Computer Science Faculty

Twitter Sentiment analysis

Assigned by: Naor Maoz, Eliyahu Peretz, Reut Kishon

Id: 204670723, 206226706, 206966517

Faculty advisor: Professor Lee-Ad Gottlieb

Abstract

To classify the sentiments that are expressed in a text is one of the most studied topics in the field of Natural Language Processing. Nowadays, this task can be addressed using modern techniques such as Bernoulli Naive Bayes, SVM and Logistic Regression,

But over the years different innovative methods have been developed.

In this report, we use tweets from Twitter as our dataset. The models that we use are Transformers, BERT, GPT 2 , and RoBERTa.

Specifically, we preprocess the data and convert the data format (i.e. text) into vectors of real numbers using GloVe, Tf-Idf, Word2Vec then we analyze their impact on the results of each model.

We evaluate each of the models in terms of accuracy and loss, and we obtain results that vary between 0.5 and 0.76 depending on the model and configuration used.

Introduction

In recent years there has been a large increase in the use of sentiment analysis. one the biggest usages is product and service reviews.

There is a great amount of user-generated data on social media platforms and websites. Customers share their thoughts, feedback, and expectations regarding companies' services and products. All of these types of content give companies important information for analyzing their brand reputation, services, and products.

Sentiment analysis comes into play here. It converts unstructured text into data that can be analyzed. Sentiment analysis is a NLP technique used to determine whether data is positive, negative or neutral.

In our project we classify tweets from Twitter to those categories we mentioned above.

We present different models trained on Twitter dataset detailed in Section III.

We explore which text representation gives better results. The Doc2Vec approach has the ability to group together vectors of similar words , TF-IDF can associate

each word in a text with a number that represents how relevant each word is in that text. Such numbers can be then used as features of the models we use..
Then we evaluate some popular models for text classification with each text representation :

PROBLEM DEFINITION

Given a dataset divided into 2 different classes, for each tweet in the dataset, the goal is to predict the class it belongs to. To do so, we implement and evaluate four different machine learning models.
In each case, the model is trained with a subset of tweets, and tested with unseen tweets to validate the performance by means of the accuracy and loss. In Fig. 1 we can see a simplified scheme of the system.

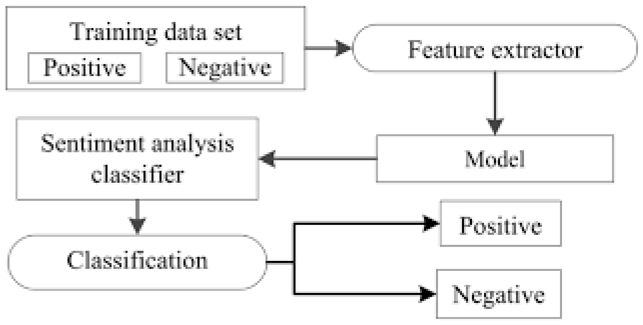


Fig. 1

Preprocessing:

A process of cleaning and preparing the text for classification. Online texts usually contain lots of noise and uninformative parts. Reducing the noise in the text should help improve the performance of the classifier and speed up the classification process. All the operations in this section are executed to try to make the text uniform. This is important because during the classification process, features are chosen only when they exceed a certain frequency in the data set. Therefore,
after the basic preprocessing operations, having different words written in the same way helps the classification.

cleaning unnecessary characters - removing URLs , hashtags (i.e #happy) or mentions (i.e @mickeymouse) .
Capitalization - Text often has a variety of capitalization like the beginning of sentences.
The most common approach is to reduce everything to lower case for simplicity , but it's important to remember that some words, like "US" to "us", can change meanings when reduced to the lower case.

Stemming- A process where words are reduced to a root by removing inflection through dropping unnecessary characters, usually a suffix. for example -
program -> program
programs -> program
programmer -> program
programming -> program
programmers -> program

Text vectorization
convert the processed text to numeric feature vectors so that we can feed it to Machine Learning models. we checked three techniques of text vectorization:

bag-of-words: a very simple technique that represents the text by counting occurrences for every word in the text and stores it in a vocabulary vector disregarding the grammatical details and the word order. Machine learning models can only be fed with fixed length input. Hence, every word is represented by a numerical vector whose length is the total number of words in the text, such that it assigns 0 for words in the vector who don't appear in some tweet.
In addition , we can tune the range of a token in the text , it's called n-gram.
For example, the text - "this is not good at all" , its vocabulary vector can be : {this, is, not, good, at, all} or if we take the range of two words each time , it'll be {this is , is not , not good , good at , at all}.

