단행본 서명의 단어 임베딩에 따른 자동분류의 성능 비교

Performance Comparison of Automatic Classification Using Word Embeddings of Book Titles

이용구 (Yong-Gu Lee)*

초 록

이 연구는 짧은 텍스트인 서명에 단어 임베딩이 미치는 영향을 분석하기 위해 Word2vec, GloVe, fastText 모형을 이용하여 단행본 서명을 임베딩 벡터로 생성하고, 이를 분류자질로 활용하여 자동분류에 적용하였다. 분류기는 k-최근접 이웃(kNN) 알고리즘을 사용하였고 자동분류의 범주는 도서관에서 도서에 부여한 DDC 300대 강목을 기준으로 하였다. 서명에 대한 단어 임베딩을 적용한 자동분류 실험 결과, Word2vec와 fastText의 Skip-gram 모형이 TF-IDF 자질보다 kNN 분류기의 자동분류 성능에서 더 우수한 결과를 보였다. 세 모형의 다양한 하이퍼파라미터 최적화 실험에서는 fastText의 Skip-gram 모형이 전반적으로 우수한 성능을 나타냈다. 특히, 이 모형의 하이퍼파라미터로는 계층적 소프트맥스와 더 큰 임베딩 차원을 사용할수록 성능이 향상되었다. 성능 측면에서 fastText는 n-gram 방식을 사용하여 하부문자열 또는 하위단어에 대한 임베딩을 생성할 수 있어 재현율을 높이는 것으로 나타났다. 반면에 Word2vec의 Skip-gram 모형은 주로 낮은 차원(크기 300)과 작은 네거티브 샘플링 크기(3이나 5)에서 우수한 성능을 보였다.

ABSTRACT

To analyze the impact of word embedding on book titles, this study utilized word embedding models (Word2vec, GloVe, fastText) to generate embedding vectors from book titles. These vectors were then used as classification features for automatic classification. The classifier utilized the k-nearest neighbors (kNN) algorithm, with the categories for automatic classification based on the DDC (Dewey Decimal Classification) main class 300 assigned by libraries to books. In the automatic classification experiment applying word embeddings to book titles, the Skip-gram architectures of Word2vec and fastText showed better results in the automatic classification performance of the kNN classifier compared to the TF-IDF features. In the optimization of various hyperparameters across the three models, the Skip-gram architecture of the fastText model demonstrated overall good performance. Specifically, better performance was observed when using hierarchical softmax and larger embedding dimensions as hyperparameters in this model. From a performance perspective, fastText can generate embeddings for substrings or subwords using the n-gram method, which has been shown to increase recall. The Skip-gram architecture of the Word2vec model generally showed good performance at low dimensions(size 300) and with small sizes of negative sampling (3 or 5).

키워드: 단어 임베딩, 자동분류, 듀이십진분류법(DDC), Word2vec, GloVe, fastText
word embedding, automatic classification, Dewey Decimal Classification(DDC), Word2vec, GloVe, fastText

* 경북대학교 문헌정보학과 부교수(yglee@knu.ac.kr)

■ 논문접수일자 : 2023년 11월 20일 ■ 최초심사일자 : 2023년 12월 8일 ■ 게재확정일자 : 2023년 12월 13일
■ 정보관리학회지, 40(4), 307-327, 2023. http://dx.doi.org/10.3743/KOSIM.2023.40.4.307

1. 서 론

다분히 고전적이고 상투적인 표현이지만, 컴퓨터는 기본적으로 0과 1로 이루어진 이진 체계로 데이터를 처리하고 이해한다. 이로 인해 컴퓨터는 태생부터 숫자를 계산하는데 있어서 일치감치 인간의 능력을 넘어섰다. 최근 들어 컴퓨터는 사물 인지(object detection)와 같이 인간만이 할 수 있다고 생각했던 영역에서도 두각을 나타내고 있어 사물에 대한 인식을 할 수 있는 것처럼 보인다. 하지만 이는 인간이 사물이나 텍스트, 더 기본적으로 언어를 이해하는 방식과는 엄연히 차이가 있다. 컴퓨터는 여전히 이러한 사물이나 텍스트에 대해 이진 체계의 숫자 형태로 입력받아 이들 데이터를 컴퓨터 방식대로 처리한다고 볼 수 있다.

문헌정보학 분야의 오랜 전통이며 아직도 특정 자원에 대한 가장 중요한 메타데이터 요소는 서명 또는 표제(title)이다. 원문(full-text)을 비롯하여 텍스트와 데이터가 넘쳐나는 시대임에도 불구하고 도서관 영역은 여전히 짧은 텍스트인 표제를 핵심 메타데이터 요소로 간주한다. 이는 우리가 알고 있는 한국목록규칙이나 기계자동화목록(MARC)의 기술 규칙을 보면 잘 알 수 있는데, 표제가 그만큼 자원의 내용을 가장 잘 대표하거나, 압축하여 표현하는 요소이기 때문이다.

문제는 이 짧은 문장이나 텍스트가 정보검색이나 기계학습과 같은 영역에서 좋은 성능을 가져오기에는 많은 어려움이 있다는 것이다. 일반적으로 이들 분야의 어플리케이션은 보다 풍부한 색인정보나 분류자질을 가져야 성능 향상을 기대할 수 있기 때문이다. 일례로 학술 논문의 제목만 이용하는 것보다 제목과 초록을 같이 활용하는 것이 더 좋은 성능을 가져옴을 알 수 있다. 특히 단행본 도서가 장서의 중심인 공공도서관이나 대학도서관의 경우 도서의 서명(book title)이 주요 분석 대상이 될 수밖에 없으므로 이러한 짧은 텍스트로 인한 한계를 안고 가야 하는 숙명일 수도 있다.

지난 10년 동안 자연언어처리(NLP) 분야에서는 많은 발전이 있었는데, 그 중 하나가 단어 임베딩(word embedding) 관련 영역이다. 이 개념은 한 단어가 말뭉치에서 사용된 용례를 바탕으로 주변 문맥을 활용하여 벡터 공간에 투영하는 기법으로, 단어의 구문론적(syntactic) 측면이나 의미론적(semantic) 측면의 함축적 정보를 담아낸다고 볼 수 있다. 달리 말하면 문맥 정보를 활용하는 단어 임베딩 기법을 통해 특정 단어를 벡터로 표현하게 되면, 전체 벡터 공간에서 유사한 의미의 단어가 가깝게 위치하게 되고 이러한 성질을 이용하면 벡터 연산 내지 활용을 통해 단어의 함축적 의미를 다양하게 처리할 수 있다. 만약 표제나 서명에 포함되는 단어에 대해 보다 풍부한 정보를 도출하여 짧은 텍스트의 한계를 넘어 응용 영역에 적용한다면 보다 풍부한 분석이나 성능 향상을 이끌어 낼 수 있을 것으로 보인다.

이에 본 연구는 짧은 텍스트인 단행본 서명에 출현한 단어에 대해 다양한 단어 임베딩 기법을 적용하여 단어 벡터를 추출하고, 이를 분류자질로 활용하여 단어 임베딩 기법에 따른 자동분류의 성능을 분석하고자 하였다. 구체적으로 먼저 주요한 단어 임베딩 기법에 대해 개관하고 다양한 분야에서 이들 기법을 활용한 선행 연구들을 살펴보았으며, 실험 데이터로 한 대학도서관에

서 수집한 사회과학분야의 단행본 신착 자료목록(이용구, 2020, 6)을 활용하였다. 이들 도서에 부여된 듀이십진분류법(DDC)의 강목을 범주로 설정하고 k-최근접 이웃(k-Nearest Neighbor, kNN) 분류기를 활용하여 자동분류 성능을 실험하였다. 이를 통해 짧은 텍스트인도서의 서명에 대해 단어 임베딩의 효과 여부와 성능 측면에서 어떤 특징이 있는지 파악할 수 있을 것으로 기대한다.

2. 이론적 배경

2.1 단어 임베딩의 개념

정보검색 분야에서는 오래 전부터 문헌이나 질의를 벡터로 표현하는 벡터공간 모형(Vector Space Model)이 고안되어 발전하였다(Salton, Wong, & Yang, 1975). 여기서 문헌 벡터는 다양한 가중치 기법을 적용하여 해당 문헌에 출현한 용어의 중요도를 수치로 나타내며, 코사인 유사계수와 같은 유사도 개념을 사용하여 문헌 벡터와 문헌 벡터 간에 비교하여 문헌 클러스터링을 수행하거나 문헌 벡터와 질의 벡터 간에 비교를 하여 정보검색에 활용한다. 이때 문헌이나 질의에 출현한 용어에 대해 벡터를 표현하는 가장 흔한 형태가 BoW(Bag-of-Words) 모형이다. 이 모형은 문헌에 출현한 용어들 사이의 관계를 독립되게 표현하여 문맥적 의미나 용어 출현의 순서를 무시하는 한계를 안고 있다. 또한 문헌 집단이 커질수록 수록된 용어가 늘어나 행렬이 커지며 문헌에 출현하지 않은 대부분의 용어가 가중치로 0의 값을 가지기에 희소 행렬(spare matrix)이 되어 유사도 계산에서 많은 연산과 컴퓨팅 자원을 필요로 하는 제약이 따른다.

