Homework 2
Support Vector Machines & Clustering

University of Bucharest, Faculty of Mathematics and Informatics
Sparktech Software

November 15, 2018

Soft Deadline: Tuesday, December 11, 2018, 11:59 PM. If you do not make this deadline, your submission will receive a 20 points deduction.

Hard Deadline: Wednesday, December 12, 2018, 2:00 PM. If you do not make this deadline, your submission will not be graded.

1 Introduction

In this homework you will analyze books to compare the writing style of various authors. You have two tasks: first, to identify the author of a given text, and second, to find similarities between books, and hence find out what authors have similar writing styles. You will submit your homework as a written report, either as a **Jupyter Notebook**, or as a **PDF file**. If you choose to submit a PDF file, you will need to also send us the Python code you wrote to produce the results and vizualisations. You will need to describe the steps you took towards reaching your conclusion.

1.1 Natural Language Processing Crash-Course

All machine learning algorithms we studied so far work with numeric data. In order to apply those algorithms to text data, we need to transform our data into numbers. We call this process **feature extraction**. There are many ways to do this, so let's start with the easiest technique we can think of.

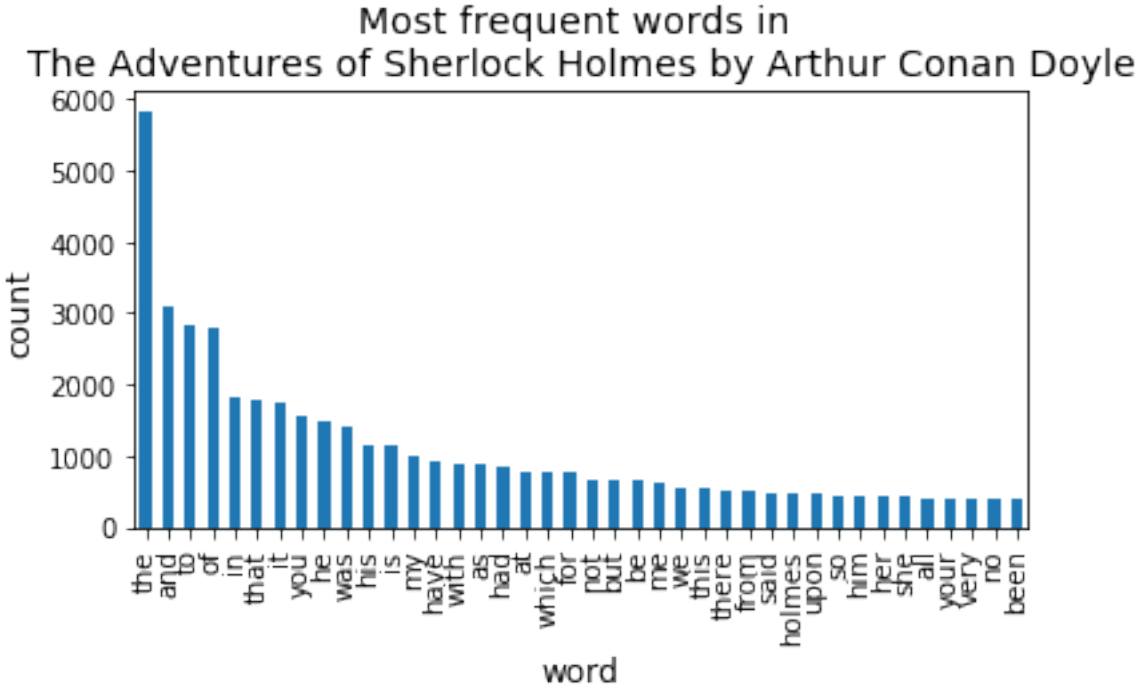
1.1.1 Text Embedding

We want for each document, a numerical representation of it, or a **feature vector**.

One way to turn a text document into a feature vector is to use a frequency count of each word in the document. We build a large dictionary of words, and for each document we return a vector with as many features as there are words, and for each word, we return

the number of times that word appears in the document (this is technically called **term frequency**, or **tf** for short). Sklearn has a CountVectorizer [1] that does just that.

One problem with this representation is the high frequency of common words like "the" or "to" or "and". Those words appear in almost all documents, so they don't offer much information. Here's an example of word counts for The Adventures of Sherlock Holmes by Arthur Conan Doyle.



These most frequent words are not representative for this book. Given this list of words, even if the proportion of them is different for each author, it may be difficult to detect the author. We have a few ways to tackle this problem. We could simply remove the most frequent words. CountVectorizer offers a parameter `max_df`, so we can set the vectorizer to remove words with a **document frequency** above a threshold. Notice that now we will count the frequency not within one document, but the frequency in all documents.

Better yet, we can combine the two metrics, namely **term frequency** and **document frequency** in a single metric. A word that appears in too many document (high document frequency) doesn't give us much information. We would like to find those words (which we consider features) that are representative for our class. We would like to find words that have are used most often by specific authors, and less so by other authors. So the more frequent a term is (high term frequency), the more important it is, and also the more specific to a document (low document frequency), the more important it is. We can now multiply those two metrics. This is called tf-idf vectorization [2].

