Paris Descartes University
UFR math info

MASTER 1

VISION AND INTELLIGENT MACHINE

Big Data project report

TWITTER SENTIMENT ANALYSIS

Realized by:

Bouzid AREZKI
Manil MASDOUA
Mouad MOUNACH
Yakoub Mohamed AZZOUNI

Table of contents

Table of tables

Table of figures

I. Introduction & Problem Statement :

Microblogging sites have developed to become a source of different sorts of data. This is because of nature of microblogs on which individuals post real time messages about their opinions on different topics, talk about current issues and express positive sentiments for items they use in daily life. Indeed, organizations assembling such items have begun to survey these microblogs to get a sense of general opinion for their item or product. One challenge is to build technology to predict and summarize an overall sentiment.

Twitter is one of the most popular microblogging websites where individuals create and interact with messages known as “tweets”. This fills in as a mean for people to communicate their considerations or sentiments about various subjects. Different various parties, for example, purchasers and advertisers have done sentiment analysis on such tweets to assemble insights into products or to direct market examination. Besides, with the ongoing headways in AI and especially machine learning algorithms, we are capable to improve the precision of our sentiment analysis predictions.

II. Data Description :

The data given is as a comma-separated values documents with tweets and their comparing sentiments. The training dataset is a csv document presented as [tweet_id, sentiment, tweet] where the tweet_id is a special number distinguishing the tweet, sentiment is either positive as 1 or negative as 0, and tweet is the tweet encased in "". Also, the test dataset is a csv document presented as [tweet_id,tweet].

We used the sentiment140 dataset from kaggle. It contains 1,600,000 tweets extracted using the twitter api. The tweets have been annotated (0 = negative, 4 = positive) and they can be used to detect sentiment.

The dataset is a blend of words, emojis, symbols, URLs and references to people. Words and emojis contribute to predicting the sentiment, yet URLs and references to people don't. In this manner, URLs and references can be disregarded. The words are additionally a blend of incorrectly spelled words, additional punctuations, and words with many repeated letters. The tweets, in this manner, must be preprocessed to normalize the dataset.

See the table 1 for the provided training data set, and for test data set we used Twitter API.

III. Related works:

Pak and Paroubek: They made a twitter corpus by gathering tweets utilizing Twitter API and annotating those tweets utilizing emoji they built up a sentiment classifier dependent on the multinomial Naive Bayes strategy that utilizations highlights like Ngram and POS-labels.

The training set they utilized was less productive since it contains just tweets with emoticons

Parikh and Movassate(2009): Conceived two models, a Naive Bayes bigram model and a Maximum Entropy model to characterize tweets. They found that the Naive Bayes classifiers worked far superior to the Maximum Entropy model.

Barbosa (2010): planned a two-stage programmed notion examination technique to classify tweets.

They classified tweets as objective or non-objective, and afterward in second stage, the abstract tweets were named positive or negative.

The component space utilized included retweets, hashtags,link, accentuation and shout stamps related to highlights like earlier extremity of words and POS.

What's more, there is part of another researcher in the SA inquire about which they utilized differentes methodes like unigram model, an element-based model and a tree portion-based model.

IV. Methodology and Implementation :

1. Pre-processing:

Raw tweets scratched from twitter for the most part bring about a noisy dataset. This is expected to the nature of individuals utilization of social networks. Tweets have certain unique characteristics, for example, retweets, emojis, user mentions, etc. which must be appropriately extracted. In this way, raw twitter data must be standardized to make a dataset which can be handily learned by different classifiers. We have applied a broad number of pre- processing steps to normalize the dataset and diminish its size. We initially do some broad pre- processing on tweets which is as follows:

- Convert the tweet to lower case.
- Replace at least 2 dots (.) with space.
- Strip spaces and quotes (" and ’) from the ends of tweet.
- Replace at least 2 spaces with a single space.

a. Uniform Resource Locator (URL):

Users regularly share hyperlinks to different website pages in their tweets. A specific URL isn't significant for text

classification as it would lead to very sparse features. Thusly, we replace all the URLs in tweets with the word URL. The regular expression used to match URLs is ((www\.[\S]+)|(https?://[\S]+)).

b. User Mention:

Each twitter user has a handle related with them. Users regularly mention different users in their tweets by @handle. We replace all user mentions with the blank.

a. Hashtag:

Hashtags are unspaced expressions prefixed by the hash (#) which is every now and again utilized by users to specify a trending topic on twitter. We replace all the hashtags with the words with the hash (#). For instance, #bigdata is replaced by bigdata. The regular expression used to match hashtags is #(\S+).

