한국어 Word2vec 모델을 위한 최적의 형태소 분석기 선정

강형석, 양장훈
서울미디어대학원 대학교 뉴미디어학부
e-mail: jiangjs@naver.com

Selection of the Optimal Morphological Analyzer for a Korean Word2vec Model

Hyungsuc Kang, Janghoon Yang
Dept. of New Media, Seoul Media Institute of Technology

요 약

본 논문의 목적은 오픈 소스로 공개된 3 가지 한국어 형태소 분석기(kkma, twitter 및 mecab-ko)를 비교해서 한국어 자연어 처리에 가장 적합한 분석기를 선정하는 것이다. 이를 위해, 자연어 처리 분야에서 중요한 단어 임베딩 방법론 중 하나인 word2vec 모델의 성능 검증 방법을 사용해서 각 형태소 분석기의 성능을 정량적으로 비교했다. 그 결과, mecab-ko 형태소 분석기가 최적임이 확인되었다. 단, 성능 검증에 사용된 어휘가 오직 명사뿐이라는 한계가 있으므로, 향후 연구에서는 좀 더 다양한 품사에 대한 성능 검증이 필요할 것으로 보인다.

1. 서론

자연어 처리(NLP, Natural Language Processing)의 전처리(preprocessing) 단계에서 토큰화(tokenization)는 필수적이다. 토큰화는 일련의 글자들을 서로 구별되는 의미 단위(토큰, token)로 분해하는 과정이다[1]. 일반적으로 영어 NLP 에서는 단순히 빈칸 단위로 단어를 토큰화한다. 하지만 형태소(morpheme, 의미의 최소 단위. 영어의 경우, 복수형 접미사 –(e)s, 현재분사형 접미사 -ing 등)를 분석해서 별도의 토큰으로 처리한다면 더 효율적일 것이다. 그리고 문법적 분석을 통해 각 토큰의 품사를 태깅(tagging)한다면, 명사와 동사의 형태가 동일한 경우가 많은 영어의 경우 더 효율적인 토큰화를 달성할 수 있을 것이다.

영어 NLP 에서 이런 형태소 분석 및 품사 태깅은 선택사항인 반면, 한국어 NLP 에서 형태소 분석/품사 태깅을 통한 토큰화는 거의 필수적이라고 할 수 있다. 한국어의 띄어쓰기 규정에 따르면, 체언(명사, 대명사, 수사)과 조사 및 용언(동사, 형용사, 서술격 조사)의 어간과 어미는 붙여 쓰는 것이 기준이다. 게다가 복합명사를 이루는 각 명사와 명사(예를 들어, 전통 문화) 그리고 본용언과 보조용언(예를 들어, 놓여 있다)을 붙여 쓰는 것도 허용하기 때문에, 2 개 이상의 형태소가 하나의 어절(띄어쓰기의 단위)로 연결된 경우가 대부분이다. 영어처럼 단어의 파생형(예를 들어, 동사의 기본형, 과거형, 과거분사형 등)이 몇 개 되지 않는 경우, 각 파생형을 별도의 토큰으로 처리하여도 별 문제가 되지 않는다. 하지만 한국어의 조사 및 어미가 각각 체언 및 어간과 결합하여 파생될 수 있는 경우의 수가 훨씬 크기 때문에, 한국어 NLP 에서 형태소 분석/품사 태깅을 통한 토큰화는 거의 필수불가결하다고 할 수 있다.

현재 다양한 한국어 형태소 분석기(morphological analyzer)가 오픈 소스(open source)로 공개되어 있어서, GPL 라이선스에 따라 자유롭게 이용할 수 있다. 이런 형태소 분석기들의 일반적인 형태소 분석 성능은 알려져 있다. 하지만 이런 성능은 정성적으로 기술될 뿐, 그 차이가 정량적으로 비교되지는 않는다. 본 논문에서는 NLP 분야에서 최근 각광받고 있는 단어 임베딩(word embedding) 방법 중 하나인 word2vec 모델에 적용하기에 가장 좋은 한국어 형태소 분석기를 선정하고자 한다. Word2vec 모델의 표준적인 성능 검증 방법을 통해 한국어 형태소 분석기의 성능을 정량적으로 비교하고, 이 정량적 비교를 기준으로 어떤 형태소 분석기가 가장 적합한지를 결정할 것이다.

