NLP Topic Classification - A Machine Learning Approach

Andreea Zelko
s4311833
a.c.zelko@student.rug.nl

Matej Kucera
s4551192
m.kucera@student.rug.nl

Himawan Indra Bayu
s5719895
h.i.bayu@student.rug.nl

Abstract

Analyzing text from documents or social media can be helpful when it comes to identifying unseen patterns, interesting insights, or trends. There are various methods of analysis such as topic classification or sentiment analysis. This paper focuses on the multi-class problem of topic classification for textual product reviews.

Six machine learning algorithms are compared, namely Naive Bayes, Decision Tree, Random Forest, k-Nearest Neighbors, Support Vector Machine, and Classifier Ensemble. For each machine learning model, best hyperparameter values are selected using a grid search and their performance in multi-class classification is showcased. The results show that the ensemble method achieves the highest accuracy, whereas Naive Bayes and Support Vector Machine are in second place with a slight difference in accuracy. Bag-of-words and TF-IDF vectorizers were also compared, with TF-IDF achieving better results with the voting classifier.

1 Introduction

Text classification is one of the tasks of machine learning that is used in several sectors or industries with various purposes. As the amount of digital information grows, the need to classify or categorize the information also increases. For example, text classification has been used to develop a chatbot (Bird et al., 2021) (Amer et al., 2021) or support information assurance and security (Atallah et al., 2001).

This study investigates the performance of prominent classifiers, such as Naive Bayes, k-Nearest Neighbors, Decision Tree, Random Tree, and Support Vector Machines, for multi-class classification by exploring the best feature preprocessing techniques and tuning the classifier hyperparameters.

1.1 Research Questions

To explore further classification performance, it is essential to tune the hyperparameters of the classifiers or choose feature transformation of the data as an approach to achieve the highest accuracy of prediction. Thus, the main questions of this study are the following:

1. Which vectorizer used in the preprocessing step achieves the best results??
2. Which machine learning classifier that achieves the highest accuracy on multi-class classification?

The machine learning models are tested on a dataset of 6000 reviews and evaluated using cross-validation. Best hyperparameter values are selected using a grid search. The results show that a soft voting ensemble of all classifiers except the decision tree performs the best, with 92% accuracy on previously unseen data.

Section 2 will illustrate related works to this study, while section 3 and section 4 will cover data exploration and methods that are used in this study. The results will explained in section 5 and it will be discussed in section 6 and will be concluded with section 7. Section 8 will show supporting data for section 4, namely confusion matrices for all classifiers.

2 Related Work

Research was conducted by (Ali et al., 2021) to classify text in the Urdu language. Several feature vector methods, such as One-Hot-Encoding,

TF and TF-IDF, and deep learning models, such as Deep Neural Networks (DNN), Recurrence Neural Networks (RNN), and Convolutional Neural Networks (CNN), were employed in the experiment. The result showed by combining TF-IDF with DNN, the prediction of DNN outperformed RNN and CNN at 84% of accuracy. When comparing the deep learning models to machine learning classifiers, it was noticed that DNN and RNN were better than machine learning classifiers since the accuracy of DNN and RNN were 84% and 81%, respectively, and the highest accuracy of machine learning was 80%. Another research by (Rabut et al., 2019) explored different word embedding methods, such as Word2vec, Word2vec with POS Tag, FastText, and FastText with POS Tag. Several hyperparameters were chosen for the classifier model, such as the ReLU activation function, adaptive learning optimizer, and loss function. The model achieved 83.2% accuracy in classifying documents.

3 Data

In this section, we will describe the characteristics of the data, including the way we split it and the distribution of the sentiment of the reviews.

3.1 Dataset

The dataset used in this experiment is a set of 6,000 textual reviews about several categories, namely Books, Camera, Dvd, Health, Music, and Software in English. The dataset is preprocessed by a tokenizer. Each review is a plaintext string with two associated labels, `category` and `sentiment`. Table 1 shows all possible label values.

Category	dvd, software, health, camera, music, books
Sentiment	pos, neg

Table 1: Data set: Category and Sentiment Labels

3.2 Data exploration

To investigate the dataset, we performed some basic data exploration tasks. Figure 1 shows the data distribution over all label combinations. The data is distributed evenly between all 12 label combinations. This shows that the dataset can be used without introducing any bias due to class imbalance.

