공학석사 학위논문

단어 임베딩을 활용한
텍스트 임베딩 모델 연구

2018 년 8 월

서울대학교 융합과학기술대학원

융합과학부 디지털정보융합전공

김성현

초 록

가변 길이의 텍스트를 텍스트의 맥락 정보를 반영한 벡터로 변환하는 방법인 문장 혹은 문단 임베딩은 다양한 기계학습 시스템에서 감성 분석 등의 텍스트 분류 문제, 텍스트의 유사도 측정, 클러스터링, 시각화 등 고정된 차원의 벡터를 입력으로 요구하는 각종 과제를 수행하기 위한 기본적인 특징 추출 방법으로 사용되고 있다.

기존의 텍스트의 벡터 표현을 위해 널리 사용되고 있었던 단어 자루(Bag-of-Words) 모형은 단순하고 효과적이지만 차원의 크기가 단어의 숫자에 비례해서 증가하며 레이블 없는 텍스트 데이터를 활용하기 어렵다는 단점이 있다. 이러한 한계를 극복하기 위해 제안된 문단 벡터(Paragraph Vector)는 새로운 데이터에 대한 벡터 표현을 학습하고 생성하는데 적용하기 위해서는 기존 모델을 새로운 데이터에 대해 추가적으로 학습시키는 추정 과정을 필요로 한다는 한계가 있다. 시퀀스 투 시퀀스(Sequence to Sequence) 모형을 기반으로 선후 문장과의 문장 간 관계를 활용해 문장의 벡터 표현을 생성하는 인코더를 학습하는 생각 생략 벡터(Skip-Thought Vectors) 모형은 학습을 위해 문장 간 선후 관계를 활용하기에 여러 문장이 포함되며 선후 관계를 설정하기 어려운 문단 수준에는 바로 적용되기 어려우며 텍스트 임베딩의 생성에 많은 연산 자원을 필요로 한다는 단점을 갖고 있다.

본 연구에서는 전이 학습의 문제와 임베딩 생성에 필요한 연산량 및 가변 길이 텍스트 처리의 용이성의 문제 등을 고려하여 단어 임베딩의 합을 통해 텍스트 임베딩을 생성하는 모형을 제안한다. 기존의 단어 임베딩이 유니그램(Unigram)을 사용하는 것과는 달리 본 연구의

모형에서는 바이그램(Bigram) 및 트라이그램(Trigram) 등의 n-그램(n-gram)을 활용해 텍스트 임베딩을 개선하였다. 또한 이를 컨볼루션 신경망을 사용한 모형과 비교하여 컨볼루션 신경망과 같은 추가적인 구조의 도입 없이도 좋은 성능적 특성을 보일 수 있음을 검토하였다.

본 연구는 문장 혹은 문단의 벡터 표현을 생성하기 위한 단순한 방법을 제시하는 동시에 부수적으로 학습한 n-그램 단어 임베딩의 특성을 분석하였다. 단어 임베딩의 특성에 대한 분석과 단어 임베딩의 단순한 결합이 보여주는 효과성을 통해 자연어 처리 과제에 요구되는 텍스트의 특성을 포착하기 위해 필요한 조건을 이해하는 것과 함께 텍스트의 고속 처리가 필요한 실용적인 상황에서 사용할 수 있는 도구로서 기여할 수 있을 것으로 기대한다.

주요어 : word embedding, paragraph embedding, n-gram, natural language processing

학 번 : 2016-26029

목 차

표 목차

그림 목차

제 1 장 서 론

제 1 절 연구의 배경

자연어 처리는 문서 인출(Document Retrieval), 감성 분석 등의 텍스트 분류, 텍스트 검색, 클러스터링, 시각화, 텍스트의 유사도 측정 등의 다양한 과제를 포괄하고 있으며 다양한 기계학습 알고리즘들이 적용되어 왔다. 기계학습 알고리즘 중 대다수, 즉 나이브 베이즈, SVM, K-평균, t-SNE 등의 알고리즘은 일반적으로 제한된 혹은 고정된 크기의 벡터를 입력으로 요구한다. 그러나 텍스트는 근본적으로 가변적인 길이를 가지고 있으며 따라서 이러한 알고리즘을 적용하기 위해서는 가변 길이의 텍스트 데이터를 특정한 크기의 벡터로 변환할 필요가 있다(Q. Le & Mikolov, 2014). 이러한 과제를 텍스트의 벡터 표현 혹은 텍스트의 임베딩(Embedding)이라 한다.

텍스트 임베딩은 텍스트의 길이와는 관계 없이 일정한 차원의 벡터를 생성해야 한다는 제약이 있기 때문에 신경망 번역에서와 같이 텍스트의 정보를 최대한 반영하는 것을 추구할 수는 없다. 대신 텍스트 임베딩 모형은 자연어 처리 과제에 활용될 수 있는 텍스트의 주요한 특징들을 요약적으로 포착하는 것을 목표로 한다.

텍스트의 벡터 표현 방법으로서 널리 사용되는 단어 자루(Bag-of-Words) 모형은 단순하면서도 효과적인 방법이다(Harris, 1954). 단어 자루 모형은 개별 단어를 하나의 차원으로 놓고 텍스트 내의 단어의 출현 횟수를 각 차원의 값으로 부여하는 방법이다. 단어 자루 모형은 그 단순성에 비해 많은 텍스트 처리 과제에서 뛰어난 성능을 보여주고

있다(Maas et al., 2011; Wang & Manning, 2012).

그러나 단어 자루 모형 혹은 tf-idf 등의 벡터 공간 모형은 기본적으로 단어의 순서 정보를 배제한다는 한계를 가지고 있다. 즉 벡터 공간 모형에서는 동일한 단어가 동일한 횟수로 문서에 등장했다면 두 문서는 동일하게 취급된다. 이는 벡터 공간 모형으로 변환된 벡터는 단어들의 조성과 구성을 통해 형성되는 의미를 포착하기 어렵다는 것을 의미한다. 이러한 한계는 n-그램(n-gram)의 사용을 통해 완화될 수 있지만, 단어 자루 모형으로 변환된 벡터의 차원은 사용되는 어휘의 수와 동일하다는 특성 때문에 n-그램을 활용하면 데이터 행렬이 크게 희소(Sparse)해지고 따라서 기계학습 알고리즘 등을 적용했을 때 차원의 저주(Curse of Dimensionality)에 빠질 위험성이 높아진다는 문제가 있다. 또한 단어 자루 모형은 모든 단어를 동등하게 취급하며(Q. Le & Mikolov, 2014), 단어의 빈도와 문서간 단어의 등장 빈도를 활용해 단어에 가중치를 부여하는 tf-idf 모형에서도 단어의 상대적 중요성을 가중치로 반영하기는 하지만 여전히 단어의 의미적 유사성 등의 정보는 소실된다는 한계가 있다. 따라서 단어 자루 모형으로는 대규모의 레이블 없는 텍스트 데이터에서 단어의 유사성 혹은 텍스트의 조성 등의 특징을 추출해 다른 자연어 처리 과제를 수행하는 것과 같은 전이 학습(Transfer Learning)에 사용되기 어렵다.

따라서 차원의 크기와 전이 학습의 용이성 등과 같은 단어 자루 모형의 한계를 극복할 수 있는 텍스트 임베딩(Text Embedding) 모형의 개발은 자연어 처리와 기계학습의 오랜 연구 주제 중 하나였다(Kiros et al., 2015). 특히 희소(Sparse)하지 않고 밀집(Dense)한 벡터 표현으로 생성되는 임베딩 벡터의 차원의 크기를 줄이는 동시에 텍스트 내에서 단어의 배치가 형성하는 구성 및 조성적 의미를 포착하는 벡터 표현으로

변환하는 방법 및 알고리즘의 개발은 꾸준히 관심을 받아온 문제였다. 조성적 의미란 단어의 특정한 배치로 인해 발생하는 단순한 단어의 출현 여부만으로는 파악하기 어려운 종류의 의미를 의미한다.

1. 이 작품이 그렇게 좋지 않은 작품이라고 보기는 어렵다. 2. 시간낭비를 즐긴다면 이 작품을 감상하는 것이 즐거울 것이다.

표 1 조성적 의미가 문장의 의미 파악에 중요한 영향을 미치는 예. 1 번 문장에서는 '좋지 않은'이라는 표현이 '보기는 어렵다'라는 표현과 결합되어 부정적이지 않게 평가한다는 의미를 획득하게 되며, 2 번 문장에서는 '즐거울 것이다'라는 표현이 '시간낭비를 즐긴다면'이라는 표현과 결합되어 부정적으로 평가한다는 의미를 갖게 된다.

임베딩의 핵심이 텍스트와 같은 비정형 데이터의 벡터화라는 것을 고려하였을 때 비정형 데이터의 분산 표현(Distributed representation), 즉 벡터화된 표현의 학습에 강점을 보여주고 있는 신경망(Neural Network)은 텍스트 임베딩의 학습에 적합한 접근 방식이라 할 수 있다(Bengio, Courville, & Vincent, 2013; Hinton, 1986), 순환 신경망(Dai & Le, 2015). 데이터의 분산 표현(Distributed Representation)이란 데이터를 데이터가 가지는 여러 특징들로 분석하여 표현하는 것을 의미한다. 이는 임베딩 문제에 적용하였을 때 텍스트를 단어의 출현 빈도 등으로 표현하는 것이 아니라 텍스트의 특징들, 즉 여러 문법적 혹은 의미적 특징들을 기술하는 방식으로 임베딩을 표현할 수 있음을 의미한다. 회귀 신경망(Socher, Perelygin, & Wu, 2013), 컨볼루션 신경망(Kalchbrenner, Grefenstette, & Blunsom, 2014), 혹은 회귀-컨볼루션 신경망(Cho, van Merrienboer, Bahdanau, & Bengio, 2014) 등 다양한 형태의 신경망 모형들이

텍스트의 벡터 표현으로의 학습을 위해 시도되었다.

그러나 이러한 방법들은 일반적으로 레이블이 있는 텍스트 데이터, 예를 들어 영화 평가 텍스트 말뭉치 데이터의 경우에는 영화에 대한 평점 등의 레이블이 텍스트와 쌍으로 존재하는 데이터에 대해 적용되어왔다. 즉 평점을 분류 문제의 대상으로 삼아 벡터 표현의 생성 방법을 학습하는 방식의 접근이 일반적이었다. 따라서 이러한 방법들은 레이블이 없는 데이터에는 적용될 수 없으며 레이블이 있는 데이터에 비해 상대적으로 많은 레이블이 없는 데이터를 벡터 표현의 생성 방법을 학습하는 것에 사용할 수 없다는 한계를 가지고 있다. 또한 학습 과정에서 벡터 표현을 레이블을 예측하는 과제에 최적화되도록 학습하게 되므로 학습된 벡터 표현이 영화 평점 분류 등 레이블을 예측하는데 적절한 특성을 위주로 반영하도록 특화된다는 문제를 가지고 있다. 즉 다른 종류의 과제에 대해 전이(Transfer)하기 위해 사용하기에는 어려울 수 있다.

따라서 효과적인 텍스트의 벡터 표현 혹은 임베딩의 생성을 위해서는 텍스트 내의 단어들의 구성적 의미 정보를 반영할 수 있는 동시에 보다 다양한 분야에 적용될 수 있도록 일반화된 벡터 표현을 생성할 수 있는 모형이 필요하다고 할 수 있다. 이러한 요구 조건을 충족하기 위해서는 모형이 단어의 순서 정보를 활용할 수 있도록 하는 구조와 표현력을 가지고 있어야 하며, 학습 과정에서 레이블 되지 않은, 혹은 그 레이블이 질적으로 다른 종류의 데이터도 사용할 수 있도록 설계되어야 한다고 할 수 있다(Kiros et al., 2015).

제 2 절 연구의 목표

본 연구는 가변 길이의 텍스트, 즉 문장 혹은 문단을 입력으로

하여 단어의 구성적, 조성적 의미를 반영하여 벡터 표현으로 변환할 수 있는 모형을 제안하고자 하는 것을 목표로 한다. 추가적으로 여러 문제에 유용하게 활용 가능하도록 새로운 데이터에 대해 텍스트의 벡터 표현, 즉 텍스트 임베딩을 생성하기 위한 추가적인 학습 과정이 필요하지 않으며, 임베딩의 생성 과정에 많은 연산력(Computational power)이 필요하지 않고, 명시적인 레이블이 없거나 잘 정의된 문단 구조가 존재하지 않는 등의 다양한 형태의 말뭉치(Corpus)에 적용될 수 있으며, 전이 학습 과제에서 좋은 성능을 보일 수 있는 모형의 개발을 목표로 한다. 이를 위해 단어 임베딩의 합을 가변 길이 텍스트 입력 데이터를 고정된 크기의 벡터로 변환하는 인코더로 사용하고, 텍스트의 단어 자루 표현을 예측하는 다중 레이블(Multi Label) 분류기 또는 순환 신경망 언어 모형을 디코더로 사용하는 모형을 제안하고자 한다(Cho, Van Merriënboer, et al., 2014; Sutskever, Vinyals, & Le, 2014). 텍스트의 인코딩을 위해 적합한 단어 임베딩의 활용 방식을 제안하고, 학습한 모형을 통해 전이 학습 과제를 실시하였을 때 나타나는 특성을 통해 자연어 처리에서의 전이 학습에서 고려할 수 있는 사항들에 대해 분석한다. 또한 텍스트 임베딩 모형의 학습 과정에서 부차적으로 생성되는 n-그램 단어 임베딩의 특성에 대해 분석하고 컨볼루션 신경망(Convolutional Neural Network)과 재귀 신경망(Recurrent Neural Network)을 활용한 언어 모델을 통해 학습된 텍스트 임베딩과 비교하여 각종 자연어 처리 과제에서 유용한 방식으로 텍스트에서 정보를 추출하기 위해 필요한 조건들에 대해 분석한다.

제 2 장 선행 연구

제 1 절 문장 및 문단 임베딩

비지도적(Unsupervised), 즉 레이블 없이 텍스트 임베딩을 위한 모형의 일반적인 접근은 언어 모형(Language Model)(Bengio, Ducharme, Vincent, & Jauvin, 2003)의 사용이라고 할 수 있다. 언어 모형은 텍스트에 확률을 부여하는 모형이다. 텍스트를 단어 등의 토큰의 나열 혹은 시퀀스라고 하면 텍스트 시퀀스 내의 토큰들 $w_1, \cdots, w_{N-1}, w_N$에 대해 확률 $p(w_1, w_2 \cdots, w_{N-1}, w_N)$를 생각할 수 있다. 언어 모형에서는 확률의 연쇄 법칙을 통해 이러한 결합 확률 분포를 다음과 같이 분해한다.

$$p(w_1, w_2 \cdots, w_{N-1}, w_N) = p(w_1)p(w_2|w_1) \cdots p(w_N|w_1, w_2 \ldots, w_{N-1})$$

따라서 텍스트 전체에 대한 결합 확률 분포 $p(w_1, w_2 \cdots, w_{N-1}, w_N)$는 이전 단어들에 대한 다음 단어의 조건부 확률 $p(w_n|w_1, w_2 \ldots, w_{n-1})$들의 곱으로 표현할 수 있다. 즉 이전 단어들을 통해 다음 단어를 예측하는 모형을 통해 텍스트 전체를 예측하는 생성 모형(Generative model)을 구성할 수 있다.

