FastText 와 BERT 를 이용한 자동 용어 추출

최규현, 나승훈
전북대학교 인지컴퓨팅 연구실
Chlrbgus321@naver.com, nash@jbnu.ac.kr

FastText and BERT for Automatic Term Extraction

Kyu-Hyun Choi, Seung-Hoon Na
Jeonbuk National University, Cognitive computing Laboratory

요 약

자연어 처리의 다양한 task 들을 잘 수행하기 위해서 텍스트 내에서 적절한 용어를 골라내는 것은 중요하다. 텍스트에서 적절한 용어들을 자동으로 추출하기 위해 다양한 모델들을 학습시켜 용어의 특성을 잘 반영하는 n 그램을 추출할 수 있다. 본 연구에서는 기존에 존재하는 신경망 모델들을 조합하여 자동 용어 추출 성능을 개선할 수 있는 방법들을 제시하고 각각의 결과들을 비교한다.

주제어: FastText, BERT, term extraction, fusion

1. 서론

자연어 처리의 task 를 수행하는데 있어 고품질의 데이터셋을 준비하는 것은 중요하다. 개체명 인식(Named Entity Recognition), 정보추출(Information Extraction), 텍스트 요약(Text summarization), 유의어 탐색(Synonym Discovery)등의 다양한 task 는 전체 문장 중 핵심이 되는 용어를 추출할 것을 요구하는데 방대한 양의 데이터에서 사람의 손으로 직접 용어들을 선정하는 것은 많은 시간과 인력이 낭비 된다. 따라서 용어를 자동으로 추출하는 기술은 자연어 처리 task 에서 중요한 도구가 된다.

자동 용어 추출은 컴퓨터 알고리즘을 통해 텍스트를 읽어 들여 용어라고 판단되는 단어를 자동으로 필터링 하는 기술이다. 룰 베이스 프로그램이나 머신러닝을 활용하여 구현할 수 있다.

용어는 텍스트에서 중요한 역할을 하는 단어를 말한다. 전문분야에서는 그 분야의 도메인에 우세하게 등장하는 단어들이 전문용어의 역할을 할 수 있고 비 전문적인 텍스트에서는 텍스트 문서 내에서 중요한 역할을 하는 단어가 텍스트의 용어 역할을 할 수 있다.

인공신경망을 사용하여 지도학습을 수행할 경우 어떤 특성을 가지는 단어를 용어로 설정하느냐에 따라 그 특성을 가지는 단어를 용어로서 인식하도록 신경망을 학습시킬 수 있다. 신경망을 사용해 언어 모델을 구축할 경우 부산물로 다차원으로 구성된 임베딩 벡터가 형성되는데 자연어처리 분야에서는 이 벡터를 단어 표현(Word representation)[1]이라고 부르기도 한다. 단어 표현은 주변단어의 빈도수 또는 앞 뒤 문맥을 반영해 형성되는데 벡터의 각 차원들은 단어 표현의 분포적, 문맥적 특성을 반영한다. 용어에 해당하는 단어들을 대표하는 벡터를 형성하여 단어 하나하나의 특성이 아닌 용어라는 문장 성분으로서의 특성을 나타내게 할 수 있다. 이 벡터와 유사한 단어를 용어로 간주하는 방식으로 텍스트에서 자동 용어 추출을 진행할 수 있다. 기존에 존재하는 TF-IDF 방식을 사용하여 용어추출을 진행할 수 있고 FastText, BERT 등의 모델을 사용해 같은 과정으로 용어추출을 진행할 수 있지만 본 논문에서 FastText 에 FFNN(Fully Connected Neural Network)을 쌓아서 추가적으로 Binary Classification 을 수행하는 방법과 BERT 의 문맥지향적 단어표현에 FastText 의 분포지향적 단어표현의 특성을 융합시킬 수 있는 모델을 통해 자동 용어 추출의 성능을 향상시킬 수 있는 방법을 소개하려 한다.

