SD201 - Mining of Large Datasets - Report

Ariel R. Ramos Vela **Wilmar Quiroga** **Mohamed Khair** **Gustavo Acioli**

Abstract

Sentiment analysis is an approach to Natural Language Processing whose task consists of categorising opinions from a text and thus, determining the writer's attitude towards a product, service or idea. In the present report, we utilise several machine and deep learning algorithms to predict the classes of movie reviews in the SST-5 (Stanford Sentiment Treebank) dataset. Additionally, we perform a quantitative analysis of the performance of the algorithms being utilised. Our code is publicly available at: `https://github.com/arielramos97/SD201-Project`.

1 Introduction

The objective of this research project is to analyse the SST5-dataset or SST fine-grained, a real-world dataset that comprises 5 classes: negative, somewhat negative, neutral, somewhat positive and positive. Therefore, to perform multi-class classification, we will employ several approaches for classification such as k-NN (k-nearest neighbours), SVM (support vector machines), Random Forests and RoBERTa, a Deep Learning model that is based on BERT.

The performance of the models will be evaluated using accuracy and F1 score. Likewise, we will compare our results having as an indicator the SST-5 Fine-grained classification leaderboard [1] where the state-of-the-art accuracy is 59.8% as shown in Fig 1.

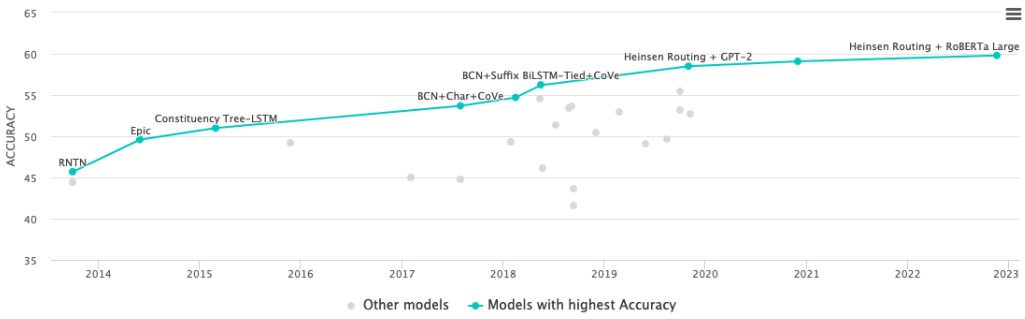

Figure 1: SST-5 Leaderboard

2 Data Visualization

The training dataset comprises 8544 movie reviews while the validation and testing datasets consist of 1101 and 2210 reviews respectively (all with their labels). Thus, the training data represents 72.07% of the data while the validation and testing datasets represent the 9.28% and 18.64% respesctively.

[1] `https://paperswithcode.com/sota/sentiment-analysis-on-sst-5-fine-grained`

With respect to the structure, it is composed by two columns, as shown in Fig 2. The first one is the label of each review and the second one is the 'movie review' itself.

label	data
4	It 's a lovely film with lovely performances b...
3	No one goes unindicted here , which is probabl...
4	And if you 're not nearly moved to tears by a ...
5	A warm , funny , engaging film .
5	Uses sharp humor and insight into human nature...

Figure 2: General Dataset structure

Additionally, we can see the general distribution of the labels for each review in percentages of the Dataset in Fig 3, that is, taking into account training, test and validation.

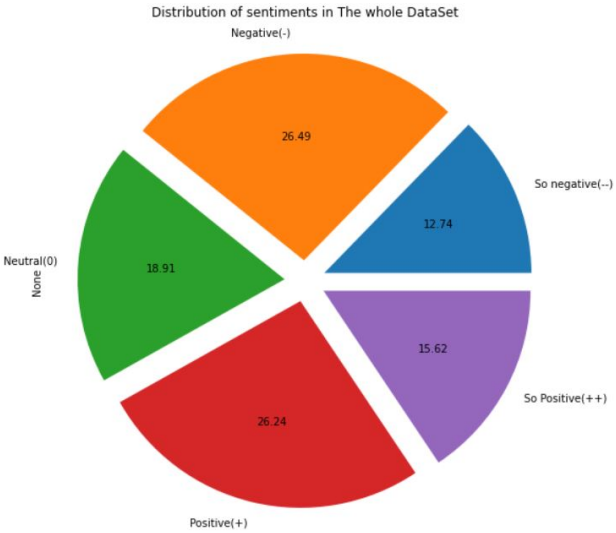

Figure 3: Distribution of Sentiments

Alternatively, we can draw a histogram Fig 4 to visualise the distribution of the labels in precise numbers.