여기에서 텍스트의 결합 확률 분포에 영향을 미치는 맥락적 정보를 추가적으로 고려할 수 있다. 예를 들어 긍정적인 정서를 반영하고 있는 텍스트에서는 긍정적인 정서라는 맥락에서 등장하는 단어들의 확률이 증가할 것이다. 즉 텍스트의 정서적 맥락을 지시하는 변수 c가 있다고 하면 c가 긍정적인 정서를 가리킬 때는 '좋은'과 같은 단어가 텍스트에 등장할 확률이 높아질 것이며 부정적인 정서를 가리킬 때는 '지루한'과

같은 단어가 텍스트에 등장할 확률이 높아질 것이다. 따라서 정서, 주관성 등 텍스트의 의미와 관계된 특성들을 단어들의 확률을 변화시키는 잠재 변수(Latent variable)로 간주할 수 있다. 따라서 단어 모형을 이러한 맥락적 정보를 담은 잠재 변수 c에 대한 조건부 확률로 확장할 수 있다(Arora, Liang, & Ma, 2017).

$$p(w_1, w_2 \cdots, w_{N-1}, w_N|c) = p(w_1|c)p(w_2|w_1, c) \cdots p(w_N|w_1, w_2 \ldots, w_{N-1}, c)$$

위의 조건부 확률을 통해 맥락 변수 c 에 대한 최대우도추정량(Maximum likelihood estimate)을 추정할 수 있다.

$$\arg\max_c p(w_1, w_2 \cdots, w_{N-1}, w_N|c) = \arg\max_c p(w_1|c)p(w_2|w_1, c) \cdots p(w_N|w_1, w_2 \ldots, w_{N-1}, c)$$

따라서 각 텍스트에 대해 단어들의 확률을 최대화하는 c를 학습하는 것으로 맥락을 반영하는 잠재 변수 c를 추정할 수 있으며 이렇게 학습된 변수 c 를 해당 텍스트의 의미 정보를 반영하는 텍스트 임베딩으로 생각할 수 있다.

이러한 관점에서 문장 및 문단 임베딩에 대한 대표적인 접근으로는 Word2vec, 문단 벡터(Paragraph Vector), 생각생략 벡터(Skip-Thought Vector) 등이 있다.

1.1 Word2vec

자연어 처리에서 가장 성공적이었던 분산 표현의 학습 모형은 단어에 대한 분산 표현 모형이었다. 단어의 분산 표현 학습 모형으로 가장 대표적인 모형이 Word2vec의 생략그램(Skip-gram) 모형이다(Mikolov, Sutskever, Chen, Corrado, & Dean, 2013).

생략그램 모형은 대상 단어를 통해 단어 주위에 등장한 단어들을 예측하는 방식으로 단어의 분산 표현을 학습한다. 즉 생략 그램 모형에서 단어 벡터 w_i 는 해당하는 단어 주위의 단어 $w_{i-k},\cdots,w_{i-1},w_{i+1},\cdots,w_{i+k}$의 확률 $p(w_{i-k},\cdots,w_{i-1},w_{i+1},\cdots,w_{i+k}|w_i)$를 최대화하는 방식으로 학습된다. 이러한 접근은 언어 모형을 이전 단어들 전체에 대해 다음 단어를 예측하는 대신 한 단어의 이전 단어와 다음 단어를 예측하도록 수정한 것이라고 생각할 수 있다.

흥미로운 것은 단어 시퀀스를 시퀀스 내에 있는 단어들에 해당하는 벡터의 합으로 표현할 수 있다는 것이다. 즉 단어 벡터를 열벡터로 갖는 단어 임베딩 벡터의 행렬 $\mathrm{W} \in \mathbb{R}^{V \times D}$(V는 단어의 수, D는 단어 벡터의 차원)에 대해 단어 시퀀스의 단어 자루 표현, 즉 $x_i =$ (단어 시퀀스에서 단어 i가 등장한 횟수)인 벡터 $\mathrm{x} \in \mathbb{R}^V$에 대해 $\mathrm{y} = \mathrm{Wx}$를 생각할 수 있다. 이는 단어 자루 벡터의 단어 임베딩 행렬에 의한 선형 변환이며 $\mathrm{y} \in \mathbb{R}^V$를 단어 시퀀스에 대한 벡터 표현으로 생각할 수 있다.

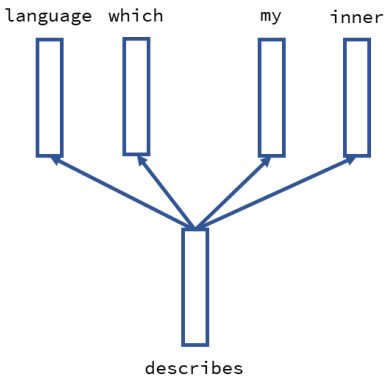

그림 1 생략그램 모형의 구조. 생략그램 모형은 대상 단어를 통해 대상 단어 주위에 공출현(Cooccurrence)한 단어들을 예측하는 방식으로 단어의 벡터 표현을 학습한다.

이러한 방식의 텍스트 임베딩은 매우 단순하지만 많은 자연어 처리 과제에서 효과적인 것으로 밝혀졌다. 그러나 단어에 대해 학습된 벡터 표현을 텍스트에 적용하였을 때, 단어 임베딩 벡터를 단순히 더하는 것만으로도 텍스트의 특성을 반영하는 벡터를 구성할 수 있는 이유는 아직 잘 밝혀지지 않았다. 한 가지 설명은 단어 임베딩의 학습 과정에서 입력 단어에 대해 입력 단어의 다음에 출현하는 단어의 확률을 예측하는 과정에서 단어 벡터가 텍스트의 맥락적 정보를 반영하도록 학습된다는 것이다. 즉 단어 w_{N-1}에 대한 단어의 벡터 표현을 v(w_{N-1})라고 한다면 단어 벡터는 학습 과정 중에서 $p(w_N|w_{N-1}) = p(w_N|\mathrm{v}(w_{N-1}))$ 를 최대화하도록 학습되는데, 이 과정에서 v(w_{N-1})에 w_N을 예측하는데 유용한 맥락적 잠재 변수인 c가 반영된다는 것이다(Arora, Li, Liang, Ma, & Risteski, 2016). 따라서 v(w)의 합 혹은 평균은 각 단어들을 예측하는 맥락 변수 c의 합 혹은 평균으로 이해될 수 있으며, 따라서 텍스트의 총체적인 혹은 평균적인 맥락적 정보를 반영하는 벡터라고 볼 수 있다는 것이다.

1.2 **문단 벡터**(Paragraph Vector)

문단 벡터(Q. Le & Mikolov, 2014)는 문장 혹은 문단 임베딩을 위한 대표적인 방법 중 하나이다. 문단 벡터는 단어 벡터(Word2vec)의 문장에 대한 확장이라고 할 수 있다. 문단 벡터에서는 각 단어 뿐만 아니라 단어가 포함된 각각의 문장 혹은 문단 또한 특정한 벡터로 사상된다. 즉 텍스트의 임베딩을 하나의 추가적인 단어로 간주한다고 생각할 수 있다. 문단 벡터는 j번째 문단에 대한 벡터 d_j와 문단 내 단어들의 시퀀스 $w_1, \cdots, w_{N-1}, w_N$, 그리고 맥락의 범위 k에 대해 $p(w_N|w_{N-k}, \cdots, w_{N-1}, d_k)$ 를 최대화하는 방식으로 학습된다. 여기서 j번째 문단 내의 모든 단어 $w_1, \cdots, w_{N-1}, w_N$에 대해 공통된 조건 혹은

맥락으로 작용하기에, d_j 는 단순히 특정 단어 이전 k 개의 단어가 제공하지 못하는 맥락적 정보를 제공할 수 있으며, 따라서 d_j가 문단의 전반적인 의미 혹은 주제에 대한 정보를 반영하게 된다고 가정할 수 있다.

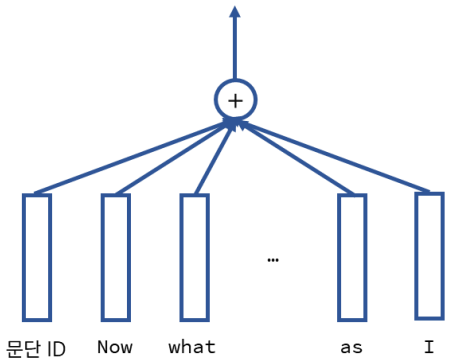

그림 2 문단 벡터 모형의 구조. 문단의 ID 를 임베딩한 벡터와 이전 단어들을 결합한 벡터를 통해 다음 단어를 예측하게 하는 방식으로 문단 ID 에 해당하는 문단의 벡터 표현을 학습한다.

문단 벡터는 문단에 대한 임베딩을 추가적인 단어로 간주하기에 데이터 내의 문단의 수만큼의 텍스트 임베딩이 학습 과정에서 생성되며 고정된다. 따라서 문단의 수만큼의 텍스트 임베딩 벡터를 저장해야 하며 또한 새로운 입력 데이터에 대한 임베딩을 곧바로 생성할 수 없고 추가적인 학습 과정을 필요로 한다. 이는 단어 벡터 $\mathbf{w}$와 가중치 $\mathbf{u}$, 편향 b를 고정한 다음 새로운 데이터와 새로운 문단 벡터 $\mathbf{d}_j$를 경사하강법(Gradient Descent) 등으로 최적화하는 방식으로 이루어진다(Q. Le & Mikolov, 2014). 이러한 추가적인 학습, 즉 최적화 과정을 피하기 위해서는 문단 임베딩의 생성 과정에서 단순히 개별 문단을 고유한 벡터로 사상하기보다는 문단을 입력으로 받아 벡터 표현을 출력하는 인코더 혹은 리더(Reader)의 구조를 가진 모형이 요구된다고 할 수

있다.

1.3 생각생략 벡터(Skip-Thought Vectors)

생각생략 벡터(Skip-Thought Vectors)(Kiros et al., 2015)는 테스트 시점에서 추가적인 추정 과정을 필요로 한다는 문단 벡터의 문제점을 극복하고 문장에 대한 일반화된 벡터 표현을 학습하고 생성하기 위해 제안된 모형이다.

생각생략 벡터는 단어 벡터의 생략 그램 모형의 문장에 대한 적용이라고 할 수 있다. 생략 그램 모형에서 단어의 벡터 표현이 단어의 주위에 등장한 단어를 예측하도록, 즉 등장 단어의 확률을 최대화하도록 학습하는 것과 같이 생각생략 벡터는 생략그램 모형을 문장에 적용해 문장 s_i에 대해 이전 문장 s_{i-1}, s_{i+1}의 확률 $p(s_{i-1}, s_{i+1}|s_i)$를 최대화하는 방식으로 학습된다(Kiros et al., 2015).

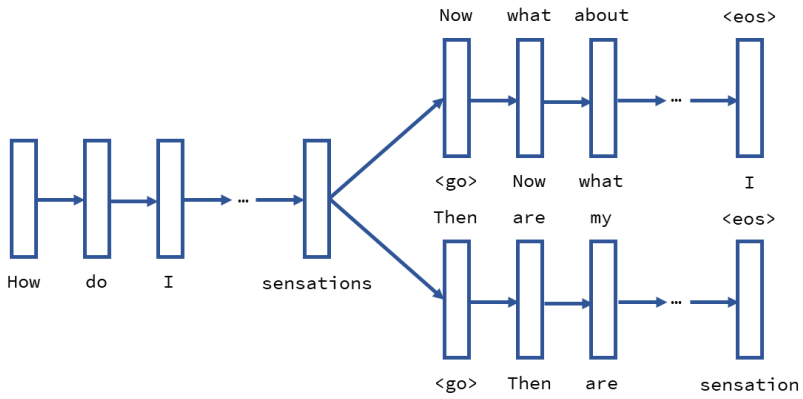

그림 3 생각생략 벡터 모형의 구조. 문장을 인코더로 인코딩한 다음 디코더에 입력해 이전 문장과 다음 문장을 디코딩하는 방식으로 문장의 벡터 표현을 학습한다.

생각생략 벡터는 이를 위해 인코더-디코더 혹은 시퀀스 투

시퀀스(Sequence-to-Sequence) 형태의 모형을 사용하였다. 인코더-디코더 혹은 시퀀스 투 시퀀스 모형이란 특히 서로 길이가 다른 시퀀스 $x_1, \cdots, x_{N_x}$와 $y, \cdots, y_{N_y}$에 대해 $p(y, \cdots, y_{N_y} | x_1, \cdots, x_{N_x})$를 학습하기 위한 모형이다(Cho, Van Merriënboer, et al., 2014; Sutskever et al., 2014). 이를 위해 시퀀스 투 시퀀스 모형은 인코더 함수 f로 $x_1, \cdots, x_{N_x}$를 특정한 벡터 c로 변환한 다음, 디코더 함수 g를 통해 $p(y, \cdots, y_{N_y} | c)$를 학습하는 방식으로 구현된다. 인코더 혹은 디코더로는 흔히 순환 신경망(Recurrent Neural Network, RNN)이 사용되며 훈련 과정에서 학습되게 된다.

생각생략 벡터는 시퀀스 투 시퀀스 모형의 구조를 활용하여 문장 s_i를 c로 인코드한 다음 $p(s_{i-1})p(s_{i+1})$를 최대화하는 방식으로 모형을 학습시켰다. 학습을 마친 다음에는 $c = f(w_1, w_2, \cdots, w_{N-1}, w_N)$를 문장 s_i에 대한 벡터 표현으로 사용하게 된다.

이는 s_i를 c로 인코드한 다음 $p(s_i)$를 최대화하는 식의 시퀀스 오토인코더(Sequence Autoencoder)(Dai & Le, 2015)에 비해서 보다 어려운 과제라고 할 수 있다. 오토인코더에 비해 생각생략 벡터와 같은 구조가 가지는 이점은 단순히 입력 문장을 재현하는 것이 아니라 주위 문장을 추정하게 함으로써 문장이 맥락 속에서 가지는 의미를 반영하는 방식으로 학습되게 되고, 따라서 보다 일반화된 의미 정보를 반영하는 벡터 표현을 생성할 수 있게 된다는 것이다. 보다 구체적으로 살펴보자면 다음과 같다. 우선 입력 문장 s_i에 대해 s_i 내의 단어를 w_k^i라고 하고

$$c = f(w_1^i, w_2^i \ldots, w_{N_i-1}^i, w_{N_i}^i)$$

라고 하면, 오토인코더는 $p\left(w_1^i, w_2^i \ldots, w_{N_i-1}^i, w_{N_i}^i | c\right)$를 학습하게 되며,

이는 확률의 연쇄법칙을 통해 다음과 같이 분해될 수 있다.

$$p(w_1^i, w_2^i \dots, w_{N_i-1}^i, w_{N_i}^i|c) = p(w_1^i)p(w_2^i|w_1^i, c) \cdots p(w_{N_i}^i|w_1^i, w_2^i \dots, w_{N_i-1}^i, c)$$

따라서 오토인코더는 이전 단어들과 c에 대해 다음 단어의 확률을 최대화하도록 학습하게 된다. 그러므로 오토인코더의 인코더 f는 이전 단어들에 대해 입력 문장 내의 단어 $w_1^i, w_2^i \dots, w_{N_i-1}^i, w_{N_i}^i$의 확률을 증가시키는 맥락 정보 혹은 벡터를 제공하도록 학습하게 된다.

반면 생각생략 벡터와 같은 구조에서는 $p(w_1^{i-1}, w_2^{i-1} \dots, w_{N_i-1}^{i-1}, w_{N_i}^{i-1}|c)p(w_1^{i+1}, w_2^{i+1} \dots, w_{N_i-1}^{i+1}, w_{N_i}^{i+1}|c)$를 최대화하도록 학습하게 된다. 따라서 생각생략 벡터의 인코더 f는 단순히 이전 단어들에 대해 입력 문장 내의 단어의 확률을 높이는 것이 아니라 해당 입력 문장의 선후에 있는 문장의 단어의 확률을 높이는 맥락을 제공하도록 학습하게 된다. 따라서 단순히 입력 문장 내에 있는 단어들이 아니라 의미적 맥락 속에서 등장할 수 있는 단어들에 대한 정보를 제공하게 되므로 생각생략 벡터 모형은 문장을 보다 일반적인 의미를 반영하는 벡터 표현으로 변환할 수 있게 된다. 이는 문장의 의미적 정보를 반영하고 추출하는 것에 있어 문장간(Intersentential) 관계를 반영하는 것이 도움이 될 수 있다는 것을 시사한다.

