3. 자연어(Nautral Language)란 인간이 일상에서 사용하는 언어를 말합니다.  자연어 처리(Natural Laguage Processing)는 기계가 자연어를 이해하고 해석하여 처리할 수 있도록 하는 일 을 말합니다. 자연어 처리 (Natural Laguage Processing) 는 줄여서 NLP라고 부릅니다.

기계가 자연어를 이해하기 위해선 컴퓨터가 이해할 수 있도록 텍스트를 적절히 숫자로 변환해야 합니다.

4

그러기 위해서   단어를 인공 신경망 학습을 통해 벡터화하는 워드 임베딩이라는 방법이 가장 많이 사용되고 있습니다. 워드 임베딩은 단어를 벡터로 표현하는 방법으로, 그 중에서도 밀집표현으로 변환하는 방법을 말합니다.

5

이를 이해하기위해선 밀집 표현과 희소표현에 대해 알아야하는데, 먼저 희소표현은 문장이 있으면 표현하고자 하는 단어의 인덱스만 1이고 나머지 단어들은 전부 0으로 표현되는 방법입니다. 이렇게 벡터또는 행렬의 값이 대부분 0으로 표현되는 방법을 희소 표현이라고 합니다. 희소표현은 문장에 단어가 10000개면, 해당 단어의 인덱스 부분만 1이고 나머진 0으로 표시가 되기에 벡터의 차원은 10000이여만 합니다. 이러한 벡터 표현은 공간적 낭비를 불러 일으킬 뿐더러, 단어의 의미를 표현하지 못한다는 단점이 있습니다.

하지만 이런 희소표현과 반대되는 밀집표현이 있는데, 밀집표현은 희소표현처럼 벡터의 차원을 단어집합 크기로 상정하지 않고, 사용자가 직접 설정한 값으로 모든 단어의 벡터표현의 차원을 맞춥니다. 따라서 더이상 인덱스가 0과 1만 가진 값이 아니라 실수값을 가지게 됩니다. 예를들어 단어 10000개가 있는데, 사용자가 차원을 128로 설정한다면 모든 단어의 벡터표현이 128로 바뀌며 모든 인덱스 값들이 실수가 됩니다. 이런경우를 벡터의 차원이 조밀해졌다고 하여 밀집 벡터라고 합니다.

6.이

이런 워드 임베딩의 방법론중 하나로 Word2Vec이 있습니다 구글에서 개발한 Word2Vec은 단어 유사도를 반영할 수 있도록 단어의 의미를 벡터화 할 수 있는 방법입니다. word2vec의 주요 아이디어는 “비슷한 분포를 가진 단어라면 비슷한 의미를 가질 것 “ 입니다. 즉 자주같이 등장할수록 두 단어는 비슷한 의미를 가짐을 의미합니다.

7. 보시다시피 word2vec은 input, hidden ,output layer총 3개의 층으로만 이루어져 학습이 빠르고 많은 단어를 학습할 수 있어 성능이 좋습니다. 또한 CBOW와 SKIP-GRAM 두가지 방식이 있는데, CBOW란 주변에 있는 단어들을 가지고 중간에 있는 단어를 예측하는 방법이고, SKIP GRAM은 중간에 있는 단어로 주변에 있는 단어들을 예측하는 방법입니다.

8.CBOW에서 윈도우는 중심단어를 기준으로 주변단어를 몇 개의 단어를 볼지 결정하는 범위 입니다. 옆에 표를 보시면 THE FAT CAT SAT ON THE MAT이라는 문장에서 중심단어가 SAT이고 윈도우 크기가 2라면 FAT,CAT,ON,THE가 주변단어가 되는 것입니다.

9.그림은 CBOW의 도식화입니다.

도식화를 간단히 설명하자면, 입력층(INPUT)에는 사용자가 정한 윈도우 크기만큼 주변 단어들이 들어가는데, 모두 원 핫 벡터 즉 희소벡터로 들어가게 됩니다. 투사층(PROJECTION)은 M이고 5개인것을 볼 수 있는데 CBOW에서 투사층의 크기는 임베딩하고 난 벡터의 차원이 됩니다. 위의 그림에서 투사층의 크기가 5인 것을 알 수 있습니다.

  입력층과 투사층 사이의 가중치 W는 V × M 행렬이며, 투사층에서 출력층사이의 가중치 W'는 M × V 행렬이라는 점입니다. 여기서 V는 단어 집합의 크기를 의미합니다. 즉, 위의 그림처럼 원-핫 벡터의 차원이 7이고, M은 5라면 가중치 W는 7 × 5 행렬이고, W'는 5 × 7 행렬이 될 것입니다.  W와 W'가중치는 랜덤한 값을 가지게 되는데 CBOW는 주변 단어로 중심단어를 더 정확히 맞추기 위해 계속해서 이 가중치 W들을 학습해나가는 구조입니다.

좀더 확대하여, 동작 메커니즘에 대해 알아보도록 하겠습니다.

10.

