creative commons

COMMONS DEED

저작자표시-비영리-변경금지 2.0 대한민국

이용자는 아래의 조건을 따르는 경우에 한하여 자유롭게

- 이 저작물을 복제, 배포, 전송, 전시, 공연 및 방송할 수 있습니다.

다음과 같은 조건을 따라야 합니다:

저작자표시. 귀하는 원저작자를 표시하여야 합니다.

비영리. 귀하는 이 저작물을 영리 목적으로 이용할 수 없습니다.

변경금지. 귀하는 이 저작물을 개작, 변형 또는 가공할 수 없습니다.

- 귀하는, 이 저작물의 재이용이나 배포의 경우, 이 저작물에 적용된 이용허락조건을 명확하게 나타내어야 합니다.
- 저작권자로부터 별도의 허가를 받으면 이러한 조건들은 적용되지 않습니다.

저작권법에 따른 이용자의 권리는 위의 내용에 의하여 영향을 받지 않습니다.

이것은 이용허락규약(Legal Code)을 이해하기 쉽게 요약한 것입니다.

Disclaimer

dCollection

이학석사 학위논문

자연어 처리에서 코퍼스의 크기와 문맥이 과제 정확도에 미치는 영향

2020년 8월

서울대학교 대학원

협동과정 인지과학전공

이 준 호

자연어 처리에서 코퍼스의 크기와 문맥이 과제 정확도에 미치는 영향

지도교수 김 청 택

이 논문을 이학석사 학위논문으로 제출함
2020년 7월

서울대학교 대학원
협동과정 인지과학전공
이 준 호

이준호의 이학석사 학위논문을 인준함
2020년 7월

위 원 장 윤 명 환 (인)

부위원장 김 청 택 (인)

위 원 이 준 환 (인)

국문초록

최근 자연어 처리 연구는 거대한 데이터로 만든 복잡한 모델이 주류를 이루고 있다. 그러나 항상 더 큰 데이터가 더 나은 성능을 보장하지는 않는다. 본 논문에서는 가장 널리 쓰이는 세 가지 워드 임베딩 기법에서 코퍼스의 크기와 문맥이 각각 과제 정확도에 어떻게 영향을 미치는지 영화평 감성 분석을 통해 확인하였다.

분석 결과, 모든 코퍼스 크기에서 임베딩 모델의 학습 코퍼스와 과제 코퍼스 간의 문맥 유사도가 높을수록 높은 성능이 나타났다. 또한, 문맥 유사도가 가장 높은 코퍼스에 비해 문맥 유사도가 떨어지는 코퍼스들은 코퍼스의 크기가 기준값의 15배, 17배, 53배 이상 커지더라도 더 낮은 성능을 보였다.

본 분석에서 세 워드 임베딩 기법은 각각 다른 특성을 보였다. Word2Vec은 비교적 학습이 빠르고 안정적인 성능을 보였으며, GloVe는 코퍼스의 크기가 일정 크기 이상이 되면 오히려 정확도가 감소하고 모델 간 정확도의 편차가 증가하였다. fastText는 한국어의 특성을 반영했을 때 가장 좋은 성능을 보였으나, 분석에 소요되는 시간이 다른 기법들의 2배 정도로 낮은 계산 효율성을 보였다.

본 연구는 워드 임베딩 기법을 활용한 자연어 처리에서 학습 코퍼스와 과제 코퍼스의 문맥 유사도를 정량적으로 측정하고, 학습-과제 코퍼스 간 문맥 유사도와 학습 코퍼스의 크기에 따라 과제 정확도가 어떻게 변화하는지 상세히 확인하였다는 것에 의의가 있다.

주요어 : 코퍼스, 문맥, 워드 임베딩, Word2Vec, GloVe, fastText
학 번 : 2018-20278

목 차

제 1 장 서 론 ····· 1
제 2 장 관련 연구 ····· 2
2.1. Word2Vec ····· 2
2.2. GloVe ····· 4
2.3. fastText ····· 6
2.4. 코퍼스 크기와 문맥에 관한 연구 ····· 9
제 3 장 분석 도구 ····· 13
3.1. 분석 대상 코퍼스 ····· 13
3.2. 코퍼스 간 유사도 ····· 16
제 4 장 분석 환경 및 방법 ····· 18
4.1. 모델의 설정 ····· 18
4.2. 분석 방법 ····· 19
4.2.1 코퍼스 간 문맥 유사도 측정 ····· 19
4.2.2 과제 정확도 평가 ····· 20
4.2.3 워드 임베딩 모델을 이용한 감성 분석 ····· 22
4.3. 예상 결과 ····· 24
제 5 장 결 과 ····· 26
5.1. 코퍼스 간 문맥 유사도 ····· 26
5.2. 감성 분석 ····· 27
5.2.1 Word2Vec ····· 27
5.2.2 GloVe ····· 30
5.2.3 fastText ····· 32

제 6 장 논의 및 결론 ····· 34
6.1. 전체 결과 요약 및 해석 ····· 34
6.1.1 Word2Vec ····· 34
6.1.2 GloVe ····· 34
6.1.3 fastText ····· 35
6.2. 연구 의의 및 후속 과제 제안 ····· 36

참 고 문 헌 ····· 39
Abstract ····· 43

표 목 차

<표 1> 각 코퍼스 별 정보와 통계값 ····· 16
<표 2> 임베딩 모델별 하이퍼 파라미터 ····· 19
<표 3> 평가 데이터의 실제값과 예측값에 따른 혼동 행렬 ····· 21
<표 4> fastText에서 코퍼스 별 형태소 분석기 사용/미사용 여부에 따른 감성 분석 최소/최대 정확도 ····· 24
<표 5> 코퍼스별 정보와 유사도 ····· 26
<표 6> Word2Vec 모델의 코퍼스 크기-문맥 별 F1 score (평균) ····· 29
<표 7> GloVe 모델의 코퍼스 크기-문맥 별 F1 score (평균) ····· 31
<표 8> fastText 모델의 코퍼스 크기-문맥 별 F1 score (평균) ····· 33

그 림 목 차

[그림 1] 목표 단어와 주변 단어(k=2), 슬라이딩 윈도우 ····· 3
[그림 2] CBOW 모델과 Skip-gram 모델의 학습 모식도 ····· 3
[그림 3] 재미잇다와 재미있다, 잼있다의 3-gram subword ····· 8
[그림 4] 필터링 전후의 코퍼스 내 영화 평점 분포 ····· 14
[그림 5] 전체 분석 순서도 ····· 23
[그림 6] 텍스트 벡터화 모식도 ····· 23
[그림 7] Word2Vec 모델의 학습 코퍼스 크기와 문맥에 따른
감성 분석 결과 ····· 27
[그림 8] GloVe 모델의 학습 코퍼스 크기와 문맥에 따른
감성 분석 결과 ····· 30
[그림 9] fastText 모델의 학습 코퍼스 크기와 문맥에 따른
감성 분석 결과 ····· 32

수 식 목 차

[수식 1] GloVe의 손실 함수 ····· 5
[수식 2] A, B 두 벡터의 코사인 유사도 ····· 20

제 1 장 서 론

최근 자연어 처리 연구들은 "더 큰 데이터, 더 강력한 컴퓨팅 파워, 그리고 더 복잡한 모델"을 추구하고 있다. 2018년에 발표되어 자연어 처리 연구의 흐름을 바꾼 구글의 BERT는 32억 개의 단어로 이루어진 있는 코퍼스를 사용하여 모델을 만들었다(Devlin, Chang, Lee, & Toutanova, 2018). 뒤이어 발표된 페이스북의 XLNet은 그보다 10배 더 큰 330억 개의 단어를 포함한 코퍼스를 사용하여 모델을 만들었다(Yang et. al., 2019). 심지어 가장 최근에 발표된 연구에서는 3,000억 개의 단어를 포함한 코퍼스로 1,750억 개의 파라미터를 학습시킨 모델을 만들었다(Tom et al., 2020). 이런 연구 경향은 학습에 사용하는 텍스트 데이터가 더 많고 다양할수록 어느 영역에도 적용할 수 있는 일반적인(generative) 모델을 만들 수 있을 것이란 믿음의 결과이다. 이렇게 크고 다양해진 데이터에 맞추어 모델은 복잡해지게 되었고, 복잡해진 모델을 구동시킬 수 있는 강력한 컴퓨팅 파워가 필요하게 되었다. 그래서 최근 자연어 처리 연구는 자체적으로 대용량 코퍼스를 구축할 수 있고, 이런 대량의 데이터를 처리할 장비가 있는 기업들이 주도하게 되었다.

그러나 언제나 더 큰 코퍼스가 더 나은 성능을 보장할까? 다수의 선행 연구에서는 더 큰 코퍼스가 항상 더 나은 성능을 보장하지 않았다(Kutuzov & Kunilovskaya, 2015; Lai, Liu, He, & Zhao, 2016; Kutuzov & Kunilovskaya, 2017; Dusserre & Padró, 2017). 본 논문에서는 큰 코퍼스의 사용이 항상 더 나은 성능에 이르는 것은 아님을 재확인하고, 나아가 학습 코퍼스와 과제 코퍼스 간의 유사도, 학습 코퍼스의 크기, 그리고 워드 임베딩 기법의 차이에 따라서 과제 정확도가 어떻게 영향을 받는지를 보고자 하였다.

제 2 장 관련 연구

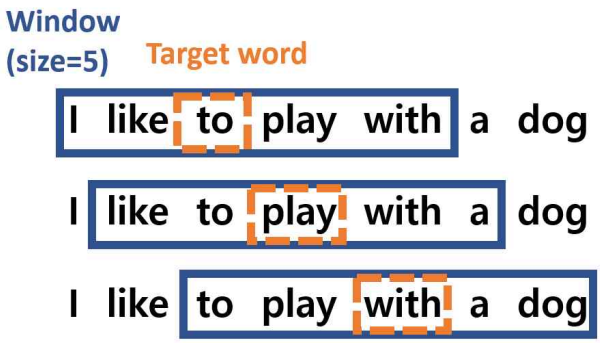
2.1. Word2Vec

Word2Vec은 Mikolov가 과거 딥러닝 기법을 자연어 처리에 적용한 연구(Bengio, Ducharme, Vincent, & Jauvin, 2003)를 개선하여 만든 워드 임베딩 알고리즘이다(2013a, 2013b). Word2Vec은 단어들을 임의의 차원의 벡터 공간에 배치하는 방법으로, 언어를 컴퓨터가 연산할 수 있는 숫자로 변환하는 방법이자, 각 단어의 의미를 N차원의 벡터 공간에 내포시키는 분산 표현(distributed representation) 기법이다.

Word2Vec은 한 단어의 의미가 그 단어 주변에서 나타나는 다른 단어들에 의해 추론될 수 있다는 가정하에, 목표 단어 주변에서 나타나는 단어들의 벡터를 통해 목표 단어의 벡터를 학습한다. 이때, 주변 단어로 목표 단어를 추론하며 학습하는 방식은 Continuous Bag Of Words(CBOW)라 하고, 목표 단어로 주변 단어들을 예측하며 학습하는 방식은 Skip-gram이라 한다.

Word2Vec의 임베딩 과정은 다음과 같다. 우선 문장을 여러 개의 작은 단위로 분해한다. 이때, 한국어의 경우는 띄어쓰기 단위로 분해할 수도 있고, 형태소 단위로 분해할 수도 있다. 이렇게 문장을 분해하는 함수를 '토크나이저(tokenizer)'라 하고, 나눠진 단위를 '토큰(token)'이라 한다. 워드 임베딩 기법은 토큰을 벡터로 변환하는 방법이다. 문장을 토큰으로 분해한 뒤에는 학습에 사용할 윈도우의 크기(window size)를 정한다. 윈도우는 학습 과정에서 목표 토큰의 주변 토큰을 어디까지 볼 것이냐에 대한 파라미터(parameter)이다. 목표 토큰은 벡터를 학습할 토큰, 주변 토큰은 목표 토큰을 학습할 때 참고하는 토큰이다. 목표 토큰과 그 전후 k개의 토큰을 윈도우라 하며, 이때 윈도우의 크기는 $2k+1$이다. 설정된 윈도우 내의 토

큰을 통해 CBOW 또는 Skip-gram 방식의 학습이 진행되고, 그 후 목표 단어가 다음 단어로 바뀌면서 윈도우도 함께 이동한다. 이렇게 윈도우가 이동하는 것을 슬라이딩 윈도우(sliding window)라 한다. 이 과정은 목표 단어가 문장의 끝이 되어 학습이 끝날 때까지 계속된다.