문장간의 관계를 이용한다는 점은 생각생략 벡터의 강점인 동시에 문단에 대해서 동일한 구조가 적용될 수 없는 이유이기도 하다. 문단 내의 문장들은 하나의 주제 혹은 요점을 공유하고 있다고 간주할 수 있으나(*Paragraph*, 2017) 하나의 글 속에 있는 문단들이라고 하여도 전반적인 주제 이상으로 문단 내의 문장이 공유하는 것과 같은 보다 밀접한 의미적 관계를 가지리라 기대하기는 어렵다. 또한 생각생략

벡터와 같은 구조를 문단에 그대로 확장하여 문단간의 관계를 반영하는 모형을 구성하려면 데이터가 문단으로 구분되어 있으며 동시에 여러 문단이 존재해야 하므로 사용할 수 있는 데이터가 보다 제한된다는 문제가 있을 수 있다.

제 2 절 텍스트 처리를 위한 신경망 구조

텍스트 내의 각 단어들이 입력으로 주어졌을 때 단어들 사이의 관계를 반영하기 위해 추가적인 신경망을 사용할 수 있다. 텍스트와 같은 가변 길이 데이터에 대해 가장 널리 사용되어온 신경망은 순환 신경망이라고 할 수 있다(Graves, 2012; Rumelhart, Hinton, & Williams, 1986). 그러나 순환 신경망은 기본적으로 순차적인 구조를 가지고 있으며 따라서 병렬화에 난점이 있으며, 입력 시퀀스 내의 장기적인 관계를 반영하는 것에 문제가 있다(Lin et al., 2017; Oord et al., 2016). 따라서 고속 처리가 가능하면서도 입력 시퀀스 내의 관계를 반영할 수 있는 신경망, 혹은 이러한 관계를 반영하는 것에 도움을 줄 수 있는 구조가 필요하다고 할 수 있다. 이러한 문제에는 컨볼루션 신경망을 사용하는 방법, 그리고 주의(Attention) 메커니즘을 활용하는 방법 등이 제안되어 왔다.

2.1 **문장 분류를 위한 컨볼루션 신경망**(Convolutional Neural Networks for Sentence Classification)

문장 분류를 위한 컨볼루션 신경망(Convolutional Neural Networks for Sentence Classification)(Kim, 2014)에서는 컨볼루션 신경망을 텍스트에 대해 적용하는 접근 방법을 제안하였다. 매개변수 공유(Parameter Sharing)라는 컨볼루션 신경망의 특징으로 인해 컨볼루션은 가변 길이 텍스트에 대해서도 적용될 수 있다(Kim, 2014).

우선 텍스트 데이터는 텍스트를 이루는 N개 단어들의 d차원 문장 벡터 $\mathbf{x}_i$를 연결한 $N \times d$ 차원 행렬 $\mathbf{X}$로 나타내어진다. 이 행렬에 대해 $1 \times k$ 컨볼루션을 수행한다. 가변 길이 문장 입력에 대해 출력된 가변 길이 출력을 고정된 크기의 벡터로 변환하기 위해 컨볼루션을 수행한 결과 행렬 $\mathbf{C}$에 대해 문장의 길이 방향으로 최대값을 추출하는 최대값 풀링(Max Pooling)을 수행한다. 추출된 최대값을 완전 연결 신경망(Fully-connected Network)에 통과시킨 다음 소프트맥스(Softmax) 출력을 통해 분류를 수행하고 역전파(Backpropagation)로 학습하게 된다.

이 모형은 단층의 컨볼루션 신경망을 인접한 k개 단어에서 특징을 추출하기 위한 방법으로 사용하였다. 따라서 컨볼루션 필터가 추출하는 특징은 k개 단어에서 추출될 수 있는 특징으로 국한된다. 이러한 문제를 극복하기 위해 에서는 다양한 k, 즉 다양한 크기(3, 4, 5)의 컨볼루션 필터를 사용해 조합함으로써 이러한 한계를 극복하고자 했다(Kim, 2014).

그러나 보다 장기적인 관계에서 나타나는 패턴을 포착하기 위해 컨볼루션 필터의 크기를 계속해서 확장하는 것은 학습해야 할 매개변수의 크기를 크게 늘리게 된다. 보다 장기적 관계를 반영하기 위해서 필터의 크기를 확장하는 대신 복층의 신경망을 사용하는 방법을 사용할 수 있다.

2.2 **구조적 자기 주의 문장 임베딩**(A Structured Self-Attentive Sentence Embedding)

가변적인 길이를 갖는 텍스트 입력을 고정된 크기의 벡터 형태로 전환하는 것은 텍스트 분류 등의 문제에 적용하기 위한 모형에서도

필수적인 부분이다. 일반적으로 문장 분류를 위한 컨볼루션 신경망에서와 같이 전체적인 최대값 풀링 혹은 평균, 또는 순환 신경망의 경우에는 순환 신경망의 마지막 상태를 텍스트 입력에 대한 벡터 표현으로 사용하는 방식 등이 사용되어 왔다.

구조적 자기 주의 문장 임베딩(A Structured Self-Attentive Sentence Embedding)(Lin et al., 2017)에서는 가변 길이 텍스트 시퀀스를 고정된 크기의 벡터로 전환하기 위해 주의(Attention) 메커니즘 사용했다. 주의 메커니즘은 시퀀스 투 시퀀스 모형이 적용된 신경망 번역과 같이 인코더와 디코더에 서로 다른 문장이 입력되고 디코더에서 인코더가 인코딩한 문장의 벡터 표현에서 적절한 벡터를 추출하기 위한 장치로써 사용되어왔다(Dzmitry Bahdana, Bahdanau, Cho, & Bengio, 2015).

그러나 텍스트 분류 등의 과제에서는 입력 문장에 대해 적절한 주의 가중치(Attention weight)를 지정하기 위해 사용될 수 있는 대응 문장이 존재하지 않는다. 따라서 구조적 자기 주의 문장 임베딩에서는 짝을 이루는 문장을 사용하는 대신 입력 문장 자체를 사용해 주의 가중치를 계산하는 자기 주의(Self-attention) 메커니즘을 사용했다.

입력 텍스트 시퀀스를 $w_1, w_2, \cdots, w_{N-1}, w_N$ 라고 하고, 시퀀스 내의 각 토큰에 대한 벡터 표현의 행렬을 $\mathrm{H} = (h_1, h_2, \cdots, h_{N-1}, h_N) \in \mathbb{R}^{n \times u}$ 라고 하면, 입력 텍스트 시퀀스에 대한 자기 주의 가중치 a 는 다음과 같이 계산할 수 있다(Lin et al., 2017).

$$a = \mathrm{softmax}(v \tanh(WH^T))$$

여기에서 v 는 $v \in \mathbb{R}^k$ 인 벡터이며 $W \in \mathbb{R}^{k \times u}$ 인 행렬이다. 따라서 $a = (a_1, a_2, \cdots, a_{N-1}, a_N) \in \mathbb{R}^n$ 이며 $\sum a_i = 1$ 인 벡터가 된다. 이후 계산된

주의 가중치 a 를 사용해 텍스트 입력의 벡터 표현을 다음과 같이 계산할 수 있다.

$$V = aH$$

따라서 V 는 주의 가중치를 사용해 계산된 토큰 벡터들의 가중 평균이라고 할 수 있다. 자기 주의 메커니즘은 문장 분류 등의 과제뿐만 아니라 신경망 번역 모형 등에 성공적으로 활용되어 텍스트 인코딩에 효과적인 방법이라는 것이 알려졌다.

제 3 장 연구 방법 및 설계

제 1 절 시스템 구조

2장에서의 선행 연구 검토를 통해, 유용한 문단 임베딩의 생성을 위해서는 문단 임베딩이 문단 내의 국소적인 특징이 아닌 문단 전체의 맥락적 정보를 반영하여야 하며, 문단을 입력으로 받아 벡터로 변환하는 인코더의 형태를 취할 필요가 있고, 또한 문장간의 관계를 모형에 반영하는 것이 보다 일반화된 벡터 표현의 생성에 도움이 될 수 있다는 점을 검토했다.

또한 문단과 같은 가변 길이의, 그리고 길이가 긴 시퀀스에 대해서는 컨볼루션 신경망을 사용한 인코더가 유용할 수 있음을 살펴보았다. 이와 같은 조건들을 반영하여 본 연구에서는 문단 임베딩을 위한 다음과 같은 모형을 제안한다.

본 연구에서 제안하는 모형은 텍스트에 대한 인코더-디코더 구조를 따르고 있다. 텍스트 입력을 벡터 표현으로 변환하기 위한 인코더로는 다음과 같은 세 가지 형태의 모형을 실험하였다.

1. 단어 임베딩의 평균. 단어 임베딩의 합이 자연어 처리 과제에서 효과적이라는 것이 알려져 있으며(Arora et al., 2017; Conneau, Kiela, Schwenk, Barrault, & Bordes, 2017; Hill, Cho, & Korhonen, 2016), 따라서 단어 입력을 임베딩으로 변환하고 임베딩의 합을 취하는 모형을 단어 시퀀스에 대한 인코더로 사용할 수 있다. 여기서 단어 임베딩의 단순한 합을 사용하거나 혹은 추가적으로 단어 벡터의 합을 텍스트 시퀀스의 길이로

나눠 평균을 사용할 수 있다. 또한 텍스트 임베딩에 단어의 조성을 통해 형성되는 의미 정보를 반영하기 위해 바이그램과 트라이그램 등 n-그램 토큰들을 유니그램에 추가할 수 있다.

2. 컨볼루션 신경망. 컨볼루션 신경망은 감성 분류 등의 문제에 효과적이라는 것이 알려져 있으며(Kim, 2014) 따라서 텍스트의 의미를 반영하는 특징(Feature)들을 추출할 수 있는 표현력을 갖고 있다고 추정할 수 있다. 컨볼루션 신경망을 인코더로 사용하는 경우 또한 토큰 축에 대한 전역 평균 풀링(Global average pooling)을 사용하거나 전역 최대값 풀링(Global max pooling)을 사용할 수 있다.

3. 컨볼루션 신경망과 자기 주의(Self-attention). 컨볼루션 신경망에 자기 주의 메커니즘을 활용하여 주의 가중치를 부여할 수 있다(Lin et al., 2017). 이는 전역 평균 풀링에서 각 토큰에 대해 다른 가중치를 부여한 가중 평균으로 생각할 수 있다. 자기 주의 메커니즘을 적용하기 전에 최대값 풀링 등으로 입력 시퀀스의 길이를 줄여 필요한 연산의 양을 감소시키고 실질적으로 입력 시퀀스에 대한 수용장(Receptive field)의 크기를 증대시키는 방법을 적용할 수 있다. 또한 각 입력 토큰 벡터 h_i에 대해 가중치 스칼라 a_i를 적용하여 $v = \sum a_i h_i$를 구하는 대신 $\sum_i a_{ij} = 1$인 벡터 가중치 $a_{ij} = (a_{i1}, a_{i2}, \dots, a_{id})$를 적용하여 $v = \sum a_i \circ h_i$ ($\circ$는 아다마르 곱(Hadamard product) 연산자)를 구하는 방법을 사용할 수 있다. 이러한 벡터 가중치를 사용하면 텍스트 시퀀스 내에서 특정한 토큰에 대해 높거나 낮은 가중치를 주는 대신 v의 i번째 차원의 값 v_i마다 텍스트 시퀀스 내의 토큰에 대한 서로 다른 가중치를 사용해 계산할 수

있게 된다. 이를 통해 개별 벡터의 값 v_i 가 갖는 정보를 계산하기 위해 적절한 토큰에 높거나 낮은 가중치를 부여할 수 있게 되리라고 기대할 수 있다. 이러한 차원별 주의 가중치는 $A = \text{softmax}(V \tanh(WH^T))$ (V는 $V \in \mathbb{R}^{u \times u}$인 가중치 행렬)와 같이 계산할 수 있으며 벡터 표현은 $1 \cdot A \circ H$ ($1 = (1, 1, \dots, 1) \in \mathbb{R}^n$)으로 계산할 수 있다.

또한 GRU 혹은 LSTM과 같은 순환 신경망을 사용한 인코더 모형을 사용할 수 있다. 그러나 본 논문에서 대상으로 하는 최대 수천 단어에 달하는 텍스트 입력에 순환 신경망을 적용하는 것은 많은 양의 메모리와 연산 능력을 필요로 하며 실용적으로 불가능할 수 있다.

인코더가 적절한 벡터 표현을 생성할 수 있도록 학습할 수 있게 하기 위해서는 벡터 표현에 대한 손실 함수(Loss function)이 필요하다. 인코더-디코더 구조의 접근에서는 인코더가 인코딩한 벡터 표현을 입력으로 받는 디코더가 벡터 표현에 대한 손실 함수로 기능한다고 생각할 수 있다. 본 연구에서는 다음과 같은 모형을 디코더로 적용하였다.

1. 단어 자루 예측. 텍스트 입력에 대한 언어 모형 $p(w_1, w_2 \cdots, w_{N-1}, w_N|c)$에 대해서 나이브 베이즈 모형과 같이 각 토큰 w_i 가 c 에 대해 조건부 독립이라고 하면 $p(w_1, w_2 \cdots, w_{N-1}, w_N|c) = p(w_1|c)p(w_2|c) \cdots p(w_{N-1}|c)p(w_N|c)$ 와 같이 분해할 수 있다. 이는 텍스트 시퀀스 내의 토큰 각각을 예측하는 것이라고 할 수 있으며 따라서 단어 자루를 예측하는 모형이라고 볼 수 있다. 추가적으로 단어 자루 모형에서 단어의 등장 빈도가 1 이상인 경우를 1 로 변환하면 이는 등장하는 단어의 확률을 1 로 예측하고 등장하지 않는 단어의 확률을

0 으로 예측하는 로지스틱 다중 분류 모형으로 생각할 수 있다. 본 연구에서는 기존 연구와 달리 바이그램과 트라이그램 또한 예측하도록 디코더를 확장하였다(Pagliardini, Gupta, & Jaggi, 2017).

2. 순환 신경망 언어 모형. c에 대한 조건부 독립을 가정하는 대신 신경망 언어 모형에서 널리 쓰이는 순환 신경망 언어 모형을 디코더로 사용할 수 있다(Bengio et al., 2003; Q. Le & Mikolov, 2014). 이전 단계의 입력 토큰 $w_1, w_2 \cdots, w_{N-1}$에 대해 순환 신경망을 통해 계산된 은닉 상태(Hidden state)를 $s(w_1, w_2 \cdots, w_{N-1})$라고 하면 순환 신경망 언어 모형은 언어 모형 $p(w_1, w_2 \cdots, w_{N-1}, w_N)$를 $p(w_1)p(w_2|s(w_1)) \cdots p(w_N|s(w_1, w_2 \ldots, w_{N-1}))$와 같이 분해하게 된다. 여기서 인코더의 출력 c를 추가적인 입력으로 추가하여 $p(w_1|c)p(w_2|s(w_1), c) \cdots p(w_N|s(w_1, w_2 \ldots, w_{N-1}), c)$ 와 같은 형태로 분해하는 것을 생각할 수 있다. 이는 은닉 상태 s_i와 입력 w_i에 대한 RNN 셀의 값 $\mathrm{RNN}(s_i, w_i)$에 c를 더하고 다음 토큰에 대한 확률을 구하기 위한 소프트맥스 함수와 선형 변환을 추가한 $\mathrm{softmax}(\mathrm{W}(\mathrm{RNN}(s_i, w_i) + c))$ 와 같은 형태로 구현된다. 언어 모형을 사용한 디코더는 단순히 텍스트에 출현한 단어의 확률을 변화시키는 것 뿐만 아니라 이전 단어들의 맥락에 대해 적절한 방식으로 확률을 변화시키도록 학습하는 것을 기대할 수 있다.

