Sentimental analysis Twitter

Wilber Bermeo Quito

July 2023

1 Introduction

The goal of this project is to create some models for sentimental analysis given a training set and two test set. One of the test sets is from the same domain from the training set but the other is focused on the Dublin city.

2 Premises

Technologies

The main technologies used to solve the home assignment are:

- Python
- Jupyter Notebook
- Google Collab
- Sklearn
- Pytorch

Source code

The source code of the project can be found on GitHub on the following repository: `https://github.com/wilberquito/sentimental-analysis-twitter/tree/main`.

The repository is distributed as:

- **ml**: python package that mainly work with Sklearn and base models
- **nn**: python package that mainly work with Pytorch and deep learning
- **utility**: python package with transversal functions

- **results**: results exported as csv files of all different trained models
- **assigment.data.zip**: zip file with all the csv data in it
- **SentimentalAnalysisTwitter.ipynb**: the work notebook

Models

The recommended base models to solve the problems are `Logistic Regression` or `Support Vector Classifier (SVC)` using `TF-IDF` or `Bag-of-Words (BoW)` with `L2 regularization`. Instead of training them separately, I employed `Cross Validation` with `SearchGrid` and `SGDClassifier` with `L2 regularization`. By tuning the loss function of `SGDClassifier` with `hinge` and `log_loss`, it emulates the behavior of `Logistic Regression` or `SVC`.

For the deep learning models, I utilized the pre-trained weights of `TinyBERT` since fine-tuning other alternatives proved to be challenging, meaning that the time spend is much more thant this alternative, of course at the cost of lossing high metrics performance.

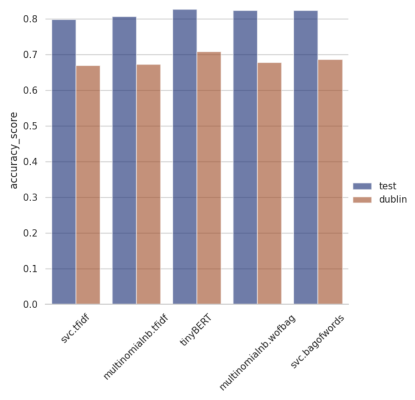
3 Research questions

What are the accuracy/f1/recall/precision of your model on the Sentiment140 dataset? And on the Dublin dataset? (Provide the metrics measurement for each dataset)

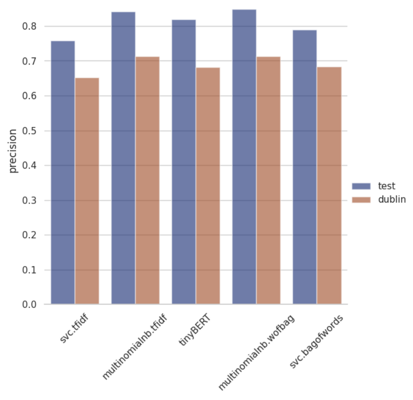
The table bellow show all metrics results for each model - dataset. After the table I'll show a graphic that illustrate clearly the model agains dataset performance.

dataset & model	precision	recall_score	f1_score	accuracy_score
S140-test.svc.tfidf	0.759	0.885	0.817	0.799
dublin.multinomialnb.tfidf	0.713	0.584	0.642	0.674
dublin.tinyBERT	0.693	0.7	0.697	0.695
S140-test.multinomialnb.wofbag	0.848	0.797	0.822	0.825
dublin.svc.bagofwords	0.683	0.697	0.69	0.687
dublin.multinomialnb.wofbag	0.713	0.595	0.649	0.678
S140-test.svc.bagofwords	0.79	0.89	0.837	0.825
S140-test.multinomialnb.tfidf	0.842	0.764	0.801	0.808
dublin.svc.tfidf	0.652	0.732	0.69	0.67
S140-test.tinyBERT	0.819	0.846	0.832	0.827

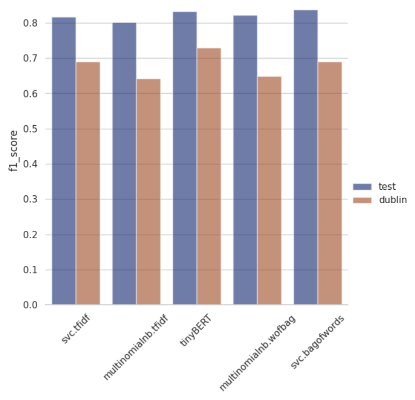
Table 1: Table of model dataset metrics performance.



