Text Mining Assignment 1: Text Categorization

Kacper Kadziolka - s4115945
Giulia Rivetti - s4026543

1. Introduction

In this assignment we have implemented text categorization, the task of assigning a label or category to an entire text or document (Jurafsky & Martin, 2024). In particular, we have used the *scikit-learn* library to perform multi-class classification in order to classify documents by topics. For this purpose, we have used the *The 20 newsgroup text dataset*, which comprises around 18,000 newsgroups posts on 20 topics split in two subsets: one for training and the other one for testing (cla).
Moreover, we have experimented the effects of using different types of feature weights (Section 2.2) and classifiers (Section 2.1), as well as employing suitable evaluation metrics to evaluate the classifiers.

2. Methodology

After loading and vectorizing *The 20 newsgroup text dataset*, we have performed feature extraction in order to extract the relevant features of the documents. During this process, we have compared not only different types of classifiers, but also three different kinds of features.

2.1. Classifiers

The classifiers that we considered for this assignment are the following:

1. **Multinomial Naive Bayes**: it is a probabilistic classifier used to calculate the probability distribution of text data. It is computationally efficient and can handle large datasets with many features which makes it a practical choice for text classification tasks, where features are often count-based (MNB).

2. **Logistic Regression**: it is a supervised machine learning algorithm used for classification tasks, where the goal is to predict the probability that an instance belongs to a given class or not. This is done by using the sigmoid function, that takes input as independent variables and produces a probability value between 0 and 1 (log).

3. **Linear Support Vector Classifier (LinearSVC)**: it is a classifier that supports linear support vector classification. In particular, it finds the hyperplane that best separates classes by maximizing margin between them.

2.2. Feature Extraction methods

In order to extract the relevant characteristics (or features) of the text documents, we have employed three different feature methods:

1. **counts (bag-of-words)**: this is a very common feature extraction procedure for sentences and documents, where we. consider each word count as a feature, ignoring their ordering. (Goldberg, 2017)

2. **tf (Term Frequency)**: it measures the frequency of a term within a document and it is expressed as the ratio of the number of times a term occurs in a document to the total number of terms in that document; it emphasizes words that are frequent within a document (tfi);

3. **tf-idf (Term Frequency - Inverse document Frequency)**: the IDF measures the rarity of a term across a collection of documents and penalizes words that are common across all documents. It is expressed as the logarithm of the ratio of the total number of documents to the number of documents containing the term. Then, the TF-IDF score for a term in a document is obtained by multiplying its TF and IDF scores (tfi).

2.3. Parameters

Besides experimenting with different classifiers and feature extraction methods, we have also tried different values for several parameters of the *CountVectorizer* function, in order to find the configuration leading to the best performance. The parameters that we have considered are the following:

1. *lowercase*, which specifies if all the characters has to be converted to lowercase before tokenizing;

2. *stop_words*, which specifies the stop-word lists to use;

3. *analyzer*, that indicates whether the feature should be made of word n-gram or character n-grams;

Classifier	Feature	Precision	Recall	F1-score
MultinomialNB	Count	0.761668	0.772836	0.75113
LogisticRegression	Count	0.791675	0.789697	0.788870
LinearSVC	Count	0.787315	0.786378	0.785076
MultinomialNB	TF	0.761668	0.771836	0.751113
LogisticRegression	TF	0.791675	0.789697	0.788879
LinearSVC	TF	0.787315	0.786378	0.785076
MultinomialNB	TF-IDF	0.821878	0.773898	0.768446
LinearRegression	TF-IDF	0.831174	0.827801	0.826136
LinearSVC	TF-IDF	0.854365	0.853160	0.851876

Table 1: Result table for the classifiers and features.

4. *ngram_range*, specifying the lower and upper boundary of the range of n-values for different word n-grams or char n-grams;

5. *max_features*, which indicates the number of top features to consider (ordered by term frequency across the corpus).

3. Experiments

As previously mentioned, we have used three different feature methods on three distinct classifiers and compared the results, in order to use the best classifier-feature combination for the last task, namely finding the optimal values for the aforementioned parameters.