최근 문헌 중심이 아닌 단어 중심의 벡터를 표현하는 단어 임베딩 기법이 자연 언어 처리, 기계 학습과 인공 지능 분야에서 많이 활용되고 있다. 이 기법은 단어나 문장을 벡터 공간으로 투영한다는 의미에서 임베딩(embedding)이라는 용어를 사용한다. 부연하여 설명하면 단어 임베딩을 다른 표현으로 단어의 분산 표현(distributed representation of words)이라고 하는데, 이 개념은 동일한 문맥에서 사용되고 출현하는 단어는 유사한 의미를 의도하는 경향이 있다는 분포 가설(distribution hypothesis)에 기반한다(Harris, 1954). 즉 문맥 측면에서 보면 비슷한 의미를 가진 단어는 주변 단어의 분포도 유사하다는 것을 뜻한다. 여기서 분산 표현(Hinton, McClelland, & Rumelhart, 1986)은 대상이나 객체를 하나의 측면이 아닌 여러 측면을 나누어 표현하기 위해 다차원 공간에 벡터화하는 것을 말한다. 요약하면 단어의 문맥을 활용하여 분산 표현을 생성함으로서 단어를 의미적으로 벡터화하는 작업을 단어 임베딩이라 부르며, 이렇게 표현된 벡터를 임베딩 벡터(embedding vector)라 한다.

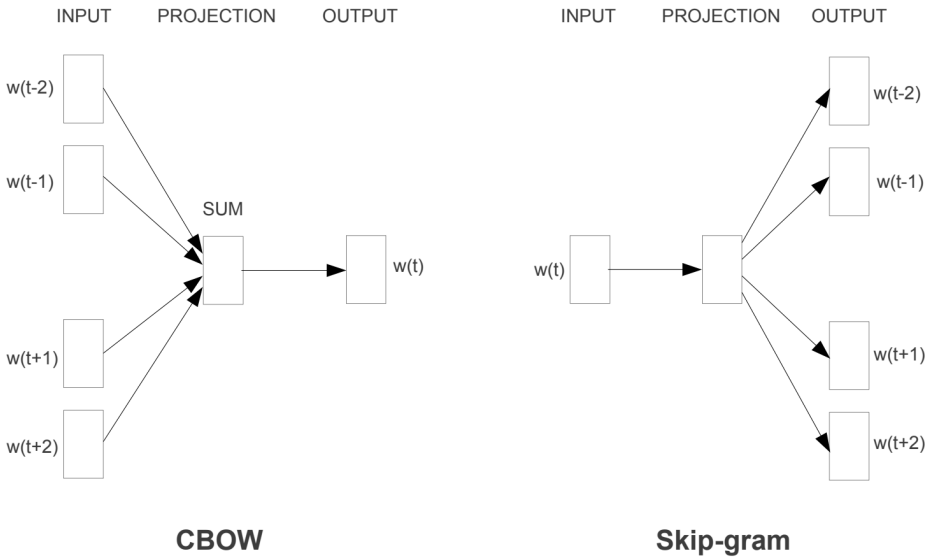
따라서 임베딩은 결과물의 표현 형식으로 벡터를 취하며, 이 벡터의 요소는 정수가 아닌 실수(real number) 값으로 표현된다. BoW 모형의 벡터는 희소 행렬인데 반해, 임베딩 벡터는 하이퍼파라미터로 설정한 크기에 해당하는 밀집 벡터(dense vector)가 되며 이를 결합하여 밀집 행렬(dense matrix) 형태로 만들어지므로 BoW 모형에 비해 연산과 컴퓨팅 자원에서 장점을 갖

는다. 더 나아가 서로 다른 단어를 대상으로 구문론적이나 의미론적 측면의 연산도 가능하게 하는 장점도 있다.

대표적인 단어 임베딩 모형은 Word2vec (Mikolov et al., 2013a), GloVe(Pennington, Socher, & Manning, 2014)와 fastText (Bojanowski et al., 2017) 등이 있는데, 이들은 임베딩의 대상을 단어에 초점을 맞추고 있어 전통적인 단어 임베딩 모형이라고 한다. 반면, 최근 딥러닝 분야에서 문장이나 문헌 전체를 이용하여 그 안에 출현한 단어와 더 세부적으로 각각의 하위단어 또는 토큰에 대해 임베딩하여 이들 단어나 하위단어가 가지는 의미론적, 구문론적 특징을 추출하는 문맥적 임베딩(contextual embedding 또는 contextualized word embedding) 모형들이 개발되어 높은 성능을 보여주고 있다(McCann et al., 2017). 이러한 대표적인 모형으로 CoVe(McCann et al., 2017), ELMo(Peters et al., 2018) 그리고 BERT (Devlin et al., 2019) 등이 있다.

2.2 단어 임베딩 모형

Tomas Mikolov는 그의 동료들과 함께 2013년에 두 편의 논문(Mikolov et al., 2013a; Mikolov et al., 2013b)을 통해 수백만 단어의 어휘와 수십억 어절로 구성된 대규모의 말뭉치로부터 높은 수준의 단어 벡터를 학습하기 위해 사용하는 새로운 기법인 Word2vec 모형을 제시하였다. 이들 연구에서는 Word2vec 모형에 대해 서로 다른 구조(architecture)를 갖는 두 종류의 하위 모형을 제시하였는데, 하나는 CBOW 모형(continuous Bag-of-Words Model)이고, 다른 하나는 Skip-gram 모형(continuous Skip-gram Model)이다. 이들 모형의 구조를 보면, <그림 1>과 같다.



<그림 1> Word2vec의 CBOW와 Skip-gram 구조(Mikolov et al., 2013a)

〈그림 1〉의 좌측을 보면, CBOW 모형은 순방향 신경망 언어 모델(feed-forward neural network language model, NNLM)과 유사하게 입력노드와 출력 노드를 가지고 있다. 일반적으로 순방향 신경망 언어 모형은 입력층, 투사층(projection layer), 은닉층, 출력층으로 구성되는데 그림의 CBOW 모형은 활성화 함수가 존재하는 은닉층이 제거된 구조와 같다. 또한 일반적으로 딥 러닝이 다수의 은닉층을 갖는 것에 비해 Word2vec 모형은 하나의 은닉층을 가져 얕은 신경망에 해당한다 할 수 있다.

CBOW 모형에서 좌측의 입력층은 대상 단어인 w(t)의 전후 문맥에 해당하는 주변 단어를 입력하게 되며, 전후 문맥의 범위를 어느 정도까지 반영할지를 결정해야 하는 크기를 문맥창(window)으로 표기한다. 이 그림에서는 문맥 창의 크기가 2이며 대상 단어를 중심으로 좌측의 단어 w(t-2), w(t-1) 2개와 우측의 단어 w(t+1), w(t+2) 2개를 포함하여 총 4개 단어가 입력되는 것을 알 수 있다. 일반적으로 기계학습에서 투사 또는 투영(projection)은 고차원의 데이터를 저차원의 데이터로 축소하는 개념을 말하는데, Word2vec에서 유일한 은닉층인 두 번째 층이 투사층에 해당한다. 이 투사층은 문맥 창에 포함된 각 단어에 대한 입력층의 벡터(one-hot vector)에 대해 신경망의 파라미터인 가중치를 곱하고 더한 다음, 평균을 내어 임베딩 벡터를 구하게 된다. 투사층을 역전파와 확률적 경사 하강법(stochastic gradient descent)을 통해 학습하여 최적화를 거치게 되며 최종적으로 대상 단어에 대한 임베딩 벡터로 생성한다(Mikolov et al., 2013a). Word2vec는 이렇게 변형된 투사층을 적용하여 NNLM 보다 시간복잡성이 낮아져 비교적 좋은 연산 효율을 갖는다.

〈그림 1〉에서 오른쪽 부분은 Word2vec의 두 번째 모형인 Skip-gram에 해당하는데, 이 모형은 CBOW와 반대로 대상 단어 w(t)를 입력하고 출력층에서는 주변 단어들인 w(t-2), w(t-1), w(t+1), w(t+2) 4개가 제시된 것을 볼 수 있다. 즉 CBOW 구조가 주변 문맥에 출현한 단어를 이용하여 대상 단어를 예측한다면, Skip-gram 구조는 대상 단어를 이용하여 주변 문맥에 출현한 단어를 예측한다고 할 수 있다. Mikolov el al.(2013a)에 의하면 Skip-gram 모형이 소규모 데이터셋에서 보다 잘 작동하며 저빈도 단어의 임베딩에서 더 좋은 성능을 보이는 것으로 나타났다. 반면 CBOW 모형은 Skip-gram에 비해 계산량이 적기 때문에 학습이 빠르며 고빈도 단어에 대해 더 좋은 임베딩 표현을 가져오는 것으로 나타났다.

Word2vec 모형은 계층적 소프트맥스(hierarchical softmax)나 네거티브 샘플링(negative sampling) 방법을 적용하여 일반 인공 신경망에 비해 계산 복잡도가 낮아 빠른 처리가 가능하다(Mikolov et al., 2013b). 일반적으로 인공신경망에서 오차를 최소화시키기 위해 역전파를 이용하는데 Word2vec 모형의 출력층은 데이터셋에 출현한 모든 단어가 되므로 이들 단어에 대해 활성화 함수를 모두 계산하게 되므로 계산 복잡도가 매우 커지게 된다. 따라서 기존 소프트맥스의 계산량을 줄이되 소프트맥스에 근사시키기 위해 트리 구조를 적용한 계층적 소프트맥스를 사용한다. 또한 계층적 소프트맥스 대안으로 적용할 수 있는 네거티브 샘플링은 역전파에서 모든 단어에 대해 가중치를 업데이트하는 것이

아니라, 대상 단어와 이 단어와 의미적으로 전혀 관련이 없는(부정적인) 단어들 중에 일부를 샘플링하여 해당 단어만 함께 적용하여 최종 손실값을 계산한다. 이렇게 함으로 모든 단어에 대해 손실값을 구하지 않으므로 인해 많은 연산을 피해갈 수 있다.