2. 한국어 형태소 분석기/품사 태거

한국어 정보처리를 위한 파이썬 패키지 KoNLPy[2, 3]는 오픈 소스로 공개된 한국어 형태소 분석기(kkma, komoran, hannanum, twitter 및 mecab-ko)를 파이썬으로 포팅한 품사 태거(POS tagger) 클래스를 제공한다. 각 품사 태거의 실행 시간과 성능에 대한 분석[3]에 따르면, 다음 3 가지가 한국어 word2vec 모델을 위한 토큰화 방법으로 적당하다.

<표 1> KoNLPy 에서 제공하는 3 가지 품사 태거의 특징

Kkma (꼬꼬마)	서울대에서 개발한 한국어 형태소 분석기. 분석 품질은 우수하지만 실행 속도가 매우 느리다는 단점이 존재한다. 용언의 어간과 어미에 대한 상세한 분석이 가능하지만, 띄어쓰기가 없는 긴 어절을 분석할 때 메모리 부족 오류가 발생하는 버그가 존재한다.
Twitter	SNS 제공 회사인 트위터에서 만든 오픈 소스 한국어 처리기. 실행 속도는 빠르지만 분

	석 품질이 좋지 않다. 품사에 대한 하위 분류(예를 들어, 일반명사와 고유명사의 구분)가 없으며, 어간과 선어말 어미를 구분하지 않는다.
Mecab-ko	일본어 형태소 분석기 mecab을 한국어용으로 포팅한 버전. 실행 속도가 제일 빠르고 분석 품질도 양호하다. 특히 띄어쓰기가 없는 어절의 분석에 강하지만, 어간과 어미를 kkma 품사 태거만큼 세분하지는 않는다.

<표 2>는 위와 같은 형태소 분석/품사 태깅의 특징을 보여 주는 전형적인 예이다.[1] 예문 1 은 띄어쓰기가 되지 않은 전형적인 문장이다. 문장의 의미상, "가방"을 명사로 분류한 (1.1)과 (1.2)의 형태소 분석/품사 태깅이 잘못되었음을 쉽게 알 수 있다. 그리고 예문 2 의 "나는"은 문맥상 "대명사+조사"의 형태가 아니라, "동사 어간(날-)+어미(-는)"로 분석되어야 하는데, (2.1)만이 정확히 분석한 것을 확인할 수 있다. 마지막으로 예문 3 의 "그쳤다"라는 용언을 "동사의 어간(그치-)+선어말 어미(-었-)+어말 어미(-다)"로 가장 자세히 분석한 경우는 (3.1)이고, (3.2)와 (3.3)은 어간과 선어말 어미를 분리하지 못했다.

<표 2> 전형적인 형태소 분석/품사 태깅의 예

예문 1	**아버지가방에들어가신다**
Kkma	(1.1) 아버지/NNG 가방/NNG 에/JKM 들어가/VV 시/EPH ㄴ다/EFN
Twitter	(1.2) 아버지/Noun 가방/Noun 에/Josa 들어가신/Verb 다/Eomi
Mecab-ko	(1.3) 아버지/NNG 가/JKS 방/NNG 에/JKB 들어가/VV 신다/EP+EC
예문 2	**하늘을 나는 자동차**
Kkma	(2.1) 하늘/NNG 을/JKO 날/VV 는/ETD 자동차/NNG
Twitter	(2.2) 하늘/Noun 을/Josa 나/Noun 는/Josa 자동차/Noun
Mecab-ko	(2.3) 하늘/NNG 을/JKO 나/NP 는/JX 자동차/NNG
예문 3	**감축하는 데 그쳤다**
Kkma	(3.1) 감축/NNG 하/XSV 는/ETD 데/NNB 그치/VV 었/EPT 다/EFN
Twitter	(3.2) 감축/Noun 하는/Verb 데/Noun 그쳤/Verb 다/Eomi
Mecab-ko	(3.3) 감축/NNG 하/XSV 는/ETM 데/NNB 그쳤/VV+EP 다/EF

