Netnews Miners

Abderrahmane Charrade (1754443)[1], Esraa Bahman (1754158)[1], Masoumeh Shekari (1719408)[1], and Subash Poudel (1754578)[1] Sven Oerther (1417750)[1]

University of Mannheim
Chair of Information Systems V: Web-based Systems, Prof. Dr. Christian Bizer
Master Programs of
Data Science
Business Informatics
Economics and Business Education

Abstract. The goal of our project is to use Natural Language Processing (NLP) in order to extract insights from the unstructured text document of 20 Newsgroup dataset. In this project, we trained different models to find out which model provides better accuracies. Along with preprocessing techniques, we also used feature engineering techniques like TF-IDF Vectorizer and Countvectorizer to feed our models with the accurate data. Various measures like accuracy, precision, recall and F1-score were used to determine the best fit model. Final results indicated that Naive Bayes performed best followed by Logistic Regression, Linear SVM and Random Forest.

Keywords: Data Mining · Natural Language Processing · 20 Newsgroups · Kaggle

1 Objective

Analyzing text in a manner that only human beings are able, becomes more important for many institutions, companies and agencies. In combination with the amount of stored text, like in forums, chat applications and business communication, it becomes also more interesting to automate the process of analyzing these texts with computational support. Therefore, approaches and ways are offered by Data Science Techniques which makes the process much easier and more efficient for the client of the demanded results. Natural Language Processing is the term of this technique. To understand what is written in a behavior that at this time only humans can understand, becomes more relevant for large text collections.

Within the framework of this project, we have decided to use a Kaggle competition entitled *20 Newsgroups*. In such competitions on www.kaggle.com, all interested parties have the opportunity to solve the given problem with their own approach. All submitted solutions will be evaluated and included in the running ranking. By June 13, 2021, a total of 23 solution approaches from around the world were submitted, of which one will be used for comparison once we have completed our work.

2 Dataset

The 20 newsgroups dataset is a popular choice in the fields of Natural Language Processing (NLP). It consists of around 20000 newsgroup documents on 20 different newsgroups. The dataset can be accessed on various websites such as kaggle or even directly fetched with a dedicated function of the popular Python library *scikit-learn*, whereby the latter is the chosen option to load the data in the following work.

The data itself is organized into 20 newsgroups which correspond to different topics each. The topics are not necessarily related to each other, some of them are related, e.g. **comp.sys.ibm.pc.hardware** / **comp.sys.mac.hardware**, while others are not, e.g **misc.forsale** / **soc.religion.christian** [1]. In Figure 1 the newsgroups are visualized in a pie chart representing their distribution.

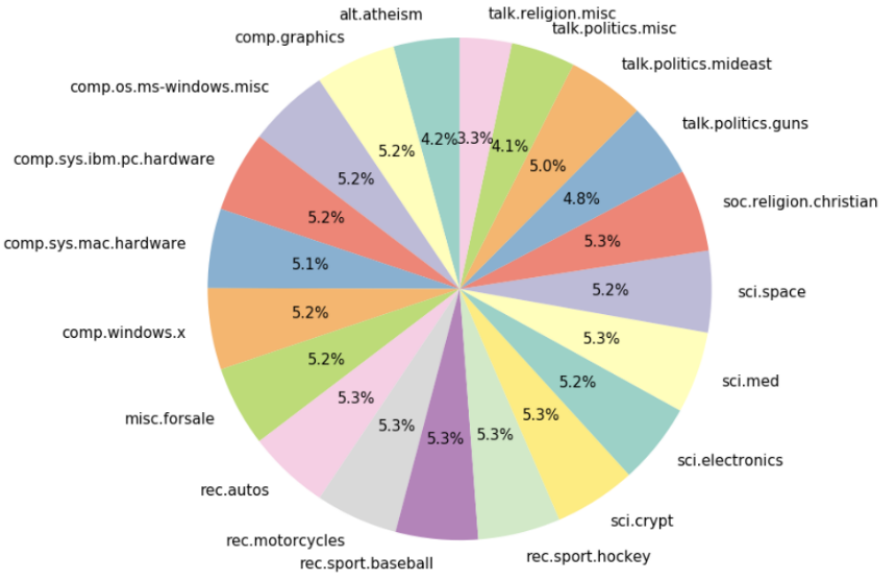

Fig. 1. Pie chart representing the distribution of each newsgroup of the dataset. Source: [2]

Each record of the dataset is a text that contains up to six headers, such as the mail of the user that posted on the topic (*From*), the subject matter (*Subject*), the university the user is affiliated with (*University*) or the date of the post (*Date*).