	Total	Unique	Average	Max	Min	Positive	Negative
Tweets	1046150	-	-	-	-	249167	796983
User Mentions	461065	-	0.4407	12	-	-	-
URLs	38515	-	0.0368	4	-	-	-
Emojis	3021	-	0.0029	16	-	1476	1545
Words	7564657	192159	7.2309	28	0		
Bigrams	6524394	2628523	6.2366	-	-	-	-
Trigrams	5510691	4691790	5.2676	-	-	-	-

Table 1: Statistics of preprocessed train dataset

b. Emojis :

Users frequently utilize various emojis in their tweet to pass on different emotions. It is difficult to thoroughly match all the various emojis utilized on social media as the number is ever expanding. However, we match some normal emojis which are utilized as often as possible. In any case, we coordinate some normal emojis which are utilized as often as possible. We replace the matched emojis with either EMO_POS or EMO_NEG relying upon whether it is passing on a positive or a negative emotion. See table 2.

Emojis	Type	Replacement
:), :), :-), (:, (:, (-:, :')	Smile	EMO_POS
:D, : D, :-D, xD, x-D, XD, X-D	Laugh	EMO_POS
;-), ;), ;-D, ;D, (;, (-;	Wink	EMO_POS
<3, :*	Love	EMO_POS
:-(, : (, :(,):,)-:	Sad	EMO_NEG
:,(, :'(, :"(	Cry	EMO_NEG

Table 2: List of emojis matched by our method

c. Retweet:

Retweets are tweets which have been sent by another user and are shared by different users. Retweets start with the letters RT. We expel RT from the tweets as it is not an important feature for text classification. The regular expression used to match retweets is \brt\b.

Subsequent to applying tweet level pre- processing we processed individual words of tweets as follows:

- Strip any punctuation ['"?!,.():;] from the word.
- Convert at least 2 letter redundancies to 2 letters. Some people send tweets like "Heeelloooooo thereeee" including various characters to emphasize certain words. This is done to deal with such tweets by changing them to "Hello there".
- Expel - and '. This is done to deal with words like t-shirt and their's by changing them to the more general structure shirt and theirs.
- Check if the word is valid and accept it only if it is. We define a valid word as a word which starts with a letter with successive characters being alphabets numbers or one of dot (.) and underscore (_).

Some model tweets from the training dataset and their lemmatization versions are shown in table 3:

Raw	is upset that he can't update his Facebook by texting it... and might cry as a result School today also. Blah!
lemmatization	is upset cant update facebook texting might cry result school today also blah
Raw	@Kenichan I dived many times for the ball. Managed to save 50% The rest go out of bounds
lemmatization	dived many time ball managed save rest go bound
Raw	my whole body feels itchy and like its on fire
lemmatization	my whole body feel itchy like fire
Raw	@nationwideclass no, it's not behaving at all. i'm mad. why am i here? because I can't see you all over there.
lemmatization	no behaving all im mad here cant see there
Raw	@Kwesidei not the whole crew
lemmatization	whole crew

Table 3: Example tweets from the dataset and their lemmatization versions.

2. Feature Extraction :

We extract three types of features from our dataset, in particular unigrams, bigrams and trigrams. We create a frequency distribution of the unigrams, bigrams and trigrams present in the dataset and pick top N unigrams bigrams and trigrams for our analysis.

a. Unigrams :

Likely the easiest and the most generally utilized features text classification is the presence of single words or tokens in the content. We extract single words from the training dataset and make a frequency distribution of these words. A total of 192159 unique words are extracted from the dataset. Out of these words, the vast majority of the words at end of frequency spectrum are noise and occur very few times to impact the classification.

