# 단어 의미의 상호작용을 모델링하는 방법

박진호(서울대학교)

## 1. 문제제기

### 1.1. isotopy

한 문장 안에서 공기하는 단어들은 의미상 서로 잘 어울리는 경향이 있다. '그는 밥을 먹었다'에서 목적어 명사 '밥'(음식물)과 동사 '먹-'은 잘 어울린다. '그는 물을 마셨다'에서 목적어 명사 '물'(액체)과 동사 '마시-'는 잘 어울린다.

이렇게 원래 잘 어울리는 단어들이 선택되어 결합하기도 하지만, 원래는 특별히 잘 어울린다고 보기 어려운 단어들도 한 문장에서 공기하면 서로 잘 어울리게끔 해석되는 경향이 있다. 약은 꼭 액체인 것은 아니지만(물약뿐 아니라 알약, 가루약 등도 있으므로), '이 약 빨리 마셔라'에서는 동사 '마시-'와 공기함으로써 액체로 해석된다.

하나의 단어가 복수의 부면(facet)을 지닌 경우, 즉 다면어(multi-faceted word)의 경우, 어느 부면이 부각될지는 문장 안에서 공기하는 단어들의 영향을 받아 결정되는 일이 흔히 있다. '책'은 텍스트로서의 부면과 구체물로서의 부면이 있는데, '책을 읽다', '책이 쉽다/어렵다', '책이 재미있다'라고 할 때는 텍스트로서의 부면이 부각되고, '책을 찢다', '책이 두껍다', '책이 무거워서 들고 있으니 팔이 아프다'에서는 구체물로서의 부면이 부각된다.

'책' 같은 명사뿐 아니라 동사에서도 공기어의 영향을 받아 여러 부면들 중 어느 것이 부각될지가 결정될 수 있다. 예컨대 '읽다'는 목적어가 무엇인가에 따라 의미에 차이가 있다. '책을 읽다'는 단순히 눈으로 글자를 보고 텍스트를 이해한다는 뜻이지만, '상대방의 마음/생각/작전을 읽다'는 여러 단서를 바탕으로 추론을 해서 눈에 쉽게 보이지 않는 추상적인 것을 간파한다는 뜻이다.

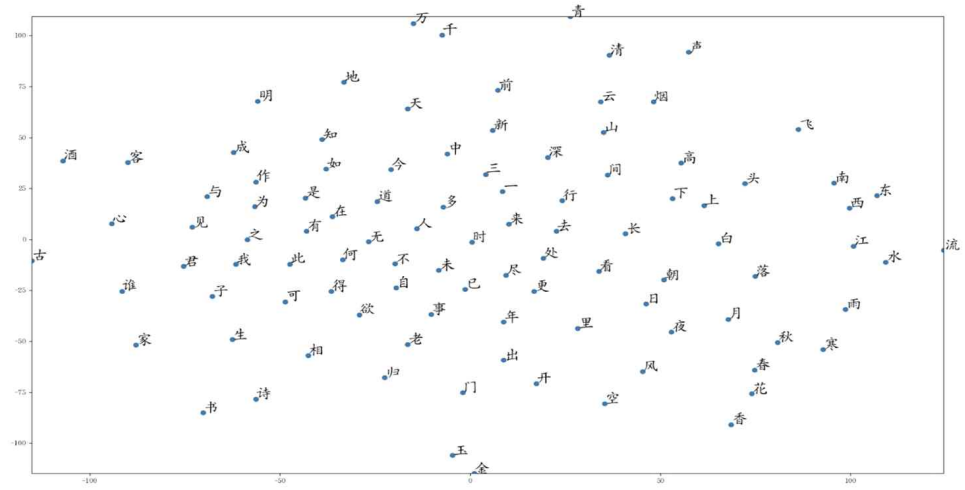
이렇게 한 문장 안에서 공기하는 단어들이 의미상 서로 어울리게끔 해석되는 현상은 Greimas에 의해 주목을 받은 바 있다. 그는 이 현상을 isotopy라고 불렀다.

### 1.2. 언어요소의 벡터화

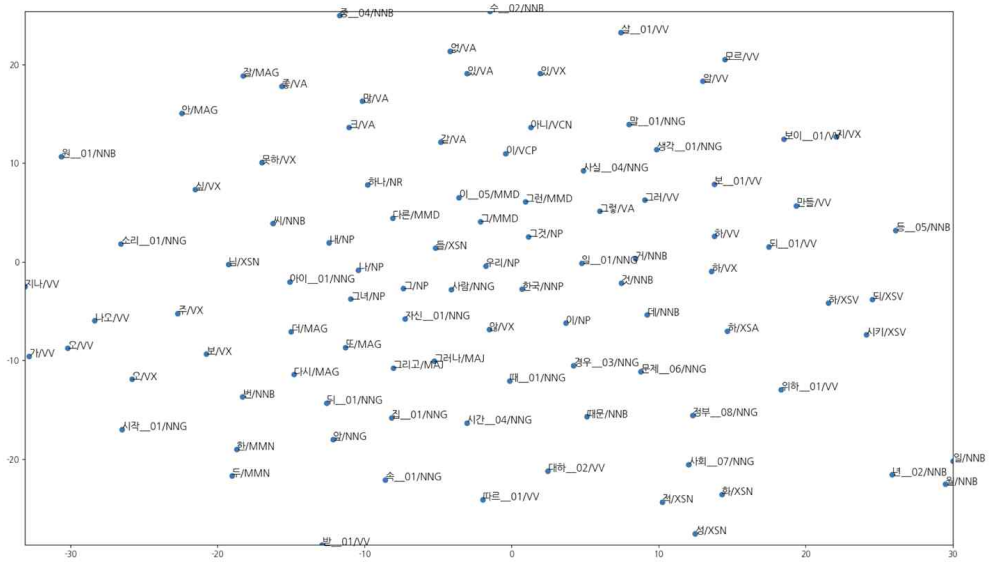
최근 자연어처리(natural language processing)에서는 단어나 문장 같은 언어요소를 벡터로 표상하는 것이 일반화되어 있다. 이러한 벡터 표상이 갖추어야 할 요건으로 ①유사성 조건, ②차이 조건, ③유추 조건을 들 수 있다.

첫째, 어떤 두 언어요소가 의미상 유사하다면 이 둘에 대응하는 벡터는 벡터 공간에서

가까이에 있어야 한다. 한문 텍스트와 한국어 텍스트에서 단어 벡터 만드는 기법인 Word2Vec으로 실험해 본 결과, 이 조건을 잘 만족시키고 있음을 알 수 있다(<그림 1, 2>).

