Authorship Attribution for YELP Reviews

Task

In this project we are trying to match unlabeled review text with its author by learning their writing style from the labeled text written by the same author. This can be modelled as a multi-class single-label classification problem in Machine Learning. The final goal of this task is to identify the true author of a review based on their writing style from a learned Deep Learning and Statistical Machine Learning model.

Work Done So Far

Implementation Details

a) Primary Usage - Expected Input and Output
   Input: Unlabelled textual review, either generated by YELP or by a Sequence model which has learnt various writing styles from the training dataset.
   Output: Predicted Author from a predefined pool of authors or UNK
b) Evaluation Metrics
   Macro-averaged F1 scores - as used in PAN CLEF Shared Task of Authorship Attribution
   This calculates F1-score for each label/candidate author, and finds their unweighted mean. This does not take label imbalance into account, although our dataset is perfectly balanced.
c) Technologies Used
   - Python 3.5
   - Jupyter Notebook
   - Scikit-learn, TensorFlow, Keras, NLTK

Organization of Code

Dataset Preparation

We use the data from review.json for our project. The data preparation steps are as follows. The reviews are grouped by author and 50 authors with the highest review count are selected. For each of these authors, we concatenate all their reviews into one huge review. From this huge review, we select the first 100,000 characters for training and the next 100,000 characters for testing. The training dataset now has 50 (review, author) tuples, where each review belongs to

one author and is 100,000 characters long. For the test set, we split each 100,000 character review into a hundred 1000 character reviews. So the test set will have 5000 (review, author) tuples, where each review is 1000 characters long, and there are 100 such reviews present for each author. We now have the training and testing set which we serialize and store as pickle files. These can be loaded from their files and used later.

TF-IDF Model

- A simple method which uses TF-IDF vectors and Support Vector Machine classifiers for identifying the author. A Support Vector Machine is a linear classifier which finds a separating hyperplane that maximizes the distance between two or more classes. In case of inseparable data, SVM uses a slack variable to allow misclassifications but adding a penalty.
- SVM is extremely fast in both training and predicting/evaluating phases, and hence is a perfect first model to try out before moving on to other complex learning models.
- SVM has been widely used for tasks like Named Entity Recognition, Authorship Profiling, such as gender and personality and gender prediction among other NLP tasks. For multi-class classification we use a OneVsRest classifier where we train an SVM model for every candidate author.
- We tried two different Feature preprocessing techniques before feeding it into the SVM:
    a) A union of word and character n-grams (unigrams and bigrams for words, bigrams and trigrams for characters). This feature set consisted of 28,206 distinct terms fitted only on the training dataset and the terms in the test texts not found in training was ignored.
    b) The texts were first pre-processed using NLTK. Stopwords and punctuations were removed before retaining only the word stems. A custom word and punctuation tokenizer from NLTK was used to tokenize the text.

  These pre-processed texts were vectorized using TF-IDF where the training vocabulary is used and terms in the test texts not found in the vocabulary are ignored.
- We trained an SVM model on the 50 training texts available with us using a LinearSVC. Predictions were generated for each of the texts based on the confidence scores assigned by the SVMs.
- With the initial test texts, the first set of features fed into the SVM model gave an accuracy of 46% and the second set of features fed into the SVM model gave an accuracy of 38%.
- The reason for such low accuracy using both feature sets was due to the fact that the average lengths of our test texts was only about 180 words, which was too short for the model to make accurate predictions. After combining 5 test texts and then predicting on

this concatenation gave an accuracy of 90% for the first set of features and 72% for the second set of features.

