Character-level Convolutional Networks for Text Classification

Xiang Zhang, Junbo Zhao, Yann LeCun

(Submitted on 4 Sep 2015 (v1), last revised 4 Apr 2016 (this version, v3))

PyCon(파이콘)은 세계 각국의 파이썬 프로그래밍 언어 커뮤니티에서 주관하는 비영리 컨퍼런스입니다.

파이썬 마을을 시작으로 한 한국 파이썬 커뮤니티는 벌써 그 역사가 15년이나 되었지만, 한국 파이썬 사용자들을 위한 파이콘은 올해 처음으로 열리게 되었습니다. 본 컨퍼런스를 준비/운영하는 파이콘 한국팀은 건강한 국내 파이썬 생태계에 보탬이 되고자 커뮤니티 멤버들의 자발적인 봉사로 운영되고 있습니다.

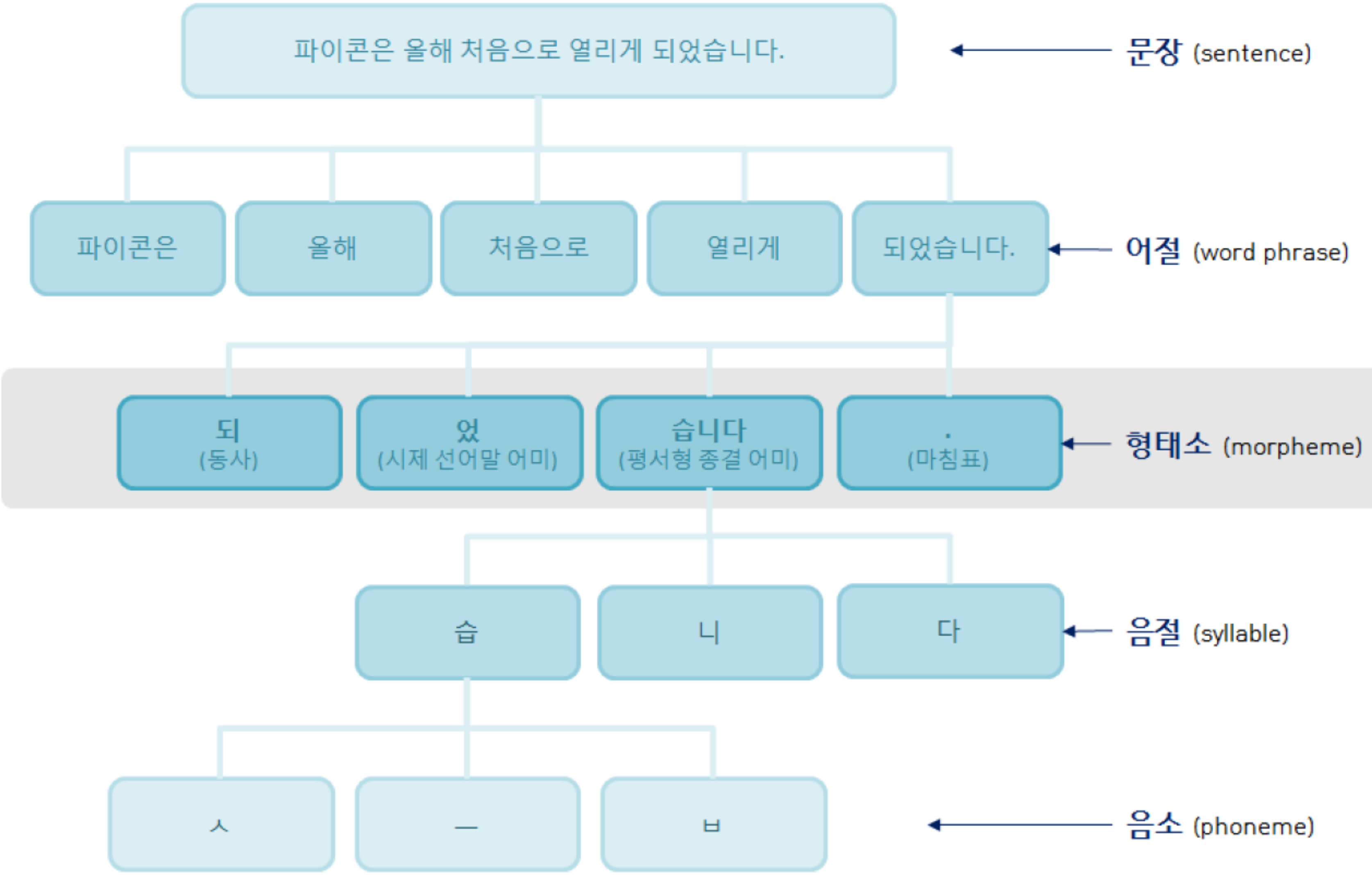
올해 처음으로 열리는 '파이콘 한국'을 통해 새로운 기술과 정보를 공유하고 참석자들이 서로 교류할 수 있는 대표적인 행사가 되기를 희망합니다.