[그림 4] 목표 단어와 주변 단어(k=2), 슬라이딩 윈도우

[그림 5] CBOW 모델과 Skip-gram 모델의 학습 모식도

Word2Vec의 학습 방법은 Softmax Regression이다. CBOW 모델을 기준으로, 윈도우 내에 있는 목표 단어와 주변 단어에 대하여 주변 단어가 나타났을 때, 목표 단어가 나타날 확률 $P(word_{target}|word_{context})$를 최대화하는 방식이다. 즉, 주변 단어를 입력으로 했을 때, 목표 단어를 예측하는 네트워크의 가중치를 학습하는 과정이다. 임의로 초기화된 초기 가중치들은 이 학습 과정에서 새로운 값으로 갱신되는데, $P(word_{target}|word_{context})$를 최대로 하기 위해 목표 단어는 주변 단어 벡터들의 평균인 V_i와 유사한

방향으로 업데이트되고, V_i 근처에 있던 다른 단어들은 반대로 V_i와 다른 방향으로 조금씩 밀려나게 된다.

여기에 학습 속도와 효율성을 높이기 위해, subsampling과 negative sampling이란 기법을 더한다. Subsampling은 자주 등장하는 단어는 여러 번 학습 기회가 주어지므로 학습에서 일부 제외하는 방법이다. Negative sampling은 softmax regression을 계산할 때, 전체 코퍼스 단어 대신 윈도우 내에서 등장하지 않는 단어(negative sample)를 전체 코퍼스로 간주하고 softmax 확률을 계산하는 방법이다. 이러한 학습 과정을 거치면서 유사한 맥락에서 나타난 단어들은 벡터 공간상에서 같은 방향으로 모이게 되고, 이 벡터들은 언어의 문맥적 특성을 반영한 벡터가 된다.

2.2. GloVe

GloVe는 스탠퍼드 대학에서 개발한 임베딩 기법으로, 기존 Word2Vec과 잠재의미분석(Latent Semantic Analysis)의 단점을 보완하고 장점을 결합하고자 한 기법이다(Pennington, Socher, & Manning, 2014). 잠재의미분석은 각각의 문서에서 각 단어의 빈도수 행렬이라는 통계값에 특잇값 분해(Singular Value Decomposition) 기법을 적용하여 차원을 축소한다. 이렇게 차원이 축소되는 과정에서 단순한 빈도수로부터 각 단어의 잠재된 의미가 도출된다(김청택, 이태헌, 2002; 강범모, 2014). 그러나 각 단어 주변에 어떤 단어가 자주 등장하는지에 대한 맥락 정보는 사용하지 못한다. 반면, Word2Vec은 앞서 본 것처럼, 각 단어 주변에 나오는 단어들을 통해 해당 단어의 의미를 학습한다. 그러나 윈도우 밖의 단어, 즉, 문서 전체에서 단어의 빈도가 갖는 통계 정보는 반영하지 못한다. GloVe는 이 두 자연어 처리 방법론의 장점을 취합하고자 한 임베딩 기법이다.

GloVe는 Word2Vec처럼 목표 단어와 주변 단어를 포함하는 윈도우를 설정하여 주변 단어로부터 목표 단어의 맥락 정보를 학습한다. 이때, Word2Vec은 주변 단어들의 벡터를 통해 목표 단어의 벡터를 업데이트하

는 반면, GloVe는 동시 출현 행렬(co-occurrence matrix)의 정보를 이용한다.

$$J = \sum_{i,j=1}^{V} f(X_{ij})(w_i^T w_j + b_i + b_j - \log X_{ij})^2$$
$$where \quad f(x) = \begin{cases} (\frac{x}{x_{max}})^{\alpha} & \text{if } x < x_{max} \\ 1 & otherwise \end{cases} \quad \cdots \text{[수식 1]}$$

수식 1은 GloVe의 손실 함수 J이다. w_i, w_j는 각각 목표 단어와 주변 단어의 벡터이며, b_i, b_j는 각 단어의 편향(bias)이다. GloVe의 학습 목표는 윈도우 내에서 등장한 목표 단어와 주변 단어에 대하여, 목표 단어 벡터(w_i)와 주변 단어 벡터(w_j)의 내적이 두 단어의 동시 출현 빈도(co-occurrence, X_{ij})에 근접하게 하는 것이다. 이때 동시 출현 빈도는 코퍼스에서 슬라이딩 윈도우로 학습 코퍼스 집합을 만들었을 때, 각 학습 코퍼스 집합 내에서 두 단어가 동시에 나타난 빈도의 합이다. 두 단어 벡터의 내적은 코사인 유사도이므로, 두 단어의 동시 출현 빈도가 높을수록 두 단어의 벡터가 유사하게 학습된다.

대부분의 코퍼스에서는 사용되는 단어의 빈도가 지수 분포를 따른다. 자주 등장하는 단어는 빈도가 아주 높은 반면, 자주 등장하지 않는 단어일수록 그 빈도가 급격하게 줄어든다. 따라서, 저빈도 단어가 고빈도 단어와 동등하게 학습된다면, 이는 고빈도 단어의 학습에 저빈도 단어가 간섭하여 정확한 벡터 임베딩을 방해하게 된다. GloVe에서는 기준 출현 빈도 이하의 빈도수를 갖는 단어는 그 출현 빈도에 맞추어 손실 함수의 값을 감소시킴으로써 고빈도 단어의 학습에 간섭하는 것을 방지했다. 이때, 기준 출현 빈도는 $x_{\max}$, α는 감쇄 비율로 GloVe의 하이퍼 파라미터(hyper parameter)이다.

GloVe는 임베딩 과정에서 Word2Vec과 마찬가지로 윈도우 내에서 함께 등장하는 단어들에 대하여 벡터값을 학습하면서, 그 학습 과정에서는 코퍼스에서 해당 단어의 동시 출현 확률이라는 통계값을 반영한다. 이렇게 GloVe는 Word2Vec과는 다른 방식으로 단어들의 의미를 벡터 공간상에 임베딩 한다.

2.3. fastText

fastText는 n-gram 모델을 사용하여 언어의 형태학(morphology)적 구조를 반영한 Character(문자) n-gram 임베딩 모델이다(Bojanowski, Grave, Joulin, & Mikolov, 2017). n-gram이란 일련의 연속적인 단어나 문자를 n개씩 묶은 쌍으로 표현하는 방법을 말한다. 이때 n은 단어나 문자를 묶는 개수이며, 그 개수에 따라 unigram(1-gram), bigram(2-gram), trigram(3-gram) 등으로 표현한다. 예를 들어, `Apple`의 bigram(2-gram) 표현은 `Ap, pp, pl, le`이다. fastText는 토큰을 벡터 공간에 임베딩 하는 다른 워드 임베딩 기법과 달리, 토큰을 문자 단위로 쪼갠 subword를 임베딩 한다. 예를 들어, 다른 임베딩 기법들이 `where`이란 단어를 하나로 학습했다면, fastText는 이를 다수의 n-gram으로 쪼개어 학습한다. 단어를 여러 character n-gram 단위로 나눌 때, 최소 n이 m*in_n*, 최대 n이 *max_n*으로 (*min_n, max_n*)은 fastText의 하이퍼 파라미터이다. (*min_n, max_n*) = (3, 6)일 때, 우선 `where`은 단어의 맨 앞과 뒤에 `<`와 `>`이 붙어 `<where>`이 된다. 그다음 `<wh, whe, her, ere, re>`, `<whe, wher, here, ere>`, ... `<where, where>`, `<where>`으로 분해된다. 그리고 단어 `where`의 벡터는 이 모든 subword 벡터들의 합으로 표현한다. 이 방법은 크게 두 가지 장점을 갖는데, 하나는 언어의 형태적 유사성을 벡터 임베딩에 반영한다는 점이고, 다른 하나는 Out Of Vocabulary(OOV) 문제를 해결할 수 있다는 점이다. OOV는 워드 임베딩 모델이 학습 과정에서 등장하지 않았던 단어는 의미를 유추할 수 없는 문제이다.

where를 학습할 때 사용된 subword <wh는 why나 what을 학습할 때에도 나타난다. 즉, 형태적으로 유사한 단어가 의미적으로도 유사할 때, 서로 다른 단어를 임베딩 하는 과정에서 동일한 subword에 동일한 의미가 담기게 된다. 위 예시에서는 where: 어디, why: 왜, what: 무엇으로 서로 다른 뜻을 가지지만, 세 단어에서 동일하게 나타난 wh~는 의문이라는 의미를 내포하고 있고, 이는 임베딩 과정에서 <wh라는 subword에 반영된다.

기존 임베딩 기법들의 모델에서는 학습하지 않은 단어는 그에 대응하는 벡터가 없다. 따라서 학습할 때 나타나지 않았던 단어는 벡터로 변환할 수 없으므로, 모델을 활용할 때 해당 단어를 사용할 수 없게 된다. 그러나 fastText는 학습 과정에서 나타나지 않은 단어를 subword들의 조합으로 유추할 수 있다. 특히 이 방식은 오탈자에서도 의미를 추출할 수 있게 해준다. 예를 들어, wherre는 where의 흔한 오탈자이다. 이때, wherre의 3-gram subword는 <wh, whe, her, err, rre, re>로, where의 subword인 <wh, whe, her, ere, re>과 비교했을 때, 4개의 subword를 공유한다. fastText에서는 한 단어의 벡터를 subword 벡터들의 합으로 표현하므로, 학습 과정에서 등장하지 않은 오탈자라 하더라도 subword를 통해 어느 정도 의미를 복원할 수 있다.

fastText 임베딩을 한국어에 적용할 때에는 한국어만의 특징을 고려할 필요가 있다. 한국어는 교착어로, 하나의 어근에 다양한 접사가 붙으면서 수많은 의미가 파생된다. 재미있다와 재미없다를 예로 들면, 두 단어는 토큰화를 하지 않으면 서로 다른 단어로 인식되어 의미를 공유하지 못하게 된다. 반면, 이를 3-gram fastText 임베딩으로 구현하면 두 단어는 각각 <재미, 미있다, 있다>와 <재미, 미없다, 없다>로 학습 단위가 나누어진다. 재미있다와 재미없다 이 두 단어는 <재미를 공유하고, 미있다/미없다, 있다>/없다>가 서로 다르다. 따라서 두 단어는 재미의 의미를 반영하는 벡터는 유사하게, 있/없에 따라 달라지는 부분의 의미는 서로 다른 벡터로 임베딩 된다. 그리고 이 벡터들의 합으로 각각의 단어의 의미를 표상하기 때문에, 단어간의 의미를 더 잘 임베딩 할 수 있다. 따라서, 토큰화를 하지 않더라도 한국어의 구조적 특징을 반영한 임베딩을 할 수 있다. 또 다른 예시로, 재미

있는, 재미있고, 재미있지만은 재미있다와 모두 <재미, 재미있 토큰을 공유한다. 즉, fastText의 subword 임베딩을 이용하면 한국어에서 형태적으로 유사한 단어들이 의미적으로도 유사한 벡터를 공유하게 된다.

