Deep Character-level Convolutional Neural Network for Authorship Attribution

Dejan Milacic

Deep Learning Project Documentation

Problem Description

Text classification is the supervised learning task of assigning one or more predefined categories to a text based on its content. Authorship attribution is a type of text classification task which aims to determine a text's author among a set of candidate authors. This project explores the task of determining the author of a blog post from a set of 10 candidate authors. This is done by implementing Naive Bayes, SVM, and CNN models.

Data Sources

The models discussed were trained and tested on the Blog Authorship Corpus:
http://u.cs.biu.ac.il/~koppel/BlogCorpus.htm
The corpus is available in CSV format on Kaggle:
https://www.kaggle.com/rtatman/blog-authorship-corpus
The corpus may be freely used for non-commercial research purposes. Any resulting publications should cite the following:

Jonathan Schler, Moshe Koppel, Shlomo Argamon, and James Pennebaker. 2006. Effects of Age and Gender on Blogging. *AAAI Spring Symposium: Computational Approaches for Analyzing Weblogs*, pages 199–205.

Literature Review

Convolutional Neural Networks (CNNs) were originally invented for computer vision, but have been shown to also be effective for Natural Language Processing tasks including text classification. In this section, I give an overview of research using CNNs for text classification tasks. I compare and contrast: shallow vs deep architectures, words vs characters as features, and authorship attribution vs other text classification tasks.

Kim (2014) trains a shallow CNN with one layer of convolution and max pooling, a variant of a model by Collobert et al. (2011). They use pre-trained word2vec word vectors to encode the inputs. They also propose a model with two word input channels: one which is updated during training, and one which remains static.

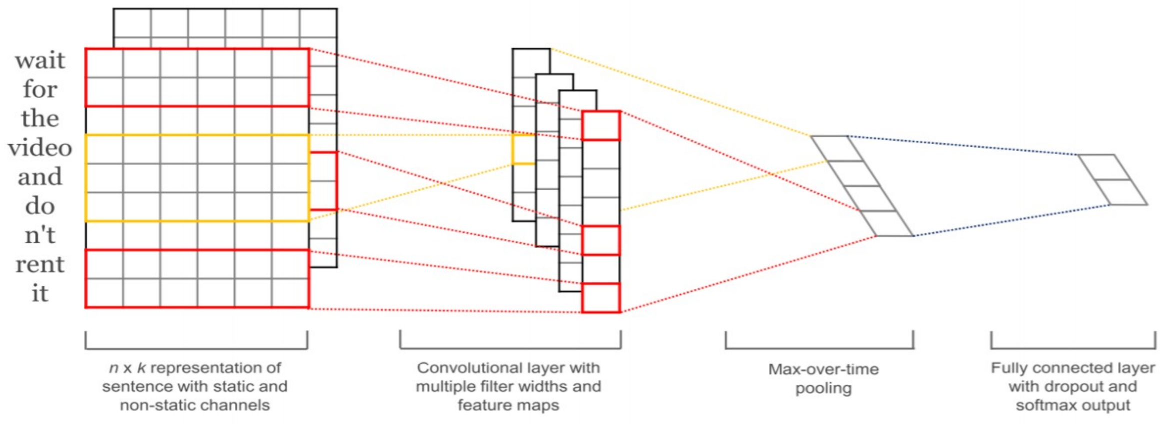
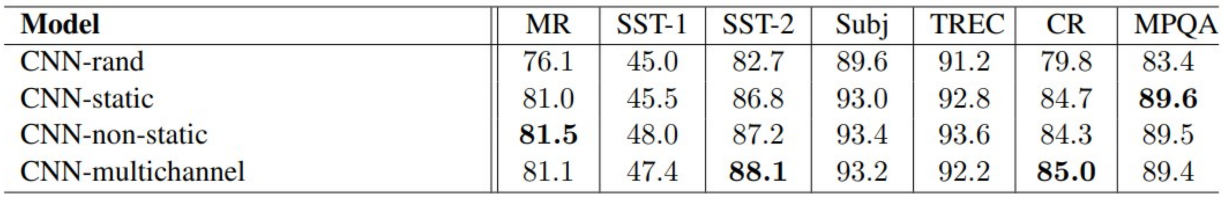

wait
for
the
video
and
do
n't
rent
it
n x *k* representation of sentence with static and non-static channels
Convolutional layer with multiple filter widths and feature maps
Max-over-time pooling
Fully connected layer with dropout and softmax output

Figure 1: Multi-channel CNN with two word channels from Kim et al. (2014)

