Twitter Sentiment Analysis using BiSense Emoji Embedding

Tanishq Chaudhary
tanishq15105@iiitd.ac.in
IIIT Delhi
Delhi, India

Ronak Kumar
ronak15080@iiitd.ac.in
IIIT Delhi
Delhi, India

Aryaman Dobhal
aryaman15016@iiitd.ac.in
IIIT Delhi
Delhi, India

Saurabh Kumar
saurabh15089@iiitd.ac.in
IIIT Delhi
Delhi, India

ACM Reference Format:
Tanishq Chaudhary, Ronak Kumar, Aryaman Dobhal, and Saurabh Kumar. 2018. Twitter Sentiment Analysis using BiSense Emoji Embedding. In *Proceedings of* . ACM, New York, NY, USA, 6 pages.

1 ABSTRACT

Sentiment Analysis has recently become a hot-topic of concern due to large scale social media content being generated both from online and offline. This data has been seen responsible for various real-world activities like emotional status monitoring of people and predicting choice of citizens during candidate elections. The study of sentiment from textual data has always been considered important but recently the most popular social media websites like Twitter and Instagram have started making use of emojis for enhancing people's opinion. These emojis have replaced the place of emoticons which were used before them as a form of communication through feelings. However, since emoticons were limited in number, people have started using emojis in their messages as they form a more graphical and colorful way for expressing one's opinion as opposed to random text. Since, emojis play a major role now in the text being generated from social media and have become a new language of expression, analyzing sentiments with them can make a huge impact in various areas of research related to opinion mining. In this paper, we present a novel approach for the problem of Sentiment Analysis by giving more attention to emojis. We apply different machine learning techniques to train our sentiment classifier on our data set consisting of tweets with positive and negative sentiment. Thus we have conducted various experiments using different algorithms for training our sentiment analysis classifier ranging from Naive Bayes, SVM, Random Forest to 4 layered neural network. We also demonstrated the results after running our final text-emoji based sentiment classifier on our testing dataset to derive more insights on the results and accuracy and displayed them in the results section.

, ,

2 INTRODUCTION

In the past few years, social media websites like Twitter have become responsible for generating huge amounts of data varying from all kinds of sentences depicting people's opinion, news, jokes, memes and what not. This rich multimedia data has attracted researchers and students towards a pool of future opportunities like Sentiment Analysis, Detecting Fake Tweets etc. Sentiment Analysis formally known as the field of opinion mining has been widely used in real-word applications like predicting public's opinion during elections, business related activities etc. Recently, emojis have started appearing on the most popular social media platforms like Twitter, Facebook and Instagram. Emojis, which consist of various graphical symbols ranging from person's face to sports like cricket etc. They are used widely to enhance a person's opinion inside the raw text to give more insights to the readers so that they understand what the person is actually saying. They have replaced emoticons in 2010, since they are more easy and fun to use. The data from the latest social media reports reveal that there are about 2,823 emojis in online social media which are being used daily as a way of communication. Also, some statistics reveal that more than 50% of Instagram posts have emojis inside them and about 92% of the total online social media population makes use of emojis. Many sources like Twitter have even released their open source emoji dataset like Twemoji, where you can freely use their emojis in any of your message. This wide use of emojis in online social media has attracted attention of a lot of researchers since they can used in the area of opinion mining and also conducting analysis on emojis along with text semantically right now is pretty new. Some previous works reveal that using a pre-trained deep neural network for emoji prediction task is useful but they failed to consider the linguistic complexities occuring in the arena of emojis. Thus, they are unable to understand emojis semantically especially in sentences which involve a dual/bi-sense/sarcastic meaning. Although, convential emoji sentiment goes in hand with the text sentiment, but in case of bi-sense sentences where the text sentiment differs from the emoji sentiment, the network fails. Thus, to solve all these problems and complexities arising for the use of emojis in sarcastic sentences, we try to build an Attention Based Long Term Short Memory network which is based on Recurrent Neural Network consisting of bi-sense embeddings as opposed to only word and emoji embeddings.

