Stance detections:
Classifying comments on Covid-19 vaccination

1 Introduction

In this paper we present our solution to the machine learning task of developing a text classifier that determines whether a given textual comment expresses an opinion that is positive or negative towards COVID-19 vaccination.

The remainder of this report is structured as follows. Section 2 describes our proposed approach: we will discuss its different steps. In Section 3, we present the experimental settings and evaluation results. Finally, Section 4 concludes the paper.

2 Proposed approach

In general, the task of supervised classification begins with a training set $D = (d_1, \ldots, d_n)$ of comments that are labeled with a class $c_i \in \{0, 1\}$ (anti- or pro-vaccination). However, there can be instances with the label -1, which represents that the annotator could not categorize the comment.

2.1 Data collection

We are using the supplied training and testing datasets consisting of "comments in English relating to COVID-19 vaccination from social media or the comment fields from online articles" that were collected jointly by students of the course Applied Machine Learning. We preprocess the given comments by removing punctuation since we believe punctuation marks are not relevant for classifying a textual comment in the context of its stance regarding COVID-19 vaccinations.

Since the comments were collected and annotated from various students the data holds in total 240 duplicates. We removed these duplicates by joining their labels. Most of the comments in the training data were annotated two or more times by various annotators. In order to create a target vector consisting only of a single label for each comment we decided on a class as follows:

Using a simple majority vote we assign to each comment the majority class. In the case of a draw, the label -1 is assigned. Ex.: Let d_1 be a comment with multiple annotations: $(0, -1)$, i.e. the first annotator classifies the given comment as being negative while the second annotator was not able to understand the comment as being positive or negative towards COVID-19 vaccination. We would then assign this comment the label -1 since we only want to include comments in our training dataset which can be clearly labeled as being positive or negative.

In the training data, the annotators agreed on 85% of the comments. Further, Krippendorff's alpha for determining inter-annotator reliability was used and an alpha of 0.6 was calculated. Values of Krippendorff's alpha range from below 0 to 1, where 0 indicates the absence of reliability, and 1 perfect reliability.

Our final training dataset consists of 22323 comments with even distributions of positive and negative opinions. The dataset used for testing contains 1136 opinions and only one annotation per comment. A sample of the training set can be found in Table 1.

Comment	*Label*
NATURAL IMMUNITY protected us since evolution Do not exist anymore	0
The bigest sideffect of vaccines is fewer dead children That is savage	1
vaccines are safe and effective Why Because WE SAID SO	0
A robber will never say he stole something Dont wait for the media to tell you the truth	0

Table 1: Samples of the final training set

2.2 Feature extraction

In this step, our objective is to build a feature vector of the comments. We have chosen to apply the tf-idf vectoriser. Tf-idf is widely used when handling textual data and intends to reflect how relevant a term (e.g., word) is in a given comment. We are using the vectoriser with its default parameters, meaning all words are set to lowercase and only unigrams are taken into account. Depending on the classification algorithm we are going to choose how we will tune hyperparameters of this vectoriser.

2.3 Classification

Our main objective is to use a supervised learning algorithm in order to classify a vectorised comment $d_i \in D$ that has the class $c_i \in \{0, 1\}$ (positive or negative). As a baseline, we have trained a Dummy Classifier which simply returns the most frequent class label in the observed training data. As expected from the equally distributed data this classifier achieved an accuracy score of 50.74% using 5-fold cross-validation on the training set.

First, we chose the algorithm that performs better with our given feature vector. To find the best model we compare various types of classifiers using 5-fold cross-validation: linear, neighbour based and Naive Bayes classifiers. We evaluate each algorithm with its default parameters in order to ensure the comparability between them.

In the second step, we tune the hyperparameters of the chosen model to prevent overfitting and to improve its performance; again, using 5-fold cross-validation.

In the final evaluation, we will analyse the model itself, report the confusion matrix on the test set, and investigate wrongly classified comments.

3 Experiment and evaluation

In order to evaluate our proposed approach, we used a confusion matrix to visualize the performance measurements. Several metrics can be drawn from the confusion matrix, namely accuracy, f1-score, precision, and recall. Since the training data set consists of 10995 anti-vaccination and 11328 pro-vaccination comments, we used the accuracy score as our main evaluation metric.

Given that the annotators only agreed on 84.89% of the comments, we don't expect the final model to perform better than this (*If the annotators can't agree on a label why should a model be able to decide?*). Given the low alpha score we consider this task to be hard but still consider the given data to be reliable since it was collected and annotated by about 100 people using various sources.

3.1 Model selection

We wanted to use classifiers that accept sparse matrices as an input and especially interpret missing values as being missing. Namely, we compared LinearSVC, LogisticRegression, MultinomialNB, and KNeighborsClassifier using the sklearn framework.

As can be concluded from the results presented in Table 2, the LinearSVC model performed best compared to all other classifiers in the metrics accuracy, f1-score and precision (avg. accuracy = 0.81, avg. f1-score = 0.82, avg. precision = 0.82).