단어 임베딩의 평가는 내재적 유형과 외재적 유형으로 나누어 볼 수 있다(Wang et al., 2019). 내재적 유형은 임베딩 표현 자체의 품질을 평가하는 반면, 외재적 유형은 자연 언어 처리나 자동분류와 같이 최종적으로 해결하려는 다운스트림(downstream) 과업에 임베딩 표현을 활용하여 그 과업의 성능을 측정하여 평가한다. 구체적으로 내재적 평가는 주로 다음에 설명하는 의미론적 유사성과 구문론적 유추로 나누어 임베딩 성능을 평가하며, 외재적 평가는 임베딩된 결과를 활용한 최종 과업에 따라 해당 영역에서 주로 사용하는 척도를 통해 성능을 평가하기에 다양하다.

내재적 평가에서 의미론적 유사성이란 두 단어의 임베딩 표현인 벡터에 대해 흔히 정보검색이나 자동분류에서 두 대상물의 유사도를 측정하는 방식처럼 코사인 유사계수 등을 이용하여 측정하며, 관련 있는 단어 쌍의 의미 관계를 파악할 수 있다. 의미론적 유사성을 측정하는 대표적인 예로 'man' : 'king' :: 'woman' : ?라는 표현식에서 물음표에 해당하는 답을 찾기 위해 벡터 연산의 수학적 표현식인 vec('man') - vec('king') + vec('woman')을 입력하면 'queen'라는 결과를 얻을 수 있다.

언어학의 구문론이 주요하게 문장의 구조나 문법적 기능 또는 구성 요소에 대해 다루듯이 단어 임베딩의 평가에서 구문론적 유추는 단·복수형이나 비교급, 시제나 문법적 성(grammatical gender) 등을 구문적인 측면에서 얼마나 잘 비교하여 표현하는지 측정한다. 구문론적 유추도 'walking' : 'walked' :: 'swimming' : ?라는 표현식의 연산 결과로 'swam'을 얻을 수 있다. Word2vec의 성능은 하이퍼파라미터인 데이터셋의 규모, 임베딩 벡터의 차원, 에포크(epochs)의 수, 모형의 구조 차이에 따라 달라진다.

단어 임베딩에서 Word2vec 모형 이외에 주요 모형으로 fastText(Bojanowski et al., 2017)와 GloVe(Pennington, Socher, & Manning, 2014)가 있다. fastText는 하위단어(subword) 모형으로 지칭되는데, Word2vec 모형이 문헌에 출현한 단어에 대해 임베딩 표현을 이용한다면 fastText 모형은 대상 단어를 n-gram 방식의 문자 수준의 하부문자열과 대상 단어 자체를 함께 이용하여 임베딩 벡터를 생성하고, 최종적으로 대상 단어의 벡터 표현은 이들 하부문자열에 의해 생성된 임베딩 벡터의 총합으로 나타낸다. 이러한 방식의 장점은 n-gram 방식에 의해 생성된 하위단어를 공통으로 가지는 서로 다른 단어들 사이에 임베딩 표현을 상호 공유할 수 있으며, 더 나아가 희귀 단어나 미등록어휘(out-of-vocabulary)에 대해 이러한 공유를 적용할 수 있어 보다 신뢰성 있고 유연한 학습이 가능하다. 또한 일본어와 같이 n-gram 방식이 효율적인 언어에서 더 좋은 임베딩 성능을 보이며, 학습 데이터의 규모를 줄이거나 작은 데이터셋에서 미등록어휘가 증가하여도 fastText 모형은 성능 측면에서 강건함을 보인다(Bojanowski et al., 2017).

GloVe 모형은 앞서의 두 모형과 달리 말뭉치 또는 데이터셋의 전체에서 단어의 통계 빈

도를 사용한다. Word2vec의 경우 하이퍼파라미터로 주어진 문맥의 크기인 윈도우에 출현한 단어만 이용하여 임베딩 벡터를 생성하기에 전체 말뭉치에서 보여지는 통계를 활용하지 않는다. 즉 Word2vec와 fastText 모형은 주어진 문맥 범위 내에 출현한 단어를 활용하지만 윈도우의 범위를 벗어나서 문헌 전체에 나타나는 동시출현 빈도는 계산할 수 없다. 반면, GloVe 모형은 말뭉치 전체에서 추출한 단어-단어 동시출현 빈도를 사용하고 행렬 인수분해(matrix factorization) 방법을 적용하여 임베딩 표현을 생성한다. GloVe 모형에서 대상 단어에 대한 임베딩 벡터를 구하기 위한 동시출현 확률은 연관된 다른 단어에 의해 확률의 비율을 적용하여 구하며 이때 계산 복잡도를 줄이기 위해 행렬에서 0이 아닌 값을 대상으로 연산을 수행하며, 손실함수는 다른 모형과 동일하게 소프트맥스를 사용한다(Pennington, Socher, & Manning, 2014).

2.3 국내외 선행 연구

단어 임베딩은 그 자체적으로 품사 태깅, 얕은 구문 분석(syntactic chunking), 개체명 인식(NER) 등의 자연언어처리 분야에 적용된다. 또한 단어 임베딩 기법은 자동분류, 감성분석(Chen & Sokolova, 2021), 기계번역(Sitender et al., 2023) 등과 같은 다운스트림 과업에서 전처리 과정에서 자질 표현을 생성하기 위해 적용된다.

자동분류 과업 중심으로 선행 연구들을 정리하면 Yang, Macdonald, Ounis(2018)는 단어 임베딩을 학습시키기 위한 서로 다른 배경의 말뭉치와 하이퍼파라미터가 자동분류의 성능에 영향을 미친다고 보고 선거 기간 동안 수집된 트위터 데이터셋을 이용하여 단어 임베딩을 통한 합성곱 신경망(convolution neural network, CNN)의 분류 성능을 TFIDF를 적용한 SVM 모형과 비교 분석하였다. 그 결과 임베딩을 학습시킬 때 사용한 말뭉치에 따라 성능 차이를 보이며 짧은 문장의 트위터 데이터에는 큰 문맥 윈도우와 대규모 자원이 더 좋은 성능을 가져옴을 보여주었다. 다만 이 연구에서는 학습횟수(epoch)에 따른 성능을 제시하지 않았다. 이 연구 이외에 트위터 데이터를 대상으로 의미론적인 자질 표현을 획득하기 위해 임베딩 기법을 활용하는 연구(Goularas & Kamis, 2019) 등이 다수 진행되었는데, 이는 단어 임베딩이 트위터 데이터에 대해 효과적이기 때문으로 비슷하게 짧은 문장으로 이루어진 서지데이터에 대해서도 주목할 할 필요가 있다.

합성곱 신경망 알고리즘을 이용한 텍스트를 자동분류 하는 연구(Dharma et al., 2022)는 20개의 토픽을 가진 1만 9천여 개의 뉴스기사로 구성된 UCI KDD 아카이브를 대상으로 Word2vec, GloVe, fastText 세 모형의 성능을 비교 분석하였는데 fastText 모형이 가장 좋은 정확도를 보였다. 다만 이들 세 모형의 정확도의 차이는 유의미하게 크진 않았다. 유사한 연구로 Wang, Nulty, Lillis(2020)도 Reuters-21578 등을 포함하여 다중 범주를 갖는 다수의 데이터셋에 대해 전통적인 세 개의 단어 임베딩 모형뿐만 아니라, ELMO와 BERT 모형과 같은 문맥적 임베딩까지 포함하여 두 부류의 임베딩으로 나누고 합성곱 신경망과 BiLSTM 알고리즘을 이용하여 임베딩에 따른 자동분류 성능을 평가하였다.