3 Preprocessing and Feature Engineering

In this project, we used NLP which is a subfield of Artificial Intelligence (AI) that is used to train the machine to understand human language. In order to feed modeling with our data, we first need to preprocess it. This includes removing stop words, lemmatizing, stemming, tokenization, and vectorization.

To preprocess the data, we first removed all the different headers of the 20 newsgroups dataset. We then loaded the *NLTK* library to remove the stopwords from our corpus, as they do not provide any further information to our text classification models. Since the posts in the newsgroups mostly are in everyday language, the phrases certain contracted words are decontracted, e.g. from won't to will not or in general from the ending n't to not. Subsequently, any occurring accents and umlauts are replaced, e.g. ä to ae. Next, we have loaded the *SnowballStemmer* from the *NLTK* library and thereby created a version of the preprocessed text with the word stems only. This aims to reduce the amount of the data and to gain better accuracy. Finally, we have added the option to use n-grams, which are sequences of n words that are commonly used in NLP models.

In the process of text cleansing, with the aim of cleaning text from surrounding noises, we applied three main steps:

Sentence Splitter: to split large files of raw text into sentences as the smallest unit of conversations. Tokenization: All the text strings are processed only after they have undergone tokenization, which is the process of splitting the raw strings into meaningful tokens as the minimal unit that a machine understands and processes at a time. Stemming: to obtain the root word of the given words known as the stem.

We took advantage of word embedding as a class of approaches in order to represent vector representations for the words in our document. To do this, we used Keras Embedding Layer which requires the input data be encoded as an integer and be represented by a unique integer. We did this using the Tokenizer API provided by Keras. We tokenized all words in the train set so that each word in the text gets a unique index. This helped us in creating dictionary mapping words. Consequently, by defining *word_index*, we made a dictionary mapping words to their respective index.

We used Global Vectors for Word Representation (GloVe), as a pre-trained word embedding for obtaining vector representations for words." Glove enabled us to intuitively transform each word in our corpus text into a position in high-dimensional space meaning to place similar words together. Using a function to read the content of the Glove vector file, we got a dictionary that maps the words to their respective word embeddings. After opening the Glove vectors file, we need to loop through each line in the file and split the line by every space into each of its components. After splitting the line, we make the assumption the word does not have any spaces in it, and set it equal to the first (or zeroth) element of the split line.

The last step in preprocessing our data is vectorization as the process of converting the text data into a machine-readable form. In the process of embedding

matrix definition, we assigned a zero vector to all words which were not included in the Glove. We splitted the data into a training set and a validation set and then loaded pre-trained word embeddings into an embedding layer. Next, we loaded the pre-trained word embeddings matrix into an embedding layer. We have set trainable to True so that we can train the data with updating weights.

In the next step, we used *scikit-learn*'s *CountVectorizer* to convert text documents to a vector of term/token counts. Once we transformed the text into numbers, we took advantage of TF-IDF as a means to score words in our data in a way to be fed to algorithms such as Naive Bayes.

4 Modelling

Within the following pages we will give an overview on the used models. In this context, we will also list performance indicators for each of the individual models.

4.1 Random Forest

Random forest consists of a large number of individual decision trees that operate as an ensemble. Each individual tree in the random forest spits out a class prediction and the class with the most votes becomes our model's prediction.The decision tree algorithm is quite easy to understand and interpret. But often, a single tree is not sufficient for producing effective results. To use the Random Forest, we have splitted our preprocessed cleaned data into a training set and test set. Then we applied the model and The results we got: Accuracy score: 0.641, Precision score: 0.65, Recall score: 0.64, F1 score: 0.64. We decided to take Random Forest because in the data understanding we found certain words that may indicate easy to classify the text. We hoped that it could find these patterns and classify them. Additionally it is less sensitive to label noise.

4.2 Naive Bayes

A Naive Bayes classifier uses probability theory to classify data. Naive Bayes classifier algorithms make use of Bayes' theorem. The key insight of Bayes' theorem is that the probability of an event can be adjusted as new data is introduced. What makes a naive Bayes classifier naive is its assumption that all attributes of a data point under consideration are independent of each other. We used Naive Bayes as our base classifier. It is a probability based classifier which treats each input (words) independently. We used MultinomialNB model for our dataset.To use Naive Bayes, we took 20 newsgroups Then we preprocessed the data by removing stopwords, punctuation, and finally normalizing the words by using lowercase for all words.

