Author Prediction for Poetry

Katrin Schmidt
3197512
@ims.uni-stuttgart.de

Carlotta Quensel
3546286
@ims.uni-stuttgart.de

Abstract

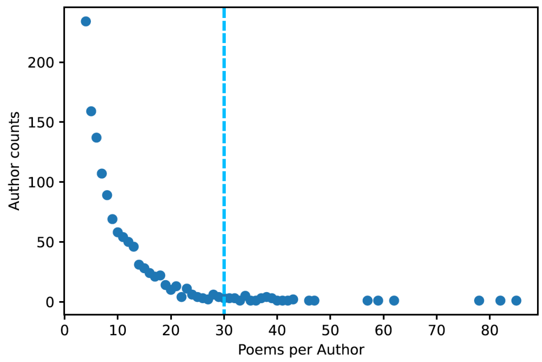
Same structure as the whole paper, but in short

1 Introduction

The task of author classification refers to the task of predicting the author of a given text by analyzing the text in terms of some features Argamon et al. (2009). These features can then be compared with those learned from some training data. A common approach to author classification "[...] consists of a combination of text analysis for extracting document features that are predictive of the author, and text categorization using Machine Learning (ML) techniques" Luyckx and Daelemans (2011). For this paper we focus on the prediction of poems in order to examine which features are particularly useful for the author classification of poems. This will give information about linguistical aspects on the one hand, that is poem specific features, and computational aspects on the other hand, that is the the question of efficiency and performance of the implementation of these features. Thus the goal is to answer the question if there exist features that are inherent to poetry and then to analyse and judge them according to their performance.
Besides from that we will take into account that style features might depend on medium (e.g. Limerick) more than on the author.

1.1 Corpus Creation

As a training data we used the collection of the Poetry Foundation which is pulled from kaggle.com as a premade csv-database. The dataset consists of 15 567 poems, written by altogether 3 309 authors. The following graphic shows the distribution of poems per author.



Since the data includes many authors who wrote between 1 and 5 poems, we used only the 30 most prolific authors to get enough data points per class for the method. Finally we ended up with 1 569 poems which is barely 10 % from the original dataset. In order to train our model with the data, we sorted the poems by author, normalized some remaining unicode strings (e.g. "ax0", which represents whitespaces) and tokenized them with the NLTK WordPunctTokenizer. Then we split the data into train and test set and converted the poems into bag-of-word vectors using the vocabulary in the train set.

2 Method

Generally speaking, a maximum entropy classifier is used for generating a probability distribution based on some training data Nigam et al. (1999). Before the classifier begins to train, the probabilities should be equally distributed, since there is no bias towards any label - therefore, the entropy is maximal in the beginning, in other words, the weights associated with the features are unknown Nigam et al. (1999). The formula for the maximum entropy classifier is given by

$$p_\lambda(y|\boldsymbol{x}) = \frac{\exp \sum_i \lambda_i f_i(y, \boldsymbol{x})}{\sum_{y'} \exp \sum_i \lambda_i f_i(y', \boldsymbol{x})}$$

where $f_i(y, \boldsymbol{x})$ is a feature and λ_i the corresponding weight Nigam et al. (1999). Furthermore, the maximum entropy classifier presupposes a dependence relation between the features Osborne (2002). This means that the classifier is not only able to differentiate between features that are relevant and features that are irrelevant for the classification task, but also to include this information in its classification process Osborne (2002). We decided to choose this classifier since we assume that features which match a certain author are relevant whereas features that don't match the author are irrelevant.

For our classification task a feature f_i contains a data property paired with a label, where $\boldsymbol{x}$ is a document vector and y is a label, so that

$$f_i(y, \boldsymbol{x}) = \begin{cases} 1 & \text{if property of } \boldsymbol{x} \text{ occurs with label y} \\ 0 & \text{otherwise.} \end{cases}$$

More precise the document vector is stored as bag-of-words vector. The feature is 1 if the property occurs together with the label and 0 if not. The features are learned from data with pointwise mutual information (PMI), which is obtained by

$$\text{PMI}(x, y) = \lg \frac{P(x, y)}{P(x)P(y)}$$

and can be understood as an association measure that helps to decide whether a feature is informative or not Bouma (2009).This ensures that only relevant features are considered Bouma (2009). By doing so, the classification process works faster and returns more reliable results. After learning the features, the classifier assumes some random weights between -10 and +10 for each feature and enters an iterative process to improve the feature weights. The iterative training of the weights is done by calculating the derivation of the weights and adding them to the current weights. Then it checks if the accuracy has been improved.