We initially apply the technique of NLP for getting initial text sentiments out of the sentence/tweet using the sentiment analyzer tool called vader. After that, we have implemented a custom class

called "tweet2vec" which embed word and emoji into a combined 300 dimension vector. We then use machine learning to classify our dataset into positive and negative sentiment. Our dataset uses a random mixture of tweets with and without emojis which are being fetched from the Streaming API. We have more than 100,000 Tweets for training our classifier. We demonstrated the final results and the accuracy using our testing dataset.

3 LITERATURE SURVEY

Due to the popularity of twitter and subsequently the use of emoji's, there have been a lot of research done prior on these topics.
Generally speaking researchers look at sentiment analysis as first and foremost an NLP problem and approach it with that in mind only. This is done by using features of keywords such that where certain keywords correspond to a sentiment vector representing the possibility of the word (i.e. what sentiment it represents).
Other approaches can range to the use of deep neural networks making the use of skip grams and continuous bag-of-words.
In the case of emoji's most approaches take the lexicon based idea in the sense that a certain score or sentiment preference is assigned prior (in say a dictionary) and on reading the emoji the sentiment of the text/tweet is assigned based on that.
However this lexicon/dictionary approach lacks the nuance that is seen in language such as the use of irony, sarcasm, multilingualism etc. Some of the related papers are as follows.

Sentiment Analysis of Twitter Data (Apoorv Agarwal, Boyi Xie, Ilia Vovsha, Owen Rambow, Rebecca Passonneau) uses POS-specific prior polariity features with giving certain emoji's and tweets a polarity (as a score) in terms of whether they are positive or negative or neutral.

Sentiment Analysis on Twitter (Akshi Kumar and Teeja Mary Sebastian) also works towards finding the sentiment of tweets with the use of a hybrid approach which uses both corpus based and dictionary based methods to get the semantic orientation of the said tweet. Just like the above mentioned method the paper also used scoring based on the tweet set tags/emoji's however in this case numeric scores were used to determine or set the sentiment of an expression.

Sentiment Analysis of Twitter Data Using Emoticons and Emoji Ideograms (Wieslaw Wolny) the paper aims to approach sentiment analysis for basic determination on whether the tweet is positive, negative or neutral. Classification is first done on the dataset using supervised learning with a lexicon-based approach. The lexicons contain a list of sentiment emojis and emoji ideograms, the tweets are classified as a whole based on what emoji is being used, so say a tweet with a smiling emoji shall be labelled as a positive sentiment.
The issue with this approach was that it assumed a tweets sentiment based just on the emoji used i.e. the emoji used in a tweet is representative of the tweet in general. While this might be true for most cases it disregards the scenario of sarcasm and other subtle differences in emoji usage.

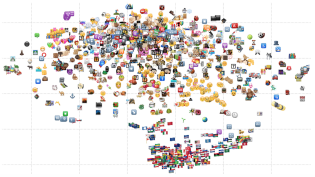
SentiReview: Sentiment Analysis based on Text and Emoticons (Ms. Payal Yadav and Prof. Dhatri Pandya This paper while not directly related to twitter per say, uses NLP and machine learning rather than a basic lexicon based approach to find the sentiment shown in a statement. It's done by feature extraction followed by classification in 3 categories namely positive, negative and neutral sentiments. However this method still has some drawbacks being that detecting sarcasm and taking multilingualism into account is not feasible.

Twitter as a Corpus for Sentiment Analysis and Opinion Mining (Alexander Pak, Patrick Paroubek) The paper works on sentiment analysis by gathering a corpus of data (tweets in this case) which are divided into three classes, namely positive, negative and objective texts aka texts with no sentiment. In terms of dealing with emoji's the paper only considers emoticons and classifies them as either sad or happy. These are then used to train a classifier to predict whether or not a tweet is showcasing a positive or negative sentiment. The classifier used was based on multinomial NaÄśve Bayes classifier which uses N-gram and POS tags as its features. While it did produce good accuracy the issue with this method was that it only considers a subset of emoji's and with the growing use of twitter and many different emoji's this is not a sufficient or long lasting method.

The Effects of Emoji in Sentiment Analysis (Mohammed O. Shiha, Serkan Ayvaz) The paper is an analysis of how emoji's effect the sentiment of a tweet or any text in general. The study used a lexicon based approach and in that used 843 emojis. To compare data sentiment was collected for an event first without taking emojis into account followed by sentiment analysis of the same event but with this time taking emojis into account. The results showcased that the use of emojis for sentiment analysis in both positive and negative events gave use an higher accuracy as compared to without using emoji's.