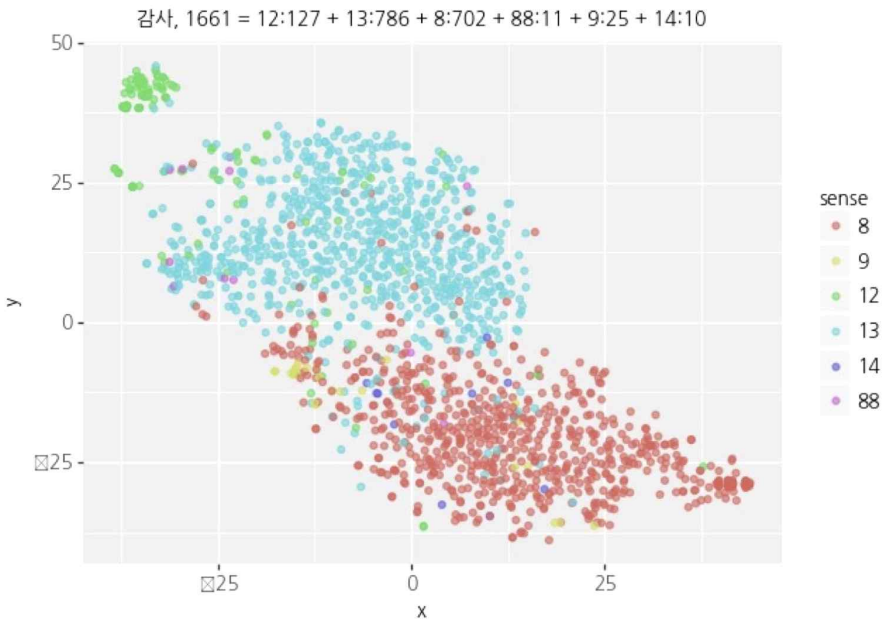


<그림 1> 한문 텍스트의 고빈도 한자들을 벡터화한 결과



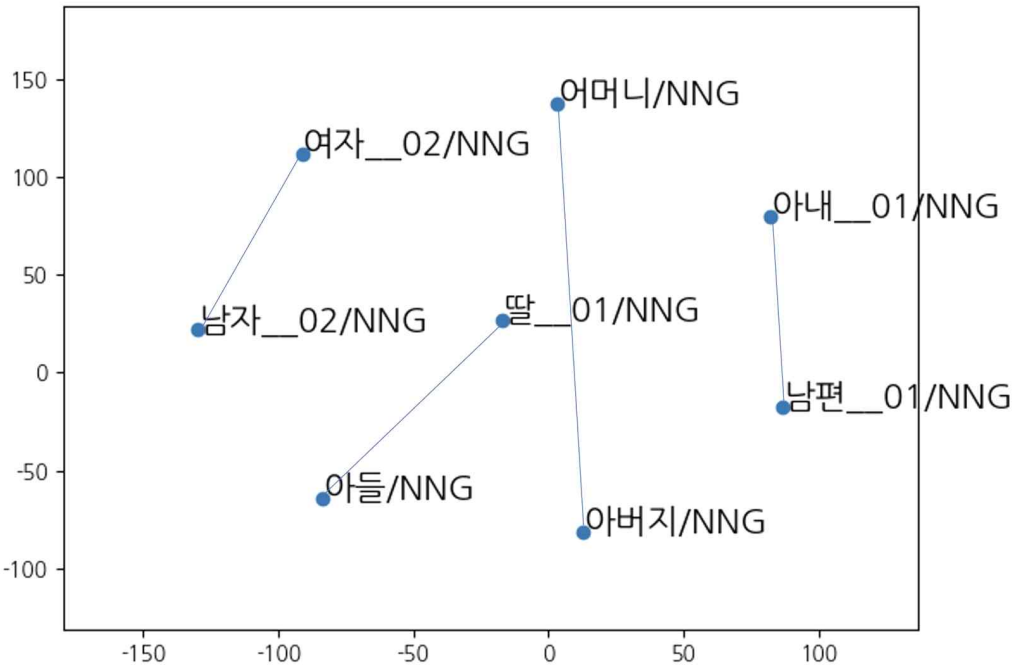
<그림 2> 세종 형태의미분석 말뭉치의 고빈도 형태소를 벡터화한 결과

둘째, 어떤 두 언어요소가 의미상 상당히 다르다면 이 둘에 대응하는 벡터는 벡터 공간에서 가급적 멀리 떨어져 있어야 한다. 이 조건은 동형어 같은 경우에 특히 중요하다. 형태상으로는 동일하지만 의미가 사뭇 다른 동형어의 경우, 이 두 의미가 벡터 공간상의 위치에 잘 반영되면 중의성 해소(disambiguation)에 유용할 것이다. Word2Vec과 문장 벡터 만드는 기법인 FSE(Fast Sentence Embedding)로 실험해 본 결과, 이 조건을 잘 만족시키고 있음을 알 수 있다(<그림 3>).




<그림 3> 동형어 '감사'의 예문들을 Word2Vec과 FSE로 벡터화한 결과

셋째, 네 언어 요소가 A:B=C:D 식으로 유추 관계에 있다면(예: 남자:여자=아버지:어머니) 이에 대응하는 벡터들을 두 개씩 연결한 선도 평행하거나 그에 가깝게 되어야 한다. '남자:여자', '아들:딸', '아버지:어머니', '남편:아내'라는 4개의 단어쌍으로 실험해 본 결과, 이 조건을 어느 정도 만족시키고 있음을 알 수 있다(<그림 4>).




<그림 4> 세종 형태의미분석 말뭉치의 4개 단어쌍을 벡터화한 결과

Word2Vec이 이와 같이 꽤 품질 좋은 벡터를 만들어내기는 하나, 특정 문장 내에서 공기한 다른 단어의 영향을 무시하고 각 단어를 고정된 벡터로 표상하기 때문에, isotopy를 제대로 포착할 수 없다. 그에 비해 최근 각광을 받고 있는 BERT라는 언어모델은 attention mechanism을 통해 isotopy를 포착할 수 있다. 이 모델에서는 각 단어 벡터(BERT식 용어로는 query)는 문장 내의 모든 단어 벡터(BERT식 용어로는 key)들의 가중평균으로 조정된다. 여기서 각 단어에 부여되는 가중치(BERT식 용어로는 value)는 신경망에 어떤 과제(가장 중요한 것은, 문장 중간중간에 masking된 단어들을 알아맞히는 과제)를 부여하여 이 과제를 잘 수행하게끔 신경망을 훈련시키는 과정에서 얻어진다. 이 가중평균 계산시 자기자신에 부여되는 가중치가 물론 가장 크지만(다른 단어들에 의해 원래 단어의 identity가 너무 많이 훼손되면 안 되므로), 문장 내의 다른 단어들에도 약간의 가중치가 부여되어 이 단어들의 영향을 받게 되는 것이다.

# 2. isotopy 실험

## 2.1. 한국어 KorBERT를 이용한 실험

다음의 12개 문장을 실험 대상으로 하였다.

(1) 재미있는 책을 읽었다.
(2) 두꺼운 책을 찢었다.
(3) 이 책은 너무 어렵다.
(4) 이 책은 너무 두껍다.
(5) 상대의 마음을 읽었다.
(6) 상대의 생각을 읽었다.
(7) 적군의 작전을 읽었다.
(8) 축구공을 차서 골문에 넣었다.
(9) 발로 문을 차서 열었다.
(10) 발로 벽을 차서 발이 아프다.
(11) 발로 앞좌석을 차서 앞사람이 화났다.
(12) 남자친구를 차서 마음이 안 좋다.

실험에 사용할 언어모델로는 한국전자통신연구원(ETRI)에서 만든 KorBERT를 선택하였다. KorBERT는 글자 단위 토큰화 모델과 형태소 단위 토큰화 모델을 제공하는데, 후자가 본 발표의 목적에 부합되므로 이를 선택하였다. 다른 기관에서 만든 한국어 BERT 모델들은 한국어와 한글에 대한 지식이 전혀 없는 language-agnostic 토크나이저를 사용했기 때문에 토큰화 결과가 형태소 단위와 일치하지 않는 경우가 많아서, 본 발표의 목적에 부합하지 않는다. KorBERT의 형태소 단위 토큰화 모델은 ETRI에서 개발한 형태소분석기를 토크나이저로 사용한다.

BERT의 신경망에 입력 문장을 집어넣으면, 토큰화하여 토큰(형태소) 단위로 분리하고,

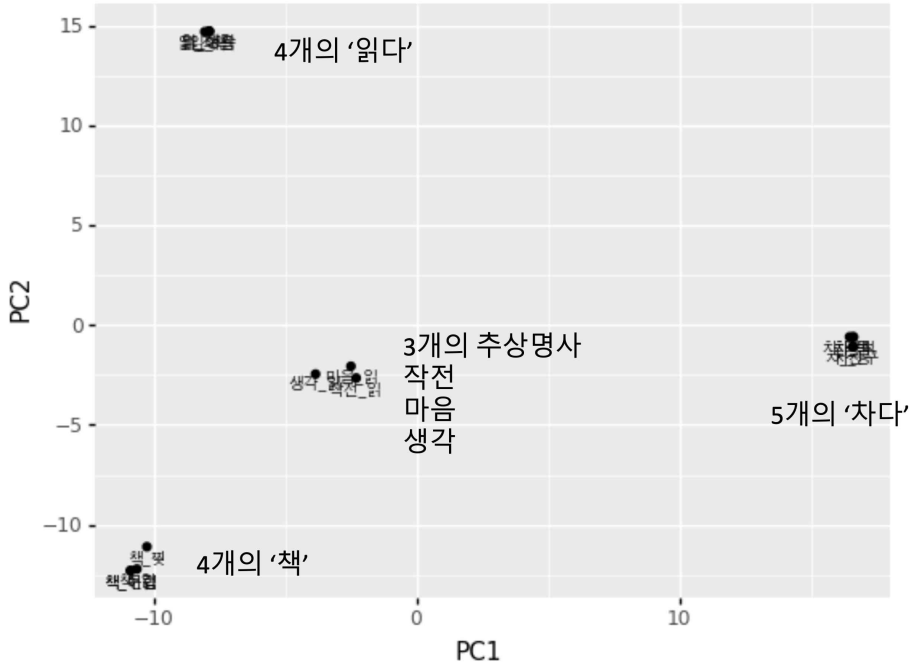
각 토큰을 일단 벡터화한다. 벡터의 차원은 사용자가 조정할 수 있는데, default는 768차원으로 설정되어 있다. 입력 층(input layer)에서 각 토큰의 벡터는 아직 문장 내 다른 토큰의 영향을 받기 전 상태이다. 그 뒤에 은닉 층(hidden layer)들을 거치면서, attention mechanism에 의해 다른 토큰의 영향을 받아 각 토큰의 벡터가 조정된다. 은닉층의 수도 조정할 수 있는데, default는 12개이다.

본 실험에서는 위의 12개 문장에 나타나는 16개 토큰의 벡터를 조사했다.