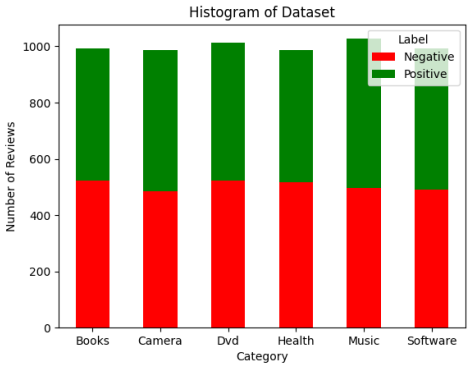

Figure 1: Category histogram

Additionally, it is important to investigate the number of tokens per data point. Figure 2 shows the distribution. As expected,m most data points are fairly short. However some outliers are visible, with the maximum value being over 5000 tokens.

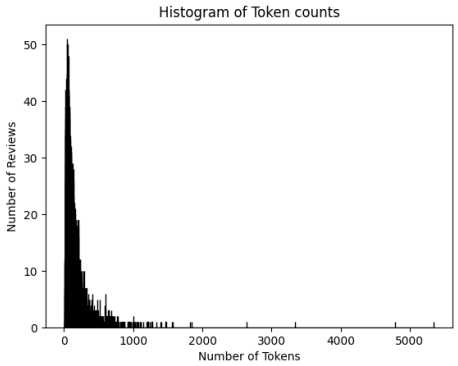

Figure 2: Distribution of data point token count

3.3 Data split

The data was split into three parts. A training and testing set were created to train the machine learning algorithms on. A dev set was left out as an unseen dataset to validate the final model. The split was 80-10-10 for train-test-dev respectively.

4 Method

The `sklearn` Python library provides implementations of common machine learning algorithms (Pedregosa et al., 2011). These implementations were used in our experiments, as mentioned later in this section. To evaluate each machine learning model, the confusion matrix and accuracy were used. The `GridSearchCV` class was used to cross-validate all results with `cv=5`. The source code can be accessed publicly on Github. Tested

hyperparameter values are shown in tables for each algorithm, with the chosen best values highlighted in bold typeface.

4.1 Features

Experiments were carried out in order to identify the best feature set. This sub-section will elaborate on what steps were taken. To be able to compare the results of each method, the same classifier was used while making sure to not change any of the parameters of the classifier. The Multinomial Naive Bayes classifier with default parameters was chosen for this task.

4.1.1 Vectorizer parameters

Experimenting with the attributes of the vectorizers was the first step in determining how to create the feature set. To do this, it was desired to try all combinations of the possible values for each attribute, as shown in Table 2.

Hyperparameter	Values
`max_df`	1.0, 0.95, **0.90**, 0.85
`ngram_range`	**(1, 1)**, (1, 2), (2, 2), (1, 3), (2, 3), (3, 3)
`max_features`	None, 100, 1000, **10000**
`preprocessor`	**custom_preprocessor**, identity

Table 2: Hyperparameter Values of Vectorizers

A custom preprocessor was implemented so that only tokens matching the `"^[a-zA-Z]+(?:'\w+)?$"` regular expression would be kept. This was done to only keep alphabetical strings since the dataset also contains non-alphabetical tokens such as `. . .`. The other evaluated preprocessor was the self-explanatory identity function.

To be able to compare the results of all combinations and cross-validate results, GridSearchCV was used. The results on these experiments will be discussed in section 5.

4.1.2 Other feature experiments

Aside from testing the best parameters for the two vectorizers, other experiments were also carried out.

In an attempt to improve the results, using the unused labels as features was implemented. The GridSearchCV implementation was ran again.

Additionally, different preprocessors were also tested. The goal was to check if lemmatizing, stemming or tagging would improve the results of the classification.

Lastly, combining the two vectorizers was also tried. This was carried out using the FeatureUnion concatenator.

The results of these experiments will be explained in section 5.

5 Results

5.1 Features

As described in section 4, experiments were carried out to find the best way to prepare the features for the classification task. LSA and PCA were also tested, but it was found that they were extremely slow due to the number of features per data point and they decreased the classification accuracy.

After testing, it was found that the best parameters for the topic classification task were:

```
max_df=0.9,
max_features=10000,
ngram_range=(1, 1),
preprocessor=custom_preprocessor.
```

Even though this paper is not performing sentiment analysis, it is interesting to compare the best hyperparameters for these two tasks. The best parameters for sentiment analysis were:

```
max_df=0.95,
max_features=None,
ngram_range=(1, 3),
preprocessor=identity
```

A noticeable difference here is the n-gram range. Sentiment analysis seems to benefit from n-gram, while topic classification does not. This is as expected, since sentiment analysis requires word combinations to be able to detect phrases like `don't like` and `not good`. Topic analysis doesn't benefit because it does not require word combinations.