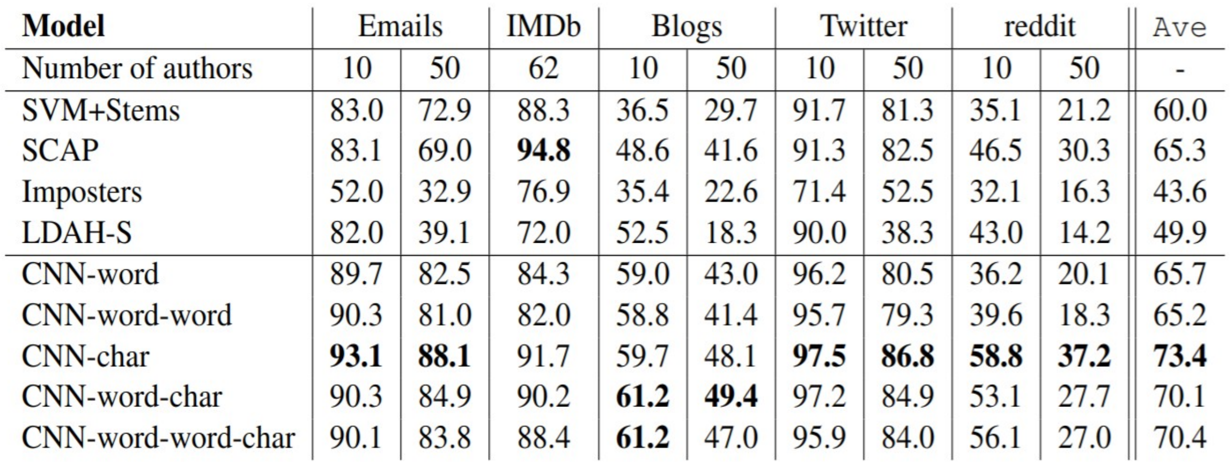
They train their models on several text classification tasks including sentiment analysis and text categorization. The number of classes ranges from 2 to 6.



Model	MR	SST-1	SST-2	Subj	TREC	CR	MPQA
CNN-rand	76.1	45.0	82.7	89.6	91.2	79.8	83.4
CNN-static	81.0	45.5	86.8	93.0	92.8	84.7	**89.6**
CNN-non-static	**81.5**	48.0	87.2	93.4	93.6	84.3	89.5
CNN-multichannel	81.1	47.4	**88.1**	93.2	92.2	**85.0**	89.4

Their models only improve upon the state of the art on binary sentiment analysis tasks. However, where their models do improve, they do so with a simpler architecture than previous models (Kalchbrenner et al., 2014; Socher et al., 2013). The author attributes this to the use of pre-trained word vectors as inputs. Results are mixed as to whether there is a clear benefit to the multi-channel architecture over the single-channel.

Ruder et al. (2016) implement a version of Kim's shallow architecture for authorship attribution. They add a character-level channel to the model for this task. They compare this model to traditional state-of-the-art models in authorship attribution on several datasets with 10 and 50 authors.



Model	Emails		IMDb	Blogs		Twitter		reddit		Ave
Number of authors	10	50	62	10	50	10	50	10	50	-
SVM+Stems	83.0	72.9	88.3	36.5	29.7	91.7	81.3	35.1	21.2	60.0
SCAP	83.1	69.0	**94.8**	48.6	41.6	91.3	82.5	46.5	30.3	65.3
Imposters	52.0	32.9	76.9	35.4	22.6	71.4	52.5	32.1	16.3	43.6
LDAH-S	82.0	39.1	72.0	52.5	18.3	90.0	38.3	43.0	14.2	49.9
CNN-word	89.7	82.5	84.3	59.0	43.0	96.2	80.5	36.2	20.1	65.7
CNN-word-word	90.3	81.0	82.0	58.8	41.4	95.7	79.3	39.6	18.3	65.2
CNN-char	**93.1**	**88.1**	91.7	59.7	48.1	**97.5**	**86.8**	**58.8**	**37.2**	**73.4**
CNN-word-char	90.3	84.9	90.2	**61.2**	**49.4**	97.2	84.9	53.1	27.7	70.1
CNN-word-word-char	90.1	83.8	88.4	**61.2**	47.0	95.9	84.0	56.1	27.0	70.4

Table 3: Micro-averaged F1 scores of our CNN variants and state-of-the-art authorship attribution methods for 10 and 50 authors (except for IMDb) in the chosen domains. We also show a model's average performance (Ave) across all scores to give an estimate of how well a model generalizes across domains and consequently might perform in a new domain.