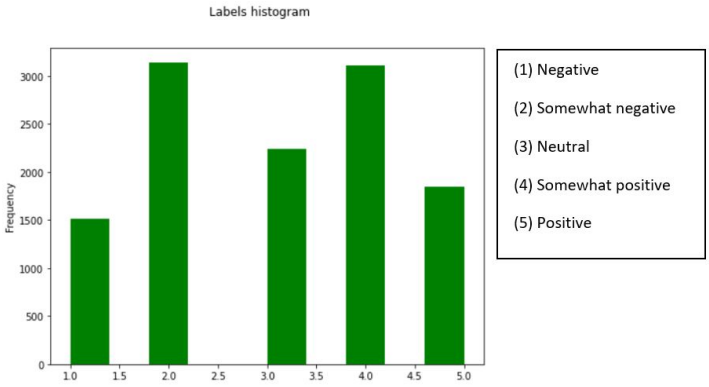

Figure 4: Labels histogram

Likewise, in Fig 5, on the left we can observe that the sentiment distribution for training data is unbalanced which can induce the models to predict the class that has a larger number of samples. Thus, oversampling (minority classes) and undersampling (majority classes) techniques were utilised to overcome this problem in the training dataset. For example, in Fig 5 on the right we can observe that after undersampling the training data was reduced to 5460 reviews.

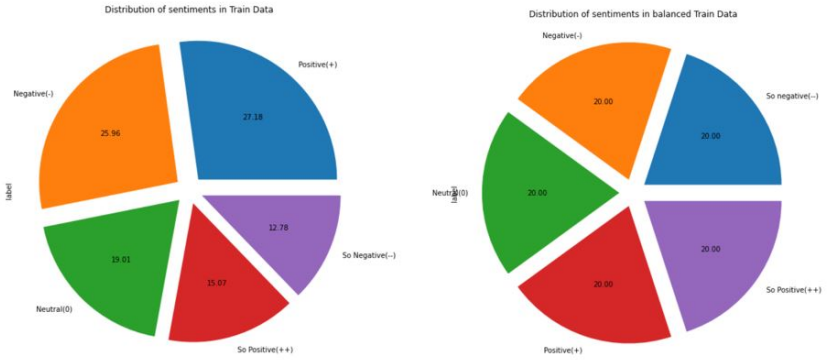

Figure 5: Distribution of sentiments in unbalanced Train Data and balance Train Data

3 Methodology

As mentioned in the introduction, the objective of this project is to analyse and have an insight of the use of different approaches for multi-class classification. Therefore, five implementation approaches were proposed:

- **Majority vote (Baseline):** From the distribution of the training dataset, the majority class is 4. Then we predict that for all the samples in the testing dataset which gives an accuracy of $\frac{510}{2210}$ that is 23.08%.
- **KNN (K-Nearest Neighbors):** a supervised learning classifier, which uses proximity to make classifications or predictions about the grouping of an individual data point.
- **SVM (Support Vector Machines):** Find the n-dimensional hyperplanes, In such a way that the points can be segregated in the regions of space formed by these hyperplanes, the number of dimensions depends on the number of features of the dataset, this hyperplane and the intersection with other hyperplanes form the decision boundary and the points or extreme vectors of each region are called support vectors.
- **Random Forest classifier:** a meta estimator that fits several decision tree classifiers for improving accuracy and handling over-fitting.
- **RoBERTa:** a transformer model that belongs to deep neural network models.

However, before performing the classification task, we need to compute the word embeddings of movie reviews. We tried 3 approaches:

Word2Vec custom model. The Word2Vec (W2V) algorithm from gensim[2] was used to extract the notion of relatedness across words. Besides, the skip-gram model was selected since it generally works better with small datasets and it better represents less frequent words according to [2]. Before training our Word2Vec model, we had to lowercase each review to standardise the data, remove special characters and punctuation, strip multiple white spaces, remove stopwords with the help of the NLTK library and apply stemming OR lemmatization as shown in Fig 6.

We decided to use a vector size of 300 for the embeddings. Then, after the model is trained we can compute the vector representation of each review by averaging (or summing) the vectors of all words contained in it.