①읽_책
②책_읽
③책_찢
④책_어렵
⑤책_두껍
⑥읽_마음
⑦마음_읽
⑧읽_생각
⑨생각_읽
⑩읽_작전
⑪작전_읽
⑫차_공
⑬차_문
⑭차_벽
⑮차_좌석
⑯차_친구

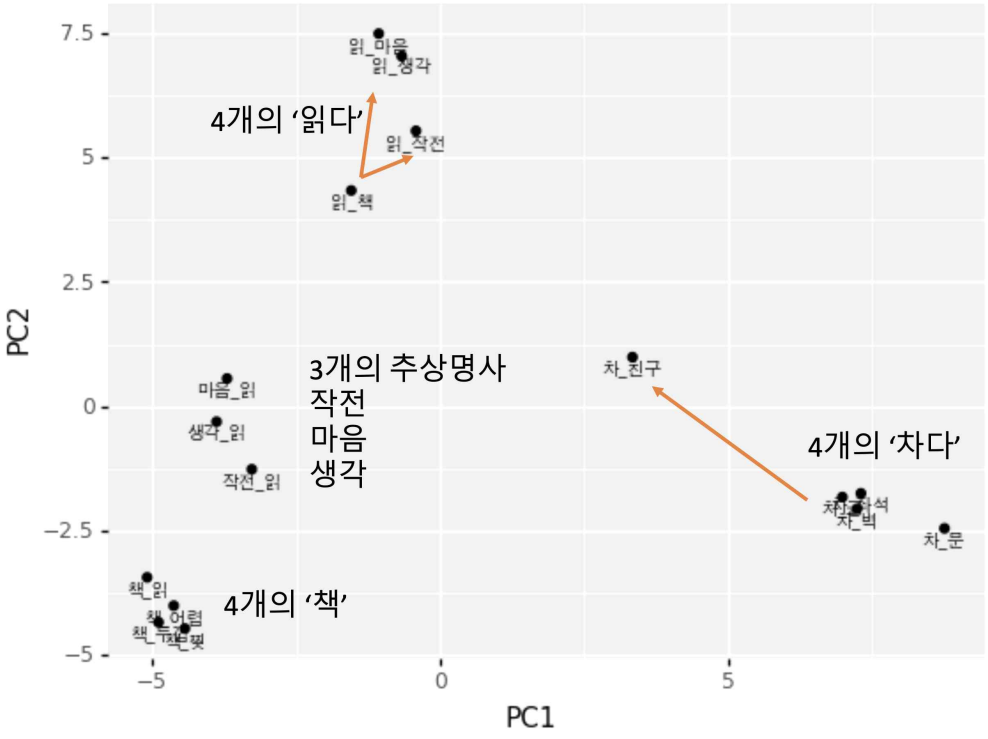
밑줄(underbar) 앞에 있는 것이 벡터화되는 타깃 토큰이고, 밑줄 뒤에 있는 것은 타깃 토큰의 벡터에 영향을 미치는 문장의 다른 토큰들 중 가장 핵심적인 요소이다. 즉 '읽_책'은 '책'을 목적어로 하는 문장에서의 '읽-'의 벡터를 의미하고, '책_읽'은 동사 '읽-'의 지배를 받는 명사 '책'의 벡터를 의미한다.

우선 문장 내 다른 토큰들의 영향을 받기 전인 입력 층에서의 이 16개 토큰의 벡터를 추출하였다. 각 벡터는 768차원이나, 시각화를 위해 주성분분석(PCA)을 통해 2차원으로 축소한 뒤, 2차원 평면에 plotting하였다(<그림 5>). 당연한 이야기지만, 4개의 '읽-', 5개의 '차-', 4개의 '책', 3개의 추상명사('생각', '마음', '작전')이 각각 매우 가까이에 몰려 있음을 볼 수 있다.




<그림 5> 16개 토큰의 입력 층에서의 벡터를 PCA로 2차원화한 결과

다음으로, 문장 내 다른 토큰들의 영향을 받은 후인 마지막 은닉 층에서의 16개 토큰의 벡터를 추출하여, 마찬가지로 PCA를 통해 2차원으로 축소하여 plotting하였다(<그림 6>).




<그림 6> 16개 토큰의 마지막 은닉 층에서의 벡터를 PCA로 2차원화한 결과

‘책’의 경우 공기한 다른 토큰의 영향이 상대적으로 작게 나타났으나, 3개의 추상명사는 공기하는 동사의 영향으로 서로서로 거리가 좀 멀어졌다. 동사 ‘읽-’의 경우, 목적어가 ‘책’일 때에 비해, 목적어가 ‘작전’일 때 어느 정도 거리가 멀어졌고, 목적어가 ‘생각’이나 ‘마음’일 때는 한결 더 멀어졌다. 동사 ‘차-’의 경우, 목적어가 ‘공’, ‘벽’, ‘좌석’인 경우는 여전히 가까이에 몰려 있으나, 목적어가 ‘문’인 경우 우측으로 약간 멀어졌고, 목적어가 ‘남자친구’인 경우 좌상 방향으로 매우 멀리 멀어졌다. 이를 통해, 특히 동사의 경우 공기하는 목적어의 영향을 받아 의미 해석이 달라지는 isotopy 현상을 BERT 모델이 어느 정도 포착하고 있음을 알 수 있다.

## 2.2. 중국어 MacBERT를 이용한 실험

다음의 12개 문장을 실험 대상으로 하였다. 대체로 2.1.의 한국어 실험에 사용한 것과 같은 의미의 문장을 사용했으나, ‘남자친구를 차다’의 경우 중국어 동사 ‘踢’는 그런 의미로 쓰지 않기 때문에 다른 숙어 ‘踢买卖’(“거래를 망치다”)를 사용하였다.

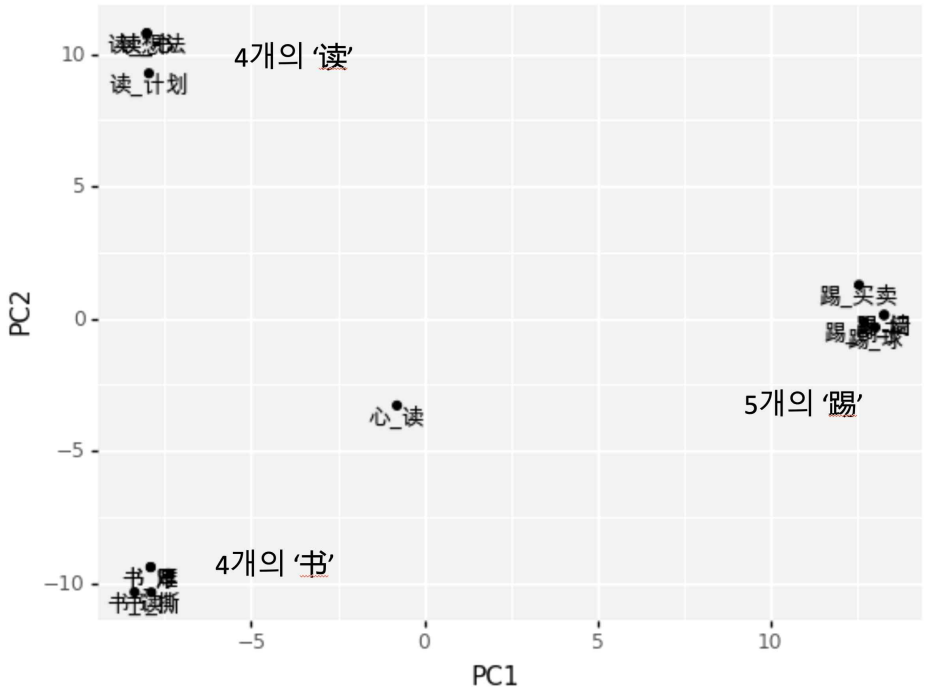
(1) 我读了一本有趣的书。
(2) 我撕了厚厚的书。
(3) 这本书太难了。
(4) 这本书太厚了。
(5) 我读懂了对方的心。
(6) 我读出了对方的想法。
(7) 我们读出了敌军的作战计划。
(8) 我把足球踢进了球门。
(9) 我用脚踢开了门。
(10) 我用脚踢了墙，脚疼。
(11) 用脚踢了椅子，前面的人生气了。
(12) 我踢了他们俩的买卖。

본 실험에서는 위의 12개 문장에 나타나는 다음 14개 토큰의 벡터를 조사했다. 명사 ‘想法’와 ‘计划’의 경우, MacBERT에서 사용하는 토크나이저가 한 덩어리로 분석하지 않고 각각 두 글자를 쪼개서 분석하였기 때문에 제외하였다. 제2절에서와 마찬가지로 ‘A_B’는 B와 공기한 A의 벡터를 의미한다.

①读_书
②书_读
③书_撕
④书_难
⑤书_厚
⑥读_心
⑦心_读

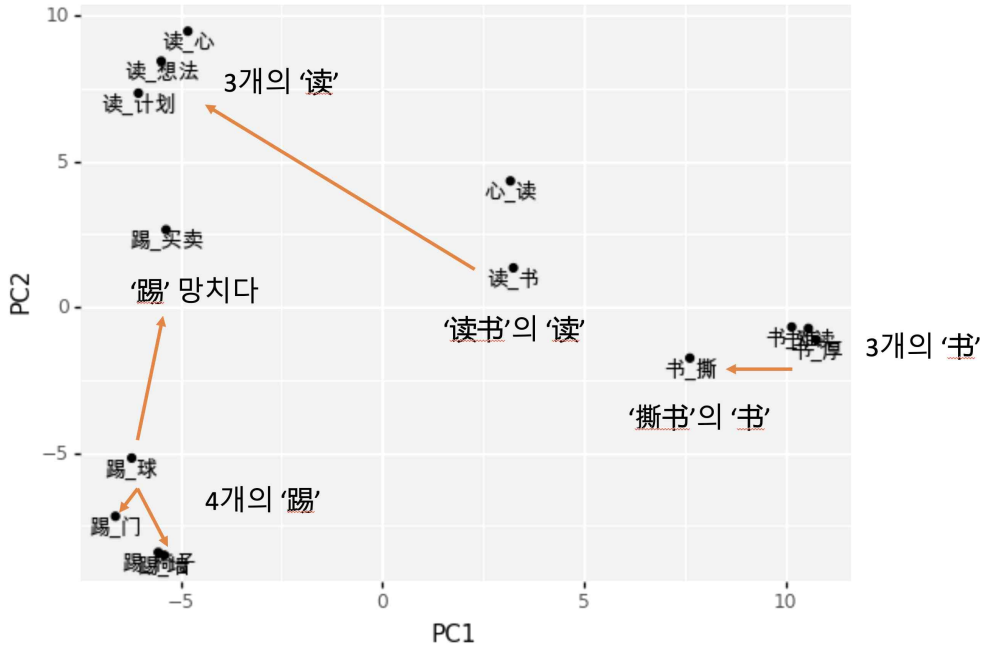
⑧读_想法
⑨读_计划
⑩踢_球
⑪踢_门
⑫踢_墙
⑬踢_椅子
⑭踢_买卖

우선 문장 내 다른 토큰들의 영향을 받기 전인 입력 층에서의 이 14개 토큰의 벡터를 추출하여, PCA를 통해 2차원으로 축소하여 plotting하였다(<그림 7>). 당연히, 4개의 '读', 5개의 '踢', 4개의 '书'가 한데 몰려 있다.