Classifier	*accuracy*	$f1-score$	*precision*	*recall*
LinearSVC	**0.8130**	**0.8156**	**0.8169**	0.8154
LogisticRegression	0.8092	0.8118	0.8132	0.8119
MultinomialNB	0.8048	0.8113	0.7973	0.8275
KNeighborsClassifier	0.7083	0.7537	0.6679	0.8802
DummyClassifier	0.5074	0.6732	0.5074	**1**

Table 2: averaged 5-fold cross-validation metrics of initial classifiers on the training set

3.2 Hyperparameter tuning

In the next step, we added regularization to the LinearSVC model in order to prevent overfitting on the training data. We also decided on not tuning the hyperparamteters of the tf-idf vectorizer. We used the two main regularization techniques:

- l1 - regularization, i.e. inbuild feature selection
- l2 - regularization, i.e. irrelevant features get weights closer to zero

By applying regularization to the classifier we don't need to do feature selection before training our model. With both kind of regularization techniques we have a way in selecting only the most relevant features by giving these features a greater weight or by removing irrelevant features (setting their weights to zero: l1-regularization).

Given both regularization techniques we tuned the hyperparameter C by narrowing our search of the best C-value in the following way: In each step we train and validate 2 models (one with l1- and the other with l2-regularization) for each C-value in a given range and a given step size. After each step we reduce the range and step size and report the model with highest accuracy score. The following figure (Fig. 1) represents the final step, showing the results of the 5-fold cross-validation on the training set.

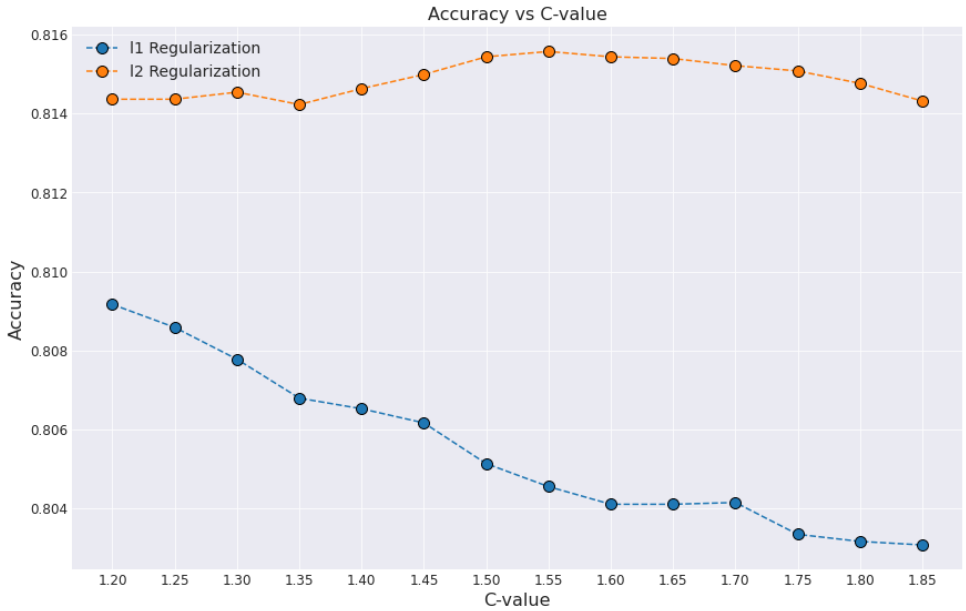

Figure 1: 5-fold cross-validation accuracy vs c-value on the training set

The best model had an average accuracy of 81.56% with the following parameters: C=1.55 and loss='hinge'.

3.3 Model interpretation

Figure 2 shows the 7 most important features, i.e. words, of our final model to decide on a stance of a given comment. As shown in the figure, words like 'antivaxxers', 'argument', and 'anti' have a weight above 4 and therefore a great importance towards a positive class label (pro-vaccination). On the other hand, words like 'not', 'never', and 'rushed' have weights below -3 and pull the comment's label towards a negative stance (anti-vaccination).

The two words with the lowest weight, i.e. words which pull the classification of a comment towards a negative label, are both negation words. The other words on that side, like 'experimental' and especially 'poison', make sense to have a large negative weight on the feature vector.

Two of the 7 features with highest weights, namely 'antivaxxers' and 'vaxxers', are directed towards people with a negative opinion on COVID-19 vaccination. As for the negative-weight words, the words with positive weights seem to make sense to classify a comment as being positive or negative.