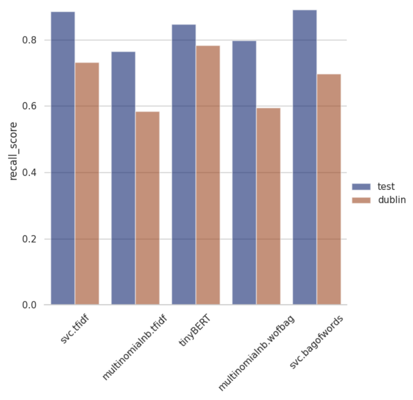
(a) Accuracy performance dataset & model.



(b) Precision performance dataset & model.



(c) F1 score performance dataset & model.



(d) Recall performance dataset & model.

Figure 1: Overall metrics performance dataset & model.

Are they or are they not the same? How large is the difference? Can you think of the cause? How would you corroborate or refute your hypothesis?

The results of the metrics are not the same for the test dataset and the Dublin test dataset. The Dublin test dataset represents a small distribution from a specific region, consisting of tweets from the outskirts of Dublin. The frequency distribution of words in this dataset differs from that of the test dataset. It's important to note that the test dataset is actually a subsample from the training dataset.

One way to corroborate this is by utilizing the `wordcloud` library, which displays the `n` most frequent words in a dataset. I observed that the most common words in the Dublin dataset are related to topics specific to that particular region. Another way to confirm this is through the inference of models on the test dataset. The results obtained from the trained dataset are similar, but when applied to the Dublin dataset, the models do not generalize as well compared to the other distribution.

What are the main classification challenges?

As the dataset is already balanced, I didn't need to search for solutions to handle imbalanced data such as stratification or under-sampling and over-sampling.

One of the key points in resolving this challenge was to notice that this is a binary classifier problem, even if the test datasets are multiclass. To obtain good working datasets for training and the two test dataframes, I needed to normalize the polarity of the test samples by dropping the "neutral" samples from the test set. Additionally, I mapped the polarities of all datasets to "Negative" $\Rightarrow$ 0 and "Positive" $\Rightarrow$ 1.

Another key point is to notice that there were some repeated entries with the same tweet and polarity. This adds noise, so I dropped out this repeated samples.

For the base models, I made use of `Sklearn` algorithms and, of course, the `GridSearch` that performs a default cross-validation. The challenge asked to try one of the base models, either `Logistic Regression` or `SVC`. I accomplished this by using the wrapper algorithm `SGDClassifier` and tuning its hyperparameter, the loss function. Each of these models was trained using both "Bag of Words" and "TFIDF" techniques.

In addition to using these basic algorithms, I also employed the `MultinomialNB` algorithm, which is a popular Bayesian learning approach in Natural Language Processing. Although it was not a requirement, I chose to use it because I had previously implemented a sarcasm detector, and this algorithm proved to be highly efficient and accurate in that case. Once again, I trained two `MultinomialNB` models, one using "Bag of Words" and the other using "TFIDF".

All those previous algorithms support working with sparse matrices, which helped me a lot because all tweets were converted to large `numpy` arrays. I initially assumed that I should follow the same approach with the `TinyBERT` model, but it took me some time to realize that it was not necessary to work with sparse matrices or sparse tensors using `PyTorch`. Instead, these models already support working directly with plain text.

Can you think of a way to measure whether the accuracy is higher for some of the project categories instead of others, e.g. whether sentiment classification accuracy is higher for the Public Spaces category than for Community and Culture?

There are several approaches to measure whether the accuracy is higher for any category. First, calculate the accuracy, precision, recall, and F1-score for each topic category based on the predictions of the classifier. These metrics will provide insights into the classifier's performance for each individual category.

Next, I can compare the measurements visually by plotting them or using appropriate visualizations. This can help me identify any patterns or differences in performance among the categories.

Additionally, I can apply a pairwise statistical analysis, such as a t-test, to determine if there are significant differences in the performance metrics between the categories. The t-test can assess whether the differences in means of the performance metrics are statistically significant.

By conducting a statistical analysis, I can quantify and validate any observed differences in performance between categories.

Can you think of any way to measure whether the metrics are higher for documents containing specific types of entities, e.g. Person vs. Location?

This has a straightforward solution, but first, I need to determine which types of entities I should search for. This step is crucial in text processing. If the text contains those entities, I can consider different approaches to measure their impact.