요컨대 3 가지 품사 태거 중, 형태소 분석 성능의 면에서는 kkma 또는 mecab-ko 태거가 우수하고, 실행 속도의 면에서는 mecab-ko 태거가 가장 우수하다. 하지만 형태소 분석 성능에서 어느 쪽이 더 우수한지를 정량적으로 비교할 수는 없다. 만약 kkma 태거가 형태소 분석 성능에서 mecab-ko 태거보다 월등하다면, 그 이득은 kkma 태거의 매우 느린 실행 속도를 감내할 수준인지도 확인할 수가 없다. 따라서 각 형태소 분석기/품사 태거의 성능을 정량적으로 비교할 방법이 필요하다.

3. Word2vec 모델의 성능 평가 방법

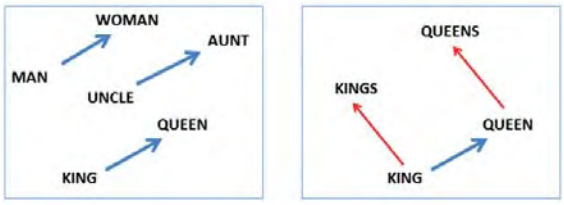
단어 임베딩 방법론 중 하나인 word2vec 은 단어들의 의미론적(semantic) 및 문법적 관계를 정확히 포착해서 분산 벡터 표상(distributed vector representation)을 학습하는 모델이다. 해당 모델은 CBOW(Continuous Bag-Of-Words)와 SG(Skip-Gram)라는 2 가지 학습 알고리즘을 제공하는데, 보통 SG 의 성능이 CBOW 보다 더 나은 것으로 알려져 있다[4, 5]. 일반적으로 word2vec 모델에 대한 성능 평가 방법에는 유사도 검사(similarity test)와 유추 검사(analogy test)가 있다.

3.1 유사도 검사

영어 word2vec 의 성능 평가에서 널리 쓰이는 방법 중 하나는 WordSim353 검사이다[6]. 이 유사도 검사는 353 개 영어 단어 쌍의 관련도/유사도를 1~10 사이의 점수로 표현한 WordSim353 데이터셋[7]을 이용한다. 구체적인 검사 방법은 다음과 같다. 학습이 완료된 word2vec 모델에서 각 단어 쌍의 벡터 값을 찾아서 코사인 유사도(cosine similarity)를 구한다. 그런 다음, 해당 단어 쌍의 실제 관련도/유사도 점수와의 상관계수를 구한다. 이 상관계수가 높일수록 해당 모델의 단어 임베딩 성능이 높은 것으로 간주한다.

이 WordSim353 데이터셋을 한국어 word2vec 에 적용하기 위해, 적절한 한국어 번역어가 없거나 미국적 특색이 강한 (예를 들어, 체스 용어인 king-rook 쌍 및 테러와 관련된 Arafat-terror 쌍) 단어 쌍 60 개를 제외한 293 개의 단어 쌍(모두 명사)을 한국어로 번역한 데이터셋을 사용했다.[2] 단, 이 유사도 검사는 영어 화자가 생각하는 각 단어 쌍의 관련도/유사도를 사용하기 때문에 한계가 있다.