문서 (document)

문단 (paragraph)

문장 (sentence)

음소-레벨 컨볼루션 네트워크를 사용한 문서,문단,문장(?) 분류



Abstract

This article offers an empirical exploration on the use of character-level convolutional networks (ConvNets) for text classification. We constructed several largescale 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**.

- without the knowledge of words, phrases, sentences and any other syntactic or semantic structures

6 Conclusion and Outlook

This article offers an empirical study on character-level convolutional networks for text classification.

We compared with a large number of traditional and deep learning models using several largescale datasets. On one hand, analysis shows that **character-level ConvNet is an effective method.**

On the other hand, how well our model performs in comparisons depends on many factors, such as **dataset size**, whether the **texts are curated** and **choice of alphabet**.

In the future, we hope to apply character-level ConvNets for a broader range of language processing tasks especially when structured outputs are needed.

1 Introduction

"1","Seven Georgian soldiers wounded as South Ossetia ceasefire violated (AFP)",
"AFP - Sporadic gunfire and shelling took place overnight in the disputed Georgian region of South Ossetia in violation of a fragile ceasefire, wounding seven Georgian servicemen."

"2","Schumacher Triumphs as Ferrari Seals Formula One Title",
" BUDAPEST (Reuters) - Michael Schumacher cruised to a record 12th win of the season in the Hungarian Grand Prix on Sunday to hand his Ferrari team a sixth successive constructors' title."

"3","A Personal Operator From Verizon","Verizon plans to offer a service that would act as a virtual switchboard operator, letting customers stay in touch at all times.
The program would send phone calls, voicemails and e-mails wherever customers designate. By Elisa Batista."

"4","Sun's Looking Glass Provides 3D View (PC World)",
"PC World - Developers get early code for new operating system 'skin' still being crafted."

1 - World
2 - Sports
3 - Business
4 - Sci/Tech

1 Introduction

[15] I. Kanaris, K. Kanaris, I. Houvardas, and E. Stamatatos. Words versus **character n-grams** for anti-spam filtering. International Journal on Artificial Intelligence Tools, 16(06):1047–1067, 2007.

[28] C. D. Santos and B. Zadrozny. Learning **character-level** representations for part-of-speech tagging. In Proceedings of the 31st International Conference on Machine Learning (ICML-14), pages 1818–1826, 2014.

[29] Y. Shen, X. He, J. Gao, L. Deng, and G. Mesnil. A latent semantic model with **convolutional-pooling structure for information retrieval.** In Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, pages 101–110. ACM, 2014.

*최근에는 byte 레벨 논문을 쓰심. https://arxiv.org/abs/1802.01817

There are also related works that use character-level features for language processing. These include using character-level n-grams with linear classifiers [15], and incorporating character-level features to ConvNets [28] [29].

This article is the **first to apply ConvNets only on characters.**

2 Character-level Convolutional Networks

In this section, we introduce the design of character-level ConvNets for text classification. The design is modular, where the gradients are obtained by back-propagation [27] to perform optimization.