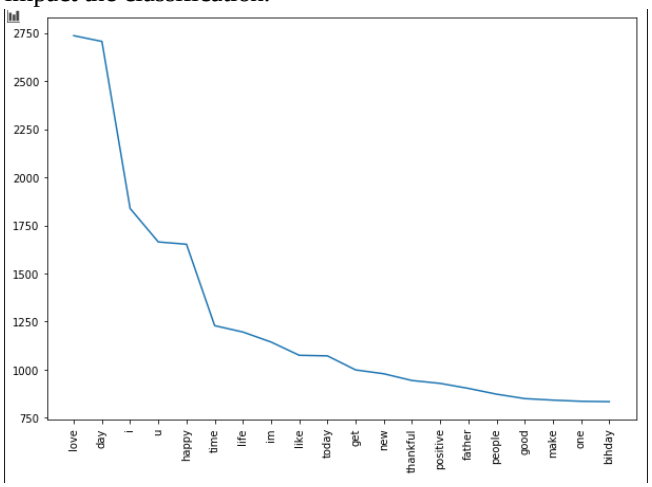

Figure 1: Top 20 unigrams

b. Bigrams :

Bigrams are word pairs in the dataset which happen in succession in the corpus. These features are a good way to show negation in natural language like the expression "I am not fine". A total of 6524394 unique bigrams were extracted from the dataset. Out of these, the vast majority of the bigrams at end of frequency spectrum are noise and occur very few times to impact classification. We in this manner utilize just top 100000 bigrams from these to make our vocabulary.

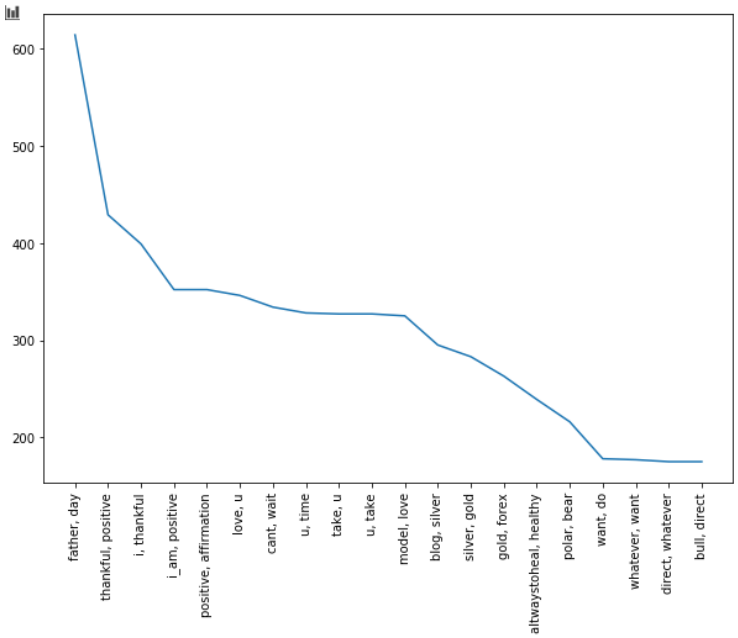

Figure 2: Top 20 bigrams

c. Trigrams:

Trigrams are a group of three consecutive written units such as letters, syllables, or words. A total of 5510691 unique trigrams were extracted from the dataset. Out of these, the vast majority of the trigrams at end of frequency spectrum are noise and occur very few times to impact classification. We in this manner utilize just top 100000 bigrams from these to make our vocabulary.

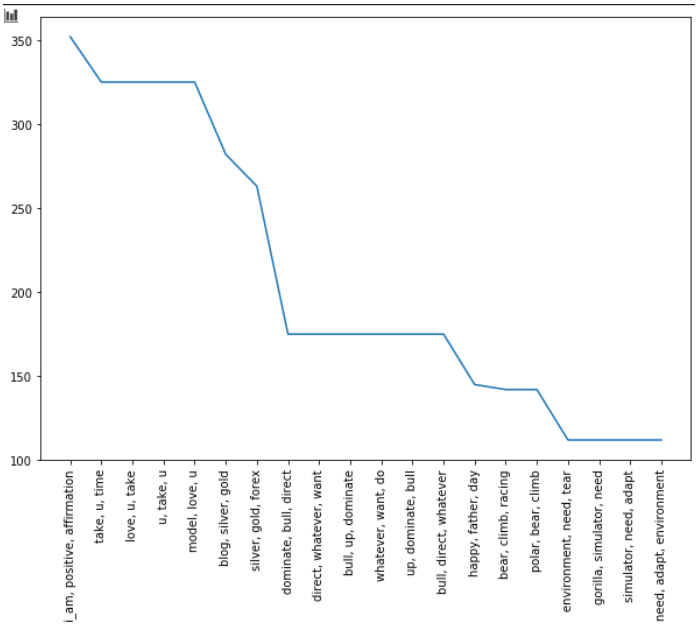

Figure 3: Top 20 trigrams

3. Feature Representation:

After the extraction of the unigrams, bigrams and trigrams, we represent each tweet as a feature vector in either sparse vector representation or dense vector representation relying upon the classification method.

a. Sparse Vector Representation :

Contingent upon whether we are utilizing bigram and trigram features, the sparse vector representation of each tweet is:

- 192159 length when using unigrams only.
- 292159 length when using unigrams and bigrams.
- 392159 length when using unigrams, bigrams and trigrams.