<그림 7> 14개 토큰의 입력 층에서의 벡터를 PCA로 2차원화한 결과

다음으로, 문장 내 다른 토큰들의 영향을 받은 후인 마지막 은닉 층에서의 14개 토큰의 벡터를 추출하여, PCA를 통해 2차원으로 축소하여 plotting하였다(<그림 8>).




<그림 8> 14개 토큰의 마지막 은닉 층에서의 벡터를 PCA로 2차원화한 결과

'书'를 목적어로 하는 '读'가 중앙에 있는 데 비해, 추상명사 '心', '想法', '计划'를 목적어로 하는 '读'는 좌상 구석에 매우 멀리 떨어져 있다. 보통의 독서 행위와 마음/생각/작전을 읽는 행위가 사뭇 다르다는 것을 잘 포착하고 있는 것이다.

'球'를 목적어로 하는 '踢'에 비해, '门'을 목적어로 하는 '踢'는 약간 아래로 내려갔고, '墙', '椅子'를 목적어로 하는 '踢'는 더 아래로 내려갔으며, 추상명사 '买卖'를 목적어로 하여 "망치다"를 의미하는 '踢'는 위로 멀리 이동하였다. 공을 차면 공이 꽤 멀리 이동하나, 문을 차면 문이 축을 중심으로 하는 회전 이동을 할 뿐 위치가 변하지는 않는다. 의자의 경우 가정의 보통 의자라면 차는 동작에 의해 약간 움직일 수 있겠으나 그리 많이 움직이지는 않고, 극장의 붙박이 의자라면 발로 차도 움직이지 않을 것이다. 그래서 목적어가 '벽'이나 '의자'일 경우는 대상물이 움직이는 경우와 사뭇 달라서 아래쪽으로 많이 이동한 것으로 보인다. 목적어가 '买卖'인 경우는 발로 차는 것과는 완전히 다른 의미이므로 가장 멀리 이동하였다.

4개의 '书' 가운데 동사 '读'와 공기할 때는 텍스트로서의 부면이 부각되고, 나머지 세 경우는 구체물로서의 부면이 부각되므로 전1자와 후3자가 대비되기를 기대했으나, 실제로는 동사 '撕'(찢다)와 공기한 '书'와 나머지 셋(读, 难, 厚)으로 나뉘었다. '읽다', '어렵다', '두껍다'는 '책'을 목적어로 한 꽤 일반적이고 정상적인 행위인 데 비해, '찢다'는 그에 비해 드물고 비정상적인 행위라는 점이 이런 결과를 낳은 것은 아닌가 생각되기도 한다.

전반적으로 한국어 KorBERT보다 중국어 MacBERT가 isotopy를 더 잘 포착하고 있는 것으로 보인다. 이는 BERT의 attention mechanism이 isotopy를 포착하기 위한 길을 열어 놓은 것은 맞지만, 그것만으로 isotopy를 완벽하게 포착하는 것은 아니며, 더 좋은 결과를 얻기 위해서는 다방면의 궁리가 필요함을 시사한다.

# 3. 동형어 중의성 해소 실험

## 3.1. Word2Vec + FSE를 이용한 실험 1: 통상적 방법

자연언어의 표현들 중에는 중의성(ambiguity)을 지닌 것들이 꽤 있다. 즉 하나의 형식이 둘 이상의 의미로 해석될 수 있는 것이다. '감사'라는 문자열은 '선생님께 감사 인사를 드렸다'라는 문장에서는 "感謝"로 해석되고, '이 회사는 회계 감사를 받고 있다'라는 문장에서는 "監査"로 해석된다. '가장'이라는 문자열은 '그는 이 집의 가장으로서 많은 책임을 지고 있다'라는 문장에서는 "家長"으로 해석되고, '아무런 가장 없이 솔직하게 자신을 표현해라'라는 문장에서는 "假裝"으로 해석된다. 이런 '감사'나 '가장' 같은 단어를 동형어(homonym)라고 한다. 인간은 문맥과 세상에 대한 지식을 바탕으로 동형어의 이런 중의성을 쉽게 해소할 수 있지만, 기계에게는 중의성 해소가 꽤 어려운 일이다.

인간이 중의성을 해소하는 기제를 본떠서 기계로 구현하려면 인간이 지닌 방대한 세상 지식을 적절히 표상해야 하고 문맥으로부터 필요한 정보를 추출하는 능력도 갖추어야 할 텐데, 이것은 너무나 어려운 일이다. Symbolic AI 또는 GOFAI(good old-fashioned AI), 규칙 기반 시스템, 전문가 시스템이 유행하던 수십 년 전에는 그런 방식이 시도되었으나 좋은 성과를 거두지 못하여 이제는 거의 사장되었다.

보다 효율적인 방법, 그리고 현재 유행하고 있는 방법은 기계학습을 이용하는 것이다. 기계학습은 어떤 과제를 수행하기 위해 필요한 정보(분류 과제의 경우 각 범주에 속하는 개체들을 구분하기 위해 알아야 할 패턴)를 인간이 기계에게 넣어 주는 것이 아니라 기계가 데이터로부터 스스로 찾아낸다. 다만 기계가 그 일을 할 수 있으려면 데이터가 수치로 표상되어야 한다.

Word2Vec 같은 기법으로 언어요소를 벡터화하여, 이 벡터를 동형어의 중의성 해소에 이용할 수 있다. 동형어가 2개의 의미(①과 ②)를 가지고 있다고 할 때, ①의 의미일 때 공기하는 단어와 ②의 의미일 때 공기하는 단어가 사뭇 다를 것이므로, 이 둘이 상당히 다르게 벡터화될 것으로 기대할 수 있는 것이다. 예컨대 '감사'가 "感謝"의 뜻일 때는 '드리-', '깊-', '-에게' 등과 흔히 공기할 것이고, "監査"의 뜻일 때는 '회계', '회사', '기관', '국정' 등과 흔히 공기할 것이다.

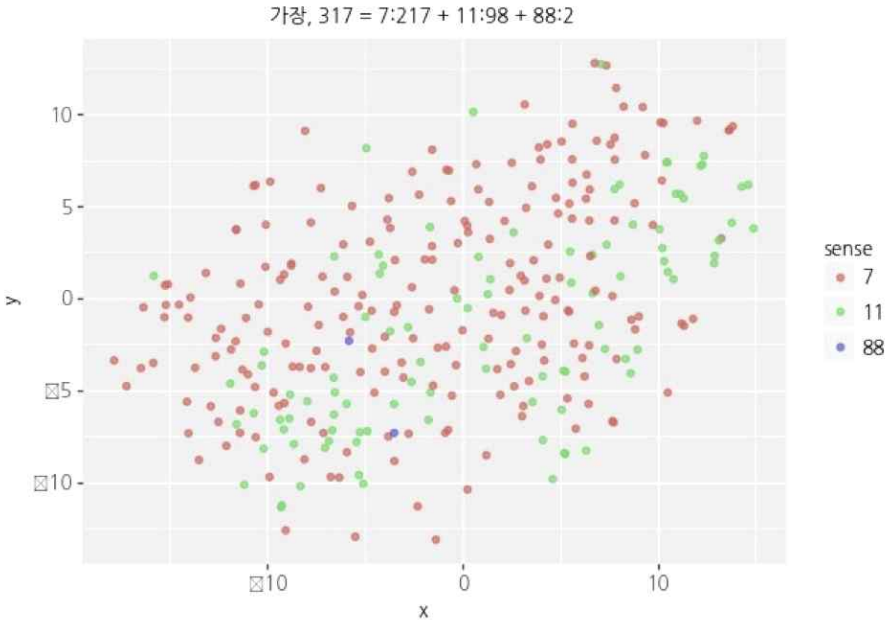
동형어의 구별에 벡터를 써먹으려고 한다면, 단어 벡터 그 자체뿐 아니라 문장 벡터를 이용하면 더 좋을 것이다. 단어 벡터를 만들 때, 말뭉치에서 그 단어 가까이에 자주 나타난 단어들의 영향이 이미 반영되어 있겠으나, 특정 문장에서 타깃 단어와 공기한 단어들의 벡터까지 종합적으로 고려하면 더 많은 정보를 얻을 수 있기 때문이다.