2.1 Key Modules

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

discrete kernel function $f(x) \in [1, k] \rightarrow \mathbb{R}$

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

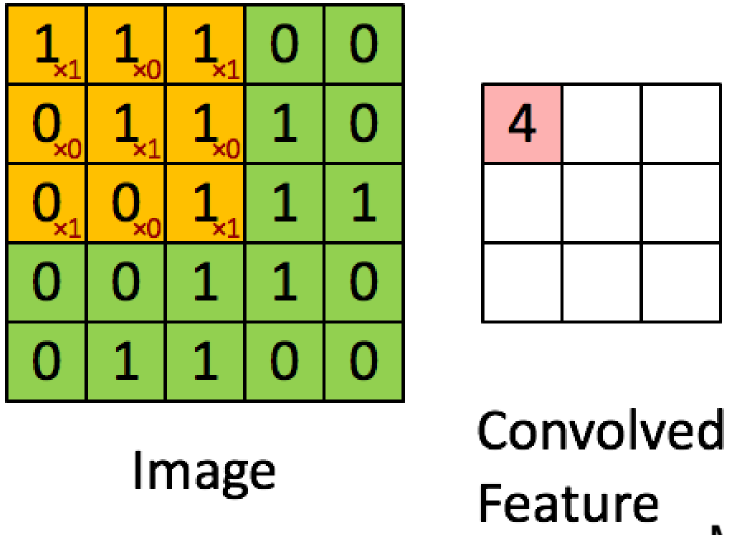
d is stride 1

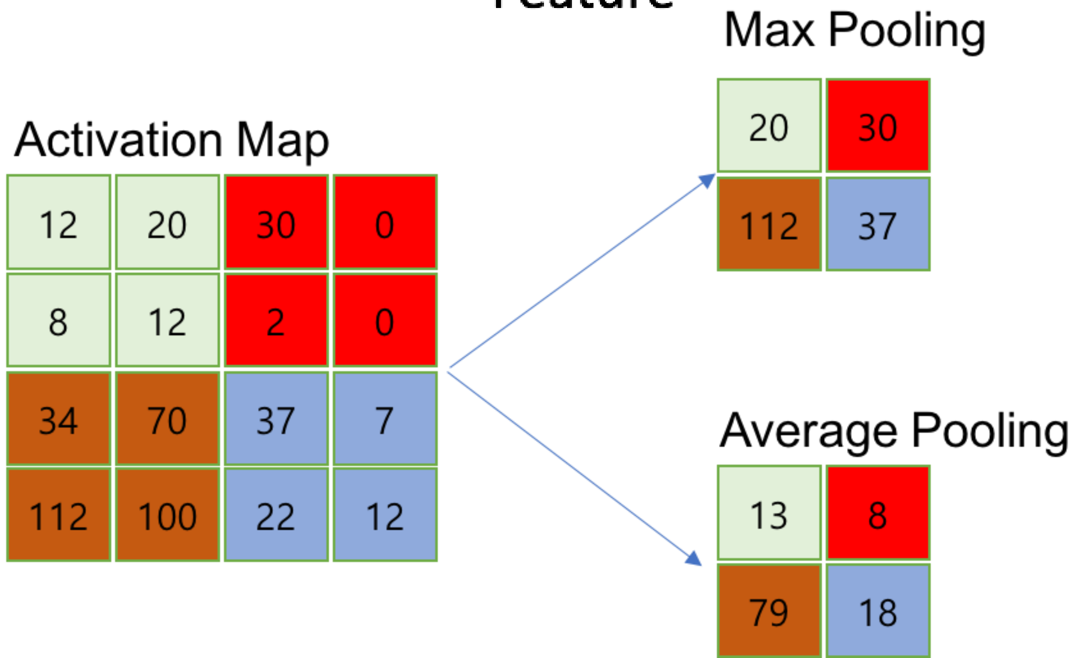
{offset}

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

max-pooling function

where $c = k - d + 1$ is an offset constant.



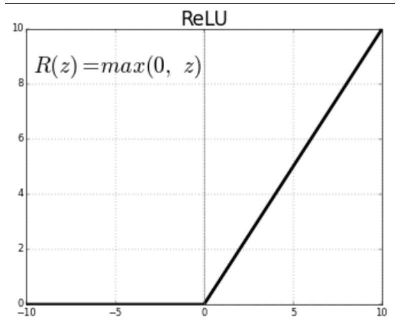


2.1 Key Modules

This very pooling module enabled us to train ConvNets
deeper than 6 layers, where all others fail. The analysis by [3] might shed some light on this.

(mini-batch)에 대해서만 loss function을 계산
-계산속도 빠름
-같은 시간에 더 많은 step을 갈 수 있음
- local minima에 빠지지 않고 더 좋은 방향으로 수렴할 가능

- input feature length is 1014
- 1-D convolution
- 6 layers
- stride 1
- non-linearity : ReLUs, $h(x) = \max\{0, x\}$, stochastic gradient descent (SGD): minibatch of size 128,
- no padding
- Torch 7



The ReLu (Rectified Linear Unit) Layer

ReLu refers to the Rectifier Unit, the most commonly deployed activation function for the outputs of the CNN neurons. Mathematically, it's described as:

$$Eq.3 : max(0, x)$$

Ref : https://blog.xrds.acm.org/2016/06/convolutional-neural-networks-cnns-illustrated-explanation/

2.1 Key Modules

```
# =====================Char CNN========================
# parameter
input_size = 1014
vocab_size = len(tk.word_index)
embedding_size = 69
conv_layers = [[256, 7, 3],
[256, 7, 3],
[256, 3, -1],
[256, 3, -1],
[256, 3, -1],
[256, 3, 3]]

fully_connected_layers = [1024, 1024]
num_of_classes = 4
dropout_p = 0.5
optimizer = 'adam'
loss = 'categorical_crossentropy'
```

Ref : https://github.com/chaitjo/character-level-cnn

2.2 Character quantization

input. : sequence of encoded characters , “one-hot” encoding

l_0 = sized vectors with fixed length(1014)
exceeding length is ignored,
blank -> all-zero vectors

```
abcdefghijklmnopqrstuvwxyz                     : 26 english letters
0123456789                                     : 10 digits
-,;.!?:’ ’ ’ /₩|_@#$%?&*? ‘+-=<>()[]{} : 34 other characters and the new line character
```

a
[1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]

b
[0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]

c
[0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]

