Text Classification*

Szymon Molitorys[1,*,†], Bartłomiej Piłot[1,†], Kacper Smyrak[1,†]

[1]Faculty of Applied Mathematics, Silesian University of Technology, Kaszubska 23, 44100 Gliwice, POLAND

Abstract

In order to achieve our project objectives, we developed a text classifier using datasets derived from a collection of books. Each dataset comprises selected words along with their frequencies of occurrence. Our objective was to evaluate the performance of three different algorithms in classifying texts based on these datasets. To this end, we employed a decision tree, a Naive Bayes classifier, and a custom probability calculation algorithm to compare the accuracy and effectiveness of these methods. By testing our models on both larger and smaller datasets, we sought to understand the impact of dataset size on classification results. Our approach involved training the algorithms on a training dataset and then testing their performance on a separate test dataset to assess their predictive accuracy. The results demonstrated varying degrees of success across the algorithms, highlighting the strengths and weaknesses of each method in different scenarios. This comprehensive comparison provided invaluable insights into the practical applications of text classification techniques and their potential for further refinement and use in real-world data analysis tasks.

[1, 2, 3, 4, 5, 6]

Keywords

text classification, decision tree, Bayesian algorithm, custom probability algorithm, dataset comparison, word frequency analysis, machine learning, data analysis

1. Introduction

Data Preparation We begin by loading the training and testing datasets using the pandas library. The datasets contain three columns: "Word" (the word itself), "Frequency" (the number of times the word appears), and "Type" (the category to which the word belongs). Decision Tree: The objective is to develop a text classifier using the Decision Tree algorithm. The aim is to train a model that can accurately predict the type of a given word based on its frequency. This approach leverages the structure of decision trees to handle the categorical nature of our data effectively.

2. Methodology

2.1. Decision Tree Algorithm

Firstly in our Decision Tree Algorithm, words were mapped to integers. Since the Decision Tree algorithm requires numerical input, a dictionary was created that maps each unique word in the training dataset to a unique integer. This mapping allows us to convert the words into numerical format. Words in the testing dataset that do not appear in the training dataset are mapped to a value of -1, ensuring that the model can handle unseen words without causing errors.

*IVUS2024: Information Society and University Studies 2024, May 17, Kaunas, Lithuania

[1,*] Corresponding author

[†] These author contributed equally.

sm305126@student.polsl.pl (S. M. B. P. K. S.)

Both the training and testing datasets are split into two sets: features (X) and labels (y). The features include the word (now represented as an integer) and its frequency, while the labels are the types. The Decision Tree classifier is initialized with specific parameters:

- Criterion: The entropy criterion is employed to assess the quality of a split, utilizing information gain.

The maximum depth of the tree is tested with values of 5, 10, and 15, to observe the impact on model performance. The accuracy of the model is evaluated using the accuracy score function from scikit-learn, which compares the predicted types to the actual types in the testing dataset.

Mathematical Entropy

Entropy is a measure of randomness or impurity in the data. It quantifies the uncertainty involved in predicting the class of a given dataset. In the context of decision trees, entropy is used to determine the best way to split the data at each node.

The formula for entropy (H) is:

$$H = -\sum_{i=1}^{n} P(x_i) \log_2(P(x_i)) \tag{1}$$

where $P(x_i)$ is the probability of occurrence of class x_i.

In our text classification problem, entropy helps us to measure the impurity of our dataset with respect to the types of words. By minimizing entropy at each split, the decision tree aims to create the most informative branches, ultimately improving the model's classification accuracy.

2.2. Comparison of the number of words in the text

First we count the number of unique words in the text and then compare it with the number in the data set. The category with the most shared words with text is likely to be the answer. There are many ways to do this kind of comparison.

We can simply increment the counter in each category according to the number of times we come across the same word in the text, and then add up the number of occurrences. This has many advantages, it is fast and reliable. However, the algorithm can be fooled by texts that use a lot of specific words that are not related to the category. Moreover, the probability of a good answer depends on the prepared data set. The preparation of a good one is not a difficult task, but the preparation of the best one is very difficult. Therefore, our choice of method is the less difficult one.

There are other ways to calculate the number for each category, but the best results are still obtained by using neural networks, which usually require a lot more calculations and are forbidden by our supervisor.