4.3 Linear SVM with SGD Classifier

Support Vector Machines (SVMs) are widely used for classification problems. The idea is to create a hyperplane or line to separate the data into classes. We give the model sets of labled training data for each category and then it will be able to categorize new text. The model works through how many times every word appears in the text and dividing it by the total number of words.

On the other hand, Stochastic Gradient Descent (SGD) Classifier is used as one of the most popular algorithms to perform optimization in the area of linear classifiers, such as the Linear SVM or Logistic Regression, but also for a wide range of other machine learning algorithms. The approach can be effectively used on large-scale data sets ($> 10^5$ samples) or with a large number ($> 10^5$) of features where other methods may lead to extremely long fit times or be infeasible to use beyond a few thousand samples or features.

Gradient descent in general is an iterative optimization algorithm for finding the local minimum of a differentiable function. It is called stochastic because it randomly selects data points at each iteration to calculate the derivative. This reduces the computation time enormously, especially when a small number of data points (batch) is sampled instead of one data point at each step. In the context of this work, we have chosen the linear Support implements equalized linear models with SGD learning. The gradient of the loss is estimated each sample at a time and the model is updated along the way with a decreasing strength schedule.

We used scikit learn pipeline class as a compound classifier and then constructed the classifier pipeline using the SGDClassifier algorithm. We set up different parameter values such as loss = 'hinge', penalty = 'l2', alpha1e-3, random_state = 42, max_iter = 5, tol = None and then we fit the model to the training data and ran the data into the test model. For the given parameter settings, the SGDClassifier implements a Linear SVM. In the last step, we calculated mean accuracy of predictions and generate labelled performance metric. Consequently, the result we got was: Accuracy score: 0.71, Precision score: 0.72, Recall score: 0.71, F1 score: 0.64.

4.4 Logistic Regression

After creating a 70/30 train-test split of the dataset, we have applied logistic regression which is a classification algorithm used to solve binary.classification problems. The logistic regression classifier uses the weighted combination of the input features and passes them through a sigmoid function. Sigmoid function transforms any real number input, to a number between 0 and 1.After the model fitting, we got a result as follows: Accuracy score: 0.728, Precision score: 0.728, Recall score: 0.724, F1 score: 0.734

4.5 LSTM

Long short-term memory (LSTM) is an advanced way of recursive neural networks (rnn) and helps the algorithm of understanding text, picture for multiple

steps. In a short description, it could be said, that it is learning by experience. Learning from training data which information in for the output. It is like a short-time memory, how human beings do have and remembers the values of the past from previous computations on the training data. Best examples are voice assistance which are able to keep a conversation.

Also in text mining, especially in our case understanding the provided texts and understanding it in preprocessed matter for prediction, will serve us a good algorithm for modelling our model. With an epoch value of max. 30 the Model computed in the 59th Epoch the results with an accuracy level of $0.8810 \approx 88\%$ in predicting.

4.6 Artificial Neural Networks

Artificial Neural Networks (ann) are information processing systems whose mechanism is modeled on the functionality of biological neural circuits (Source: [4]). The term neural networks includes a variety of models and learning methods of data mining. On closer inspection, these can be classified as nonlinear statistical models. For example, there is the simple one layer neural network which we used for the analysis of our data set.

Simple One Layer Neural Network The model, also called single layer perceptron network, is the first proposed neural model created. Thereby, weighted vectors make up the content of the local memory of the neurons. A single-layer perceptron is calculated by calculating the sum of the input vector, each with the value multiplied by the corresponding element of the vector of weights. At the output, the value displayed is the input of an activation function.

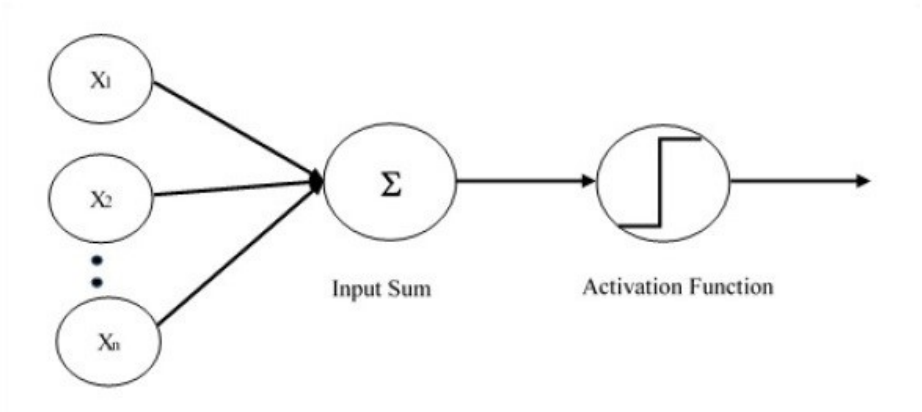

Fig. 2. Illustration of a single layer perceptron. Source: [4]

5 Evaluation

For testing and training we used the holdout method by splitting out dataset up into 70% training and 30% testing data. We were able to apply this method because we had a lot of sample data available. We used this method because we wanted to avoid overfitting. Because we balanced the dataset in the prepossessing, stratifying sampling wasn not necessary.