[2] `https://radimrehurek.com/gensim/models/word2vec.html`

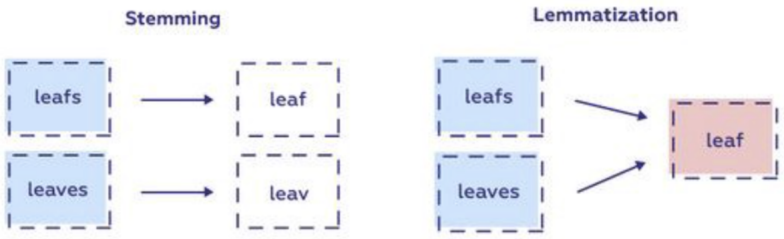

Figure 6: Illustration of stemming and lemmatization.

Pretrained Word2Vec model. We used the pre-trained Google News corpus (3 billion running words) word vector model to obtain the word embeddings of our reviews. For this approach we did not have to train the model, simply average the vectors of the words that appear in a review.

RoBERTa embeddings. This approach was only used for the Deep Learning model (RoBERTa). Essentially, the embeddings were obtained by using the RoBERTa tokenizer. More details in section 4.4.

4 Experiments

The performance of our models was assessed using F1 and accuracy having as a reference the results obtained in the SST-5 Fine-grained classification leaderboard.

4.1 Support Vector Machines (SVM)

One of the approaches used is SVM (support vector machine), for which we use for model selection the GridSearchCV library. Using different values for C: parameter regularization, for gamma and different kernels. We obtain the final parameters $C : 1, gamma : 0.1, kernel :' rbf'$
Using the model described above on test data, we obtain the following evaluation metrics for each class and in general, figure: 7

	precision	recall	f1-score	support
0	0.37	0.51	0.43	279
1	0.50	0.33	0.40	633
2	0.30	0.32	0.31	389
3	0.40	0.44	0.42	510
4	0.52	0.58	0.55	399
accuracy			0.42	2210
macro avg	0.42	0.43	0.42	2210
weighted avg	0.43	0.42	0.42	2210

Figure 7: Evaluation metrics on test data

And the evaluation metrics for the validation data can be seen in the figure: 8

	precision	recall	f1-score	support
0	0.35	0.55	0.42	139
1	0.41	0.27	0.33	289
2	0.34	0.31	0.33	229
3	0.40	0.35	0.37	279
4	0.46	0.66	0.54	165
accuracy			0.39	1101
macro avg	0.39	0.43	0.40	1101
weighted avg	0.39	0.39	0.38	1101

Figure 8: Evaluation metrics on validation data

A tool that helps us to better visualize the preditions is to draw a heat map figure:9, which in the ideal case would concentrate the preditions made by the model on the diagonal of the matrix, but although this is not the case, because our accuracy for the test data is 0.419. We see that the incorrect preditions made by the model are in most cases in a neighboring class, which would be useful for a classification between positive, negative or neutral feelings and would improve the accuracy for this new classification.

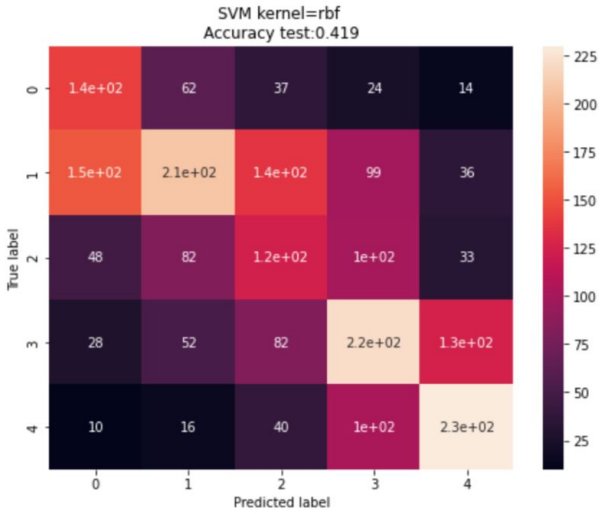

Figure 9: SVM heat map

4.2 K-Nearest Neighbors(KNN)

In order to study all possibilities to obtain a better performance for the model, the KNN method was also used. As studied in class, this is a supervised learning classifier, which uses proximity to make classifications or predictions about the grouping of an individual data point.

First, we tried to verify the performance of the algorithm for several values of k using the test embeddings. The result obtained can be verified through the graph below.