4 DATASET

Most of the big datasets containing tweets such as Sentiment140 were pre-processed and didn't had any emoji in them. Therefore we created our own dataset using Twitter REST API. Thus using that we were able to successfully fetch around tweets. Moreover after stripping non-english tweets our dataset comprises of around 193,323 tweets. We then used vader for labeling our dataset. Vader provides a compound score between [-1,+1], -1 is the maximum score achieved by negative tweet whereas +1 is the maximum score achieved by positive tweet. Furthermore inorder to avoid outliers and neutral tweets we didnt consider tweets with the compound score between [-0.05,0.05]. Below is the visualisation of the Emoji2vec model that we our using in our Tweet2vec model.



(Visualisation of emoji2vec embeddings)

5 METHODOLOGY

5.1 Modeled as binary classification

Inspired by Word2vec, Doc2vec and Emoji2vec we created a custom class called "Tweet2vec" which allows to combine the power of these models to provide a multi-modal vector associated with a tweet containing emojis. It basically takes the average of all the words and emojis and returns a final 300 dimension vector representing the tweet.

Pre-processing: First the data was parsed from the CSV file containing the tweets and the label. Then each tweet was tokenized with the help of TweetTokeniser() from NLTK. This helps in tokenising the data along while preserving the twitter oriented entities such as hashtags, links and emojis. Furthermore inorder to faciliate and read the emojis, all the tokens were decoded to utf-16. After that all the tokens were converted to lower case, stop words and punctuations were removed inorder to further clean the data.

Tweet2vec Conversion: All the processed tweets were then converted to vectorised tweets. Tweet2vec combines the vectors of emojis and words to create a single 300 dimension numpy array of the particular tweet. Both the vectors and labels were saved in pickle files for fast retrieval.

Classification: The data was shuffled and splitted in 80:20 train-test ratio for classification. Following classifiers were trained :-

(1) Naive Bayes
(2) Support Vector Machine
(3) Decision Tree Classifier
(4) Random Forest Classifier
(5) K-Nearest Neighbor
(6) Multi Layer Perceptron
(7) 4-Layered Neural Network

- **Naive-Bayes (Baseline) -** It is based on baye's theorem, which predicts by calculating posterior probability of the class.
- **SVM -** Support Vector Machine (SVM) is a discriminative classifier formally defined by a separating hyperplane. This is a supervised learning based classifier in which the algorithm outputs an optimal hyperplane which categorizes new examples. For our project we use the svm with the gamma set to 'scale' and the decision function shaped to 'ovo'.
- **Decision Tree Classifier -** A decision tree is a flowchart-like structure in which there are several internal node represents a test on an attribute, each branch represents the outcome of the test, and each leaf node represents a class label. Here we use the decision tree classifier with the max depth set to 5.
- **Random Forest Classifier -** A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. We used the estimator (number of trees) to be 100 in this case with the 'gini' criterion to measure the quality of the split.
- **K-Nearest Neighbor -** At its most basic level, it is essentially classification by finding the most similar data points in the training data, and making an educated guess based on their classifications. We trained KNN by taking n=2 i.e 2 clusters (1 positive, 1 negative)
- **Multi Layer Perceptron -** A multilayer perceptron is a class of feedforward artificial neural network. A MLP consists of at least three layers of nodes: an input layer, a hidden layer and an output layer. It also has the advantage that it can distinguish data that is not linearly separable.
- **4-Layered Neural Network** - Finally we also trained a 4 layer neural network. The input layer has the size of 300, Moreover the 1st hidden layer also has the size of 300 followed by the 2nd layer of size 30 and finally the output layer of size 2. The hidden layers used Rectified Linear Units as there activation function, followed by the sigmoid function in the output layer. The network was trained with 100 epochs to lower down the loss error.

5.2 Modeled as 3-class classification

Not all tweets express either positive or negative sentiments. Some tweets don't express either and hence are neutral. Therefore we thought of trying out a different approach. We modeled the problem as a 3-class classification problem with the classes being "Positive", "Neutral", and "Negative".

Dataset: Since this was an experimental approach so we worked with a subset of the data. After pre-processing we ran Vader [7] on it for getting the sentiment score. Tweets which had score between (0.6 to 1) were labeled as "Positive", score between (-1 to -0.6) was labeled "Negative", and rest was labeled as "Neutral".