단어 벡터들을 종합하여 문장 벡터를 얻는 상식적인 방법은 단어 벡터들의 평균을 내는 것이다. 한 학급의 수학 성적을 하나의 수치로 나타내려면, 그 학급 학생들의 수학 성적의 평균을 내는 것과 같은 이치이다. 다만 문장을 이루는 모든 단어의 비중/중요도가 같은 것은 아니므로 적절히 가중치를 부여하여 가중평균(weighted average)을 내는 것이 좋다. 매우 많은 문장에 두루 나타나는 단어는 타깃 단어의 중의성 해소에 별 도움이 안 되므로 가중치를 낮추고, 소수의 문장에만 특징적으로 나타나는 단어는 가중치를 높여야

한다. FSE(Fast Sentence Embedding)는 바로 그런 아이디어를 구현한 문장 벡터화 알고리즘이다.

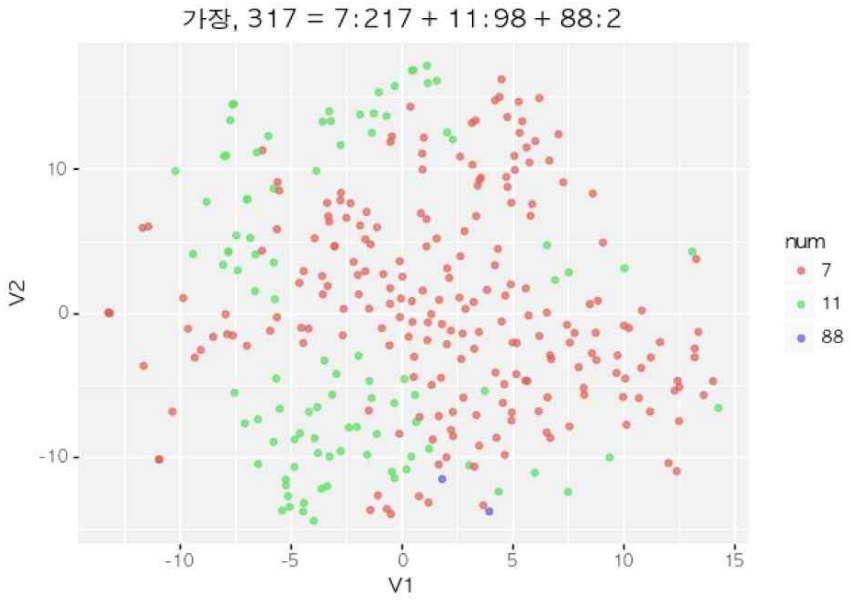
세종 형태의미분석 말뭉치에 나타나는 명사 '감사'의 1660여개의 예문들에 대해, Word2Vec으로 단어를 벡터화하고 FSE로 문장을 벡터화한 결과는 앞서 제시한 <그림 3>과 같다. 원래 하나의 단어, 하나의 문장을 100차원의 벡터로 나타냈으나, TSNE 알고리즘으로 2차원으로 축소하여 시각화한 것이다. '감사08(感謝, 적색)', '감사13(監査, 청색)', '감사12(監事, 연두색)'가 비교적 잘 구분되어 있다.

그런데 이 방법이 항상 좋은 결과만을 낳는 것은 아니다. '가장'의 경우에는 "家長"과 "假裝"의 경우들이 뒤섞여 있어서, 이것으로는 중의성 해소를 제대로 할 수 없다(<그림 9>).




<그림 9> '가장'의 벡터화: Word2Vec + FSE

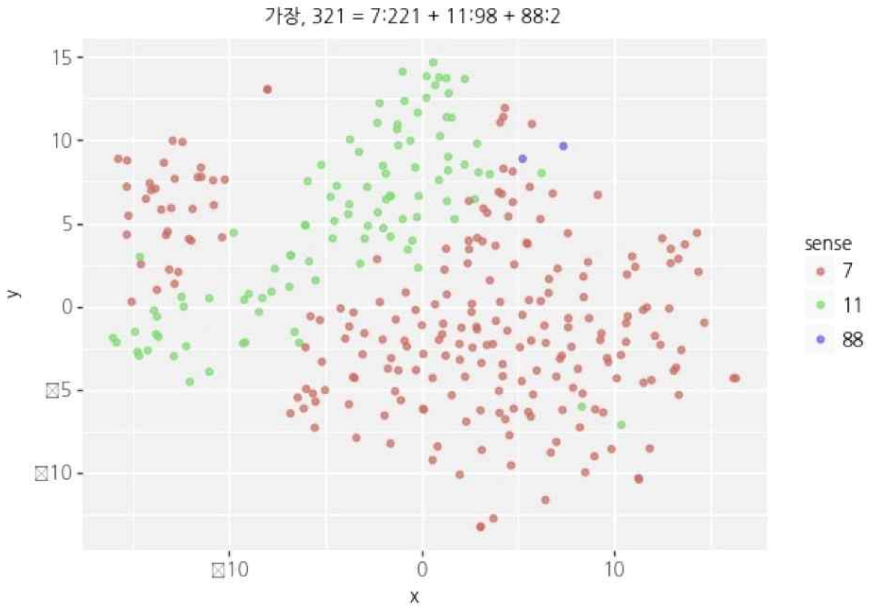
문장을 너무 길게 끊으면, 문장 안에 중의성 해소를 위한 단서뿐 아니라 노이즈도 많이 포함되게 되어 이것이 중의성 해소를 방해할 가능성도 있다. 그래서 문장을 약간 짧게 끊어서 벡터화를 해 보았다(<그림 10>). '가장'의 두 의미가 뒤섞이는 현상이 약간 나아지기는 했으나, 여전히 매우 불만족스럽다.



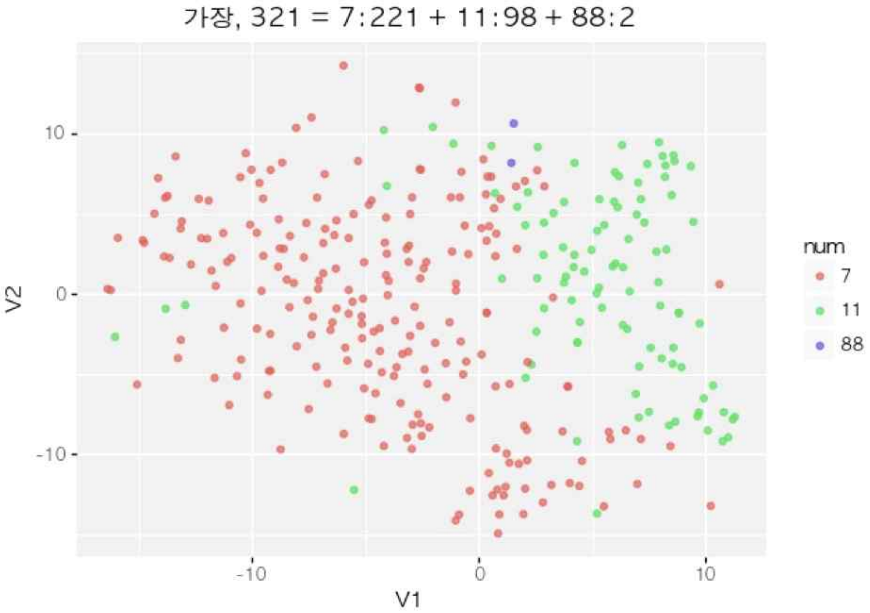

<그림 10> '가장'의 벡터화: Word2Vec + FSE (문장 짧게)

### 3.2. Word2Vec + FSE를 이용한 실험 2: 구문분석 활용

여기서 언어학의 도메인 지식이 도움이 될 수 있다. 문장을 구성하는 형태소나 단어들이 서로 맺는 관계의 긴밀도가 모두 동일한 것은 아니다. '맛있는 사과를 빨리 먹었다'와 '잘못했으면 상대방에게 사과를 빨리 하는 게 좋다'에서 동형어 '사과'의 중의성 해소에 '맛있-', '먹-', '상대방', '-에게' 등은 도움이 되지만 '빨리' 같은 것은 별로 도움이 되지 않는다. 대개 타깃 단어와 통사적으로 밀접한 관계에 있는 요소는 중의성 해소에 도움이 되지만, 그 외의 요소는 별 도움이 안 될 때가 많고 오히려 방해가 될 때도 있다. 따라서 문장의 모든 토큰을 포함해서 벡터화할 게 아니라, 중의성을 해소하고자 하는 타깃 단어와 통사적으로 긴밀한 관계에 있는 요소들만 뽑아서 벡터화하는 게 더 좋을 것이다. 이러한 아이디어를 바탕으로, '가장'의 예문들을 구문분석한 뒤 타깃 단어 '가장'과 통사적으로 긴밀한 관계에 있는 요소들만 뽑아서 Word2Vec+FSE로 벡터화한 결과는 <그림 11, 12>와 같다. '가장'의 두 의미가 뚜렷하게 구분됨을 알 수 있다. 요컨대 문장의 통사 구조에 대한 언어학적 고려가 동형어의 중의성 해소에 도움이 될 수 있는 것이다.