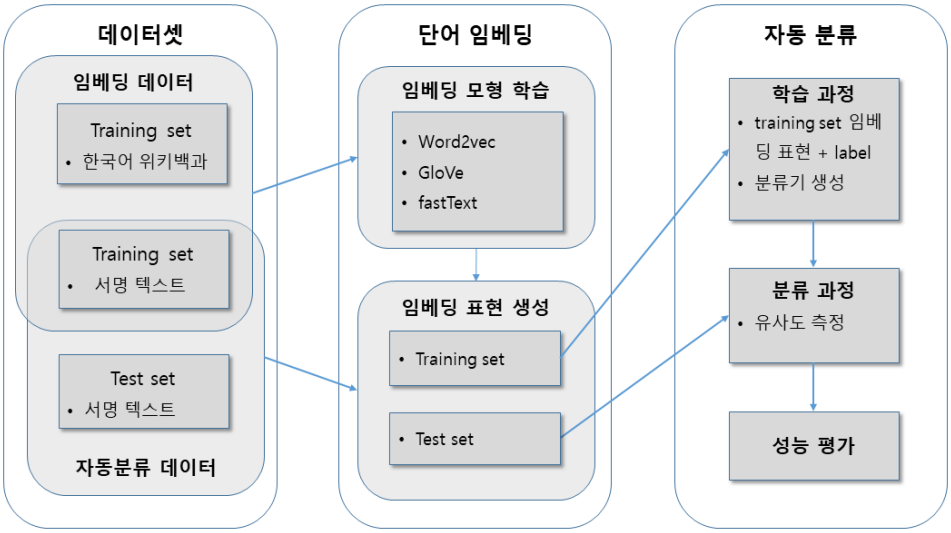
한국어 관련하여 단어 임베딩 연구를 살펴보면 Park et al.(2018)은 교착어에 속하는 한국어의 특성을 반영하여 하위단어를 글자(음절) 단위와 자모 단위로 분해하고 이들에 대한 Skip-gram 모형을 통해 하위단어 벡터를 학습하고 의미론적 유사성, 구문론적 유추, 그리고 감성 분석을 통한 평가를 수행하였다. 자모 단위 분해는 한글의 한 글자를 구성하는 초성, 중성, 종성을 분리하여 각각의 낱글자(자음과 모음)를 순서대로 추출하되 이를 n-gram 방식으로 처리하였다. 이때 종성이 없는 경우 그 의미를 나타내기 위해 특수 문자로 대체하여 적용하였다. 데이터셋으로는 한국어 위키백과, 뉴스 기사, 그리고 세종 말뭉치를 사용하였으며 단어 단위의 Word2vec 모형과 글자, 자모 단위로 fastText 모형을 적용하되 하이퍼파라미터는 차원 300, 네거티브 샘플 5, 윈도우 크기 5로 설정하고 각 모형에 대해 n-gram의 크기를 달리하여 성능 평가를 수행하였다. 그 결과 세 개의 평가 영역에서 글자 단위 1-6 크기의 n-gram과 자모 단위 3-5 크기의 n-gram을 조합한 fastText의 Skip-gram 모형이 다른 모형에 비해 더 우수한 성능을 보였다.

강형석과 양장훈(2020)은 Word2vec와 fastTest 모형을 비교하기 위해 두 모형에 대해 다양한 하이퍼파라미터, 글자(음절) 단위, 자모 단위 그리고 품사 태깅을 포함한 5가지 방식으로 나누어 나무위키와 위키백과 데이터를 이용하여 학습한 후, 평가 방법으로 의미론적 유사성과 구문론적 유추를 적용하였다. 유사성 평가에서는 자모 단위 fastText가 더 우수한 결과를 보였으며 유추 평가에서는 종합적으로 품사 태깅을 적용한 Word2vec가 다소 우수한 결과를 보였다. 다만 임베딩 모형을 구축할 때 Skip-gram 모형에 한정하고 비교적 단순한 하이퍼파라미터만 적용한 한계를 가진다.

이다빈과 최성필(2019)은 대규모 말뭉치를 대상으로 크기나 종류에 따른 주요 단어 임베딩 모형의 성능을 비교 평가하였다. 말뭉치로는 세종 말뭉치와 크롤링한 뉴스기사 데이터셋을 활용하였으며 이들 데이터에 대해 다양한 하이퍼파라미터에 적용하여 형태소 단위를 기반으로 하는 단어 임베딩을 생성하였다. 세종 말뭉치에서는 fastText 모형이 유사성과 유추 평가 모두에서 성능이 우수했으며, 뉴스 기사에서는 Word2vec 모형이 다소 우수하였다. GloVe 모형의 경우 두 말뭉치에서 가장 낮은 성능을 보였다. 세종 말뭉치에서 차원과 윈도우가 커질수록 fastText 모형이 우수하였으며, 유추 성능에서는 차원 50과 윈도우 크기 3에서는 Word2vec 모형이 우수함을 보였다.

3. 실험 설계

이 연구는 비교적 짧은 텍스트에 해당하는 단행본 도서의 표제 또는 서명을 대상으로 임베딩 기법을 적용하여 분류자질 표현으로 단어 벡터를 생성하고 이를 이용하여 도서에 부여된 DDC 강목에 대한 자동분류를 수행하였다. 이 과정은 크게 두 부분으로 나누어 볼 수 있는데, 하나는 표제에 출현한 용어에 대해 단어 임베딩 기법을 적용하는 과정이고 다른 하나는 임베딩 결과를 텍스트 자동분류에 활용하여 최종 성능을 평가하는 과정이다. 보다 자세한 실험 과정은 〈그림 2〉와 같다.



〈그림 2〉 실험 과정의 개요

먼저 이 연구에서 사용한 자동분류용 실험 데이터는 사회과학분야 도서의 서명과 통계학(Statistics) 분야를 제외한 300대의 강목에 해당하는 DDC 분류기호로 구성된 6,207건의 서지데이터에 해당하는데, 기존의 연구(이용구, 2020)에서 구축된 것을 추가로 크리닝하여 얻었다. 이 중에서 학습셋은 4,965건이며, 테스트셋은 1,242건이다.

일반적으로 임베딩을 통해 단어의 벡터 표현을 구하기 위해서는 학습 데이터로써 대규모의 말뭉치가 필요하다. 말뭉치의 규모가 클수록 임베딩하고자 하는 단어의 주변 문맥이 다양하고 풍부해져서 그 단어의 정확한 의미를 보다 잘 반영할 수 있기 때문이다. 앞서 단어 임베딩 모형을 제시한 선행 연구들(Bojanowski et al., 2017; Mikolov et al., 2013a; Pennington, Socher, & Manning, 2014)도 대부분 수십억의 어절과 수백만의 어휘를 갖는 데이터셋을 사용하였다. 이 연구에서는 단어 임베딩 모형을 학습시키기 위해 데이터셋으로 대표적으로 많이 사용되는 대규모 한국어 말뭉치인 한국어 위키백과(https://ko.wikipedia.org/)를 활용하였다. 또한 임베딩 모형에 서명 텍스트의 특성을 반영하기 위해 자동분류용 실험 데이터 중에서 학습 데이터셋(training set)도 포함하였다. 이들 전체 데이터셋에 대해 한국어는 Mecab 형태소 분석기에 의해 단어를 형태소 단위로 구분하여 토크나이즈를 적용하였으며, 영어의 경우 특수문자 제거와 소문자화를 적용하였다. 최종적으로 실험에 쓰인 임베딩용 학습 말뭉치는 약 2억 개의 어절과 1백만 개의 고유 단어로 구성되었다. 이는 주요 단어 임베딩 모형을 제시한 외국 선행 연구들에서 사용한 말뭉치 보다 규모가 크게 작아서 임베딩 성능에 제약이 될 수 있다.

전통적인 단어 임베딩 영역에서는 주로 3가지 모형들(Word2vec, GloVe, fastText)이 사용된다. 이 연구에서도 적합한 단어 임베딩 모형을 찾아내기 위해 이들 모형 모두를 이용하

였다. 위키백과와 자동분류를 위한 서명 텍스트의 학습 데이터셋을 활용하여 세 모형의 임베딩 학습을 수행하였으며 이때 다양한 하이퍼파라미터를 설정하였다. 하이퍼파라미터는 모형에 따라 약간 상이한데, 세 모형의 공통된 하이퍼파라미터로는 임베딩 벡터의 크기를 나타나는 차원, 대상 단어의 주변 문맥의 크기인 윈도우, 그리고 학습 횟수에 해당하는 에포크가 있다. 이외에 Word2vec와 fastText의 경우 모형의 구조 유형(CBOW 또는 Skip-gram), 계층적 소프트맥스나 네거티브 샘플링 적용 여부가 있다. 마지막으로 fastText는 단어에 대해 n-gram 방식을 적용할 때 최소 최대 크기를 설정해야 한다. Word2vec와 fastText 모형은 오픈 소스인 gensim(https://radimrehurek.com/gensim/) 패키지를 사용하였으며, GloVe는 원저자가 공개한 패키지(https://nlp.stanford.edu/projects/glove/)를 사용하여 임베딩 표현을 추출하였다.

하이퍼파라미터를 구체적으로 살펴보면 차원(약어로 dim 표기)은 300, 500, 768 크기로 설정하였으며 패키지의 기본값인 100과 그보다 작은 50의 경우 성능이 상대적으로 크게 낮아 실험에서 제외하였다. 768 크기의 차원은 최근 좋은 성능을 가져오는 BERT 모형의 임베딩 차원의 기본값과 동일한 값으로 향후 비교분석을 위한 가능성을 고려하였다. 문맥 또는 윈도우의 크기(약어 ws)와 네거티브 샘플링(약어 ns)의 경우 gensim 패키지의 기본값인 5부터 10, 15, 20까지 적용하였으며 에포크(약어 ep)도 기본값인 5부터 20, 50, 100까지 적용하였다. 다만 임베딩 학습에 사용된 데이터셋의 크기가 작아 20 이하 에포크의 분류 성능은 크게 저하되어 50과 100 값 위주로 적용하였다. fastText 모형에서 n-gram의 최소 크기와 최대 크기의 범위는 기본값인 3-6과 1-3, 2-5 범위를 추가적으로 적용하였다. 이외의 하이퍼파라미터는 패키지의 기본값을 그대로 적용하였다.