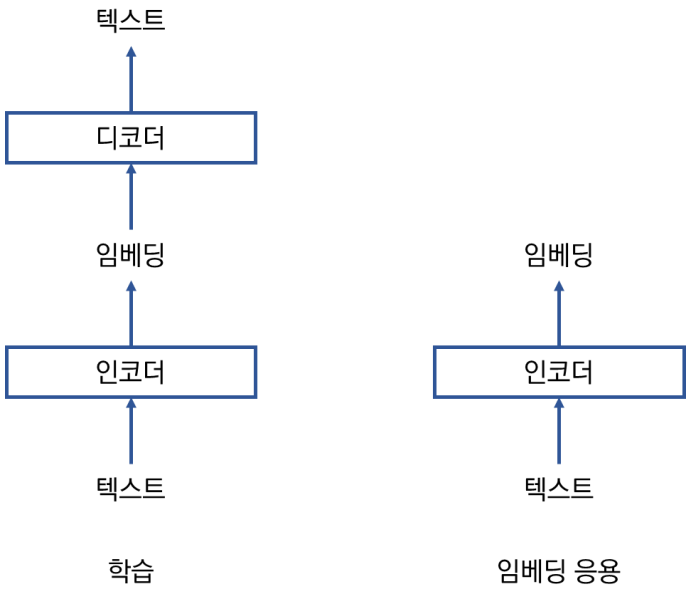

그림 4 본 연구에서 제안하는 모형의 개념적 구조. 모형의 학습 과정에서는 텍스트 입력을 인코더로 인코딩한 결과인 텍스트 임베딩을 디코더에 입력해 텍스트를 출력하는 방식으로 학습된다. 테스트 과정에서는 텍스트를 인코더로 인코딩한 결과를 입력 텍스트에 대한 임베딩, 즉 텍스트의 벡터 표현으로 사용한다.

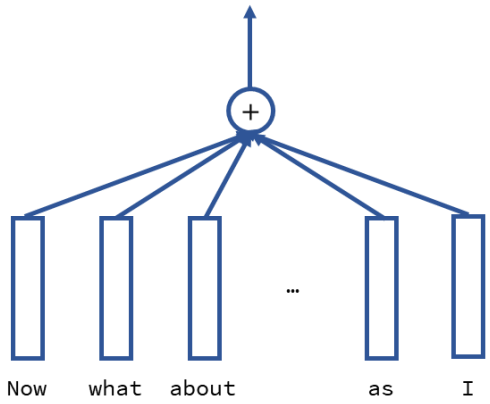

그림 5 단어 임베딩의 평균을 활용한 인코더. 개별 단어에 대한 임베딩 벡터의 합 혹은 평균을 텍스트 시퀀스에 대한 벡터 표현으로 사용한다.

추가적으로 유니그램 토큰 뿐만 아니라 바이그램, 트라이그램 등 n-그램 토큰을 사용하여 인코더의 성능을 보강할 수 있다.

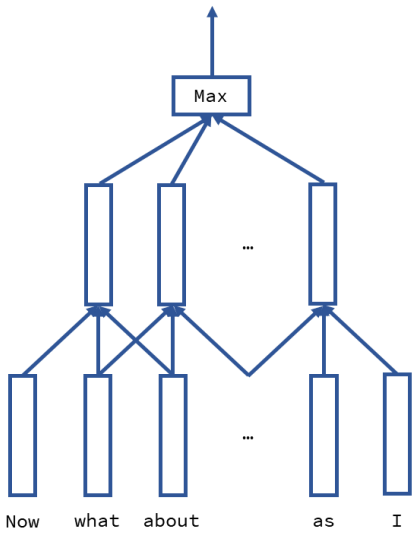

그림 6 컨볼루션 신경망을 활용한 인코더. 단어 임베딩의 시퀀스를 컨볼루션 신경망에 입력한 다음 시퀀스의 길이 방향으로 전역 최대값 혹은 전역 평균 풀링(그림에서는 전역 최대값 풀링)을 취한 결과를 텍스트 시퀀스의 벡터 표현으로 출력한다.

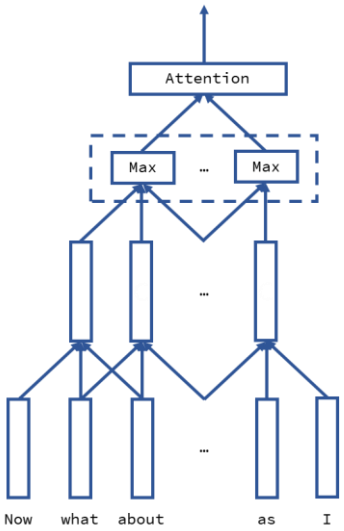

그림 7 컨볼루션 신경망과 자기 주의 메커니즘을 활용한 인코더. 컨볼루션 신경망에서와 같이 단어 임베딩의 시퀀스를 컨볼루션 신경망에 입력한 다음 컨볼루션 신경망을 통과한 시퀀스에 대해 자기 주의 레이어에서 계산된 주의 가중치로 가중 평균을 계산하여 텍스트 시퀀스의 벡터 표현으로 출력한다. 점선으로 표시된 최대값 풀링은 선택적으로 적용이 가능하다.

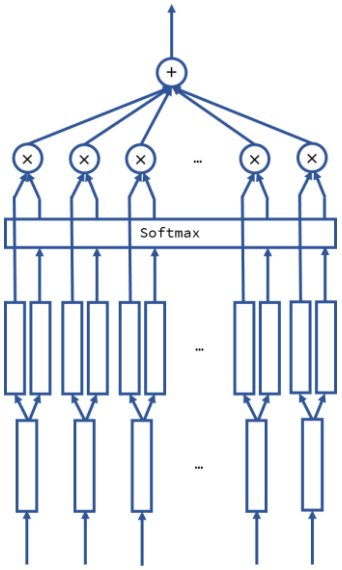

그림 8 자기 주의 레이어의 구조. 입력 시퀀스는 선형변환을 통해 두 개의 시퀀스로 사상(Projection)된다, 하나의 시퀀스는 Softmax 함수를 통과하여 총합이 1인 주의 가중치 벡터로 변환된다. 주의 가중치 벡터를 다른 하나의 시퀀스에 곱하고, 계산된 시퀀스 벡터들을 더한 벡터를 출력한다.

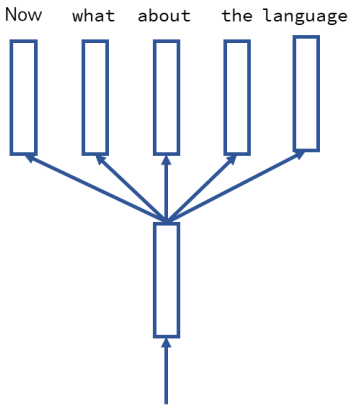

그림 9 단어 자루 디코더의 구조. 단어 자루 디코더는 입력 벡터를 통해 원 텍스트 시퀀스에 등장한 단어들을 예측하는 방식으로 입력 벡터를 디코딩하도록 학습된다.

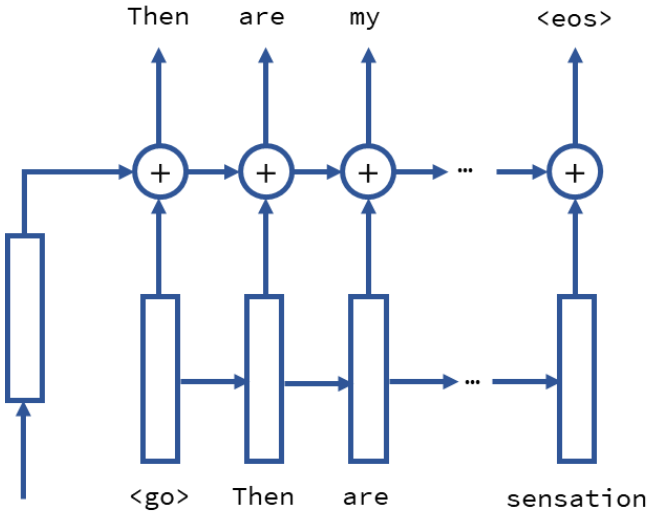

그림 10 순환 신경망 언어 모형 디코더의 구조. 순환 신경망 언어 모형 디코더는 순환 신경망을 사용한 일반적인 언어 모형과 같이 이전 단어 입력들을 통해 다음 단어를 예측하도록 학습된다. 여기에서 순환 신경망 언어 모형 디코더는 추가적으로 입력된 텍스트 임베딩 벡터를 매 토큰에 대한 순환 신경망 셀의 출력에 더한 벡터를 통해 다음 단어를 예측하도록 학습된다.

디코더의 입력은 인코더의 입력과 동일하나 디코더의 입력에는 시퀀스의 시작을 가리키는 토큰이 추가되었다. 디코더의 학습 목표는 한 단계 이후의 시퀀스를 예측하는 언어 모형(Language model)이며 학습 목표 시퀀스의 끝에는 시퀀스의 끝을 가리키는 토큰이 추가되었다.

제 2 절 데이터셋과 전처리 및 학습 절차

2.1 데이터셋

텍스트의 벡터 표현 모형의 학습을 위해 다음과 같은 데이터셋을 사용하였다.

1. Amazon review. Amazon review 데이터셋(He & McAuley, 2016)은 1996 년 5 월에서 2014 년 7 월 까지의 Amazon 의 제품 리뷰를 수집한 데이터이다. 전체 데이터는 총 1 억 4 천만 개의 리뷰를 포함하고 있다. 본 연구에서는 Amazon review 데이터셋 중 영화와 TV 에 대한 리뷰 170 만 개를 사용하였다.

2. WikiText-103 데이터셋. WikiText-103 데이터셋(Merity, Xiong, Bradbury, & Socher, 2016)은 위키피디아의 Good, Featured 문서를 수집한 데이터셋이다. 본 연구에서는

WikiText-103 데이터셋에서 개행으로 구분된 각 문단을 하나의 문서로 보고 모형을 학습하였다.

위의 데이터셋으로 학습된 모형을 평가하기 위해서 다음과 같은 데이터셋을 활용하였다.

1. IMDB. IMDb 영화 리뷰 데이터셋(Maas et al., 2011)은 IMDb 에서 수집된 100,000 개의 영화 리뷰로 구성되어 있다. 이 데이터셋은 평점 레이블이 존재하는 25,000 개의 학습용 데이터와 25,000 개의 테스트 데이터로 구성되어 있으며 비지도 학습에 사용될 수 있는 평점 레이블이 없는 50,000 개의 데이터가 추가적으로 포함되어 있다. 학습용 데이터와 테스트 데이터는 각각 12,500 개의 긍정적 리뷰와 부정적 리뷰로 구성되어 있으며 긍정 리뷰는 영화 평점이 6 점 이상인 리뷰이며 부정적 리뷰는 영화 평점이 5 점 이하인 리뷰이다.

2. MR. Movie Review 데이터셋(Pang & Lee, 2005)은 Rotten Tomatoes 에서 수집된 단일 문장 리뷰로 구성된 데이터셋이며 긍정 혹은 부정 리뷰로 분류되어 있다.

3. MR-2k. Movie Review 2k 데이터셋(Pang & Lee, 2004)은 IMDb 의 rec.arts.movies.reviews 뉴스그룹 아카이브에서 수집된 데이터셋이다. 각각 1000 개의 긍정 리뷰와 부정 리뷰로 구성되어 있다.

4. SST. SST(Socher, Perelygin, Wu, et al., 2013)는 MR 데이터셋을 재가공하여 구축된 데이터셋으로 MR 데이터셋에 다시 레이블을 부여하고 중립으로 분류된 데이터를 제거한 데이터셋이다.

5. CR. Customer Review 데이터셋(Hu & Liu, 2004)은 Amazon 에서 수집된 전자 제품(디지털 카메라, MP3 플레이어 등)에 대한 사용자 리뷰를 수집하여 구성된 데이터셋이다.

6. SUBJ. Subjectivity 데이터셋(Pang & Lee, 2004)은 Rotten Tomatoes 의 리뷰에서 추출한 문장을 주관적인 문장으로 레이블하고 IMDb 영화 플롯 요약에서 추출한 문장을 객관적인 문장으로 레이블하여 수집된 데이터셋이다.

7. MPQA. MPQA 데이터셋(Wiebe, Wilson, & Cardie, 2005)은 뉴스 기사에서 추출한 문장들로 구성된 데이터셋이다. 뉴스 기사에서 각 문장의 의견이 긍정적인지 부정인지를 분류하여 구축된 데이터이다.

8. TREC. TREC(Roth & Li, 2002)은 질의문으로 구성되어 있으며 각 질의문을 6 가지 카테고리, 약어(Abbreviation), 개체(Entity), 설명(Description), 사람(Human), 장소(Location), 수치(Numeric Value)로 분류한 데이터셋이다.

데이터셋	데이터 수	토큰 수	평균 길이	어휘의 수
Amazon review	1.7M	267,979,184	158	717K
WikiText-103	1.8M	81,801,351	45	218K

표 2 모형 학습을 위한 데이터셋의 통계

데이터셋	과제	데이터 수	평균 길이	어휘의 수
IMDB	감성 분류 (영화)	50,000	231	392K
MR	감성 분류 (영화)	10,662	21	19K
MR-2k	감성 분류 (영화)	2,000	787	51K
SST	감성 분류 (영화)	9,613	19	16K
CR	제품 리뷰	3,775	19	5,340
SUBJ	주관성/객관성	10,000	23	21K
MPQA	의견의 방향	10,606	3	6,246
TREC	질문의 종류	5,952	10	9,592

표 3 테스트 데이터셋의 통계

2.1 학습 절차 및 모형 상세

모든 데이터셋은 소문자화한 이후 NLTK 3.2.4(Bird, Klein, & Loper, 2009)로 단어 토큰을 추출하였다. 불용어(Stopword)는 제외하지 않았으며 알파벳으로 구성된 단어만 포함하였다.

단어 임베딩 벡터의 합 혹은 평균을 활용한 인코더의 경우에는 512 차원과 768 차원의 단어 임베딩을 사용하였으며 따라서 텍스트 시퀀스의 벡터 표현의 크기 또한 512 차원 혹은 768 차원이 되었다. 합 혹은 평균을 사용하는 것의 특성을 비교하기 위해 합을 사용한 경우와 평균을 사용한 경우를 나눠 학습하여 결과를 비교하였다.

컨볼루션 신경망을 활용한 인코더의 경우에는 Kim (2014)에서와 같이 다양한 크기의 커널을 갖는 컨볼루션 레이어를 결합(Concatenation)하는 방식을 사용하였다. 커널의 크기가 갖는 특성을 분석하기 위해 커널 크기가 1, 2, 3인 컨볼루션 레이어를 결합하는 경우와 크기가 Kim (2014)에서와 같이 커널 크기가 3, 4, 5인

컨볼루션 레이어를 결합하는 경우를 비교하였다. 커널 크기가 5인 단일 컨볼루션 레이어를 사용하는 경우 또한 비교하였다. 세 개의 컨볼루션 레이어를 결합하는 경우에는 각각의 컨볼루션 레이어의 채널(Channel) 크기는 256으로 설정하였으며 단일 컨볼루션 레이어를 사용하는 경우에는 768로 설정되었다. 따라서 최종 벡터 출력의 크기는 768로 같았다. 또한 전역 최대값 풀링과 전역 평균 풀링을 활용하는 경우의 차이를 비교하기 위해 전역 최대값 풀링을 사용한 모형과 전역 평균 풀링을 사용한 모형을 학습하였다. 컨볼루션 레이어에 단어를 입력하기 위한 임베딩 레이어의 크기는 256으로 설정하였다.

