Feature-based Authorship Attribution with Python

Wencke Liermann

Universität Potsdam, Potsdam, Germany
Bachelor Computerlinguistik - 4. Semester
SoSe20
wliermann@uni-potsdam.de

Abstract. This paper serves as an introduction to the python program it is provided with and is at the same time a reflection of the processes involved in its creation. As a final project in the context of the advanced course *Programmierung II* and in correspondence with our task to implement a text classification system a program was written that performs authorship attribution for a small subset of the data included in the Gutenberg data set (Lahiri, 2014).

1 Introduction

1.1 Introduction to the task

Given a set of candidate authors, text samples of known authorship covering all the candidate authors (= *training set*) and text samples of (supposedly) unknown authorship (= *test set*), authorship attribution describes the task of deducing an author profile from the former to be used in attributing a candidate author to each of the later (Stamatatos, 2009). Applications of authorship attribution include but are not limited to plagiarism detection (e.g. college essays), determining the writer of inappropriate anonymous comments (e.g. threatening or harassing e-mails), as well as resolving historical questions of unclear or disputed authorship (Sanderson and Guenter, 2006).

1.2 Introduction to the data

The Gutenberg data set (Lahiri, 2014) includes data from all in all 142 different authors. But I decided on using only the data belonging to the ten authors that had the largest amount of data available under their name. This decision reflects my intention to later have a system that can classify data as belonging to one out of ten different classes, one class for each author. The filtering and partition into the three sets were done automatically by a python script that made use of the function getsize() from the os.path module as an approximation of the number of characters in a file. As the files are written in English and encoded in UTF-8 I deem this a very exact approximation. Byte size is obviously the measure of text size with the lowest computational costs compared to number of words or sentences. The combination of files from each author with a summed up byte number that is nearest to the 10% or 20% respectively was extracted and included in the validation (or test) set. As a consequence the data of each author is distributed equally between the three parts.

Below you find the chosen authors together with the number of their works in the training set:

- Anthony Trollope (50)
- Charles Dickens (40)
- Charlotte Mary Yonge (42)
- George Alfred Henty (63)
- Henry Rider Haggard (37)
- James Fenimore Cooper (26)
- R M Ballantyne (62)
- Robert Louis Stevenson (52)
- Sir Walter Scott (25)
- William Dean Howells (59)

Finally, the files in all parts were preprocessed. They were word and sentence tokenized, separating single tokens with whitespaces and placing each sentence on its own line. The files in one set are not concatenated explicitly but for training all the files in a folder are regarded as one big file and used to extract cumulative representation of that author's style.

2 Feature Selection

Using the most common words determined over the texts of all the candidate authors has been proven to be quite reliable for authorship attribution, maybe due to the unlikeliness of these words being subject to the conscious control of the author (Burrows, 1987; Argamon and Levitan, 2005) as well as the fact that their occurrence is not linked to specific topics (Mosteller and Wallace, 1964). Those words include but are not limited to what we understand under the term stopwords. Of course I had to decide on a specific amount of these words to be used as features. Since Argamon and Levitan (2005) have identified a potential risk of over fitting when using a too large set, I decided on only using the 90 most common words. I say words but what is actually meant are tokens in their lemmatized form even including punctuation marks, but not those that are used to divide a text into sentences or sentences into units, namely '.', ',', ';', '!', '?'. Those a collected separately and normalized among themselves.

Further easily available features for any natural language text that have been proven to be quite useful to quantify the writing style are sentence length counts and word length counts (Mendenhall, 1887). Additionally to the relative frequency of lengths I also calculated the average and standard deviation and included them as a feature.

Vocabulary richness is a measure for the text complexity, higher scores being associated with more advanced texts. There have been different measures of vocabulary richness proposed, while most however fail to correctly reflect on the role of text length, McCarthy (2005) claim to have found no correlation of text length and their MTLD measure as soon as the texts reach a length over 200 words. Which is why I implemented and used this measure as an additional feature.