Each unigram, bigram and trigram are given a one of a unique index contingent upon its rank. The feature vector for a tweet has a positive value at the indices of unigrams, bigrams and trigrams which are present in that tweet and zero somewhere else which is the reason the vector is sparse. The positive value at the indices of unigrams, bigrams and trigrams relies upon the feature type we determine which is one of "presence" and "frequency".

- Presence: When using presence, we'll have "1" in the feature vector at indices of unigrams, bigrams and trigrams that are present in a tweet, otherwise it will be "0".
- Frequency: When using frequency, we'll have a positive integer in the feature vector at indices of unigrams, bigrams and trigrams which is the frequency of unigrams, bigrams and trigrams in that tweet, otherwise it will be "0".

A matrix of such term-frequency vectors is created for the whole training dataset and afterward each term frequency is scaled by the Inverse Document Frequency (IDF) to assign higher values to significant terms. The Inverse Document Frequency (IDF) of a term "t" is represented as follows:

$$idf(t) = \log\left(\frac{1+n_d}{1+df(d,t)}\right)+1$$

- nd: Total number of documents.
- df(d, t) : number of documents in which the term t occurs.

One of the issues we faced in this project is Memory Issues, dealing with sparse vector representations, the feature vector for each tweet is of length 392159 and the total number of tweets in the training set is 1046150 which implies allocation of memory for a matrix of size 1046150 × 392159, so you can imagine the amount of memory we would need, which is far greater than the memory available in our mid-end laptops, and obviously it took a lot of time to process the data, so we had to do various modification to adapt the whole project to our available hardware.

b. Dense Vector Representation :

In the case of dense vector representation, we utilize the whole vocabulary of unigrams of size 192159. We assign an integer index to each word contingent upon its rank (beginning from 1), which implies that the most common word is allotted the number 1, the second most common word is allotted the number 2, etc. Each tweet is then represented by a vector of these indices which is a dense vector.

4. Classifiers :

a. SVM:

SVM, otherwise called support vector machines, is a non-probabilistic binary linear classifier. For a training set of points (xi, yi) where x is the feature vector and y is the class.

we need to find the maximum-margin hyperplane that partitions the points with yi = 1 and yi = −1.

The equation of the hyperplane is as follow: $w \cdot x - b = 0$

We need to maximize the margin, signified by γ, as follows:

$$\max_{w,\gamma} \gamma, s.t. \forall i, \gamma \leq y_i(w \cdot x_i + b)$$

so as to separate the points well.

b. *Maximum Entropy:*

Maximum Entropy Classifier model is based on the Principle of Maximum Entropy. The fundamental thought behind it is to pick the most uniform probabilistic model that boosts the entropy, with given requirements. In contrast to Naive Bayes, it doesn't expect that features are restrictively independent of one another. In this way, we can include features like bigrams without worrying about feature overlap. In a binary classification problem like the one we are tending to, it is equivalent to utilizing Logistic Regression to find a distribution over the classes. The model is represented by:

$$P_{ME}(c|d,\lambda) = \frac{exp[\sum_i \lambda_i f_i(c,d)]}{\sum_{c'} exp[\sum_i \lambda_i f_i(c,d)]}$$

Here, c is the class, d is the tweet and λ is the weight vector. The weight vector is found by numerical optimization of the lambdas in order to expand the conditional probability.

c. *Naive Bayes:*

Naive is a basic model which can be utilized for text classification. In this model, the class cˆ is allotted to a tweet "t":

$$\hat{c} = \underset{c}{argmax}\ \mathrm{P(c|t)}$$

$$\mathrm{P(c|t)} \propto \mathrm{P(c)} \prod_{i=1}^{n} \mathrm{P}(f_i|\mathrm{c})$$

In the formula above, fi represents the i-th feature of total n features. P(c) and P(fi |c) can be obtained through maximum likelihood estimates.

d. *Convolutional Neural Networks:*

Convolutional Neural Networks or CNNs are a kind of neural networks which include layers called convolution layers which can interpret special data. A convolution layers has various filters or kernels which it figures out how to separate explicit kinds of features from the data. The kernel is a 2D window which is slided over the input data performing the convolution operation. We utilize temporal convolution in our experiments which is appropriate for examining successive data like tweets.

