DAT341: Assignment 3

Johannes Berger
Personal number: 000917
johberge@chalmers.se

Cecilia Nyberg
Personal number: 990106
cecnyb@chalmers.se

Elin Stiebe
Personal number: 000210
elinsti@chalmers.se

Abstract

This paper aims to create a well-performing document classifier on opinions on COVID-19 vaccinations. The classifier is created by investigating which ML system has the highest performance in classifying pro- and anti-covid vaccination comments. In this paper, the logistic regression classifier gave the highest accuracy. The model got the highest performance when features were extracted using Tfidf (term frequency–inverse document frequency). The model had an accuracy of $89.2\,\%$ but had a hard time classifying comments containing negations such as "not".

1 Introduction

This assignment and code aim to classify text data as pro- or anti covid vaccinations. Document classification has a wide range of use cases, such as email spam detection, and classifying opinions on social media [1]. Several papers have been written specifically on the classification of positive or negative opinions. "Classifying Positive or Negative Text Using Features Based on Opinion Words and Term Frequency - Inverse Document Frequency" by Sasiporn and Niwan [2] is one of them, stating that the knowledge of opinions can be a powerful tool in domains such as politics and can aid in marketing and business decision-making.

2 Problem Formulation

The subject of this paper can be broken down into two subproblems. The first one regards transforming the data. Since the data consists of text, this text has to be encoded into a numerical representation before the classifiers are trained. This can be done in different ways, and in this report, two different approaches will be used and compared. The first one uses a TfidfVectorizer, and the second one uses a sentence transformer.

The second subproblem is about the classification of the encoded data. This paper aims to find which model performs the best in the classification of pro- and anti-vaccination comments. The objective further is to report that model's accuracy and discuss some of the incorrect classifications made and why the model had a hard time classifying those comments.

3 Method

The data of this assignment was sourced from the internet by students taking the course DAT341 during the third out of four reading periods. The resulting data sets contained over $50\,000$ comments for the training data and 2000 for the test set. Each instance was annotated by the students and a majority of the training data instances had two or more annotations each from different annotators. The data was labeled as 0 if it was considered to be against COVID vaccination and 1 if it was supportive. A label of -1 was added if an annotator thought a comment was not part of the scope.

The data was preprocessed in several ways. The data was cleaned by removing everything that was not a letter, apostrophe, or a whitespace. Further, instances with annotations of -1 in them were removed since at least one annotator had deemed the comment irrelevant. Instances were also removed if any two annotators did not agree on its label. Thus the models were not trained on any ambigious data.

There are a large number of potential models that could be used for classifying the comments. 9 different ones were chosen to primarily try out. These were the following,

- DummyClassifier
- Gradient Boosting Classifier

- Random Forest Classifier
- Perceptron
- Decision Tree Classifier
- Logistic Regression
- Multinomial naive Bayes (NB)
- Linear Support Vector Machine (SVC)
- k-Nearest Neighbors Classifier (KNN)

The nine above were chosen since they have varying complexity and do the classification in vastly different ways. The dummy classifier is included as a benchmark throughout the report. The classifier has the accuracy of the majority class and thus any well-performing algorithm should beat it.

First, all models were run with their default hyperparameters. Second, their hyperparameters were tuned to try and increase their accuracy. A helper method `training_pipeline` was created to try and find more optimal hyperparameters. The method also tuned TfidfVectorizer's hyperparameters when it was used as encoder. The method took four arguments (X, Y, model, parameters) where the parameter argument denotes a dictionary of different hyperparameter selections. The method returned the best-performing hyperparameters, model object, and test set accuracy found by the *RandomizedSearchCV*. The randomized search reduces complexity, which is significantly higher for other popular options like grid search [3].