컨볼루션 신경망과 자기 주의 메커니즘을 활용한 인코더의 경우에는 커널 크기 5의 컨볼루션 레이어를 사용하였다. 풀링의 활용으로 시퀀스 길이의 축소와 수용장 크기의 확대가 성능에 미치는 영향을 분석하기 위해 최대값 풀링을 적용한 모형과 적용하지 않은 모형을 학습하여 비교하였다. 또한 개별 채널에 대해 서로 다른 주의 가중치를 적용하는 경우와 모두 같은 가중치를 적용하는 경우도 비교하였다. 컨볼루션 레이어의 채널 크기는 768이었으며 주의 가중치를 계산하기 위해 사용되는 선형 변환의 결과 차원의 크기도 동일하게 768이었다.

컨볼루션 신경망의 경우 추가적인 컨볼루션 레이어를 추가하여 신경망의 표현력 향상을 꾀할 수 있다. 그러나 텍스트 분류 등의 문제에 대해서 다층 컨볼루션 신경망을 통한 표현력 향상이 도움이 되는가에 대해서는 회의적인 결과가 보고되었다(H. T. Le, Cerisara, & Denis, 2017). 본 연구에서도 컨볼루션 레이어를 중첩하는 것이 모형의 성능 향상에 유의미한 영향을 미치지 않는다는 것을 발견하였고 따라서 컨볼루션 신경망을 인코더로 활용한 경우에는 단층의 컨볼루션 레이어를 사용하였다. 컨볼루션 레이어에는 모두 배치 정규화를 적용하였으며

활성화 함수(Activation function)로는 ReLU를 사용하였다.

단어 자루 디코더는 단어 개수만큼의 차원으로 벡터 입력을 변환하는 선형 사상과 함께 시그모이드 활성화 함수를 통해 구현되었다. 단어 자루 디코더는 추가적으로 부적 샘플링(Negative sampling) 등을 통해 가속화될 수 있다(Mikolov, Chen, Corrado, & Dean, 2013).

순환 신경망 언어 모형 디코더로는 단순 RNN 셀을 사용한 순환 신경망 레이어를 사용하였다. 텍스트의 전역적인 의미와 맥락적 정보는 텍스트 임베딩 벡터에 반영된다고 가정하므로 장거리 관계를 반영하기 위한 LSTM 등의 모형은 필요하지 않았다. 또한 LSTM 등의 모형을 사용한 경우 순환 신경망이 스스로 텍스트의 맥락적 정보를 이전 입력 토큰들을 통해 획득할 수 있으므로 학습 과정에서 텍스트 임베딩 벡터가 맥락적 정보를 저장하는 것을 방해할 위험 또한 존재한다. tanh를 사용한 2층의 셀 크기 300 순환 신경망이 사용되었으며 순환 신경망의 입력으로는 크기 100의 임베딩 레이어가 사용되었다. 또한 순환 신경망의 입력과 텍스트 임베딩 벡터를 결합하여 다음 단어의 확률을 예측하는 연산을 고속화하고 필요한 GPU 메모리의 양을 축소하기 위해 적응적 소프트맥스(Adaptive Softmax)를 사용하였다. 적응적 소프트맥스의 역치(Threshold)는 (5000, 10000, 20000)으로 적용하였다. 추가적으로 절삭 역전파(Truncated backpropagation)이 활용되었으며 메모리의 요구치에 따라 절삭 시퀀스의 길이는 32 혹은 48로 지정되었다.

모형은 PyTorch 0.2.0.post4(Abadi et al., 2016)로 구현되었으며 하나의 NVIDIA GeForce GTX 1080 그래픽 카드에서 수행되었다. 모형의 최적화에는 Adam(Kingma & Ba, 2015)을 사용하였으며

Adam의 매개변수는 학습율(Learning rate) 0.0003, $\beta_1 = 0.9, \beta_2 = 0.999, \varepsilon = 1e-8$ 으로 지정되었다. 순환 신경망 디코더를 사용한 경우에는 그래디언트는 전체 그래디언트의 노름이 10을 넘지 않도록 클리핑되었다(Clip by global norm)(Pascanu, Mikolov, & Bengio, 2013).

모형의 학습 이후 학습 데이터와 테스트 데이터에서 모형을 통해 추출된 문단 임베딩을 사용해 리뷰의 긍부정 분류를 위한 분류기를 학습시켜 문단 임베딩의 성능을 평가하였다. 분류기로는 scikit-learn 0.19.의 SVM 구현을 사용하였으며 하이퍼파라미터 C는 0.01로 지정되었다.

시각화를 위해서 scikit-learn의 t-SNE(t-distributed Stochastic Neighborhood Embedding)(Maaten & Hinton, 2008) 구현과 umap-learn의 UMAP 구현을 사용하였다. t-SNE의 당혹도(Perplexity)는 30으로 지정하였으며 척도(Metric)로는 코사인 거리를 사용하였다. 시각화는 학습된 인코더로 IMDB 테스트 데이터를 인코딩한 다음 3000개를 샘플링하여 t-SNE 혹은 UMAP을 통해 2차원에 임베딩한 결과로 산포도를 그리는 방식으로 수행되었다.

제 4 장 실험 결과

제 1 절 데이터셋에 대한 전이 학습 결과

표 1에는 학습을 마친 인코더로 임베딩된 문단 벡터를 입력으로 하여 각종 자연어 분류 과제에 학습시킨 SVM에 대해 테스트 데이터로 정확도(Accuracy)를 측정한 결과가 나타나 있다.

각 테스트 데이터에서 학습된 모형들은 기존의 결과와 유사한 결과를 보여주었다. 특히 Amazon review 데이터에 대해 학습된 모형들은 영화 리뷰와 관련된 데이터(IMDB, MR, MR-2k, SST)에서 더 나은 결과를 보여주었다. 또한 CR과 SUBJ는 Amazon에서 수집된 리뷰라는 것과 Rotten Tomatoes, IMDb에서 수집된 리뷰와 플롯 요약이라는 특징이 있기 때문에 Amazon review에 학습된 모형이 전이되기 용이했다고 생각된다(Radford, Jozefowicz, & Sutskever, 2017).

반면 WikiText-103에 대해 학습된 모형은 IMDB, MR, MR-2k, SST 등의 데이터에 대해서는 Amazon review 데이터에 학습된 모형들에 비해 나은 성능을 보여주지 못했다. 그러나 MPQA와 TREC과 같이 영화 리뷰가 아닌 뉴스 기사 혹은 질문 등으로 구성된 데이터셋에 대해서는 Amazon review 데이터에 대해 학습한 모형에 대해 유사하거나 혹은 더 나은 성능을 보여주었다.

모형	IMDB	MR	MR-2k	SST	CR	SUBJ	MPQA	TREC
SVM-uni	87.0	76.2	86.3	-	79.9	90.8	86.1	-
NBSVM-bi	91.2	79.4	89.5	83.1	81.8	93.2	86.3	-
word2vec	-	77.7	-	79.7	79.8	90.9	88.3	83.6
GloVe	-	78.7	-	79.8	78.5	91.6	87.6	83.6
Paragraph Vector	-	61.5	-	-	68.6	76.4	78.1	55.8
Paragraph Vector (DBOW)	-	60.2	-	-	66.9	76.3	70.7	59.4
SDAE	-	74.6	-	-	78.0	90.8	86.9	78.4
Skip-Thought	-	76.5	-	82.0	80.1	93.6	87.1	92.2
FastSent	-	70.8	-	-	78.4	88.7	80.6	76.8
CaptionRep (bow)	-	61.9	-	-	69.3	77.4	70.8	72.2
DictRep (bow)	-	76.7	-	-	78.7	90.7	87.2	81.0
NMT En-to-Fr	-	64.7	-	-	70.1	84.9	81.5	82.8
제안 모형 - 단어 자루 디코더								
임베딩 평균	90.7	77.6	88.5	80.8	78.0	91.0	80.1	83.4

컨볼루션 (커널 크기 3, 4, 5)	89.0	76.3	88.1	80.2	77.4	90.0	80.0	82.8
주의 + 토큰 단위 가중치	89.7	76.0	88.2	79.2	76.5	88.9	78.0	76.0
임베딩 평균 + WikiText-103	82.1	70.5	79.7	73.3	75.7	88.6	82.3	84.8
제안 모형 - 순환 신경망 언어 모형 디코더								
임베딩 평균	90.3	78.4	88.9	83.4	80.0	90.1	83.5	85.0
컨볼루션 (커널 크기 3, 4, 5)	89.3	75.8	87.2	78.0	75.2	85.7	81.7	76.8
주의 + 토큰 단위 가중치	88.8	76.1	87.2	78.9	76.4	84.3		75.0

표 4 테스트 데이터에 대한 성능 비교. SVM-uni 및 NBSVM-bi 는 각 데이터셋에 대한 지도학습(Supervised Learning) 결과이며 Wang & Manning (2012)의 결과를 따름. word2vec (Mikolov, Sutskever, et al., 2013)과 GloVe (Pennington, Socher, & Manning, 2014)는 Toronto Book Corpus (Zhu et al., 2015)에 대해 학습됨(Conneau et al., 2017). FastSent (Hill et al., 2016), Skip-Thought (Kiros et al., 2015), SDAE (Hill et al., 2016), FastSent (Hill et al., 2016), CaptionRep (Chen et al., 2015), DictRep (Hill, Cho, Korhonen, & Bengio, 2015)의 결과는 Hill et al. (2016)의 결과를 따름. 임베딩 평균 + WikiText-103 을 제외한 제안된 모형들은 Amazon review 데이터셋에 대해 학습됨. 제안된 모형에 해당하는 값 중 가장 높은 값은 굵게 표시되었으며 전체 모형 중에서 가장 높은 값은 밑줄 표시됨.

전반적으로 학습된 모형 중에서는 임베딩 평균을 활용한 모형이 가장 나은 성능을 보여주었다. 컨볼루션 신경망을 사용한 모형의 경우 테스트 데이터에 따라 임베딩 평균 모형과 근접한 성능을 보여주기도 했지만 임베딩 평균보다 나은 경우는 나타나지 않았다. 자기 주의 메커니즘을 활용한 모형의 경우 영화 리뷰 데이터에 대해서는 컨볼루션 신경망을 사용한 모형과 유사한 성능을 보여주었지만 그 밖의 데이터에 대해서는 컨볼루션 신경망 모형에 미치지 못하는 성능을 보여주었고 특히 TREC에 대해서 취약한 모습을 보여주었다. 이는 문서의 평균 길이가 158인 Amazon review에 대해 학습된 주의 가중치들이 평균 길이 3인 TREC으로 전이되었을 때 문제가 되었을 것이라고 생각된다. 컨볼루션 신경망과 전역 풀링을 조합한 모형에 대해 표현력이 증가한 모형이나 표현력의 증가가 유의미한 성능 향상으로 이어지지는 않은 것으로 보인다.

디코더 모형의 구조에 따른 차이는 크게 나타나지 않았지만 임베딩 평균 인코더를 사용한 경우에는 순환 신경망 언어 모형 디코더가, 컨볼루션 신경망 인코더를 사용한 경우에는 단어 자루 디코더에서 더 나은 성능이 나타났다. 이는 인코더와 디코더 모형 선택에 따라 상호작용이 존재할 수 있다는 점을 시사한다.

제 2 절 모형의 변형에 따른 성능 변화

다음으로 모형의 각종 변형에 대해 나타나는 성능 변화의 결과가 나타나 있다.

모형	IMDB
전역 최대값 풀링 (커널 크기 3, 4, 5)	85.7
전역 평균 풀링 (커널 크기 3, 4, 5)	89.0
단일 컨볼루션 레이어 (커널 크기 5)	89.2
컨볼루션 레이어 (커널 크기 1, 2, 3)	89.0

표 5 컨볼루션 신경망을 사용한 인코더의 변형에 대한 IMDB 분류 성능 변화

컨볼루션 신경망을 사용하는 경우 전역 평균 풀링을 사용하는 것에 대해 전역 최대값 풀링을 사용하는 경우 성능의 감소가 두드러졌다. 전역 평균 풀링을 사용하는 경우 커널 크기를 3, 4, 5로 설정한 경우, 1, 2, 3으로 설정한 경우, 그리고 단일 커널을 사용한 경우 사이에는 두드러지는 성능의 차이가 나타나지 않았다.

모형	IMDB
주의 + 채널 단위 가중치	88.5
주의 + 채널 단위 가중치 + 최대값 풀링 (풀링 크기 4)	89.0
주의 + 토큰 단위 가중치	89.3

표 6 자기 주의 메커니즘을 사용한 인코더의 변형에 대한 IMDB 분류 성능 변화

자기 주의 메커니즘을 활용하는 경우에는 큰 차이가 드러나지는 않았으나 채널 단위 가중치를 설정하는 것에 비해 토큰 단위 가중치를 설정하는 것이 더 나은 성능을 보여주었다. 또한 최대값 풀링을 사용하는 경우와 사용하지 않는 경우 사이에도 큰 차이가 나타나지는 않았다. 최대값 풀링과 토큰 단위 가중치가 필요한 연산의 양을 감소시킨다는 것을 고려할 때 더 나은 특성을 갖는다고 생각할 수 있다.

n-그램	n-그램 및 코사인 유사도			
film	effects_are	0.36	the_cast_are	0.32
	cast_are	0.36	are_supposed	0.32
	film_are	0.34	special_effects_are	0.32
	scenes_are	0.33	are_supposed_to	0.32
	the_film_are	0.33	movie_are	0.32
the_film	of_the_film	0.47	and_the_film	0.38
	the_film_is	0.47	the_film_was	0.36
	in_the_film	0.45	is_the_film	0.33
	not	0.40	the_film_and	0.32
	to_the_film	0.38	the_film_he	0.30
of_the_film	the_film	0.47	minutes_of_the	0.26
	the_film_he	0.29	look_of_the	0.25
	the_film_we	0.28	look_of	0.25
	the_film_when	0.27	the_film_the	0.23
	the_film_are	0.26	of_the_season	0.23
good	film_there	0.44	the_end_there	0.42
	the_film_there	0.44	of_course_there	0.41
	though_there	0.42	think_there	0.41
	course_there	0.42	even_though_there	0.41
	the_end_there	0.42	film_there_are	0.40
good_movie	a_good_movie	0.66	good_movie_i	0.50
	very_good_movie	0.65	really_good_movie	0.50
	pretty_good_movie	0.55	good_movie_with	0.46
	good_movie_and	0.51	good_movie_for	0.46
	good_movie_i	0.50	good_movie_it	0.43
a_good_movie	good_movie	0.66	good_movie_to	0.39
	good_movie_the	0.47	good_movie_and	0.38
	good_movie_that	0.45	pretty_good_movie	0.37
	good_movie_it	0.44	good_movie_with	0.37
	good_movie_but	0.41	very_good_movie	0.37
bad	each_season	0.41	final_season	0.36
	second_season	0.37	the_whole_season	0.36
	next_season	0.37	season_but	0.36

	first_season	0.36	the_next_season	0.35
	season_so	0.36	entire_season	0.35
bad_movie	a_bad_movie	0.93	a_bad	0.41
	bad_movie_but	0.74	in_a_bad	0.33
	is_a_bad	0.45	such_a_bad	0.31
	it_a_bad	0.42	be_a_bad	0.28
	was_a_bad	0.42	with_a_bad	0.26
a_bad_movie	bad_movie	0.93	it_a_bad	0.41
	bad_movie_but	0.81	not_a_bad	0.40
	a_bad	0.57	such_a_bad	0.39
	was_a_bad	0.55	in_a_bad	0.38
	is_a_bad	0.54	with_a_bad	0.38