&
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]

2.2 Character quantization

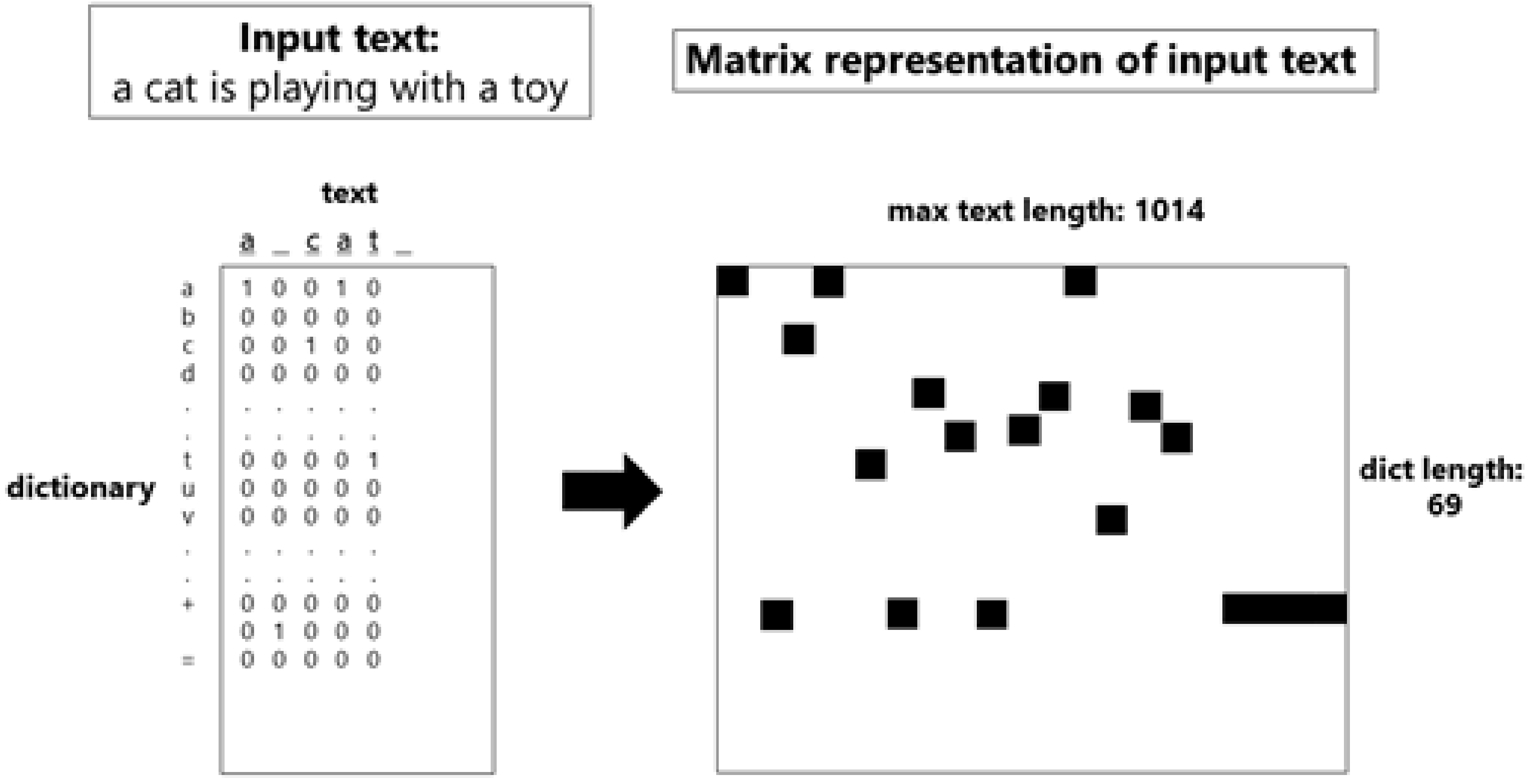


Fig. 1: Scheme of character encoding. Each sentence is encoded as a 69×1014 matrix.

*Ref : https://blogs.technet.microsoft.com/machinelearning/2017/02/13/cloud-scale-text-classification-with-convolutional-neural-networks-on-microsoft-azure/

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.