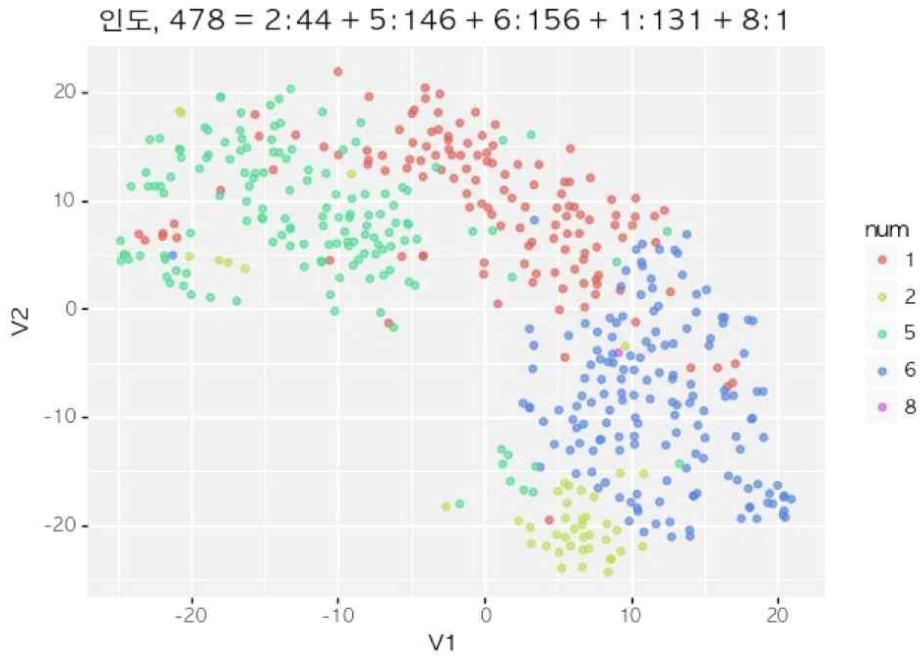



<그림 11> '가장'의 벡터화: Word2Vec + FSE + 구문분석 (문장 길게)

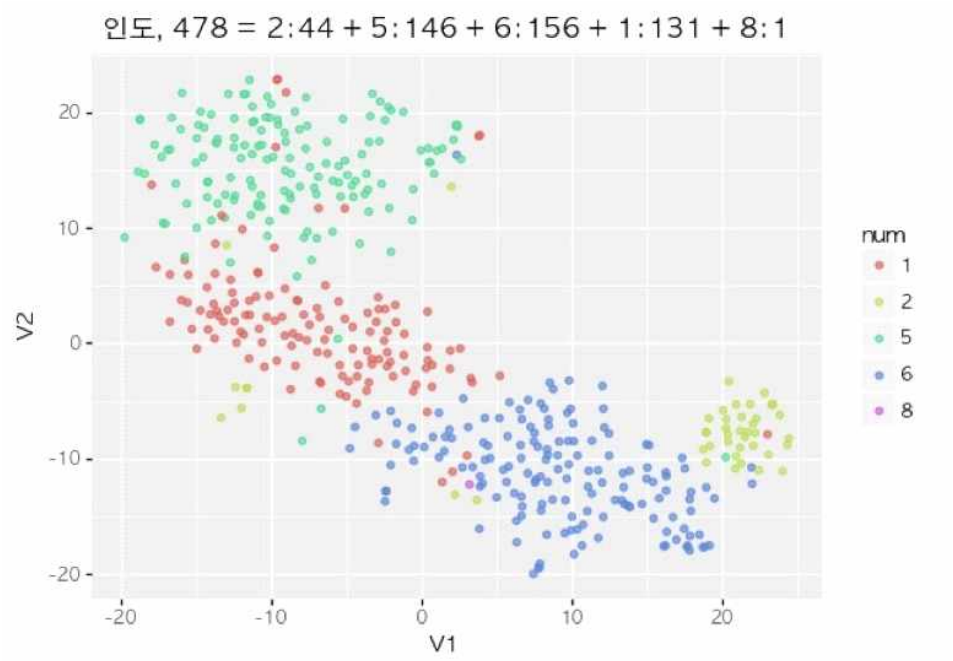



<그림 12> '가장'의 벡터화: Word2Vec + FSE + 구문분석 (문장 짧게)

'인도'의 경우 구문분석을 고려하지 않고 통상의 방법으로 벡터화했을 때도 용법들이 꽤 잘 구분되나(<그림 13>), 구문분석을 고려하여 벡터화하면 더 잘 구분된다(<그림 14>).

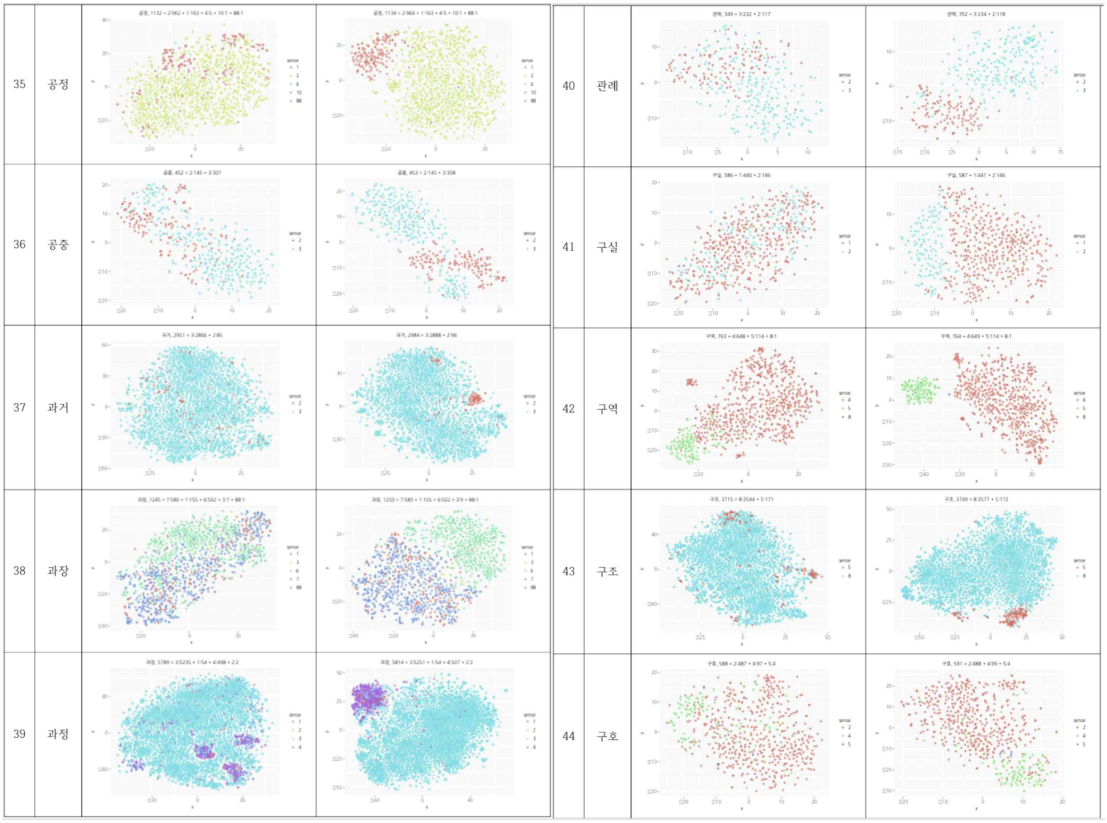



<그림 13> '인도'의 벡터화: Word2Vec + FSE




<그림 14> '인도'의 벡터화: Word2Vec + FSE + 구문분석

285개의 동형어 명사에 대해 Word2Vec+FSE의 통상적인 벡터화 방법과 구문분석을 활용한 벡터화 방법을 비교해 보면, 전자보다 후자가 일관되게 더 좋은 결과를 낳음을 알 수 있다. 10개만 제시하면 <그림 15>와 같다.