3.2 유추 검사



(그림 1) 단어 벡터의 의미론적/문법적 관계

Word2vec 모델은 (그림 1)과 같은 단어들의 의미론적 및 문법적 관계를 벡터 공간에 잘 표상하는 것으로 알려져 있다. (그림 1)의 왼쪽 그림은 남녀 관계에 있는 단어 쌍들의 의미론적 관계를 잘 보여 준다. 즉, 벡터 공간에 임베딩된 해당 단어 벡터 쌍의 거리(파란색 화살표)가 일정하다는 것은 해당 단어 쌍들의 의미론적 차이가 벡터 공간에서 잘 구현되었음을 의미한다. 반면에 오른쪽 그림은 단수-복수 관계에 있는 명사 쌍들의 문법적 관계를 보여 주는데,

[1] 품사 태그에 대한 자세한 정보는 다음의 구글 공유 문서를 참조하라. https://docs.google.com/spreadsheets/d/1OGAjUvalBuX-oZvZ_-9tEfYD2gQe7hTGsgUpiiBSXI8/edit#gid=0

[2] 실제 유사도 검사에 쓰인 데이터셋은 다음의 구글 공유 문서를 참조하라. https://docs.google.com/spreadsheets/d/1WWQZKWmvY3a8vkFEJ3ZtnAcRa0-smOVXHkX_dUL98Hw/edit?usp=sharing

단수 단어의 벡터와 복수 단어의 벡터 사이의 거리(빨간색 화살표)가 일정함을 통해 해당 단어 쌍들의 문법적 관계가 벡터 공간에서 잘 구현되었음을 보여 준다.

Word2vec 모델의 유추 검사는 특정 관계에 있는 이런 단어 쌍의 벡터 거리가 얼마나 일정한지를 검사한다. 예를 들어, 다음과 같은 벡터 관계식이 성립하는지를 검사한다고 할 수 있다.

v(KING) – v(QUEEN) + v(MAN) = v(WOMAN) (식 1)

즉, KING 의 단어 벡터에서 QUEEN 의 단어 벡터를 빼고 MAN 의 단어 벡터를 더한 값이 WOMAN 의 단어 벡터에 해당하는지를 확인하는 것이다. 이런 단어 벡터 사이의 유추 관계를 검사하는 것은 word2vec 모델의 성능을 평가하는 표준적인 방법이다[4, 5].

한국어의 경우 영어와 같은 문법적 관계(예를 들어, 명사의 단수형-복수형 관계, 형용사의 원급-비교급 관계, 동사의 기본형-과거형 관계 등)가 존재하지 않으므로, 영어에 적용되는 문법적 유추 검사를 바로 적용할 수는 없다. 따라서 한국어 word2vec 모델에 대해서는 일단 의미론적 유추 검사만을 적용하겠다.

실제 본 논문의 의미론적 유추 검사에 사용된 단어 쌍은 국가-수도 15 쌍(예를 들어, 대한민국-서울), 남자-여자 15 쌍(예를 들어, 신랑-신부) 및 국가-통화 8 쌍(예를 들어, 미국-달러)이고, 이로부터 총 476 개의 테스트셋(예를 들어, 신랑-신부-왕자-공주)을 생성했다.[3] 각 테스트셋의 첫 3 가지 단어에 대해 위의 (식 1)과 같은 벡터 연산을 수행한 후, 그 결과와 벡터 공간에서 가장 가까운 단어가 마지막 단어이면 해당 테스트셋을 정답으로 처리하고 아니면 오답을 처리한다. 테스트셋의 단어가 word2vec 모델링에 사용된 말뭉치(corpus)의 어휘에 없는 경우(OOV, Out-Of-Vocabulary), 해당 단어는 학습되지 않는다. 따라서 이런 단어가 하나라도 포함된 테스트셋은 미학습으로 처리한다. 유추 검사의 결과는 전체 테스트셋에 대한 정답률(accurate %) / 오답률(inaccurate %)/미학습률(OOV %)로 표시한다. 전체 테스트셋은 의미론적 관계에 따라, 섹션 1(국가-수도 쌍), 섹션 2(남자-여자 쌍) 및 섹션 3(국가-통화 쌍)으로 구분되어 있다.