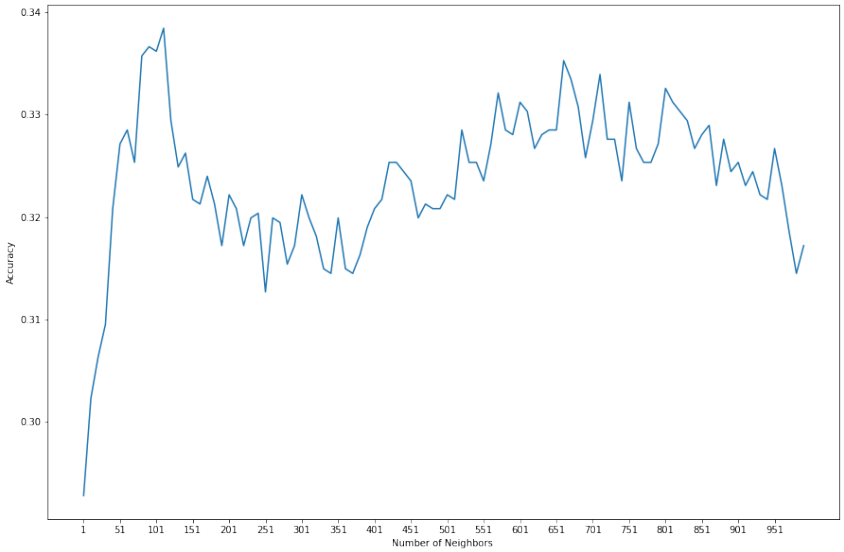

Figure 10: KNN performance for different values of k

Through the graph shown in Figure 10, one can see an interesting behaviour of the model. The algorithm presents the best performances for values of k between the intervals 50/150 and 400/950. As the test dataset was used to obtain these values, it can be inferred that the optimal neighbour values will probably be present in one of these two intervals. However, for this value to be measured more accurately, this graph alone is not enough.

The Hyperparameter Tuning method was then used to try to determine this best k metric. Hyperparameter tuning works by running multiple trials in a single training job. Each trial is a complete run of your training application, with the values of your chosen hyperparameters set within the limits you set. The results of each attempt are tracked and adjustments are made for subsequent attempts. After the job completes, you will receive a summary of all attempts and configurations of values that worked best based on the criteria you set.

In our case, we decided to test various values of k using uniform and distance weights and minkowski, euclidean and manhattan metrics. The validation dataset was used so that the result can be as faithful as possible.

At the end of the process, it was discovered that for the KNN algorithm to be more effective, the ideal value of k should be 400 for our dataset. Also, the most suitable weight is distance and the most effective metric is manhattan.

Using the parameters obtained through the Hyperparameter Tuning method, we can access the maximum accuracy value that the KNN algorithm can provide. Because it is an extremely sophisticated problem that we are trying to solve in this work, it was expected that this KNN algorithm would not provide us with the best accuracy of all. However, as it is a simple algorithm, it managed to fulfill its role with dignity, giving us an accuracy of 33%.

4.3 Random Forests

We also applied the Random Forest classification. In order to choose the suitable parameters, and after digging and experimenting some values, we see that most of these hyperparameters don't induce a big change on the performance. So, instead of following a Grid-search way, we focused on the main parameters that have an impact: Doing the Grid-search is too much time consuming while the performance doesn't change too much. We have shown this for the case of the minimum of samples required to split an internal node for instance (a benchmark). It is also the case for criterion or for the number of features to consider when looking for the best split, among other hyperparameters that aren't of great influence. On the other hand, we did benchmarks for both: the number of estimators and the maximum depth (for the last one, it is in fact just about the length of the leaves so we can leave it as default but it can be reduced in order to shorten the computational time of the benchmarks

to be realised). The final parameters were: 'number of features to consider when looking for the best split' = 20 ,'number of estimators' = 600,'maximum depth' = 32, 'random state' = 0.

We get an accuracy and F1 scores around 37%. We have a better performance than the Baseline but it is not yet a sufficient one.

4.4 Deep Learning model (RoBERTa)

To obtain the word vectors of reviews, we can also employ deep neural networks approches. Indeed, for our project we used RoBERTa (Robustly Optimised BERT Pretraining Approach), a transformer model that is pretrained on a large corpus of English data that has managed to improve the state-of-the-art scores in a variety of NLP tasks. It is built on top of BERT, having as the main difference, the removal of the next-sentence pretraining objective and the fact that it is trained with larger mini-batches and learning rates.

For our model implementation we used the pretrained model provided by the **HuggingFace library** [3] and we modified it accordingly to solve the current classification task.