For the other feature tests, the results were quite simple. Using the unused labels as features resulted in lower accuracy, so they were excluded from the default feature preparation method. Lemmatizing, stemming and tagging did not lower the accuracy, but they also did not improve it. Lastly, combining the vectorizers using the FeatureUnion concatenator also decreased the accuracy of the system.

At the end of the testing, the best setup was:

```
CountVectorizer(
    max_df=0.9,
    ngram_range=(1, 1),
    max_features=10000,
    preprocessor=custom_preprocessor,
    tokenizer=identity
)
```

5.2 Classifiers

Each of the following classifiers was optimized using a grid search over possible hyperparameter values using cross-validation with 5 folds. The grid search allowed various combinations of hyperparameters to be tested with ease and it returned the combination with the best accuracy. The `GridSearchCV` class from sklearn was used for this with `cv=5`. Confusion matrices for each classifier are report in section 8.

5.2.1 Naive Bayes

The Naive Bayes classifier uses a probability based classification method. It takes only two hyperparameters, which were iterated over using grid search. The possible values can be found in Table 3.

Hyperparameter	Values
`alpha`	0.1, 0.5, **1**, 2, 5
`fit_prior`	**True**, False

Table 3: Hyperparameter Values for Naive Bayes

After running the grid search, it was found that the best performing parameters are actually the default ones: `alpha=1` and `fit_prior=True`. The accuracy was 90%.

5.2.2 k-Nearest Neighbors

The k-Nearest Neighbors (k-NN) classifier uses similar features to assign new data points to the correct class. In this experiment, k-NN was implemented with various hyperparameters, as shown in Table 4. At the beginning of the predicting process where `k = 1`, the accuracy was only 64%. By increasing the k value, it was noticed that the accuracy is improving to its peak of 84.8% with `k = 21`. However, since the accuracy difference between `k = 11` and `k = 21` was not high, it was merely 0.5%; thus, `k = 11` was chosen to avoid overfitting. The accuracy values for varying k values are shown in Table 5.

Hyperparameter	Values
`k`	1, 3, 5, 7, 9, **11**, 13, 15, 17, 19, 21
`Weight`	Uniform, **Distance**
`Distance Metric`	**Euclidean**

Table 4: Hyperparameter Values of k-Nearest Neighbors

k values	Accuracy (%)
1	64.4
3	76.5
5	81.3
7	83.1
9	81.3
11	**84.4**
13	84
15	84.5
17	84.4
19	84.4
21	84.8

Table 5: The prediction accuracy based on the k value

5.2.3 Support Vector Machine

The Support Vector Machine (SVM) uses a hyperplane to separate the two classes of data points. When a new data point is given, SVM uses the hyperplane to classify the new data point into the correct class. In this experiment, two different kernels were tested to compare the performance.

Kernel function	Accuracy (%)
`Linear`	**90.5**
`Radial Basis Function`	90.1

Table 6: Comparation of accuracy from different kernels

When the kernels were performed to classify the train and dev sets, both kernels achieved a high accuracy with a difference of only 0.4%. However, since the Linear Kernel Function performed faster than the Radial Basis Function, the Linear Kernel function is more preferable to be chosen.

5.2.4 Decision Tree

The decision tree classifier is a fairly simple classifier model which offers simplicity and interpretability. As it is a tree, the hyperparameters all relate to the tree node splits and depth. Table 7 shows the hyperparameter values which were tested. Unfortunately, the simple nature of a decision tree limits its usability in this task and the highest achieved accuracy was around 76%. This is still quite good considering how simple and efficient a decision tree is but it was found that more advanced methods like the random forest can achieve much higher accuracy.

An interesting fact is that the decision tree with default parameters had a maximum depth of over 70 and a similar accuracy. However it was found that setting a `max_depth` of 30 very slightly im-

Hyperparameter	Values
max_depth	10, 20, 25, **30**, 35, 40, None
criterion	**gini**, entropy, log_loss
min_samples_leaf	**1**, 2, 4
min_samples_split	**2**, 4, 8
max_features	**None**, sqrt, log2

Table 7: Hyperparameter Values for Decision Tree

proves the accuracy with a much smaller tree. The tree performed notably poorly on the `health` class as can be seen in the confusion matrix (precision on the health class is 0.53). This could be explained by the fact that the reviews tagged as `health` are quite confusing even to humans. After reviewing some of them, it was observed that the health category contains items from various categories which are not very closely related, such as humidifiers, dryers, sleeping aids, razors and hats just to name a few.