한편, 영어와 달리 한국어는 자모가 분리된 형태에서도 의미가 보존되는 경우가 있다. 예를 들어, 잡다, 잡히다, 잡혔다를 3-gram subword로 나누어보면, 이를 완성형 한글 형태로 임베딩 했을 때, 각각 <잡다, 잡다>, <잡히, 잡히다, 히다>, <잡혔, 잡혔다, 혔다>로 subword가 구성되어 공유하는 부분이 없다. 이때, 잡다는 ㅈㅏㅂㄷㅏ_처럼 초성-중성-종성으로 분해할 수 있으며, 여기서 _는 비어있는 종성을 의미한다. 다른 두 단어도 이처럼 자모를 분리하면, 잡히다는 ㅈㅏㅂㅎㅣ_ㄷㅏ_, 잡혔다는 ㅈㅏㅂㅎㅕㅆㄷㅏ_로 분해할 수 있다. 이를 다시 3-gram으로 보면, 세 단어는 <ㅈㅏ, ㅈㅏㅂ을 공유하며, 특히 의미적으로 더 유사한 잡히다와 잡혔다는 <ㅈㅏ, ㅈㅏㅂ, ㅏㅂㅎ으로 더 많은 subword를 공유한다. 따라서, 한국어로 fastText를 사용할 때에는 자모를 분리한 character n-gram 임베딩이 보다 효과적인 워드 임베딩 형식이 된다.

위 영어 예시와 마찬가지로, 오탈자 또한 자모를 분리하여야 subword를 통한 OOV 문제 해결이 가능해진다. 재미있다의 오탈자인 재미잇다와 비표준어인 잼있다는 완성형 한글로 subword를 구성하면 공유하는 부분이 거의 없어 의미의 추론이 어렵다. 그러나 자모 분리를 통해 subword를 구성하면 재미잇다와 재미있다, 잼있다의 subword는 각각 그림 5과 같이 구성된다.

재미있다	<ㅈㅐ	ㅈㅐ_	ㅐ_ㅁ	_ㅁㅣ	ㅁㅣ_	ㅣ_ㅇ	_ㅇㅣ	ㅇㅣㅆ	ㅣㅆㄷ	ㅆㄷㅏ	ㄷㅏ_	ㅏ>
재미잇다	<ㅈㅐ	ㅈㅐ_	ㅐ_ㅁ	_ㅁㅣ	ㅁㅣ_	ㅣ_ㅇ	_ㅇㅣ	ㅇㅣㅅ	ㅣㅅㄷ	ㅅㄷㅏ	ㄷㅏ_	ㅏ>
잼있다	<ㅈㅐ	ㅈㅐㅁ	ㅐㅁㅇ				ㅁㅇㅣ	ㅇㅣㅆ	ㅣㅆㄷ	ㅆㄷㅏ	ㄷㅏ_	ㅏ>

[그림 6] 재미잇다와 재미있다, 잼있다의 3-gram subword

재미있다와 재미잇다는 대부분의 subword를 공유하며, 잼있다 또한 그 구성 subword 9개 중 6개를 재미있다와 공유한다. 따라서 재미있다의 subword

임베딩을 통해 오탈자인 재미잇다와 비표준어인 잼있다가 학습 과정에서 나타나지 않았다 하더라도, 그 의미를 파악할 수 있게 된다. 이런 이유로 한국어 워드 임베딩에서 fastText를 사용하는 경우에는 자모를 분리하여 임베딩 해야 문자 단위 임베딩 기법의 장점을 보다 잘 활용할 수 있다.

2.4. 코퍼스 크기와 문맥에 관한 연구

워드 임베딩 모델에서 코퍼스의 크기와 문맥이 학습에 미치는 영향에 관한 다수의 연구가 있다(Kutuzov & Kunilovskaya, 2015; Kutuzov & Kunilovskaya, 2017; Dusserre & Padró, 2017; Lai, Liu, He, & Zhao, 2016).

Kutuzov와 Kunilovskaya(2015)는 잘 구성된 소규모 코퍼스와 웹에서 크롤링한 대규모 코퍼스를 워드 임베딩 기법으로 비교하였다. 여기서 소규모 코퍼스는 Russian National Corpus(RNC)로, 러시아 언어학자들이 러시아어를 가장 잘 대표할 수 있는 문서들을 선별하여 약 10여 년간 개발한 코퍼스이며, 2.3억 개의 토큰으로 구성되어 있다. 대규모 코퍼스는 다수의 러시아 웹 사이트에서 크롤링한 문서들(Web corpus)로, 18억 개의 토큰으로 구성되어 있다. 연구진들은 각 코퍼스에서 전처리 과정을 거쳐 RNC는 7천만 개의 토큰, Web corpus는 6.2억 개의 토큰을 가지도록 코퍼스를 구성하였다. 각 코퍼스를 Word2Vec CBOW를 통해 학습하고, 각 모델로 의미적 유사도(semantic similarity) 과제를 진행하였다. 그 결과, RNC를 사용한 모델은 동의어, 상의어, 하의어 등 정확한 의미적 관련도를 맞추는 과제에서는 더 뛰어났으나, 단순히 연관성만을 맞추는 문제에서는 Web corpus를 사용한 모델이 더 우세하였다. 연구진들은 이러한 결과를 복잡한 과제를 할 때는 빅 데이터보다 균형 잡히고 잘 구성된 코퍼스를 사용하는 것이 더 나은 성과를 보인다고 해석하며, 코퍼스의 크기가 아닌 질이 중요함을 논의하였다.

뒤이어 Kutuzov와 Kunilovskaya(2017)는 위 연구를 확장한 연구를 진행하였다. 소규모 코퍼스는 위 연구와 동일하게 RNC를 사용하였으며(약

2억 개의 단어) 대규모 코퍼스는 웹 크롤링으로 구성된 Araneum Russicum Maximum(약 100억 개의 단어)을 사용하였다. 각각의 코퍼스를 학습 데이터로 사용해 Word2Vec Skip-gram 모델을 학습시키고, 두 테스트 데이터 세트에 대하여 의미적 유사성과 관련도(semantic similarity and relatedness)를 비교하였다. 여기서 의미적 유사성(semantic similarity)은 A와 B 단어가 서로를 대체할 수 있는지(예: 커피-아메리카노)에 대한 정도이며, 의미적 관련도(semantic relatedness)는 A와 B 단어가 서로 의미적으로 연관되어있는지(예: 커피-머그잔)에 대한 정도이다. 분석 결과, 의미적 관련도를 맞추는 과제에서는 대규모 코퍼스를 사용한 모델이, 의미적 유사성을 맞추는 과제에서는 소규모 코퍼스를 사용한 모델이 우세한 정확도를 보였다. 연구진들은 앞선 연구와 마찬가지로 단순히 단어 간 관련성을 추론할 때에는 큰 데이터를 통해 학습하는 편이 낫지만, 보다 어려운 의미적 연관성 추론을 해야 할 때는 작더라도 잘 구성된 코퍼스를 사용하는 것이 낫다고 논의했다.

다른 코퍼스와 평가 방법으로 유사한 결론에 이른 연구도 있다(Dusserre & Padró, 2017). 연구자들은 작지만 영역 특수적인(domain specific) 코퍼스와 크고 포괄적인(generic) 코퍼스를 비교하였다. 전자는 통신판매 회사의 고객 리뷰 데이터(telecommunication corpus)이고, 557,676개의 단어로 이루어진 코퍼스이다. 후자는 프랑스 웹 사이트에서 크롤링을 통해 만든 코퍼스(frWac 코퍼스)로 약 16억 개의 단어를 포함하고 있다. 각각의 코퍼스를 Word2Vec Skip-gram 모델로 학습한 다음, 의미 유사도 과제를 시행했다. 이 과제의 데이터 세트는 통신판매 영역에서 seed word와 그와 동의어인 단어들로 구성되었다. 각 모델은 seed word가 입력되었을 때, 임베딩 벡터 공간에서 seed word 주변 일정 범위의 단어들을 seed word의 동의어로 간주하며, 이때 범위 내에 정확한 동의어를 포함했는지, 동의어가 해당 범위 밖에 있어 포함하지 못했는지로 정확도를 계산하였다. 즉, 임베딩 모델이 단어들 간의 의미를 정확히 학습했다면, 주어진 seed word 주변에 동의어들이 가깝게 분포하고 있어야 한다. 이 과

제에서 frWac 코퍼스로 만든 모델은 통신판매 코퍼스로 만든 모델보다 1/10 이상 낮은 정확도를 보였다. 심지어 통신판매 코퍼스를 워드 임베딩 기법을 사용하지 않고, 임의로 벡터 공간상에 배치한 모델의 정확도가 frWac 코퍼스로 만든 모델의 최고 정확도보다 높았다. 즉, 워드 임베딩 모델의 활용에서, 코퍼스의 크기는 늘리는 것보다 모델의 학습 코퍼스와 과제 코퍼스의 영역(domain)을 일치시키는 것이 중요하다는 결론을 내렸다.

코퍼스의 영역과 크기에 대한 다양한 비교를 수행한 연구도 있었다 (Lai, Liu, He, & Zhao, 2016). 이 연구에서 연구진들은 위키피디아 코퍼스(Wiki 코퍼스, 약 1억 개의 토큰)와 뉴욕 타임스 코퍼스(NYT 코퍼스, 약 12억 개의 토큰)를 사용한 모델로 각각 의미 유추 과제 성능을 비교했다. 그 결과, Wiki 코퍼스를 사용한 모델이 NYT 코퍼스를 사용한 모델보다 코퍼스 크기가 1/12로 작음에도 불구하고 더 높은 정확도를 보였다. 연구진은 Wiki 코퍼스의 내용이 NYT 코퍼스보다 더 포괄적인 지식을 다루고 있어, 의미 유추 과제에 더 유리하였기 때문에 이러한 결과가 나타났다고 해석하였다. IMDB 영화평 감성 분석 과제에서는 IMDB 리뷰 코퍼스(IMDB 코퍼스, 토큰 수 1,300만 개)와 Wiki 코퍼스와 NYT 코퍼스를 합친 코퍼스(W&N 코퍼스, 토큰 수 1,300만 개)를 일정 비율로 결합해 과제를 수행했다. 이때, 감성 분석의 대상이 IMDB 리뷰였기 때문에, IMDB 코퍼스를 영역 내 코퍼스(In domain corpus, 과제 코퍼스와 같은 분야의 코퍼스)로, W&N 코퍼스를 영역 외 코퍼스(Out domain corpus, 과제 코퍼스와 다른 분야의 코퍼스)로 구분하였다. 분석 결과, 이 과제에서 투입된 IMDB 코퍼스의 비율의 크기와 무관하게 (20%~100%) W&N corpus의 크기가 늘어날수록(0%→100%) 감성 분석의 정확도가 감소하는 경향을 발견하였다. 즉, 영역 특수적인 과제에서는 코퍼스의 크기보다 코퍼스의 분야가 과제 정확도 향상에 더 중요한 역할을 하며, 코퍼스의 크기 증가는 영역 내 코퍼스에 한해서만 정확도 향상에 기여했다.

이처럼 선행 연구들은 서로 다른 내용을 가진 두 개의 코퍼스를 비교하

였다. 그러나 이 두 코퍼스는 연구자의 주관에 의해 구분되었으며, 코퍼스의 내용이 서로 얼마나 다른지는 측정되지 않았다. 또, 대부분의 연구는 내용뿐 아니라 크기도 동시에 다른 두 코퍼스를 사용해 임베딩 모델의 평가 성능을 비교했고, 동일 코퍼스 내에서 학습에 사용되는 코퍼스의 크기가 달라질 때는 평가 성능이 어떻게 변하는지에 대한 언급은 없었다. Lai 등의 연구(2016)에서는 이 부분을 고려했으나, 서로 다른 두 영역의 코퍼스를 임의로 혼합하여 사용했고, 영역 외 코퍼스만을 사용한 분석이 없었으며, 큰 영역 외 코퍼스와 작은 영역 내 코퍼스를 사용한 모델 간의 성능 비교가 없었다. 평가 과정도 의미적 유사도 평가가 주를 이루었는데, 영어로 된 평가 데이터 세트를 번역해서 쓰거나, 모델 간 성능 비교에서 최고 정확도가 30%에 이르지 못하는 등 성능 평가 방식의 엄밀함이 부족한 모습을 보였다. 그리고 각 연구들은 하나의 워드 임베딩 알고리즘만을 사용하여 평가를 진행하였다.

