Paper 8 | Character-level Convolutional Networks for Text Classification by Xiang Zhang et. al – published by 4th April 2016

Abstract

1. Empirical exploration on the use of character-level CNN for text classification
2. Comparisons against traditional models such as BoW, n-grams and their TFIDF variants, and deep learning models such as word-based CNN and RNN (LSTM)

Introduction

1. Text classification involves assigning predefined categories to free-text documents
2. Time-delay networks used in early days of deep learning are essentially convolutional networks that model sequential data
3. This paper is the first to apply CNN only on characters. The paper shows that when trained on large-scale datasets, deep CNN do not require the knowledge of words, in addition to the conclusion that CNN do not require the knowledge about the syntactic or semantic structure of a language
4. Working only on characters also has the advantage that misspellings and emoticons may be naturally learnt

Character-level CNN

1. The main component is the temporal convolutional module, which simply computes a 1-D convolution. The temporal max-pooling allows us to train deeper models. The non-linearity used in the model is the rectifier or thresholding function, which is similar to ReLUs. The algo used is SGD with a minibatch of 128 using momentum 0.9 and initial step size of 0.01, which is halved every 3 epoches for 10 times
2. Character embedding consists of 70 characters, including 26 English characters, 10 digits, 33 other characters and the new line character
3. Model design
   - The paper designed 2 CNN – one large and one small. Both have 9 layers with 6 convolutional layers and 3 fully-connected layers

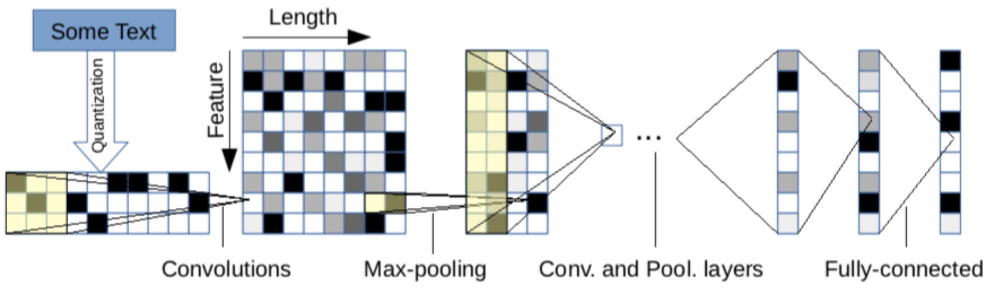

Figure 1: Illustration of our model

   - There are 2 dropout modules in between the 3 fully-connected layers to regularise, with 0.5 probability

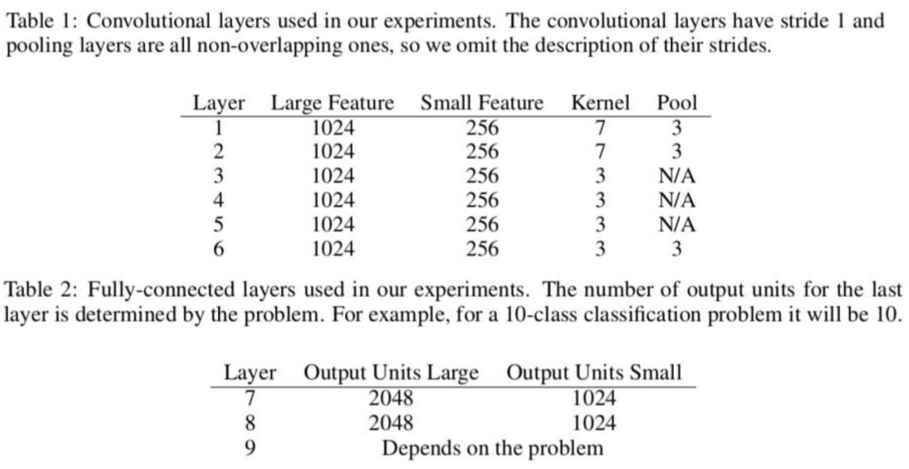

Table 1: Convolutional layers used in our experiments. The convolutional layers have stride 1 and pooling layers are all non-overlapping ones, so we omit the description of their strides.

Layer	Large Feature	Small Feature	Kernel	Pool
1	1024	256	7	3
2	1024	256	7	3
3	1024	256	3	N/A
4	1024	256	3	N/A
5	1024	256	3	N/A
6	1024	256	3	3

Table 2: Fully-connected layers used in our experiments. The number of output units for the last layer is determined by the problem. For example, for a 10-class classification problem it will be 10.