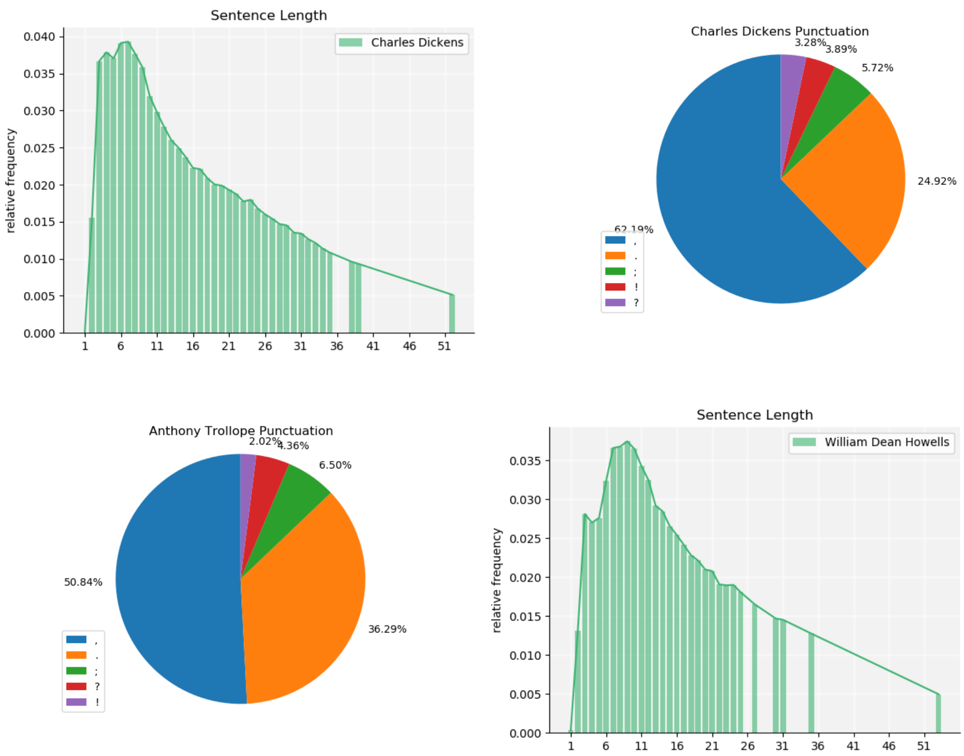

Fig. 1: Visualization of some statistics

Another feature I really wanted to use are rewrite rules extracted from constituency syntax trees. Baayen et al. (1996) argue that the lowest frequency ranges of rewrite rules carry discriminatory potential. While high frequency rewrite rules often are typical for the population as a whole or may be the result of literary training, the least frequent rules reflect the creativity and capability to experiment with the English language of the author. Surprisingly, I had to notice that pretrained constituency parsers are not as readily available as dependency parsers. I experimented with the stanford parser and its pretrained model for English. I found the download and installation to be quite cumbersome and I often encountered unpredictable connection errors when using the parser I didn't know how to catch properly. Another downside was the time it took for parsing, it was unimaginable to parse whole books so the first measure I took was limiting the amount of rewrite rules extracted per file to 200. A raw version of a group of functions to extract this feature can be found in a second branch on GitHub, keep in mind that this group can't be executed as is. I did once run my system including this feature but the positive effects, even though they were there, were not considerable enough to make up the disadvantages mentioned. That's why I decided not to include this feature in the final project, despite the time it took me to implement everything.

As an alternative, in order to at least partly encode syntactic choices of the candidate authors I used POS tag trigrams (Argamon et al., 1998). Compared to word or character based trigrams this feature is not prone to encode content instead of style.

3 Evaluation

The system performs better on some authors than on others. While perfect accuracies of 100% were achieved for two of the authors, one author stands out negatively with an accuracy of below 10%. Notable, out of the two former ones Henry Rider Haggard was only predicted ten times, always correctly. The author that was predicted incorrectly most often was James Fenimore Cooper. 19 out of all 41 incorrect predictions were him. A more detailed inside into the test files and their correct vs. predicted classes can be gained from the file *data/results.csv*. Overall the system achieved a rather satisfying accuracy of 68.70%. In comparison, a naive baseline classifier that always predicts the majority class would only achieve an accuracy of 16.03%.