본 연구에서는 이런 선행 연구들의 부족한 부분을 개선하여 다음과 같은 분석 방법을 택했다. 우선, 모델의 학습 코퍼스를 4개로 구성하였다. 각 코퍼스는 연구자의 정성적 평가에 의해 선택된, 과제 코퍼스와 매우 높은/높은/낮은/매우 낮은 유사도를 갖는 코퍼스이다. 그리고 각 코퍼스 간 유사도를 측정하여 유사도에 대한 연구자의 평가를 정량적으로 검증한다. 또한, 학습 코퍼스를 일정한 크기로 샘플링하여 서브 코퍼스(sub corpus)를 만들고 이를 학습 데이터로 사용해 각 모델의 과제 성능을 측정한다. 이 과정을 통해 코퍼스의 크기에 따른 과제 정확도를 분석한다. 평가 과제는 워드 임베딩 모델의 성능을 수치적으로 잘 나타낼 수 있는 감성 분석을 평가 과제로 한다. 끝으로 가장 많이 쓰이는 세 가지 워드 임베딩 기법인 Word2Vec, GloVe, fastText로 동일한 분석을 수행하여, 임베딩 기법에 따라 분석 결과에 차이가 나타나는지 확인한다.

제 3 장 분석 도구

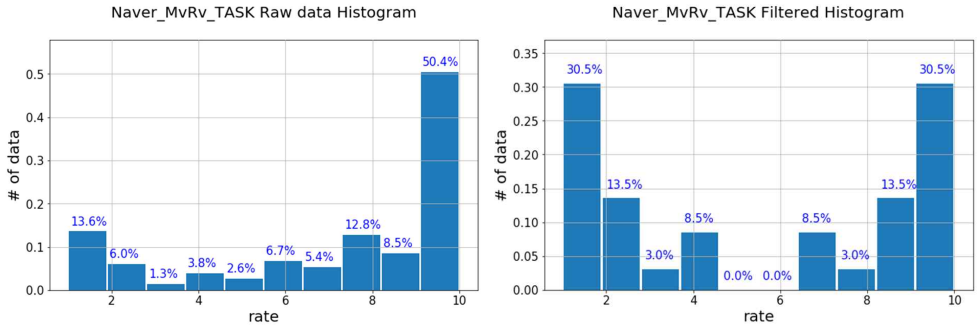
3.1. 분석 대상 코퍼스

본 연구에서 사용된 코퍼스는 총 5가지이다.

1. 과제 코퍼스: *Naver_MvRv_TASK*

감성 분석 평가 대상 코퍼스는 네이버 영화 리뷰 데이터를 사용하였다. 감성 분석의 경우, 과제 평가에 텍스트와 라벨(Positive/Negative)이 동시에 있어야 한다. 따라서, 특정 영화에 대한 리뷰 텍스트와 평가 점수가 있는 네이버 영화 리뷰 데이터를 본 연구에서 감성 분석의 과제 코퍼스로 선정하였다. 데이터는 네이버 영화 네티즌 평점 페이지[1)]에서 2019년 12월 20일~2020년 5월 31일까지의 리뷰를 수집하여 구성하였다. 데이터를 확인한 결과, 1~10점으로 나누어진 평점에서 약 절반의 리뷰가 10점으로 평점이 부적 편포 되어있음을 확인하였다. 따라서, 감성 분석에서의 편향을 방지하고자 데이터의 평점 균형을 맞추었다. 1~4점을 부정, 7~10점을 긍정으로 정한 뒤, 전체 리뷰 수가 6만 개가 되도록, 1~4점과 7~10점의 리뷰를 각각 3만 개씩 추출하였다. 이때, 어느 한 점수에 리뷰가 편중되지 않도록 1, 2, 3, 4점의 리뷰 수에 맞추어 각각 10, 9, 8, 7점의 리뷰 수를 동일하게 맞추었다.

1) https://movie.naver.com/movie/point/af/list.nhn



[그림 7] 필터링 전후의 코퍼스 내 영화 평점 분포

2. 모델 코퍼스 1. : *Naver_MvRv_MODEL*

이 코퍼스는 다수의 한국어 감성 분석 연구에서 사용된 Naver sentiment movie corpus v1.0의 데이터[2)]이며, 과제 코퍼스와 동일한 출처의 영화 리뷰이다. 자료의 출처에서 제공하는 전처리 된 데이터는 1~3점을 부정, 10점을 긍정으로 분류한 데이터였기 때문에, 과제 코퍼스와 동일한 방식을 유지하기 위해 전처리 전의 원본 데이터를 사용하여 코퍼스를 새롭게 구성하였다. 과제 코퍼스와 중복되지 않는 리뷰를 선별하였으며, 과제 코퍼스와 마찬가지로 코퍼스 내에 평점 간 균형이 유지되도록 총 5만 개의 리뷰를 추출하였다.

3. 모델 코퍼스 2.: *WebUser_MvRv*

이 코퍼스는 온라인 영화 커뮤니티에 올라온 유저들의 영화평 리뷰를 크롤링 한 데이터이다. 데이터는 2016년 01월 18일~2020년 5월 31일까지의 게시물을 수집하였다. 위 두 코퍼스의 출처인 네이버 영화 리뷰와 비교하여 평점 정보가 없으며, 리뷰의 길이가 다소 긴 특징이 있다. 또한, 네이버 영화 리뷰는 한 줄 평으로 되어있으나, 이 코퍼스의 데이터는 제목과 글,

2) https://github.com/e9t/nsmc/

댓글이 있는 게시물의 형태인 것이 차이점이다. 그러나 위 두 코퍼스와 마찬가지로 인터넷 유저들이 평소 온라인에서 사용하는 용어를 사용해 작성한 일반인의 영화 감상평이라는 유사한 맥락을 가진다. 코퍼스를 구성할 때에는 각 게시물의 댓글은 제외하고 제목과 본문을 이어서 구성하였다.

4. 모델 코퍼스 3.: *Movie_News*

모델 코퍼스 3은 다수의 국내 언론사의 영화 기사를 크롤링 한 코퍼스이다. 데이터의 내용은 영화 개봉 소식, 영화 관련 이슈, 배우・감독 등 영화 관계자 인터뷰, 영화제 수상 소식 등이다. 위 코퍼스들과 비교하여, 보다 긴 문서 길이, 정확하게 지켜진 맞춤법과 띄어쓰기가 특징이다. 또한, 영화라는 주제는 위 코퍼스들과 공유하지만, 영화 리뷰와는 다소 거리가 먼 내용으로 구성되어 있다. 크롤링 데이터에서 뉴스 제목과 본문을 연결하여 코퍼스를 구성하였으며, 본문 중간 광고, 기자명, 사진의 캡션 등은 코퍼스에서 제외하였다.

5. 모델 코퍼스 4.: *KorQuAD*

KorQuAD는 LG CNS가 제공하는 한국어 질의응답 평가 데이터 세트이다(임승영, 김명지, 이주열, 2018)[3]. 이는 스탠퍼드 대학의 SQuAD의 한국어 버전으로, 제시된 한국어 위키피디아 페이지에 대한 질의응답 세트이다[4]. 이 코퍼스는 데이터 세트에 사람의 검수가 반영되어있어 단순 크롤링 데이터에 비해 고품질의 코퍼스이다. 이 코퍼스는 한국어 위키피디아를 기반으로 하기 때문에, 영화와는 무관한 내용이 대부분이며, 특히 과제 코퍼스의 맥락인 영화 리뷰와는 더더욱 거리가 먼 내용으로 구성되어 있다. 본 분석에서는 KorQuAD가 제공하는 training, test 데이터 세트를 모두 코퍼스로 사용하였으며, 표나 주석을 제외한 위키피디아 본문과 질의응답을 합

3) The Korean Question Answering Dataset, https://korquad.github.io/
4) Stanford Question Answering Dataset, https://rajpurkar.github.io/SQuAD-explorer/

쳐 코퍼스로 구성하였다.

<표 1> 각 코퍼스 별 정보와 통계값

코퍼스 이름	코퍼스 출처	총 문서 개수	총 토큰 개수	문서 당 평균 토큰 수
Naver_MvRv_TASK	네이버 영화 리뷰	60,000	2,302,966	38.38
Naver_MvRv_MODEL	네이버 영화 리뷰	50,000	1,233,481	24.67
WebUser_MvRv	영화 전문 인터넷 커뮤니티의 영화 리뷰 게시물	44,760	19,631,183	437.83
Movie_News	여러 신문사의 영화 카테고리 뉴스	30,857	21,046,011	313.38
KorQuAD	한국어 위키피디아	47,968	66,521,419	1386.79

3.2. 코퍼스 간 유사도

본 분석에 사용되는 코퍼스는 각각 과제 코퍼스와 서로 다른 유사도를 가지도록 구성되었다. 각 코퍼스는 저마다의 출처와 그에 따른 고유한 문맥을 가진다. 과제 코퍼스는 네이버 영화평 페이지에서 크롤링 한 데이터이다. 이 데이터는 한 줄 평 또는 단문 위주로 영화를 평가하는 일반 사람들의 영화평이다. *Naver_MvRv_MODEL* 코퍼스는 *Naver_MvRv_TASK* 코퍼스와 동일한 출처에서 크롤링 한 데이터이다. 코퍼스의 출처가 동일하기 때문에, 과제 코퍼스와 매우 높은 맥락 유사도를 보일 것이다. *WebUser_MvRv* 코퍼스는 영화 전문 인터넷 커뮤니티에서 영화 리뷰 게시판의 게시물을 크롤링 한 데이터이다. 이 코퍼스는 *Naver_MvRv_*

TASK 코퍼스와는 출처가 다른 웹 사이트의 리뷰이면서, 게시물의 형태이기 때문에 *Naver_MvRv_TASK* 코퍼스와 다소간의 차이가 있을 것으로 보인다. 그러나 일반 사람들의 영화평이라는 맥락은 두 코퍼스가 공유하므로, 유사도는 높을 것이라 예상했다. *Movie_News* 코퍼스는 다수의 신문사에서 수집한 영화 관련 뉴스 기사이다. 각 신문사의 영화 카테고리에 있는 기사만을 수집하였으며, 그 내용은 신작 개봉 소식, 감독 및 배우의 인터뷰, 영화제 및 수상 소식 등 등이다. 영화라는 맥락은 *Naver_MvRv_TASK* 코퍼스와 공유하지만, 전문 기자가 작성한 글이라는 점, 영화 리뷰가 아닌 영화 관련 기사라는 점에서 유사도가 다소 낮을 것으로 예상하였다. 끝으로, *KorQuAD* 코퍼스는 한국어 위키피디아를 바탕으로 구성된 코퍼스이다. 이 코퍼스는 *Naver_MvRv_TASK* 코퍼스와 비교했을 때, 영화와는 상관이 없는 내용으로 구성되어 있으며, 그 형태에서도 일반인의 영화평과 백과사전이라는 차이를 보인다. 따라서 유사도가 가장 낮을 것이다.

