Klaudia Biczysko & Justyna Sikora

Course: Information Retrieval

Implementation of a document classifier based on Latent Semantic Analysis

1. Introduction

We decided to implement a document classifier in Python, which is based on Latent Semantic Analysis. As our source data, we used Reuter's dataset. Beside Python as our programming language, we used tools like: sklearn library, pandas library, BeautifulSoup, re library, and ipynb notebook.

The code with respective comments is implemented in an ipynb notebook.

2. Preliminaries

- Reuters dataset

  It is a multilabel and multi-class dataset, which consists of documents with news articles, created for document classification. The dataset consists of 90 classes, 7769 training documents and 3019 testing documents. Reuters articles may have more than one topic assigned to one article. However we decided to limit our classifier to multiclass classification. Hence - if an article has more than one topic - we are taking only the first one into account.

- Latent Semantic Analysis

  Latent Semantic Analysis (LSA) is a method that aims to create a document's vector representation. Using LSA and obtaining such representations, enables us to compare two (or more documents). This is done by computing the distance between given documents vector representations. We can further use these calculations in e.g. document classification. LSA employs Singular Value Decomposition (SVD), which helps to reduce dimensionality. In turn, dimensionality reduction results in creating similar vectors for terms used in similar context and thus provides a way to handle synonymy. We assume

that documents and sentences have underlying ('latent') features that are connected to their meaning, apart from the particular words that appear.

3. Preprocessing

To preprocess data, we decided to use BeautifulSoup. By using this library, we were able to extract HTML tags like 'reuters', 'topics', and 'text'. After extracting the articles, which could be found under the 'reuters' tag, we looped through them and extracted the topics and text of every article. As mentioned before, we appended only one topic per article, if an article did not have any topic (length=0), we appended an empty string. In the end, the topics' list has been appended to raw_y, which is a dataset for labels. When it comes to the text of articles, we lowered the data and - by using regular expressions – removed special characters (punctuation, etc.) and replaced numbers to nn. If an article did not have any text, we appended an empty string to the main raw_X list.

4. Topic modeling

Although the topic modelling is not directly a part of building the classification pipeline, we decided to include it in our project to demonstrate the capabilities of LSA. In the section "Singular Value Decomposition" we present how the LSA modelling works by providing the key words for each of the 130 latent topics found by the LSA.

5. Model

To build the pipeline for the articles' classification we decided to implement tf-idf vectorizer with the k-Nearest Neighbor model. Additionally, we employed latent semantic analysis (LSA) in the form of the truncated SVD transformer, which is used on the tf-idf matrix in order to reduce its dimensionality. The transformed dimensions represent latent topics found by the TruncatedSVD. Dimensionality reduction is commonly used with kNN models as the higher dimensionality may decrease the predictive power of the model. Hence, using SVD - and constraining the number of dimensions, should improve the model's performance.

In order to find the best hyperparameters we conducted the grid search on different values for the number of neighbours as well as the number of components (dimensions) for the TruncatedSVD. The grid search was performed on the train set in order to avoid overfitting.

The best result - 80.84% accuracy, obtained the model trained on k=3 NN and with 150 LSA dimensions. Therefore, we have employed these values into our final model.

6. Evaluation

To evaluate the model's performance we used the cross-validation metric as well as we calculated micro and macro precision, recall and f1 scores. The results are included in the .ipynb file. The cross-validation scores show that the model performs better when compared to the simple baseline model (tf-idf vectors feed into the kNN model).

We have also built a simple baseline model to measure our model's performance. Comparing the accuracy obtained on both models, we can see that using the LSA model slightly improved the results of the classifier.