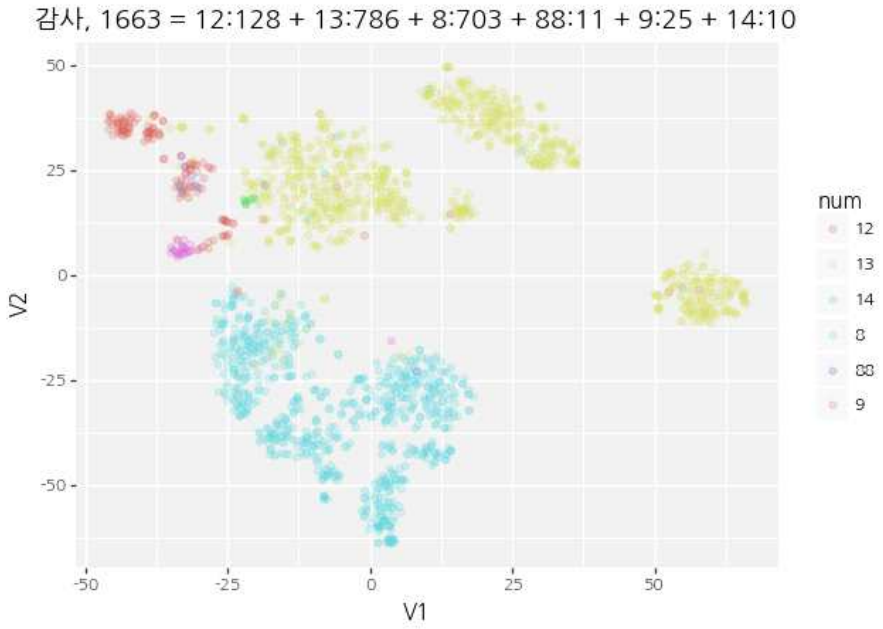



<그림 15> 통상적인 벡터화와 통사구조를 고려한 벡터화의 비교: 10개의 동형어 명사

### 3.3. KorBERT를 이용한 실험

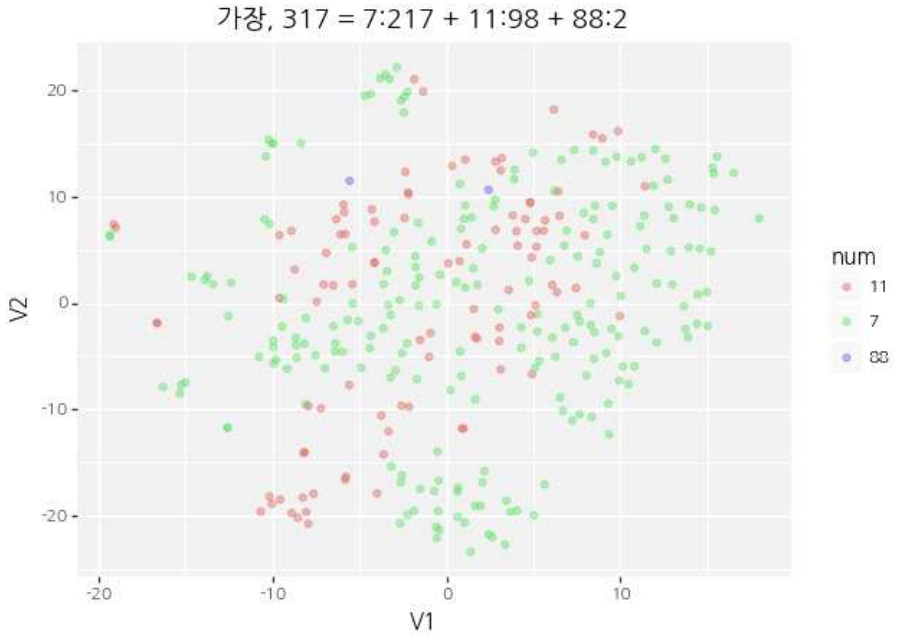
명사 '감사'의 용례 1660여개를 KorBERT에 넣어서 벡터를 구해 보았다. KorBERT에 문장을 넣기 위해서는 문장을 우선 ETRI의 Open API에서 제공하는 형태소분석기로 토큰화해야 한다. 그런데 이 형태소분석기는 완벽하지 않기 때문에 분석의 오류가 간혹 있다. '감사'에 대해 보통명사(NNG)로 태깅해야 하는데 이를 다른 품사로 오인하는 경우도 있고, '감사'를 두 개의 토큰으로 잘못 쪼개는 경우도 있다. 이렇게 '감사'를 보통명사로 제대로 태깅하지 않은 경우에는, 보통명사 '감사'라는 토큰이 존재하지 않으므로 이것의 벡터를 얻을 수 없다. 그런 경우에는 차선책으로 문장 전체의 벡터를 구했다. KorBERT의 단어 벡터나 문장 벡터는 768차원인데 TSNE를 통해 2차원으로 축소하여 plotting한 결과는 <그림 16>과 같다.

같은 색깔로 표시된 하나의 의미의 예문들이 둘 이상의 덩어리로 쪼개져 있는 모습을 볼 수 있다. 이는 태깅 오류에 의해 '감사/NNG'라는 토큰이 있는 경우에는 이 토큰의 벡터를 구하고, 없는 경우에는 문장 전체의 토큰을 구했기 때문에 생긴 현상으로 보인다. 이로 인해 각 색깔의 군집이 둘 이상으로 쪼개져 있기는 하나, 여러 색깔의 관측점들이 뒤섞이지는 않았기 때문에, 동형어의 중의성 해소에는 지장이 없다.

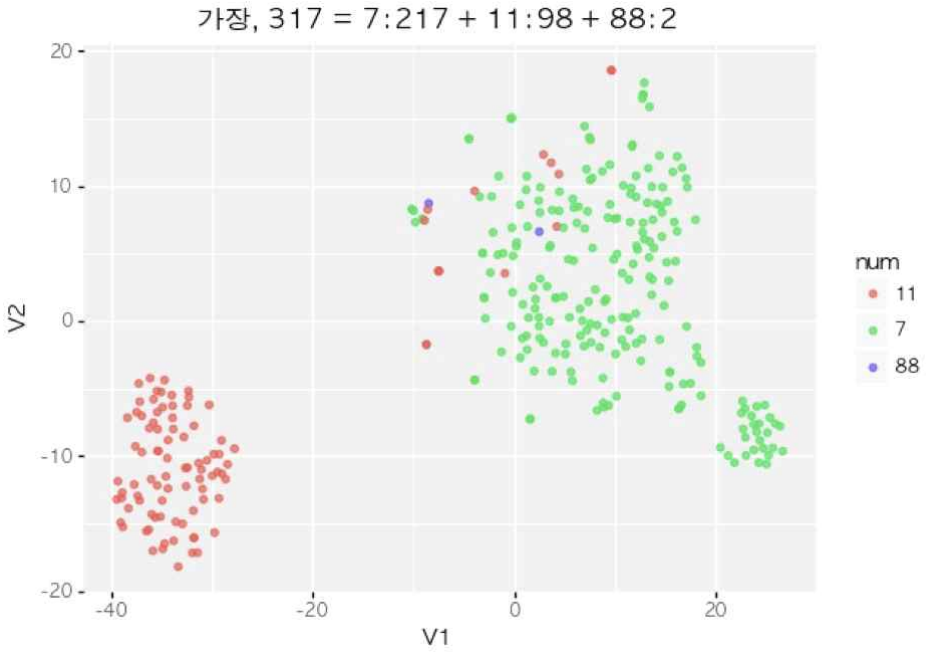



<그림 16> '감사'의 벡터화: KorBERT

'가장'의 경우 문장을 짧게 끊었을 때는 용법들이 명확히 구분되지 않으나(<그림 17>), 문장을 짧게 끊으면 용법들이 잘 구분된다(<그림 18>).

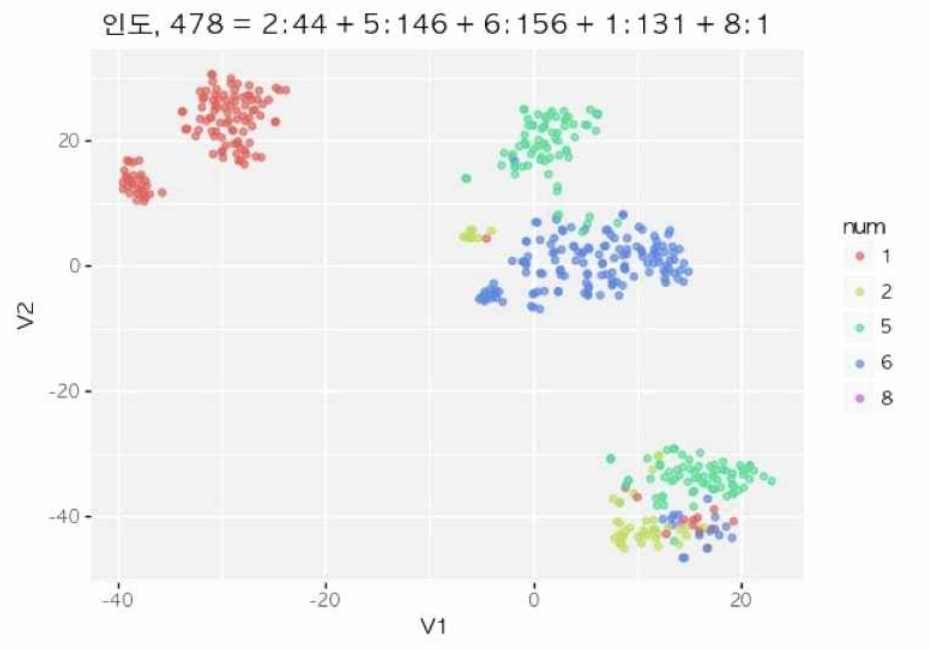



<그림 17> '가장'의 벡터화: KorBERT (문장 길게)




<그림 18> '가장'의 벡터화: KorBERT (문장 길게)

'인도'의 경우 각 용법들이 상당히 멀리 떨어져서 잘 구분된다(<그림 19>). 다만 같은 색깔의 군집이 둘 이상으로 쪼개지는 현상은 여전히 나타난다.




<그림 19> '인도'의 벡터화: KorBERT

KorBERT를 이용했을 때는 굳이 구문분석 정보를 활용하지 않아도 동형어 구분이 상당히 잘 됨을 알 수 있다. 아마도 BERT 모델 훈련 과정에서 문장의 통사 구조에 대한

정보를 모델이 어느 정도 섭취했을 가능성이 있다.