....

2. 관련 연구

고전적인 자동 용어 추출 방식으로 TF-IDF 방식이 사용되어왔다. TF-IDF 방식은 문서 내 용어빈도 TF(Term frequency), 역문서 빈도 IDF(Invserse document frequency)를 구해 두 값을 곱하는 방식으로 텍스트에서 중요한 단어를 추출할 수 있게 해주는 방식이다. 문서 내에서 자주 등장하는 단어는 중요한 단어로 간주할 수 있는데 TF 가 이를 반영한다. 영문 텍스트에서 a, the, he, she 같은 단어들은 문서 내에 자주 등장 하는 단어들이지만 텍스트 내에서 유의미한 단어라고 할 수는 없다. 여러 문서들을 비교 후 여러 문서에서 공통으로 다수 등장하는 단어들은 특정 문서에서 등장하는

단어들에 비해 비교적 중요도가 떨어진다고 간주할 수 있다. 이런 단어들에 패널티를 주기 위해 IDF 를 계산한다. TF 와 IDF 를 곱하면 두 가지 특성을 반영하는 TF-IDF 값을 얻을 수 있다.

Evans 와 Lefferts 가 Clarit-trec-experiments 논문에서 TF-IDF 방식을 사용하였다.

최근에 Suman외 1명의 연구에서 TF(term frequency)계산이 Domain Relevance(DR), Domain Consensus(DC), Lexical Cohesion(LC) 등을 사용한 용어 추출 방식[2]에 응용되었다.

언어 모델을 구축하기 위해 사용하는 대표적인 모델 중 Word2Vec[3], FastText[4], BERT[5] 등이 있다. Word2Vec 은 주변 단어들을 사용해 중심단어를 예측하는 CBOW 방식과 중심 단어를 사용해 주변 단어들을 예측하는 skip-gram 방식이 있다. FastText 는 Word2Vec 방식을 상속받아 Word 를 Subword 로 쪼개는 방식을 추가해 OOV 나 Rare Word 에 대응할 수 있게 한 모델이다. Word2Vec 과 FastText 모두 주변 단어의 빈도수를 고려하는 분포가설에 기반을 두는 반면 BERT 는 RNN 이나 LSTM 처럼 이전 문맥을 반영해 언어모델을 구축할 수 있으나 BERT 는 앞뒤 문맥을 모두 고려하여 언어모델을 구축할 수 있다. 언어모델들은 훈련의 부산물로 다차원으로 구성된 임베딩 백터를 형성하는데 자연어처리 분야에서 이 벡터를 단어 표현이라고 부르기도 한다. 단어 표현의 각 차원들은 단어의 의미를 반영하는데 단어 표현은 벡터이므로 단어 표현간 유사도를 계산하는 것이 가능한데 이러한 특성을 이용하여 의미가 유사한 단어들간의 유의어 셋을 만들 수 있다. 이러한 원리를 자동 용어 추출에 응용할 수 있는데 텍스트 내 용어들의 평균을 구하여 용어들을 대표하는 하나의 벡터를 형성할 수 있고 이 벡터는 어떤 특정한 의미를 지닌 단어가 아닌 문장 내 용어들의 공통적인 특성을 나타내므로 문장 전체에 대해서 n 그램을 만들면 각 n 그램들과 용어를 대표하는 벡터 사이의 유사도를 계산하여 용어라고 간주해도 될 만한 n 그램들을 골라 낼 수 있다.

BERT 를 활용한 파생모델 중 Yile Wang 외 2 명이 사용한 BERT 와 skip-gram 을 융합한 모델[6]이 있다. 이 모델은 skip-gram 의 중심단어를 BERT 의 단어표현으로 대체하고 skip-gram 의 주변단어들과 attention product 를 계산한 후 하나의 context 벡터를 형성한다. 중심단어에 해당하는 BERT 단어 표현을 입력, 형성된 context 벡터를 타겟으로 설정하여 테스트 단계에서 BERT 단어 표현을 입력했을 때 context 벡터를 잘 예측하도록 NCE loss(noise contrastive estimation loss)[7]를 사용해 학습을 진행한다.