For the evaluation we were especially focusing on precision and f1-score because It is important for us that our model will classify text correctly.

How good are our models in predicting the unseen records in comparison to each other.

To compare our models in their performance we use for every model their output on precision, recall, f1-score and accuracy. What we do not take in part of this evaluation part is the runtime complexity of each model. This does not interest us at this point.

In the table below we see a listing of these values for each model. logistic regression: After creating a 70/30 train-test split of the dataset, we have applied logistic regression which is a classification algorithm used to solve binary.classification problems. The logistic regression classifier uses the weighted combination of the input features and passes them through a sigmoid function. Sigmoid function transforms any real number input, to a number between 0 and 1.After the model fitting, we got a result as follows: accuracy score: 0.728, precision score: 0.728, recall score: 0.724, f1 score: 0.734.

Model	**Precision**	**Accuracy**	**F1-Score**	**Rank**
Naive Bayes	0.756	0.748	0.734	1
Logistic Regression	0.728	0.727	0.721	2
Linear SVM	0.716	0.71	0.716	3
Random Forest	0.65	0.641	0.64	4

Number 1 in the rank of the best model in prediction the class / label on our test set is Naïve Bayes. The probability of the event before seeing (prior probability) gives us an accuracy score of 0,748 $\approx$ 75% in correct prediction. According to this model the f1-Score (including the false predicted classes by the model) returns us a score of 0,734 $\approx$ 74%. One shortcoming could be the ignoring / isolating of Noise Points in our Training Data.

But as mentioned, depending on our actual case it ´s acceptable in comparison to other cases like medical or pharmaceutical datasets in predicting diseases for an example.

According to our use case, predicting the newsgroup the words belong to by learned models on the training set, we do have the opportunity to not setting our focus on recall rather than precision and the f1-score. The accuracy on the hand gives us the relation on precision and recall.

In the table below (output from code) you see the results from the model Naive Bayes, which shows us for each class / label the precision p, recall r, f1-

Score, support and in the bottom the accuracy score. Additionally to that the micro-average and macro average score are also calculated. In order of caring less in the time complexity and the approach of efficient workflow we waive for plotting the confusion matrix as graph.

```
Training baseline model.
NB BASE LINE
                          precision    recall  f1-score   support

              alt.atheism       0.39      0.69      0.50       160
            comp.graphics       0.66      0.68      0.67       195
  comp.os.ms-windows.misc       0.72      0.69      0.70       197
 comp.sys.ibm.pc.hardware       0.65      0.75      0.70       196
    comp.sys.mac.hardware       0.81      0.76      0.78       193
           comp.windows.x       0.84      0.83      0.84       198
             misc.forsale       0.81      0.76      0.79       195
                rec.autos       0.80      0.73      0.76       198
          rec.motorcycles       0.81      0.72      0.76       199
       rec.sport.baseball       0.91      0.81      0.86       199
         rec.sport.hockey       0.92      0.88      0.90       200
                sci.crypt       0.80      0.74      0.77       198
          sci.electronics       0.76      0.62      0.69       197
                  sci.med       0.90      0.83      0.86       198
                sci.space       0.89      0.80      0.84       197
   soc.religion.christian       0.74      0.86      0.79       199
       talk.politics.guns       0.67      0.79      0.72       182
    talk.politics.mideast       0.87      0.87      0.87       188
       talk.politics.misc       0.62      0.62      0.62       155
       talk.religion.misc       0.56      0.39      0.46       126

                accuracy                           0.75      3770
               macro avg       0.76      0.74      0.74      3770
            weighted avg       0.76      0.75      0.75      3770

0.7488063660477453
```

Short Recap about Precision, Recall, Accuracy and F1-Score
Based on the values we got for true positive, false positive, true negative and false negative we calculated the needed values.