Their CNNs with character-level input channels outperform the traditional state-of-the-art methods on most datasets. This is due to their superior ability to capture stylistic information encoded on the character level. The CNNs are also much faster than traditional authorship attribution methods. The authors also conclude that CNNs can handle the class imbalances inherent in real-world applications.

Zhang et al. (2015) implement a deep character-level CNN for text classification. Characters in their prescribed alphabet are given one-hot encodings and documents are encoded to the first 1014 characters.

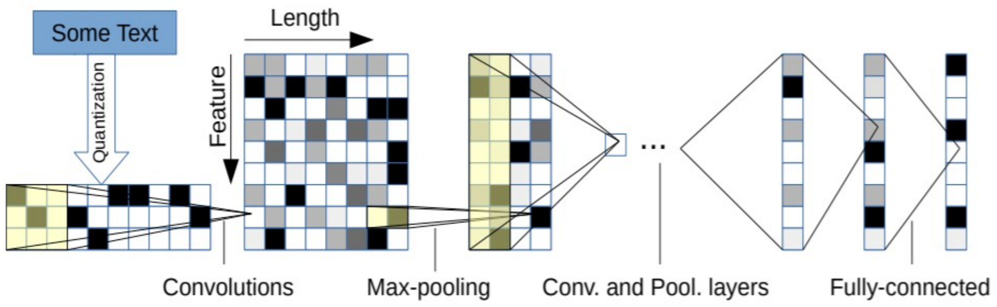
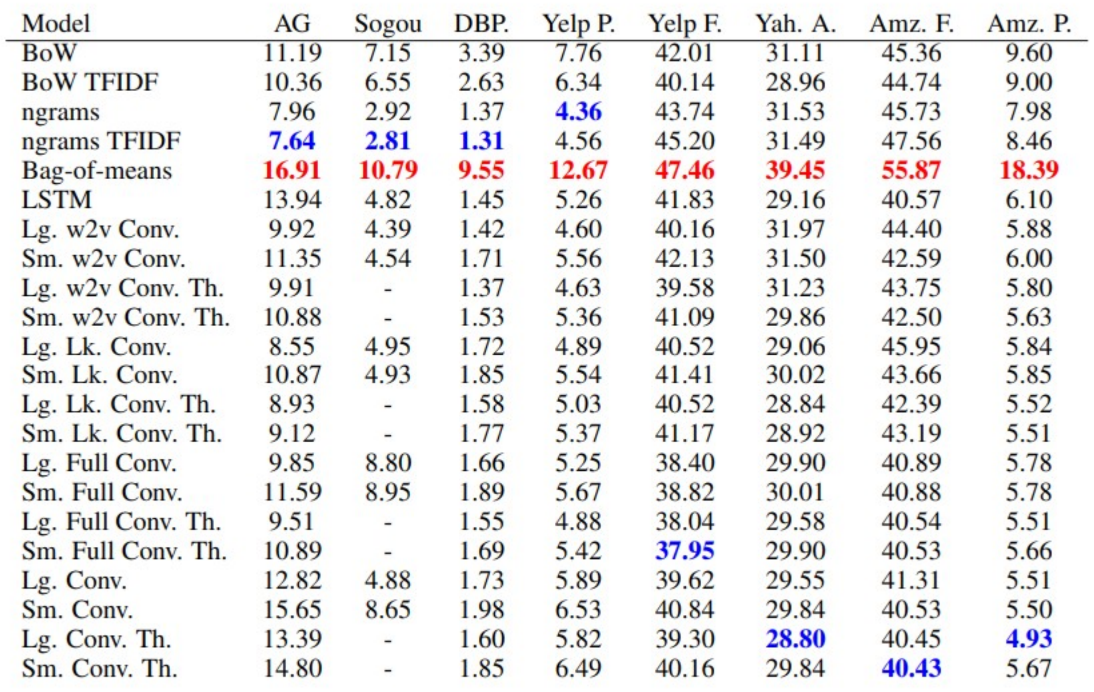


Figure 1: Illustration of our model

The model was tested on several text classification tasks including sentiment analysis and text categorization with between 2 and 14 classes. Results were compared with traditional text classification algorithms.