When the data was encoded by a sentence transformer, the chosen transformer was the ”all-mpnet-base-v2”. The MPNet model was chosen because it has received a high score for its embedding quality and is a general-purpose model [4]. The model works by mapping sentences to a 768-dimensional dense vector space [5]. In opposition to the TfidfVectorizer, the sentence transformer captures the semantic meaning of a comment instead of focusing solely on which words are used [6]. Since the model was already pretrained and all-purposed, and because of computational constraints, the MPNet model was not finetuned, it was only the classifiers' hyperparameters that were altered during the randomized search.

The evaluations of the models classifications relied on the accuracy score metric and a confusion matrix. And the learning algorithm with the highest accuracy score was chosen as the optimal one. The optimal model was analyzed further with some of its incorrect classifications etc. More about the chosen model in section 4.3

4 Results and Discussion

In this section, each part of the method's results are described and discussed.

4.1 Default Parameters

Table 4.1 shows the test accuracy for each model using its default hyperparameters. All perform better than the DummyClassifier, meaning they are at least better than the benchmark model. The multinomial NB classifier was not run with the MPNet model since it can not handle negative input values.

Model	**Tfidf**	**MPNet**
RandomForestClassifier	0.851	0.845
LogisticRegression	0.848	0.846
LinearSVC	0.848	0.863
MultinomialNB	0.835	-
Perceptron	0.796	0.783
GradientBoostingClassifier	0.773	0.826
DecisionTreeClassifier	0.761	0.720
KNeighborsClassifier	0.740	0.869
DummyClassifier	0.500	0.500

Table 1: Table of all models and their accuracy score using default hyperparameters.

4.2 Tweaked Parameters

All models were tuned with the help of `training_pipeline` method to try and increase their accuracy further. As mentioned, the method was utilized to include the hyperparameters for the model and the TfidfVectorizer, giving each model the highest accuracy. The MPNet parameters were kept constant and not fine-tuned due to the computational effort.

4.3 Model with the Highest Accuracy

As can be seen in table 2, the logistic regression classifier used with the TfidfVectorizer had the highest reported performance in accuracy score. Its confusion matrix is shown in figure 1. The figure demonstrates the high accuracy of the model and that the differences in sensitivity and specificity are small.

It is interesting to look at what the comments it misclassified looked like. Below is one example. The model did not accurately classify this text

Model	Tfidf	MPNet
LogisticRegression	0.892	0.866
LinearSVC	0.885	0.867
MultinomialNB	0.873	-
KNeighborsClassifier	0.826	0.875
Perceptron	0.798	0.833
RandomForestClassifier	0.785	0.852
GradientBoostingClassifier	0.783	0.869
DecisionTreeClassifier	0.695	0.718
DummyClassifier	0.500	0.500

Table 2: Table of accuracy score with finetuned hyperparameters.

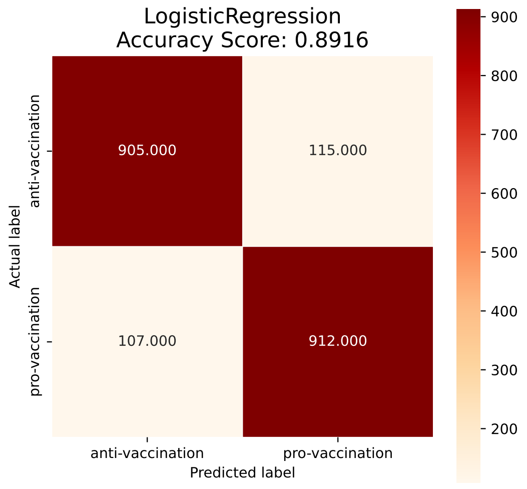

Figure 1: Confusion matrix for Logistic Regression.

when using the tfidfvectorizer. The reason is simple. Without the negation ”not” this text would be pro-vaccine. This is something that the model misses when it uses tfidf for feature extraction.

Text: I've not had the vaccine and I feel great
Classified as: pro-vaccination
True Label: anti-vaccination

On the other hand, using the MPNet model, the logistic regression classifier correctly classified the comment above. The miss-classifications made with the MPNet model looked a bit different, one example is presented below. Most misclassifications in this case were either done on very short comments, like the one below, or long comments with confusing and far-fetched parables.

