Implementing a Pro- vs. Anti-Vaccination Text Classifier Using Machine Learning

Albin Ekström
Department of Computer Science and Engineering,
Gothenburg, Sweden
albeks@chalmers.se

Jonas Nordin
Department of Computer Science and Engineering,
Gothenburg, Sweden
jonnordi@chalmers.se

Abstract

This paper outlines the process of building a machine learning system for a specific task of classifying comments about the covid-19 vaccine as either positive or negative. The paper include comparisons and analysis of systems built using different procedures for data pre-processing, feature extraction and selection of classifier. Systems are optimized using different feature extraction methods and by tuning of hyper parameters, and are compared by evaluating metrics such as accuracy, recall and cross-validation scores.

1 Introduction

Ever since the first vaccines were released there has been a movement against the injections. This movement is quite small nowadays, but received a large audience and became louder when the vaccination program for COVID-19 was established (Malagoli et al., 2021). The debate that arose during the pandemic between pro- and anti-vaccinationists (hereinafter referred to as pro- and anti-vaxxers) is in its simplicity about individual autonomy and public health. The pro-vaxxers often argue from a larger health perspective (i.e. at the societal level), while anti-vaxxers argue more about individual autonomy and the right to decide over their own body (Raballo et al., 2022).

When analyzing discussions in social media, primarily on Twitter, about vaccination, it becomes clear that there is a prevalent negative sentiment towards topics related to the anti-vaccination movement (Malagoli et al., 2021). The article also proves that some topics such as health, work, and religion, are often common in vaccine-related tweets. Another interesting conclusion was that tweets related to anti-vaxx often discussed topics such as death, anger, and negative emotions.

The purpose of this report is to use machine learning (ML) techniques to distinguish between comments written by pro- and anti-vaxxers. The data used was collected by each student from social medias, including both pro- and anti-vaxx comments. In order to build an accurate text classifier, different ML models will be tested to assess the dataset and determine the best-performing classifier.

2 Data Preprocessing and Annotations

For an ML model to perform as well as possible, data preprocessing is required. This chapter will discuss the annotation steps taken to prepare a dataset for an ML model to classify the pro- and anti-vaccine comments. It describes how annotations were made, how the consensus between the annotators was evaluated, and an overview of the rules applied to achieve a common annotation for each comment. Furthermore, how the dataset was cleaned and handled.

2.1 Annotation of Data

The data collected by each student were 50 comments containing a balanced set of pro- and anti-vaxxer comments. The comments were selected based on their clarity, such that a person without any contextual knowledge could easily determine whether the comment is in favor or against vaccination. Each comment was assigned a label of 1 as pro-vaccination or 0 as anti-vaccination. When these comments were submitted each student was handed another approximately 100 already labeled comments which they were supposed (without knowing the label) to be classified as pro- or anti-vaxx, and if the comment was unclear a label of -1 was used.

The final dataset was divided into a training and test set containing 38848 and 1524 comments respectively. The dataset contained two columns di-

vided up into annotations and comments. In the 'annotations' column, all the annotations of that specific comment were stored separated by '/', and in the 'comment' column the related comment was stored, see Table 1 for a visualization.

Annotations	Comment
1/1	I'll only consume if I know what's inside it. Still him drinking monster which has taurine
0/-1	It is easier to fool a million people than it is to convince a million people that they have been fooled. - Mark Twain
1/1/1/-1	The bigest sideffect of vaccines is fewer dead children That is savage

Table 1: shows 4 not preprocessed comments from the dataset. In the first column, the annotations are separated with a '/', and in the second their related comment.

As shown in Table 1 annotators don't always agree if a comment should be labeled as pro- or anti-vaxx. The consensus rate of all the annotations was evaluated by computing the amount of the most common annotation[1] and then dividing it by the total number of annotations for each comment (e.g. in the annotations [1,1,1,-1] the most common annotation is '1' with the amount of 3. Then the consensus rate was computed by dividing $\frac{3}{4} = 75\ \%$). When all individual consensus rates were computed the mean of all comments gave the total consensus rate over the whole dataset of $92.6\ \%$.