Table 4: Testing errors of all the models. Numbers are in percentage. “Lg” stands for “large” and “Sm” stands for “small”. “w2v” is an abbreviation for “word2vec”, and “Lk” for “lookup table”. “Th” stands for thesaurus. ConvNets labeled “Full” are those that distinguish between lower and upper letters



Model	AG	Sogou	DBP.	Yelp P.	Yelp F.	Yah. A.	Amz. F.	Amz. P.
BoW	11.19	7.15	3.39	7.76	42.01	31.11	45.36	9.60
BoW TFIDF	10.36	6.55	2.63	6.34	40.14	28.96	44.74	9.00
ngrams	7.96	2.92	1.37	**4.36**	43.74	31.53	45.73	7.98
ngrams TFIDF	**7.64**	**2.81**	**1.31**	4.56	45.20	31.49	47.56	8.46
Bag-of-means	**16.91**	**10.79**	**9.55**	**12.67**	**47.46**	**39.45**	**55.87**	**18.39**
LSTM	13.94	4.82	1.45	5.26	41.83	29.16	40.57	6.10
Lg. w2v Conv.	9.92	4.39	1.42	4.60	40.16	31.97	44.40	5.88
Sm. w2v Conv.	11.35	4.54	1.71	5.56	42.13	31.50	42.59	6.00
Lg. w2v Conv. Th.	9.91	-	1.37	4.63	39.58	31.23	43.75	5.80
Sm. w2v Conv. Th.	10.88	-	1.53	5.36	41.09	29.86	42.50	5.63
Lg. Lk. Conv.	8.55	4.95	1.72	4.89	40.52	29.06	45.95	5.84
Sm. Lk. Conv.	10.87	4.93	1.85	5.54	41.41	30.02	43.66	5.85
Lg. Lk. Conv. Th.	8.93	-	1.58	5.03	40.52	28.84	42.39	5.52
Sm. Lk. Conv. Th.	9.12	-	1.77	5.37	41.17	28.92	43.19	5.51
Lg. Full Conv.	9.85	8.80	1.66	5.25	38.40	29.90	40.89	5.78
Sm. Full Conv.	11.59	8.95	1.89	5.67	38.82	30.01	40.88	5.78
Lg. Full Conv. Th.	9.51	-	1.55	4.88	38.04	29.58	40.54	5.51
Sm. Full Conv. Th.	10.89	-	1.69	5.42	**37.95**	29.90	40.53	5.66
Lg. Conv.	12.82	4.88	1.73	5.89	39.62	29.55	41.31	5.51
Sm. Conv.	15.65	8.65	1.98	6.53	40.84	29.84	40.53	5.50
Lg. Conv. Th.	13.39	-	1.60	5.82	39.30	**28.80**	40.45	**4.93**
Sm. Conv. Th.	14.80	-	1.85	6.49	40.16	29.84	**40.43**	5.67

The deep CNN architectures only outperformed traditional methods on datasets with a very large number of training instances (>560,000).

In this project I applied a deep character-level CNN to an authorship attribution task with a much smaller amount of training instances and compared its performance to traditional machine learning algorithms.

Technologies and Issues

The models were run using Python 3 on Google Colab, using the GPU accelerator to run the CNN model. Pandas and NumPy were used to read and process the data from the CSV file. Natural Language Toolkit (NLTK) was used to pre-process the data for the traditional machine learning algorithms. Keras was used to pre-process the data for the CNN model and to implement the CNN model itself. Scikit-learn (sklearn) was used to encode the class labels for the CNN model, split the data into training and test sets, implement the traditional machine learning models, and calculate the metrics to evaluate the performance of each model.

I tried to implement the LDAH-S model (Seroussi et al., 2011; Ruder et al., 2016) as one of the traditional machine learning models for comparison, but it would not run in a reasonable amount of time.

Pre-processing

From the 19,320 bloggers included in the corpus, I separated the top 10 bloggers with the most posts:

Blogger	**Posts**
1	4,221
2	2,301
3	2,294
4	2,261
5	2,244
6	2,237
7	2,114
8	2,068
9	1,951
10	1,843
Total	23,534

Following Ruder et al. (2016), I do not equalize the number of posts and keep the imbalanced distribution common in real-world applications. The user IDs of each of the top 10 bloggers were encoded as 10-dimensional one-hot vectors to correspond with the 10 units in the output layer of the CNN model.

For the traditional machine learning models (Naive Bayes and SVM), the posts were lowercased and tokenized into alphanumeric strings and the resulting words were stemmed using the Porter stemmer (Porter, 1980). Stopwords were removed using NLTK's list of English stopwords.

For the CNN model, the posts were integer encoded using an alphabet of 86 characters, including both uppercase and lowercase English letters and 33 other characters. All digit characters were given the same encoding. Spaces and other characters not in the alphabet were given an out-of-vocabulary encoding, ‘UNK’.

```
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz
-,;.!?:'"/\|_@#$%^&*~`+=<>()[]{}
```

Encoded sequences were padded with zeros or cropped to a length of 1014 characters. The character alphabet was adapted from Zhang et al. (2015) and Ruder et al. (2016). The input feature length follows Zhang et al. (2015).