Text: Poison
Classified as: pro-vaccination
True Label: anti-vaccination

The features that were given the highest weights by the logistic regression classifier are presented in table 3. These features make intuitive sense, showing the classifier's proficiency in identifying words associated with both pro and anti-vaccine.

Rank	Class 1 Feature	Class 0 Feature
1	antivaxxers (6.137)	never (6.477)
2	anti (6.093)	poison (6.039)
3	available (4.600)	not (5.833)
4	science (4.175)	experimental (5.571)
5	today (4.107)	forced (5.295)
6	vaxxers (3.980)	rushed (4.078)
7	get (3.944)	heart (3.885)
8	yes (3.830)	money (3.885)
9	scientists (3.778)	experiment (3.879)
10	hope (3.763)	force (3.870)

Table 3: Top Features for Class 1 and Class 0 for Logistic regression

4.4 Model with the Lowest Accuracy

The Decision tree classifier combined with the Tfidfvectorizer performed the worst of the models. It had a hard time identifying the pro-vaccine comments. This fact becomes clear in figure 2 where the false negatives are almost as high as the true positives. It is a 50/50 chance of pro-vaccine comments to get the right or wrong label.

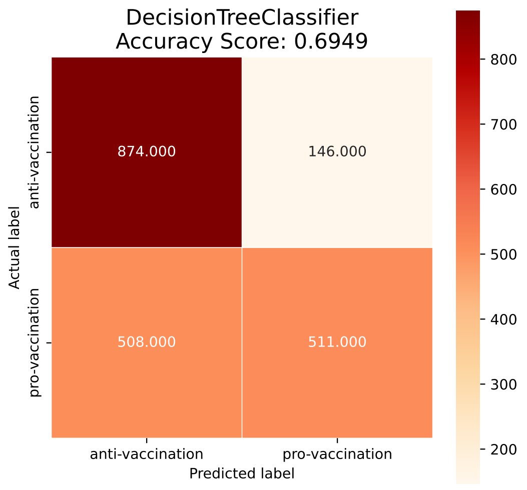

Figure 2: Confusion matrix for the Decision Tree Classifier.

The top ten most important features were extracted and presented in table 4 to understand why the decision tree performed badly. Many words are similar to the ones found by the Logistic Regression Classifier. Nevertheless, the decision tree struggles to find meaningful patterns in the data and might lack too much complexity for this task.

Additionally, the decision tree got a lower accuracy score after changing from the default hyperparameters. This suggests that the search space of the hyperparameters was suboptimal. The search space could have been changed or increased to obtain higher results or at least match the results of the default hyperparameters.

Word	Importance
poison	0.027
today	0.018
vaxxers	0.017
pandemic	0.014
heart	0.014
term	0.013
arm	0.012
taking	0.011
feel	0.011
experimental	0.010

Table 4: Top Features for the Decision Tree Classifier

4.5 Evaluating Annotator Precision

Badly annotated data would give suboptimal grounds to train the model on. The model was not trained on data where one of the annotations was labeled as a -1 since these instances were considered irrelevant by one or more of the annotators. Further, about 3.75 % of the annotations only contained one annotation label. The models were trained on these instances, though they are not included in the evaluation of the annotator accuracy since there is nothing to compare them to. The method created to evaluate the accuracy only considered annotations as correctly annotated if all annotators agreed on their annotations. For example, if 5 annotators made annotations and 4 gave it a label of 1 but one person labeled it as a 0, then the annotation was considered to be non-consensus. This is a harsh way of evaluating the annotators' performance. Yet the consensus reached 84.4 %, a good score. Thus, the annotation consensus is good.