V. Experiments :

To do our experiments we utilized different classifiers and different packages, for instance, we used "sklearn.naive_bayes" package when performing experiment with Naive Bayes classifier, "MaxentClassifier" from "nltk" library when performing experiment with Maximum entropy classifier, we also used "tweepy", "tensorflow" and some other tools like "anaconda" on python.

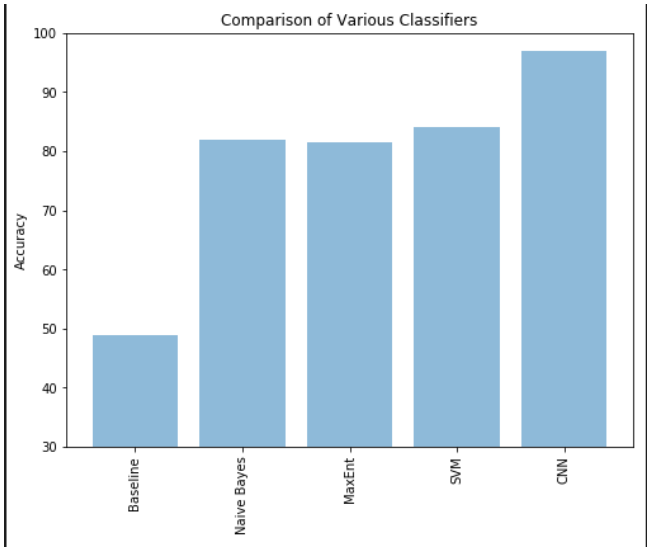

Figure 4: Comparison of various classifiers

1. Baseline:

In the case of Baseline we utilize a simple positive and negative word counting technique to give sentiment to each tweet. In situations when the quantity of positive and negative words are equivalent, we give it a positive

sentiment. We did a classification using emojis, but the results were useless, also did a classification using the method of bad and good words, but the accuracy was not that good. However, utilizing this baseline model, we achieve a classification accuracy of 48.93%.

2. SVM:

We run SVM with both Unigram, Unigrams + Bigrams and lastly Unigrams + Bigrams + Trigrams. We run the configurations with presence and frequency. When using Frequency feature type with 150.000 Unigrams, we got an accuracy of:82.7233%. When using Frequency feature type with all the Unigrams: 192.159, we got an accuracy of:83.0139%. When using Frequency feature type with 100.000 Bigrams, all the Unigrams, we got an accuracy of:84.1266%. When using Frequency feature type with 100.000 Trigrams, 100.000 Bigrams and all the Unigrams, we got an accuracy of:83.9612% When using Presence feature type with all Unigrams, we got an accuracy of: 82.7482%. When using Presence feature type with all Unigrams and 100.000 Bigrams, we got an accuracy of: 83.1745%. When using Presence feature type with all Unigrams, and 100.000 Bigrams and 100.000 Trigrams, we got an accuracy of: 83.2452%. We concluded that the feature type Frequency we achieved the best accuracy when using all the Unigrams and 100.000 Bigrams.

3. Naive Bayes:

In the case of Naive Bayes we utilize sparse vector representation for classification and ran experiments using both presence and frequency feature types, with both Unigram, Unigrams + Bigrams and lastly Unigrams + Bigrams + Trigrams. When using Frequency feature type with all words 192.159 Unigrams, we got an accuracy of: 78.9198%. When using Frequency feature type with only: 150.000 unigrams, we got an accuracy of: 80.3451%, we did also an experiment with only 100.000 unigrams and it had the best result as 80.4569 % of accuracy for only unigrams, so we kept it. When using Frequency feature type with 100.000 Bigrams and 100.000 Unigrams, we got an accuracy of: 82.0102%, and when adding 100.000 of trigrams we got an accuracy of: 81.5925%. After that we did the experiment on Presence feature type, with 100.000 Unigrams we got an accuracy of: 81.8439%, when adding 100.000 Bigrams we got an accuracy of 81.5428%, and finally we added 100.000 trigrams, we got an accuracy of 81.4147%. We concluded that the feature type Frequency we achieved the best accuracy when using all the 100.000 Unigrams and 100.000 Bigrams.

