6. 잠재 의미 분석 – Latent Semantic Analysis (LSA)

TF-IDF와 Bag of Words는 단어 기반의 유사도이기 때문이다.

Pizza, hamburger => 유사도 0 = pizza, ramen

A = 행렬 => 행은 단어, 열은 문장으로 이루어진 행렬  
=> 이것을 SVD 행렬 분해를 실행

A = U \* 시그마 \* VT (U= 단어가 행으로, topic을 위한 단어 행렬, 시그마 = topic의 강도, VT = topic을 위한 문장(문서) 행렬)

하지만 우리가 관심있는건 문장 벡터 => 시그마 \* VT를 구하면 된다. => topic(단어들중 주제가 되는 단어, 후보로는 시그마 행렬의 대각성분의 단어가 올 수 있다.) 중 가장 높은거 2개만 따져본다.

피자와 햄버거, 피자햄버거 쿠키는 같은 축에 있게 된다.

또한, 라멘, 스시, 라멘스시 라는 문서(문장)은 같은 y축(단어 2, t2)에 오게 된다.

=> 유사도 1

7. word2vec

Text를 숫자로 변환하는 과정 = 인코딩

가장 사랑받는? => one hot encoding.

* 각 단어에 0 부터 숫자를 주기
* 원핫 인코딩으로 [1,0,0], [0,1,0] 으로 주기

하지만 원핫 인코딩에는 유사도가 없다. => 유사도는 유클리디안 이용하는데, 원핫 이용시 모든 단어간의 거리가 같게 된다. 또한 전부 90도라 유사도 발견 못함.

따라서 인코딩 대신에 임베딩을 보통 이용, 원핫 인코딩보다 저차원이고, 유사도를 갖게 된다. 비슷한 단어는 가깝게, 유클리디안(L2 norm) 거리가 가까워진다.

Word2vec skipgram windowsize =1 이용(size =2 면 좌우 2칸씩 관계)

“king brave man”, “queen beautiful woman” => king- brave, brave-king, brave – man …..

Brave는 king과 man을 neighbor로 얻게됨(neighbor가 ylabel로 작동한다.). = word2vec 학습위한 데이터로 작동

* 이제 다시 각 단어들을 원핫처리해서 6->2->6의 뉴럴넷으로 학습함.(softmax + crossentropy loss)
* 학습 끝나면 6->2의 weight가 정해질 것이고 그 output은 word2vec의 임베딩 값으로 사용가능 = 히든레이어값 그 자체가 워드 벡터로 사용된다.

[1,0,0,0,0,0] = king => lookuptable([1,1],[1,2],[1,3],[5,5],[5,6],[5,7])

8. WMD 문서 유사도

Word mover’s distance = 문서 유사도 구하는데 각광, word2vec의 유클리디언 거리 이용이 기반

이미 나와있는 word2vec 이용해도됨(구글뉴스)

심지어 king-man = queen – woman 이라는 값도 나온다.

“prince fearless guy” vs “king brave man” vs “queen beautiful woman”, 1,2번이 가까운 것을 알 수 있다.

먼저 Normalized bows with stopword removal 알아야함.

“the president greets the press in Chicago”

“obama speaks to the media in Illinois”

“the band gave a concert in Japan”

“Obama speaks in Illinois”

=> the, in, to, a 등등 없앰 => bows에서 나온 횟수/총 나온 횟수로 인덱스를 표현 => 그리고 기존 bows와 달리 0들은 다 없애도 된다. 처음 문장 => [0.25,0.25,0.25,0.25] = 단어의 함유량 정도로 볼 수 있다.

D1, d2비교시 이제 Obama – president 유클리디언 거리(가장 가까운놈)에 \* obama의 함유량 곱해줌. => 이것을 모든단어에 대해 적용후 summation

D1, d4 비교시 단어 개수 4vs3으로 다른데 어떡해? 함유량도 0.25vs0.33으로 다름

아까처럼 가장 가까운 단어에 0.25 주고, 그 다음으로 가까운 단어에 나머지인 0.08을 준다.

계산속도 너무 오래걸림(O(p^3logp) => 그나마 빠르게 = Relaxed WMD(O(p^2))) =>성능 비슷(p = 단어 개수)

9. 텐서?

Rank 0->1->2->3->n(n-tensor), n dimension tensor

Scalar = [1] => vector = [1,1] => matrix[[1,1],[1,1]] => 3 tensor = [[[1,1],[1,1]],[[1,1],[1,1]]]

문장 3개

Hi john, hi james, hi brian

* Hi = index 0 = [1,0,0,0] ……. 스칼라가 벡터로

Hi john = 매트릭스로 가능 = [[1,0,0,0],[0,1,0,0]]

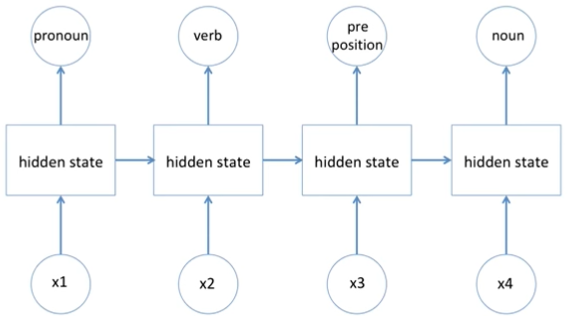
미니배치로 또 가능 = 텐서형태로 가능 = [ M1, M2, M3] …. = 3\*2\*4의 3d 텐서

Gray scale 의 5\*5 이미지 들 => 3\*5\*5 만약 컬러? => 3\*5\*5\*3(rgb 채널수)

5d tensor? 3\*5\*5\*5\*3 => frame뭉치 수 \* 뭉치당 frame 개수 \* 열\*행\* rgb

10. rnn 기초

Pos tagging(품사) 을 위한 rnn? Sequence의 중요성을 강조



Ht = tanh(wxh \* xt + whh+ht-1 + b)

11. LSTM

12. sequence to sequence + attention

13. transformer (attention is all you need)

14. GPT-3(GPT1,2,)

15. ELMo

16. BERT