Analysis

Following Ruder et al. (2016), I split off 10% of the documents for the test set and 10% of the remaining documents for the validation set. The rest of the documents were used for training.

For the Naive Bayes model, the stemmed documents were vectorized using a term-document matrix of term counts in each document with a vocabulary of the 10,000 most frequent terms in the test set. The vectorized representations were then used to train and test the Naive Bayes classifier with Laplace smoothing.

For the SVM model, the stemmed documents were vectorized using a term-document matrix of TF-IDF features with a vocabulary of the 10,000 most frequent terms in the test set. All vectors were normed to unit length. The vectorized representations were then used to train and test the SVM classifier. The SVM used a one-vs-rest classifier with a linear kernel.

The CNN model takes the integer encoded documents as inputs. The embedding layer converts this representation from integer encoding to one-hot encoding. Padding characters are encoded as all-zero vectors and out-of-vocabulary characters are given a one-hot encoding. The network itself is an implementation of the “small” CNN model designed by Zhang et al. (2015). It is 9 layers deep with 6 convolutional layers and 3 fully-connected layers. There are 2 dropout modules between the fully-connected layers with dropout probability 0.5 for regularization. The layer configurations are listed in the tables below.

Convolutional Layers			
Layer	**Filters**	**Filter Size**	**Pooling Size**
1	256	7	3
2	256	7	3
3	256	3	None
4	256	3	None
5	256	3	None
6	256	3	3

Fully-connected Layers	
Layer	**Units**
7	1024
8	1024
9	10

The convolutional layers have stride 1 and pooling layers are non-overlapping. The output layer has 10 units for the 10 classes (authors). The pooling layers use max pooling and all layers use ReLU activation except for the output layer, which uses softmax. The weights are initialized using the Glorot uniform initializer. The model is trained using stochastic gradient descent with mini-batch size 128 for 15 epochs using the Adam optimizer.

I compare these models with those of Ruder et al. (2016), who also tested their models on the top 10 authors of the Blog Authorship Corpus. Their CNN models use only one convolutional layer with max pooling. Ruder et al. (2016) also tested some multi-channel architectures. In models with multiple word channels, the second is static and is not updated during training. From the results in the table below, we see that the deep character-level CNN implemented here achieves the highest performance of these models as measured by micro-averaged F1 score. We may conclude that the multiple convolutional layers allow this model to achieve a higher performance.

	Model	**Micro-averaged F1 score**
Ruder et al. (2016)	SVM+Stems	36.5
	SCAP	48.6
	Imposters	35.4
	LDAH-S	52.5
	CNN-word	59.0
	CNN-word-word	58.8
	CNN-char	59.7
	CNN-word-char	61.2
	CNN-word-word-char	61.2
This project	Naive Bayes	44.1
	SVM+Stems	45.6
	CNN-char	**67.2**

Lessons Learned

From this project, I learned that authorship attribution tasks are different than other text classification tasks. Though deep character-level CNNs required a very large number of training instances for good results on other text classification tasks, here the model obtained better results than traditional methods with a comparatively very small number of training instances.

References

Ronan Collobert, Jason Weston, Leon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural Language Processing (almost) from Scratch. J*ournal of Machine Learning Research*, 12:2493–2537.

Yoon Kim. 2014. Convolutional Neural Networks for Sentence Classification. *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1746–1751.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A Convolutional Neural Network for Modelling Sentences. In *Proceedings of ACL 2014*.

Martin Porter. 1980. An Algorithm for Suffix Stripping. *Program*, 14(3):130–137.

Sebastian Ruder, Parsa Ghaffari, and John G. Breslin. 2016. Character-level and Multi-channel Convolutional Neural Networks for Large-scale Authorship Attribution. *arXiv:1609.06686*.

Jonathan Schler, Moshe Koppel, Shlomo Argamon, and James Pennebaker. 2006. Effects of Age and Gender on Blogging. *AAAI Spring Symposium: Computational Approaches for Analyzing Weblogs*, pages 199–205.

Yanir Seroussi, Ingrid Zukerman, and Fabian Bohnert. 2011. Authorship Attribution with Latent Dirichlet Allocation. *Proceedings of the Fifteenth Conference on Computational Natural Language Learning*, pages 181–189. Association for Computational Linguistics.

Richard Socher, Alex Perelygin, Jean Y. Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. 2013. Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank. In *Proceedings of EMNLP 2013*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level Convolutional Networks for Text Classification. *Advances in Neural Information Processing Systems*, pages 649–657.