2.3. Naive Bayes Classifier using Bag-of-words method

For the Naive Bayes Classifier we have used the Bag-of-words to prepare the data for the analysis. The algorithm will use the training dataset as the basis for the calculation of the probabilities of

test sets. The Bag-of-words method allows us to use discrete values instead of continuous values by counting the occurrence of unique words in the text without regard for their alphabetical or word order. The goal of this algorithm is to classify the document based of the highest value of conditional probability of chosen text x to be of class C_j which is based on the formula

$$P(C_j \mid x) = \frac{P(C_j)\, P(x \mid C_j)}{P(x)}$$

(1) which derived from Bayes' theorem. The $P(C_j)$ is the probability of the choosing the class from all the classes based on the number of documents for each class in comparison to the total number of documents, $P(x \mid C_j)$ is our likelihood of choosing text x given we have class C_j. The probability $P(x)$ is constant as we are not manipulating the contents of datasets and is the probability of choosing the text x from all the available texts. Because of that $P(x)$ will not affect the results of our calculations and can be omitted. Consequently the probability we want to calculated can be expressed as following using joint probability.

$$P(C_j, x) = P(C_j)\, P(x \mid C_j)$$

Now that we are calculating for each of the "feature" of the text which is for each word we can transform our current equation. Moreover since our classifier is "naive" we assume that each probability is independent from one another, giving use the following equation, substituting the text with its words denoted by w_i

$$P(C_j, w_1, w_2, ..., w_n) = P(w_1, w_2, ..., w_n, C_j) = P(x_1 \mid x_2, ..., x_n, C_j)P(x_2 \mid x_3, ..., x_n, C_j)$$

$$...P(x_{n-1} \mid P(C_j))P(x_n \mid C_j)P(C_j)$$

By assuming that probabilities are independent

$$P(x_i \mid x_{i+1}, ..., x_n, C_j) = P(x_i \mid C_i)$$

we have

$$P(C_j, x_1, x_2, ..., x_n) = P(x_1 \mid C_j)P(x_2 \mid C_j)...p(C_j) = P(C_j)\prod_{i=1}^{n} P(x_i \mid C_j)$$

(2) giving us

$$P(C_j \mid x) = P(C_j \mid x_1, x_2, ..., x_n) \propto P(C_j, x_1, x_2, ..., x_n) = P(C_j)\prod_{i=1}^{n} P(x_i \mid C_j)$$

Because of the insignificance of $P(x)$ in equation (1) we will therefore calculate our probability of text x being of class C_j by calculating $P(C_j, x_1, x_2, ..., x_n)$ and choosing the class for which the highest value of probability we will achieve.

All 3 datasets are first downloaded. The base training set is then grouped by the class of the document the words belong to and the sorted in descending order. The test datasets are also

sorted in descending order for each document. the sorting of the data sets allows us to see the patterns in the occurrence of words for each text and text class more clearly.

In order to begin we choose the text from the test dataset. Each of the texts will be checked for each of the available classes of text. After we choose the class we want to calculate the probability for we have to calculate the probability of choosing of each of the classes $P(C_j)$ (which has been denoted in code as **Pclass**) of the document using the following formula, where j indicates the index of the class.

$$P(C_j) = \frac{\text{Number of documents in class}}{\text{Sum of document numbers}}$$

In our case the number of documents of each class is even and we are working with three possible classes. As a result the probability $P(C_j)$ is equal 1/3. This probability will be our prior probability needed for the calculating the main Next we need to find the conditional probability
of the features (*words*) of the text which is denoted by $\prod_{i=1}^{n} P(x_i \mid C_j)$ from equation (2).

To do
this we need to know the probability of each word given they are in class C_j. We will calculate each of the probabilities $P(x_i \mid C_j)$ using formula

$$P(x_i \mid C_j) = \frac{N_i}{N_{\text{C total}}}$$

(3) where N_i is the number of words that occurred in test text from the base dataset(denoted in code as **CountWordBase**) and N_{Ctotal} is the total number of words in the dataset belonging to the chosen class (denoted in code as **CounttotalBase**).