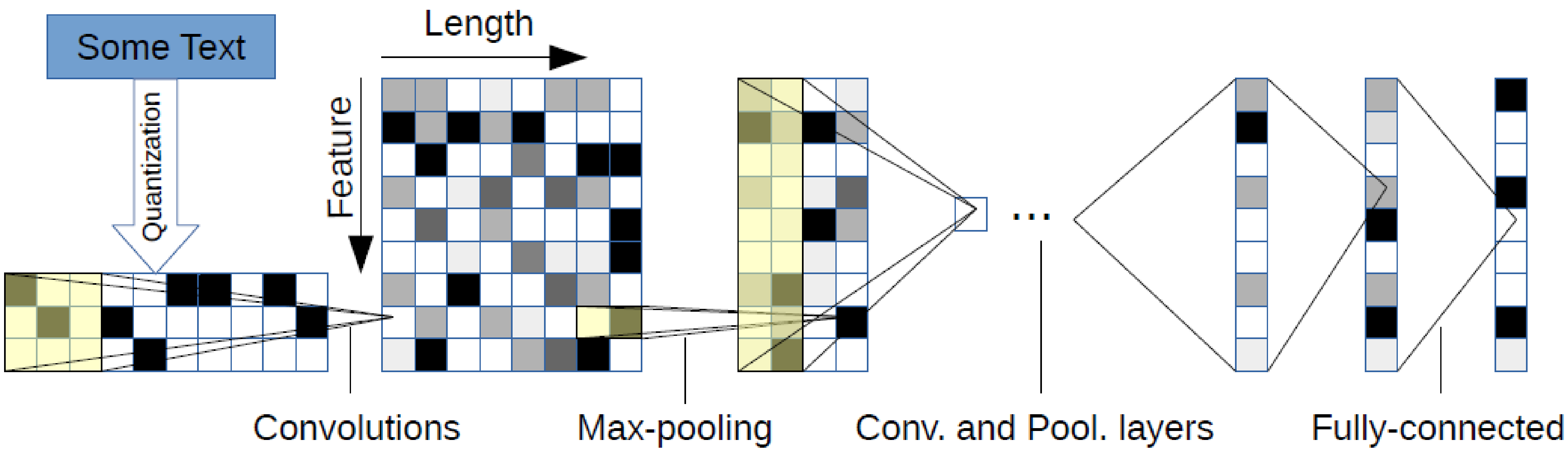


Figure 1: Illustration of our model

- 2 ConvNets – one large and one small.
- They both have 6 convolutional layers and 3 fully-connected layers, with different number of hidden units and frame sizes.

2.3 Model Design

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.

*Kernel Size

*Pooling Size

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

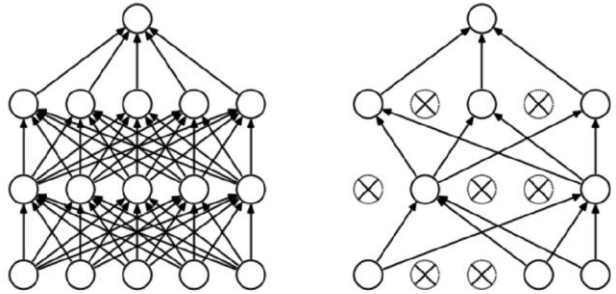
```
# ====================Char CNN======================
# parameter
input_size = 1014
vocab_size = len(tk.word_index)
embedding_size = 69
conv_layers = [[256, 7, 3],
               [256, 7, 3],
               [256, 3, -1],
               [256, 3, -1],
               [256, 3, -1],
               [256, 3, 3]]

fully_connected_layers = [1024, 1024]
num_of_classes = 4
dropout_p = 0.5
optimizer = 'adam'
loss = 'categorical_crossentropy'
```

- large feature와 small feature로 2개의 ConvNet을 구성
- 정규화를 위해 2개의 dropout, dropout의 확률은 0.5
- filter의 stride는 1
- pooling은 non-overlapping

2.3 Model Design

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	



(a) Standard Neural Net

(b) After applying dropout.

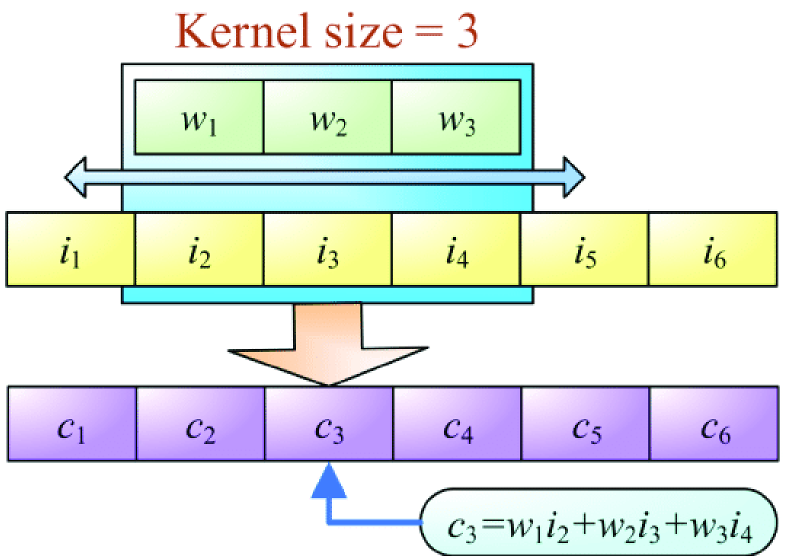
input feature length is 1014.
2 dropout modules[10] in between the 3 fully-connected layers to regularize, with probability of 0.5.