자동분류를 위한 기계 학습 분류기는 텍스트 분류에서 많이 사용되고 성능도 우수하며 비교적 구현과 이해가 쉬운 kNN 분류기 모형을 사용하였다. 이 분류기에서 중요한 변수 또는 하이퍼파라미터인 최적의 k값을 결정하기 위해 이 연구에서는 k값을 1부터 100까지 변화시켜 최고의 마이크로 평균 F1(micro-average F1, microF1)을 보이는 값을 기준으로 분류기의 성능을 평가하였다. 분류해야 할 테스트 문헌에 대해 학습 데이터셋에서 가장 가까운 문헌을 찾기 위해 유사도로 코사인 유사계수를 이용하였으며, 이러한 과정을 구축하고 분류성능을 평가하기 위한 scikit-learn 패키지(https://scikit-learn.org/)를 사용하였다. 또한 임베딩의 효과와 분류 성능을 비교 평가하기 위해 TFIDF 가중치 기반의 kNN 분류기를 기준 모형(baseline model)으로 설정하였다.

Word2vec와 같이 은닉층이 하나인 얕은 신경망을 채택한 단어 임베딩의 경우 인공 신경망을 이용하는 다른 기법들처럼 초기에 신경망의 가중치를 무작위의 작은 값으로 초기화하고 확률적 경사 하강법 알고리즘을 적용하므로 그때마다 임베딩 벡터가 미세하게 다른 값을 가지며 이는 최종 결과인 자동분류 성능에 영향을 미친다. 따라서 보다 정확한 최종 성능을 얻기 위해 이 연구에서는 특정 하이퍼파라미터에 따른 각 모형마다 최소 3번에서 최대 5번까지 동일한 실험을 수행하고 이를 평균하여 최종 성능으로 간주하였다. 일반적으로 실험을 3번씩 반복하여 평균하여 산출하였지만, 그 성능이 최상위권에

드는 경우 보다 정확한 결과를 얻기 위해 최대 2번을 추가로 같은 실험을 실시하고 모두 평균하여 최종 성능을 산출하였다.

단어 임베딩 모형의 다양한 하이퍼파라미터에 따른 kNN 분류기의 성능 평가는 다양한 척도를 적용하였다. 이 연구가 DDC 300대의 9개 강목을 분류 범주로 사용하고 도서가 하나의 범주에 부여되는 다중클래스(multi-class) 분류에 해당하므로 마이크로 평균 F1 값을 포함하여 마이크로 평균 정확률과 재현율, 정확도(accuracy)가 모두 같은 값이므로 대표로 기본인 마이크로 평균 F1 척도를 평가 기준으로 하였다. 추가적으로 매크로 평균 F1(macro-average F1, macroF1) 척도를 활용하였으며, 보다 세부적으로 이 값을 구하기 위해 필요한 매크로 평균 정확률(precision)과 재현율(recall)도 사용하였다.

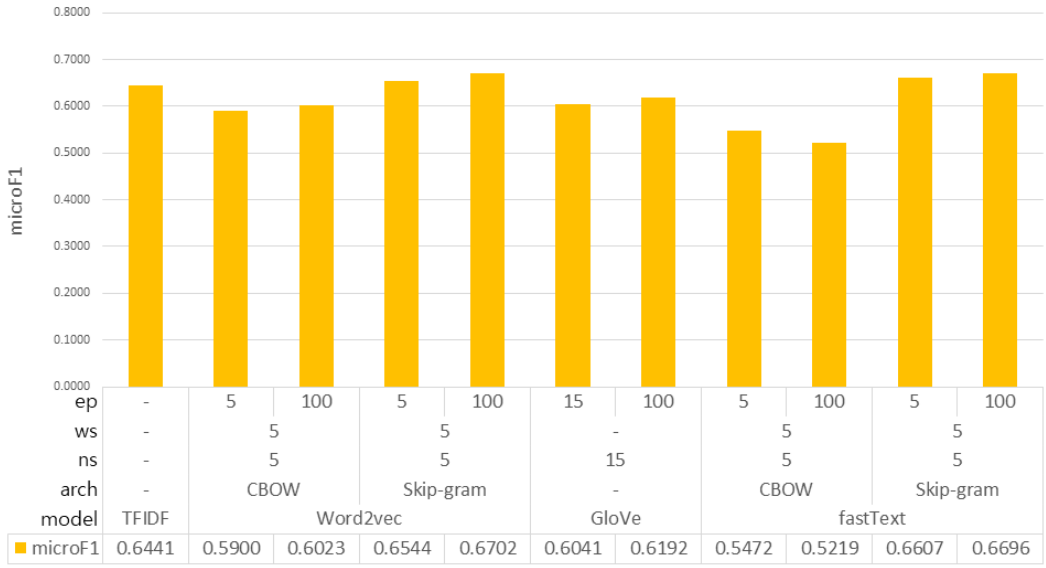
4. 단어 임베딩에 따른 자동분류 실험

4.1 임베딩 모형의 분류 성능

대규모 말뭉치로 학습된 단어 임베딩 모형(Word2vec, GloVe, fastText)을 이용하여 단행본 도서의 서명 텍스트를 임베딩 벡터로 표현하여 kNN 분류기를 구축하고 사회과학 분야 도서에 부여된 DDC 300대 강목을 자동 분류하였다. 전통적인 문헌 내 단어빈도와 역문헌빈도를 이용하여 벡터 표현에 의한 kNN 분류기를 적용한 기준 모형(TFIDF)을 구축하고 단어 임베딩 모형의 분류 성능을 서로 비교 평가하였다. 단어 임베딩 모형은 일부 공통된 하이퍼파라미터를 가지기도 하지만 서로 다른 고유한 하이퍼파라미터를 가지므로 실험을 완전히 동일하게 설정하여 평가하기는 어렵다. 따라서 공통된 하이퍼파라미터 이외에는 패키지가 제공하는 기본값(default)을 적용하였다. 구체적으로 모든 단어 임베딩 모형의 차원은 동일하게 300이며, Word2vec와 fastText 모형은 각각 CBOW와 Skip-gram 구조를 설정하였으며, 그리고 gensim 패키지의 기본값으로 문맥 크기 5, 네거티브 샘플링 5, 에포크 5를 설정하였다. GloVe 모형은 패키지 기본값으로 문맥 크기 15와 에포크 15를 설정하였다. 다만 임베딩 표현을 얻기 위한 학습용 말뭉치의 크기가 다소 작아 성능 하락으로 인해 에포크 100을 추가하여 실험하였으며 그 결과 성능은 〈그림 3〉과 같다.

단어 임베딩 모형에서 에포크 5회 내지 15회를 기준으로 자동분류 성능을 살펴보면, fastText의 Skip-gram 모형이 마이크로 평균 F1 0.6607로 가장 좋은 성능을 보였으며, 다음으로 Word2vec의 Skip-gram 모형 0.6544의 값을 보였다. 이들 Skip-gram 모형은 TFIDF와 코사인 유사도를 적용한 기준 모형(0.6441) 보다 근소하게 좋은 모습을 보였다. GloVe 모형의 성능은 Skip-gram 모형과 기준 모형 보다 크게 낮았으나 CBOW 모형들 보다는 거의 같거나 약간 높았다. 다만 fastText의 CBOW 모형이 가장 낮은 성능을 보이는 것으로 나타났다.

학습 횟수인 에포크를 100회로 높였을 때는 Word2vec의 Skip-gram 모형이 0.6702로 가장 높은 성능을 보였으며, 그 다음으로 fastText의 Skip-gram 모형(0.6696)이 거의 비슷한 성능을 보였다. 많은 횟수의 에포크를 통해 Word2vec의 Skip-gram 모형이 보다 더 큰 폭의 성능 향



〈그림 3〉 기본 모형에 따른 자동분류 성능(300차원 기준)