앞서 말했듯 입력층은 주변단어들의 희소벡터 즉 원핫 벡터들로 이루어져있고 , 이 벡터들과, 가중치와 곱하게 되는데, 이과정에서, 입력층의 벡터는 해당 인덱스는 1이고 나머지는 다 0이기에 가중치 행렬과 곱하게 되면, 해당 행을 그대로 가져오게 됩니다. 이작업을 LOOKUP TABLE이라고 합니다.

11.주변단어들을 모두 LOOKUP TABLE작업을 거친후 평균을 구하게 되는데, 그림으로 예를 들면 4개의 주변단어 원핫 벡터를 모두 더한 후 4로 나눠주면 평균이 나오게 됩니다.

12. 이해에 도움이 되기 위해 그림을 보시면 0번인덱스 1번인덱스 2번인덱스 값들끼리 더한후 4로 나눠 평균값을 구한것을 볼 수 있습니다.

13. 그 이후 이 평균벡터와 가중치 W'와 곱하게 됩니다. 평균벡터는 크기가 M이고 W'은 M x V이므로 결과는 다시 V 즉 주변단어의 원 핫 벡터 크기만큼 됩니다.

14. 평균벡터와 가중치를 곱한값(Z)를 SOFTMAX함수에 넣어주게 되는데, 이로인하여 출력값은 원소의 총 합니 1이되는 벡터입니다.

각인덱스의 숫자는 해당 인덱스가 중심 단어가 될 확률을 나타내게 됩니다.

15. SKIP-GRAM 은 중심 단어에 대해서 주변 단어를 예측하므로 투사층에서 벡터들의 평균을 구하는 과정은 없습니다. 또한 여러 논문에서 성능 비교를 진행했을 때 전반적으로 Skip-gram이 CBOW보다 성능이 좋다고 알려져 있습니다.

16.

이제 한국 기사 댓글 50만개를 가져와 WORD2VEC을 이용한 워드임베딩하여 연관된 단어를 찾아내는 실습을 설명하겠습니다.

파이썬의 gensim 패키지에는 Word2Vec을 지원하고 있어 설치를 하고, 기사 댓글들이 있는 깃허브 URL을 읽는 모듈과, 한글 형태소를 분석하는 0Kt를 설치한 후

17.

이제 임베딩을 하기 위해선 문장으로 되어있는 한글을 전처리를 할 것인데, 기사댓글에 비어있는 댓글들을 지우고, 한국어를 제외한 언어를 제거하고, 불용어를 제거하는 과정으로 전처리를 할 것입니다.

18.

null값이 존재하는것을 확인후 85개를 지운 모습입니다

19.

수업때 배운 정규표현식으로 한국어 이외의 문자를 삭제한 모습입니다. 원래는 온점이라던가 반점등이 사라진 모습입니다.

20.

언어 분석시 의미가 있는 단어와 의미가 없는 단어나 조사 등이 있다. 그중에서 의미가 없는 것을 stopwords즉 불용어 라고 합니다. 데이터 분석을 하는 것에 있어서는 큰 도움이 되지 않는 단어들이기 때문에 전처리 과정에서 이를 제거하여 사용하고 있습니다. 또한 토큰화를 진행하는데 0kt함수를 쓰는 모습입니다. 토큰화는 쉽게말해 문장에 띄어쓰기를 기준으로 모든단어들을 다 분리해서 저 tokenized\_data에 append하는 과정입니다. 그와중 9행에 okt는 여러가지 함수가 있는데 그 중 morphs는 텍스트를 형태소 기준으로 분리하는 함수입니다, 그이후 불용어를 제거하고 리스트에 추가하게되면 토큰화가 끝이납니다.

21.

간단하게 그래프로 댓글에길이와 개수에관해 그래프로 표시하였습니다.

22.

이제 word2vec을 이용할 차례인데 간단하게 word2vec의 하이퍼 파라미터에 대해 설명하겠습니다.

23.

24. 이런식으로 모델에 most\_similar 함수를쓰면 연관된 토큰들이 나오는것을 확인할수 있습니다.

25. 이 사이트는 시각화 사이트인데, 앞의 작업물을 메타데이터와 텐서 데이터파일들을 tsv로 받고 이 사이트에서 넣으면 이런식으로 29000개의 단어들이 시각화되는 것을 볼 수 있는데 가까울수록 서로 연관된 단어인 것을 알 수 있는데, 현재 단어가 너무 많아 서로 얼마나 가까운지 알 수 없다는 아쉬움이 있습니다.

26.

이런 word2vec의 한계점은

동음이의어에대해 항상 같은 벡터값을 반환한다는 사실이 있었습니다. 예문으로 배에 탔더니 배가아프다라 했을때 배를 동일 벡터값으로 취급한다는 문제가 있었고,

또 속도가 느리다는 단점이 있는데 이또한 소프트맥스 함수를 사용해서 집합크기 벡터 내의 모든값을 바꿔야하는데, 데이터셋이 커질수록 많은 연산량으로 인하여 속도, 메모리측면에서 비효율적일 수 있다는 사실이 있습니다.

이런점들이 보완된 bert가 있다~는 식으로 연결해서 진행