3 Experiments

3.1 Experimental Design

For all model configurations, our hyperparameters of the number of authors, the maximum training iterations, the accuracy threshold and the number of features per author were left untouched. The training stopped when the accuracy improvement fell below 0.001 or after 100 iterations (which was never reached during training).

With at least 30 poems per author and 1569 datapoints, the split between test and training data was pseudo-randomized 75 to 25 to ensure sufficient coverage of each author in training and evaluation. This way the least prolific author (Edmund Spenser) had 25 poems for training and eight for the evaluation. The poems alloted for training were also used to compute the pointwise mutual information because of the aforementioned (**??**) data sparsity. This was done despite the risk of overfitting since a second division of the data would leave us with ten to fifteen poems per author for both the feature extraction and training, which is not sufficient for either method (PMI or Maximum Entropy training).

The features themselves consisted of individual tokens, the number of verses and stanzas as well as the rhyme scheme for each poem. The word features were obtained by converting the poem into a bag-of-words vector and retrieving the value (0 or 1) for a specific word in the vocabulary. This vocabulary was built from the tokenized training data as the classifier will only learn weights for features that are seen during training. A simple classifier with only the tokens was trained as a baseline for the poetry specific features.

The rhyme scheme was obtained from the first four lines of the poems. While there might be rhymes that span more than four lines (***abcda***), this is highly unlikely without a repetition of the first rhyme or another rhyme pair in the lines in between 1 and 5. The scheme was constructed by consecutively taking the last word of the first unmatched line and checking all other unmatched lines for rhymes with `pronouncing.rhymes(word)` (Parrish, 2015). This lead to a four letter string for each poem of the form:

$$\text{“}a\{a, b\}\{a, b, c\}, \{a, b, c, d\}\text{”}$$

The number of verses per poem were sorted in x-line steps after looking at the distribution in the training data. The steps were converted into bins of at least and at most x number of verses. Similarly, the number of blank lines in a poem was used to determine the number of stanzas and sorted into steps of x, y or z stanzas.

For our first experiments, the classifier was initialized with the 30 most informative features per author, which for 30 authors resulted in 900 features whose weights were trained. We compared the baseline of just words to a classifier with all features ("full"), combinations of words and only one of the advanced features (from here on referred to by the name of that feature, i.e. stanza

	Baseline	Full	Verse	Stanza	Rhyme
Accuracy					
Precision					
Recall					
micro F_1					
macro F_1					

Table 1: Evaluation of the different model configurations on the unseen test data and the model's accuracy on the training data for comparison.

model, rhyme model, verse model) and a model with all features ecxept for the words. After comparing these models with fixed hyperparameters, we changed the number of learned features and the size of the author set to observe the parameters' effect.

3.2 Results

For the training, the models rarely performed more than ten optimization steps before the accuracy stopped changing. Tracking the aggregated loss as well as the accuracy showed us that the loss was still high when the accuracy stopped improving. here graph? Training of the baseline model terminated with an accuracy 50% on the training data, which dropped to a micro f_1 score of .115 with the unseen test data. This pattern was also present in the other model configurations as shown in table 1.

3.3 Error Analysis

Given the configurations in the Results section, what are frequent sources of errors

- specifics and numbers about errors?
- overprediction of alphabetically first author
- many authors not predicted (uneven data distribution or bad features)
- feature weights converge similarly (no real weighting)

4 Summary & Conclusion

Explain and summarize your results on a more abstract level. What is good, what is not so good. What are the main contributions in your experiments?

5 Future Work

What did you have in mind what else your would have liked to experiment with? Other ideas?

- other models (e.g. Neural Net)
- other features (Topics from Poetry Foundation website)
- feature interdependencies/more data analysis
- genre interaction with author classification (multitask learning?)

References

Shlomo Argamon, Moshe Koppel, James W. Pennebaker, and Jonathan Schler. 2009. Automatically profiling the author of an anonymous text. *Communications of the ACM*, 52(2):119–123.

Gerlof Bouma. 2009. Normalized (pointwise) mutual information in collocation extraction. In *Proceedings of GSCL 30*, pages 31–40.

Kim Luyckx and Walter Daelemans. 2011. The effect of author set size and data size in authorship attribution. *Literary and linguistic Computing*, 26(1):35–55.

Kamal Nigam, John Lafferty, and Andrew McCallum. 1999. Using maximum entropy for text classification. *IJCAI-99 workshop on machine learning for information filtering*, 1(1):61–67.

Miles Osborne. 2002. Using maximum entropy for sentence extraction. In *Proceedings of the ACL-02 Workshop on Automatic Summarization*, pages 1–8.

A Contributions

Who implemented what? Who participated in the design of which components? Who wrote which part of the review?

B Declaration of Originality