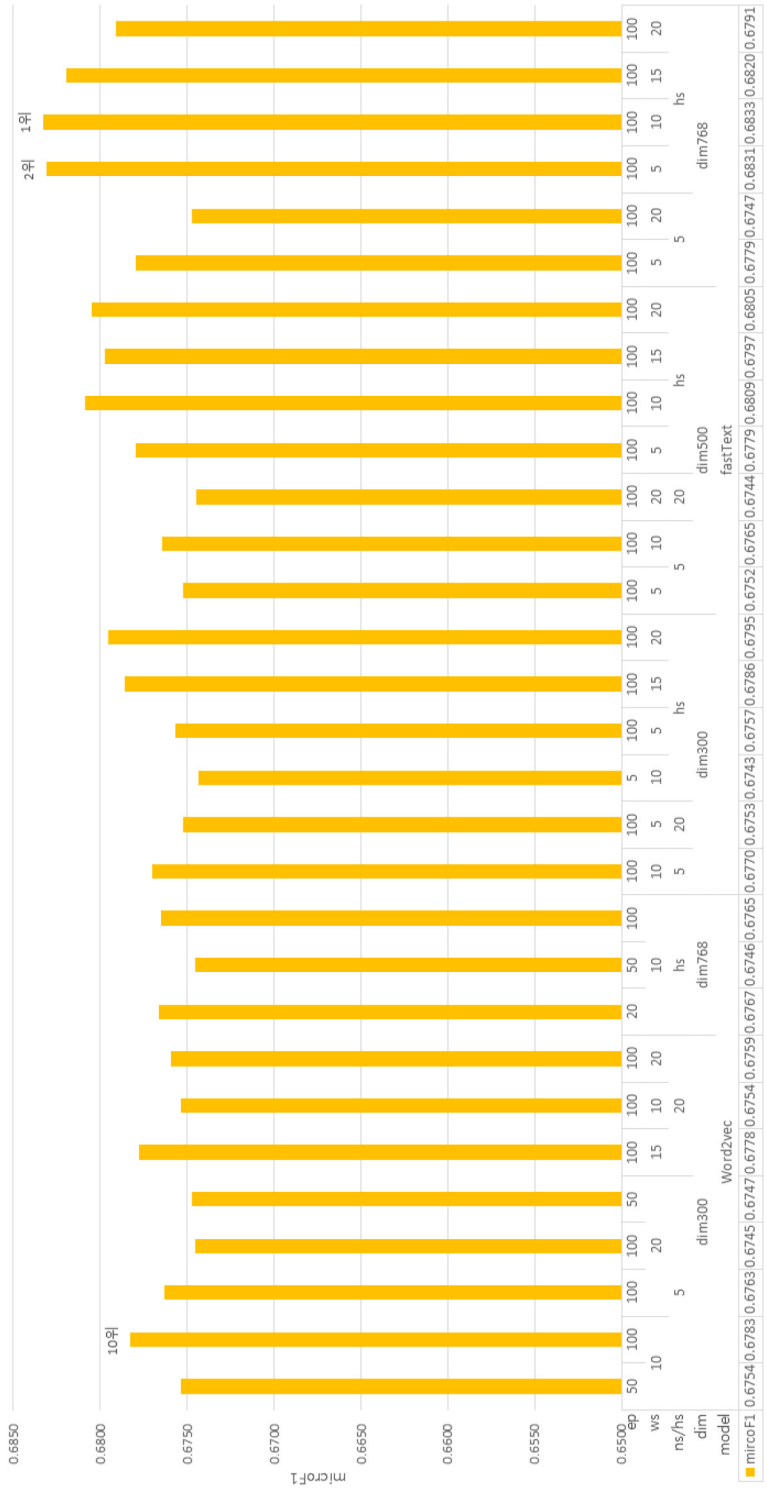
범례: 에포크(ep), 문맥 크기(ws), 네거티브 샘플링(ns), 모형 구조(arch), 모형명(model)

상을 가져왔다. GloVe 모형은 에포크를 늘려도 큰 차이가 없었으며 fastText의 CBOW 모형은 에포크 5회보다 100회에서 오히려 많은 학습에 의한 과적합으로 성능 하락을 보였다.

4.2 하이퍼파라미터 최적화와 분류 성능

단어 임베딩 모형은 다양한 하이퍼파라미터를 가지며 이 값을 상황에 맞게 조정함으로써 성능의 향상 내지 최적화를 꾀할 수 있다. 앞서 설명하였듯이 주요한 하이퍼파라미터는 임베딩 벡터의 크기인 차원(dim), 학습횟수인 에포크(ep), 네거티브 샘플링이나 계층적 소프트맥스 사용 여부(ns/hs), 문맥 또는 윈도우 크기(ws) 등이 있다. 기준 모형과의 성능 비교에서 우수한 모습을 보인 Word2vec와 fastText의 Skip-gram 모형을 대상으로, 하이퍼파라미터에 따른 자동분류 성능을 알아보기 위해 다양한 실험을 수행하였다. 모든 하이퍼라미터의 조합은 총 480개의 가짓수를 갖지만, 하이퍼파라미터에 따른 전반적인 성능을 알아보기 위해 극단적인 값(예로 ns 5와 ws 20, 또는 그 반대)과 중간 값(ns 10와 ws 10)의 조합 위주로 220개의 조합으로 실험을 진행하였다. 그 결과 마이크로 평균 F1 척도를 기준으로 상위 30개에 해당하는 성능과 하이퍼파라미터를 제시하면, 〈그림 4〉와 같다.

차원 768, 계층적 소프트맥스 사용, 윈도우 크기 10, 그리고 에포크 100의 하이퍼파라미터로 설정한 fastText의 Skip-gram 모형이 마이크로 평균 F1 0.6833으로 가장 우수한 분류성능을 보여주었다. Word2vec의 Skip-gram 모형에서는 차원 300, 네거티브 샘플링 5, 윈도우 크기 10, 그리고 에포크 100으로 설정한 경우가



〈그림 4〉 하이퍼파라미터에 따른 마이크로 평균 F1 성능(상위 30위 기준)

마이크로 평균 F1 값 0.6783으로 가장 좋은 성능을 보이지만, 이는 전체에서는 10위에 해당하였다. 즉 1위부터 9위까지 다양한 하이퍼파라미터에 따른 fastText 모형이 차지하였으며 전체 상위 30위 중에서 19개가 fastText 모형이며, 11개가 Word2vec 모형에 해당하여 fastText 모형이 전반적으로 더 좋은 성능을 보임을 알 수 있다. 다만, 〈그림 4〉에서 보이는 것처럼 하이퍼파라미터에 따른 상위 30개의 실험 사이에서는 성능이 큰 폭으로 차이가 나진 않는다. 하지만 앞서 〈그림 3〉에서 제시된 기본 모형의 성능(0.6441)과 하이퍼파라미터를 최적화하여 얻는 〈그림 4〉의 분류성능과는 약 3~4% 정도 차이가 나며, 패키지의 기본값에 의한 차이도 적게는 1.5% 많게는 약 3% 정도에 해당함을 알 수 있다.

다양한 하이퍼파라미터 중에서 단어 벡터의 크기인 차원에 따른 성능의 변화 경향을 살펴보면, Word2vec의 Skip-gram은 300차원처럼 낮은 차원에서 좋은 성능을 보이는 반면, fastText의 Skip-gram 모형은 높은 차원에서 좋은 성능을 보이는 것을 알 수 있다. 또한, fastText 모형의 경우 성능에 보다 결정적인 하이퍼파라미터는 계층적 소프트맥스의 사용 여부이다. 이 모형에 대해 동일한 차원 크기로 고정하면 하이퍼파라미터로 네거티브 샘플링 보다 계층적 소프트맥스를 적용한 실험이 대부분 더 좋은 성능을 보이는 것을 알 수 있다.

Word2vec 모형에서 네거티브 샘플링이나 계층적 소프트맥스 관련 하이퍼파라미터를 살펴보면 주로 300 차원의 네거티브 샘플링 또는 768 차원의 계층적 소프트맥스를 적용한 경우가 상위권의 성능을 보인 반면, 나머지 차원이나 하이퍼파라미터는 상대적으로 좋지 않은 성능을 보였다. fastText 모형에서는 계층적 소프트맥스 파라미터가 낮은 차원의 큰 윈도우에서 좋은 성능을 보이며, 높은 차원에서는 작은 윈도우에서 좋은 성능을 보이는 것을 알 수 있다. 이러한 특성은 상대적으로 다른 하이퍼파라미터로 설정된 모형이 과대 적합 내지 과소 적합으로 최적화에 이르지 못함을 의미한다.

높은 에포크는 많은 계산량에 따른 소요 시간과 컴퓨팅이 필요하지만 상대적으로 높은 성능을 가져온다. 〈그림 4〉에서 보면 대체로 100 에포크가 다수(25개)에 해당하며 나머지 50회가 3개, 20회와 5회가 각각 1개에 해당하는 것을 통해 알 수 있다. 눈여겨 볼만한 하이퍼파라미터 설정으로서 30위에 해당하는 fastText 모형을 들 수 있는데 차원이 300이며 계층적 소프트맥스를 적용하고 윈도우 크기 10, 에포크가 5회에 해당한다. 이 경우는 다른 파라미터에 비해 빠른 학습과 적은 계산량으로 비교적 준수한 성능을 얻을 수 있음을 의미한다. 모형 사이에 특정 하이퍼파라미터를 동일하게 설정하여 최종 임베딩 벡터를 만드는 필요한 소요 시간 측면에서 보면, Word2vec 모형이 fastText 모형 보다 학습의 소요 시간이 비교적 짧으며, 하이퍼파라미터 중에 네거티브 샘플링 값이 커질수록 처리에 소요되는 시간은 비례해서 오래 걸린다. 계층적 소프트맥스는 동일한 윈도우 크기를 전제로 네거티브 샘플링 5일 때 보다 비교적 짧게 소요되는데, 이는 계층적 소프트맥스 하이퍼파라미터가 시간 측면이나 성능 측면에서 유용함을 알 수 있다.

앞서 Skip-gram 구조가 더 좋은 성능을 보였으므로 이를 이용하여 Word2vec과 fastText 모형에 따른 마이크로와 매크로 평균 F1 척도의 성

능 차이를 비교하였다. 이를 위해 하이퍼파라미터 중에서 성능 편차가 심한 윈도우 크기와 에포크는 각각 10, 100 값으로 고정하였다. 또한 실험에 사용된 차원(300, 500, 768)과, 네거티브 샘플링(5, 10, 15, 20)과 계층적 소프트맥스의 5가지를 적용하여 수행된 성능 전체를 모형별로 구분하여 평균으로 계산한 결과는 〈표 1〉과 같다.