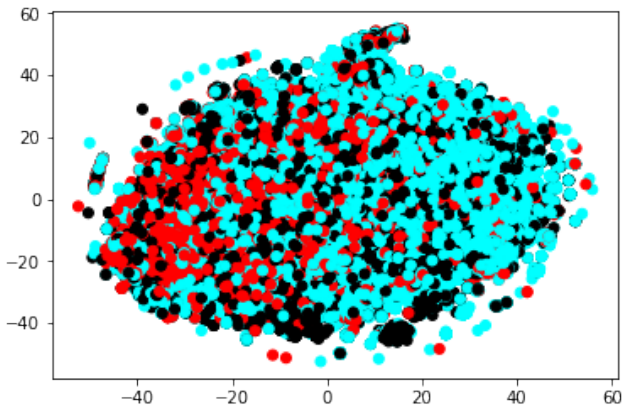
Pre-processing: We used a tweet pre-processing library called `tweet-preprocessor` [8]. We filtered out URLs, mentions, emails, and hashtags from the tweets because these do not contribute to sentiment analysis.

Conversion of tweets to vectors: We used open source pre-trained model `emoji2vec` [9] which is based on a paper of the same name [10]. This pre-trained model is used just like the pre-trained `word2vec` [11] model. The `word2vec` model is trained on the Google News word2vec dataset.
The `emoji2vec` library [9] provides useful methods to convert a phrase to a vector of a given dimension. We used the `phrase2vec` method of the library to convert our pre-processed tweets to a vector of 300 dimension. This method uses both `word2vec` and `emoji2vec` to convert the *text* and *emoji* parts of a tweet into a vector. We choose the vector length of 300 because it works well with `word2vec`. We then saved these vectors and their corresponding labels to a file for later use.

Visualizing the vectors formed: After pre-processing and forming the vectors, we had 55, 921 data points of 300 dimension. We tried to visualize these points in order to gain some insights as to how we should proceed with our training and classification. In order to draw a 2-dimensional plot of the data-points we used the

, ,

TSNE method of the sklearn library [12] for reducing the dimension from 300 to 2. After the dimensionality reduction we got the following plot of the vectors:



This visualization showed that it is difficult to separate it linearly.

Data Standardization In order to get better classification accuracy we standardized our data (of 55, 921 data points) and made the mean zero and a standard deviation as one (unit variance).

Training and classification: Once our problem was converted into a 3-class classification problem we ran an array of classifiers on it to get better classification accuracy. We shuffled and split our data into training and test set. 80% of the data was used for training and the rest 20% for testing the model's accuracy. The following classifiers were used:

- ELM
- SVM
- Logistic Regression
- MLP classifier
- Random Forest
- AdaBoost
- Gradient Boosting
- KNN

6 RESULTS

6.1 Result of binary classification

Algorithm	Accuracy %	F1-Score %	Precision %	Recall %
4-Layer DNN	86.82	83.52	85.33	82.20
SVM	86.74	83.52	85.04	82.38
MLP Classifier	85.18	82.29	82.22	82.35
Adaboost	82.37	78.17	79.31	77.30
Random Forest Classifier	80.84	73.10	81.14	70.72
Decision Tree Classifier	75.33	66.32	70.76	65.05
KNN	73.19	70.99	70.69	74.26
Naïve Bayes	54.55	54.45	62.08	63.05

Top 20 Emoji	Positive Context	Top 20 Emoji	Negative Context
😂	2540	😩	2052
😍	869	😭	778
😭	771	🙄	303
😊	536	😫	293
😌	520	🙃	129
❤	514	💯	129
😘	385	😑	128
✨	334	💀	127
👍	332	🔥	117
🙏	315	😍	113
😩	314	😢	109
💜	294	😅	100
🔥	287	😊	84
🙄	269	😒	82
💔	241	😔	79
❤	230	😡	73
💙	226	😌	71
👏	215	👍	60
🙌	213	😑	58
💯	210	❤	58