표 7 각 n-그램 및 n-그램과 코사인 유사도가 높은 n-그램들

제 2 절 학습된 모형의 특성 분석

임베딩 평균 인코더를 사용한 경우 일반적인 유니그램 단어들에 대한 임베딩 뿐만 아니라 바이그램과 트라이그램에 대한 임베딩도 학습되게 된다. 이렇게 학습된 n-그램 임베딩의 의미적 특징을 살펴보기 위해 선정한 n-그램과 코사인 유사도(Cosine similarity)가 가장 높은 n-그램 임베딩에 해당하는 단어들을 추출한 결과가 표 7 에 나타나 있다. 유니그램의 경우 선정된 유니그램이 포함되지 않은 n-그램에서도 유사도가 높게 나타난 경우가 보이는 반면 바이그램과 트라이그램의 경우 선정한 바이그램과 트라이그램이 포함되고 문법적 기능을 하는 전치사 등이 변화한 n-그램들에서 유사도가 높게 나타났다. 이는 바이그램과 트라이그램에서 유니그램에 비해 단어의 의미가 구체화 되었기 때문이라고 생각할 수 있다. 예를 들어 good 이라는 단어는 Amazon review 데이터라는 특성을 고려하더라도 다양한 맥락에서 사용될 것이라고 예상할 수 있다. 그러나 good_movie 혹은

a_good_movie 라는 n-그램을 고려하면 '좋은 영화'를 가리키는 것으로 의미가 구체화되고 따라서 '좋은 영화'라는 의미를 갖는 n-그램 단어들과 '좋은 영화'라는 의미를 크게 변화시키지 않는 문법적 기능의 전치사들이 변화한 단어들이 높은 유사도가 나타나게 된다고 생각할 수 있다. 이는 바이그램 및 트라이그램과 유사도가 높은 n-그램 중에 유니그램 단어가 거의 포함되지 않았다는 점에서도 유추할 수 있다. 즉 good_movie 는 good 이라는 단어가 일반적으로 갖는 '좋은'이라는 의미를 포함하고 있지만 movie 라는 단어로 인해 추가적으로 '좋은 영화'라는 의미를 갖게 되었기에 good 라는 유니그램과의 유사도가 낮아진 것이라고 생각할 수 있다.

그림 11 과 그림 12 에는 IMDB 리뷰 데이터를 t-SNE 와 UMAP 로 시각화한 결과가 나타나 있다. 특징적인 점은 임베딩 평균 인코더를 사용한 경우와 컨볼루션 신경망 인코더를 사용한 경우의 양상이 다르게 나타난다는 점이다. 임베딩 평균 인코더를 사용한 경우 분류기가 보여주는 성능에 비해 시각화된 결과는 긍정 리뷰와 부정 리뷰가 매우 중첩된 것으로 나타난다. 반면 컨볼루션 신경망 인코더를 사용한 경우에는 여전히 긍정 리뷰와 부정 리뷰가 크게 중첩되어 있기는 하지만 임베딩 평균 인코더를 사용한 경우에 비해 각각의 리뷰가 서로 밀집한 양상이 나타나고 있다. 물론 t-SNE 와 UMAP 으로 임베딩의 차원이 축소되면서 임베딩의 공간적 관계에 손실이 발생하므로 시각화한 결과를 절대적인 것으로 해석할 수는 없지만 여전히 두 인코더를 사용한 모형이, 디코더가 동일함에도 서로 다른 특성을 갖는 임베딩을 생성했다고 추정할 수 있다.

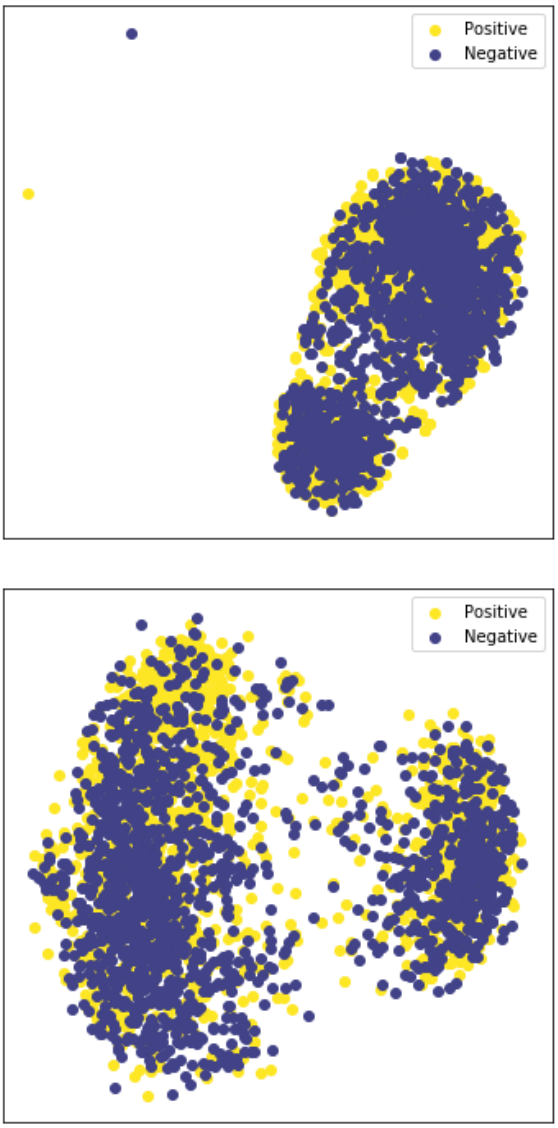

그림 11 임베딩 평균을 사용한 IMDB 리뷰의 t-SNE 와 UMAP 시각화 결과. (위) t-SNE (아래) UMAP

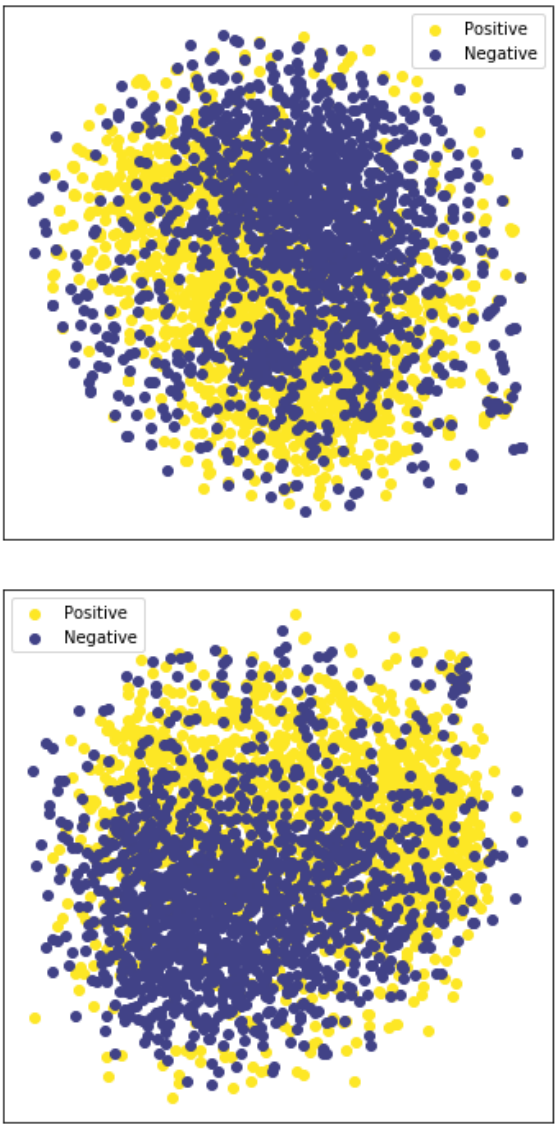

그림 12 컨볼루션 (커널 크기 3, 4, 5)를 사용한 IMDB 리뷰의 t-SNE 와 UMAP 시각화 결과. (위) t-SNE (아래) UMAP

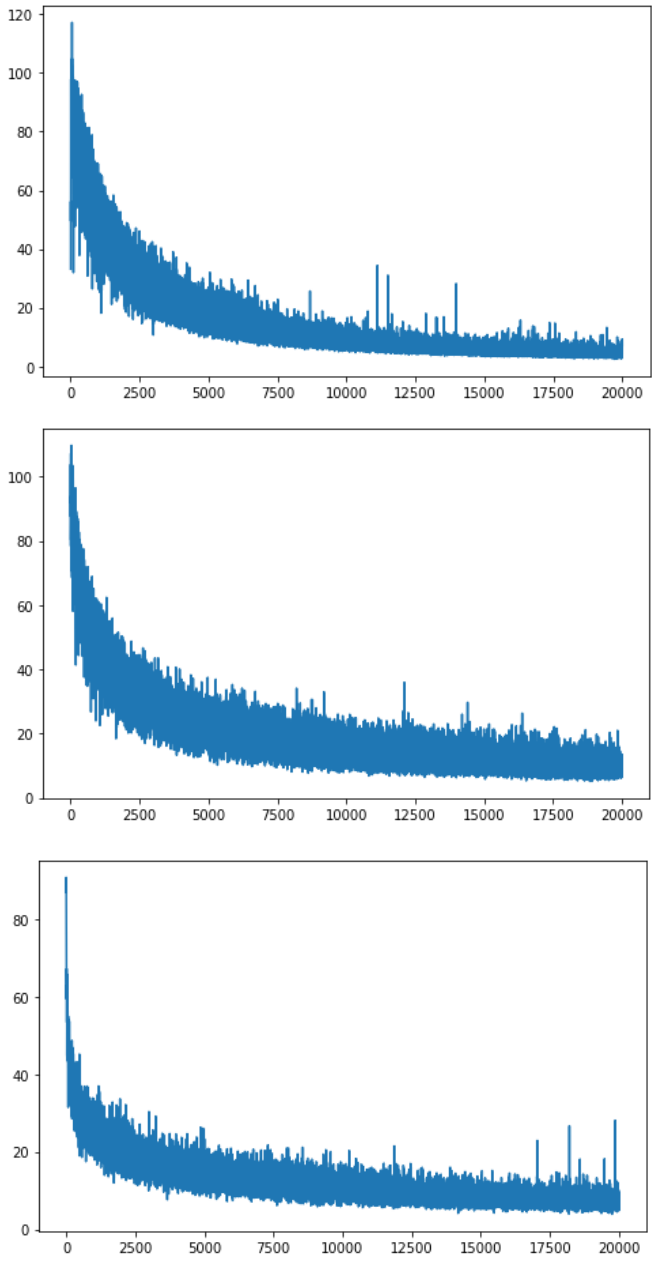

그림 13 임베딩 평균 인코더를 통해 학습된 n-그램 임베딩의 n-그램 빈도 순위에 따른 노름(Norm). (위) 유니그램 (중간) 바이그램 (아래) 트라이그램

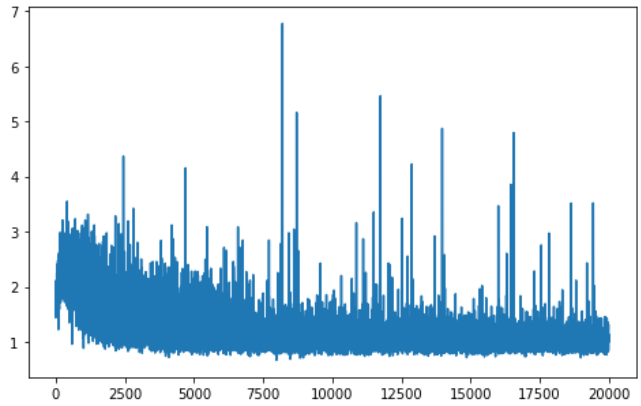

그림 14 컨볼루션 신경망 인코더를 통해 학습된 단어 임베딩의 빈도 순위에 따른 노름

또한 이 두 인코더는 학습한 단어의 임베딩의 특성에 대해서도 차이를 보인다. 임베딩 평균 인코더를 사용한 경우에는 GloVe와 같은 단어 임베딩 모형들에서 나타나는 것과 같이 단어의 빈도가 증가할수록 단어에 해당하는 임베딩 벡터의 유클리드 노름(Norm)이 증가하는 추세가 현저히 두드러진다(Pennington et al., 2014). 컨볼루션 신경망 인코더를 사용한 경우에도 단어의 빈도가 증가함에 따라 노름이 커지는 경향이 나타나기는 하지만 그 정도가 임베딩 평균 인코더를 사용한 경우에 비해 약한 모습을 보여준다.

컨볼루션 신경망의 커널 크기가 임베딩의 특성에 미치는 영향을 분석하기 위해 커널 크기가 3, 4, 5로 설정된 신경망에서 생성된 임베딩과 커널 크기가 1, 2, 3으로 설정된 신경망에서 생성된 임베딩을 변환하는 모형을 시험하였다. 커널 크기 3, 4, 5인 신경망으로 생성한 임베딩을 입력 데이터로 하고 커널 크기가 1, 2, 3인 신경망에서 생성된 임베딩을 예측 데이터로 사용하여 은닉 레이어가 없는 선형 신경망을 학습하였다. 이후 학습된 신경망으로 임베딩을 변환하여 IMDB 분류

과제를 수행하였을 때 나타나는 성능을 측정하였다. 손실 함수로는 L_1 손실을 사용하였다.

컨볼루션 (커널 3, 4, 5)	89.02
컨볼루션 (커널 1, 2, 3)	88.99
L_1 손실 (학습 이전)	0.740
L_1 손실 (학습 이후)	0.067
변환된 임베딩의 정확도	88.16

표 8 커널 크기가 다른 컨볼루션 신경망에서 생성된 임베딩 사이의 변환

실험 결과 선형 모형으로도 학습 이후 낮은 손실이 관측되었고 선형 모형으로 변환한 임베딩으로 학습된 모형으로 예측한 결과가 원 임베딩으로 예측한 결과와 크게 차이가 나타나지 않는다는 것을 관찰할 수 있었다. 이는 커널 크기가 다른 신경망으로 학습된 모형에서 생성된 임베딩이 서로 선형적인 관계를 갖고 있다고 추정할 수 있다.

n-그램	IMDB
유니그램	88.70
바이그램	87.08
트라이그램	80.94
유니그램 + 바이그램	90.00
유니그램 + 트라이그램	90.40
바이그램 + 트라이그램	87.58
유니그램 + 바이그램 + 트라이그램	90.71

표 9 포함한 n-그램 임베딩에 따른 정확도 변화

임베딩 평균 인코더로 학습된 임베딩이 가지는 정보의 특성을 분석하기 위해 학습된 임베딩에서 특정 n-그램을 제외한 임베딩으로 생성한 텍스트 임베딩으로 학습된 분류기의 성능을 측정한 결과가 표 9에 나타나 있다.

제 5 장 논의

각종 테스트 데이터셋에 대해 제안된 임베딩의 평균을 사용한 인코더를 순환 신경망 언어 모형 디코더와 조합하였을 때 단순하면서도 좋은 성능을 나타낼 수 있는 것으로 나타났다. 컨볼루션 신경망을 사용한 인코더의 경우에도 임베딩의 평균을 사용하는 것과 근접한 성능을 보일 수 있었지만 컨볼루션 신경망을 사용하는 것에서 발생하는 추가적인 연산을 고려하면 임베딩의 평균이 실용적인 강점을 갖고 있다고 할 수 있을 것이다. 또한 임베딩의 평균을 인코더로 사용하고 선형 모형을 자연어 분류 등의 과제에 사용하는 경우 임베딩과 가중치의 내적을 저장하여 선형 모형의 연산 속도를 감소시킬 수 있다는 장점이 존재한다. 선형 모형 $f(x) = w^T x$ 에 대해 임베딩의 평균을 입력으로 사용하는 경우 $f\left(\frac{1}{N}\sum x_i\right) = w^T(\frac{1}{N}\sum x_i) = \frac{1}{N}\sum w^T x_i$ 이므로 각 단어의 임베딩과 가중치의 벡터의 내적 $w^T x_i$를 저장해두면 단어 자루 입력을 선형 모형과 사용하는 경우와 같이 N 번의 연산으로 모형의 결과를 연산할 수 있다. 이는 실용적인 목적으로 고속 처리가 필요한 경우에 유용한 특징일 수 있다.