Tf-Idf (term frequency-inverse document frequency): this technique is based on bag-of-words, but the difference is it rescales the frequency of words in the vocabulary vector by how often they appear in all tweets so that the scores for frequent words like “the” that are also frequent across all tweets are penalized.
Calculation: $TF(i,j) = n(i,j)/\Sigma\, n(i,j)$ where,
n(i,j)= number of times nth word occurred in a document
Σn(i,j) = total number of words in a document.
$IDF(i,j) = log(N/dN)$ where , N=Total number of documents in the dataset
dN=total number of documents in which the nth word occurs.
The TF-IDF is obtained by: TF-IDF=TF*IDF

Doc2Vec: For each unique word in the corpus assigned a corresponding vector in the space. Word vectors are positioned in the vector space in a way such that words that have common contexts in the corpus are located close to one another in the space.
Moreover, it allows us to use vector arithmetics to work with analogies, for example:
king - man + woman = queen.
With the Word2Vec model, we can calculate the vectors for each word in a document. But we want to calculate a vector for the entire text. We could average the vectors for each word in the sentence.

Machine learning

supervised learning

Algorithms in that group are feeded with training data which contains inputs and the corresponding desired outputs. The supervised learning models should be able to learn the patterns, in order to predict the output correctly of a new unseen data.

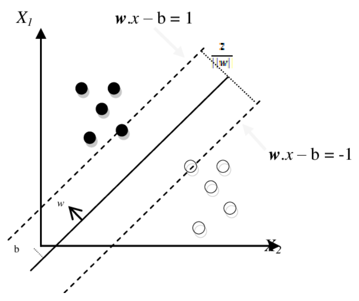
In our research we used common supervised learning algorithms that suit classification tasks,
in order to compare with the advanced algorithms we'll discuss later.
We examine SVM , Naive Bayes , Logistic regression , Cnn , KNN, Random Forests.

Naive Bayes

Let X=(x1,x2,…,xk) be a given input with k features and let Y be the label we want to predict.
The prediction is as follow: $max\,\{\, P(Y = y \mid X = (x1, x2,.. xk))\,\}$.
In other words , the prediction will be y which gives high probability of the above condition.
calculation: $P(Y|X) = \frac{P(X|Y)*P(Y)}{P(X)}$
Theoretically, it is not hard to find P(X|Y). However, it is much harder in reality as the number of features grows.
To solve this problem, a naive assumption is made. We pretend all features are independent. With the help of this naive assumption , we can make classification with much fewer parameters.
that is , if $X1\ and\ X2\ are\ independent\ ,\ P(X1, X2 \mid Y) = P(X1|Y) * P(X2|Y)$
$P(X = (x1, x2,.., xk) \mid Y = y)$ is called a parameter in Naive Bayes. hence if $xj\ \forall j$ is binary , there are $2^{k+1} - 1$ parameters.
With the assumption there are $2k$ parameters!
the number of parameters has changed from exponential to linear. This means that Naive Bayes handles high-dimensional data well which is very important in our case as we have a very high number of features.

SVM

SVM performs classification by finding the hyper-plane that differentiate the classes we plotted in n-dimensional space. SVM draws that hyperplane by transforming the data with the help of mathematical functions called "Kernels"**.** Types of Kernels are linear, sigmoid, RBF, non-linear, polynomial, etc. we use the linear Kernel - which is for linear separable problems. Since our problem is linear(just positive and negative) here, we will go for “linear SVM”.



.

Logistic Regression

This model is used to calculate or predict the probability of a binary event occurring. In our case positive/negative. Logistic regression assumes that the factors, or the independent variables, that influence the outcome are independent of each other, and the dataset is relatively big.

Logistic regression uses an equation as the representation.

Input values (x) are combined linearly using weights or coefficient values to predict an output value (y). A key difference from linear regression is that the output value being modeled is a binary value (0 or 1) rather than a numeric value.