5.2.5 Random Forest

A random forest is an ensemble of decision trees which are randomized to produce different outputs. The core concept is that an ensemble of weak classifiers can be used a strong classifier. Table 8 shows the hyperparameters values tested. Note that the values are similar to the decision tree hyperparameters, with one important added parameter, `n_estimators` which controls the number of decision trees in the random forest classifier.

Hyperparameter	Values
n_estimators	10, 100, **500**, 1000, 5000
max_depth	10, 20, 25, 30, 35, **40**, None
criterion	**gini**, entropy, log_loss
min_samples_leaf	1, **2**, 4
min_samples_split	**2**, 4, 8
max_features	**sqrt**, log2, 0.001

Table 8: Hyperparameter Values for Random Forest

The random forest was able to achieve high accuracy of around 87%, notably performing significantly better than a single decision tree.

5.2.6 Voting Ensemble Classifier

To take the idea of an ensemble of classifiers further, a `VotingClassifier` was used which combined the previously described classifiers with the exception of the decision tree. The soft voting method was found to be more accurate. This classifier combines the already quite strong classifiers described previously and improves upon them by combining them. It can also avoid pitfalls of each individual model since it has 4 different classifiers which can balance out their strengths and weaknesses for a more universal classifier which generalizes better than its individual parts.

Testing revealed that the `Tf-Idf` vectorizer performs better for this classifier than the `Count` vectorizer, therefore it was used for the final result. This classifier using the best hyperparameters for each individual classifier (as described above) achieved an accuracy of 92%.

Model	Accuracy (%)
Naive Bayes	91
kNN	84
SVM	91
Decision Tree	76
Random Forest	87
Voting Ensemble	**92**

Table 9: Accuracy of all models

6 Discussion

Regarding the results from different feature sets and classifiers, it was found that TF-IDF works better than CountVectorizer in this experiment for the advanced classifiers. CountVectorizer did actually provide better results for the Naive Bayes model because of its simplicity. TF-IDF emphasizes how many times the words appear in the corpus and the relative importance of the words. Thus, combining TF-IDF with advanced classifiers can improve the accuracy of classification.

Conversely, the classifiers showed a different percentage of accuracies where the Decision Tree was the lowest at 76%. The accuracy may be caused by overfitting where the classifier keeps generating new nodes, making the tree too complex. Meanwhile, Naive Bayes and SVM had similar accuracy at 91%; this was because Naive Bayes uses independent assumptions of the words, making it work well in categorizing the data between classes. SVM is one of the powerful algorithms used to categorize data using a hyperplane as a boundary to distinguish the classes. Also, SVM has kernel functions that work well in handling complex, high-dimensional data. Though in this case the kernel functions were not used since the SVM achieved better accuracy with the linear kernel. This is because the data is very high-

dimensional and therefore the SVM can linearly separate it without using non-linear kernel functions, which cause overfitting.

To answer the research question. (1) TF-IDF is preferable in this study since it also highlights the importance of words, not only considering the frequency of the words in the corpus. It performs better with nearly all classifiers used in this paper. (2) The ensemble classifier was the best classifier with 92% of accuracy. However, since this method combines all individual classifiers, it can come at a high computational cost. Hence, Naive Bayes and SVM can be secondary options for handling text classification tasks if speed is of importance or the dataset is much larger.

7 Conclusion

To conclude, this study aims to compare the performance of different classifiers in text classification. Choosing TF-IDF features can improve the prediction accuracy of each classifier. The ensemble method was the best classifier, and Naive Bayes and SVM were the second-best classifiers. This shows that text classification is a relatively easy task even for these basic machine learning classifiers.

For further work, it can be interesting to compare the accuracy and performance of these classifiers on larger datasets, potentially with more classes. There is clearly a trade-off between accuracy and performance which could be interesting to investigate. Another interesting future idea could be to look at reviews in multiple languages and to see whether combining languages and some kind of translation or LSA preprocessing step would yield good results.

References

[Ali et al.2021] Daler Ali, Malik Muhammad Saad Missen, and Mujtaba Husnain. 2021. Multiclass event classification from text. *Scientific Programming*, 2021(1):6660651.

