Spooky Author Identification

Osman Malik, Kyle Cordes, Hasan Demirci, Akhil Dixit, Santosh Devarakonda
Introduction

- Explore the suitability of Naive Bayes, LSTM and CNN classifiers in predicting the author of excerpts from horror stories by Edgar Allan Poe, Mary Shelley, and HP Lovecraft.

- Use Naive Bayes as a base classifier to show the benefit of more advanced classifiers
Data Preprocessing and Word Embeddings

- NLTK Tokenizer
- Why we think stopwords and punctuation marks matter?
- Pre-trained vs generated embeddings
- GloVe vs Word2vec similarities
  - GloVe: Gold ~ Silver = 0.866
  - GloVe: Gold ~ Water = 0.391 (Almost half of above)
  - Word2vec: Gold ~ Silver = 0.210
  - Word2vec: Gold ~ Water = 0.072 (One third of above)
Word Embeddings Visualizations (t-SNE)
Word Embeddings Visualizations (t-SNE)
Naive Bayes Classification (Theory)

- Training = calculating conditional probabilities and class probability
- To classify a document. Calculate the probability for each class given the feature vector using Bayes Theorem
- Use log to get better precision
- Remove stop words?
- Take most occurring words?
- Take least occurring words?

\[
P(A|B) = \frac{P(B|A)P(B)}{P(A)}
\]

\[
P(C_i|X) = \frac{P(x_1|C_i)P(x_2|C_i)P(x_3|C_i) \cdots P(x_n|C_i)P(C_i)}{P(X)}
\]

\[
\text{argmax}_{C_i \in \mathcal{C}}(P(C_i|X)) = \text{argmax}_{C_i \in \mathcal{C}}(P(x_1|C_i)P(x_2|C_i)P(x_3|C_i) \cdots P(x_n|C_i)P(C_i))
\]

\[
\text{argmax}_{C_i \in \mathcal{C}}(\log(P(C_i|X))) = \text{argmax}_{C_i \in \mathcal{C}}(\log(P(x_1|C_i)P(x_2|C_i)P(x_3|C_i) \cdots P(x_n|C_i)P(C_i)))
\]

\[
= \text{argmax}_{C_i \in \mathcal{C}}(\log(P(x_1|C_i)) + \log(P(x_2|C_i)) + \cdots + \log(P(x_n|C_i)) + \log(P(C_i)))
\]
Naive Bayes Classification (Results)

- Removing stop words shows improvement
- Training with less of the vocabulary increases accuracy
- Best result is training with top 100 most frequently used words
Classification using Weka

<table>
<thead>
<tr>
<th>Method</th>
<th>Accuracy</th>
<th>F measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random Forest</td>
<td>44.21%</td>
<td>0.425</td>
</tr>
<tr>
<td>Multi Class Logistic Regression</td>
<td>41.355%</td>
<td>0.331</td>
</tr>
<tr>
<td>Support Vector Machine</td>
<td>40.36%</td>
<td>0.286</td>
</tr>
<tr>
<td>Naïve Bayes</td>
<td>35.99%</td>
<td>0.28</td>
</tr>
<tr>
<td>K-NN K = 15</td>
<td>38.636%</td>
<td>0.357</td>
</tr>
<tr>
<td>Random Tree</td>
<td>38.456%</td>
<td>0.384</td>
</tr>
<tr>
<td>J48 decision tree</td>
<td>38.11%</td>
<td>0.379</td>
</tr>
</tbody>
</table>
Test options
- Use training set
- Supplied test set
- Cross-validation Folds 10
- Percentage split % 66

Classifier output

Summary:
- Correctly Classified Instances: 2572, 38.036%
- Incorrectly Classified Instances: 4805, 61.964%
- Kappa statistic: 0.8405
- Mean absolute error: 0.4268
- Root mean squared error: 0.4815
- Relative absolute error: 97.4266%
- Root relative squared error: 380.4939%
- Total Number of Instances: 657

Detailed Accuracy By Class:

Confusion Matrix:

a  b  c
1672 263 666
1392 317 456
1349 159 583

Test options
- Use training set
- Supplied test set
- Cross-validation Folds 10
- Percentage split % 66

Classifier output

Summary:
- Correctly Classified Instances: 2572, 38.1083%
- Incorrectly Classified Instances: 4128, 61.8917%
- Kappa statistic: 0.8592
- Mean absolute error: 0.4297
- Root mean squared error: 0.5904
- Relative absolute error: 90.6256%
- Root relative squared error: 125.4667%
- Total Number of Instances: 657

Detailed Accuracy By Class:

Confusion Matrix:

a  b  c
1378 608 825
788 586 579
870 449 733

result list (right-click for options)
LSTM Classification

- Single Layer LSTM, how loss varies with:
  - Batch size
  - Number of LSTM cells
  - Dropout
  - Sequence Length
  - Number of Layers
LSTM Classification

- Tune number of LSTM units
- Start with 12 units, increase until 250 units.
- Model does not improve much.
- After 50 units, model overfits quickly.
LSTM Classification

- Tune batch size
- Start with 50 sentences, increase until 1000.
- No effect on predictive performance.
- No clear advantage for particular batch size.
- Models with smaller batch size overfits quicker.
LSTM Classification

- Tune sequence length.
- Start with maximum 10 words per input.
- Every increment increases predictive power.
- Diminishing returns.
LSTM Classification

- Tune number of layers
- Candidates are 1-2-3 layers.
- Use a subset of original data, this is harder to train.
- Does not improve.
CNN Classification

- 2 Models:
  - 3 Layer Convolution + Max Pooling
  - 1 Layer Convolution + Max Pooling

- How loss varies with:
  - Training Epochs
  - Embedding Algo - GloVe or Gensim
  - Filter Size
  - Multiple Filters
  - Dropout
  - Punctuation
CNN Classification

- 3-layer CNN not suitable for NLP
- 1-layer CNN optimum at 10 epochs
- Gensim, GloVe no difference => same criteria of semantic similarity

![Graph showing Test Loss vs Optimization Epochs for Filter Size = 3](image)
CNN Classification

● V-shape:
  ○ Too small => no context
  ○ Too large => many features washed by pooling

● 2 optimum
  ○ Negation phrases
  ○ Adjective-noun pairs
  ○ Conjunction pairs: ‘and the’

● Future: Tune filter size with pooling function
CNN Classification

- Use filters sized 2, 3
- Global Max Pool and aggregate data
- Dropout not useful
  - Underfits model => dropout rate too high
- Punc. not useful
- Custom embedding better than globally trained GloVe.6B
Future Work

● LSTM: No punctuation, stop words.
● CNN: Tune filter size with pooling function
  ○ Each filter: own pooling function for specific feature extraction
● Replace punctuation with context tags
  ○ ‘!’ replaced by <fright>, <surprise>, <excite> etc.
Conclusion

● Word Embeddings
  ○ Word2vec runs slower than GloVe
  ○ GloVe tends to use huge amount of memory
  ○ Accuracy is almost the same!

● LSTM: Best score at 0.48120, accuracy is 83%.

● CNN: Best score at 0.39822
  ○ 1-Layer CNN with 2, 3 word filters

● Naive Bayes: Best Score 20.22 and roughly 41% accuracy
  ○ Most occurring 100 words trained. English stop words removed.
Thank You
References

- Understanding Convolutional Neural Nets for NLP, Denny Britz, WildML
- Convolutional Neural Networks for Sentence Classification, 2014, Yoon Kim
Backup

- Why CNN Better?
- Author style may favor short filtering of CNN compared to long sentence sequence of LSTM