4. Maximum Entropy:

In the case of Maximum entropy, we utilize sparse vector representation for classification and ran experiments using both presence and frequency feature types, with both Unigram, Unigrams + Bigrams and lastly Unigrams + Bigrams + Trigrams. When using Frequency feature type with all words 192.159 Unigrams, we got an accuracy of: 77.4965%. When using Frequency feature type with 100.000 Bigrams and 192.159 Unigrams, we got an accuracy of: 81.4797%, and when adding 100.000 of trigrams we got an accuracy of: 80.4965%. After that we did the experiment on Presence feature type, with 192.159 Unigrams we got an accuracy of: 81.8439%, when adding 100.000 Bigrams we got an accuracy of 81.5428%, and finally we added 100.000 trigrams, we got an accuracy of 81.4147%. We concluded that the feature type Frequency we achieved the best accuracy when using all the 192.159 Unigrams and 100.000 Bigrams.

5. Convolutional Neural Networks:

For CNN (Convolutional Neural Networks), We utilized keras with TensorFlow backend to execute the Convolutional Neural Network model. We utilized the dense vector representation of the tweets to train our CNN models. We utilized a vocabulary of top 80000 words from the training dataset in the first try, after that we started incrementing the numbers more and more to get better results, so we ended up using all the words (vocab_size = 192160). We represent each word in our vocabulary with a number from 1 to 192160, each number represents the rank of the word in the dataset. The number 0 is reserved for the special padding word. Every one of these 192160+1 words is represented by a 200-dimensional vector. The first layer of our models is the Embedding layer which is a matrix represented as (v+1)×d, "v" is vocabulary size (=192160) and "d" is the dimension of each word vector (=200). We instate the embedding layer with random weights from N(0, 0.01). Each row of this embedding matrix represents the 200-dimensional word vector for a word in the vocabulary. For every word in our vocabulary that matchs GloVe word vectors provided by the StanfordNLP group, we seed the relating row of the embedding matrix from GloVe vectors. GloVe is an unsupervised learning algorithm for obtaining vector representations for words. Training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space. Each tweet for example, its dense vector representation is padded with 0s toward the end until its length is equivalent to max_length. In order to train our model, we utilized binary cross entropy loss with the weight update scheme being the one defined by Adam et. al. We likewise did the experiments using SGD + Momentum weight updates and discovered that it takes longer (≈100 epochs) to converge compared to

validation accuracy identical to Adam. We also ran our model up to 10 epochs. Utilizing the Adam weight update scheme, the model converges extremely quick (≈6,7 epochs) and starts to overfit severely after that, where croos validation error increases and training error approaches zero.

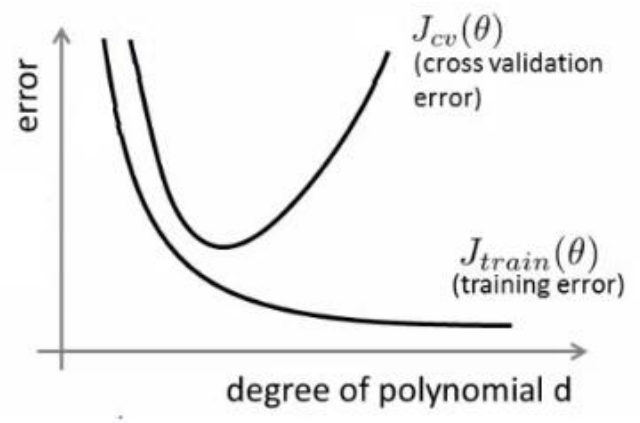

Figure 5: Croos validation error && training error

We, consequently, use models from 6 or 7th epochs for our results. We attempted four different CNN architectures which are as follows:

- 1-Conv-NN: As the name proposes, it's an architecture with 1 convolution layer. We did a temporal convolution with a kernel size of 3 and 0 padding. Afterward, we apply relu activation function (as follows: f(x) = max(0,x)), thereafter, we performed Global Max Pooling after some time to lessen the dimensionality of the data. Ans then, we pass the result of the Global Max Pool layer to a fully-connected layer which at that point outputs a single value which is passed through sigmoid activation function to convert it to a probability value. We utilize a tweet max_length of 40 in this system with a vocabulary of 80000 words, then 90000 words and after that 192160 word. The complete architecture of the system is embedding_layer (192161×200) → dropout(0.2) → conv_1 (600 channels) → relu → global_maxpool → dense(500) → relu → dropout(0.2) → dense(1) → sigmoid.

