Word2Vec에 대한 얕은 조사

정재영

1. Word Embedding

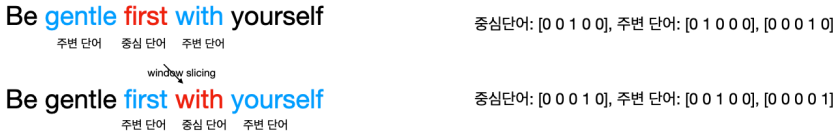
word2vec를 공부하기에 앞서, 워드 임베딩에 대해 간략히 알아보자. 우리 인간이 사용하는 자연 언어를 컴퓨터가 이해하기 위해서는, 유저가 우리 언어를 컴퓨터가 이해할 수 있도록 적절히 변환해줄 필요가 있다. 여기서 유저가 어떻게 우리 언어를 변환해주는지에 따라 컴퓨터의 자연어 처리 능력이 크게 달라진다. 이 변환 방법 중 하나인 워드 임베딩은 단어 word를 특정 크기의 벡터 및 행렬로 표현하는 방법이다. 앞에서[1] 우리는 **원핫-인코딩**은 단어를 (단어의 인덱스 값만 1이고 나머지는 0인) 희소 sparse 벡터로 표현하는 방법인 반면에, **분산 표현**은 단어를 밀집 dense 벡터로 표현하는 방법이라는 것을 알아보았다. 또한 우리는 전자의 방법은 (1) 차원의 저주에 곤경에 빠지고 (2) 단어들 간의 유사도를 비교할 수 없다는 한계를 가지는 반면에, 후자의 방법은 각 단어를 여러 word feature real numbers와 연관시키기 때문에 앞의 곤경들을 다소 해소한다는 이점을 갖는다는 것도 알아보았다. 워드 임베딩은 후자의 방법처럼 단어를 밀집 벡터로 표현하는 방법을 가리킨다. 이것의 방법론으로는 LSA, Word2vec, FastText, Glove 등이 있다.

2. Word2Vec

Word2Vec에는 2 가지 방식이 있다: CBOW (Continuous Bag of Words)와 Skip-Gram이 그것이다. 전자는 “주변에 있는 단어들을 가지고, 중간에 있는 단어를 예측하는 방법”인 반면에, 후자는 반대로 “중간에 있는 단어로 주변 단어를 예측하는 방법”이다. 먼저, CBOW에 대해 알아보자.

2.1 CBOW

다음 예문을 고려해보자: “Be gentle first with yourself”. 이 문장을 스플릿해주어 {“Be”, “gentle”, “first”, “with”, “yourself”}이 주어졌다고 해보자. CBOW를 가지고 우리가 수행할 일은 앞의 set 중 {“Be”, “gentle”, “with”, “yourself”}으로부터 “first” 를 예측하는 것이다. 그러면, CBOW 방법론에 따를 때 “first”는 중심 단어 center word이고, 나머지 앞의 set 내의 단어들은 주변 단어 context word라고 간주된다. 아래 예를 보자.



window = 1인 경우

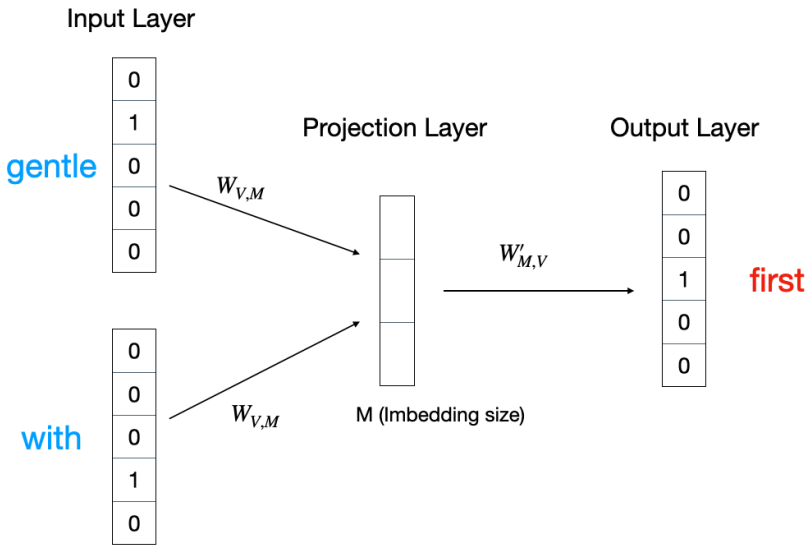
CBOW에서 중심 단어 주변에 있는 단어를 앞, 뒤로 각각 몇 개의 단어를 참고할지를 나타내주는 것을 **윈도우** window라고 한다. 그리고 위 그림과 같이 중심 단어와 주변 단어를 오른쪽으로 한 칸씩 이동하며 학습을 위한 데이터 셋을 만드는 것을 **윈도우 슬라이싱**이라고 일컫는다.