Below is an example logistic regression equation:

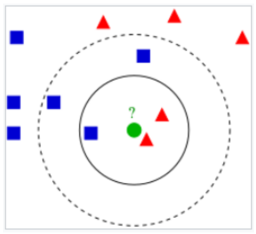
y = e^(b0 + b1*x) / (1 + e^(b0 + b1*x))

Where y is the predicted output, b0 is the bias and b1 is the coefficient for the single input value (x). Each column in your input data has an associated b coefficient (a constant real value) that must be learned from your training data.

Knn:

A nearest neighbor algorithm assumes that similar points will be close to each other.

Therefore, given a point and K, we will check the K nearest points in the feature space (by calculating the distances) and determine the classification of the new point according to the class majority.



Unsupervised learning

Learning where data is *not* labeled and the motivation is to find patterns in given data. In this case, you are asking the machine learning model to process the data from which you can then draw conclusions.

BERT:
Bidirectional Encoder Representations from Transformers is a transformers-based machine learning technique for natural language processing pre-training developed by Google.
Using this technique yields much better results than the left-to-right model. The model basically reads the sequence of words at once and learns the context according to the environment of each word (on the left and right together).

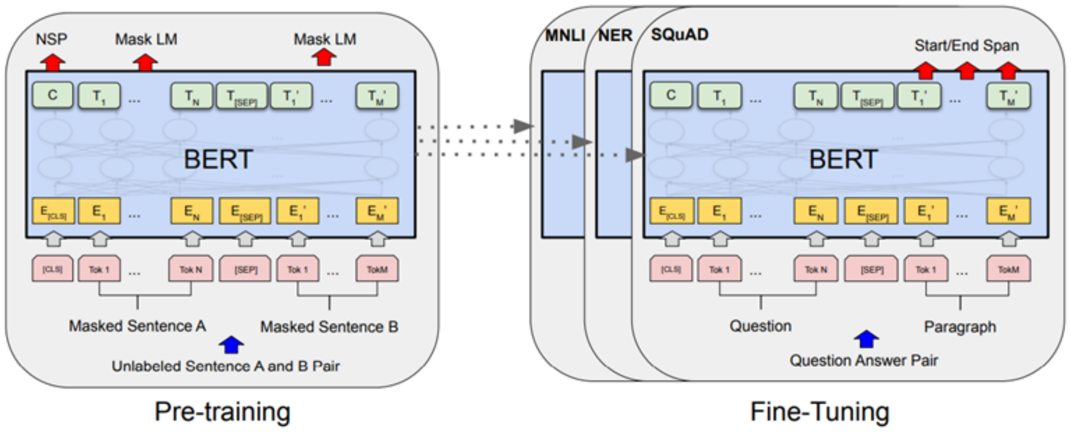
The model is divided into 2 stages:

1. Pre-training

    · Masked LM

    · Next Sentence Prediction

2. Fine-Tuning

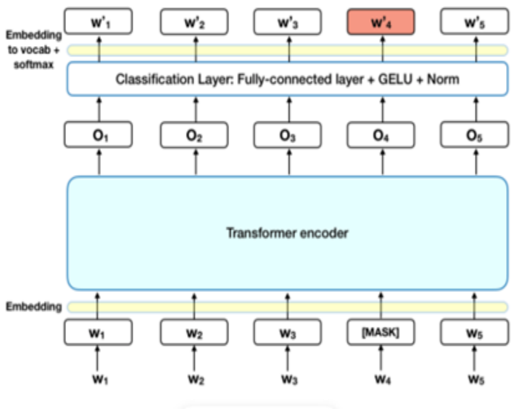


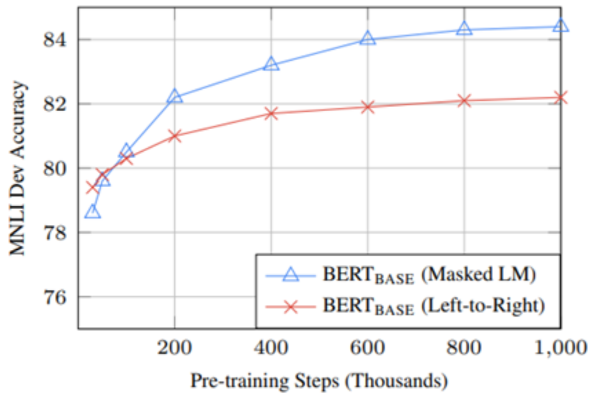
<u>Pre-training</u>

<u>Masked LM</u>

The Masked LM is a new technique that allows you to perform bidirectional training. A certain percentage(15%) of the words in the sequence will be randomly marked as MASK tokens and we will try to predict only the MASK values by the context of the words that are non-masked.