- 2-Conv-NN: In the case of the 2-Conv-NN which is an architecture with 2 convolution layers, we utilized quit the same architecture as 1-Conv-NN, but me added +1 convolution layer with 300 filters.

- 3-Conv-NN: In the case of the 3-Conv-NN which is an architecture with 3 convolution layers, we tried again to add +1 convolution layer and also sweep all the possibilities (by changing the number of filters, the activation function and the size of the kernel), by we couldn't get a better results that the 2-Conv-NN.

- 4-Conv-NN: In the case of the 4-Conv-NN which is an architecture with 3 convolution layers, we utilized quit the same architecture as 2-Conv-NN, but me added +2 convolution layer firstly with 150 filters and then with 75 filters.

We notice that 4-Conv-NN and 2-Conv-NN gives us the same accuracy but not the same cross validation error which is 0.1501 of error for 4-Conv-NN and 0.1842 of error for 2-Conv-NN. The accuracy of each architecture is as follow:

- 1-Conv-NN (90000 words): 0.9394.
- 1-Conv-NN (All words): 0.9621.
- 2-Conv-NN: 0.9607.
- 3-Conv-NN: 0.9545.
- 4-Conv-NN: 0.9697.

As a conclusion, we take 4-Conv-NN as the best architecture.

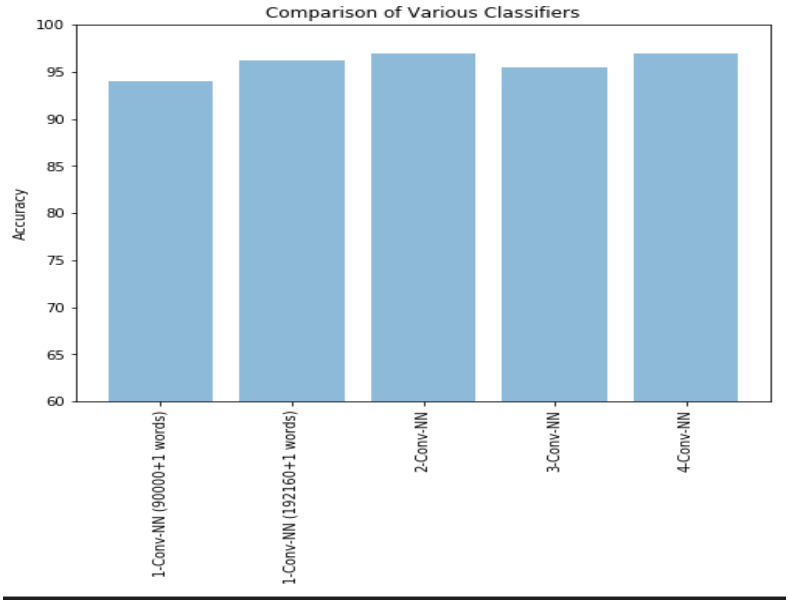

Figure 6: Comparison of CNN architectures

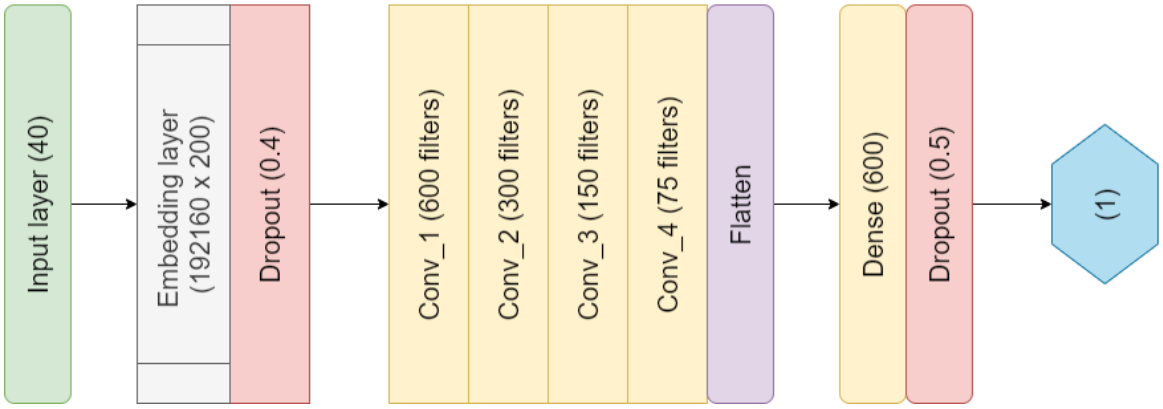

Figure 7: Neural Network Architecture with 4 Conv Layers

Algorithms	Presence			Frequency		
	Unigrams	Unigrams + Bigrams	Unigrams + Bigrams + Trigrams	Unigrams	Unigrams + Bigrams	Unigrams + Bigrams + Trigrams
Naive Bayes	81.8439	81.5428	81.4147	78.9198	82.0102	81.5925
Max Entropy	81.8439	81.5428	81.4147	77.4965	77.4965	80.4965
SVM	82.7482	83.1745	83.2452	83.0139	84.1266	83.9612

Table 4: Comparison of various classifiers

VI. Conclusion :

We attempted to build a sentiment analysis system by understanding and implementing algorithms of machine learning. We implemented SVM (support vector machines), Naive Bayes, Maximum entropy and CNN (Convolutional Neural Networks) algorithms. Baseline model was the poorest with no doubt because it had least number of features. We used 3 types of features, unigrams, bigrams and trigrams for classification, and the conclusion was that increasing the feature vector with bigrams and sometimes trigrams also gave us a better accuracy. Thereafter, when the feature was extracted it was represented as sparse vector or a dense vector, and we concluded also that Neural methods performed better than other classifiers. Sentiment analysis system is an active field of research and we can always improve our algorithms by performing more improvements on the algorithms, evaluating various things and ideas in preprocessing to get best accuracy.

