1. How do we represent the meaning of a word?

Words는 rich meaning을 가짐, 또한 많은 nuance가 존재.

사전적 meaning => word와 phrase로 표현되는 idea / 사람이 words와 signs를 통해 표현하기를 원하는 idea

Linguist(다개국어자)는 지시적 의미? Denotational semantics

2. How do we have usable meaning in a computer?

WordNet = thesaurus 사전(유의어 사전) // 혹은 수직관계 (동물 – 판다) 등의 정보를 알 수 있다.

nltk라는 라이브러리에서 wordnet 사용(그냥 basic하게 사용한다.)

3. problems with resources like WordNet

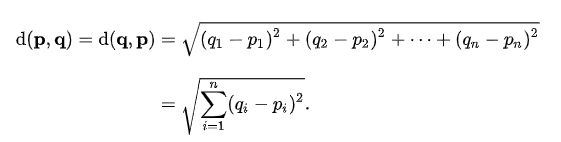
하지만 그렇게 잘 작동하지 않는다..(그저 관계형 DB에서

* 목록에 없는 단어 존재(새로운 단어)
* 단어에 대한 의미(함축(connoation), 뉘앙스(nuance))를 파악하지는 못함
* 모든 단어에 대한 관련어 업데이트가 불가
* 생성, 관리에 지속적인 인력 투입.
* 단어간 유사도 측정 불가

4. Representing words as discrete symbols

Traditional NLP는 대략 2012년까지의 기술들, 2013년부터는 neural net(deep learning)을 통한 representation이 증가.

Localist representation – 옛날 방식, 단어의 그저 discrete한 symbol! => 단어들을 one-hot vector로 표현한다. 하지만 엄청나게 많은 단어, 파생 단어가 있을 것이 분명하고 따라서 벡터가 엄청나게 커진다.

또한, 단어간의 relationship 파악에 어렵다. 모든 단어들이 각 축에 위치해서 코사인 유사도 측정이 불가하다. 모두 orthogonal 또한 L2 distance도(유클리드 거리) 모든 벡터가 같다. 

따라서 one-hot 벡터를 그대로 사용하기는 어려움이 존재 유사성을 담아내는 벡터 표현이 필요해서.

이런 것은 그저 데이터에서 그 단어를 포함하는 document를 골라올뿐…(“seattle motel” 검색 시 이 구 포함하는 문서를 제공)

5. word2vec

Distributional semantics 단어의 의미는 그 단어 근처에 자주 등장하는 단어들을 통해 파악할 수 있음!

단어의 context를 파악해 encoding하기 위해 관찰할 주변 단어의 개수 = fixed-size window, 이제는 그저 localist representation이 아닌 단어의 맥락과 유사도를 가지고 특정 단어를 vector로 표현(word vector) = word embeddings, word representations

사실 단어가 100 차원인데 2차원으로 projection해서 비쥬얼라이징 해주는 것이다.

이 작업을 하는 것이 word2vec(Mikolov 2013) word vector learning

Idea -> 충분한 corpus(문장의 모음, 글, 말뭉치), 모든 word는 fixed vocab에서 vector로 표현한다.

이때 한 확률을 계산해야하는데, P(w-t+j | w-t)이다.<확률 세미콜론 = 파라미터> 또는 P(w-t | w-t+j)도 된다. T는 center word라고 보면되고 j는 좌우로 있는 window size보다 작은 값이다. 따라서 outside context words로 여길수 있다.

확률 값을 최대화하는 방향으로 word vector의 값을 변경

그렇다면 objective function은 어떻게 구하나? Position이 t=1~T라 할때

즉 인풋이 w-t이고 아웃풋이 w-t+j = context word가 현재 단어의 context에서 나타날 확률들의 곱을 단어수, window size 크기에 따라 구한다. => 그러면 그 가능성, 확률의 likelihood가 나온다. 이것을 maximize 하는 것이 목표인데. 얘의 값의 범위는 0~1이니까 -log를 취하고 1쪽으로 간다면 error가 줄어드는 것! => 이거로 gradient 구하고 파라미터 업데이트 가능

V = vocab의 단어의 수 N은 사용자 지정, v = 입력->은닉 층의 파라미터, u = 은닉 -> 출력층의 파라미터 / u-o, v-c

그런데 내적을 하는 이유? 벡터 내적은 cos값으로 나타낼수 있다는 것인데 벡터 내적 값이 클수록 cos의 각도는 좁아지니까 내적 값을 크게 해야댐 = 확률을 크게 해야댐 => exp는 모든 숫자를 양수화 시키고(내적값은 음수가능) 분모는 정규화(normalization) 이용한 softmax

이때 분모값은 낮을수록 좋은데 윈도우 크기 내에 등장하지 않는 단어 유사도는 감소시킴.

-추가공부:softmax

#training a model by optimizing parameters

그냥 GD처럼

U와 V는 처음에는 random vector ( U, V의 두 벡터(행렬)을 이용한다는 것이 keypoint)

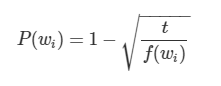
SG로 하지만, CBOW도 있다 => negative smpling은 더 도움 될 것이다.

Minibatch = stochastic

CBOW는 주변단어로 중심단어를 예측, 똑같이 4번 업데이트하는 것이 아닌 1번 업데이트 => 논문으로 => 현재는 SG가 대동단결(학습량 4배차이)

#Subsampling frequent words

If 단어수 10만개, 임베딩 100차원 => 파라미터 2천만개(행렬 성분개수)

자주등장 => 그 파라미터만 계속 업데이트

T= 0.00001 권장 => f = 단어 빈도/ 전체 단어수(V) => P가 0.96이면 96퍼 센트확률로 학습시 제외 (은/는 같은 조사는 학습시킬 필요 없지)

#negetive sampling(이전까지는 naïve softmax)

Exp를 모든단어에? 10만개면 계산량 어쩔까. 내적까지.. 소프트맥스 확률의 분모를 전체 대상 아니고 일부만.. 학습 자체의 스킵은 아니다! = 윈도우 내에 등장하지 않은 단어 5~20개 뽑고 그 사이에 정답단어 1개 추가// 단어가 뽑힐 확률