[Amer et al.2021] Eslam Amer, Ahmed Hazem, Omar Farouk, Albert Louca, Youssef Mohamed, and Michel Ashraf. 2021. A proposed chatbot framework for covid-19. In *2021 International Mobile, Intelligent, and Ubiquitous Computing Conference (MIUCC)*, pages 263–268.

[Atallah et al.2001] Mikhail J. Atallah, Craig J. McDonough, Victor Raskin, and Sergei Nirenburg. 2001. Natural language processing for information assurance and security: an overview and implementations. In *Proceedings of the 2000 Workshop on New Security Paradigms*, NSPW '00, page 51–65, New York, NY, USA. Association for Computing Machinery.

[Bird et al.2021] Jordan J. Bird, Anikó Ekárt, and Diego R. Faria. 2021. Chatbot interaction with artificial intelligence: human data augmentation with t5 and language transformer ensemble for text classification. *Journal of Ambient Intelligence and Humanized Computing*, 14(4):3129–3144, aug.

[Pedregosa et al.2011] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

[Rabut et al.2019] Benedict A. Rabut, Arnel C. Fajardo, and Ruji P. Medina. 2019. Multi-class document classification using improved word embeddings. In *Proceedings of the 2nd International Conference on Computing and Big Data*, ICCBD 2019, page 42–46, New York, NY, USA. Association for Computing Machinery.

8 Appendix

This section reports confusion matrices for each classifier described in section 4. For improved formatting, the tables are shown on the next page.

Class	Precision	Recall	F1-Score	Support
Books	0.92	0.91	0.92	90
Camera	0.83	0.92	0.87	112
DVD	0.94	0.88	0.91	101
Health	0.99	0.80	0.88	94
Music	0.92	0.96	0.94	114
Software	0.83	0.91	0.87	89
Accuracy		0.90 (600)		
Macro Avg	0.90	0.90	0.90	600
Weighted Avg	0.90	0.90	0.90	600

Table 10: Classification Report for Naive Bayes

Class	Precision	Recall	F1-Score	Support
Books	0.88	0.83	0.86	90
Camera	0.74	0.92	0.82	112
DVD	0.79	0.86	0.82	101
Health	0.89	0.63	0.74	94
Music	0.87	0.78	0.82	114
Software	0.84	0.91	0.87	89
Accuracy		0.82 (600)		
Macro Avg	0.84	0.82	0.82	600
Weighted Avg	0.83	0.82	0.82	600

Table 11: Classification Report for k-Nearest Neighbor

Class	Precision	Recall	F1-Score	Support
Books	0.93	0.92	0.93	90
Camera	0.93	0.87	0.90	112
DVD	0.90	0.89	0.90	101
Health	0.86	0.88	0.87	94
Music	0.92	0.91	0.92	114
Software	0.86	0.93	0.89	89
Accuracy		0.90 (600)		
Macro Avg	0.90	0.90	0.90	600
Weighted Avg	0.90	0.90	0.90	600

Table 12: Classification Report for Support Vector Machine

Class	Precision	Recall	F1-Score	Support
Books	0.88	0.80	0.84	90
Camera	0.92	0.68	0.78	112
DVD	0.89	0.74	0.81	101
Health	0.53	0.93	0.68	94
Music	0.91	0.81	0.86	114
Software	0.75	0.73	0.74	89
Accuracy		0.78 (600)		
Macro Avg	0.81	0.78	0.78	600
Weighted Avg	0.82	0.78	0.79	600

Table 13: Classification Report for Decision Tree

Class	Precision	Recall	F1-Score	Support
Books	0.90	0.94	0.92	90
Camera	0.97	0.81	0.88	112
DVD	0.94	0.87	0.90	101
Health	0.73	0.91	0.81	94
Music	0.94	0.92	0.93	114
Software	0.82	0.81	0.81	89
Accuracy		0.88 (600)		
Macro Avg	0.88	0.88	0.88	600
Weighted Avg	0.89	0.88	0.88	600

Table 14: Classification Report for Random Forest

Class	Precision	Recall	F1-Score	Support
Books	0.93	0.94	0.94	90
Camera	0.89	0.90	0.90	112
DVD	0.94	0.90	0.92	101
Health	0.95	0.86	0.91	94
Music	0.93	0.96	0.94	114
Software	0.86	0.93	0.89	89
Accuracy		0.92 (600)		
Macro Avg	0.92	0.92	0.92	600
Weighted Avg	0.92	0.92	0.92	600

Table 15: Classification Report for Voting Ensemble