Despite it being a correct formula for each factor needed to calculate our main probability a certain problem occurs. Since the test datasets and base datasets may not have the same set of words the resultant probability for a word that does not occur in base set will be equal zero. Consequently the value of the probability of $P(C_j, x_1, x_2, ..., x_n)$ will also be zero, falsifying our results. To counter this we will use modified formula (3) that uses Laplace smoothing in order to get rid of such problems. To use it we need to increment the numerator by certain value α (in this case we chose the value $\alpha = 1$) and denominator by value $\alpha |V|$ (where $|V|$ is the total number of unique words in dataset, denoted in code as (len(BaseClass))) Consequently the modified formula (3) looks as following

$$P(x_i \mid C_j) = \frac{N_i + 1}{N_{\text{C total}} + \alpha |V|}$$

(4) In code we implemented the +1 in numerator by setting the default value of **CountWord- Base** as 1.

The second problem we have encountered with the formula is the fact that for the num- ber of words in our texts and bases which reach more than 1000 the resultant probability $P(C_j, x_1, x_2, ..., x_n)$ reaches fraction of such small order of magnitude that they cannot be stored in the computer using standard variable formats such as float or double. to counter this we had to modify the formula for the $P(C_j, x_1, x_2, ..., x_n)$ so that it will result in values that are storable and computable using standard variable formats such as float or double. To do

this we have to express we took the logarithm of both sides. Because of the properties of the logarithm we can convert our multiplication into sum

$$ln(a * b) = ln(a) + ln(b)$$

Because the conditional probabilities for each word are in range between [0, 1] the log probabili- ties will have values in range [− inf, 0] but despite the possibility of reaching really high values the resultant log probabilities can be store in float and double variable format due to their low order of magnitude (in our test the values for log probabilities were reaching values between -1000 and -10000) which solved our problem. As for value of log probabilites and keeping the relation in term of greatness between them, those features are still preserved. According to the properties of $ln(x)$ function the values of greater value are less negative after using the $ln(x)$ function on them. This means that we can use the log probabilities instead of the ordinary ones calculated from equation (2). The resultant equation for log probabilities using the equation (4) is as following

$$\ln P(C_j, x_1, x_2, ..., x_n) = \ln P(C_j) + \sum_{i=1}^{n} \ln P(x_i \mid C_j)$$

after calculating the log probability we will have to choose the the highest value of the probability that were calculated for all available classes for chosen text. The class that has the highest value of probability (in this case least negative) will be chosen as the class of the test text. From the probability model we can write the function that uses our Naive Bayes classifier to asses the class of the test text $\hat{y}$.

$$\hat{y} = C_j$$

$$\hat{y} = \arg \max_{j \in \{1,2,...,J\}} \ln P(C_j) + \sum_{i=1}^{n} \ln P(x_i \mid C_j)$$

3. Experiments

3.1. Dataset Description

We utilized three distinct datasets in our experiments: one training dataset and two test datasets. These datasets were constructed from books and articles referenced in our study, with the words counted using a specially prepared Python script. Each dataset comprises between 2000 and 5000 rows of data. The books were evenly distributed among categories.

In our database, each entry is characterized by the word itself, its frequency of occurrence, and its category. This structured approach allowed us to conduct thorough experiments on our algorithm's performances across different datasets and categories.

3.2. Experimentation with Max Depth in Decision Tree

To find the optimal complexity of our Decision Tree model, we experimented with different values for the maxdepth parameter: 5, 10, and 15. This parameter controls the maximum

depth of the tree, influencing how well the model generalizes to unseen data. We approach the experiment with thesis that:

• Max Depth 5: With a shallow tree, the model may underfit, failing to capture the complexity of the data.

• Max Depth 10: This depth is a middle ground, potentially balancing the trade-off between underfitting and overfitting.

• Max Depth 15: A deeper tree may overfit the training data, capturing noise rather than the underlying pattern.

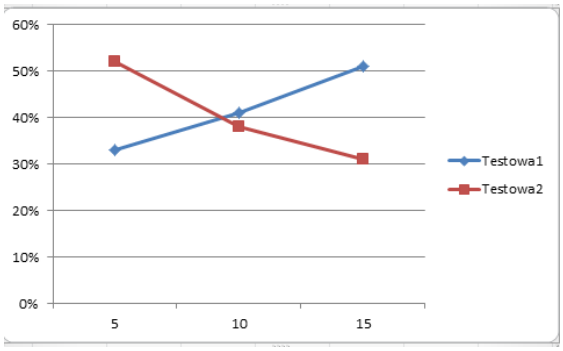

Figure 1: Decision Tree Comparison

It was surprisingly different. Depending on the database, the data behaved differently.

