Bi-Sent2Vec을 이용한 문맥 기반 단어 의미 결정 기능 재설계를 통한 성능 향상

이예준 ○ 전석희
경희대학교 컴퓨터공학과
yejuny1528@gmail.com, jeon@khu.ac.kr

Improvement of context-based meaning selection using Bi-Sent2Vec

Ye-Joon Lee ○ Seok-Hee Jeon
Department of Computer Science and Engineering, KyungHee University

요 약

최근 다양한 영상매체를 통해 전파되는 K-POP이나 K-DRAMA 등 한류의 영향으로 자연스럽게 한국어 공부를 희망하는 외국인들의 수도 증가하고 있다. 이러한 수요에 맞추어 미리내는 다양한 한국어 교육 서비스들을 제공한다. 이 중에서 한국어 문장의 영어 번역 문장 문맥을 기반으로 각 문장 내 단어들의 영어 의미를 결정해주는 기능이 있는데, 문장 내에 동음이의어가 2개 이상 존재하는 경우 의미 결정에 오역이 발생하는 문제점이 있다. 이에 본 논문에서는 Bi-Sent2Vec을 통한 cross-lingual numerical representation을 가지고 기존 의미 결정 알고리즘을 재설계하여 문제점 개선과 더불어 성능을 강화시키고자 한다.

1. 서 론

최근 몇 년간 BTS로 인해 K-POP이 다시 외국에서 큰 인기를 누리고 있다. 이러한 한류는 한국어 공부에 크게 영향을 끼친다. 2018년도를 기준으로 여러 해외 전문가들의 분석 [1]을 살펴보면 K-POP을 통해 한국어 공부를 희망하는 외국인들이 증가했음을 알 수 있다. 또한 [1]을 통해 외국인들의 기준에서 한국어를 공부하는 것이 꽤 어렵다는 것을 알 수 있는데, 미리내 [2]는 이러한 외국인들을 대상으로 보다 쉽게 한국어 공부를 할 수 있도록 돕는 한국어 교육 서비스이다.

문맥 기반 다의어 의미 결정(Context-based meaning selection) 기능은 위에서 소개한 미리내 서비스의 여러 기능 중 하나이다. 유저는 미리내 서비스 메인 화면에서 한국어 문장을 입력하고 그에 대한 분석 정보들을 확인할 수 있는데, 위의 기능은 입력된 한국어 문장을 기준으로 영어로 번역된 전체 문장을 가져와서 번역된 문장의 문맥을 기반으로 각각의 단어들이 해당 문장에서 어떤 영어 의미로 쓰였는지를 알려준다. 그러나 해당 기능은 현재 한 문장을 기준으로 문장 내에 다의어를 가지는 동음이의어가 2개 이상인 경우 제대로 의미 결정을 해주지 못하는 문제점을 가지고 있다.

본 논문은 2절에서 기존 기능 설계 방식과 이에 따른 문제점에 대해서 조사한다. 3절에서는 2절에서 조사한 기존 기능의 문제점에 대한 해결 방안과 현재까지 진행된 개발 과정을 서술하고, 4절에서는 지금까지 연구한 내용에 대한 중간 결론 및 향후 계획을 제시한다.

2. 기존 연구

2.1 Word embedding by FastText based on gensim

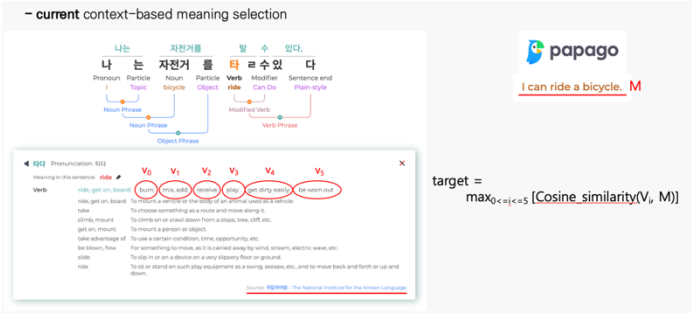
Fasttext [3]는 google의 Word2Vec과 유사한 방식인 단어에 대한 Dense-representation 중 하나이다. Word2Vec은 Corpus의 각 Word를 하나의 Atomic한 Entity로 보고 각 Word에 대하여 Vector를 생성한다. 이와 다르게 Fasttext는 Word를 바라볼 때 한 번 더 분해하여 각 Word를 Character n-gram의 합으로 본다. 따라서 각 Word에 대한 Vector를 생성할 때 Character 각각의 n-gram vector들에 대한 Hash-map을 미리 생성하고 이를 조합하여 만들어낸다.