3.1. Classifiers and feature extraction methods

From the results in Table 1, it is evident that the *LinearSVC* classifier, when combined with *TF-IDF*, achieves superior performance across all evaluated metrics, with a precision of 0.854, a recall of 0.853, and an F1-score of 0.852. We believe that this notably strong performance is due to the ability of *TF-IDF* to effectively down-weight common terms and emphasize more informative, discriminative words. This type of feature extraction is known for its high-dimensionality and space requirements, as it creates a new dimension for each unique word in the text corpus. Meanwhile, *LinearSVC* is well-regarded for its capability to handle high-dimensional data, making it particularly well-suited for text categorization tasks where the feature space (i.e., terms) is large (Lin).

Given its strong performance, we will proceed with the combination of *TF-IDF* and *LinearSVC* in the next phase of our experiments.

3.2. CountVectorizer function fine-tuning

In this phase of our experiment, we tested the best classifier and feature extraction method from the previous experiment, using the *CountVectorizer* instead. The purpose was to explore how different hyperparameters affect the performance of our model.

We conducted experiments with the following parameters:

1. *lowercase*: whether to convert all text to lowercase (True or False).

2. *stop_words*: None for no removal or 'english' to remove English stop words

3. *analyzer and n-gram range:*: we considered unigrams, bigrams, or a combination of both.

4. *max_features*: we experimented with None (all features), 5,000, and 10,000 features.

We used *GridSearchCV* to test a variety of configurations of these parameters with *LinearSVC* and *CountVectorizer*. We performed a 5-fold cross-validation and used weighted *F1-score* as the evaluation metric. The pipeline integrated the vectorization and classification steps.

After training and testing across 36 different parameter combinations, the best-performing configuration used lowercasing set to True, no stop-word removal, and a $(1, 2)$ n-gram range with no feature limit (all available features were used). The classifier using this configuration achieved the following metrics on the test set: Precision: 0.859, Recall: 0.857, F1-Score: 0.856.

4. Conclusion

In this assignment, we explored various text categorization techniques using different classifiers and feature extraction methods on the 20 newsgroups dataset. Initially, we tested three classifiers: *Multinomial Naive Bayes*, *Logistic Regression*, and *LinearSVC*, in combination with four feature extraction techniques: *CountVectorizer*, *Term Frequency (TF)*, *TF-IDF*, and naive *bag of words (BoW)*.

The results from the first phase of experiments demonstrated that *LinearSVC* combined with *TF-IDF* outperformed other combinations, and achieved superior performance across all three metrics (Table 1). In the second phase of experiments, we explored different configurations of *CountVectorizer* by tuning parameters through *GridSearchCV*, we tested 36 different parameter combinations and found the best-performing configuration (Section 3.2).

In conclusion, our experiments showed that *LinearSVC* combined with *TF-IDF* and *CountVectorizer* with optimized parameters are effective approaches for text categorization.

References

Support Vector Machines. https://scikit-learn.org/stable/modules/svm.html. scikit-learn.

Multinomial Naive Bayes . https://www.geeksforgeeks.org/multinomial-naive-bayes/. GeeksforGeeks - 28 Jan, 2024.

Classification of text documents using sparse features. https://scikit-learn.org/stable/auto_examples/text/plot_document_classification_20newsgroups.html. scikit-learn.

Logistic Regressionn in Machine Learning . https://www.geeksforgeeks.org/understanding-logistic-regression/#what-is-logistic-regression. GeeksforGeeks - 20 Jun, 2024.

TF-IDF in NLP (Term Frequency Inverse Document Frequency) . https://medium.com/@abhishekjainindore24/tf-idf-in-nlp-term-frequency-inverse-document-frequency-e05b65932f1d. Medium - 04 Feb, 2024.

Goldberg, Y. *Neural Network Methods for Natural Language Processing*. 2017. Page 69.

Jurafsky, D. and Martin, J. H. *Speech and Language Processing*. 3th edition, 2024.