4.6 Differences in feature representation

When evaluating the effect of the different text encoding approaches, it is important to recognize that the MPNet sentence transformer and Tfidfvectorizer work in different ways. While the Tfidf concentrates on each word in a sentence [7], the MPNet instead captures the semantic meaning of it [6]. A major drawback of the Tfidf is that it does not account for ordering. The example error for the logistic regression classifier in section 4.3 shows that the Tfidf can misclassify sentences where a negation is included. This is not the case for the sentence transformer, but it can instead miss patterns in specific word choices. When inspecting the data for this task, it can be seen that similar words are often used by people belonging to the same class. The error example presented earlier clearly illustrates this: when focusing on words often appearing in anti-vaccine comments, the comment saying ”poison” could easily be identified as anti, but when focusing on the semantic meaning of the sentence, this comment was mislabeled. Which text encoding approach is most beneficial, therefore, depends on the data and task. In the matter of classifying these comments, both missing negations and ignoring specific word choices can be disadvantageous. The results indicate that recognizing specific words holds more importance in this case since the use of the TfidfVectorizer could reach a higher accuracy than the sentence transformer.

The results further show that the two different text encodings work differently well on different classifiers. For the TfidfVectorizer, the logistic regression classifier produced the highest accuracy and for the MPNet model, it was the KNeighbors classifier. For more simple classifiers, the increased complexity of the data that follows from using the MPNet might be disadvantageous since it can make it harder for a simple model to generalize. This goes the other way as well, more complex models might perform worse on simple data since that can make them more prone to overfitting.

4.7 Limitations

Choices were made during many steps in this paper. Each choice puts the result under limitations. First the preprocessing of the data. All tokens not being letters, apostrophes, or whitespaces were removed. This could affect the results. For example, when examining the comments, it seemed that people who were against the vaccine appeared more heated and used more exclamation marks and emojis. Including these could potentially lead to higher accuracy.

Second, about the search space of each model's hyperparameters. Decisions were made about which parameters to include in the search space and further what values each parameter was evaluated on. These search spaces could have been investigated further to gain higher accuracies for each model. Further, currently, the same search space of hyperparameters was used for both tfidf and MPNet though they could probably been changed to fit each of the embedding's vastly different behaviours.

Third, there is a gap in this paper related to the accuracy of the MultinomialNB model for MPNet. It would not be run since MultinomialNB cannot handle negative values. Thus there were no results to report on this model. This could have been solved by scaling the features and is something that could have been improved on.

5 Conclusion

The best-performing model to classify anti- and pro-covid vaccination comments was the logistic regression model, which had an accuracy of 89.2 % when working with features extracted with the Tfidf. The model had difficulties correctly classifying comments that had negating in them.

References

[1] AltexSoft, *Document classification: An overview*, `https://www.altexsoft.com/blog/document-classification/`, Accessed: February 15, 2024, Year the post was published, if available.

[2] S. Tongman and N. Wattanakitrungroj, "Classifying positive or negative text using features based on opinion words and term frequency - inverse document frequency," in *2018 5th International Conference on Advanced Informatics: Concept Theory and Applications (ICAICTA)*, 2018, pp. 159–164. DOI: `10.1109/ICAICTA.2018.8541274`.

[3] scikit-learn. "Comparing randomized search and grid search for hyperparameter estimation." Accessed on February 14, 2024. (n.d), [Online]. Available: `https://scikit-learn.org/stable/auto_examples/model_selection/plot_randomized_search.html`.

[4] "Pretrained models — sentence-transformers documentation." (), [Online]. Available: `https://www.sbert.net/docs/pretrained_models.html` (visited on 02/14/2024).

[5] H. Face. "Mpnet model for sentence embeddings." (), [Online]. Available: `https://huggingface.co/sentence-transformers/all-mpnet-base-v2#fine-tuning`.

[6] H. Face. "Using sentence transformers at hugging face." Accessed: February 15, 2024. (), [Online]. Available: `https://huggingface.co/docs/hub/sentence-transformers#using-sentence-transformers-at-hugging-face`.

[7] M. Chaudhary. "Tf-idf vectorizer scikit-learn." 6 min read, Apr 24, 2020. Accessed: February 15, 2024. (2020), [Online]. Available: `https://medium.com/@cmukesh8688/tf-idf-vectorizer-scikit-learn-dbc0244a911a`.