BERT 의 문맥적 단어 표현과 Word2vec 의 분포적 단어표현의 특성을 융합하려는 시도를 했던 사례가 더 있다.

Qianchu 외 2 명의 연구진들은 BERT 같은 모델을 통해 얻어질 수 있는 contextualized embeddings 을 사용하여 static anchors 를 형성하고 FastText 같은 모델을 통해 얻어질 수 있는 static embeddings 를 정답 레이블 삼아 학습하여 두 가지 종류의 임베딩의 특성을 융합하는 방식[8]을 제안했다.

3. 제안 모델

기존 모델 중 FastText, BERT를 선택하여 이 모델들의 성능을 높이기 위해 두 가지 조합을 시도해 보았다.

하나의 모델은 FastText 와 FFNN 을 융합시킨 모델로 FastText 를 통해 얻어진 단어표현들을 이진분류를 위한 FFNN 모델(Feed forward neural network Model for Binary Classification)을 사용하여 추가학습 하여 용어 추출의 성능 향상에 기여하도록 하였다. 또 다른 모델은 Yile Wang 의 파생모델을 응용하여 BERT 에서 추출된 중심단어를 FastText 의 주변단어들과 attention product 를 거쳐 하나의 context 벡터를 만든 후 BERT 중심단어가 context 벡터를 예측하도록 NCE loss 를 사용하여 학습시켜 기존 BERT 모델의 임베딩이 FastText 임베딩의 특성을 동시에 갖도록 하여 용어 추출 성능 향상에 기여하도록 하였다..

3.1 FastText + Binary Classification Model

주어진 Corpus 의 문장들은 $s = w_1, w_2, \ldots, w_n$의 N 그램 또는 Entity 셋으로 구성되어있다. Entity 셋은 추출할 용어의 특성을 나타내 줄 정답레이블에 해당한다.

그림 1 의 (a)와 같이 모든 Entity 에 대한 단어표현 e_i를 추출하고 모든 Entity 를 대표하는 E를 구한다.

$$\mathrm{E} = \frac{1}{N}\sum_{i=1}^{N} e_i \quad (4)$$

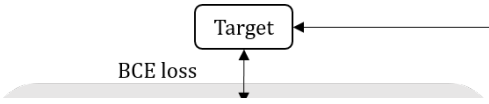
E와의 유사도를 비교하여 0.5 를 넘는 N 그램에는 1 을 0.5 를 넘지 않는 N 그램에는 0 을 태깅하고 FNN 을 사용하여 이진분류 학습을 진행한다. 손실함수로는 BCE loss 를 사용한다.

테스트 단계에서 N 그램들을 입력한 후 순전파 시켜 Sigmoid 함수를 통과시키면 0 과 1 사이의 실수 값을 얻을 수 있는데 값이 1 에 가까울수록 추출하고자 하는 용어에 가까운 단어 표현이라고 할 수 있다.

3.2 BERT + FastText

3.1과 동일한 과정으로 데이터셋을 준비한다. 그림 1의 (c)와 같이 BERT를 사용해 FastText에서 활용할 중심단어의 단어표현u_i를 추출한다. FastText는 BERT에서 추출한 중심단어의 주변단어들 $w_{i-ws}, w_{i+1}, \ldots, w_{i+ws}$의 단어표현들$v'_{i-ws}, v'_{i+1}, \ldots, v'_{i+ws}$을 추출하여 BERT중심단어 w_i의 단어표현u_i와 차례대로 attention distribution을 계산한다.