Word-embedding 이전에 단어를 벡터로 표현하기 위해 고안된 방식으로 one-hot encoding이 있다. 이를 Sparse-representation 라고 하며 제시된 Word 집합의 개수만큼 Vector의 차원이 결정되기 때문에 공간적으로 매우 비효율적인 방식이라고 볼 수 있다. 또한 1을 위치시키기 위해 인덱싱하는 과정에서 특별히 주의를 기울이지 않는 이상 각 단어의 의미를 담지 못 한다는 단점도 있다.

반면에 위에 소개한 Word2Vec과 Fasttext는 Word

embedding 하는 과정에서 주변 단어들을 통해 중심 단어를 예측하거나 중심 단어를 통해 주변 단어들을 예측하는 CBOW(Continuous Bag of Words) 혹은 Skip-gram 방식을 사용하고 Word2Vec의 경우 단어들을 벡터 평면에 배치시켜서 문맥적 의미마저 보존이 가능하기 때문에 단어의 의미를 담을 수 있다. 또한 Fasttext의 경우 각 Word의 Sub-word들을 Character n-gram을 통해 Hash에 저장했기 때문에 Out-of-Vocabulary(OOV)에도 훌륭히 대응이 가능하다. 따라서 기존 Meaning selection 기능을 위한 Word embedding 방식으로 Fasttext를 선정했다.

2.2 Select meaning by Cosine similarity

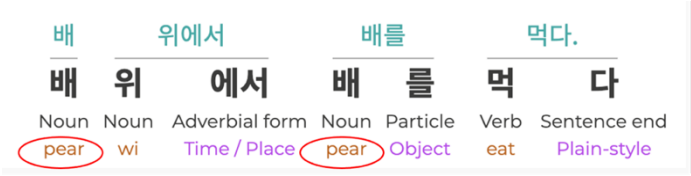


[그림 1] 기존 방식 매커니즘

위의 Fasttext를 통해 Training 시킨 결과물인 Embedding vectors를 더해 Papago에서 가져온 한국어 문장의 영어 번역 전체 문장의 문맥 벡터 M을 생성한다. 그리고 의미를 결정지어줘야 하는 Target 다의어에 대한 n개의 영어 후보 단어들을 국립국어원에서 가져온 뒤에 각각 V_i로 치환하고 문맥 벡터 M과의 Cosine-similarity를 구해서 가장 큰 값을 반환한다.

2.3 기존 연구의 문제점

2.3.1 Several homonyms in one sentence



[그림 2] 한 문장 내에 동음이의어가 발생하는 경우

기존 기능 설계 방식을 적용하는 과정에서 각각의 후보 벡터인 V_i를 항상 전체 문장 벡터인 M과 유사도를 계산하는 과정을 거치기 때문에 문맥에서 가장 영향력이 큰 서술어 부분으로 인해 '먹는'행위가 가능한 pear가 반환된다. 또한 이는 모든 '배'라는 글자에 반환되기 때문에 동음이의어를 제대로 처리하지 못하는 결과가 발생한다.

2.3.2 Mistranslation of papago

문맥 기반 다의어 의미 결정(Context-based meaning selection) 기능의 결정 단계에 있어서 유사도를 구하는 과정은 매우 중요함을 알 수 있다. 따라서 그의 Arguments 중 하나인 문맥 벡터 M의 영향 역시 큰데, 만약 Papago에서 잘못된 번역이 넘어온다면 이 역시도 의미 결정에 있어서 큰 오류가 생길 수 있다.

3. 해결방안 및 진행상황

3.1 CBOW(Continuous Bag of Words) for selecting

CBOW는 주변 단어들을 통해 중심 단어를 예측하는 방법이다. 2절에 언급한 것처럼 밀집 표현 방식인 Dense representation에 쓰이며 이 덕분에 문맥적으로 가까운 단어끼리 군집화 할 수 있었다. 동음이의어에 대한 문제점이 나타나는 [그림 2]처럼 첫 번째로 나타나는 '배'와 두 번째로 나타나는 '배' 모두를 전체 문장 벡터인 M과의 유사도를 통해 의미를 결정하지 말고 각각에 대한 특정 window size를 지정해 주변 단어들의 벡터 합으로 M을 대체하고자 한다. 또한 이 과정에서 기존에 사용했던 벡터 M은 papago를 통해 영어로 이루어진 문장들이었지만 오역 방지를 위해 한글 형태소들을 가져와 벡터 M을 새롭게 생성할 것이다. 예를 들어 window size가 2라고 가정했을 때, 첫 번째 '배'라는 글자의 영어 의미는 '위'와 '에서'의 벡터합인 M과 '배'라는 뜻을 가진 영단어들 Vi를 통해 구하고, 두 번째 '배'에 대한 벡터 M은 '위', '에서', '를'과 '먹'의 벡터합으로 구해준다. 따라서 Word-embedding을 한국어와 영어 둘 다 해주어야 한다.