When working with a larger dataset, it is generally beneficial to use a greater maximum depth for the decision tree, as it allows the model to capture more complex patterns in the data. Conversely, for smaller datasets, a shallower maximum depth is preferable to avoid overfitting and ensure the model generalizes well to unseen data.

3.3. Experimentation with words occurance

For our experimental data, both of these algorithms have given the correct answers every time. This means that the dataset is close enough to the tests we provide. Running time increases when working with larger data sets. So good initial data is important. Testing with more examples is needed.

3.4. Experimentation with Naive Bayes classifier

To correctly asses the class of each of the texts in test sets we were using the training set as the base for calculating probabilities and assuming classes for the texts in test set 1 and test set 2. Additionally since the test datasets also contained the information on were the classes of the test texts we could use test sets 1 and 2 as a base instead of train- ing set. Below is the example of calculations of one pair of datasets on which we ran our program

i

Text class name	Assumed class name	log probability value
History	History	-2635.23
History	sport	-6616.30
History	cooking	-7078.77
sport	History	-3599.93
sport	sport	-1415.50
sport	cooking	-3536.01
cooking	History	-3416.40
cooking	sport	-3136.54
cooking	cooking	-1219.84

Table 1
Example of values for classification for base dataset "treningowa.csv" and test dataset "testowa1.csv"

The column with label "Assumed class" contains the name of the class that was chosen by the algorithm, the column with label "Text class" contains the true class name of the text. As we can see the most probable classes for each texts have the highest (least negative) values. In this case the algorithm calculated the highest values for correct classes for each text, resulting in 100% accuracy.

The algorithm creates tables such as the example and the uses **argmax** function to choose the result with the highest probability. Afterwards we can calculate the accuracy of the algorithm for each pair of datasets. The accuracy results of the algorithm running on dataset pairs specified before are as following Based on the values of accuracies that were presented in table

Base dataset name	Test dataset name	Accuracy percentage
treningowa	treningowa	100%
treningowa	testowa1	100%
treningowa	testowa2	100%
testowa1	testowa2	100%
testowa2	testowa1	100%

Table 2
Table of accuracy of the Naive Bayes classifier for pairs of datasets

2 we can see that Naive Bayes classifier has correctly guessed the classes of every text in the dataset. Therefore for the current amount of datasets used in the experiment the accuracy of our algorithm is 100%. Moreover for each pair of datasets the log probability values of guesses were significantly greater (more than 2 times in most cases) than for the incorrect guesses as seen in the table 1.

4. Conclusion

When we look on decision tree, the objective of this study is to develop an effective model for text classification based on word frequency. To this end, we have implemented a Decision Tree classifier and conducted experiments with different tree depths. Based on the results of the experiments as seen on the Figure 1 we can see that the decision tree algorithm is quite accurate

in the term of text classification. Although the results of the classification vary depending on the test article and the depth value we can see that the algorithm can reach satisfying accuracy of over 50% when given the right depth value for the decision tree. Another thing is that the lowest accuracy for this algorithm does not go below 30% for unadjusted depth values, meaning that even if the algorithm is not fitted for certain types of articles it retains some accuracy for assuming the right class to the right article. Since the results vary we can also speculate that for different types of training sets and different values of decision tree depth the algorithm will reach higher accuracy. Nevertheless those adjustments may vary since for the testowa2 dataset the algorithm shows decreasing, linear trend for accuracy with increasing value of tree depth, while for testowa1 the accuracy values show increasing, linear trend for increasing tree depth value as seen on figure 1. This might prove finding the best settings for the highest accuracy difficult.

The second algorithm that we used which is the Naive Bayes classifier shows very high accuracy of 100% for classifying the documents for every checked combination of the treningowa, testowa1 and testowa2 datasets as shown on Table 2. For each test text of possible classes: History, sport and cooking the algorithm correctly guessed the class of the test article. The high accuracy and the correctness of the calculations of the algorithm are proven by the results shown in Table 1. In mentioned table the values of the log probabilities for the correctly assumed classes are significantly (almost two times) higher than for the incorrect guesses. This difference in values shows that the algorithm correctly calculates the probability for two different texts that vary in vocabulary range, word count and topic. The possibility that the algorithm chose the correct class due to the random error in calculations or due to the very small difference in probability is out of the question.