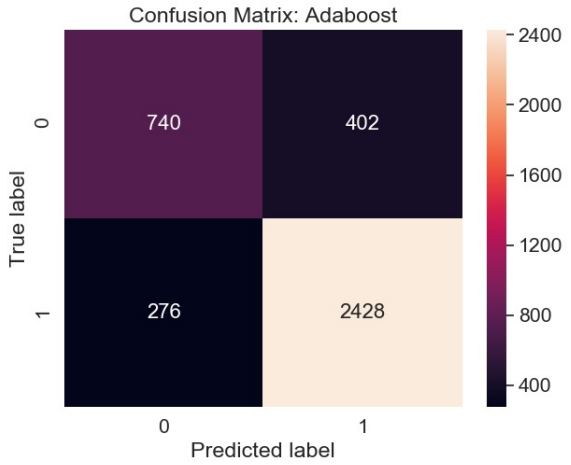
6.2 Result of 3-class classification

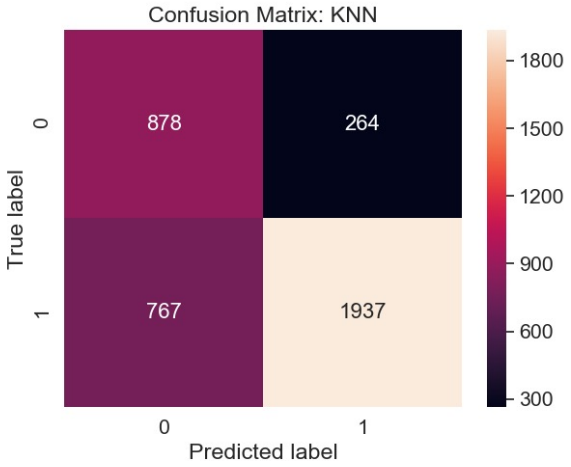
Table 1: Accuracy on different classifiers

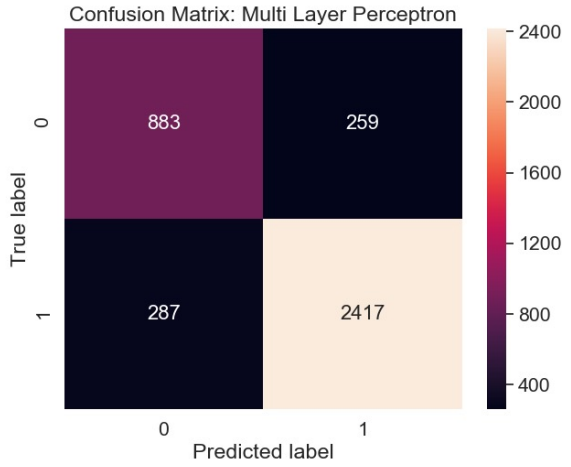
Classifier	Accuracy
ELM	0.41135
KNN	0.55011
AdaBoost	0.55619
Random Forests	0.56620
Gradient Boosting	0.58426
SVM	0.59418
Logistic Regression	0.59901
MLP	**0.59982**

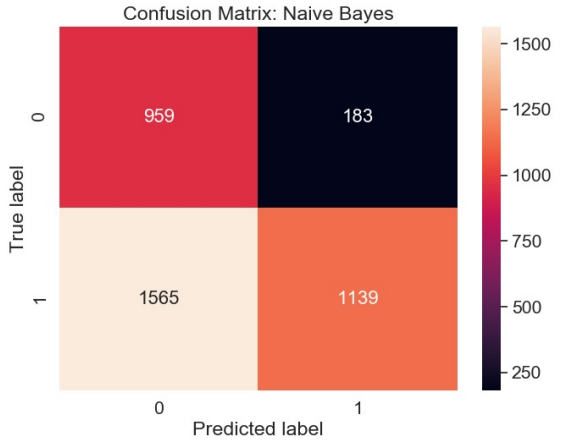
As we can see that the accuracy scores of 3-class classification were not very good although we used a lot of classifiers to improve our result. The best accuracy we got was with `Multilayer Perceptron` giving accuracy of **60%**. Therefore we thought that it is better to keep the **binary classification model** and do further analysis on that.