제 4 장 분석 환경 및 방법

4.1. 모델의 설정

Word2Vec, GloVe, fastText는 모두 워드 임베딩 기법이기 때문에, 주요 하이퍼 파라미터들을 공유한다. 이 하이퍼 파라미터는 명확한 결정 기준이 없기 때문에, 여러 사전 연구들과 예비 연구를 통해 값을 선정했다. 윈도우 사이즈는 클수록 주제적 유사성, 작을수록 기능적이고 통사적인 유사성에 따라 벡터를 임베딩 한다는 연구(Goldberg, Y, 2016)와 다양한 윈도우 사이즈 중 윈도우 사이즈가 10일 때 대체로 성능이 가장 좋다는 연구(최상혁, 설진석, 이상구, 2016; 강형석, 양장훈, 2019; 이다빈, 최성필, 2019)를 참조하여 10으로 정하였다. 임베딩 차원은 의미를 잘 담으면서도 학습 시간이 너무 길어지지 않도록 300으로 설정하였다(Pennington. et al., 2014; 강형석, 양장훈, 2019; 이다빈, 최성필, 2019; 조현수, 이상구, 2017; Heu, 2019). 반복 횟수(iteration, epochs)는 충분히 학습하되, 과적합이 일어나지 않도록 30으로 정하였다(Lai. et al., 2016). 최소 등장 횟수는 5회로 하였다. GloVe의 하이퍼 파라미터인 $x_{\max}$는 GloVe 논문에서 제시한 기본값 100으로 설정하였으며(Pennington. et al., 2014), fastText의 하이퍼 파라미터인 (*min_n, max_n*) 은 fastText를 한국어에 적용한 선행 연구(조현수, 이상구, 2017)에 따르면, (1, 3)이 최적의 결과로 나타났다. 그러나 이는 자모 분리를 하지 않은 완성형 단어 단위 모델의 최적 하이퍼 파라미터이다. 한국어의 자모를 분리하면 문자의 길이가 3배로 길어지기 때문에, 이를 반영하여 라이브러리의 기본값인 (3, 6)에서 (3, 9)로 변경하였다. 끝으로, 위에 언급하지 않은 하이퍼 파라미터들은 모두 라이브러리의 기본값을 사용했으며, 모든 모델의 학습 방법은 CBOW보다 더 나은 성능을 보이는 것으로 알려진 Skip-gram 방식을 사용하였다.

<표 2> 임베딩 모델별 하이퍼 파라미터

임베딩 모델	윈도우 크기	반복 횟수	임베딩 차원	학습 방법	학습률 (learning rate)	비고
Word2Vec					0.025 (*default*)	-
GloVe	10	30	300	Skip-gram	0.05 (*default*)	$x_{\max} = 100$ $\alpha = 0.75$ (*default*)
fastText					0.025 (*default*)	$(min_n,\ max_n)$ $= (3,\ 9)$

분석에서 Word2Vec과 fastText는 python의 gensim 라이브러리를 사용하였으며, GloVe는 glove_python 라이브러리를 사용하였다. 코퍼스의 토큰화는 konlpy 라이브러리의 mecab 토크나이저를 이용해 진행되었다.

4.2. 분석 방법

4.2.1 코퍼스 간 문맥 유사도 측정

본 논문에서는 임베딩 모델을 생성할 때 투입하는 코퍼스의 문맥과 크기에 따라 과제 정확도가 달라진다는 것을 검증하고자 한다. 그렇기 때문에 서로 다른 코퍼스 간의 문맥 유사도를 정량적으로 평가할 척도가 필요하다. 본 연구에서는 그 방법으로 코사인 유사도(cosine similarity)를 사용하였다.

코사인 유사도는 두 벡터 사이의 코사인값을 두 벡터의 유사도로 간주하는 측정법이다. 코사인 유사도는 [수식 2]와 같이 나타낼 수 있다.

$$\cos\theta = \frac{A \cdot B}{|A||B|} \quad \cdots \text{[수식 2]}$$

코사인값은 −1~1 사이의 값을 가지고, 각도가 작을수록 값이 커진다. 두 벡터의 방향이 가까울수록 두 벡터 간 각도는 작아지고, 두 벡터 사이의 코사인값은 1에 가까워진다. 따라서 계산된 값이 1에 가까운 값을 가질수록 더 높은 유사도를 의미한다.

워드 임베딩의 경우 각 단어들이 N차원의 벡터로 구성되기 때문에, 각 단어들, 혹은 문서 간의 유사도를 코사인 유사도로 측정할 수 있다. 이 방식은 어휘의 의미 표상을 비교할 때 자주 사용되는 척도이다(Finkelstein et al., 2002; Hill, Reichart, & Korhonen, 2015). 본 연구에서도 코퍼스 간 문맥(context)의 유사도를 객관적으로 측정하기 위해 코사인 유사도를 사용하였다. 본 분석에서 사용되는 다섯 개의 코퍼스와 독립적이면서도 각 코퍼스의 서로 다른 문맥을 모두 포괄할 수 있는 큰 임베딩 모델이 필요했다. 이를 위해 나무위키 덤프 데이터를 사용하여 Word2Vec 모델을 생성하였다. 여기서 사용된 나무위키 덤프 데이터의 코퍼스 크기는 이후 분석의 기준값이 되는 *Naver_MvRv_MODEL* 코퍼스의 약 520배였다. 각 코퍼스를 하나의 문서로 간주하여, 코퍼스 내 모든 토큰의 벡터값을 평균 내어 하나의 벡터로 변환하였고, 이 벡터로 코사인 유사도를 계산하였다. 계산할 때 모든 코퍼스의 맥락과 무관하게 공통적으로 등장하는 불용어를 제거하기 위해, mecab 토크나이저 기준을 적용하여 체언, 용언, 독립언, 부호, 그리고 미분류 토큰만을 사용하였다. 체언은 명사, 용언은 동사와 형용사를 포함하는 형태소이며, 독립언은 감탄사와 함께 일부 이모티콘과 같은 단어들, 부호에는 특수 문자, 그리고 미분류 토큰은 토크나이저가 토큰화하지 못했지만, 인터넷 게시물에서 자주 발견되는 어구들이 포함되었다.

4.2.2 과제 정확도 평가

본 분석에서 사용하는 과제 평가는 True/False로 이루어진 감성 분석으로, 이진 분류 과제에 해당한다. 따라서 평가 데이터의 실제값과 모델의 예측값에 따라 표 3과 같은 혼동 행렬(confusion matrix)을 그릴 수 있다.

<표 3> 평가 데이터의 실제값과 예측값에 따른 혼동 행렬

		실제값	
		True	**False**
예측값	**True**	True Positive (TP)	False Positive (FP)
	False	False Negative (FN)	True Negative (TN)

여기서, 일반적으로 모델의 성능을 평가할 때 쓰는 정확도(accuracy)는 *맞춘 개수 / 전체 변인 수* 이기 때문에 위 행렬에서 $\frac{TP+TN}{TP+TN+FP+FN}$ 에 해당한다. 이 수치는 전체 데이터에서 True와 False 데이터의 비율, 모델의 TP와 TN 비율이 고려되지 않는 수치이다.

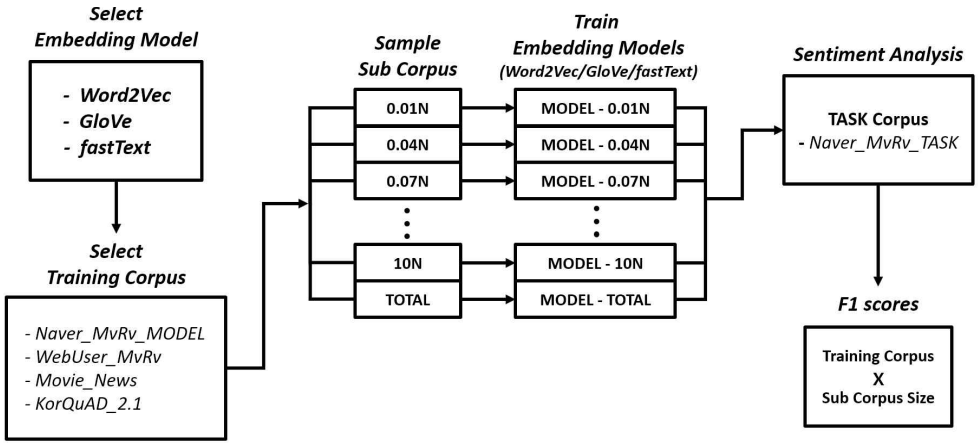
F1 score는 F-measure라고도 하며, 정밀도(precision)와 재현율(recall)의 조화 평균(harmonic mean)이다(Chinchor, 1992). 정밀도와 재현율은 각각 $precision = \frac{TP}{TP+FP}$, $recall = \frac{TP}{TP+FN}$ 으로 *<정답이라 예측한 것 중, 실제 정답의 개수>와 <전체 정답 중, 정확히 예측한 경우의 개수>*를 의미한다. 이때, F1 score는 $F1\ score = \frac{2 \times precision \times recall}{precision + recall}$ 이다(Sasaki, 2009). F1 score는 정밀도와 재현율을 모두 고려하면서, 두 측정치 중 작은 쪽으로 평균을 계산하는 조화 평균 측정치이기 때문에 분류 과제에서 라벨 간 비율이 불균형하더라도 사용할 수 있는 정확도 지표이다(Singhal, Y., Jain, A., Batra, S., Varshney, Y., & Rathi, M, 2018). 본 연구에서 사

용하는 과제 코퍼스의 라벨은 긍정:부정의 비율이 1:1로 맞추어져 있으나, 랜덤 샘플링을 통해 서브 코퍼스를 구성하는 과정에서 데이터가 불균형하게 표집 될 가능성이 있다. F1 score는 하나의 라벨(긍정 또는 부정)에 대한 측정치이기 때문에, 이를 보정하고자 긍정/부정 라벨 각각의 F1 score를 구한 뒤, 각 라벨에 해당하는 데이터의 수에 맞추어 두 F1 score의 가중 평균(weighted average)을 계산해 최종 F1 score로 사용하였다.

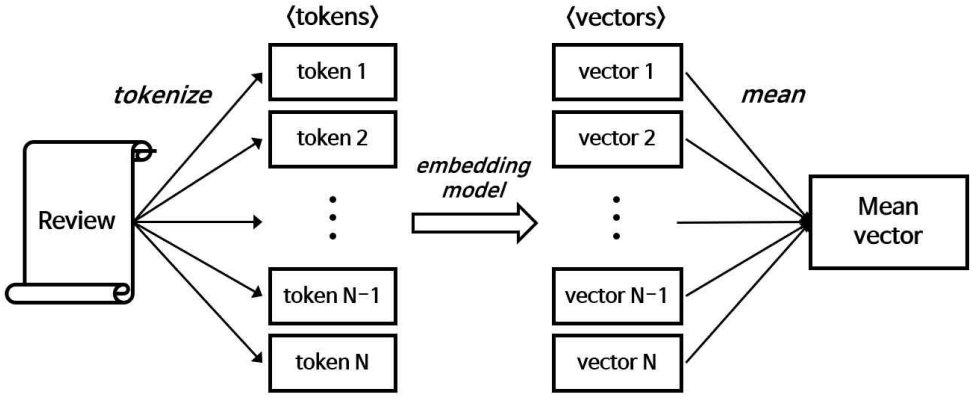
4.2.3 워드 임베딩 모델을 이용한 감성 분석

분석에 들어가기에 앞서, 서브 코퍼스 샘플링의 기준을 정하였다. 기준이 되는 값은 *Naver_MvRv_MODEL* 코퍼스의 전체 토큰 개수이며, 그 값은 1,233,481이다. 이 값을 1N으로 두고, 전체 코퍼스에서 1N에 대하여 일정 비율로 서브 코퍼스를 랜덤 샘플링 한다. 이때 샘플링 비율은 [0.01N, 0.04N, 0.07N, 0.1N~0.9N, 1N~10N]이며, 마지막에 코퍼스의 전체 데이터를 추가한다. 예를 들어, *Movie_News* 코퍼스의 서브 코퍼스들의 크기는 [0.01N, 0.04N, 0.07N, 0.1N, 0.2N, 0.3N, 0.4N, 0.5N, 0.6N, 0.7N, 0.8N, 0.9N, 1N, 2N, 3N, 4N, 5N, 6N, 7N, 8N, 9N, 10N, 17.06N]이다.