In addition, we will also use a classification layer and SoftMax to predict the output words.


W'1
W'2
W'3
W'4
W'5
Embedding to vocab + softmax
Classification Layer: Fully-connected layer + GELU + Norm
O1
O2
O3
O4
O5
Transformer encoder
Embedding
W1
W2
W3
[MASK]
W5
W1
W2
W3
W4
W5



Next Sentence Prediction

Understanding the relationship between 2 sentences, we will train the model so that it knows whether sentence A is after sentence B or not. When training the model 50% of the time. B will be the next sentence after A (isNext) and 50% it will be a random sentence from the corpus (NotNext).

Calculate the probability (if IsNext or NotNext) using SoftMax.

Fine-Tuning

To perform classification tasks, we will need a classification layer at the end of the transformer similar to Next Sentence Prediction

Database extension:

We wanted to study if it's possible to achieve better results predicting the sentiment of text while combining the vectorized text with other arbitrary data.

In our Twitter database there is only a date column except the text column , so that’s the only information that we can relate to. But the date of a tweet can tell a lot and can help us with our mission. In some research, it was found that people tend to get depressed in the winter.

Seasonal affective disorder (SAD) is a type of depression that's related to changes in seasons. SAD begins and ends at about the same times every year. Basically, the symptoms start in the fall and continue into the winter months, sapping your energy and making you feel moody.

Some people experience depression at night. Work, school, or social activities act as a distraction during the day. But at night, when you settle down to sleep, there’s nothing but you and your thoughts.

In addition , findings from the University of Rochester study on the "weekend effect" determined that people felt happier on the weekend largely because "weekends were associated with higher levels of freedom and closeness”. and specially in Israel , soldiers return back home on the weekend which is a real freedom.

Hence, the day, time (morning/night) and the season very affect a person's mood and feelings and as a result affect the tweet content that he writes.

We remove the date column and add a few new columns instead: day in month , day in the week, part of the day (morning/afternoon/night).

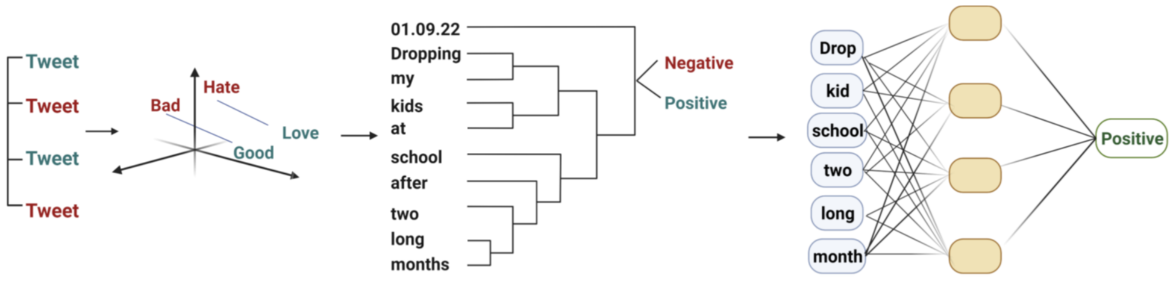
We handle the non-text features by converting them into text (in a meaningful way) and concatenating them with the tweet text.

For example , "PROUD 2 BE A MOTHER I HAVE MY SON &amp; HIS COUSIN OVER SPENDING A NITE, IT'S A BOYZ NITE, &amp; THEY DRIVING ME CRAZY BUT I LUV EM.."

The date that this tweet was written is Sat May 09 18:15:28 PDT 2009

.

We can change the tweet to be: "Now it's evening. today is Sat, 09th of May. Today is the weekend. PROUD 2 BE A MOTHER I HAVE MY SON &amp; HIS COUSIN OVER SPENDING A NITE, IT'S A BOYZ NITE, &amp; THEY DRIVING ME CRAZY BUT I LUV EM.."
And then treat this problem as we did. (just text feature).