Also depending on the business-case some values are much higher ranked in priority as others.

$$precision = \frac{TP}{TP + FP}$$

$$recall = \frac{TP}{TP + FN}$$

$$F1 = \frac{2 \times precision \times recall}{precision + recall}$$

$$accuracy = \frac{TP + TN}{TP + FN + TN + FP}$$

$$specificity = \frac{TN}{TN + FP}$$

6 Hyperparameter Tuning

The process of choosing the parameters that give the best result for a machine learning model, is called hyperparameter tuning. To do so, we look up the parameters of the given model, then we set the combination of the parameters we want to test on the model. These are then fed to the GridSearchCV of scikit-learn, which optimizes the parameters of the model by a cross-validated grid-search over the parameter grid that we have defined. We use a default 5-fold cross-validation and since in our case the 20 newsgroups are a multiclass classification problem, a StratifiedKFold is being used. Finally, the model is trained on the 20 newsgroups data. The attribute *best_params* gives back a dictionary of the parameter combinations that gained the best results on the model.

Most of the parameters we have chosen initially were set to the best of our knowledge and the experience of the NLP community that we found on different Kaggle challenges and Keras blogs using the 20 newsgroup dataset. Since tuning the hyperparameters for each model would have been out of the scope of this work, we have decided to leave out the neural network models as they have far more parameters than the statistical models, such as logistic regression.

The results of the grid search gave us a lower accuracy than what we had with our initially chosen parameters. However, most of the parameters were the same as we have chosen, e.g. L2 for penalty or using IDF for the TfidfTransformer. In the case of the numeric parameters, e.g. the regularization term alpha, the results were different from ours. For example, for Naive Bayes the grid search yielded 73% accuracy for $\alpha = 0.0001$ whereas we initially have chosen $\alpha = 0.015$ and yielded 74% accuracy (see Chap. 5). This might be first due to the choice of the wrong parameter combinations for the grid search and also to different measures that could have been taken in the preprocessing of the data. But since the increase of hyperparameters also increases the time complexity and memory consumption, a thorougher tuning was not possible in the given time.

7 Comparison to other competitors on Kaggle in 20 NewsGroup Dataset

We took comparison with the author of the project "Text Classification (LSTM, BERT, ML models)" (Source: [3]). Introducing on his approach he followed after his preprocessing and splitting his dataset into training and test subset an overview of his data for a first overview and absolute Values. Counting the number of Words in total, tokenizing them, cleaning by removing stop words and combining as list were part of his cleansing for the further Data Mining operations.

Continuing with his preprocessing steps of bag of words (BOW) and TF IDF Method prepares his data structure for model. After splitting it with the SMOTE-Technique he used Logistic Regression as model for predicting the unseen test data. The model was also applied with TF IDF values on his training data to expand the possible results on Model.

The next model he used is Naïve Bayes which he also used on TF IDF. To prevent a variation of different accuracy scores he used Stratified K-Fold Cross validation to improve his model on the by this method splitted training data. In comparison to us we implemented TF IDF directly in our model codeblocks and not separately after without using TF IDF in our model. His approach is to collect as many evaluation results on his models to maximize the validation for his machine learning techniques.

Increasing the performance of artificial neural network for natural language processing we use long short – term memory (LSTM). The results of the computing by these models shows us following results. The higher the epochs (number of recursive passage) the accuracy score gets higher. The rate of false prediction which leads to false positive or false negative values gets by this model minimized and that's what we see in the results.

With an epoch value of max. 30, the model computed in the 59th epoch the results with an accuracy level of $0.8810 \approx 88\%$ in predicting.

In comparison with our other values of the other models it seems to be attractive for the prediction by the computed accuracy. Adjustments in the parameter of batches could give more precisely results, on the other hand the runtime could expand to an amount which does fits the purpose. The higher the batches the less adjustments are necessary. Usually a size range of $32 - 512$ is commonly used for LSTM.

References

1. 20 Newsgroups Data Set, http://qwone.com/ jason/20Newsgroups/. Last accessed 13 Jun 2021
2. Text classification - simple way to organize your data, https://krakensystems.co/blog/2018/text-classification. Last accessed 13 Jun 2021
3. Text Classification (LSTM , BERT, ML models) https://www.kaggle.com/dikshabhati2002/text-classification-lstm-bert-ml-models
4. TensorFlow - Single Layer Perceptron, $https : //www.tutorialspoint.com/tensorflow/tensorflow_{s}ingle_{l}ayer_{p}erceptron.htm$
5. Using Stochastic Gradient Descent to Train Linear Classifiers https://towardsdatascience.com/using-stochastic-gradient-descent-to-train-linear-classifiers-c80f6aeaff76. Last accessed 13 Jun 2021