본 분석은 다음과 같은 단계를 거쳐 진행된다. 우선 분석에 사용할 워드 임베딩 모델을 선택한다. 그다음, 각 학습 코퍼스의 서브 코퍼스를 구성한다. 그리고 생성된 각각의 서브 코퍼스를 사용하여 워드 임베딩 모델을 생성한다. 마지막으로 생성된 각각의 모델을 사용해 감성 분석을 시행해, 각 <학습 코퍼스 종류 x 서브 코퍼스 크기>마다 F1 score를 기록한다(그림 5). 감성 분석은 로지스틱 회귀분석을 통해 진행되었다. 토크나이저를 이용해 리뷰를 토큰화하고, 임베딩 모델을 사용하여 각각의 토큰을 벡터로 변환하였다. 그리고 변환한 벡터들을 평균하여 하나의 벡터로 만들고, 이 평균 벡터를 입력으로 하여 로지스틱 회귀분석을 수행하였다(그림 6).



[그림 8] 전체 분석 순서도



[그림 9] 텍스트 벡터화 모식도

fastText의 경우, 토큰화를 한 뒤, 자모 분리를 하여 분석한 모델이 더 성능이 낫다는 선행 연구(조현수, 이상구, 2017)가 있었다. 본 연구에서도 토큰화를 적용한 모델과 하지 않은 모델을 각각 테스트해 본 결과, mecab 형태소 분석기가 단어를 잘못 분할하는 경우가 있음에도 불구하고, 토큰화를 한 모델의 성능이 더 높았다. 따라서 fastText 또한 형태소 분석기를 사용해 분석을 시행했다.

<표 4> fastText에서 코퍼스 별 형태소 분석기 사용/미사용 여부에 따른 감성 분석 최소/최대 정확도

학습 코퍼스	형태소 분석기 사용 여부	F1 score	
		min	*max*
Naver_MvRv_MODEL	O	**0.7569**	**0.8215**
	X	0.6509	0.7921
WebUser_MvRv	O	**0.7224**	**0.8159**
	X	0.6213	0.7841
Movie_News	O	**0.7189**	**0.7942**
	X	0.6135	0.7413
KorQuAD	O	**0.7005**	**0.7811**
	X	0.5845	0.7237

분석은 각 임베딩 기법에서 모든 학습 코퍼스의 모델 생성과 감성 분석이 끝나는 것을 1회로 하였고, 각 임베딩 기법 별 20회의 분석을 진행하였다. 성능 평가는 매 분석의 F1 score 평균과 표준편차를 이용하였다. 끝으로 모든 분석은 분석 환경을 통제하기 위해 Google Colab에서 진행되었다.

4.3. 예상 결과

임베딩 모델은 학습에 사용하는 코퍼스가 크고 다양할수록 토큰 간의 관계에 대한 정보 즉, 문맥이 잘 반영된 벡터를 생성한다. 따라서 학습 코퍼스의 크기가 작을 때에는 다양한 문맥을 학습할 수 없어 과제 정확도가 낮게 나타나고, 학습 코퍼스의 크기가 커질수록 벡터들이 보다 정확한 문맥을 내포하도록 임베딩 되어 과제 정확도(F1 score)가 높아질 것으로 예상한다.

그러나 이러한 학습 코퍼스의 크기가 정확도에 미치는 영향은 과제 코퍼스와 학습 코퍼스 간 유사도의 영향력보다 작을 것으로 예상한다. 앞서

살펴본 선행 연구들은 학습-과제 코퍼스 간 문맥 유사도가 학습 코퍼스의 크기보다 중요함을 시사하였다. 4개의 코퍼스는 각각 학습 코퍼스의 크기가 증가할 때 정확도가 증가하는 정도가 다르게 나타날 것이며, 이는 그래프에서 기울기에 해당한다. 따라서 각 코퍼스의 정확도는 학습 코퍼스의 크기가 동일할 때, 학습-과제 코퍼스 간 유사도 순(*Naver_MvRv_MODEL* > *WebUser_MvRv* > *Movie_News* > *KorQuAD*)으로 배열될 것이고, 기울기 또한 학습-과제 코퍼스 간 유사도 순으로 크게 나타날 것이므로 그래프가 교차하지 않고 정확도 순위를 유지하는 형태로 나타날 것이다.

한편, 동일한 학습 코퍼스 크기를 가진다고 하더라도 학습 코퍼스를 구성할 때에는 항상 전체 코퍼스에서 랜덤 샘플링을 하였으므로, 모든 모델은 서로 다른 학습 코퍼스를 가지게 된다. 이때 학습에 사용한 코퍼스의 문맥에 따라 서로 다른 임베딩 벡터가 생성된다. 어떤 코퍼스로 학습했느냐에 따라 서로 다른 임베딩 벡터를 가지게 되고, 서로 다른 임베딩 벡터는 모델 간 과제 정확도의 편차를 일으킨다. 이는 학습 코퍼스보다 과제 코퍼스의 문맥이 다양하기 때문에 발생하는 것으로, 학습하지 못한 문맥에 대한 평가가 부정확하기 때문에, 임의로 추출한 학습 코퍼스의 문맥이 다양한 정도에 따라 모델의 정확도가 달라지게 되는 것이다. 학습 과정에서 비교적 다양한 문맥을 고르게 학습한 모델은 상대적으로 높은 정확도를 보이지만, 편향된 문맥에 대해 학습한 모델은 학습하지 못한 문맥의 리뷰에 대해서는 낮은 예측 결과를 보일 것이다. 모델의 학습 코퍼스가 커지게 되면 학습 과정에서 다양한 문맥의 텍스트를 학습하게 되고, 모델 간 과제 정확도의 편차가 작아지게 된다. 결론적으로, 학습 코퍼스의 크기가 증가함에 따라, 과제 정확도의 표준편차는 줄어들 것이다.

제 5 장 결 과

5.1. 코퍼스 간 문맥 유사도

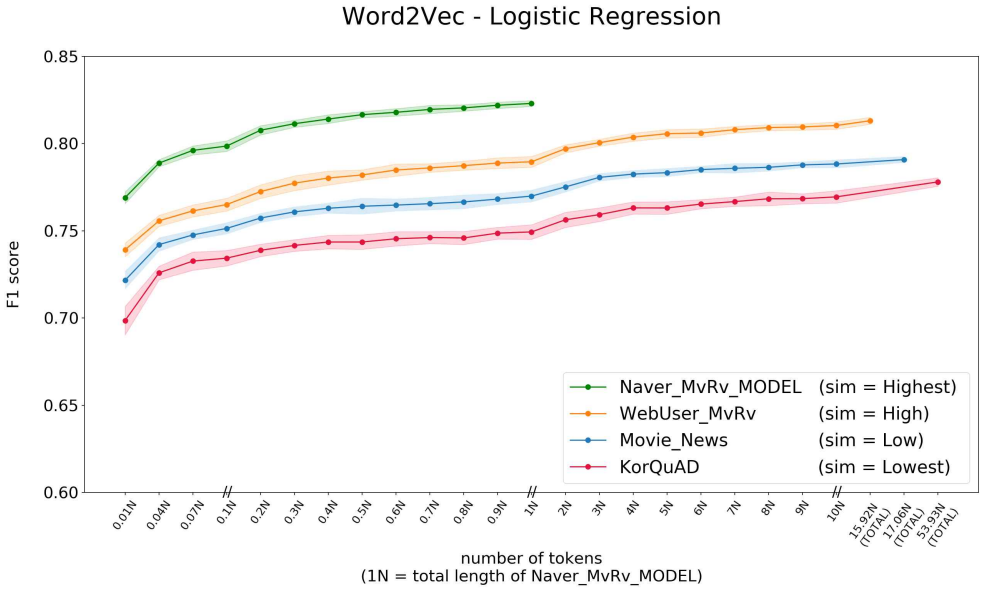
코퍼스 간 문맥 유사도는 표 4의 마지막 열에 제시되어 있다. 문맥 유사도 측정은 4.2.1에서 설명한 코사인 유사도로 측정하였다. *Naver_MvRv_TASK* 코퍼스와 같은 출처의 *Naver_MvRv_MODEL* 코퍼스가 99.82%로 가장 높은 유사도를 보였고, 그 뒤로 같은 맥락을 공유하지만 다른 출처의 *WebUser_MvRv* 코퍼스가 98.71%의 높은 유사도를, 영화라는 맥락은 공유하지만, 리뷰가 아닌 신문 기사인 *Movie_News* 코퍼스가 94.06%로 다소 낮은 유사도를 보였다. 끝으로, 영화와는 관계없는 데이터인 *KorQuAD* 코퍼스는 예상했던 대로 87.86%로 가장 낮은 유사도를 보였다.

<표 5> 코퍼스별 정보와 유사도

코퍼스 이름	코퍼스 출처	코퍼스 내용 (문맥)	예상 유사도	실제 유사도
Naver_MvRv_TASK	네이버 영화 리뷰	일반 사람들의 한 줄 영화평	(기준값)	1
Naver_MvRv_MODEL	네이버 영화 리뷰	일반 사람들의 한 줄 영화평	**가장 높음**	**0.9983**
WebUser_MvRv	영화 전문 인터넷 커뮤니티의 영화 리뷰 게시물	일반 사람들의 영화평 게시물	높음	0.9871
Movie_News	여러 신문사의 영화 카테고리 뉴스	영화 관련 뉴스 (신작, 영화제, 인터뷰 등)	낮음	0.9406
KorQuAD	한국어 위키피디아	한국어 위키피디아 본문 + 질의응답	**가장 낮음**	**0.8786**

5.2. 감성 분석

5.2.1 Word2Vec



[그림 10] Word2Vec 모델의 학습 코퍼스 크기와 문맥에 따른 감성 분석 결과 (실선은 평균 F1 score, 주변 영역은 표준편차)

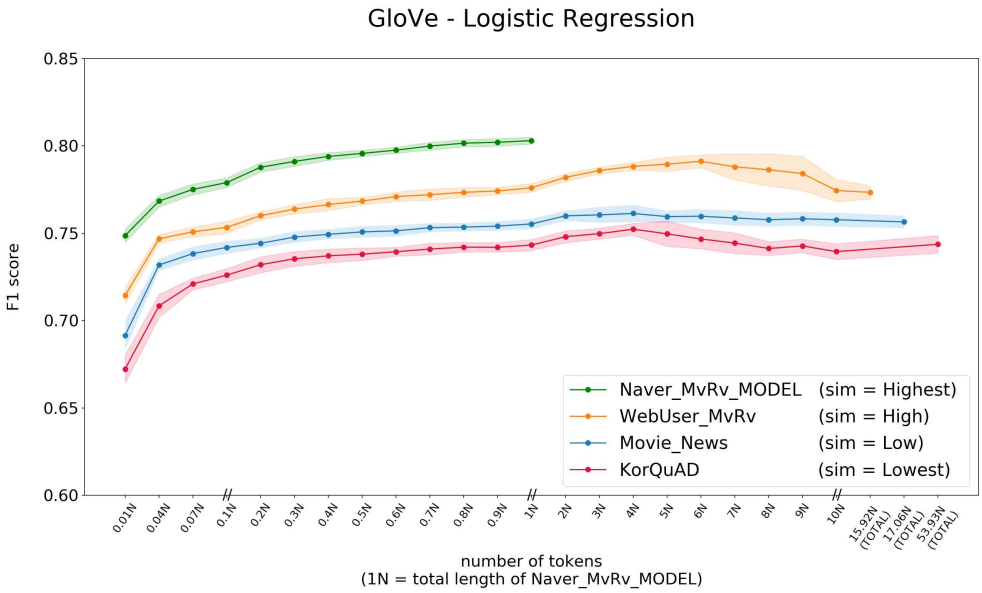
Word2Vec의 코퍼스 문맥-크기 별 감성 분석 결과는 그림 10과 같다. 본 분석에서 사용한 서브 코퍼스의 크기는 0.01N에서 53.93N으로 상당히 광범위하다. 이를 일정한 크기로 나타내면, 서브 코퍼스의 크기가 작은 구간은 F1 score가 과도하게 밀집되어 나타나고, 서브 코퍼스의 크기가 큰 구간은 F1 score 간 간격이 너무 넓게 떨어져 나타난다. 따라서 의미 있는 시각화를 위해 그래프의 x축 구간을 임의로 분할하여 나타냈다. 축 눈금 간격의 크기가 바뀌는 위치에는 //으로 표기를 하였다. 문맥별 감성 분석 정확도를 보면, 코퍼스별로 표 5의 유사도 순서대로 그래프가 교차 없이 배열되었다. 동일한 크기의 코퍼스 간 F1 score를 비교했을 때에도 표 5의

코퍼스 별 유사도 순서대로 나타났다. 코퍼스 크기가 늘어날수록 F1 score의 평균이 증가하고 표준편차가 감소했다. 코퍼스별로 상한 한계가 존재하나, 대체로 단조 증가하는 모습을 보였다. 따라서 코퍼스별 최고 F1 score는 각 코퍼스를 전부 사용한 지점에서 나타났다. 끝으로 분석 환경에서 전체 분석을 1회 시행하는 데 약 20시간이 소요되었다.