4. 한국어 Word2vec 모델에 적용된 형태소 분석기의 정량적 성능 비교

한국어 Word2vec 모델링 및 그 성능 평가를 위해, 오픈소스 파이썬 라이브러리인 gensim 패키지[8, 9]를 사용했다. 그리고 모델링에 사용된 말뭉치는 한국어 위키백과 덤프 파일(2017 년 12 월 20 일 덤프 파일, 2.6GB)에서 추출된 텍스트 파일(433.3MB)을 사용했다.

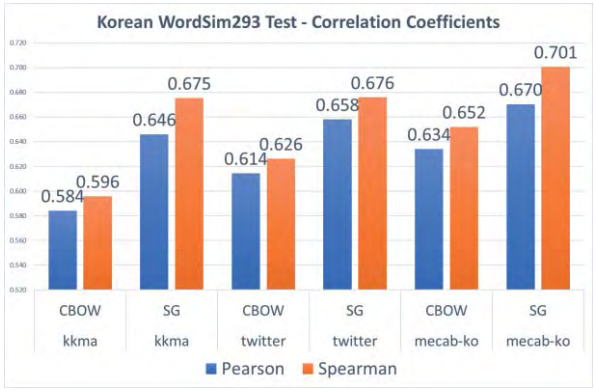
4.1 실행 속도

<표 3>은 433.3MB 의 말뭉치 텍스트 파일을 형태소 분석하는 데 걸린 시간을 정리한 표이다. 일반적으로 kkma 및 mecab-ko 태거의 형태소 분석 성능이 비슷함을 고려할 때, kkma 태거의 실행 시간(mecab-ko 태거 실행 시간의 약 80 배)은 큰 단점이 된다. 더욱이 일반적으로 품사 태거의 실행 시간은 말뭉치가 커질수록 기하급수적으로 증가하므로[3], 향후 대용량 말뭉치에 kkma 태거를 적용하는 것은 상당한 컴퓨팅 자원을 요구할 것으로 예상된다. 따라서 실행 속도의 측면에서 최선의 품사 태거는 mecab-ko 태거라고 할 수 있다.

<표 3> 형태소 분석에 소요된 품사 태거별 실행 시간

품사 태거	Kkma	Twitter	Mecab-ko
실행 시간	12.12 시간	40.72 분	9.12 분

4.2 유사도 검사

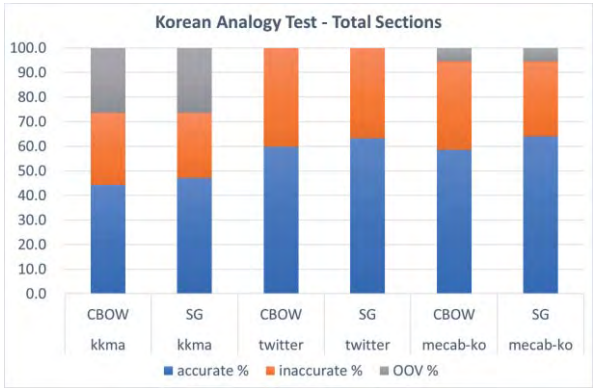


(그림 2) 품사 태거별 유사도 검사 결과

위 도표는 각 품사 태거로 전처리한 한국어 위키백과 말뭉치에 대해 유사도 검사를 시행한 결과를 정리한 것이다. 2 가지 학습 알고리즘(CBOW, SG)과 3 가지 품사 태거(kkma, twitter, mecab-ko)의 6 가지 조합에 대한 Pearson 및 Spearman 상관계수를 비교했다.

전반적으로 CBOW 알고리즘보다 SG 알고리즘의 경우가 더 높은 상관계수를 보인다. 그리고 학습 알고리즘이 동일할 경우, kkma, twitter 및 mecab-ko 태거의 순으로 상관계수가 높게 나왔다. 따라서 유사도 검사의 결과만 놓고 보면, SG 알고리즘과 mecab-ko 태거를 사용한 경우 단어 임베딩의 성능이 가장 높다고 할 수 있다.