2.2 Data Preprocessing

In making an ML model train and predict the comments with as high accuracy as possible - the comments had to be labeled with one common annotation. Hence, it was important to make rules for which classification each comment should have. The common annotation was always set to '-1' if the rules were fulfilled, otherwise, it was set to the most frequent annotation in the set. All the rules were motivated by the same principle - if there was disagreement between the annotators that could indicate that the comment was unclear without context. Hence, it was better to give the common annotation '-1' to make the dataset more robust, where each comment with high probability has a correct classification. The rules were:

- If there was equally many as there was unique annotations (e.g. [1,0,-1]), label the comment as unclear, '-1'
- If there was an annotation that was set to '-1' in the set of annotations, label the comment as unclear, '-1'
- If there was more than one annotation and the amount of each annotation was equal (e.g. [1,1,0,0]), label the comment as unclear, '-1'

After the rules had been applied to the training dataset the classifications shown in Table 2 were observed. All the comments with a label of '-1' were dropped.

Annotation	1	0	-1
No. comments	15653	15390	6842

Table 2: shows the number of comments for each annotation.

There was some further data preprocessing done to minimize the number of unnecessary characters and words. In every comment, 'punctuation':s from the 'string' library was removed including the characters `"#$%&'()*+,-./:;<=>?@[\]^_`{|}~` (Foundation, 2021). However, all words were set to lowercase in the vectorizers used before the ML algorithms used (described in chapter 3.1), hence there was no need to set the text to lowercase during the preprocessing.

3 Performance Evaluation of Machine Learning Models

This chapter will give an introduction to the feature extraction methods and models tested and evaluated. The results of the evaluations will be presented at the end. In order to select the most appropriate ML model, feature extraction and hyper-parameter setup, evaluation is necessary. To evaluate models, scoring metrics are compared against other models, using different feature extraction methods and hyper-parameter combinations to find the best-performing one.

3.1 Feature extraction by vectorizers

For feature extraction of the comments, three different vectorizers are used and tested with all of the classification models. The vectorizers are used to represent the comments numerically as vectors, to train classifying models.

One vectorizer used is the *Counter Vectorizer*.

[1] if there were equally many as unique annotations; the most common annotations was set to the first appearing annotation in the set.

This is a relatively naive feature extraction method as it's only based on the frequency of words to represent text in a numerical vector. Another, more sophisticated vectorizer, used is the *TF-IDF Vectorizer*, which works similarly to how the *Count Vectorizer* works but instead of using the frequency of words as scoring it uses a TF-IDF score giving more weight to words that are rare in the texts. This may lead to more meaningful representations as common words like ”and”, ”or”, ”this”, etc. often do not provide any useful knowledge of a comment.

Lastly, a *Hashing vectorizer* was also used. This type of vectorization is based on a hashing function to convert words into indices. One advantage of using this type of vectorization, in relation to the *Counter-* and *TF-IDF* vectorizers is the memory efficiency because it does not have to store the whole vocabulary, making it useful when working with large datasets. All of the vectorizers used are set to automatically consider every word as if it is in lowercase, resulting in not treating for example ”this” and ”This” as separate words with different meanings and weights.

3.2 Machine Learning Algorithms

Initially, a baseline score was set using a *dummy classifier* (DC) to only classify comments as the most common class in the dataset it was trained on.

Further, a *linear support vector machine* (L-SVM) was used. This type of model works by finding the hyperplane that maximizes the distance between the hyperplane itself and the closest data points. A similar model used is the *logistic regression* (LR) classifier which just like the L-SVM creates a linear boundary used for binary classification of data points.

Other classification algorithms tried are *decision trees* (DT) and *random forests* (RF). A decision tree is fundamentally a tree-based model that recursively splits the space made up by the training data into subsets based on the values of its features. This results in a tree with a depth decided by a hyper-parameter, where each internal node of the tree represents a decision based on the value of a specific input feature, and where each leaf node represents a class label. A random forest is an ensemble of multiple decision trees trained on different parts of the input data and then averaged over when classifications are made.

Approaches using *artificial neural network* (ANN) models were also tested, both a single layer *perceptron* (FNN) and a *multi-layer* (MLNN). Neither of the two gave impressive results. When testing the MLNN the computing power was a limitation as the maximum amount of hidden layers we could try in a reasonable time frame was one layer of 5 hidden neurons.