$$a_j = softmax(\frac{u_i^T v'_j}{\sqrt{d_{v'}}}) \quad (1)$$



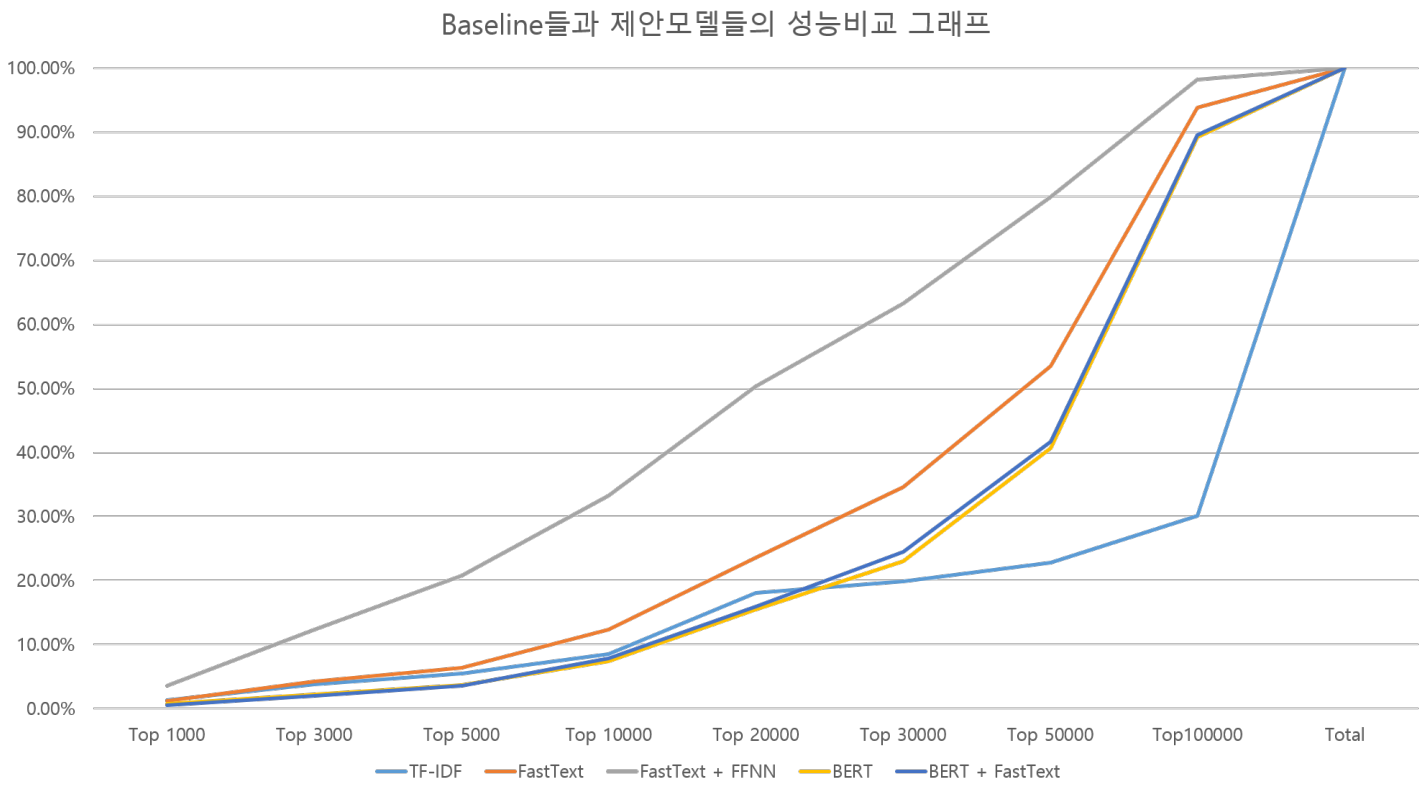


그림 2: Baseline들과 제안모델들의 성능비교 그래프

표 1: Baseline들과 제안모델들의 성능비교 표

	TF-IDF	FastText	FastText + FFNN	BERT	BERT + FastText
Top 1000	1.3022%	1.2593%	3.5646%	0.7898%	0.5550%
Top 3000	3.8233%	4.2049%	12.4013%	2.2839%	2.0491%
Top 5000	5.4406%	6.4461%	20.7257%	3.6926%	3.5859%
Top 10000	8.4968%	12.4013%	33.2764%	7.4280%	7.8762%
Top 20000	18.0283%	23.6286%	50.3735%	15.5176%	15.9658%
Top 30000	19.8835%	34.6425%	63.2657%	23.0736%	24.4610%
Top 50000	22.8148%	53.5326%	79.9360%	40.7044%	41.6649%
Top 100000	30.0809%	93.8527%	98.1857%	89.2423%	89.5838%
Total	100.0000%	100.0000%	100.0000%	100.0000%	100.0000%

각 attention distribution 의 결과값들을 FastText 단어표현들과 곱한 후 전부 더하여 $v'_{i_context}$를 얻는다.

$$v'_{i\_context} = \sum_{1\le|j|\le ws} a_{i+j} v'_{i+j} \quad (2)$$

이렇게 형성한 Context 벡터를 NCE loss 를 사용한다.

$$L = -\sum_{c=1}^{N}\sum_{i=1}^{n_c}(log\sigma(v'^{T}_{i_{context}} u_i) + \sum_{m=1}^{k}\mathbb{E}_{w_{neg_m}\sim P(w)}[log\sigma(-v'^{T}_{neg_m} u_i)]) \quad (3)$$