4.3 유추 검사



(그림 3) 품사 태거별 유추 검사 결과 – 전체 정답률/오답률/미학습률

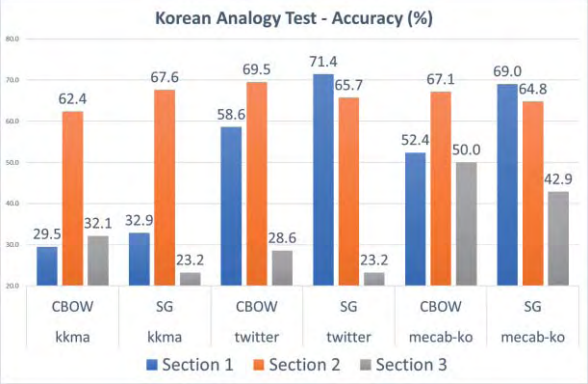
(그림 3)은 품사 태거별 유추 검사의 결과를 나타낸 것으로, 학습 알고리즘과 품사 태거의 6 가지 조합에 대한 정

[3] 실제 유추 검사에 쓰인 데이터셋은 다음의 구글 공유 문서를 참조하라. https://docs.google.com/document/d/104OmfjlsZ4z8ESKa9u9hjS8-WmlrR3sm1ZyyfQsdNQY/edit?usp=sharing

답률/오답률/미학습률을 보여 준다. 유사도 검사와 마찬가지로, 전반적으로 CBOW 알고리즘보다 SG 알고리즘에서 유추 검사의 정답률이 더 높게 나온다. 하지만 그 차이는 그다지 크지 않다. 그리고 정답률 기준으로 twitter 및 mecab-ko 태거의 성능은 비슷하고, kkma 태거의 성능이 가장 낮게 나옴을 확인할 수 있다.

Kkma 태거의 경우 미학습률이 높은 이유는 수도/통화와 관련된 일부 고유명사(예를 들어, 쿠알라룸푸르, 자카르타, 모스크바, 이스탄불, 루블, 루피 등)가 말뭉치에서 학습되지 않았기 때문이다. 이런 고유명사가 실제로 말뭉치에 없어서 학습되지 않은 것이 아니라, 품사 태거가 형태소를 잘못 분석한 결과인 것으로 보인다. 예를 들어, 말뭉치의 '모스크바'라는 단어를 kkma 태거는 '모스크/NNG 바/NNG'와 같이 2 개의 일반명사로 분석했고, 테스트셋에서는 '모스크바/NNP'로 설정되어 있었다. 따라서 이런 고유명사들은 한국어 word2vec 모델이 제대로 학습하지 못한 것이다. 마찬가지로 Mecab-ko 태거도 통화와 관련된 일부 고유명사(예를 들어, 루블, 루피 등)가 말뭉치에서 제대로 학습되지 않았다. 일반적으로 품사 태거들은 고유명사나 신조어에 대한 분석에 취약한 것으로 보인다. 단, twitter 태거는 품사를 하위 분류로 세분하지 않기 때문에 미학습률이 0%로 나온 것으로 보인다.

마지막으로 학습 알고리즘과 품사 태거의 6 가지 조합에 대한 각 섹션별 정답률은 (그림 4)에 주어져 있다. 전반적으로 6 가지 조합 모두에 대해, 남자-여자 쌍으로 이루어진 섹션 2 의 정답률은 60%를 상회하는 결과를 보인다. 이는 섹션 2 에 속하는 어휘가 말뭉치에서 출현하는 빈도가 높기에 이런 단어 쌍의 관계가 잘 학습된 것으로 볼 수 있다.