---

[1] https://github.com/Intelligence-Engineering-LAB-KU/Seminar/blob/master/summer_2020/0825_NPLM(2003).ipynb

[1] CBOW의 과정:

(1) 먼저 입력값으로서 주변 단어들이 원핫 인코딩 된다.
(2) 그 뒤에 단어가 임베딩된 뒤에 만들어지는 **투사층 projection layer**가 있다.[2]
(3) 마지막으로, 출력값으로는 원핫 인코딩된 중심 단어가 나온다.



위 그림에서 주의할 점은 가중치 W, W' 은 서로 전치 행렬 관계가 아니라는 것이다. 이들은 단지 서로 다른 랜덤 가중치일 뿐이다.

[2] 입력층 => 투사층

입력층에서 투사층으로 입력값들이 가중치를 곱해서 사상될 때, 투사층에 입력되는 값은 다음과 같다: 이 예에서는 윈도우 n = 1 이기 때문에, 각 투사층에 입력되는 값의 개수는 2n 으로서 2 개이다. 즉, 투사층의 하나의 값은 $v = \frac{v_{gentle} + v_{with}}{2n}$ 이다.[3]

[3] 투사층 => 출력층

투사층의 벡터는 W'와 곱해짐으로써 다시 입력값의 크기와 동일한 벡터가 된다. 그런데 우리는 이것을 다시 원핫 벡터 형태로 만들어주기 위해, softmax를 사용한다.

[4] 역전파를 수행하여 W, W'를 학습시킨다.

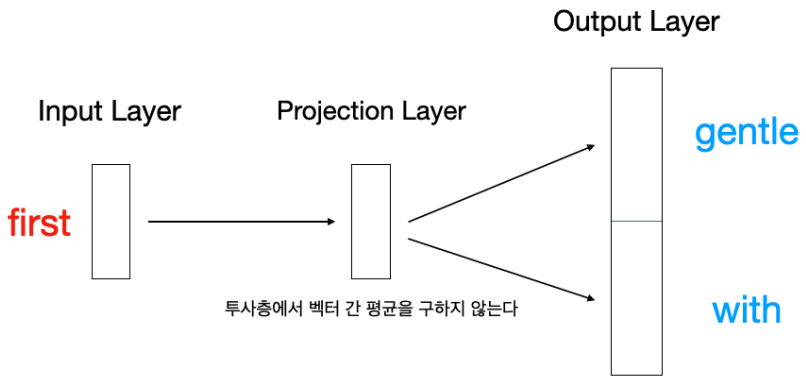
---

[2] 이것은 은닉층과 구분되는데, 그 이유는 word2vec에서 이 투사층은 오직 하나이기 때문에 심층 신경망이 아닌 얕은 신경망 shallow NN이기 때문이고, 또한 word2vec에서는 활성화 함수가 존재하지 않고 대신 lookup table 연산을 담당하기 때문이다. 이러한 이유 때문에 은닉층과 구분하고자 투사층이라는 용어를 사용한다.

[3] 뒤에서 알아볼 skip-gram 방법에서는 입력되는 중심 단어가 하나이기 때문에 투사층에서 벡터의 평균을 구하지 않는다.

2.2 Skip - Gram

Skip-Gram의 메커니즘은 CBOW의 그것에 반대되는 형태를 지닌다. 그런데 많은 논문에서는 CBOW 보다 Skip-Gram이 더 성능이 좋다고 평가한다. 이 스킵그램의 메커니즘은 아래 같이 도식화될 수 있다:



3. NNLM과 Word2Vec 간의 차이

	NNLM	**Word2Vec**
학습 속도	비교적 느림	비교적 빠름
예측 대상	다음에 나올 단어	중심 단어
구조	은닉층	1. 활성화 함수가 있는 은닉층 제거하고, **투사층 사용** 2. 계층적 소프트맥스 Hierarchical Softmax 사용 3. 네거티브 샘플링 Negative Sampling 사용
연산량	nm+nmh+hV	nm+mlog(V)

4. Word2Vec에서 연산량을 줄이는 2 가지 전략

4.1 Hierarchical Softmax
4.2 Negative Sampling

References

[1] https://wikidocs.net/22660 내용과 그림 모두 참고.
[2]https://shuuki4.wordpress.com/2016/01/27/word2vec-%EA%B4%80%EB%A0%A8-%EC%9D%B4%EB%A1%A0-%EC%A0%95%EB%A6%AC/