# Spooky Authors Identification

Eduardo Brandao, Mohammad Poul Doust

Msc Machine Learning and Data Mining (MLDM)
Jean Monnet University

15th January 2020

# Overview


1. Introduction
2. Data Description
3. Methods
4. Results
5. Conclusion and outlook
6. Future work

# Literary text $\neq$ standard text

From Poe's 25-page *The Cask of Amontillado* :

- *"Ugh! ugh! ugh!—ugh! ugh! ugh!—ugh! ugh! ugh!—ugh! ugh! ugh!—ugh! ugh! ugh!"*
- *"Nemo me impune lacessit."*
- *In painting and gemmary, Fortunato, like his countrymen, was a quack — but in the matter of old wines he was sincere.*

# Several difficulties

- Rendering of character speech
- Use of non-standard vocabulary
- Semantics is context dependent (different characters).

# Standard techniques don't work well

| **Difficulty** | **Approach** |
| --- | --- |
| Removing stopwords and punctuation reduces semantic information but reduces noise | *Combine independent models with and without stopword and punctuation.* |
| Pretrained models capture semantics best, but are trained on modern text, and semantics evolves in time | *Combine models that use pretrained historical embeddings, and others trained on the problem corpus.* |
| Literary authors use language in a specific way. | *Combine models that can capture subword information, and models that take into account parts-of-speech and other syntactic information.* |
| There is no single model that can resolve all these different axis at the same time | *Use an ensemble of methods, while trying to keep them, in some sense, orthogonal.* |

# Our approach

Split the problem into 4 orthogonal axis:

- Classify on *vocabulary* $\longrightarrow$ TFIDF
- Classify on *meaning* of the vocabulary $\longrightarrow$ Pre-trained embeddings + handcrafted features
- Classify on *morphology* $\longrightarrow$ Fast Text
- Classify on *context* $\longrightarrow$ LSTM

... and then combine the results $\longrightarrow$ Ensemble

# Data Description

The structure contains three columns: id, text and author. The author columns values are:

- EAP: Edgar Allan Poe
- MWS: Mary Shelley
- HPL: HP Lovecraft

| **id** | **text** | **author** |
|---|---|---|
| id16607 | Here we barricaded ourselves, and, for the present were secure. | EAP |
| id22605 | To be near him, to be loved by him, to feel him again her own, was the limit of her desires. | MWS |
| id17569 | It never once occurred to me that the fumbling might be a mere mistake. | HPL |

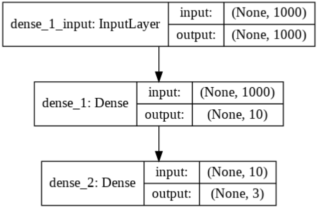
It is worth noticing that the dataset is not perfectly balanced.

# Methods: Simple Dense with Tf-idf

Data Processing:

- Punctuation To Words
- Tokenization
- Stop Words Removal
- Stemming

Goal: Capturing Vocabulary (axis) information



# Methods: Historical Embeddings

Motivation: for this choice that the authors in the dataset dates back to different time span:

- HP Lovecraft: 1890 to 1937
- Edgar Allan Poe: 1809 to 1849
- Mary Shelley: 1797 to 1851

Data Processing:

- Punctuation To Words
- Lowercase
- To Sequence
- Words N-Grams
- Tokenization

Goal: Capturing meaning of the terms (axis) for each author

# Methods: Functional Model

Data Processing:

- Punctuation To Words
- To Sequence
- Words N-Grams
- Tokenization

Learns features from different axis:

- Embeddings
- Tf-idf
- Hand-crafted features

# Methods: Functional Model

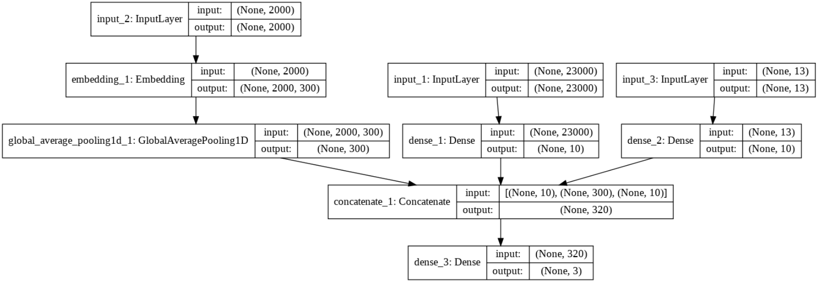



Figure 1: Historical Embedding Model Architecture

# Methods: Functional Model

| Feature | Description |
|---|---|
| num_words | # of words in each text |
| unique_word_fraction | Fraction of unique words to the number of words in the text |
| num_chars | # of characters |
| num_stopwords | # of stop words |
| punctuations_fraction | Fraction of punctuation over total number of characters |
| num_words_upper | # of Uppercase words |
| num_words_title | # of words that beings with an uppercase |
| mean_word_len | Average length of the words in the text |
| fraction_noun | Fraction of nouns over total words |
| fraction_adj | Fraction of adjectives over total words |
| fraction_verbs | Fraction of verbs over total words |
| fraction_adverbs | Fraction of adverbs over total words |

Table 1: Handcrafted features

Goal: Enhance previous models by introducing new crafted features capturing author style

# Methods: Fast Text Embeddings

The main motivation for this model is to capture subword information.