By reevaluating our results we can see that the best algorithm for the text classification is the Naive Bayes classifier, this algorithm has shown the highest accuracy of our algorithms and the results of said classification using this algorithm did during the experiment. Nevertheless the Decision tree algorithm also showed potential for high accuracy of classification of texts due to the trends of the accuracy depending on the setting. Nonetheless the accuracy results for this algorithm were significantly lower than for the Naive Bayes classifier. The highest accuracy reached by the Decisive tree algorithm was 52% which is almost half of the accuracy of the Naive Bayes classifier. Decisive tree algorithm however shows possibility for it to be improved. The experiment could be improved by using bigger datasets with bigger number of articles. Moreover the datasets could include more types of classes such as science, medicine, entertainment or in case of bigger texts such as extracts from the book the book genres such as fantasy, sci-fi etc. This could show us more accurately the behaviour of our algorithms when given more input data. What is more, the bigger amount of information could show any possible anomalies and outliers as well as the trends in the results, allowing us to improve our algorithms accordingly. As for the Decisive tree algorithm we could check the results of classification for higher value of max tree depth. This could show us if our assumptions of linear trend of the classification accuracy were correct or not, giving us information on how to improve said algorithm.

References

The datasets utilized in this project were derived from a collection of books and various online sources. For reference and further exploration, the links to these books and articles are provided in the bibliography section below. These resources form the foundation of our text classification study, offering a diverse range of textual data essential for our analysis.

- **History Of The Decline And Fall Of The Roman Empire**, Edward Gibbon, Esq. With notes by the Rev. H. H. Milman. Complete Contents, Chapter XXI.
- **The European Experience: A Multi-Perspective History of Modern Europe, 1500–2000**.
- The Most Amazing Chocolate Chip Cookies.
- Liverpool vs. Tottenham Premier League Match Report.
- Homemade Pizza Recipe.
- NASCAR Cup Series at Kansas Speedway.
- **The Project Gutenberg eBook of Cleopatra**, Chapter 1.
- **What the German Conscript Thinks** by Arnold Bennett. Copyright, 1914, by The New York Times Company.
- **The History of Christianity** by John S. C. Abbott, Chapter XIII.
- **How to Make a Blueberry Pie**.
- **Rice Cooker Asian Chicken Rice, Grilled Duck Breast With Morels**.
- Kylian Mbappé on Final Season at PSG.
- Arsenal Boss Mikel Arteta on the 'Painful' Chase of Manchester City.
- How 'Introverted' Iga Świątek Became a Four-Time Grand Slam Champion.

[1] B. Liu, Y. Dai, X. Li, W. S. Lee, and P. S. Yu, "Text Classification by Labeling Words," *Proceedings of the 19th National Conference on Artificial Intelligence (AAAI-04)*, San Jose, California, USA, July 25-29, 2004.

[2] S. Kotsiantis, D. Kanellopoulos, and P. Pintelas, "Handling imbalanced datasets: A review," *GESTS International Transactions on Computer Science and Engineering*, vol. 30, no. 1, pp. 25-36, 2006.

[3] T. Joachims, "Text Categorization with Support Vector Machines: Learning with Many Relevant Features," *Proceedings of the 10th European Conference on Machine Learning (ECML-98)*, Chemnitz, Germany, April 21-23, 1998.

[4] K. Nigam, A. K. McCallum, S. Thrun, and T. Mitchell, "Text Classification from Labeled and Unlabeled Documents using EM," *Machine Learning*, vol. 39, no. 2/3, pp. 103-134, 2000.

[5] J. R. Quinlan, "Induction of Decision Trees," *Machine Learning*, vol. 1, pp. 81-106, 1986.

[6] D. Mladenic and M. Grobelnik, "Feature Selection for Unbalanced Class Distribution and Naive Bayes," *Proceedings of the 16th International Conference on Machine Learning (ICML-99)*, Bled, Slovenia, June 27-30, 1999.