models combined and executing in parallel gives us better recommendations on the hashtags that could be used, as it recommends both newer and older hashtags. The buffering system is threaded, which takes care of the speed too. We need not wait indefinitely for a process to complete and waste time. This method takes into account its actual implementation in the real world.

This approach on building a recommender system to show users the possible hashtags that may be appropriate for the content of the tweet is highly time efficient. It can be used in the long run to give trending tweets providing social media admins an effective way to learn about trending hashtags that they could use with respect to their field, or in general, the current happenings about the world. Although this may be a very efficient method to give out recommendations, it could tend to be computationally very expensive, and must be run on a separate server, but it is definitely feasible in the real world.

7. REFERENCES

[1] Godin, Frderic, Viktor Slavkovikj, Wesley De Neve, Benjamin Schrauwen, and Rik Van de Walle. 'Using topic models for twitter hashtag recommendation.' In Proceedings of the 22nd International Conference on World Wide Web, pp. 593-596. ACM, 2013.

[2] Krokos, Eric, Hanan Samet, and Jagan Sankaranarayanan. 'A look into twitter hashtag discovery and generation.' In Proceedings of the 7th ACM SIGSPATIAL International Workshop on Location-Based Social Networks, pp. 49-56. ACM, 2014.

[3] Kywe, Su, Tuan-Anh Hoang, Ee-Peng Lim, and Feida Zhu. 'On recommending hashtags in twitter networks'. Social Informatics (2012): 337-350

[4] Sriram, Bharath, Dave Fuhry, Engin Demir, Hakan Ferhatosmanoglu, and Murat Demirbas. 'Short text classification in twitter to improve information filtering.' In Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval, pp. 841-842. ACM, 2010.

[5] Popescu, Ana-Maria, Marco Pennacchiotti, and Deepa Paranjpe. 'Extracting events and event descriptions from twitter.' In Proceedings of the 20th international conference companion on World wide web, pp. 105-106. ACM, 2011.

[6] Bengio, Yoshua (2009). 'Learning Deep Architectures for AI'. Foundations and Trends in Machine Learning. 2 (1): 1127. doi:10.1561/2200000006

[7] Schmidhuber, J. (2015). 'Deep Learning in Neural Networks: An Overview'. Neural Networks. 61: 85117. arXiv:1404.7828 Freely accessible. doi:10.1016/j.neunet.2014.09.003. PMID 25462637.

[8] Szegedy, Christian; Toshev, Alexander; Erhan, Dumitru (2013). 'Deep neural networks for object detection'. Advances in Neural Information Processing Systems.

[9] Peter Norvig. 'Spelling Corrector in Python 3'. Copyright (c) 2007-2016 MIT license: www.opensource.org/licenses/mit-license.php

[10] Jeffrey Pennington, Richard Socher, Christopher D. Manning. 'GloVe: Global Vectors for Word Representation'

[11] Raja Jurdak , Kun Zhao, Jiajun Liu, Maurice AbouJaoude, Mark Cameron, David Newth. 'Understanding Human Mobility from Twitter'