윈도우와 에포크를 고정한 두 모형의 평균 성능을 보면, 마이크로 평균 F1은 거의 유사한 것을 볼 수 있다. 매크로 평균 F1은 fastText가 상대적으로 약간 좋으며 그 이유로는 매크로 평균 재현율이 더 좋음을 알 수 있다. 이는 일반적으로 n-gram 방식을 적용하는 fastText 모형의 특성에서 기인하는 것으로 보인다. 반대로 Word2vec의 경우에는 미세하게 매크로 평균 정확률이 좋은데, 임베딩 표현이 정확한 단어 중심으로 구현되는 것에서 기인하는 것으로 보인다.

하이퍼파라미터 중에서 네거티브 샘플링과 계층적 소프트맥스에 따른 분류 성능은 〈표 2〉

〈표 1〉 모형에 따른 성능 차이(평균; Skip_gram, ws 10, ep 100 기준)

모형	차원	ns/hs	윈도우	에포크	매크로 평균			마이크로 평균
					Precision	Recall	F1	F1
TFIDF	-	-	-	-	0.6964	0.5387	0.5783	0.6441
fastText	100	5	5	5	0.6894	0.5380	0.5750	0.6512
Word2vec	100	5	5	5	0.6851	0.5280	0.5622	0.6516
fastText	전체*	전체**	10	100	0.7008	0.5667	0.5995	0.6720
Word2vec	전체*	전체**	10	100	0.7041	0.5533	0.5865	0.6718

* 차원은 300, 500, 768 크기의 3가지에 해당함.
** ns/hs는 5, 10, 15, 20 또는 계층적 소프트맥스 적용여부의 5가지에 해당함.

〈표 2〉 네거티브 샘플링과 계층적 소프트맥스의 분류 성능

모형	차원	ns/hs	윈도우	에포크	매크로 평균			마이크로 평균
					Precision	Recall	F1	F1
fastText	300	5	10	100	0.6862	0.5841	0.6145	0.6779
		10	10	100	0.6872	0.5604	0.5935	0.6665
		20	10	100	0.7009	0.5481	0.5824	0.6703
		hs	10	100	0.6941	0.5783	0.6121	0.6749
	768	5	10	100	0.6881	0.5684	0.6016	0.6669
		10	10	100	0.7038	0.5599	0.5940	0.6664
		20	10	100	0.6850	0.5519	0.5837	0.6626
		hs	10	100	0.7122	0.5679	0.6038	0.6744
Word2vec	768	5	10	100	0.7017	0.5500	0.5824	0.6707
		10	10	100	0.6975	0.5466	0.5772	0.6648
		15	10	100	0.7130	0.5187	0.5468	0.6610
		20	10	100	0.7067	0.5399	0.5714	0.6629
		hs	10	100	0.7131	0.5604	0.5971	0.6746

와 같다. 이 하이퍼파라미터의 성능 차이를 분석하기 위해 매우 다양한 값을 가지는 윈도우와 에포크는 각각 10과 100인 경우의 실험만을 대상으로 분류성능을 계산하였다.

먼저 네거티브 샘플링의 경우 하이퍼라파미터 값이 커질수록 모형이나 차원에 관계없이 매크로 평균 F1이 감소하는데, 이는 매크로 평균 재현율이 하락하는 것에서 기인한다. 네거티브 샘플링이 비교적 작은 값일수록 마이크로와 매크로 평균 F1 두 척도 모두에서 더 높은 값을 보인다. 다만 fastText 모형의 300 차원에서는 매크로 평균 정확률이 높아지면서 상대적으로 매크로 평균 F1의 값을 덜 감소시키는 경향이 있다. 한편, 계층적 소프트맥스에서는 매크로 평균 F1을 보면 주로 매크로 평균 정확률에 의해 상승하는 것에서 기인한다. 〈표 2〉에서 fastText 모형(768차원, 네거티브 샘플링 5)의 매크로 평균 F1과 재현율 값이 각각의 계층적 소프트맥스와 거의 유사하지만, 정확률은 계층적 소프트맥스가 더 높은 것을 알 수 있다. 이는 300 차원에서도 거의 유사하다.

4.3 DDC 강목에 따른 분석

DDC 300대의 10개 강목이 있지만, 실험 데이터를 구축할 당시 310대의 통계학이 5미만으로 수집되어 이를 제외하여 9개 강목으로 구성되었으며 이는 그대로 범주(label)로 간주하였다. 이들 강목에 대한 단어 임베딩에 따른 분류 성능을 파악하기 위해 〈그림 4〉에서 평균적으로 가장 좋은 성능을 보인 하이퍼파라미터(Skip-gram 구조, 차원 768, 계층적 소프트맥스 적용, 윈도우 크기 10, 에포크 100)에 대해 Word2vec 모형과 fastText 모형의 분류성능을 산출하면 〈표 3〉과 같다. 다만 이때 두 모형의 다수의 실험결과 중에서 되도록 마이크로 평균 F1 값이 거의 동일한 사례를 선정하였다.

자동분류용 실험 데이터셋 6,207건의 강목 범주에 따른 분포를 살펴보면 33X(Economics)와 34X(Law)가 각각 1,424개(이 중에서 테스트셋은 285개임)와 1,413개(테스트셋 283개)로 다수 범주에 속하며, 39X(Customs, etiquette and folklore)와 38X(Commerce, communications

〈표 3〉 강목에 따른 분류 성능(차원 768, hs, ws 10, ep 100 적용)

강목	테스트 문헌수	Word2vec			fastText		
		Precision	Recall	F1	Precision	Recall	F1
30X	194	0.4906	0.6753	0.5683	0.5297	0.6443	0.5814
32X	114	0.6905	0.5088	0.5859	0.6078	0.5439	0.5741
33X	285	0.6296	0.7754	0.6950	0.6049	0.7789	0.6810
34X	283	0.8712	0.8127	0.8410	0.8649	0.7915	0.8266
35X	68	0.6545	0.5294	0.5854	0.8049	0.4853	0.6055
36X	104	0.5783	0.4615	0.5134	0.5851	0.5288	0.5556
37X	129	0.8220	0.7519	0.7854	0.8051	0.7364	0.7692
38X	34	0.8000	0.2353	0.3636	0.8750	0.2059	0.3333
39X	31	1.0000	0.3226	0.4878	0.9412	0.5161	0.6667
micro avg	1242			0.6755			0.6755
macro avg	1242	0.7263	0.5637	0.6029	0.7354	0.5812	0.6215

and transportation)가 153개(테스트셋 31개)와 171개(테스트셋 34개)로 소수 범주에 속한다. 이렇게 데이터가 매우 불균형적이기에 다수 범주인 33X와 34X 범주는 상대적으로 F1 값이 좋으며, 더 나아가 이렇게 다수 범주가 성능이 좋기에 마이크로 평균 F1도 0.6755로 두 모형의 매크로 평균 F1에 비해 성능이 더 좋다. 또한 이용구(2020)의 연구에서 밝혔듯이 교육학의 37X 범주는 특정성이 높아 정확률이 높으며 이로 인해 F1 값이 좋은 것을 알 수 있다.

비록 두 모형의 마이크로 평균 F1 값이 0.6755로 거의 동일하지만 각 범주(강목)마다 분류성능은 다르다. 각 강목별로 두 모형을 비교하여 살펴본다면, Word2vec 모형은 32X와 39X 강목에서는 상대적으로 높은 정확률을 보이며, fastText 모형에서는 정확률은 35X와 38X 강목에서 비교적 높아 보이고 재현율과 F1은 36X와 39X에서 높다. Word2vec 모형의 경우 재현율과 F1은 범주의 크기에 비례하여 낮아지는 경향이 있으며, fastText 모형은 재현율에 있어서 비슷하게 범주 크기에 비례하지만 F1은 그렇지 않음을 알 수 있다. 다만 두 모형의 매크로 평균 F1은 약 2%정도의 차이로 다소 크지 않다. 두 모형은 매크로 평균 정확률보다 매크로 평균 재현율에서 차이가 더 많이 나며, 특히 ngram 방식의 형태를 취하는 fastText 모형이 재현율에 우호적인 것을 알 수 있다.

5. 결 론

이 연구는 도서의 서명을 활용함에 있어 짧은 길이의 제약을 넘어 다양한 영역에서 풍부한 분석이나 성능 향상을 가능하게 하는 방법으로 단어 임베딩 기법을 주목하고 이를 자동분류 실험에 적용하였다. 즉 단어 임베딩 기법을 서명에 적용하여 단행본 도서에 대한 자동분류를 수행하고 그 성능을 분석하여 임베딩의 효과를 파악하고자 하였다. 단어 임베딩 모형으로는 Word2vec, GloVe, fastText 모형을 사용하였으며 이들 모형을 학습시키기 위해 한국어 위키백과 말뭉치와 단행본 도서의 서지데이터를 활용하였다. 자동분류 실험은 학습된 3개의 모형으로부터 서명에 대한 임베딩 벡터를 분류자질로 생성하고, 이를 kNN 분류기에 적용하여 도서에 부여된 DDC 분류체계의 300대 강목을 자동 분류하였다.

Word2vec, GloVe, fastText 세 모형의 임베딩 성능이나 효과를 파악하기 위해 비교대조용 분류자질로 단어빈도와 역문헌빈도 가중치(TFIDF)를 이용하였다. 실험 데이터는 대학도서관의 단행본 신착 자료목록으로부터 사회과학분야 6,207건에 대한 서명과 DDC 분류기호를 추출하였으며 분류성능 평가는 마이크로 평균 F1을 중심으로 하였다. 단어 임베딩 모형의 다양한 하이퍼파라미터에 따른 분류 성능을 파악하여 단어 임베딩에 따른 효과를 분석하였다.

