Sentence Embeddings Using Korean Corpora

Paper review : Character-level Convolutional Networks for Text Classification

1. Introduction

Character-level Convolutional Networks for Text Classification*

Xiang Zhang **Junbo Zhao** **Yann LeCun**
Courant Institute of Mathematical Sciences, New York University
719 Broadway, 12th Floor, New York, NY 10003
{xiang, junbo.zhao, yann}@cs.nyu.edu

Abstract

This article offers an empirical exploration on the use of character-level convolutional networks (ConvNets) for text classification. We constructed several large-scale datasets to show that character-level convolutional networks could achieve state-of-the-art or competitive results. Comparisons are offered against traditional models such as bag of words, n-grams and their TFIDF variants, and deep learning models such as word-based ConvNets and recurrent neural networks.

- 2015년 발표된 문자 단위의 뉴럴넷(for text classification)
- 일반적으로 text classification task 수행시 분석의 단위는 단어(word)이나, 이를 탈피
  - Word 기준 tokenization은 단어의 semantic / synthetic 측면의 의미를 담고 있는 단위가 word이기 때문
  - 해당 논문은 이러한 통념을 깨고, character-level convolution network 제안
  - SOTA 달성은 실패, 성능은 준수
- Misspelling & emoticon 등 word-level tokenization에서는 “비정상 문자 조합”으로 인식되는 사항들도 자동적으로 학습될 수 있음

2. Character-level Convolutional Networks

2.1. Key Moduls

discrete한 input 함수 $g(x) \in [1, l] \rightarrow \mathbb{R}$

discrete한 kenel 함수 $f(x) \in [1, k] \rightarrow \mathbb{R}$

Convolution $h(y) \in [1, \lfloor (l-k)/d \rfloor + 1]$

⬇ Conovlution은 모든 input에 대해 kenel이 모두 계산

$$h(y) = \sum_{x=1}^{k} f(x) \cdot g(y \cdot d - x + c)$$

⬇ Max pooling

$$h(y) = \max_{x=1}^{k} g(y \cdot d - x + c)$$

- 2015년 발표된 문자 단위의 뉴럴넷(for text classification)
- 일반적으로 text classification task 수행시 분석의 단위는 단어(word)이나, 이를 탈피
  - Word 기준 tokenization은 단어의 semantic / synthetic 측면의 의미를 담고 있는 단위가 word이기 때문
  - 해당 논문은 이러한 통념을 깨고, character-level convolution network 제안
  - SOTA 달성은 실패, 성능은 준수
- Misspelling & emoticon 등 word-level tokenization에서는 “비정상 문자 조합”으로 인식되는 사항들도 자동적으로 학습될 수 있음

2. Character-level Convolutional Networks

2.2. Character Quantization

문자 단위로 tokenization 진행 단계에서 사용한 문자는 총 70가지

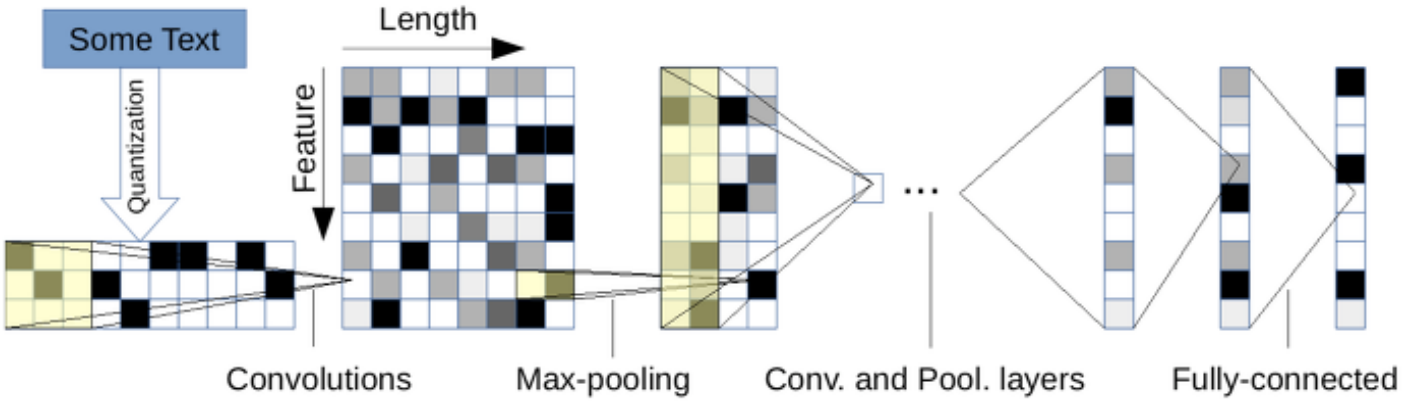
- 알파벳(26) : a,b,c,…,z
- 숫자(10) : 0,1,2,…,9
- 특수문자(33) : -,;.!?:’’’/₩|_@#$%ˆ&*˜‘+-=<>() [] {}
- 개행문자(1)

- 문자 단위 tokenization시 70개 문자를 대상으로 one-hot encoding 실시
  - 미포함 문자는? Zero vector
  - 알파벳 소문자와 대문자를 구분한 모델과 비교 예정

