Machine learning challenge

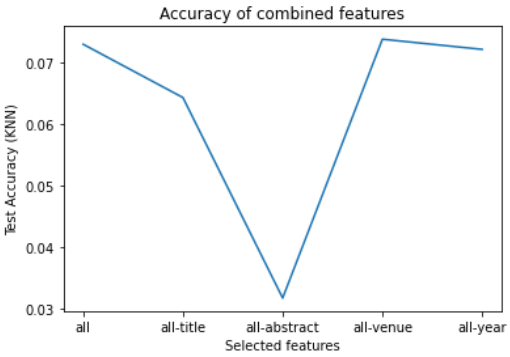
Group 46

Iris van Velzen 2086760, Athina Filiou 2082862, Sihui Wang 2088587, Renee Pex 2064669

In this report, we describe the process of building our machine learning model to predict the author of a scientific paper based on its title, abstract, year, and venue. The goal of this project is to develop a model that can accurately predict the author of a paper given its features.

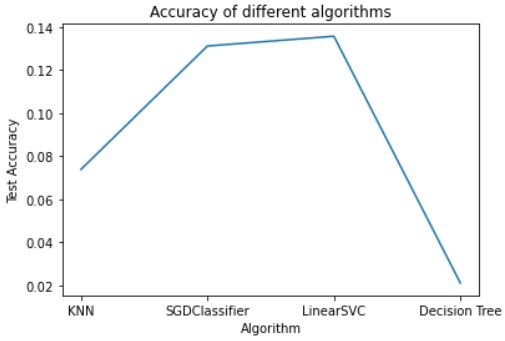
Feature engineering

The first step in building our model was to preprocess and clean the dataset. We checked for any missing values but did not find any. To preprocess the text data, we used regular expressions to keep only letters, converted the text to lowercase, and split it into words. We then removed stopwords and performed stemming or lemmatization to reduce the words to their base forms. Next, we performed feature engineering to create new features from the text data. We used the TfidfVectorizer to vectorize the title and abstract features, and we converted the venue feature into a categorical variable. We then tested the performance of the model with different combinations of the features, and found that using the title, abstract, and year resulted in the best performance.



Learning algorithms

We have tried multiple algorithms for the learning part of our analysis, including Decision Tree, K-nearest neighbors (KNN), Linear SVC, and SGDClassifier. The ones that gave us the best and yet similar results regarding accuracy were LinearSVC and SGDClassifier. However, the SGD classifier was chosen as the final learning algorithm because it has several advantages over linear SVC, such as faster training time, the ability to use mini-batch learning, and the ability to use regularization to prevent overfitting. Furthermore, it can be trained on large datasets effectively and it is an effective method for text classification.



Hyperparameter tuning

After we decided on our final model, we started experimenting with the different hyperparameters to see which combination had the best accuracy on the validation data. We tried several different combinations of hyperparameters for the SGD classifier, including different values for the learning rate, regularization strength, loss functions and validation fraction. After experimenting with different hyperparameters, we changed the following hyperparameters to find the optimal accuracy.

- We used the epsilon_insensitive loss function, which is suitable for regression tasks.

- We set the max_iter to 800, which allowed the classifier to run for enough iterations to find a good solution.
- We set the n_iter_no_change to 15, which allowed the classifier to continue running for a few more iterations after the loss function stopped improving in case the optimum was reached in the subsequent iterations.

Performance of solution

To evaluate the performance of the solution we split the dataset into a training and validation set. The model performed extremely well on the training set, achieving an accuracy score of 0.9998. However, when tested on the validation set, the model's accuracy was significantly lower, at 0.1303. This indicates that the model may be overfitting to the training data and is not able to generalize well to unseen data. Despite this, the model's performance on the test dataset was relatively good when compared to the scores of other groups on Codalab. In fact, the model's performance on Codalab was 0.2089, which is higher than the baseline score set for the assignment. These results suggest that while the model may be overfitting, it is still able to perform well on unseen data.

Codalab

The codalab submissions for our group project have been submitted under the username “athina_f”.

Specification of work

The work on the challenge was split in the following way: Sihui and Athina worked on preprocessing the text data and converting it into a sparse matrix. Iris and Renee worked on selecting the features to be used in the model.

After all group members researched our first algorithm which is Multilayer Perceptron and tried implementing it without promising results, we decided it was time to try different models. Renee and Sihui tried the KNN, Decision Tree, and SGDClassifier algorithms. Athina worked on LinearSVC and Iris worked on MultinomialNB. Next, we worked together to try different hyperparameters for the SGDClassifier model to find the combination that would give the best accuracy results. Finally, we evaluated the performance of the final model and reported the accuracy results, with Sihui working on outputting the prediction as a JSON file.

Overall, all group members actively participated in group discussions over zoom and on campus and contributed to the various tasks involved in the project.

References

(n.d.). Retrieved from github.com: https://github.com/lisanka93/text_analysis_python_101/blob/master/Dummy%20movie%20dataset.ipynb

(n.d.). Retrieved from scikit-learn.org: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDClassifier.html

(2019, January). Retrieved from austingwalters.com: https://austingwalters.com/classify-sentences-via-a-multilayer-perceptron-mlp/

Bedi, G. (2018, November). Retrieved from medium.com: https://medium.com/@bedigunjit/simple-guide-to-text-classification-nlp-using-svm-and-naive-bayes-with-python-421db3a72d34

Shaikh, J. (2017, July). Retrieved from towardsdatascience.com: https://towardsdatascience.com/machine-learning-nlp-text-classification-using-scikit-learn-python-and-nltk-c52b92a7c73a

Tekman, M. (n.d.). Retrieved from Kaggle.com: https://www.kaggle.com/code/mehmetlaudatekman/text-classification-svm-explained