FFNN 을 통해 학습한 후 테스트 단계에서 BERT 단어표현이 NCE loss 를 통해 형성된 Context 벡터를 예측하도록 한다.

4. 실험

4.1 실험방법

한국어 위키백과에서 POS-tagging 된 10000 문장을 임의로 선별하여 Corpus 로서 준비하였다. 문장 내 일부 단어들은 사람이름, 장소, 기관 등으로 Entity-tagging 되어 있다. 10000 개의 POS-tagging 된 문장들은 형태소 단위로 토큰화 되어 있는데 문장을 구성하는 단어들 중 Entity-tagging 된 단어들은 1 개의 토큰으로 구성된 것부터 9 개로 구성된 것까지 존재한다. 10000 개의 문장 내 존재하는 총 129918 개의 Entity 중 4 개부터 9 개까지의 토큰으로 구성된 Entity 의 개수는 1083 개로 전체 Entity 수의 0.83%에 불과하다. 따라서 불필요한 계산을 피하기 위해 입력으로 사용할 n 그램은 Trigram 까지만 만들었다.

각 모델에 맞게 전처리 과정을 거쳐 학습을 수행하였다. TF-IDF 모델의 경우 완성된 문장들을 투입하고 N 그램 목록을 사용하여 string matching 을 통해 TF-IDF 를 계산하였다. TF-IDF 점수가 큰 순서대로 정렬 후 각 구간 내 N 그램들과 Entity 목록의 교집합의 개수를 산출하고 전체 N 그램 개수로 나누어 구간별 백분율 점수를 산출하였다.

각 모델 별로 용어임을 잘 나타내는 순서로 정렬할 수 있는 점수들을 산출하고 Top1000, Top3000, Top5000, Top10000, Top20000, Top30000, Top50000, Top100000, Total 순서대로 구간을 정해 전체 n 그램 중 용어에 해당하는 n 그램들이 각 구간 내에 잡히는 비율을 측정하였다.

FastText 모델의 경우에는 Entity 목록에 해당 하는 FastText 단어표현들을 추출 후 평균을 구해 Entity 단어표현들을 대표하는 하나의 벡터를 구해 N 그램 단어표현들과의 코사인 유사도를 구하여 유사도가 가장 높은 순으로 위 실험과 같게 구간을 나누고 Entity 포함 비율을 측정하였다.