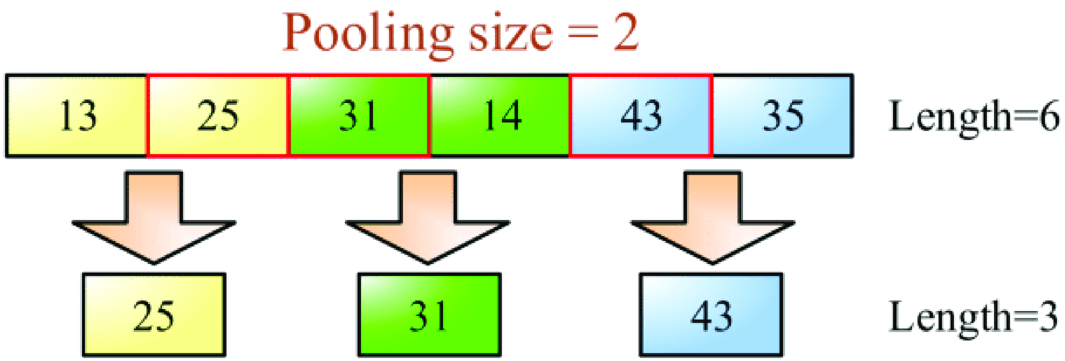
완전연결 레이어

- 1024개,2048 유닛
- 2개의 드롭아웃 적용(0.5) - 학습할 때 레이어의 일부 노드를 제외합니다./여러개의 네트워크를 앙상블하는 효과를 발휘합니다.
- Relu 활성화 함수

(Overfitting 방지)

2.3 Model Design





2.4 Data Augmentation using Thesaurus

We experimented data augmentation by using an **English thesaurus**, which is obtained from the mytheas component used in LibreOffice1 project. That thesaurus in turn was obtained from Word-Net [7], where every synonym to a word or phrase is ranked by the semantic closeness to the most frequently seen meaning.

The data augmentation can do even more magic. It could be designed in a way that adding the word 'not' before a slot value will change its value, which will work for every trained model for every domain.

*데이터 검색을 위한 키워드(색인어)간의 관계, 즉 동의어, 하위어(下位語 : 그 색인어에 속하는 용어), 관련어 등의 관계를 나타낸 사전을 시소러스라고 한다.

Ref : https://chatbotsmagazine.com/how-we-improved-nlp-error-rate-fourfold-and-achieved-94-accuracy-28e7639e8195

http://nmhkahn.github.io/CNN-Practice

3 Comparison Models

To offer fair comparisons to competitive models, we conducted a series of experiments with both traditional and deep learning methods.

We tried our best to choose models that can provide comparable
and competitive results, and the results are reported faithfully without any model selection.

3.1 Traditional Methods

- **Bag-of-words and its TFIDF.**
  상위 빈도 50,000개의 단어들을 가지고 출현수를 단어의 feature로한 bag-of-words와 출현 수 대신 TF-IDF로 한 모델

- **Bag-of-ngrams and its TFIDF.**
  5-grams 까지 중 가장 frequent한 n-gram 500,000개

- **Bag-of-means on word embedding.**
  train data에 word2vec을 사용한 것에 k-means clustering을 하여 분류.
  5회이상 출현한 모든 단어를 고려
  embedding의 dimension은 300
  The **bag-of-means features** are computed the same way as in the bag-of-words model.
  The number of means is 5,000.

The classifier used is a **<u>multinomial logistic regression</u>** in all these models.

3.1 Traditional Methods – TF-IDF(Term Frequency - Inverse Document Frequency)

TF-IDF는 **<u>단어 빈도와 역문서 빈도의 곱</u>**이다. 두 값을 산출하는 방식에는 여러 가지가 있다. **단어 빈도** tf(*t*,*d*)의 경우, 이 값을 산출하는 가장 간단한 방법은 단순히 문서 내에 나타나는 해당 단어의 총 빈도수를 사용하는 것이다. 문서 d 내에서 단어 t의 총 빈도를 f(*t*,*d*)라 할 경우, 가장 단순한 tf 산출 방식은 tf(*t*,*d*) = f(*t*,*d*)로 표현된다. 그 밖에 TF값을 산출하는 방식에는 다음과 같은 것들이 있다.[1]:118

$$\mathrm{tf}(t,d) = 0.5 + \frac{0.5 \times \mathrm{f}(t,d)}{\max\{\mathrm{f}(w,d) : w \in d\}}$$

$$\mathrm{idf}(t,D) = \log \frac{|D|}{|\{d \in D : t \in d\}|}$$

역문서 빈도는 한 단어가 문서 집합 전체에서 얼마나 공통적으로 나타나는지를 나타내는 값이다. 전체 문서의 수를 해당 단어를 포함한 문서의 수로 나눈 뒤 로그를 취하여 얻을 수 있다.