Lastly, models such as *multinomial naïve Bayes* (NB), which is based on *Bayes theorem*, and *k-neighbors neighbors* classifier (k-NN), which is based on clustering data points close to each other, were tested.

3.3 Evaluation

Each classifier with tuned hyper-parameters was evaluated together with each vectorizer to test which performed the best on classifying comments as pro- or anti-vaxx. The tuning was done using different approaches depending on the complexity and computing power needed for the specific model. When comparing the performances between models the metrics: accuracy, precision (PPV), recall, and F1-scores were used. Test scores of the models with their respective optimal hyper-parameter configuration are listed in table 3.

Model	Opt. param	Acc	Rec	F1	PPV
DC	'most_frequent'	0.5	1.0	0.667	0.5
kNN+TFIDF	n=80	0.792	0.862	0.806	0.756
DT+CV		0.765	0.766	0.765	0.764
RF+TFIDF	k=13 size=150	0.791	0.777	0.79	0.803
MLNN+TFIDF	hid_layers=(5,)	0.801	0.798	0.8	0.802
FNN+CV	penalty=None	0.817	0.778	0.81	0.884
LR+TFIDF	'liblinear'	0.847	0.84	0.846	0.852
NB+TFIDF	alpha=0.5 fit_prior=False	0.852	0.853	0.852	0.851
L-SVM+HV	C=1 dual=False 'squared_hinge' penalty=L2 tol=0.001	**0.852**	**0.845**	**0.851**	**0.858**

Table 3: Test scores for each of the evaluated models

The hyper-parameter tuning of the k-NN, decision- and random forest model was done by testing a range of different values for the parameters of neighbors, ensemble size, and max depth. As for the FNN and MLNN models, they are models requiring heavier computation power and also a lot of different parameter configurations, hence no parameter tuning was carried out, because of the time frame.

The two best-performing models with their standard parameter setups, the multinomial NB and the L-SVM, were investigated further. To find the optimal hyper-parameters of those models a grid-search was used with parameters covering a large number of possible configurations of the models.

4 Analysis of the Best Performing Model

The best-performing model, evaluated from test scores in Table 3, is the combination of an L-SVM with a hashing vectorizer. It's especially good at binary classifications as it builds on finding a linear hyperplane separating the feature space in two. An L-SVM also handles noise in the data well because the hyperplane only depends on the points close to it, meaning that outliers (non-sense comments) do not affect the classification of other data points. The score listed in Table 3 was found by grid searching the optimal hyper-parameters. The grid search itself uses cross-validation scores to choose which of the tested hyper-parameter configurations performed the best.

The model receives an accuracy and recall of 89.4 % from the training data indicating that it is approximately as good at prediciting the training data as the test data. The overfitting is handled primarily by making use of regularization and by the feature selection method done by the vectorizer.

The classifier uses the words listed in Table 4 as the most important features[2].

Words	Anti	Pro
1	poison	antivaxxers
2	pressured	antivaxers
3	forced	getvaccinated
4	experimental	ventilator
5	never	response

Table 4: shows the top 5 most important features (words) for classifying pro- and anti-vaxx comments.

The comments which are incorrectly labeled are primarily sentences with negotiation or irony, shown in Table 5

5 Discussion

This chapter discusses the outcomes of the dataset and the evaluation of the models, as well as the performance of the selected ML algorithm L-SVM + HV.

[2]The feature importance was given from the L-SVC + TFIDF

Comment	Actual	Pred
let the rich take the shots first	0	1
i've had three doses as have all of my kids and coworkers and no no seizures or any other effects	1	0
you dont have to be an antivaxxer to understand the stupidity this year	0	1

Table 5: shows 3 incorrectly predicted classifications of pro- and anti-vaxx comments.