Character based Language Model (GRU-RNN)

- Recurrent Neural Networks with word embeddings have been used successfully in other text classification tasks, such as sentiment analysis.Instead of using frequency based features, as we discussed in the TF-IDF model , a Neural Network can “read” a text sequentially, similarly to how a human might. The network can then learn the most relevant words and relations, and discriminate between classes.
- GRU is more generalized than LSTM and takes lesser time for training and predictions. Since we need the generalized writing style of a candidate author to be learnt, GRU seemed like the best option for this task and is supported by existing literature [3].
- We use an encoder architecture, which reads each review one word at a time using a recurrent neural network. The resulting document vector is fed to a fully connected layer for classification. We tried only unidirectional encoding as [4] states that bi-directional encoding showed no improvement.
- Pre-trained Glove embeddings (glove.6B.300d) of size 300 was used as an input where the vocabulary was built out of the entire training text. We built the vocabulary out of the whole training set. Unknown words were replaced with UNK token. We experimented with several vocabulary sizes by taking the most frequently occurring words. For each author, we split the 50000 character chunks into ten 5000-character samples for training. We tokenized each text chunk into words and transformed them into a matrix of one hot-encoded vectors. We padded the length of each sequence to 900 words.
- The one-hot-encoded reviews extract the actual word embedding vector from an 128-dimensional word embedding matrix. The resulting sequence of vectors was fed into the Gated Recurrent Unit layer with a dimension of 256. The final output of the GRU layer was then passed to a fully connected output layer with softmax activation to directly identify the author.
- We employed Batch Normalization in the recurrent layer to improve training speed and Dropout with a keep probability of 0.1 and 0.3 as a form of regularization. During training, we used 10% of the training data as a validation set to measure generalization.
  We used the Adam optimizer and cross entropy as the loss function. We trained the network for 10 epochs for each candidate model.
- Each such GRU model was trained for every candidate author to identify distinctive writing style for every author. For predicting, we first pre-processed the text into the expected embedding size and evaluated on every trained candidate model. The model

that gave the lowest cross entropy loss was then chosen and the author id associated with this author model was the final prediction.