Results and conclusions

<u>Sentiment analysis of the tweets without any additional features:</u>

Models / Text vectorization	Knn	SVM	Naive Bayes	Logistic Regression
Bag-of-words	66.03%	74.23%	74.94%	76.1%
Tf-Idf	57.725%	75.96%	75.295%	77.25%
Doc2Vec	50%	68.3%	70.12%	73.76%

Bert:77.515%

Sentiment analysis of the tweets with additional features:

Models / Text vectorization	Knn	SVM	Naive Bayes	Logistic Regression
Bag-of-words	69.175%	77.875%	78.09%	78.715%
Tf-Idf	70.59%	79.315%	79.045%	80.54%
Doc2Vec	62.45%	73.11%	74.67%	75.32%

Bert: 80.66%

Analyzing

From the results above we can see that the prediction performance improves when we use TfIdf as a vector representation.
Bag-of-words and Tf-Idf ignore the location information of the word.
The location information is a very important attribute in the text.
For example: “today is off” and “Is today off”, have the exact same vector representation in these models.
Another thing is they don’t take into account the semantics of the words.
For example, the words ‘soccer’ and ‘football’ are often used in the same context. However, the vectors corresponding to these words are quite different in these models in contrast to Doc2Vec. but yet, doc2vec doesn't work well on short sentences and has a very slow computation.
The Logistic Regression algorithm has performed relatively well compared to the other supervised algorithms.
Naive Bayes algorithm makes many assumptions that aren’t necessarily held, as its predictions are more probabilistic then other models .

The KNN algorithm doesn’t work well with high dimensional data, it becomes difficult for the algorithm to calculate the distance in each dimension.
SVM has a similar accuracy like logistic regression, but it shows a higher improvement rate at predicting as the SVM can achieve better prediction as we increase the data points.
We used the F1 score to measure the success rate in our algorithms where it is measuring the balance between the True Positive vs the False Positive and False Negatives.
The equations is: $\frac{TP}{TP+(1/2)+(FP+FN)}$.
The Bert transformer was the unknown factor in our research as it predicts words according to the relations of the word by their location in the sentence which complicates our theory and the algorithm is implemented in a complete and independent way which required a different aspect of implementation thats why its not a part of the table as its has its own embedding and vectorization of the text.
But even after the unique implementation we managed to improve the model predictions with the data we added.

Conclusion

In this paper we have explored both Natural language processing and machine learning algorithms, from Stemming and Lemmatization to Vectorization algorithms such as Tf-idf, Bag of Words and Doc2Vec.
Using those algorithms we were able to translate our data to a numerical representation which can serve as an input for the machine learning algorithms to learn and predict sentiment analysis on the Twitter dataset.
We have examined the differences between different approaches in our research where unrelated algorithms from separated branches in the machine learning algorithms are all examined whilst still proving our theory that the prediction improves with the new data that is added to the vectorized text.
We succeeded with each and every algorithm and improved the accuracy by only using the additional information of the tweets date.

What we extracted was the day in the week,day in the month and the period of time in the day.
All those three attributes were enough to improve the predictions with our algorithms.
We are sure that if the data was more varied and informative (e.g, taking tweets from all months, information about the place the tweet was written) we could have achieved even better results.
We managed to improve the prediction of one of the best methods known(Bidirectional Encoder Representations from Transformers - BERT) in the Natural Language Processing and Machine Learning worlds with our study.

Bibliography

- Ahmad, M., Aftab, S., Salman, M., & Hameed, N. (2018). Sentiment Analysis using SVM: A Systematic Literature Review. *International Journal of Advanced Computer Science and Applications*, *9*(2). https://doi.org/10.14569/ijacsa.2018.090226
- Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *Proceedings of the 2019 Conference of the North*, *1*, 4171–4186. https://doi.org/10.18653/v1/n19-1423
- Goyal, G. (2021, June 11). *Twitter Sentiment Analysis- A NLP Use-Case for Beginners*. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2021/06/twitter-sentiment-analysis-a-nlp-use-case-for-beginners/
- Great Learning Team. (2020, September 11). *An Introduction to Bag of Words (BoW) | What is Bag of Words?* My Great Learning. https://www.mygreatlearning.com/blog/bag-of-words/
- Horev, R. (2018, November 17). *BERT Explained: State of the art language model for NLP*. Towards Data Science. https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270
- Zhang, Z. (2019, August 14). *Naive Bayes Explained*. Towards Data Science. https://towardsdatascience.com/naive-bayes-explained-9d2b96f4a9c0