실험 결과를 정리하면, 우선 서명에 대한 단어 임베딩에 따른 kNN 분류기의 자동분류 성능을 보면, Word2vec와 fastText의 Skip-gram 모형이 비교대조용 기준인 TFIDF 가중치에 기반한 자질보다 더 우수한 성능을 보였다. 다만, Word2vec와 fastText의 CBOW 모형은 기준 자질보다 낮은 성능을 보였으며 이는 GloVe 모형도 마찬가지였다. 이때 임베딩 모형들은 하이퍼파라미터를 최적화하지 않고 해당 패키지에서 제공

하는 기본값을 적용하였다.

둘째, 짧은 텍스트인 서명에 대한 분류성능을 통한 임베딩 효과는 3개의 단어 임베딩 모형 중에 fastText의 Skip-gram 모형이 전반적으로 가장 우수했다. 이 모형은 n-gram 방식으로 하부문자열 또는 하위단어에 대한 임베딩을 생성할 수 있는데 이를 통해 재현율을 높이는 것으로 나타났다. 다음으로 Word2vec, GloVe 순이었다. 이러한 결과는 다른 연구와 비슷한 경향을 보인다.

셋째, 단어 임베딩 모형의 하이퍼파라미터를 달리 설정하면 최종 분류 성능에서 차이가 나기에 하이퍼파라미터의 최적화 실험을 수행하였으며, 그 결과 가장 우수한 분류성능을 보인 것은 fastText의 Skip-gram 모형이다. 이 모형의 하이퍼파라미터로는 계층적 소프트맥스를 사용하고 768차원과 같이 큰 차원일수록 더 우수한 성능을 보였다. 이들 하이퍼파라미터에 윈도우 크기 10 그리고 에포크 100에서 마이크로 평균 F1 0.6833으로 가장 좋은 성능을 보였다. 반면 Word2vec의 Skip-gram 모형은 대체로 300차원과 같이 낮은 차원과 3이나 5와 같은 작은 크기의 네거티브 샘플링에서 좋은 성능을 보였다.

넷째, DDC 300대의 9개 강목에 따른 분류 성능을 파악한 결과, 일반 분류기처럼 다수를 차지하는 범주와 특정성이 높은 범주에 대해 좋은 분류 성능을 보였다. 다만 강목에 따른 분류 성능에서는 Word2vec 모형과 fastText 모형이 다소 다른 모습을 보였는데, 이는 두 모형의 프레임워크 차이에서 기인하는 것으로 보인다.

단어 임베딩이나 문맥적 임베딩은, 특히 트위터 데이터와 같은 짧은 텍스트에 대해 진행된 다수의 연구가 있듯이, 이미 수년 전부터 다양한 영역에 적용되어 왔으며 좋은 성능을 보이고 있다. 서명 관련하여 단어 임베딩을 적용한 사례가 없어 이 연구에서는 임베딩 영역에서 가장 기본이며 단순한 단어 임베딩을 적용하여 분류성능이 전반적으로 다소 낮으나 도서관 영역이든 다른 영역에서 보다 다양한 메타데이터를 활용한 후속 연구가 필요하다. 예를 들면 단행본의 경우에 제목만 사용하는 것보다 제목과 목차를 함께 활용하는 것이 더 나은 성능을 기대할 수 있을 것으로 판단된다. 아울러 임베딩 성능을 높이기 위해 수십억 어절의 규모의 대규모 말뭉치를 구축하고 적용하는 연구도 필요하다. 뿐만 아니라 최근 단어 임베딩보다 좋은 성능을 가져오는 문맥적 임베딩으로 사전학습 언어모델(pre-trained language model)이나 초거대 언어모델(Large Language Model)을 파인튜닝하여 활용한다면 자동분류를 포함하여 다양한 자연언어처리 영역에서 좋은 결과를 가져올 수 있을 것으로 기대한다.

참 고 문 헌

강형석, 양장훈 (2020). Word2vec 및 fastText 임베딩 모델의 성능 비교. 디지털콘텐츠학회논문지, 21(7), 1335-1343. http://dx.doi.org/10.9728/dcs.2020.21.7.1335

이다빈, 최성필 (2019). 대용량 텍스트 자원을 활용한 한국어 형태소 임베딩의 모델별 성능 비교 분석.

정보과학회논문지, 46(5), 413-418. https://doi.org/10.5626/JOK.2019.46.5.413

이용구 (2020). 목차 정보와 kNN 분류기를 이용한 사회과학 분야 도서 자동분류에 관한 연구. 정보관리학회지, 37(1), 1-21. https://doi.org/10.3743/KOSIM.2020.37.1.001

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. https://doi.org/10.1162/tacl_a_00051

Chen, Q. & Sokolova, M. (2021). Specialists, scientists, and sentiments: Word2Vec and Doc2Vec in analysis of scientific and medical texts. SN Computer Science, 2(5), 414. https://doi.org/10.1007/s42979-021-00807-1

Devlin, J., Chang, M., Lee, K., & Toutanova, K. (2019). BERT: pre-training of deep bidirectional transformers for language understanding. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics, Volume 1, 4171-4186. https://doi.org/10.18653/v1/N19-1423

Dharma, E. M., Gaol, F. L., Warnars, H. L. H. S., & Soewito, B. (2022). The accuracy comparison among Word2vec, GloVe, and fastText towards convolution neural network (CNN) text classification. Journal of Theoretical and Applied Information Technology, 100(2), 349-359. https://doi.org/10.29207/resti.v6i3.3711

Goularas, D. & Kamis, S. (2019). Evaluation of deep learning techniques in sentiment analysis from twitter data. In 2019 International Conference on Deep Learning and Machine Learning in Emerging Applications(Deep-ML), 12-17. https://doi.org/0.1109/Deep-ML.2019.00011

Harris, Z. S. (1954). Distributional structure. Word, 10(2-3), 146-162. https://doi.org/10.1080/00437956.1954.11659520

Hinton, G. E., McClelland, J. L., & Rumelhart, D. E. (1986). Distributed Representations. In Rumelhart, D. E., McClelland, J. L., & the PDP Research Group eds. Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume I. Cambridge: Massachusetts Institute of Technology Press, 77-109.

McCann, B., Bradbury, J., Xiong, C., & Socher, R. (2017). Learned in translation: contextualized word vectors. Proceedings of the 31st International Conference on Neural Information Processing Systems, 6297-6308.

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013a). Efficient Estimation of Word Representations in Vector Space. https://doi.org/10.48550/arXiv.1301.3781

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. Advances in Neural Information Processing

Systems, 26.

Park, S., Byun, J., Baek, S., Cho, Y., & Oh, A. (2018). Subword-level word vector representations for Korean. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, 2429-2438. https://doi.org/10.18653/v1/P18-1226

Pennington, J., Socher, R., & Manning, C. D. (2014). GloVe: global vectors for word representation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, 1532-1543. https://doi.org/10.3115/v1/D14-1162

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., & Zettlemoyer, L. (2018). Deep contextualized word representations. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics, Volume 1, 2227-2237. https://doi.org/10.18653/v1/N18-1202

Salton, G., Wong, A., & Yang, C. S. (1975). A vector space model for automatic indexing. Communications of the Association for Computing Machinery, 18(11), 613-620. https://doi.org/10.1145/361219.361220

Sitender, Sangeeta, Sushma, N. S. & Sharma, S. K. (2023). Effect of GloVe, Word2Vec and fastText embedding on english and hindi neural machine translation systems. In Proceedings of Data Analytics and Management 2022, 433-447. https://doi.org/10.1007/978-981-19-7615-5_37

Wang, B., Wang, A., Chen, F., Wang, Y., & Kuo, C. C. J. (2019). Evaluating word embedding models: methods and experimental results. Asia Pacific Signal and Information Processing Association Transactions on Signal and Information Processing, 8, e19. https://doi.org/10.1017/ATSIP.2019.12

Wang, C., Nulty, P., & Lillis, D. (2020). A comparative study on word embeddings in deep learning for text classification. In Proceedings of the 4th International Conference on Natural Language Processing and Information Retrieval, 37-46. https://doi.org/10.1145/3443279.3443304

Yang, X., Macdonald, C., & Ounis, I. (2018). Using word embeddings in twitter election classification. Information Retrieval Journal, 21, 183-207. https://doi.org/10.1007/s10791-017-9319-5

• 국문 참고문헌에 대한 영문 표기
(English translation of references written in Korean)

Kang, Hyungsuc & Yang, Janghoon (2020). Performance comparison of Word2Vec and fastText

embedding models. Journal of Digital Contents Society, 21(7), 1335-1343. http://dx.doi.org/10.9728/dcs.2020.21.7.1335

Lee, Da-Bin & Choi, Sung-Pil (2019). Comparative analysis of various Korean morpheme embedding models using massive textual resources. Journal of KIISE, 46(5), 413-418. https://doi.org/10.5626/JOK.2019.46.5.413

Lee, Yong-Gu (2020). A study on book categorization in social sciences using kNN classifiers and table of contents text. Journal of the Korean Society for Information Management, 37(1), 1-21. https://doi.org/10.3743/KOSIM.2020.37.1.001