(그림 4) 품사 태거별 유추 검사 결과 – 각 섹션별 정답률

국가-수도 쌍으로 이루어진 섹션 1 의 경우 kkma 태거의 정답률이 현저히 낮은데, 이는 수도와 관련된 일부 고유명사(예를 들어, 쿠알라룸푸르, 자카르타, 모스크바, 이스탄불 등)가 아예 학습되지 않았기 때문이다. 그리고 국가-통화 쌍으로 이루어진 섹션 3 의 정답률은 나머지 섹션의 정답률에 비해 전반적으로 낮고, 특히 kkma 및 twitter 태거의 경우가 mecab-ko 태거의 경우보다 훨씬 더 낮다. Kkma 태거의 경우는 통화 관련 일부 어휘(예를 들어, 루블, 루피 등)가 아예 학습되지 않았기 때문이다. 그리고 twitter 태거의 경우 정답률이 낮은 것은 통화 관련 일부 어휘(예를 들어, 원, 위안 등)가 동음이의어를 갖기 때문인 것으로 보인다. 즉, 품사를 세분화하지 않은 탓에 동음이의어인 어휘(예를 들어, 중국의 통화인 '위안'과 '위로'를 의미하는 '위안')가 구분되어 학습되지 않을 결과로 보인다.

유추 검사의 정답률을 기준으로 해도, 유사도 검사에서 처럼 mecab-ko 태거의 성능이 가장 좋다고 할 수 있다.

5. 결론

비록 kkma 태거는 용언의 어간과 어미를 상세히 분석한다는 장점을 갖고 있지만, 고유명사에 대한 형태소 분석이 취약하고 실행 시간이 오래 걸린다는 단점이 존재한다. 그리고 twitter 태거는 품사를 세분화하지 않는 단점이 존재한다. 따라서 실행 시간, 유사도 검사 및 유추 검사에 대한 이상의 결과를 종합하면, 3 가지 품사 태거 중 mecab-ko 태거가 한국어 word2vec 모델에 가장 적합한 한국어 형태소 분석기라는 결론 내릴 수 있다.

하지만 본 연구의 한계는 유사도 검사와 유추 검사에 사용된 어휘가 명사에만 국한되어 있다는 점이다. 따라서 용언의 분석에 강한 kkma 태거의 장점이 위의 검사에서 확인되지 못했을 수도 있다. 향후 명사 이외의 다양한 품사에 대한 유추 검사 방안의 개발이 요청된다.

6. 사사

이 논문은 2018 년도 정부(미래창조과학부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임(과제번호: NRF-2017R1A2B4007398)

참고문헌

[1] Kaplan, R. M. "A method for tokenizing text." *Inquiries into words, constraints and contexts* (2005): 55.

[2] 박은정, 조성준. "KoNLPy: 쉽고 간결한 한국어 정보처리 파이썬 패키지." *제26회 한글 및 한국어 정보처리 학술대회 논문집*(2014).

[3] 박은정. "KoNLPy: 파이썬 한국어 NLP." Internet: konlpy.org/ko/latest/ [Sept. 7, 2018].

[4] Mikolov, Tomas, et al. "Efficient estimation of word representations in vector space." *arXiv preprint arXiv: 1301.3781*(2013).

[5] Mikolov, Tomas, et al. "Distributed representations of words and phrases and their compositionality." *Advances in neural information processing systems*. 2013.

[6] 최상혁, 설진석, 이상구. "한국어에 적합한 단어 임베딩 모델 및 파라미터 튜닝에 관한 연구." *제28회 한글 및 한국어 정보처리 학술대회 논문집*(2016)

[7] Alfonseca, Enrique. "WordSim353 - Similarity and Relatedness." Internet: alfonseca.org/eng/research/wordsim353.html [Sept. 7, 2018].

[8] Rehurek, Radim, and Petr Sojka. "Software framework for topic modelling with large corpora." *In Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. 2010.

[9] Rehurek, Radim. "Gensim: topic modelling for humans." Internet: radimrehurek.com/gensim/ [Sept. 7, 2018].