<표 6> Word2Vec 모델의 코퍼스 크기-문맥 별 F1 score (평균)

	Naver_MvRv_Model	*WebUser_MvRv*	*Movie_News*	*KorQuAD*
0.01N	0.769	0.739	0.721	0.698
0.04N	0.788	0.755	0.742	0.725
0.07N	0.796	0.761	0.747	0.732
0.1N	0.798	0.765	0.751	0.734
0.2N	0.807	0.772	0.757	0.738
0.3N	0.811	0.777	0.760	0.741
0.4N	0.814	0.780	0.762	0.743
0.5N	0.816	0.781	0.764	0.743
0.6N	0.817	0.784	0.764	0.745
0.7N	0.819	0.786	0.765	0.746
0.8N	0.820	0.787	0.766	0.745
0.9N	0.821	0.788	0.768	0.748
1N	**0.822**	0.789	0.769	0.749
2N		0.797	0.775	0.756
3N		0.800	0.780	0.759
4N		0.803	0.782	0.763
5N		0.805	0.783	0.763
6N		0.805	0.785	0.765
7N		0.807	0.785	0.766
8N		0.809	0.786	0.768
9N		0.809	0.787	0.768
10N		0.810	0.788	0.769
15.92N (TOTAL)		**0.813**		
17.06N (TOTAL)			**0.790**	
53.93N (TOTAL)				**0.777**

5.2.2 GloVe



[그림 11] GloVe 모델의 학습 코퍼스 크기와 문맥에 따른 감성 분석 결과
(실선은 평균 F1 score, 주변 영역은 표준편차)

GloVe의 코퍼스 문맥-크기 별 감성 분석 결과는 그림 11과 같다. 문맥별 감성 분석 정확도는 코퍼스별로 표 5의 유사도 순서대로 교차 없이 나타났다. 동일한 크기의 코퍼스 간 F1 score 비교에서도 이와 같은 결과가 나타났다. 그러나 코퍼스 크기가 일정 크기 이상이 되면 오히려 F1 score의 평균이 감소하고 표준편차가 커지는 모습을 보였다. 이는 특히 *WebUser_ MvRv* 코퍼스에서 두드러지게 나타났다. 분석 환경에서 전체 분석을 1회 시행하는 데에는 약 16시간이 소요되었다.

<표 7> GloVe 모델의 코퍼스 크기-문맥 별 F1 score (평균)

	Naver_MvRv_Model	*WebUser_MvRv*	*Movie_News*	*KorQuAD*
0.01N	0.749	0.713	0.691	0.674
0.04N	0.768	0.747	0.732	0.709
0.07N	0.775	0.750	0.738	0.721
0.1N	0.779	0.753	0.741	0.726
0.2N	0.787	0.759	0.744	0.732
0.3N	0.790	0.763	0.748	0.735
0.4N	0.794	0.766	0.749	0.737
0.5N	0.795	0.768	0.751	0.738
0.6N	0.797	0.770	0.751	0.739
0.7N	0.799	0.771	0.753	0.741
0.8N	0.801	0.773	0.753	0.742
0.9N	0.802	0.774	0.754	0.742
1N	**0.803**	0.775	0.755	0.743
2N		0.781	0.759	0.748
3N		0.785	0.760	0.749
4N		0.788	**0.761**	**0.752**
5N		0.789	0.759	0.749
6N		**0.791**	0.759	0.746
7N		0.788	0.758	0.744
8N		0.785	0.757	0.741
9N		0.784	0.758	0.742
10N		0.774	0.758	0.739
15.92N (TOTAL)		0.773		
17.06N (TOTAL)			0.756	
53.93N (TOTAL)				0.743

5.2.3 fastText

[그림 12] fastText 모델의 학습 코퍼스 크기와 문맥에 따른 감성 분석 결과
(실선은 평균 F1 score, 주변 영역은 표준편차)

fastText 모델의 코퍼스 문맥-크기 별 감성 분석 결과는 그림 12와 같다. 코퍼스 별 F1 score의 평균은 위 두 임베딩과 마찬가지로 학습 코퍼스 간 교차 없이 표 5의 유사도 순서대로 나타났다. F1 score는 학습 코퍼스의 크기가 증가할수록 단조 증가하여 학습 코퍼스가 가장 큰 모델에서 최고 F1 score가 나타났다. 분석 환경에서 전체 분석을 1회 시행하는데 소요된 시간은 약 39시간이다.

<표 8> fastText 모델의 코퍼스 크기-문맥 별 F1 score (평균)

	Naver_MvRv_Model	*WebUser_MvRv*	*Movie_News*	*KorQuAD*
0.01N	0.764	0.734	0.719	0.693
0.04N	0.790	0.757	0.746	0.727
0.07N	0.799	0.765	0.752	0.734
0.1N	0.803	0.770	0.756	0.737
0.2N	0.811	0.779	0.762	0.743
0.3N	0.815	0.782	0.766	0.747
0.4N	0.818	0.786	0.768	0.749
0.5N	0.820	0.788	0.769	0.750
0.6N	0.821	0.790	0.771	0.752
0.7N	0.822	0.792	0.772	0.752
0.8N	0.823	0.792	0.774	0.753
0.9N	0.825	0.793	0.775	0.752
1N	0.826	0.795	0.775	0.755
2N		0.801	0.780	0.764
3N		0.805	0.784	0.767
4N		0.807	0.787	0.767
5N		0.809	0.788	0.768
6N		0.810	0.789	0.770
7N		0.811	0.789	0.770
8N		0.812	0.791	0.771
9N		0.812	0.792	0.772
10N		0.814	0.793	0.773
15.92N (TOTAL)		0.816		
17.06N (TOTAL)			0.796	
53.93N (TOTAL)				0.783

제 6 장 논의 및 결론

6.1. 전체 결과 요약 및 해석

6.1.1 Word2Vec

Word2Vec은 세 모델 중에서 가장 안정적인 모습을 보였다. 학습 코퍼스의 크기가 커질수록 각 토큰의 정확한 의미가 임베딩 되어 과제 정확도가 상승하고 모델 간 F1 score의 편차가 작아질 것이라는 예상에 일치하는 결과를 보였다. 다른 임베딩 기법과 비교했을 때, 작은 코퍼스 크기에서 안정적으로 높은 정확도를 보였으며, 학습에 소요되는 시간 또한 상대적으로 적었다.

6.1.2 GloVe

GloVe는 세 임베딩 모델 중 가장 저조한 정확도를 보였다. 이는 한국어를 대상으로 여러 워드 임베딩 기법을 비교한 다른 연구들(조현수, 이상구, 2017; 이다빈, 최성필, 2019)에서 GloVe가 가장 낮은 정확도를 보인 것과 일치하는 결과이다.

특히, GloVe는 다른 임베딩 기법과는 다르게 코퍼스의 토큰 수가 일정 크기 이상이 되면 F1 score의 평균이 감소하고 표준편차가 증가하는 모습을 보였다. 이는 *WebUser_MvRv* 코퍼스에서 두드러지게 나타났고, *KorQuAD* 코퍼스에서도 나타났다. 이는 GloVe의 손실 함수가 가지는 특성 때문으로 해석된다. GloVe의 손실 함수는 수식 1에서 알 수 있듯, 학습하는 두 단어의 내적이 동시 등장 확률이 되도록 설정되어있다. 이때 동시 등장 확률은 $\frac{\text{두단어의동시등장횟수}}{\text{전체단어수}}$ 이다. 코퍼스의 크기가 커지면 두 단

어의 동시 등장 횟수와 전체 단어 수가 동시에 증가한다. 이때, 코퍼스의 크기가 일정 수준을 넘어서면 동시 등장 횟수보다는 전체 단어 수가 훨씬 더 빠르게 증가하고, 이는 학습하는 두 단어의 동시 등장 확률을 낮추게 된다. 따라서, 코퍼스가 커질수록 두 단어는 덜 유사한 벡터로 학습될 가능성이 커진다. 게다가 전체 코퍼스에서 저빈도로 나타나는 단어는 일종의 노이즈로 작용해, 고빈도로 나타나는 단어가 정확하게 벡터를 학습하는 것을 방해한다(Pennington et al., 2014).

이를 종합해보면, 감성 분석에서 판단에 중요하게 작용하는 단어들(예: 재미있다, 최고, 별로, 최악 등)의 벡터를 학습하는 과정에 있어, 학습 코퍼스 크기가 일정 수준 이상에 이르게 되면 이 단어 간의 동시 등장 확률이 낮아지기 때문에 F1 score의 평균이 감소하게 된다. 또, 서브 코퍼스를 만드는 과정에서 전체 코퍼스의 각 문서가 랜덤 샘플링되는데, 이때 위 주요 단어들의 학습을 방해하는 저빈도 단어들이 얼마나 포함되느냐에 따라 F1 score가 영향을 받게 된다. 이는 서브 코퍼스의 크기가 커질수록 F1 score의 표준편차가 증가하다가 모든 코퍼스를 다 쓰는 TOTAL 조건에서는 다시 감소하는 것을 설명한다.

한편, 총 학습 시간은 약 16시간으로, Word2Vec 모델의 약 80% 정도이다. 따라서 GloVe는 세 임베딩 모델 기법 중 계산 효율성이 가장 높다고 평가할 수 있다.

6.1.3 fastText

선행 연구(조현수, 이상구, 2017)와 마찬가지로 토큰화를 적용한 경우가 그렇지 않은 경우보다 성능이 높게 나왔다. 토크나이저를 사용하지 않고 각 토큰을 띄어쓰기를 기준으로 분리한 뒤 자모 분리만 시행한 경우, F1 score가 Word2Vec 모델보다 낮아 GloVe 모델과 유사하게 나타났으나, 토크나이저를 사용하여 토큰화를 한 뒤 자모 분리를 한 fastText 모델은 Word2Vec 모델보다 좋은 성능을 보였다. 특히, 토큰화를 하지 않은 경우 최고 F1 score는 GloVe와 유사했으나, 학습 코퍼스 크기가 작은 구간에서

는 GloVe보다 낮은 F1 score를 보였다. 이는 한국어에서 자모를 분리한 character n-gram 모델을 사용한다 하더라도, 어간의 의미를 효과적으로 공유하지 못함을 의미한다.

한편, 토크나이저를 사용했을 때에는 Word2Vec과 근사하지만 세 가지 기법 중 가장 높은 F1 score를 보였는데. 이는 fastText가 다른 임베딩 기법에 비해 높은 성능을 보인 선행 연구들의 결과를 지지한다(조현수, 이상구, 2017; García-Silva, & Gómez-Pérez, 2018; 이다빈, 최성필, 2019).