기계 번역 등의 과제에서 좋은 결과를 얻기 위해서는 높은 수준의 표현력이 요구된다는 것을 고려하면(Wu et al., 2016) 좋은 임베딩을 얻기 위한 과제에서 임베딩의 평균만으로 좋은 성능을 확보할 수 있으며 추가적인 표현력이 성능에 반드시 도움이 되지 않을 수 있다는 것은 상반된 사실로 생각할 수도 있다. 그러나 기계 번역 등의 과제에서는 문장의 모든 특성, 문법적 특징들과 정확한 의미적 구조와 관계를 분산 표현에 반영해야 하는 반면 분류 등의 자연어 처리 과제에 적용하기

위한 텍스트의 벡터 표현은 전반적인 맥락적 특징들만을 포착하는 것으로 충분하며 오히려 문법적인 특징 등을 반영하는 것은 대다수의 과제에 도움이 되지 않을 수 있다는 차이가 존재한다. 이를 통해 텍스트 임베딩에 높은 수준의 표현력이 요구되지는 않는다고 생각할 수 있다. 이는 텍스트 감성 분류와 같은 과제에 깊은 신경망이 성능에 이점을 주지 않는다는 사실에서도 확인할 수 있는 특징이다(H. T. Le et al., 2017).

커널 크기가 3, 4, 5 인 컨볼루션 신경망은 잠재적으로 트라이그램을 넘어 4-그램, 5-그램에서 발생하는 관계를 포착할 수 있다. 그러나 본 연구에서는 커널 크기가 1, 2, 3 인 컨볼루션 신경망에 대해 커널 크기가 증가하여 확장된 표현력을 갖는 컨볼루션 신경망에서 유의미한 성능 향상이 발견되지 않았다. 이 점은 커널 크기가 다른 두 컨볼루션 신경망으로 생성된 임베딩 사이의 선형 변환을 큰 정보의 손실 없이 학습할 수 있었다는 것에서 확인할 수 있다. 이는 감성 등의 정보를 포착하기 위한 과제에 4-그램, 5-그램 등의 특성이 추가적인 정보를 제공하지 못하기 때문일 수 있다. 즉 유니그램, 바이그램, 트라이그램이 문서에서 주요한 맥락적 정보를 이미 포함하고 있기 때문일 수 있다. 또한 사용된 디코더가 확장된 표현력을 충분히 활용하지 못했기 때문일 수 있다. 단어 자루 디코더에서는 문서에 포함된 유니그램, 바이그램, 트라이그램을 예측하게 되는데 이러한 예측 과제에서 트라이그램을 넘는 4-그램, 5-그램 차원의 정보가 유의미한 부가적 정보를 제공하지는 못할 수 있다.

텍스트 분류에 사용되는 컨볼루션 신경망의 경우에는 흔히 전역 최대값 풀링이 사용되어왔다(Kim, 2014; H. T. Le et al., 2017). 그러나 본 연구에서는 최대값 풀링보다 평균 풀링이 더 나은 성능을 보여주었다.

이는 컨볼루션 신경망은 일반적으로 개별 문장 단위의 분류에 적용되어 왔다는 점을 고려해 이해할 수 있다. 개별 문장 단위에서는 가장 두드러진 특징을 포착하는 것으로 문장의 특성을 충분히 포착할 수 있다. 즉 한 문장 내에서 긍정적인 정서를 강하게 갖는 n-그램이 존재하는 동시에 전반적으로 부정적인 정서를 갖는 n-그램들이 존재하기는 어렵다. 그러나 다양한 문장이 모인 문서 전체 단위에서는 충분히 가능한 특성이라고 볼 수 있다. 영화에 대한 호평이 포함된 한두 문장이 존재하는 동시에 영화에 대해 부정적인 태도를 보이는 나머지 문장들이 존재하는 문서가 가능하기 때문이다. 이는 다양한 문장으로 구성된 문서를 특징적인 문장 하나만으로 규정하기는 어렵다는 사실을 확인해주는 사실이라고 볼 수 있다.

자기 주의 메커니즘을 통해 각 토큰에 대해 가중치를 부여하는 것은 모형에 추가적인 표현력을 제공하는 방법이다(Lin et al., 2017). 그러나 본 연구의 실험에서는 주의 가중치가 큰 성능적 이점을 보여주지 않았다. 임베딩 평균 인코더에서 학습된 임베딩의 노름 등을 살펴보았을 때 이는 단순한 임베딩의 가중치 없는 평균 등을 사용하는 경우에도 학습 과정에서 임베딩에 필요한 가중치가 부여된 것이라고 생각할 수 있다. 즉 빈도 등의 특징을 바탕으로 임베딩이 서로 다른 가중치를 갖게 되었다고 볼 수 있다(Arora et al., 2017).

위와 같은 실험 결과를 바탕으로 보다 나은 텍스트 임베딩 모형이 갖춰야 할 조건들에 대해 생각해볼 수 있다.

n-그램, 혹은 단순히 인접한 단어 사이의 관계를 넘어 서로 떨어진 단어들 사이의 관계를 포착할 수 있도록 설계된 인코더는 잠재적으로 더 높은 표현력을 가질 수 있다. 서로 떨어진 단어 사이의 관계를 반영하는 것은 다층의 자기 주의 메커니즘을 통해 가능하며 기계 번역과 같은

과제에서는 효과성이 입증되었다(Vaswani et al., 2017). 그러나 앞서 밝혔던 것처럼 이러한 장기적 특성을 반영하는 것이 분류와 같은 자연어 처리 과제에서 반드시 필요한 것인지는 의문의 여지가 있다. 감성 분류와 같은 텍스트에 대한 지도 학습 모형에서도 깊은 컨볼루션 신경망을 사용하여 텍스트에 대한 수용장을 확대한 모형이 그렇지 않은 모형에 대해 성능적 이점을 보여주지 못했다고 한다면 마찬가지로 감성 분류와 같은 과제에 사용하게 될 텍스트 임베딩 모형에서 n-그램을 넘어서는 수준의 수용장을 필요로 한다고 보기는 어렵다. 이는 표 9 에 나타나 있는 임베딩 평균 인코더에서 n-그램을 포함하거나 제외한 다음의 결과에서도 간접적으로 추측할 수 있다. 유니그램 단어 수준에서도 리뷰의 감성에 대한 상당한 수준의 정보가 반영되어 있다는 것을 관찰할 수 있다. 바이그램 혹은 트라이그램이 감성에 대한 추가적인 정보를 갖고 있기는 하지만 유니그램에 바이그램과 트라이그램이 추가되었을 때의 성능 변화와 바이그램과 트라이그램 모두 추가되었을 때의 성능 변화를 통해 바이그램과 트라이그램이 서로 유사한 정보를 갖고 있다는 것 또한 확인할 수 있다. 즉 바이그램에서 트라이그램으로 단어 하나 만큼의 조성적 정보가 추가된 경우에도 그 의미적 관계에 대해 추가적인 정보를 크게 제공하지 못한다는 것이다. 이는 표 10 에서 나타난 것처럼 바이그램과 트라이그램을 사용해 지도학습 모형을 학습하였을 때의 성능적 특성에서도 보여지는 특성이다. 이를 통해 추가적인 n-그램의 확장이 더 많은 정보를 제공하기는 어렵다고 추정할 수 있다.

n-그램	IMDB
유니그램	88.61
유니그램 + 바이그램	91.56
유니그램 + 바이그램 + 트라이그램	91.87

표 10 n-그램 특징의 사용에 따른 SVM 의 성능 변화. 유니그램에서 바이그램을 추가한 경우에 비해 트라이그램을 추가한 경우의 성능 변화가 작다는 것이 나타난다 (Mesnil, Mikolov, Ranzato, & Bengio, 2014)

또한 인코더의 표현력을 충분히 활용하기 위해서는 그 표현력을 활용할 수 있는 디코더의 설계가 필요하다고 할 수 있다. 커널 크기가 서로 다른 컨볼루션 신경망으로 인코딩된 텍스트 임베딩이 서로 유사했다는 점은 디코더가 인코더의 향상된 표현력을 충분히 활용하지 못한 결과라고 볼 수 있다. 이러한 점에서 디코더에는 인코더의 표현력을 충분히 잘 활용할 수 있는 보다 어려운 문제를 부과할 필요가 있다. 그러나 과제의 난이도가 상승한다고 하여 반드시 생성된 임베딩의 특성이 개선되리라고 기대하기는 어렵다. 이는 기계 번역과 같은 난해한 종류의 문제에 대해 학습된 인코더로 생성된 텍스트 임베딩이 반드시 나은 성능을 보여주지 못한다는 사실에서 추정할 수 있다(Hill et al., 2016).

텍스트 임베딩 모형의 유용성은 레이블 없는 대규모의 텍스트를 학습하여 다른 자연어 처리 과제, 특히 레이블된 데이터의 수가 희소한 과제에 적용될 수 있다는 점에 있다. 그림 15에서 나타난 것과 같이 레이블이 있는 데이터가 희소한 경우에 자연어 처리 과제의 성능을 향상시키기 위해 적용될 수 있다. 동시에 단어 임베딩의 효과성과 성능은 단어 임베딩이 학습된 말뭉치(Corpus)의 분량에 크게

의존한다는 점을 고려할 때 대규모의, 그러나 레이블이 있는 데이터보다 용이하게 확보할 수 있는 레이블 없는 데이터를 활용하여 모형의 성능을 개선할 수 있다. 또한 텍스트 임베딩은 레이블 있는 데이터가 충분히 많은 경우에도 분류 모형의 성능을 개선하기 위해 사용될 수 있다(Mesnil et al., 2014).

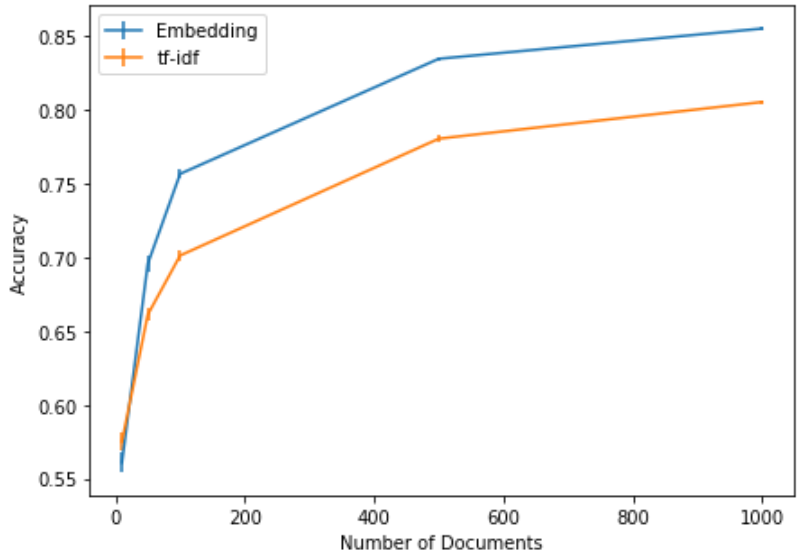

그림 15 문서의 수에 따른 정확도의 변화. 각각 10 개, 50 개, 100 개, 5000 개, 1,000 개의 문서에 대해 학습되었다. 임베딩의 유니그램 수와 동일한 20,000 개의 유니그램을 사용하였으며 적은 숫자의 문서에 대해서는 성능을 감소시켰기에 바이그램 및 트라이그램은 사용되지 않았다.

동시에 효과적인 전이 학습을 위해서는 말뭉치의 특성을 고려할 필요가 있다(Radford et al., 2017). 이는 984,846,357개의 토큰을 포함하고 있는 Toronto Book Corpus에 비해 10% (WikiText-103)에서 30% (Amazon review) 수준의 토큰을 가진 데이터에서도 Word2vec과 대등하거나 혹은 더 나은 성능을 보여줄 수 있었다는 점에서 관찰할 수

있다. 즉 영화 리뷰 분류 등의 과제에 대해서 보다 유사한 말뭉치인 리뷰 데이터 등으로 학습된 모형이 주로 소설 등의 텍스트를 포함하고 있는 말뭉치보다 효과적이었다는 것이다. 이는 또한 n-그램의 활용이 자연어 처리 과제에서 유의미한 추가적 정보를 제공한다는 것을 확인하는 사례이기도 하다. Word2vec과 같은 단어 공출현(Cooccurrence)에 기반한 단어 임베딩 모형들은 일반적으로 단어와 단어의 반의어를 서로 유사하게 임베딩하는 경향이 있다. 이는 단어와 단어의 반의어가 서로 유사한 맥락에서 출현하는 경향이 높기 때문이다(Nguyen, Walde, & Vu, 2016). 즉 '좋은'과 '나쁜'은 서로 유사한 맥락에서 나타날 수 있기 때문에 서로 유사도가 높은 벡터 표현으로 학습되게 된다. 유니그램을 넘어선 n-그램의 사용은 표 7에서 나타나듯 단어의 의미를 보다 구체화하여 이러한 문제를 해소할 수 있다.

본 연구에서는 컨볼루션 신경망을 활용하였으나 시퀀스 데이터에 대해 흔히 적용되는 순환 신경망을 인코더로 활용하는 경우 등에 대해서는 검토하지 않았다. 또한 인코더 신경망이 잠재적으로 가질 수 있는 표현력을 충분히 활용할 수 있는 형태의 디코더와 손실 함수에 대해서도 제한적인 탐색을 수행하였다. 또한 조성적 의미가 n-그램과 같이 바로 인접한 단어들 사이에서만 발생하는 것은 아니며 서로 떨어진 단어들 사이에서도 발생할 수 있다는 점을 고려하였을 때 본 연구에서 실험된 인코더와 디코더 모형으로는 이러한 잠재적인 조성적 의미의 파악을 위해서는 충분하지 않은 점이 존재한다.

제 6 장 결론 및 향후 연구를 위한 제언

본 연구에서는 텍스트 임베딩을 학습하고 생성하기 위한 인코더와 디코더 구조들을 탐색하였으며 단순하며 고속 처리가 용이한 방법인 단어 임베딩의 평균과 그에 적절한 순환 신경망 언어 모형 디코더를 제안하였다. 단순한 형태의 단어 임베딩의 평균이 컨볼루션 신경망 및 자기 주의 메커니즘을 통해 모형의 표현력을 증대시킨 경우와 비교했을 때에도 효과적이라는 것이 나타났다. 이는 단어 자루 모형이 많은 자연어 처리 과제에서 효과적이라는 것이 시사하듯 텍스트의 맥락적 정보가 상당 부분 순서와는 관계 없이 텍스트에 등장한 단어들의 유형에 반영되어 있음을 시사한다. 또한 임베딩 평균 모형에는 n-그램 등의 토큰을 추가하여 단어들이 이루는 조성적 의미를 단순하면서도 효과적으로 보강할 수 있다(Pagliardini et al., 2017).