3.2 Khaiii for morpheme analysis

일반적으로 한국어 문장에 대한 Word-embedding을 진행하기 전에 영어와는 다르게 특별한 전처리 과정을 거치게 된다. 문장을 의미를 가진 가장 작은 단위인 형태소 단위로 분해해줘야 한다. 따라서 영어 문장에 대해선 간단하게 공백을 기준으로 분해했고, 한국어 문장에 대해서는 현재 미리내에서 사용하고 있는 형태소 분석기인 카카오의 Khaiii [4]를 이용했다. 왜냐하면 현재 본 논문에서 연구하는 기능이 미리내 서비스와 직결되기 때문에 형태소 분해를 하는 방식 자체가 미리내에서 쓰이고 있는 것과 같아야 한다. 형태소 분해 과정을 진행한 Corpus들은 미리내에서 보유중인 AI-hub의 데이터들을 사용했고 대략 160만개의 한국어-영어 pairs를 사용했다.

3.3 Training by Bi-Sent2Vec based on Fasttext

Fasttext를 기반으로 동시에 2개의 언어에 대하여 Word-embedding을 훈련시킬 수 있는 Bi-Sent2Vec [5]을 통해 위에서 전처리한 데이터들을 training 시켰다. 2가지 언어의 구분을 위해 각 형태소의 뒤에 특별한 태그 값인 _ko, _en 을 추가했다.

```
[sksbbb@nipa2021-1434:~$ python3 train.py trainingData.txt model-ko-en
Read 90M words
Number of words:  750000
Number of labels: 0
Progress: 20.0%  words/sec/thread: 46652  lr: 0.160000  loss: 1.852828  eta: 0h4m 4m
Saving Model ----- Checkpoint 1
Progress: 49.3%  words/sec/thread: 44863  lr: 0.101377  loss: 1.852818  eta: 0h2m
Saving Model ----- Checkpoint 2
Progress: 77.6%  words/sec/thread: 44404  lr: 0.044819  loss: 1.852816  eta: 0h1m
Saving Model ----- Checkpoint 3
Progress: 100.0%  words/sec/thread: 44075  lr: 0.000000  loss: 1.852816  eta: 0h0m
done
[sksbbb@nipa2021-1434:~$ ls
```

[그림 3] Training by Bi-Sent2Vec

현재는 모든 Hyper-parameter를 default 값으로 고정시킨 뒤에 훈련을 진행했다. 데이터의 양이 많아서 [그림 3]과 같이 총 3개의 checkpoint 를 거쳐서 완료되었다. 형태소들에 대한 벡터값만을 담은 .vec 확장자 파일은 2GB정도였으며 따로 OOV 처리를 위한 Hash값도 가진 .bin 파일은 4GB 가량 차지했다.