끝으로, fastText는 세 임베딩 기법 중 가장 긴 학습 시간이 소요되었다. 총 소요 시간은 약 39시간으로 모델을 학습하는데 다른 임베딩 기법보다 2배가량의 시간이 걸렸다. 이는 다른 임베딩 기법과는 달리, fastText가 각 토큰을 subword 단위로 연산해야 하기 때문이다. 각 토큰에 대해 학습을 할 때는 subword로 분할하여야 하며, 만들어진 모델로 입력 토큰에 대한 벡터값을 찾을 때도 해당 토큰을 subword로 분해하여 각 subword들의 벡터를 찾아 합해야 한다. 따라서 매 과정마다 subword를 생성해야 하기 때문에 다른 임베딩 기법에 비해 많은 학습 시간이 걸린 것으로 보인다.

6.2. 연구 의의 및 후속 과제 제안

본 연구는 자연어 처리에서 임베딩 모델에 사용되는 코퍼스의 크기와 문맥이 과제 정확도에 미치는 영향을 확인하였다. 학습 코퍼스와 과제 코퍼스의 문맥 유사도는 과제 정확도에 가장 큰 영향을 미친 요인이었다. 모든 워드 임베딩 모델에서 과제 코퍼스와 유사도가 낮은 학습 코퍼스를 사용한 모델은 학습 코퍼스의 크기가 기준값의 15배, 17배, 53배 이상이 되어도 기준 정확도(*Naver_MvRv_MODEL* 코퍼스의 1N에서의 정확도)를 넘기지 못하였다. 즉, 문맥 유사도에 따라 학습 코퍼스 크기 증가에 따른 과제 정확도 상승의 상한 한계가 나타났다.

워드 임베딩 기법에서 학습 코퍼스 간 차이를 비교한 기존 연구들은 단순히 영역 내/영역 외 코퍼스를 나누어 비교하거나, 코퍼스의 크기를 크고

작음으로 나누어 비교했다(Kutuzov & Kunilovskaya, 2015; Lai, et. al., 2016; Kutuzov & Kunilovskaya, 2017; Dusserre & Padró, 2017). 본 연구에서는 코퍼스의 크기를 체계적으로 나눔으로써 학습 코퍼스의 크기가 커질수록 과제 정확도가 점진적으로 상승한다는 결론을 얻었다. 나아가 코퍼스 크기 증가에 따른 정확도 상승에 상한선이 존재하며, 그 상한선은 과제 코퍼스와의 맥락 유사도에 따라 달라진다는 부분까지 확인했다는 것에 본 연구의 의의가 있다.

Word2Vec, GloVe, fastText 각 워드 임베딩 기법의 특징과 그에 따른 코퍼스의 크기-문맥과의 상호작용도 확인했다. Word2Vec은 작은 학습 코퍼스에서도 안정적으로 높은 성능을 보였다. GloVe는 학습에 소요된 시간은 가장 적었으나, 전체적인 과제 정확도는 세 임베딩 기법 중 가장 낮았다. 특히, 손실 함수의 특성 때문에 학습 코퍼스의 단어 구성에 따라 과제 정확도의 편차가 크게 나타났으며, 코퍼스의 크기가 일정 수준 이상으로 커지면 과제 정확도가 감소했다. fastText는 토크나이저 사용 유무에 정확도가 크게 영향을 받았다. 학습 시간 또한 다른 기법들에 비해 훨씬 많은 시간이 필요했다. 따라서 fastText는 다른 임베딩 기법보다 성능은 좋으나 계산 비용 또한 높다고 평가할 수 있다. 이와 같은 분석들은 선행 연구에서 다루지 않았던 부분이다. 따라서, 본 연구에서 수행한 코퍼스의 크기와 문맥에 따른 워드 임베딩 기법 간 양상 비교와 특성 확인 또한 본 연구의 의의이다.

Word2Vec, GloVe, fastText의 코퍼스 크기-문맥 별 과제 정확도 양상이 각각 달랐던 것처럼, 다른 자연어 처리 방법론에서도 서로 다른 결과가 나타날 수 있다. 딥러닝을 활용한 기법에는 자연어 처리에 주로 사용되는 Recurrent Neural Network(RNN, Rumelhart, Hinton, & Williams, 1986)와 Long-Short Term Memory 모델(Hochreiter, & Schmidhuber, 1997), 그리고 Convolutional Neural Network(CNN, LeCun, Bottou, Bengio, & Haffner, 1998)을 자연어 처리에 적용한 TextCNN(Kim, 2014)을 예로 들 수 있다. 이 모델들은 본 연구에서 사용한 기법들에 비해 더 깊고 넓은 임

베딩 레이어를 가지고 있어, 보다 나은 성능을 보이는 것으로 알려져 있다. 또한, 최근 자연어 처리 연구에서 각광받고 있는 전이학습 모델도 있다. 전이학습 모델은 BERT(Devlin et.al., 2018), ALBERT(Lan, Chen, Goodman, Gimpel, Sharma, & Soricut 2019), ELECTRA(Clark, Luong, Le, & Manning, 2020) 등이 있으며, 거대하고 일반적인 코퍼스로 모델을 선행 학습한 뒤(pretraining), 각 과제에 맞추어 학습된 임베딩 벡터를 조정하는(fine-tunning) 방식으로 작동한다. 이때, fine-tunning을 하는 코퍼스에 따라 동일했던 기존 모델의 벡터가 서로 다르게 갱신된다. 그러므로 전이학습 모델 또한 입력 코퍼스에 따라 모델의 과제 정확도가 달라진다. 따라서, 딥러닝 기반 모델과 전이학습 모델 모두 본 논문의 분석 절차와 동일하게 코퍼스의 크기와 문맥에 따른 과제 정확도 평가를 분석할 수 있으며, 코퍼스의 크기보다 문맥이 과제 정확도 상승에 더 중요한 요인이 될 수 있다. 또한, 본 논문에서 확인했듯 분석 방법에 따라 서로 다른 양상의 과제 정확도가 나타날 수 있다. 이런 사항들을 고려하면 본 논문에서 분석한 임베딩 기법 외에 현재 사용되고 있는 다양한 자연어 처리 방법론에 대해서 위와 같은 비교 분석을 시행해야 한다고 제안하는 바이다.

참 고 문 헌

강형석, 양장훈 (2019). 한국어 단어 임베딩을 위한 Word2vec 모델의 최적화. **한국디지털콘텐츠학회 논문지, 20(4)**, 825-833.

강범모 (2014). 텍스트 맥락과 단어 의미: 잠재의미분석. **언어학, (68)**, 3-34.

김청택, 이태헌 (2002). 뇌와 인지 모형: 잠재의미 분석을 사용한 문서분류. **한국심리학회지: 인지 및 생물, 14(4)**, 309-319.

이다빈, 최성필 (2019). 대용량 텍스트 자원을 활용한 한국어 형태소 임베딩의 모델별 성능 비교 분석. **정보과학회논문지, 46(5)**, 413-418.

임승영, 김명지, 이주열 (2018). KorQuAD: 기계독해를 위한 한국어 질의응답 데이터셋. **한국정보과학회 학술발표논문집**, 539-541.

조현수, 이상구 (2017). FastText 를 적용한 한국어 단어 임베딩. **한국정보과학회 학술발표논문집**, 705-707.

최상혁, 설진석, 이상구 (2016). 한국어에 적합한 단어 임베딩 모델 및 파라미터 튜닝에 관한 연구. **제 28 회 한글 및 한국어 정보처리 학술대회논문집.**

Bengio, Y., Ducharme, R., Vincent, P., & Jauvin, C. (2003). A neural probabilistic language model. *Journal of Machine Learning Research, 3(Feb),* 1137-1155.

Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2017). Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics, 5,* 135-146.

Chinchor, N. (1992, June). MUC-4 evaluation metrics. In *Proceedings of the Fourth Message Understanding Conference (MUC-4)* (pp. 22-29).

Clark, K., Luong, M. T., Le, Q. V., & Manning, C. D. (2020). ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators. *arXiv preprint arXiv:2003.10555.*

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding *arXiv preprint arXiv:1810.04805.*

Dusserre, E., & Padró, M. (2017). Bigger does not mean better! We prefer specificity. In *Iwcs 2017-12th international conference on computational semantics-short papers.*

Finkelstein, L., Gabrilovich, E., Matias, Y., Rivlin, E., Solan, Z., Wolfman, G., & Ruppin, E. (2001, April). Placing search in context: The concept revisited. In *Proceedings of the 10th international conference on World Wide Web* (pp. 406-414).

Goldberg, Y. (2016). A primer on neural network models for natural language processing. *Journal of Artificial Intelligence Research, 57,* 345-420.

Heu, J. U. (2019). Analysis and Comparison of Query focused Korean Document Summarization using Word Embedding. *The Journal of The Institute of Internet, Broadcasting and Communication, 19(6)*, 161-167.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation, 9(8)*, 1735-1780.

Kim, Y. (2014). Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882.*

Kutuzov, A., & Kunilovskaya, M. (2017, July). Size vs. structure in training corpora for word embedding models: Araneum Russicum Maximum and Russian National Corpus. In *International Conference on Analysis of Images, Social Networks and Texts* (pp. 47-58). Springer, Cham.

Kutuzov, A., & Kuzmenko, E. (2015, April). Comparing neural lexical models of a classic national corpus and a web corpus: the case for Russian. In *International Conference on Intelligent Text Processing and Computational Linguistics* (pp. 47-58). Springer, Cham.

Lai, S., Liu, K., He, S., & Zhao, J. (2016). How to generate a good word embedding. *IEEE Intelligent Systems, 31(6),* 5-14.

Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., & Soricut, R. (2019). ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. *arXiv preprint arXiv:1909.11942.*

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE, 86(11),* 2278-2324.

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013a). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781.*

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems* (pp. 3111-3119).

Pennington, J., Socher, R., & Manning, C. D. (2014, October). GloVe: Global Vectors for Word Representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)* (pp. 1532-1543).

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *nature, 323(6088),* 533-536.

Sasaki, Y. (2007). The truth of the F-measure, School of Computer Science, University of Manchester MIB, 131 Princess Street, Manchester, M1 7DN. *Version: 26th October*.

Singhal, Y., Jain, A., Batra, S., Varshney, Y., &Rathi, M. (2018, December). Review of bagging and boosting classification performance on unbalanced binary classification. In *2018 IEEE 8th International Advance Computing Conference (IACC)* (pp. 338-343). IEEE.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared

Kaplan, Prafulla Dhariwal, ... Dario Amodei. (2020). Language Models are Few-Shot Learners. *arXiv preprint arXiv:2005.14165.*

Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. R., & Le, Q. V. (2019). XLNet: Generalized Autoregressive Pretraining for Language Understanding. In *Advances in neural information processing systems* (pp. 5754-5764).

Abstract

The Effect of Corpus Size and Context on Task Accuracy in Natural Language Processing

Lee Junho

Interdisciplinary Program in Cognitive Science

The Graduate School

Seoul National University

Recent studies for natural language processing are being dominated by complex models based on big data. However, bigger data does not always guarantee better performance. In this paper, we confirmed how the size and context of the corpus each affect the accuracy of the task. The task was sentiment analysis on movie reviews and we used the three most widely used word embedding techniques: Word2Vec, GloVe, fastText.

The result of the analysis showed that the higher the context similarity between the training corpus of the embedding model and the task corpus, the higher the evaluation performance will be in all corpus sizes. The corpus with the lesser context similarity yielded lower performance even if the training corpus size was 15, 17, and 53 times bigger than its reference size.

The three-word embedding techniques showed different accuracy patterns depending on the size and context of the corpus. In specific, Word2Vec showed fast and stable performance, while GloVe showed a decrease in evaluation performance and the deviation of accuracy between models has increased when the size of the training corpus became bigger. On the other hand, fastText showed the best performance when reflecting on the characteristics of Korean text, but the time required for the analysis was about twice that of other techniques.

In summary, the present study measured quantitatively the context similarity between training corpus and task corpus in natural language processing using the word embedding technique. In addition, we confirmed in detail how the task accuracy changes according to the context similarity between the training-task corpus and the size of the training corpus.

keywords : Corpus, Context, Word embedding, Word2Vec, GloVe, fastText

Student Number : 2018-20278