Tensorflow Model Summary

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_5 (Embedding)      (None, None, 300)         25500
_________________________________________________________________
dropout_8 (Dropout)          (None, None, 300)         0
_________________________________________________________________
batch_normalization_8 (Batch (None, None, 300)         1200
_________________________________________________________________
gru_4 (GRU)                  (None, 256)               427776
_________________________________________________________________
dropout_9 (Dropout)          (None, 256)               0
_________________________________________________________________
batch_normalization_9 (Batch (None, 256)               1024
_________________________________________________________________
dense_4 (Dense)              (None, 85)                21845
=================================================================
Total params: 477,345
Trainable params: 476,233
Non-trainable params: 1,112
_________________________________________________________________
```

Hyperparameter Tuning for our GRU-RNN model

Hyperparameters	Value
Batch Size	**128**, 256
Epochs	5, **10**, 15
Embedding Size	100, 200, **300**
GRU Hidden Layers	128, **256**
Dropout Probability	**0.1**, **0.3**, 0.5
Dense Layer Activation	**softmax**
Loss Function	**CrossEntropy**, SGD
Optimizer	**Adam**, Adagrad
Optimizer Learning Rate	**0.001**
Validation Split	**0.1**, 0.2

- We achieved an accuracy of 94.8% on our test data. Time taken for training was about 10 hours and the time taken for predicting on 500 test texts was about 16 hours. The huge difference between training and testing times is due to the fact that the model only has to learn the writing styles for 50 authors, whereas it has to predict the writing style of 500 authors. Each test text is evaluated on each of the 50 trained author models and took about 3 minutes per text.
- Although the generative language modelling predictions took an excessively long time to generate, these are easily parallelizable. Because each language model needs to evaluate each text, it would be trivial to share this workload among multiple cores or machines by deploying one model and all test texts per machine.

Completed Milestones

- A random baseline approach would have a 1=50 chance of guessing the correct author, so we would expect an accuracy of 0:02.
- Given that the test texts are relatively short (5 000 characters is approximately 900 words), the task is not easy and it is evident that both the SVM with TF-IDF model and the character-based language (GRU-RNN) model are proficient at discriminating between authorship style.
- We are pleased with the indication that the less-common GRU neural approach proved suitable for the task.

Results

Model / Parameters	**TF-IDF based SVM**	**Character based GRU-RNN**
Accuracy	90.8%	94.8%
Macro averaged F1 - score	90.03%	93.11%
Time Taken	A few seconds	10 hours for training, 3 minutes per prediction

Upcoming Milestones

We have tried both statistical and deep learning models already as specified in our milestone document. We will be proceeding with the following models in the coming weeks:

- **Ensemble method** combining multiple models like TF-IDF, word2vec and n-gram.
- Vanilla **word2vec** and **Glove** embeddings as features.
- Combining **TF-IDF vectors with word embeddings**.
- Trying **LSTM, CNN** and other neural network models which have worked well for sequence modeling (Ex. BERT)

References

1. Joachim Diederich, J¨org Kindermann, Edda Leopold, and Gerhard Paass. 2003. Authorship attribution with support vector machines. Applied intelligence 19(1):109–123.
2. Moshe Koppel and Jonathan Schler. 2003. Exploiting stylistic idiosyncrasies for authorship attribution. In Proceedings of IJCAI'03 Workshop on Computational Approaches to Style Analysis and Synthesis. volume 69, page 72.
3. Douglas Bagnall. 2015. Author identification using multi-headed recurrent neural networks. arXiv preprint arXiv:1506.04891
4. Douglas Bagnall. 2016. Authorship clustering using multi-headed recurrent neural networks. arXiv preprint arXiv:1608.04485
5. Mart Busger op Vollenbroek, Talvany Carlotto, Tim Kreutz, Maria Medvedeva, Chris Pool, Johannes Bjerva, Hessel Haagsma, and Malvina Nissim. 2016. GronUP: Groningen User Profiling—Notebook for PAN at CLEF 2016. In Krisztian Balog, Linda Cappellato, Nicola Ferro, and Craig Macdonald, editors, CLEF 2016 Evaluation Labs and Workshop – Working Notes Papers, 5-8 September, E´vora, Portugal. CEUR-WS.org. http://ceur-ws.org/Vol-1609/.
6. Moshe Koppel and YaronWinter. 2014. Determining if two documents are written by the same author. Journal of the Association for Information Science and Technology 65(1):178–187.
7. Kim Luyckx and Walter Daelemans. 2008. Authorship attribution and verification with many authors and limited data. In Proceedings of the 22nd International Conference on Computational Linguistics- Volume 1. Association for Computational Linguistics, pages 513–520.
8. Efstathios Stamatatos. 2009. A survey of modern authorship attribution methods. Journal of the American Society for information Science and Technology 60(3):538–556.
9. Dani Yogatama, Chris Dyer,Wang Ling, and Phil Blunsom. 2017. Generative and discriminative text classification with recurrent neural networks. arXiv preprint arXiv:1703.01898 .
10. M Rahgouy, HB Giglou, T Rahgouy, MK Sheykhlan, E Mohammadzadeh. Cross-domain Authorship Attribution: Author Identification using a Multi-Aspect Ensemble Approach -

Notebook for PAN at CLEF 2019. In CLEF 2019 Evaluation Labs and Workshop–Working Notes Papers. CEUR-WS. org.

11. Rangel Pardo, F., Montes-y-Gómez, M., Potthast, M., Stein, B.: Overview of the 6th Author Profiling Task at PAN 2018: Cross-domain Authorship Attribution and Style Change Detection. In: Cappellato, L., Ferro, N., Nie, J.Y., Soulier, L. (eds.) CLEF 2018 Evaluation Labs and Workshop – Working Notes Papers, 10-14 September, Avignon, France. CEUR-WS.org (Sep 2018)
12. Murauer, B., Tschuggnall, M., Specht, G.: Dynamic Parameter Search for Cross-Domain Authorship Attribution—Notebook for PAN at CLEF 2018. In: Cappellato, L., Ferro, N., Nie, J.Y., Soulier, L. (eds.) CLEF 2018 Evaluation Labs and Workshop – Working Notes Papers, 10-14 September, Avignon, France. CEUR-WS.org (Sep 2018)