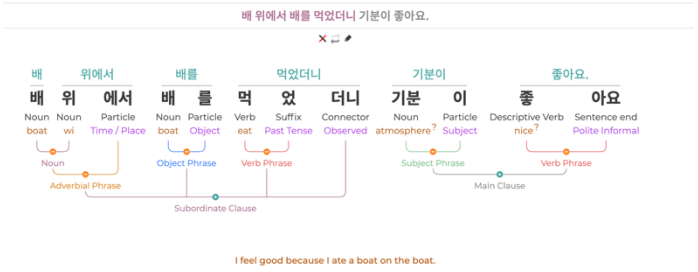
```
(pytorch_p36) ubuntu@nipa2021-1434:~/user/Yejoon$ (pytorch_p36) ubuntu@nipa2021-
Warning : `load_model` does not return WordVectorModel or SupervisedModel any mo
re, but a `FastText` object which is very similar.
이예준_ko doesn't exists
[ 5.59349824e-03  4.65829980e-05  5.84095996e-03 -1.14798378e-02
 -3.88935884e-03 -1.48636466e-02 -1.26722055e-02 -6.85916690e-04
 -1.63150225e-02  4.94023692e-03 -5.61519759e-04 -6.26555411e-03
 -4.39065136e-03  2.82162754e-03 -6.33190095e-04  1.07011879e-02
  1.32069141e-02  6.14525797e-03 -1.58856604e-02  3.17631382e-03
 -5.34677738e-03 -6.68943627e-03  1.07514753e-03  4.51424997e-03
  6.88877888e-03 -3.29978997e-03 -2.98457704e-02  1.24454945e-02
  1.72166564e-02 -2.09773658e-03 -8.91449116e-03 -1.06118964e-02
  9.19918437e-03 -1.82337519e-02  3.31473816e-03 -1.38636651e-02
  2.07882840e-03 -2.35198560e-04  2.31905351e-03  1.26826810e-02
  6.73656585e-04 -1.31355347e-02 -5.28034614e-03  2.13248492e-03
 -1.27677368e-02  1.48164136e-02 -7.04112416e-03  2.16194820e-02
 -7.62907090e-03 -3.53342295e-03  3.82514717e-03  1.14024403e-02
 -8.10497254e-03  1.82586082e-03 -1.22972187e-02 -5.39734447e-03
 -6.95190998e-03 -3.56388348e-03  8.76418222e-03  1.69300772e-02
  1.22539466e-02  1.07633518e-02 -1.60615481e-02 -1.13725348e-03
  3.98520520e-03 -1.91433588e-03  7.59040122e-04 -9.47983842e-03
  2.04762025e-03  6.33774092e-03  1.84733025e-03 -1.10376673e-02
  1.44017907e-03 -6.68675406e-03 -2.68826098e-03  3.62160313e-03
 -2.28068233e-02 -1.33070471e-02 -2.05584671e-02  6.24577142e-03
  1.06597841e-02 -1.35130971e-03 -4.16385289e-03  6.85085033e-05
 -6.99208351e-03  9.34011210e-03  4.73274896e-03 -1.22962939e-02
  6.75544376e-03  4.89197532e-03 -6.07322110e-03  1.78995244e-02
 -3.05684446e-03 -5.65339299e-03  2.83620204e-03 -6.81320904e-03
  3.05580022e-03  1.42570855e-02  6.31560897e-03  3.86112020e-03
  5.86650940e-03  8.65617860e-03  3.07405414e-03 -3.43616400e-03
 -6.89282455e-03 -1.84163789e-03  1.92505284e-03 -4.10929509e-03
 -7.79395644e-03  3.52924806e-03 -1.92760434e-02  1.46441227e-02
  9.94451344e-04 -2.18742271e-03 -1.48435147e-03  3.53058480e-04
  1.78052357e-03  1.77036040e-03 -5.39420033e-03 -4.93546994e-03
 -2.14552251e-03  3.46460682e-03 -2.00602156e-03  5.95558900e-03
```

[그림 4] Generate vector with OOV case

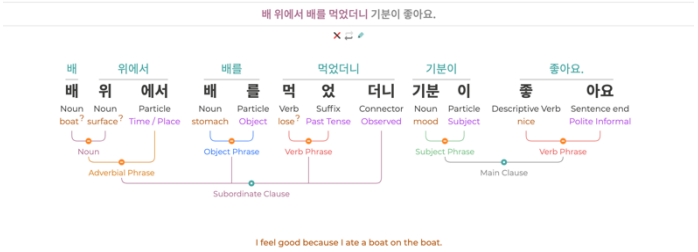
위의 [그림 4]와 같이 Fasttext object로 성공적으로 load가 된 후 훈련에 포함되지 않았던 본 저자의 이름에 대한 벡터 값도 훌륭히 생성해내는 것을 확인할 수 있었다.

3.4 Test by CBOW & Bi-Sent2Vec

Bi-Sent2Vec을 통한 Embedded vector와 새로운 방식의 의미 결정 설계를 통해 테스트를 진행해보았다. 아래 [그림 5]는 기존 방식대로 진행했을 경우의 결과이고 [그림 6]은 새로운 방식을 도입했을 때의 결과이다.



[그림 5] 기존 연구에서의 결과



[그림 6] 현재 연구 및 개발 중인 결과

위의 [그림 5]와 [그림 6]에서 동음이의어인 '배'에 대해 살펴보면 [그림 5]에선 하나의 영어 단어로 통일이 되는 현상이 나타나지만 [그림 6]에선 서로 다르게 나타났다. 첫 번째 '배'는 맞췄지만 두 번째로 나타나는 '배'는 맞추지 못했다. 하지만 이 과정이 무의미 했다고 볼 수 없다. 우선 기존 연구에서처럼 쌍둥이 벡터가 되어서 같은 결과값을 나타내는 것보다 발전했고 동음이의어 처리에 대한 가능성을 보았다.

4. 결론 및 향후 연구

현재까지 진행된 연구결과를 바탕으로 미루어 보아 동음이의어 처리에 대한 가능성을 발견했다. 이제 여러 예시 문장들을 입력하여 틀린 결과가 나오는 부분을 기준으로 그 원인에 대해 분석하고 그를 해결하기 위한 추가적인 알고리즘을 도입할 계획이다.

참고문헌

[1] Matt Pickles, "K-pop drives boom in Korean language lessons", BBC News global education-business, 2018. 07. 11

[2] Mirinae,
https://mirinae.io/
[3]Fasttext,
https://github.com/facebookresearch/fastText
[4] Khaiii,
https://github.com/kakao/khaiii
[5] Bi-Sent2Vec,
https://github.com/epfml/Bi-Sent2Vec