VII. References:

[J. Passonneau 11] Rebecca J. Passonneau, " Sentiment Analysis of Twitter Data", Department of Computer ScienceColumbia UniversityNew York, NY 10027 USA, 2011, https://www.researchgate.net/publication/247935218_Sentiment_Analysis_of_Twitter_Data

[emcamara 12] EUGENIO MART´INEZ-C ´AMARA, " Sentiment Analysis in Twitter", Computer Science Department, University of Ja´en, Campus Las Lagunillas, 23071 Ja´en, Spainemail, 2012, https://www.researchgate.net/publication/234052505_Sentiment_analysis_in_Twitter

[Boguslavsky 17] "Semantic Descriptions for a Text Understanding System. In Computational Linguistics and Intellectual Technologies. Papers “Dialogue” (2017)

[Rosenthal 17] S., Farra, N., & Nakov, P. (2017) SemEval-2017 task 4: Sentiment analysis in Twitter.

[Ortigosa 14] A., Martín, J. M., & Carro, R. M. (2014). Sentiment analysis in Facebook and its application to e-learning, https://www.aaai.org/ocs/index.php/ICWSM/ICWSM11/paper/download/2857/3251

[Hamid Bagheri 14] Computer Science Department Iowa State University (2014). Sentiment analysis of twitter data. https://arxiv.org/ftp/arxiv/papers/1711/1711.10377.pdf

[Deepali 18] Shaunak Joshi, Deepali Deshpande, Department of Information Technology Vishwakarma Institute of Technology (2018). Twitter Sentiment Analysis System. https://arxiv.org/ftp/arxiv/papers/1807/1807.07752.pdf

[Johan Bollen 11] Johan Bollen, Huina Mao, and Alberto Pepe, "Modeling public mood and emotion: Twitter sentiment and socioeconomic phenomena."

[Nabizath Saleena 18] Nabizath Saleena, An Ensemble Classification System for Twitter Sentiment Analysis, International Conference on Computational Intelligence and Data Science (ICCIDS 2018), https://www.sciencedirect.com/science/article/pii/S187705091830841X

[Razia Sulthana 18] Razia Sulthana, A K Jaithunbi, L Sai Ramesh, Department of Information Technology, SRM University, https://iopscience.iop.org/article/10.1088/1742-6596/1000/1/012130/pdf

[Vishal A. Kharde 16] Vishal A. Kharde, S.S. Sonawane, Department of Computer Engg, Pune Institute of Computer Technology,Pune, https://www.ijcaonline.org/research/volume139/number11/kharde-2016-ijca-908625.pdf