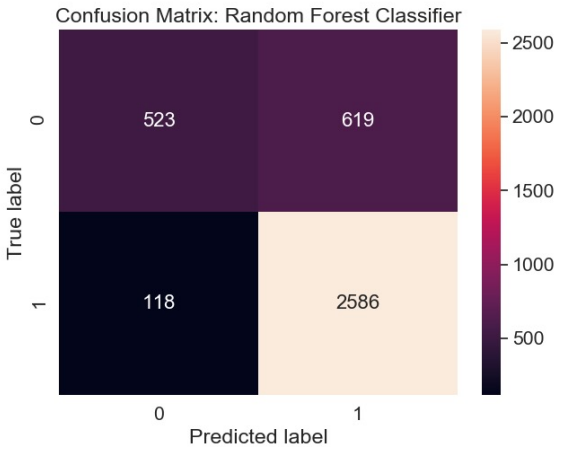
7 ANALYSIS OF RESULTS

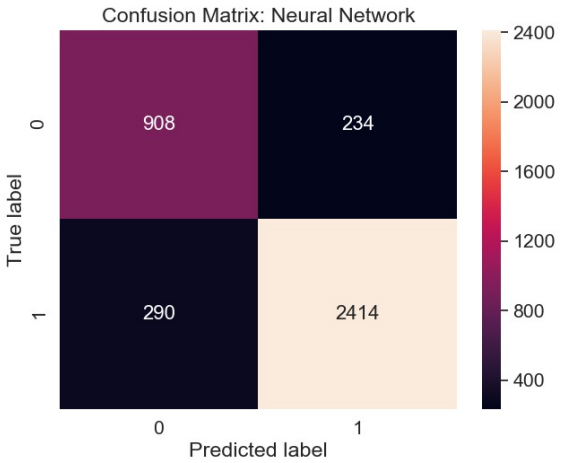












As can be seen from the confusion matrix, our baseline model performed the worst out of all classifiers. Lot of false positive and

false negatives can be seen in the table. However the performance increases as more complex classifiers are trained on the tweet2vec embeddings. This can be validated from the classifier's evaluation metrics such as precision, recall, f1 score and confusion matrix.

8 CONCLUSION

This paper presents a novel approach for various Machine Learning Algorithms. We conducted our classification with the baseline model Naive Bayes which gave us just 54.5% accuracy. We then experimented with different classifiers such as K-Nearest Neighbour which gave us better results and we got accuracy close to 73%. Then, we tried to improve accuracy of the algorithm further by using complex ML classifiers like SVM and Multi Layer Perceptron. We also tried ensemble classifier such as Adaboost and Random forest classifier. SVM provided us much higher accuracy and after SVM, we implemented a 4-Layer Deep Neural Network which had a slightly higher accuracy than SVM. Overall, our method involves by fusing the word2vec and emoji2vec models into a custom Tweet2vec model for a combined multi-modal text and emoji embedding. This approach is more robust than previous works that have been done in this area of research.

9 INDIVIDUAL CONTRIBUTION

(1) Tanishq - Preprocessing, Tweet2vec and Training and testing of ML/DL Binary Classifers
(2) Ronak - Vader classification, Generating Dataset using Twitter's API, Visualization & Naive Bayes.
(3) Aryaman - Dataset setup, pre-processing for 3-class classification and visualization of data.
(4) Saurabh - Vader 3-class labeling, tweet to vector conversion, training and testing ML classifiers for 3-class classification.

REFERENCES

[1] http://www.cs.columbia.edu/ julia/papers/Agarwaletal11.pdf
[2] https://www.ijcsi.org/papers/IJCSI-9-4-3-372-378.pdf
[3] https://pdfs.semanticscholar.org/eef0/4bd58707b827af8fb4e6f6792bf730bf96db.pdf
[4] https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7975659
[5] http://crowdsourcing-class.org/assignments/downloads/pak-paroubek.pdf
[6] http://www.ijcee.org/vol9/943-T048.pdf
[7] Clayton J. Hutto and Eric Gilbert. 2014. VADER: A Parsimonious Rule-Based Model for Sentiment Analysis of Social Media Text. In Proceedings of the Eighth International Conference on Weblogs and Social Media, ICWSM 2014, Ann Arbor, Michigan, USA, June 1-4, 2014.
[8] https://pypi.org/project/tweet-preprocessor/
[9] https://github.com/uclmr/emoji2vec
[10] https://arxiv.org/pdf/1609.08359.pdf
[11] https://code.google.com/archive/p/word2vec/
[12] https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html