2. Character-level Convolutional Networks

2.3. Model Design

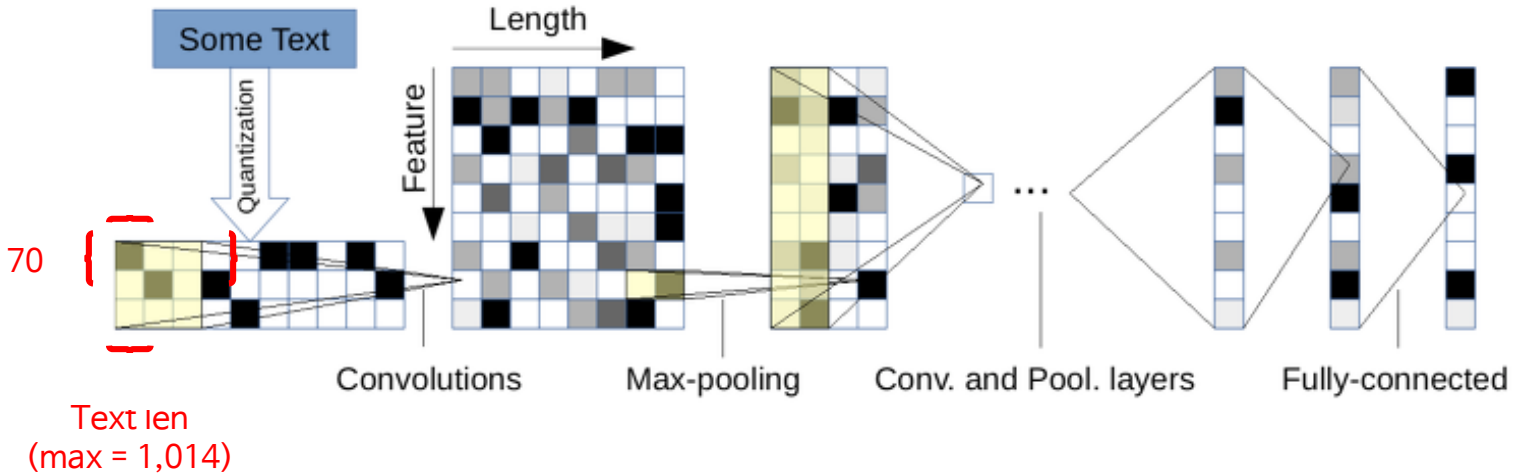
We designed 2 ConvNets – one large and one small. They are both 9 layers deep with 6 convolutional layers and 3 fully-connected layers. Figure 1 gives an illustration.



2. Character-level Convolutional Networks

2.3. Model Design

We designed 2 ConvNets – one large and one small. They are both 9 layers deep with 6 convolutional layers and 3 fully-connected layers. Figure 1 gives an illustration.

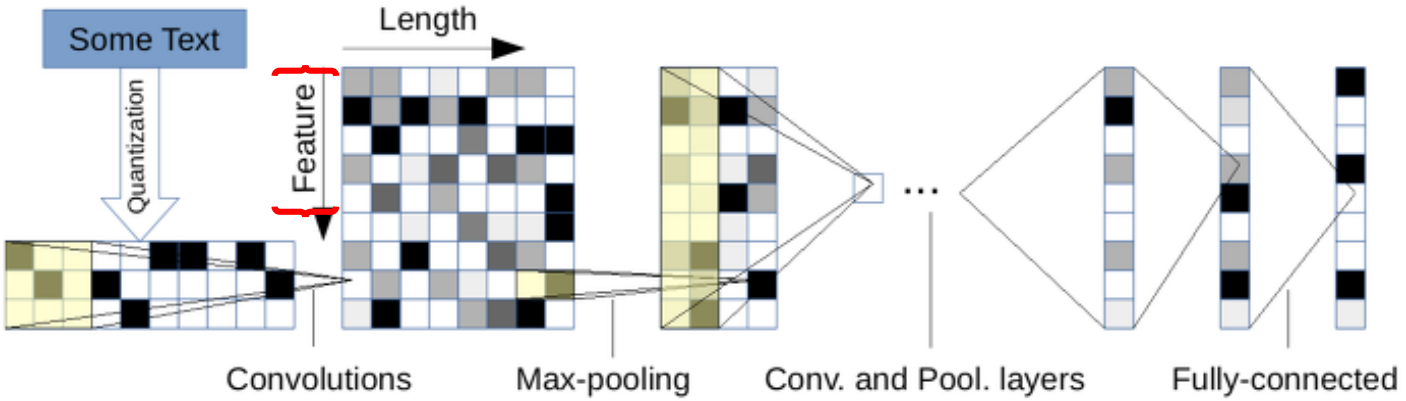


The input have number of features equal to 70 due to our character quantization method, and the input feature length is 1014. It seems that 1014 characters could already capture most of the texts of interest. We also insert 2 dropout [10] modules in between the 3 fully-connected layers to regularize.