One approach is to calculate the frequency of the entity within the entire text. This can involve counting the absolute number of occurrences or determining if the entity appears at least once. Another approach is to calculate the relative frequency, such as the proportion of the entity's occurrences compared to the total number of words in the text.

Next, I would evaluate the performance metrics of interest, including accuracy, precision, recall, and F1-score, separately for each entity type. These metrics can be calculated based on the predictions of the classifier.

Finally, to analyze the significance of the differences between entity types, I can apply a statistical test such as a pairwise t-test. This test will help determine if the performance metrics significantly differ between entity types.

Can you make a proposal to address any/all the issues you have encountered in the evaluation?

I did not encounter many problems while evaluating any model. However, if I have to mention some of them, I would say that one issue was the mapping of the polarities of the test sets. Additionally, since I was working with "Google Colab," there was a risk of losing the session and consequently losing the trained models. To address this issue, I resolved to train the models and export the results into CSV files, with each model corresponding to each test dataset.

Another challenge I faced was with the transformer model, as it took a significant amount of time to train (only 3 epochs). To overcome this, I decided to download its weights using the `PyTorch` API and then perform inference by loading the transformer weights. The results were then exported to CSV file.

Once all the results were exported, I created a simple script to load them and generate a comprehensive dataframe of results. This dataframe was then utilized to perform comparisons, both per model and per dataset, to determine the final comparisons.

Can you bring any idea of new feature to develop for Social Understanding after working on this home assignment?

Yes, I think I could develop a "Emoticon Analysis" system or "Domain-Specific Keywords" system.

For the "Emoticon Analysis" is that easy as analyzing the presence and meaning of emoticons within the text. For "Domain-Specific Keywords" as seen in this project, the distribution of the words are not the same in the "hole" dataset than in the dataset focused in the Dublin area.

4 Error analysis

The error analysis is based on the base models as required in the problem statement.

1) Test dataset and using `SVC`, "Bag of words" and test dataset.

good news, just had a call from the Visa office, saying everything is fine.....what a relief! I am sick of scams out there! Stealing!

This tweet is labeled as positive but of course contains words that are tipically used in negative text like "sick", "scam" and "stealing".

insects have infected my spinach plant :(

This was predicted as negative but it's labeled as positive. Is it because spinach plant are related to healthy?

2) Test dataset and using `SVC`, "Bag of words" and dublin dataset.

@dlrcycling True! Also, fun fact - the heavy traffic congestion of Dublin City centre has been an issue since the early 1950s!!

This tweet was labeled as a positive tweet and predicted as negative. Of course this a difficult tweet because we need context. There are some words used on it that are normally used in negative scenarios such as "congestion", "traffic", etc.

3) Test dataset and using `SVC`, "TFIDF" and test dataset.

#lebron best athlete of our generation, if not all time (basketball related) I don't want to get into inter-sport debates about __1/2

The prediction said that it is a negative polarity but it's real polarity was labeled as positive. The problem with this tweet is that it contains the word "debates" which I can understand that the model could classified it as complaint.

@morind45 Because the twitter api is slow and most client's aren't good.

The prediction said that it is a positive polarity but it's real polarity was labeled as negative. The problem with this tweet is that it contains the word ”good” which I can understand that the model could classified it as positive tweet.

omgg i ohhdee want mcdonalds damn i wonder if its open lol =]

This tweet is labeled as positive, and the model predicted a negative tweet. But for me as human is difficult to say that this a positive tweet.

3) Test dataset and using `SVC`, ”TFIDF” and dublin dataset.

Fat kids are miserable - we need to build cycling infrastructure so they can get out on bikes and love their city and countryside https://t.co/WaJcUQMp2b

This tweet is labeled as a negative tweet and predicted as a positive tweet. The problem I see here is that it contains the word ”love”.

...

5 Proposal for improvement

There are several ways to enhance this project. One approach would involve text processing using libraries such as `nltk` to obtain word roots and synonyms. This technique can reduce the number of words for training and potentially capture the overall message more effectively.

Additionally, there are other aspects of text processing that can be improved, such as removing URLs or Twitter usernames. It may also be beneficial to eliminate non-standard symbols while retaining emoticons, as assigning meaning to them can improve the accuracy of sentiment prediction for the tweets.

Finally, regarding the models, I could enhance the training of the transformer model by increasing the number of epochs while incorporating regularization techniques like `Dropout`. Additionally, creating an ensemble of models would further enhance generalization and improve overall performance.