Author	Test size in number of files	Accuracy in %
Anthony Trollope	13	76.92
Charles Dickens	10	50.00
Charlotte Mary Yonge	12	83.33
George Alfred Henty	18	88.89
Henry Rider Haggard	10	100.0
James Fenimore Cooper	7	71.42
R M Ballantyne	17	88.24
Robert Louis Stevenson	16	62.50
Sir Walter Scott	7	100.0
William Dean Howells	21	9.52
Total	**131**	**68.70**
Total (without William D. H.)	**110**	**80.00**

Table 1: Results for *data/gutenbergident.txt* with ten pretrained author classes

4 Reflection

As I have already once implemented a ngram-based language classifier for the *Programmierung I* course, the general structure of classifiers was not new to me and I already knew what major classes I would have to implement. Consequently, I could focus on the selection of appropriate features. After deciding on a task and data set the next thing on my agenda was a little research. I collected and studied papers on the topic authorship attribution and took notes of described features. Instead of putting a lot of work into experimenting with features that might not be a fitting choice for the task, I wanted to make sure that whatever I implement has been proven to be helpful. While implementing the feature extraction I encountered the first real obstacle; how to write nice neat python functions for small subsets of features without having to traverse a file more than once. As I did not find a satisfying solution I ended up extracting all features in a single function. Although I would describe this function as the ugly duckling of my project I chose this option over dividing my features according to an arbitrary criterion into two functions. My selected feature set is still quite limited and anyone setting out to improve or extend my system will certainly add more features. This is possible but should be definitely done in a separate function that can then be called together with the existing function responsible for feature extraction.

In contrast to this function my work on the project was indeed divided into two major parts with a break of three weeks in between. This had to be done due to the postponed exam period and the strict deadline. Something I would neither recommend nor do again, after the exams I had lost track of what I had done for what reason and felt the need to write some functions basically from scratch again because I was no longer satisfied with them. I would have really appreciated more time for the project so I would have had the resources to experiment a little more. For example when dividing the corpus into parts I faced the NP-complete problem of finding a subset with a certain sum and with more time I would have loved to experiment with existing pseudo polynomial time algorithms or dynamic programming approaches. Instead, I had to limit myself to solving an easier version of the problem with a not optimal but good solution.

Moreover, at times I felt overwhelmed by the freedom given to us in regard to what packages we were allowed to use. Importing functionalities instead of researching algorithms and contemplating programming paradigms sometimes felt as if one wasn't really doing much by oneself. —- review—-

Bibliography

Argamon, S., Koppel, M., and Avneri, G. (1998). Routing documents according to style. In *First International workshop on innovative information systems*, pages 85–92.

Argamon, S. and Levitan, S. (2005). Measuring the usefulness of function words for authorship attribution. In *Proceedings of the 2005 ACH/ALLC Conference*, pages 4–7.

Baayen, H., Van Halteren, H., and Tweedie, F. (1996). Outside the cave of shadows: Using syntactic annotation to enhance authorship attribution. *Literary and Linguistic Computing*, 11(3):121–132.

Burrows, J. F. (1987). *Computation into criticism: A study of Jane Austen's novels and an experiment in method*. Clarendon Pr.

Lahiri, S. (2014). Complexity of Word Collocation Networks: A Preliminary Structural Analysis. In *Proceedings of the Student Research Workshop at the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 96–105, Gothenburg, Sweden. Association for Computational Linguistics.

McCarthy, P. M. (2005). *An assessment of the range and usefulness of lexical diversity measures and the potential of the measure of textual, lexical diversity (MTLD)*. PhD thesis, The University of Memphis.

Mendenhall, T. (1887). *Characteristic Curves of Composition*. Moore & Langen, printers and binders.

Mosteller, F. and Wallace, D. (1964). Inference and disputed authorship: The federalist.(1964).

Sanderson, C. and Guenter, S. (2006). Short text authorship attribution via sequence kernels, markov chains and author unmasking: An investigation. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 482–491.

Stamatatos, E. (2009). A survey of modern authorship attribution methods. *Journal of the American Society for information Science and Technology*, 60(3):538–556.