3.1 Traditional Methods – bag of words

The **bag-of-words** model is a simplifying representation used in natural language processing and information retrieval(IR). Also known as the vector space model[1].

In this model, a text (such as a sentence or a document) is represented as the bag(multiset)of its words, disregarding grammar and even word order but keeping multiplicity. The bag-of-words model has also been used for computer vision.[2]

The bag-of-words model is commonly used in methods of document classification where the (frequency of) occurrence of each word is used as a feature for training a classifier[3].

3.1 Traditional Methods

(1) John **likes** to watch **movies**. Mary **likes movies** too.
(2) John also likes to watch football games.

```
{ "John": 0,
  "likes": 1,
  "to": 2,
  "watch": 3,
  "movies": 4,
  "also": 5,
  "football": 6,
  "games": 7,
  "Mary": 8,
  "too": 9 }
```

Index	0	1	2	3	4	5	6	7	8	9
(1)	1	2	1	1	2	0	0	0	1	1
(2)	1	1	1	1	0	1	1	1	0	0

(1)John likes likes to watch movies movies Mary too
(2)John likes to watch also football games

Ref : https://en.wikipedia.org/wiki/Bag-of-words_model
http://darkpgmr.tistory.com/125
http://pythonkim.tistory.com/92

3.1 Traditional Methods – Bag-of-ngrams

Examples [edit]

Figure 1 *n*-gram examples from various disciplines

Field	Unit	Sample sequence	1-gram sequence	2-gram sequence	3-gram sequence
Vernacular name			**unigram**	**bigram**	**trigram**
Order of resulting Markov model			**0**	**1**	**2**
Protein sequencing	amino acid	... Cys-Gly-Leu-Ser-Trp ...	..., Cys, Gly, Leu, Ser, Trp, ...	..., Cys-Gly, Gly-Leu, Leu-Ser, Ser-Trp, ...	..., Cys-Gly-Leu, Gly-Leu-Ser, Leu-Ser-Trp, ...
DNA sequencing	base pair	...AGCTTCGA...	..., A, G, C, T, T, C, G, A, ...	..., AG, GC, CT, TT, TC, CG, GA, ...	..., AGC, GCT, CTT, TTC, TCG, CGA, ...
Computational linguistics	character	... to_be_or_not_to_be...	..., t, o, _, b, e, _, o, r, _, n, o, t, _, t, o, _, b, e, ...	..., to, o_, _b, be, e_, _o, or, r_, _n, no, ot, t_, _t, to, o_, _b, be, ...	..., to_, o_b, _be, be_, e_o, _or, or_, r_n, _no, not, ot_, t_t, _to, to_, o_b, _be, ...
Computational linguistics	word	... to be or not to be ...	..., to, be, or, not, to, be, ...	..., to be, be or, or not, not to, to be, ...	..., to be or, be or not, or not to, not to be, ...

The bag-of-ngrams models are constructed by selecting the **500,000** most frequent n-grams (**up to 5-grams**) from the training subset for each dataset.

The feature values are computed the same way as in the bag-of-words model.

3.1 Traditional Methods – Bag-of-Means

We also have an experimental model that **uses k-means on word2vec** [23] learnt from the training subset of each dataset, and then use these learnt means as representatives of the clustered words.

We take into consideration all the words that appeared more than **5 times** in the training subset.

The dimension of the **embedding is 300**.

The bag-of-means features are computed the same way as in the bag-of-words model. The number of **means is 5000**.

3.2 Deep Learning Methods

We choose two simple and representative models for comparison,
in which one is **word-based ConvNet** and the other a simple **long-short term memory (LSTM)** [11] recurrent neural network model.

We offer comparisons with both using the pretrained word2vec [23] embedding [16] and using lookup tables[5].

3.3 Choice of Alphabet

For the alphabet of English, one apparent choice is whether to distinguish between **upper-case** and **lower-case** letters.

We report experiments on this choice and observed that it usually (but not always) **gives worse results** when such distinction is made.

One possible explanation might be that semantics
do not change with different letter cases, therefore there is a benefit of regularization.

알파벳 대/소문자 구분해보았으나 **성능이 더 좋지 않았다**.

이 경우에 대/소문자에 따라 semantic이 변하지 않아 regularization으로 작용하지 않았나 추측한다.