5.1 Data Collection and Preprocessing

The final dataset contained 15653 pro-vaxx and 15390 anti-vaxx comments (as shown in 2), which is considered to be a balanced dataset with no bias toward a specific class. However, one might reason that the rules for setting the common annotation to '-1', for comments with annotations containing a '-1', the ML model might be biased to understand the "easily" classified comments. For example, if a comment is annotated [1,1,1,-1], then the common annotation will be set to '-1', but including all annotations, most annotators agree that the comment should be classified as '1'. We tested to remove this rule, but then the evaluation over all metrics dropped, hence the rule made sense to include. After investigating some comments (see Table 6) including a '-1' in the annotations, it was clear that these comments were often unclear without context (or very hard to understand with misspellings words etc.).

Annotations	Comment
1/-1	just because I don't have dr in front of my name doesn't mean that my opinion is less valid yes, it does, actually.
1/0/-1/0	90% of people that get vaccinated don't get the virus Wooow what's in the vaccine then? I'm very pro vaccination, but in my opinion Covid 19 vaccine is just sweetened water.

Table 6: shows an example of two unclear comments labeled with a '-1'.

Even if there was some unclarity in the comments shown in Table 6, the overall consensus was very good at 92% for the not preprocessed dataset. The improvements that could have been made are in the data collection process. By giving the annotators even clearer instructions about finding only very easily classified comments, the consensus could have been even greater. However, in other data collection efforts, it may be important to carefully consider potential sources of bias (such as bi-

ased sampling methods or biased annotator selection) in order to ensure that the resulting dataset is representative and unbiased.

5.2 Evaluation of L-SVM + HV

The model performed with an equivalent score as P. Nijhum did with SVM + CBOW (continuous bag of words) ML model (Paul and Gokhale, 2020) (although Nijhum got an accuracy of 87 % using RF + CBOW).

As seen in Table 4 the most frequent words for classifying anti-vaxx comments are related to the topics of death, anger and negative emotions which is confirmed by the human analysis of tweets related to COVID-19 done by L. G. Malagoli (Malagoli et al., 2021). Another interesting point is that to classify pro-vaxx comments the words antivaxxers and antivaxers (and not shown in the table, but at 10th place antivaxx) have the highest feature importance for the pro-vaxx label. After analyzing the pro-vaxx comments this is probably because these are hashtags used by provaxxers, but because we remove # in the preprocessing it's not shown in the feature importance.

Although this system may be effective at predicting pro- and anti-vaxx comments, it has certain limitations. For example, if we were to expand the problem to classify comments by multiple labels, the performance may decrease due to the linearity of the system. Since it can only make binary classifications, a one-vs-the-rest strategy must be used to address multi-labeling problems. If we were to also have some imbalance in the dataset; the model may have difficulties placing the hyperplane correctly, which may result in it being closer to the majority class.

6 Conclusion

Building a machine learning model made to solve a specific task requires analysis and comparison in mainly three different fields. The first one is the collection and processing of data. The data used in this project was collected and cross-annotated by students, and the preprocessing of it was mainly about finding the actual label of a comment by evaluating its multiple annotations. Further, feature extraction of the data must be done, this was done by using and comparing different vectorizers capturing different aspects of the texts.

Lastly, the classifier L-SVM + HC used made good predictions compared to other projects trying to classify pro- and anti-vaxx comments. To improve the results even further probably an ANN should be used with more than 5 neurons, which have shown some really good results in other text classifying projects.

References

Python Software Foundation. 2021. String methods - python 3.10.0 documentation. `https://docs.python.org/3/library/string.html`. [Accessed: February 17, 2023].

Larissa G. Malagoli, Julia Stancioli, Carlos H. G. Ferreira, Marisa Vasconcelos, Ana Paula Couto da Silva, and Jussara M. Almeida. 2021. A look into covid-19 vaccination debate on twitter. In *13th ACM Web Science Conference 2021*, WebSci '21, page 225–233, New York, NY, USA. Association for Computing Machinery.

Nijhum Paul and Swapna S. Gokhale. 2020. Analysis and classification of vaccine dialogue in the coronavirus era. In *2020 IEEE International Conference on Big Data (Big Data)*, pages 3220–3227.

Andrea Raballo, Michele Poletti, and Antonio Preti. 2022. Vaccine hesitancy, anti-vax, covid-conspirationism: From subcultural convergence to public health and bioethical problems. *Frontiers in Public Health*, 10.