[1] `https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html`

[2] `https://en.wikipedia.org/wiki/Tf-idf`

This can be achieved with TfidfTransformer [3], or TfidfVectorizer[4], in sklearn. As such, using tf-idf vectorization, we have converted our problem of classifying text into a problem of classifying numbers, which we are already familiar with; all algorithms you've applied so far work just as well.

Those methods are called **Bag of Words** (BoW) models. Notice that the order of the words is lost. All that is kept is the frequency of words used. We can improve the model by considering **n-grams**. Instead of creating a dictionary of words, we consider all groups of n successive words. Check the ngram_range parameter in CountVectorizer. Keep in mind that the feature vector becomes very large and sparse with this approach. You might want to reduce the dimensionality of the dataset.

1.1.2 Bonus: Better representations

A crucial part of the performance of machine learning algorithms in the context of natural language processing is constructing the optimal representations (embeddings, features) for the corpus to best address the problem. We have seen a basic method for vectorization, that can work for some cases, but has limitations when it comes to the semantic meaning of the text. As such, better embeddings have been developed, namely Word2Vec [2], [3], Doc2Vec [1] or GloVe [4].

Each of them has implementations in *gensim*[5]. You are not required to use this, but for bonus points it would be nice to experiment with them.

2 Task 1: Text Classification

For this task you will need to use an SVM to predict the Author of a text fragment. The dataset you will be using is composed of samples of text from 20 different authors. Each sample has 50.000 characters and there are 20 samples for each author. Firstly, you will need to transform each text sample into a feature vector. To do that you will need to use at least some of the techniques described above. On the transformed dataset you will have to fit a Support Vector Classifier. You will need to use the sklearn implementation of the algorithm. The goal is to find the hyperparameters (kernel, C, gamma) that makes your SVC best at predicting the Author of a text fragment. After you are done tuning your model apply PCA to reduce the dimensionality of your features and retrain the algorithm. Compare the performance of the algorithm with and without PCA.

3 Task 2: Text Clustering

For this task you will need to compare different clustering methods. Your will have to compare 3 clustering methods: DBCAN, KMeans and Hierarchical(Agglomerative). You

[3] `https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfTransformer.html`

[4] `https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html`

[5] `https://radimrehurek.com/gensim/`

will need to use the sklearn implementations of the algorithms. For DBSCAN you will have to explore the eps and and min_samples params. For KMeans: n_clusters and init. For Agglomerative: n_clusters. Use different metrics such as: silhouette score, homogeneity score, completness score. You can use PCA for visualizations.

Try to extract as much information as you can from the dataset using each algorithm. Try experimenting with different distance metrics, different word embeddings. Compare and contrast what you find. Does the clustering gives you back the books clustered by author? Are the books grouped differently? Can you find what authors have similar writing styles? Or are the books grouped by genre? Keep in mind that with unsupervised learning, the problem is not well defined. It is an investigation of a dataset. You need to find some structure in the dataset and explain *what* that structure represents.

4 Grading Policy

This homework doesn't have any automated testing, or any other *strict* criteria of evaluation. As such, you will be graded based on the thoroughness of your analysis, and the presentation of results according to the following criteria.

1. (**5p**) The submitted code is sufficient for us to reproduce the results described in your report.
2. (**5p**) The project uses NumPy arrays and Pandas DataFrames where appropriate, rather than Python lists and dictionaries. Where possible, vectorized operations and built-in functions are used instead of loops.
3. (**5p**) The report documents any changes that were made to clean the data
4. (**5p**) You should state and describe the problem you are trying to solve. Describe the data (type of data, summary statistics), the desired output, and the proposed model.
5. (**10p**) You should describe any features transformation you make.
6. (**10p**) Visualizations made in the report depict the data in an appropriate manner, which allows plots to be readily interpreted. Include both exploratory plots and plots to visualize the behavior of the models.
7. (**10p**) You should experiment with multiple values for hyperparameters. Describe the strategy used to find the best values.
8. (**15p**) Select the best hyperparameters for the classification problem. Justify your decision.
9. (**15p**) Make a thorough analysis of the clustering methods.
10. (**20p**) Your decisions must be supported by arguments (i.e. table with hyper-parameter values, metrics used, other explanations etc.)
11. (**15p**) Bonus points. For a thorough analysis that goes beyond the scope of the course.

12. (**15p**) More Bonus points. For using clustering and dimensionality reduction techniques on your first homework dataset.

Total: 130p

5 Other Considerations

- Each student will receive **his own personal dataset**. This means that the parameters you find will likely be unique and optimal for your own dataset.
- We also have a separate **test set** on which we will evaluate your models.
- Make sure all **plots are properly labeled** (each axis should have a label), and the title is concise, but clear. Use legends when necessary. The meaning of each plot should be clear without the need to inspect the code that produced it.
- **Collaboration** is allowed and encouraged regarding techniques used, model details (how it works, how to tune it), descriptive statistics, etc. but keep in mind that the report and the code submitted has to be your own work.
- **All code must be written in Python v3.6** (or greater).

References

[1] Quoc V. Le and Tomas Mikolov. Distributed representations of sentences and documents. *CoRR*, abs/1405.4053, 2014.

[2] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.

[3] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc., 2013.

[4] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.