4 Large-scale Datasets and Results

Table 3: Statistics of our large-scale datasets. Epoch size is the number of minibatches in one epoch

Dataset	Classes	Train Samples	Test Samples	Epoch Size
AG's News	4	120,000	7,600	5,000
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
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

학습반복 횟수

4 Large-scale Datasets and Results - Dataset size increases from left to right

Model	120,000/4 AG	450,000/5 Sogou	560,000/14 DBP.	560,000/2 Yelp P.	650,000/5 Yelp F.	1,400,000/10 Yah. A.	3,000,000/10 Amz. F.	3,600,000/2 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

- "Lg" stands for "large"
- "Sm" stands for "small"
- "w2v" is an abbreviation for "word2vec",
- "Lk" for "lookup table"
- "Th" stands for thesaurus

- best result in blue and worse result in red.

user-generated text

5 Discussion There is no free lunch : 모든 경우에 뛰어난 방법은 없었다. 적용에 참고해라.

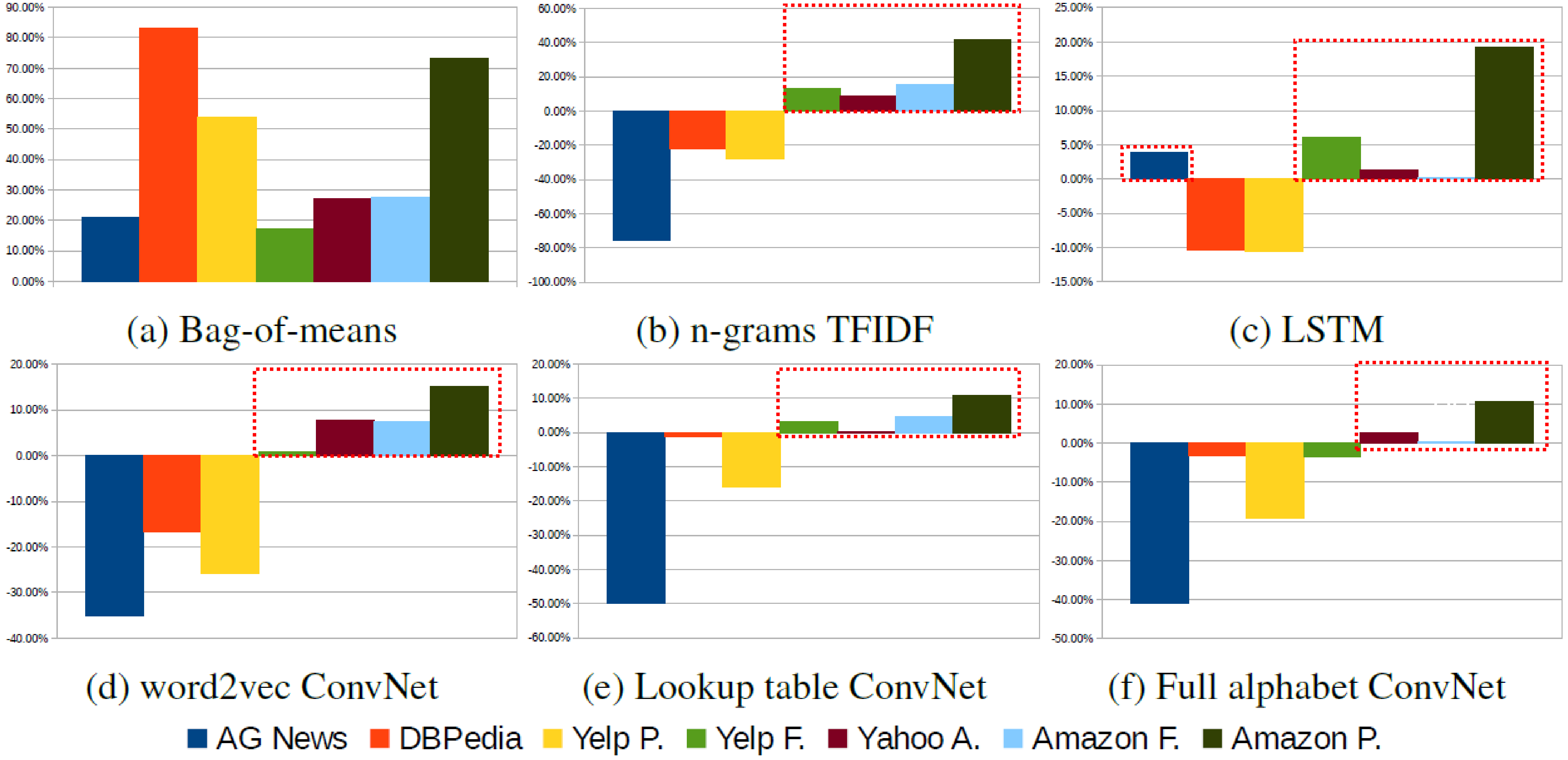


Figure 3: Relative errors with comparison models

감사합니다.