Layer	Output Units Large	Output Units Small
7	2048	1024
8	2048	1024
9	Depends on the problem	

- Initialise the weights using a Gaussian distribution. The mean and standard deviation used for initialising the large model is (0, 0.02) and a small model (0, 0.05)

4. Data augmentation using Thesaurus
   - Useful for controlling generalisation error for deep learning models
   - The best way to do this is to use human rephrases of sentences but this is unrealistic and expensive. Therefore, an alternative is to replace words/phrases with their synonyms – use **English thesaurus**, where every synonym to a word or phrase is ranked by the semantic closeness to the most frequently seen meaning
5. Choice of alphabet
   - Whether to distinguish between upper-case and lower-case letters
   - The paper experimented on this choice and observed that it usually gives worse results when such distinction is made. One possible reason might be that semantics do not change with different letter cases, there is a benefit of regularisation

Model comparisons and Results

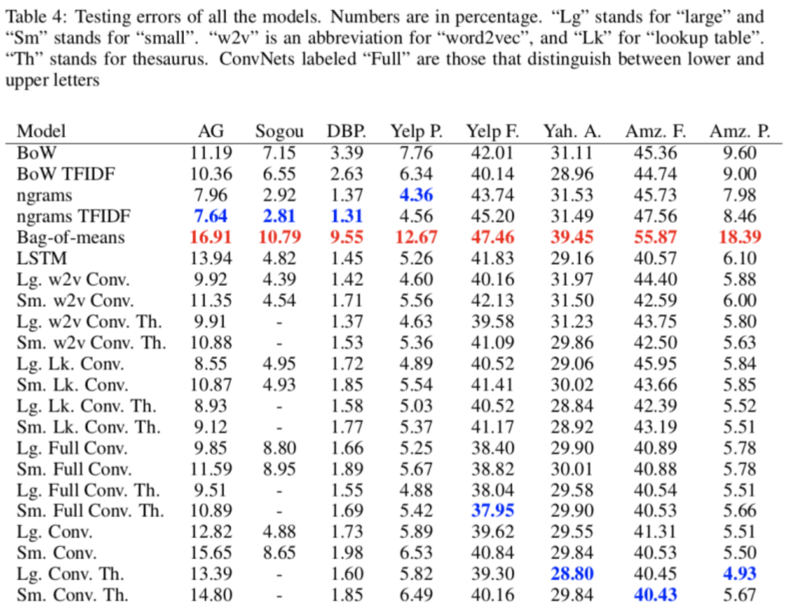

Table 4: Testing errors of all the models. Numbers are in percentage. "Lg" stands for "large" and "Sm" stands for "small". "w2v" is an abbreviation for "word2vec", and "Lk" for "lookup table". "Th" stands for thesaurus. ConvNets labeled "Full" are those that distinguish between lower and upper letters

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

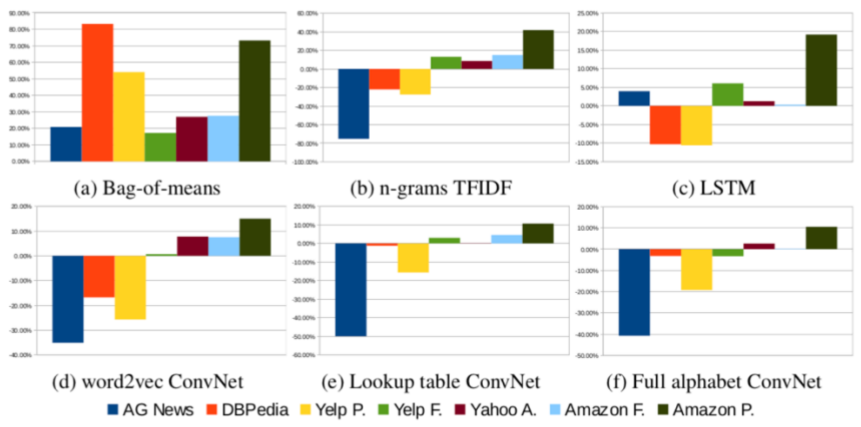

Figure 3: Relative errors with comparison models

1. All CNN in the figure are the large models with thesaurus augmentation
2. **Character-level CNN is an effective method**
   - Our result shows that the proposed model can work for text classification without the need for words. This is a strong indication that language could also be thought of as a signal no different from any other kind
3. **Dataset size forms a dichotomy between traditional and CNN models**
   - Larger datasets tend to perform better
   - Traditional methods like n-gram TFIDF remain a strong model for dataset of size up to several hundreds of thousands and only until the dataset goes to scale of several millions do we observe that character-level CNN start to do better
4. **CNN may work well for user-generated data**
   - User-generated data vary in the degree of how well the texts are curated
   - Figures 3c, 3d, and 3e show that character-level CNN work better for less curated user-generated texts
   - **Further validation is needed to validate the hypothesis that CNN are truly good at identifying misspellings and emoticons**
5. **Choice of alphabet makes a difference**
   - Figure 3f shows that distinguishing between uppercase and lowercase letters could make a difference. For million-scale datasets, it seems that not making such distinction usually works better
6. **Semantics of tasks may not matter**
   - The datasets consist of two kinds of tasks: sentiment analysis and topic classification. This dichotomy in task semantics does not seem to play a role in deciding which method is better
7. **Bag-of-means is a misuse of word2vec**
   - This model performed the worse in every case. This suggests that a simple use of a distributed word representation may not give us an advantage to text classification

Conclusion

1. On one hand, the analysis shows that character-level CNN is an effective method
2. However, how well our model performs in comparisons depends on many factors, such as dataset size, whether the texts are curated and choice of alphabet