그렇기는 하지만 BERT 모델은 훈련시에 문장의 길이(토큰의 수)가 길어짐에 따라 길이의 제곱에 비례해서[quadratically] 훈련 시간이 길어진다. 이러한 부담을 줄이기 위해서는 문장 내의 모든 토큰들 사이의 관계를 다 고려하기보다는, 중요한 관계가 뽑아서 고려할 수 있는 방법이 필요하다. 이럴 때 구문분석 정보를 활용해서 타깃 토큰과 통사적으로 긴밀한 관계에 있는 토큰만 고려하면 될 것이다.

예컨대 '재미있-는 책-을 읽-었-다'에서 '재미있-는'과 '읽-었-다'는 직접적인 관련이 없다. '책-을'은 '읽'과 관련을 맺고, '었, 다'와는 직접적인 관계가 없다. '재미있-는'은 '책'과 관련을 맺고 '을'과는 직접적인 관계가 없다. 따라서 실제로 고려해야 할 관계는 '책, 을, 읽' 셋 사이의 관계, '읽, 었, 다' 셋 사이의 관계, '재미있, 는, 책' 셋 사이의 관계뿐이다. 통상의 attention mechanism에서 고려해야 할 관계의 수가 원래 $7^2=49$개였는데 이것이 $3^2+3^2-1+3^2-1=25$개로 줄어든다.

'어려운(어렵-은) 책-은 재미없-다'의 경우에도 '어렵-은'과 '재미없-다'는 직접적인 관련이 없다. '책-은'은 '재미없'과 관련될 뿐, '다'와는 관계가 없다. '어렵-은'은 '책'과 관련될 뿐, '은'과는 관계가 없다. 따라서 실제로 고려해야 할 관계는 '책, 은, 재미없' 셋 사이의 관계, '재미없, 다' 둘 사이의 관계, '어렵, 은, 책' 셋 사이의 관계뿐이다. 고려해야 할 관계의 수가 $6^2=36$개에서 $3^2+2^2-1+3^2-1=20$개로 줄어든다.

### 3.4. AutoML을 이용한 벡터의 분류

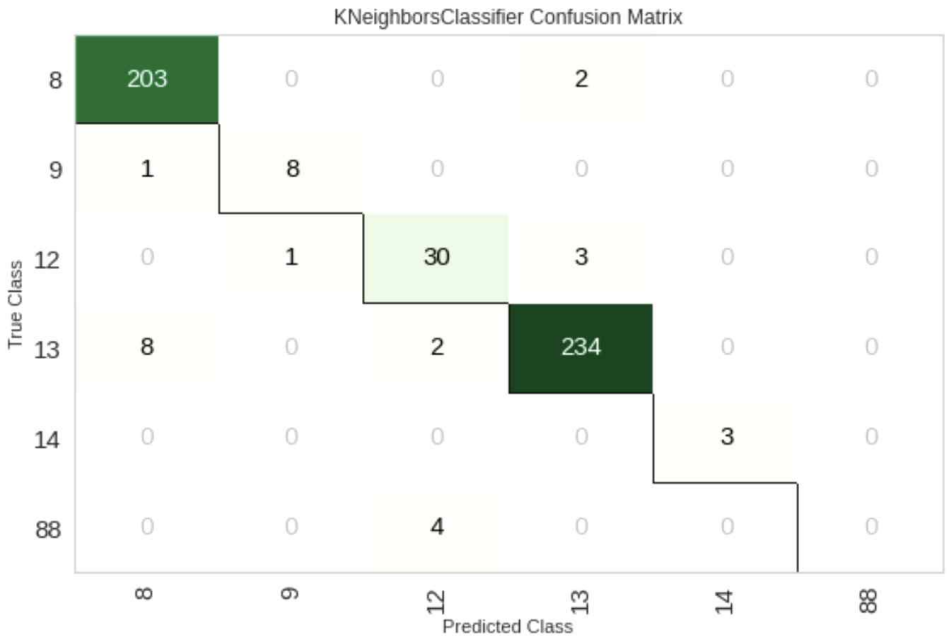
동형어의 벡터들이 잘 구분되어 있으면, 여기에 기계학습 알고리즘(이 경우 분류 알고리즘)을 적용하여 매우 높은 정확도로 중의성 해소를 할 수 있다. KorBERT는 각 토큰과 문장을 768차원의 벡터로 나타내는데, '감사'의 1,660여개 예문에 대해 PyCaret이라는 AutoML 라이브러리를 이용하여 다양한 기계학습 기법을 적용해 본 결과 KNN(K-Nearest-Neighbor) 기법이 가장 좋은 성능을 보였다(<그림 20>).

| | Model | Accuracy | AUC | Recall | Prec. | F1 | Kappa | MCC | TT (Sec) |
|---|---|---|---|---|---|---|---|---|---|
| knn | K Neighbors Classifier | 0.9682 | 0.3955 | 0.8486 | 0.9636 | 0.9656 | 0.9464 | 0.9466 | 0.1270 |
| lightgbm | Light Gradient Boosting Machine | 0.9656 | 0.3982 | 0.8087 | 0.9596 | 0.9620 | 0.9417 | 0.9420 | 2.5500 |
| lr | Logistic Regression | 0.9631 | 0.3983 | 0.8463 | 0.9585 | 0.9604 | 0.9375 | 0.9378 | 0.2540 |
| nb | Naive Bayes | 0.9605 | 0.3951 | 0.8660 | 0.9604 | 0.9590 | 0.9331 | 0.9337 | 0.0950 |
| svm | SVM - Linear Kernel | 0.9588 | 0.0000 | 0.8368 | 0.9569 | 0.9570 | 0.9301 | 0.9306 | 0.1020 |
| ridge | Ridge Classifier | 0.9570 | 0.0000 | 0.8288 | 0.9532 | 0.9546 | 0.9273 | 0.9277 | 0.0190 |
| et | Extra Trees Classifier | 0.9570 | 0.3983 | 0.7124 | 0.9482 | 0.9505 | 0.9260 | 0.9272 | 0.1060 |
| lda | Linear Discriminant Analysis | 0.9502 | 0.3918 | 0.8590 | 0.9472 | 0.9481 | 0.9158 | 0.9162 | 0.0440 |
| rf | Random Forest Classifier | 0.9485 | 0.3975 | 0.6809 | 0.9384 | 0.9415 | 0.9114 | 0.9126 | 0.1500 |
| gbc | Gradient Boosting Classifier | 0.9433 | 0.3957 | 0.6518 | 0.9448 | 0.9427 | 0.9039 | 0.9047 | 9.6980 |
| dt | Decision Tree Classifier | 0.9064 | 0.3673 | 0.6738 | 0.9133 | 0.9070 | 0.8430 | 0.8445 | 0.1450 |
| ada | Ada Boost Classifier | 0.8368 | 0.3345 | 0.4510 | 0.7723 | 0.7968 | 0.7063 | 0.7210 | 0.3450 |
| dummy | Dummy Classifier | 0.4656 | 0.2000 | 0.1867 | 0.2168 | 0.2959 | 0.0000 | 0.0000 | 0.0150 |
| qda | Quadratic Discriminant Analysis | 0.3943 | 0.2252 | 0.3180 | 0.2632 | 0.3122 | 0.1021 | 0.1253 | 0.0350 |

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=-1, n_neighbors=5, p=2,
                     weights='uniform')
```

<그림 20> '감사'가 포함된 문장들을 벡터화한 뒤 PyCaret으로 분류 기법들을 적용한 결과

혼동행렬(confusion matrix)을 보면 대부분의 경우 정확히 분류하고 있음(대각선 셀들)을 볼 수 있고, 소수의 경우 어떤 범주를 어떤 범주로 잘못 분류했는지도 알 수 있다(<그림 21>).




<그림 21> 동형어 '감사'의 중의성 해소에 KNN을 적용한 결과 (혼동행렬)

# 4. 말

문장 내에서 함께 공기하는 단어들은 서로 영향을 주고받아서 서로 어울리게끔 의미 해석이 이루어진다(isotopy). Word2Vec처럼 단어의 의미를 고정된 벡터로 표상하는 언어 모델은 isotopy를 포착할 수 없다. BERT는 attention mechanism에 의해 문장 내 모든 단어의 영향을 받게끔 단어 벡터를 조정하므로, isotopy를 포착할 수 있는 길을 열었다.

동형어의 중의성을 해소할 때, 단어와 문장을 벡터화하는 여러 기법들이 유용하게 쓰일 수 있는데, 구문분석 정보를 활용하면 더 좋은 결과를 얻을 수 있다. BERT 모델은 훈련 시에 이미 문장의 통사 구조에 대한 정보를 어느 정도 섭취한 것으로 추측되나, 문장이 길어질수록 훈련 시간이 너무 오래 걸린다는 단점이 있다.

문장 내에서 모든 단어들이 서로 대등한 정도로 상호작용하는 것은 아니다. 더 긴밀히 상호작용하는 단어 쌍도 있고, 상호작용이 미미하거나 전혀 없는 단어 쌍도 있다. 이러한 차이를 포착하려면 역시 구문분석을 활용해야 한다. 구문분석을 활용하면 (중의성 해소를 더 잘 할 수 있다는 장점뿐 아니라) BERT 모델의 훈련 시간을 단축한다는 경제적 측면에서의 이점도 있다.

박진호(2020), 문장 벡터를 이용한 동형어 구분, 《한국(조선)어교육연구》 16, 7-48.