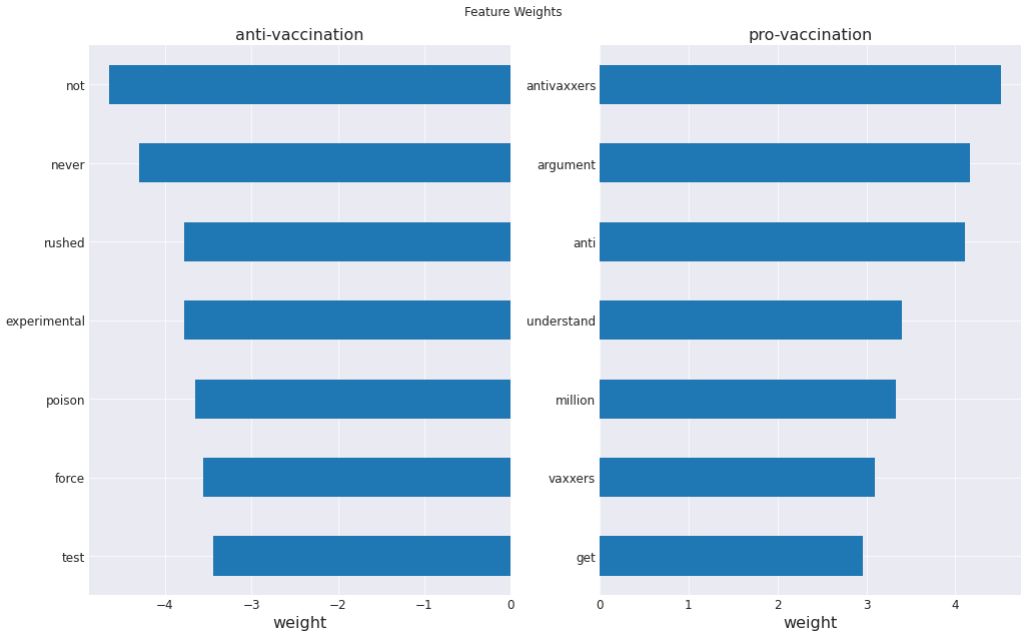

Figure 2: Feature weights of the final model

3.4 Final Testing

After training the final model on the whole training data and applying it to the test set we achieved an accuracy of 86.6% and the confusion matrix as shown in Figure 3.

In the Table 3 the scores of the final model on the test set are shown.

$Classifier$	$accuracy$	$f1-score$	$precision$	$recall$
LinearSVC	0.8662	0.86667	0.8697	0.8636

Table 3: averaged 5-fold cross-validation metrics of final classifier on the test set

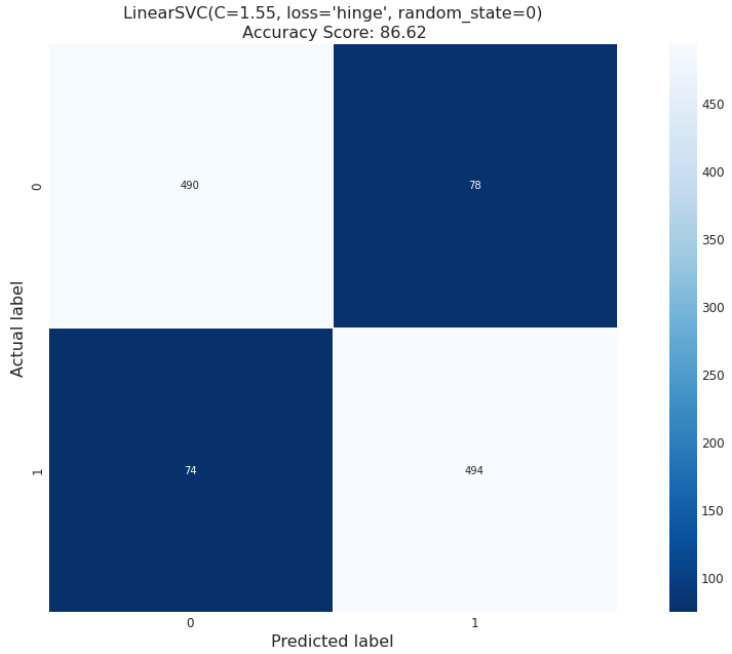

Figure 3: Confusion matrix of the final model predictions on the test set

3.4.1 Wrongly classified comments

When testing the final model, we also looked into wrongly classified comments using the top 100 anti- and pro-vaccination words, i.e. words having very low or very high weights. There were pro-vaccination words in 61 (78.21%) of the wrongly classified anti-vaccination comments (i.e. anti-vaccination comments classified as being pro-vaccination) and anti-vaccination words in 56 (75.68%) of the wrongly classified pro-vaccination comments. On average a wrongly classified anti-vaccination (negative) comment consisted of 7.26% pro words and a wrongly classified comment with a positive opinion towards COVID-19 vaccination consisted of 10.39% anti words. The presence of words corresponding to the other class could be the reason why these comments were classified incorrectly.

4 Conclusion and Discussion

In this report, we presented an efficient linear approach for classifying textual comments as expressing an opinion that is positive or negative towards COVID-19 vaccination. Our final model yielded high accuracy (above 86%) with evenly distributed false negatives and false positives. Given that the annotators only agreed on 85% of the comments our final model's performance can be interpreted as excellent.

Future work. Our proposed approach and final model could be improved in several ways: The selection of collected comments could be done more carefully, for example by joining highly similar comments or by removing incomplete or dummy entries. Secondly, hyperparameters of the tf-idf vectorizer could be tuned. In our current approach we are only using the default parameters of the tf-idf vectorizer and, especially, only evaluating a comment by its single words and not by their combinations. We assume an increased performance when including ngrams.