2. Character-level Convolutional Networks

2.3. Model Design

We designed 2 ConvNets – one large and one small. They are both 9 layers deep with 6 convolutional layers and 3 fully-connected layers. Figure 1 gives an illustration.



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

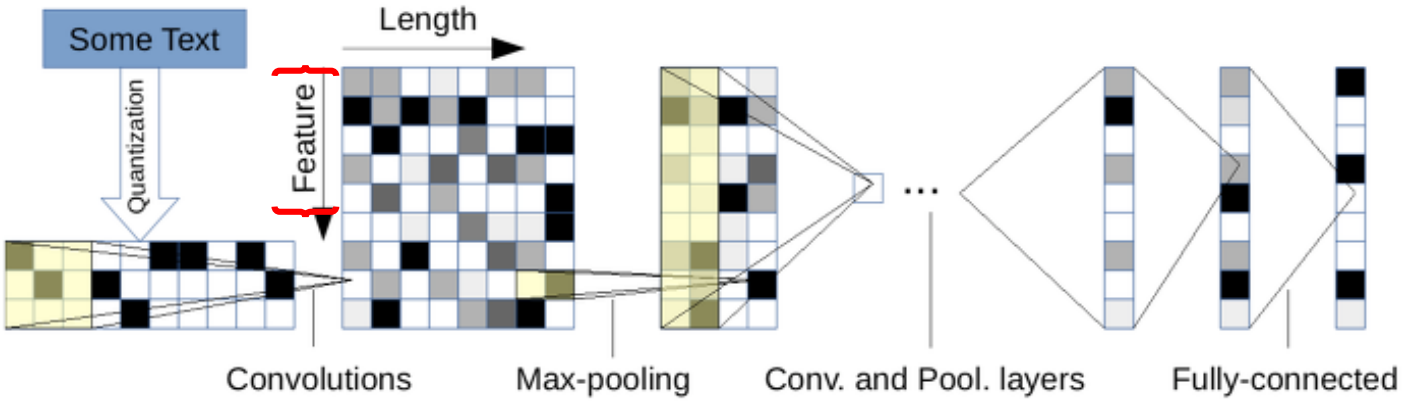
6개 각각의 convolution layer에서 1,024의 feature 수와 256의 feature 수를 가지도록 conv. 연산 수행

How? Filter size를 다르게 해서.

2. Character-level Convolutional Networks

2.3. Model Design

We designed 2 ConvNets – one large and one small. They are both 9 layers deep with 6 convolutional layers and 3 fully-connected layers. Figure 1 gives an illustration.



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

Classification category에 맞추어 설정
Binary problem -> 2

3개의 fully-connected layer 사이에 확률 0.5의 dropout을 2번 사용

Layer 가중치의 초기값은 Gaussian distribution 사용, 평균/분산은 large model과 small model 각각 (0, 0.02), (0, 0.05)

3. Comparison Models & Datasets

Traditional Methods

- ✓ Bag-of-words and its TFIDF — 최빈 5만개 단어를 / 출현 빈도수로 임베딩 & TFIDF 임베딩 / 이후 최대 출현빈도로 나누어 정규화
- ✓ Bag-of-ngrams and its TFIDF — 최빈 50만개의 n-gram(up to 5-grams)을 / 상동 / 상동
- ✓ Bag-of-means on word embedding — 각 데이터셋을 W2V 임베딩 / text를 k-means 클러스터링(k=5,000) / 빈도수 기반 bow, 정규화

Deep learning Methods

- ✓ Word-based ConvNets. — W2V 임베딩 벡터(word based)를 통해 학습
- ✓ Long-short term memory — W2V 임베딩 벡터(word based)를 통해 학습

Dataset	Classes	Train Samples	Test Samples	Epoch Size	
AG's News	4	120,000	7,600	5,000	Relatively small dataset
Sogou News	5	450,000	60,000	5,000	
DBPedia	14	560,000	70,000	5,000	
Yelp Review Polarity	2	560,000	38,000	5,000	Relatively large dataset
Yelp Review Full	5	650,000	50,000	5,000	
Yahoo! Answers	10	1,400,000	60,000	10,000	
Amazon Review Full	5	3,000,000	650,000	30,000	
Amazon Review Polarity	2	3,600,000	400,000	30,000	

4. Large-scale Datasets 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

best result in blue and worse result in red.

IMPLICATIONS

- ✓ 글자 단위의 Convolutional Network도 문서 분류에서 높은 성능을 보인다.
- ✓ 작은 데이터셋에서는 전통적인 NLP방식이 DL방식보다 더 높은 성능을 보인다.
- ✓ ConvNet은 사용자가 만든 데이터에서 좋다. (오타를 잘 잡는다)
- ✓ Alphabet의 선택에 따라 성능이 많이 달라진다. (구분시 일반적으로 worse performance)
- ✓ Bag-of-means 모델은 안 좋다.
- ✓ 모든 데이터셋에 있어 최적의 모델은 없다. (많은 실험을 통해 데이터셋에 가장 적합한 모델을 찾아야 한다)
- ✓ Task의 semantic 정보의 유용성에 대한 의문. Character level로도 좋은데?

참고: Reniew's blog, https://reniew.github.io/29/