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

Short motivation and explanation of relevancy of your task, research questions/hypothesis

- what is author classification
- why poetry
- research question
  - Goal: find features inherent to poetry
    - is our goal possible
    - which features are good
  - Problem: style features might depend on medium (e.g. Limmerick) more than on the author
- Motivation: ??

2 Method

2.1 Description of the Methods

Generally speaking, a maximum entropy classifier is used for generating a probability distribution based on some training data. 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 shown below, where $f_i(y, \boldsymbol{x})$ is a feature and λ_i the corresponding weight:

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

Furthermore, the maximum entropy classifier presupposes a dependence relation between the features. 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 $\vec{x}$ is a document vector and y is a label. 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, as you can see below:

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

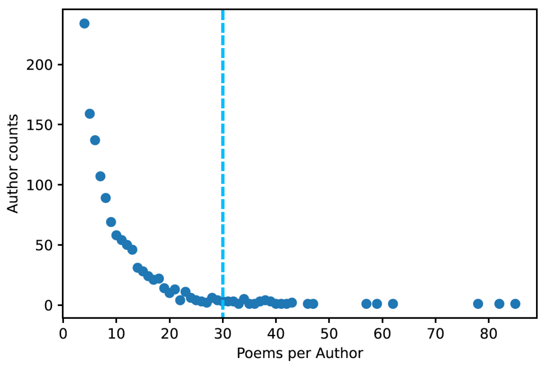
The features are learned from data with pointwise mutual information (PMI), which is an association measure that helps to decide whether a feature is informative or not (Bouma 2009). This ensures that only relevant features are considered. By doing so, the classification process works faster and returns more reliable results. The formula for PMI looks like the following:

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

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.

2.2 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.

3 Experiments

3.1 Experimental Design

Explain how you perform your experiments, which data is used, statistics of data.

- data statistics (decision for number of authors as hyperparameter)
- Train/Test split
- Program:
  - Hyperparameters: accuracy threshold, track loss & accuracy, #features/author
- Baseline (bag of words), Advanced: #verses, #stanzas, rhyme scheme

3.2 Results

Explain how your model performs, different models or configurations of your models.

- Feature combinations: baseline=BoW, advanced=all, other=?
- recall/precision/f_1 for all combinations (table)

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?)

A Contributions

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

B Declaration of Originality