FastText 모델에 FFNN 이진분류기를 적층한 경우엔 이진분류기의 Sigmoid 함수를 통과한 값을 큰 순서로 정렬하여 각 구간별 Entity 포함 비율을 측정하였다.

BERT 모델의 경우에는 파인튜닝과정 없이 사전학습된 모델 그대로를 사용하여 N 그램의 임베딩을 추출하고 Entity 를 대표하는 하나의 평균 벡터와 코사인 유사도를 측정하여 유사도가 높은 순서대로 정렬하고 각 구간별 Entity 가 포함되는 비율을 측정하였다. BERT 의 사전학습 모델과 토크나이저는 [9]의 것을 사용하였다.

BERT + FastText 모델에서는 최종적으로 NCE 학습된 N 그램 임베딩 그룹의 각 벡터들을 Entity 를 대표하는 평균벡터와 코사인 유사도를 측정하여 유사도가 큰 순서대로 정렬하고 각 구간별 Entity 포함 비율을 측정하였다.

4.2 실험결과

TF-IDF, FastText, BERT 모델들을 BaseLine 으로 설정하고 FastText + Binary Classification 모델은 1Layer 에 드랍아웃[10]을 주지 않았고 BERT + FastText 모델은 1Layer 에 0.1 드랍아웃을 설정하였다.

본 논문이 제안한 모델들과 Baseline 의 세 모델들의 결과에서 정해진 구간 Ngram 대비 용어의 비율을 구하고 서로 서로 비교하였다. TF-IDF 를 제외한 모델들 중에서는 FastText + Binary Classification 이 가장 좋은 성능을 나타냈고 FastText, BERT + FastText, BERT Baseline 이 뒤를 이었다. BERT + FastText 모델에서는 Top 5000 구간 까지는 BERT baseline 이 우세하다가 Top10000 구간 이후부터 BERT + FastText 모델이 역전하는 결과를 보였다. TF-IDF 모델은 Top20000 구간까지는 BERT Baseline 보다 우수한 성능을 보였으나 Top30000 으로 가는 구간에서 BERT 보다 성능이 떨어져 Top100000 구간에서도 30%의 용어밖에 잡아내지 못하는 저조한 성능을 보였다.

5. 결론

FastText 와 FNN 을 사용한 용어추출 모델이 가장 우수한 성능을 보임이 확인되어 용어추출을 위한 태스크에서 FastText 와 FNN 을 사용한 모델을 사용할 것을 제안한다. 아울러 용어추출이 아닌 태스크에서도 FNN 을 추가적으로 사용하거나 또 다른 알려진 모델을 조합하는 것이 단일한 모델을 사용했을 때보다 더 성능을 끌어올리기 위한 좋은 선택지가 될 것으로 기대한다.

참고문헌

[1] Tomas Mikolov 외 3 명, Linguistic Regularities in Continuous Space Word Representations, NAACL, 2013

[2] Suman Dowlagar 외 1 명, Unsupervised Technical Domain Terms Extraction using Term Extractor, LTRC, 2021

[3] Tomas Mikolov 외 3 명, Efficient Estimation of Word Representations in Vector Space, ICON, 2013.
[4] Poitr Bojanowski 외 3 명, Enriching Word Vectors with Subword Information, TACL, 2017.
[5] Jacob Devlin 외 3 명, BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, Google AI Language, 2019.
[6] Yile Wang 외 2 명, How Can BERT Help Lexical Semantics Tasks?, DBLP, 2020.
[7] Michael Gutmann 외 1 명, Noise-contrastive estimation: A new extimation principle for unnormalized statistical models, PMLR, 2010
[8] Qianchu Lui 외 2 명, Towards Better Context-aware Lexical Semantics: Adjusting Contextualized Representations through Static Anchors, EMNLP, 2020
[9] https://aiopen.etri.re.kr/service_dataset.php
[10] Nitish Srivastava 외 4 명, Dropout: A Simple Way to Prevent Neural Networks from Over_tting, Journal of Machine Learning Research, 2014