- Models that learn word representations ignore *morphology*
- Authors in this classification task use unique vocabulary
- Fast Text can incorporate subword information and is fast, simple, and efficient [**joulin2016bag**],[**bojanowski2017enriching**].

# What is Fast Text

Fast text is a (one layer) deep model [**bojanowski2017enriching**].

- Words+ngrams fed into lookup layer, word representations created
- Word embeddings averaged and fed into hidden layer.
- Averaged vector fed into a *linear* classifier, minimizing the log-likelihood over classes, returns softmax

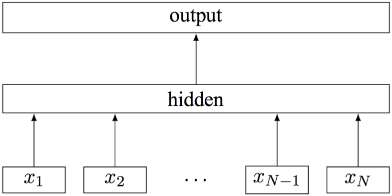



Figure 2: Fast Text Model: model architecture for a sentence with $N$ ngram features, $x_1, \ldots, x_N$

.

# Three different preprocessing

- with punctuation and stopwords (0.57 loss)
- without punctuation and stopwords (0.82 loss)
- without punctuation but with stopwords (0.62 loss)

# Keeping punctuation and stopwords is important

# Methods: LSTM

We chose LSTM in an attempt to capture the context-sensitive semantics of terms in literary text [**gers2001lstm**].

- We trained the model on the training set with categorical cross-entropy loss
- 'Adam' optimizer, over 3 epochs, with a batch size of 16
- Tried a variety of architectures, with and without pretrained embeddings (also bidirectional) with results in terms of loss the 0.7 – 1.2 range.
- Settled on the simplest architecture, which gave gave the best result (0.47 loss).

# LSTM data processing

Nltk tokenized text to a keras embedding layer. Alternatives' lackluster scores, and LIME[**ribeiro2016should**] motivated this choice.

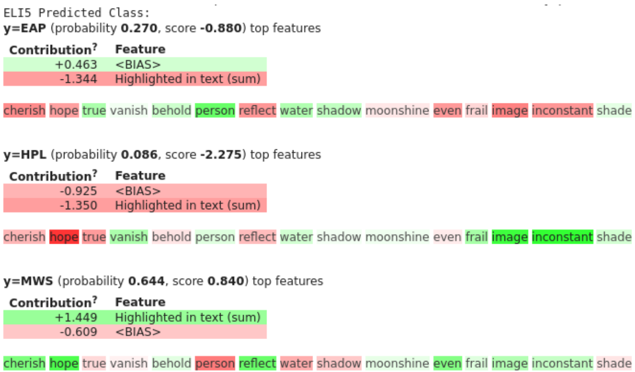



Figure 3: True class MWS. Decisions based on words like "person" and "appeared", which don't carry author specific meaning. True for all LSTM pretrained embeddings models. Best results with the **simplest** method.

# Methods: Final Ensemble

We considered a number of alternatives for the ensemble:

1. **normalized average classifier probabilities**
2. random convex combinations of classifier probabilities
3. normalized linear combination of classifier probability, weighted by inverse test loss
4. normalized maximally confident (per text) classifier for each class
5. normalized linear combination of classifier probability, weighted by confidence (all corpus)
6. normalized linear combination of classifier probability, weighted by confidence (per text)
7. normalized linear combination of classifier probability, weighted by precision (all corpus)

# Results

Evaluation calculated using multi-class logarithmic loss:

$$\text{logloss} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij}\log\left(p_{ij}\right)$$

N: Test set size

M: 3 classes

| **Method** | **Result (Loss)** |
|---|---|
| Simple Tf-idf Model | 0.45 |
| Historical Embedding MLP | 0.36 |
| TF-idf + Historical Embedding + handcrafted features | 0.35 |
| Fast Text Embedding | 0.57 |
| LSTM | 0.47 |
| Ensemble of different methods | 0.33 |

Log loss more penalizes the wrong confident classification. Hence, the ensemble achieved the best among others.

# Conclusion and outlook

- Main idea: splitting literary text classification into several axis.
- Used only deep learning tools
- Strove to make models interpretable, using LIME [**ribeiro2016should**] and analysing word embedding vector space.
- Exploration led to iteratively adapting our approach
- Combined axis into an ensemble of classifiers focusing on different axis of literary authors' text.

# Improving our approach

- **Functional model:** adding part-of-speech, sentiment analysis, and gender, for example, to the list of features – in line with our analysis of the particularities of literary text.
- **Historical embeddings:** parameter tuning.
- **Fast Text model:** use a pretrained historical model – but that would remove the orthogonality between our approaches.
- **LSTM model:** draw inspiration from previous attempts and the literature.
- **Metric Learning:** could be explored in such problem by training Siamese networks.