Model architecture. The model was inspired by [1]. Similar to a traditional classifier it fits the summaries to the labels sent as inputs. The key difference is that the transformer first performs the encoding of the corpus based on its tokenizer and subsequently it realises a standard forward pass on the neural network followed by an optimization step on cross entropy loss.

The main architecture can be summarised as follows:

- RoBERTa Model from pretrained 'roberta-base'
- Dropout layer with rate 0.3
- Linear layer from 768 to 64
- Normalization layer
- Dropout layer with rate 0.3
- Linear layer from 64 to 5 (number of classes)

The inputs for the Roberta Model are the following:

- The input ids from movie reviews which were computed by the Roberta tokenizer (also from 'roberta base').
- The attention mask list that indicates which tokens should be attended to, and which should not. This is useful when the reviews are shorter than 512 tokens.

Hyperparameters. For better performance, some parameters were tuned according to several combinations:

- Linear transformations: (768 to 64 and 64 to 5), (768 to 128 and 128 to 5), (768 to 256 and 256 to 5).
- Learning rate: 1e-05, 1e-06, 1e-07.
- Dropout rate: 0.1, 0.2, 0.3

The best accuracy obtained was 51.90%. It was achieved by employing the following combination of hyperparameters: linear transformations (768 to 64 and 64 to 5), learning rate 1e-05 and dropout rate 0.3.

Likewise, in order to stop the algorithm from overfitting, the number of epochs is only increased if the validation loss has decreased.

5 Analysis of Results.

Table 1: Runtime(s) (including training and testing), F1-score micro, F1-score macro and accuracy (%) of different approaches to classify the SSST-5 dataset.

	Time (s)	F1-score micro (%)	F1-score macro (%)	Accuracy (%)
Majority Vote **(Baseline)**	-	-	-	23.08
W2V custom model + KNN	3	30.79	18.41	30.79
pretrained W2V model + KNN	2	33.60	31.72	33.60
W2V custom model + SVM	26	35.97	19.88	35.97
pretrained W2V model + SVM	15	41.92	41.77	41.92
W2V custom model + Random Forests	3	30.12	17.04	30.12
pretrained W2V + Random Forests	45.41	37.19	36.82	37.19
Deep Learning - RoBERTa	4408.36	51.90	50.21	**51.90**

- **KNN model** was able to obtain an accuracy of 33.60%, which is relatively lower compared to the other models. However, it was already expected that this model would not be the best, as it is a somewhat limited model for a very complex project. (Better than the Baseline)
- **The SVM Model** achieved an accuracy of 41.92% which is close to the state-of-the-art for algorithms implemented for the same dataset, which use the same word to vector approach and then generate embeddings to implement traditional machine learning algorithms. (Better than the Baseline)
- **Random Forests model** reached an accuracy of 37.19%, which is relatively low compared to the state of the art but it is quite better than the Baseline. Considering that we have a NLP problem, this also shows the limits of this model that isn't optimal for this kind of problems.
- **The Deep Learning model** reached 51.90% which largely outperforms the other approaches used. Indeed, it is close to the state-of-the-art approaches in the SST-5 leaderboard. Nonetheless, it takes longer to run (around 1 hour 13 minutes). (Better than the Baseline)

6 Conclusion

In our report we presented several methods to perform classification on the SST-5 dataset. We utilised machine and deep learning models where the best performance was achieved by RoBERTa. Nonetheless, the use of neural networks is still a black box which means that the predictions made by the model can not be explained. This in turn affects its practicality given that in an industry environment like banks, decisions might need to be explained to customers e.g when asking for a denied loan.

With respect to future improvements, a different approach to obtain word embeddings could be employed such as GloVe. In addition, further preprocessing could be performed to reviews to deal with negation e.g 'not good' and contractions. Likewise, for the RoBERTa model, the 'roberta-large' base can be utilised to further increase the accuracy of the model. For doing this, more GPU and memory resources will be needed which is why it could not be attempted by the present work.

Finally, the hypertuning of parameters is still a big challenge that needs to be solved to avoid its manual setting and to reduce the amount of effort and time, especially if a Grid Search approach is applied.

Contributions

- **Ariel:** Preprocess of data, Word2Vec custom model, pretrained Word2Vec model, RoBERTa embeddings, Deep Learning model (RoBERTa).
- **Wilmar:** Data visualization and SVM model
- **Gustavo:** KNN model
- **Khair:** Random Forest

References

[1] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019. cite arxiv:1907.11692.

[2] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[3] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, October 2020. Association for Computational Linguistics.