컨볼루션 신경망의 사용은 단순한 임베딩의 평균만큼의 효과성을 보여주지 못했다. 단어의 분산 표현을 통해 컨볼루션 신경망은 바이그램 혹은 트라이그램을 넘어 4-그램, 5-그램에서 발생하는 단어의 조성적 의미를 파악할 수 있는 가능성을 가지고 있다. 그러나 단어의 조합이 가지는 다채로운 의미적 가능성을 충분히 포착하기 위해서는 높은 수준의 표현력과 적절한 학습 목표가 요구되며, 따라서 단어의 의미를 충분히 효과적으로 추출하기에는 컨볼루션 신경망과 디코더의 특성이 제한적이었기 때문일 수 있다고 생각된다.

또한 본 연구에서 나타난 실험 결과는 단어의 출현 여부와 함께 텍스트 전체의 전반적인 특성의 반영 여부도 텍스트 임베딩의 성능에 큰 영향을 미친다는 것을 시사한다. 이는 컨볼루션 신경망에서 전역 최대값

풀링과 전역 평균 풀링을 사용한 경우의 성능 차이에서 드러난다. 전역 최대값 풀링은 국소적으로 두드러지게 검출된 특징들을 포착하므로 비교적 국소적 특징들로 텍스트 임베딩이 구성된다고 볼 수 있고, 전역 평균 풀링은 전반적으로 검출된 특징들을 포착하므로 전역적인 특징들로 텍스트 임베딩이 구성된다고 볼 수 있다. 이 점에서도 단어 임베딩의 평균은 강점을 갖는다. 단어 임베딩의 평균은 단어들의 순서를 무시하고 모든 단어를 동일하게 취급하므로 기본적으로 평균적이고 전반적인 특징을 반영하게 된다고 볼 수 있다.

이후 연구에서는 자연어와 같은 시퀀스 데이터 처리에 널리 사용되는 순환 신경망을 인코더로 사용하는 방안을 탐구할 수 있다(Mikolov, Karafiát, Burget, Cernocký, & Khudanpur, 2010). 그러나 컨볼루션 신경망의 실험 결과가 시사하는 것처럼 단어의 조성적 의미를 반영하기 위해서는 높은 수준의 표현력이 필요할 수 있으며 텍스트의 전반적 의미를 반영할 수 있어야 할 것이다. 이는 LSTM 등의 장거리 관계를 학습하기 위해 설계된 순환 신경망의 경우에도 어려운 과제일 수 있다(Lin et al., 2017).

단어 임베딩이 텍스트의 임베딩에도 효과적이라는 것을 고려했을 때 단어의 의미를 반영하는 것 뿐만 아니라 텍스트 임베딩에 사용되었을 때에도 좋은 성능을 보일 수 있는 단어 임베딩 모형을 개발하는 것은 단어 임베딩 모형의 유의미한 미래 연구 방향이 될 수 있을 것이다. 본 논문에서는 텍스트 임베딩에 활용될 수 있는 방식으로 단어 임베딩을 학습하는 하나의 방법을 제안하였다. 이후 단어 임베딩 모형의 발전들이 결합된다면 단어 임베딩과 텍스트 임베딩에 모두 효과적인 모형으로의 연구가 진전될 수 있을 것이다.

또한 단어의 조성적 구조에서 어떠한 방식으로 텍스트의 맥락적 정보가 형성되는가를 탐구하는 것은 텍스트 임베딩과 자연어 처리를 위한 중요한 연구 과제라고 할 수 있다. 4-그램, 5-그램 등의 고차 n-그램을 통해 단어의 조성적 의미를 충분히 반영할 수 있는지, 또는 텍스트 내의 서로 다른 위치의 단어들의 조합이 형성하는 의미가 자연어 처리 과제에서 중요한가를 탐색하는 것이 미래의 연구 방향이 될 수 있을 것이다. 본 연구에서는 잠재적으로 n-그램에서 발생하는 의미적 정보를 추출할 수 있는 인코더를 실험하였으나 그러한 인코더를 통해 추가적인 성능을 획득할 수는 없었으며, 장기적 관계를 제한적으로 반영할 수 있는 모형인 자기 주의 모형에서도 추가적인 이점을 얻을 수는 없었다. 이는 인코더 모형의 제약과 함께 인코더를 학습하는 과정에서 사용되는 디코더 모형의 특성이 원인인 것으로 추정된다.

따라서 텍스트의 복잡한 조성적 의미를 추출해낼 수 있는 인코더를 학습하기 위해서는 그러한 인코더의 표현력을 활용할 수 있는 디코더를 설계하는 것이 필요하다. 따라서 이와 같은 디코더의 구조를 탐색하는 것도 중요한 연구 주제가 될 것이다. 언어 모형이 어떠한 방식으로 구성되었을 때 인코더가 텍스트에서 적절한 의미 및 맥락 정보를 추출하도록 학습할 수 있는지, 혹은 언어 모형 이외의 디코더가 텍스트 임베딩에 적절하게 사용될 수 있는지를 탐색할 필요가 있다. 또한 텍스트 임베딩 과제에 적절한 디코더는 실용적으로 널리 사용되고 있는 인코더-디코더 구조에서 효과적인 디코더를 설계하는 것에 도움이 될 수 있을 것이다(Cho, van Merrienboer, et al., 2014).

참고문헌

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., ⋯ Zheng, X. (2016). Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*. https://doi.org/10.1038/nn.3331

Arora, S., Li, Y., Liang, Y., Ma, T., & Risteski, A. (2016). A latent variable model approach to pmi-based word embeddings. *Transactions of the Association for Computational Linguistics*, *4*, 385–399. Retrieved from https://transacl.org/ojs/index.php/tacl/article/viewFile/742/204

Arora, S., Liang, Y., & Ma, T. (2017). A simple but tough-to-beat baseline for sentence embeddings. *International Conference on Learning Representations*, 1–16.

Bengio, Y., Courville, A., & Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, *35*(8), 1798–1828.

Bengio, Y., Ducharme, R., Vincent, P., & Jauvin, C. (2003). A neural probabilistic language model. *Journal of machine learning research*, *3*(Feb), 1137–1155.

Bird, S., Klein, E., & Loper, E. (2009). *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc."

Chen, X., Fang, H., Lin, T.-Y., Vedantam, R., Gupta, S., Dollar, P., & Zitnick, C. L. (2015). Microsoft COCO captions: Data collection and evaluation server. *arXiv preprint arXiv:1504.00325*. Retrieved from http://arxiv.org/abs/1504.00325

Cho, K., van Merrienboer, B., Bahdanau, D., & Bengio, Y. (2014). On the properties of neural machine translation: Encoder-decoder approaches. *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, 103–111.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*. Retrieved from https://arxiv.org/abs/1406.1078

Conneau, A., Kiela, D., Schwenk, H., Barrault, L., & Bordes, A. (2017). Supervised learning of universal sentence representations from natural language inference data. *arXiv preprint arXiv:1705.02364*. Retrieved from http://arxiv.org/abs/1705.02364

Dai, A. M., & Le, Q. V. (2015). Semi-supervised sequence learning. In *Advances in Neural Information Processing Systems* (pp. 3079–3087).

Dzmitry Bahdana, Bahdanau, D., Cho, K., & Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. *International Conference on Learning Representations*, 1–15.

Graves, A. (2012). Supervised sequence labelling. In *Supervised Sequence Labelling with Recurrent Neural Networks* (pp. 5–13). Springer.

Harris, Z. S. (1954). Distributional structure. *Word*, *10*(2–3), 146–162.

He, R., & McAuley, J. (2016). Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. *Proceedings of the 25th International Conference on World Wide Web*, 507–517. https://doi.org/10.1145/2872427.2883037

Hill, F., Cho, K., & Korhonen, A. (2016). Learning distributed representations of sentences from unlabelled data. *arXiv preprint arXiv:1602.03483*. Retrieved from https://arxiv.org/abs/1602.03483

Hill, F., Cho, K., Korhonen, A., & Bengio, Y. (2015). Learning to understand phrases by embedding the dictionary. *arXiv*

preprint arXiv:1504.00548. Retrieved from https://arxiv.org/abs/1504.00548

Hinton, G. E. (1986). Learning distributed representations of concepts. In *Proceedings of the eighth annual conference of the cognitive science society* (Vol. 1, p. 12). Amherst, MA.

Hu, M., & Liu, B. (2004). Mining and summarizing customer reviews. *Proceedings of the 2004 ACM SIGKDD international conference on Knowledge discovery and data mining*, 168. https://doi.org/10.1145/1014052.1014073

Kalchbrenner, N., Grefenstette, E., & Blunsom, P. (2014). A convolutional neural network for modelling sentences. Retrieved from https://arxiv.org/abs/1404.2188

Kim, Y. (2014). Convolutional neural networks for sentence classification. Retrieved from https://arxiv.org/abs/1408.5882

Kingma, D. P., & Ba, J. L. (2015). Adam: A method for stochastic optimization. *International Conference on Learning Representations 2015*, 1–15.

Kiros, R., Zhu, Y., Salakhutdinov, R. R., Zemel, R., Urtasun, R., Torralba, A., & Fidler, S. (2015). Skip-thought vectors. In *Advances in neural information processing systems* (pp. 3294–3302).

Le, H. T., Cerisara, C., & Denis, A. (2017). Do Convolutional

Networks need to be Deep for Text Classification? *arXiv preprint arXiv:1707.04108*. Retrieved from http://arxiv.org/abs/1707.04108

Le, Q., & Mikolov, T. (2014). Distributed representations of sentences and documents. *International Conference on Machine Learning - ICML 2014*, *32*, 1188–1196. https://doi.org/10.1145/2740908.2742760

Lin, Z., Feng, M., Santos, C. N. dos, Yu, M., Xiang, B., Zhou, B., & Bengio, Y. (2017). A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130*. Retrieved from https://arxiv.org/abs/1703.03130

Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., & Potts, C. (2011). Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1* (pp. 142–150). Association for Computational Linguistics.

Maaten, L. van der, & Hinton, G. (2008). Visualizing data using t-SNE. *Journal of Machine Learning Research*, *9*(Nov), 2579–2605.

Merity, S., Xiong, C., Bradbury, J., & Socher, R. (2016). Pointer sentinel mixture model. *arXiv preprint arXiv:1609.07843*. Retrieved from http://arxiv.org/abs/1609.07843

Mesnil, G., Mikolov, T., Ranzato, M., & Bengio, Y. (2014). Ensemble of generative and discriminative techniques for sentiment analysis of movie reviews. *arXiv preprint arXiv:1412.5335*. Retrieved from http://arxiv.org/abs/1412.5335

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*. Retrieved from https://arxiv.org/abs/1301.3781

Mikolov, T., Karafiát, M., Burget, L., Cernocký, J., & Khudanpur, S. (2010). Recurrent neural network based language model. In *Interspeech* (Vol. 2, p. 3).

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

Nguyen, K. A., Walde, S. S. im, & Vu, N. T. (2016). Integrating distributional lexical contrast into word embeddings for antonym-synonym distinction. *arXiv preprint arXiv:1605.07766*. Retrieved from http://arxiv.org/abs/1605.07766

Oord, A. van den, Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., … Kavukcuoglu, K. (2016). Wavenet: A generative model for raw audio. *CoRR abs/1609.03499*, *21*(9), 1120–1124. https://doi.org/10.1109/LSP.2014.2325781

Pagliardini, M., Gupta, P., & Jaggi, M. (2017). Unsupervised learning of sentence embeddings using compositional n-gram features. *arXiv preprint arXiv:1703.02507*. Retrieved from http://arxiv.org/abs/1703.02507

Pang, B., & Lee, L. (2004). A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd annual meeting on Association for Computational Linguistics* (p. 271). Association for Computational Linguistics.

Pang, B., & Lee, L. (2005). Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd annual meeting on association for computational linguistics* (pp. 115–124). Association for Computational Linguistics.

Paragraph. (2017). Merriam-Webster Online.

Pascanu, R., Mikolov, T., & Bengio, Y. (2013). On the difficulty of training recurrent neural networks. In *International Conference on Machine Learning* (Vol. 5, pp. 1310–1318). https://doi.org/10.1109/72.279181

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 (EMNLP)* (Vol. 14, pp. 1532–1543).

Radford, A., Jozefowicz, R., & Sutskever, I. (2017). Learning to generate reviews and discovering sentiment. *arXiv preprint arXiv:1704.01444*. Retrieved from http://arxiv.org/abs/1704.01444

Roth, D., & Li, X. (2002). Learning question classifiers. *Proceedings of the 19th International Conference on Computational Linguistics*, *1*, 1–7.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, *323*(6088), 533–536. https://doi.org/10.1038/323533a0

Socher, R., Perelygin, A., & Wu, J. (2013). Recursive deep models for semantic compositionality over a sentiment treebank. *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, 1631–1642. https://doi.org/10.1371/journal.pone.0073791

Socher, R., Perelygin, A., Wu, J. Y., Chuang, J., Manning, C. D., Ng, A. Y., & Potts, C. (2013). Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP)* (Vol. 1631, p. 1642). Citeseer.

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in neural information processing systems* (pp. 3104–3112).

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., … Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems* (pp. 5998–6008). Retrieved from http://arxiv.org/abs/1706.03762

Wang, S., & Manning, C. (2012). Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2* (pp. 90–94).

Wiebe, J., Wilson, T., & Cardie, C. (2005). Annotating expressions of opinions and emotions in language. *Language Resources and Evaluation*, *39*(2–3), 165–210. https://doi.org/10.1007/s10579-005-7880-9

Wu, Y., Schuster, M., Chen, Z., Le, Q. V, Norouzi, M., Macherey, W., … Macherey, K. (2016). Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*. Retrieved from https://arxiv.org/abs/1609.08144

Zhu, Y., Kiros, R., Zemel, R., Salakhutdinov, R., Urtasun, R., Torralba, A., & Fidler, S. (2015). Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE International Conference on Computer Vision* (pp. 19–27).

Abstract

On Text Embedding Model Using Word Embeddings

Kim Seonghyeon

Digital Contents and Information Studies

The Graduate School

Seoul National University

Sentence or paragraph embedding, which is a method of converting variable-length text into a vector representing contextual information of a text, is a method which is used in various machine learning systems such as text classification problems like emotional analysis, text similarity measurement, clustering, and visualizations. And text embedding is used as a basic feature extraction method for performing various tasks

requires vectors to be have predefined dimensions as an input.

The Bag-of-Words model, which has been widely used for the vector representation of existing texts, is simple and effective, but the size of the dimension increases in proportional to the number of words and it is difficult to utilize unlabeled text data. To overcome these limitations, the models like Paragraph Vector that utilizes distributed representation of the texts proposed. But it has a limitation that it requires an estimation process in which an existing model should be retrained to learn and generate a vector representation of new data. The Skip-Thought Vectors model, which learns encoders that generate vector representations of sentences using sentence-to-sentence relations based on Sequence-to-Sequence model. It has a disadvantage that it utilizes inter-sentential relationships of sentences so it is difficult to apply immediately to a paragraph level where it is difficult to establish a mutual relationships and requires a lot of operation resources to generate text embedding.

In this study, we propose a model that generates text embedding through sum of word embedding considering the problem of transfer learning, the computation amount required for embedding generation, and the ease of variable length text processing. Unlike traditional word embedding using Unigram, our model improves text embedding by using n-grams such as Bigram and Trigram. In addition, it is compared with the model using the convolutional neural network, and it is examined that it can show good

performance characteristics without introduction of additional structure such as convolutional neural network.

This study presents a simple method for generating a vector representation of a sentence or paragraph, while analyzing the characteristics of n - gram word embedding, which is learned incidentally. The effectiveness of the simple combination of word-embedding and analysis of the characteristics of word-embedding can be used in practical situations where high-speed processing of text is required, as well as understanding the conditions necessary to capture the characteristics of text required for natural language processing tasks.

.....

Keywords : word embedding, paragraph embedding, n-gram, natural language processing

Student Number : 2016-26029