1) Intro to Natural Language Processing(NLP)

1. NLP

- low-level parsing
    - Tokenization: 주어진 문장을 단어 단위로 쪼개나가는 것
    - stemming: 어미가 다른 단어를 단어의 다양한 의미변화를 없애고 의미만을 보존하는 단어의 어근을 추출하는 것
- Word and phrase level
    - Named entity recognition(NER): 단일 언어나 여러 단어로 이뤄진 고유명사를 인식하는 task
    - part-of-speech(POS) tagging: 단어들이 문장 내에서의 품사나 성분이 무엇인지 알아내는 task
- Sentence level
    - Sentiment analysis: 긍정과 부정 등 감정을 구분
    - machine translation: 문장을 다른 언어로 번역해줌
- Multi-sentence and paragraph level
    - Entailment prediction: 두 문장간의 모순관계를 예측
    - question answering: 질문에 대해 답을 구함
    - dialog systems: 챗봇
    - summarization

2. Text mining

- 뉴스데이터에서 키워드로부터 트렌드를 파악하는 등 텍스트데이터에서 유용한 정보와 인사이트를 뽑아냄
- Document clustering: 서로 다른 주제에 따라 뉴스 데이터 등을 grouping 해줌 (e.g. topic modeling)

- 사회 과학 분야에서 유용하게 사용되는데, SNS 데이터를 바탕으로 유의미한 내용을 얻어내는 방법으로 많이 사용

3. Information retrieval

- 구글이나 네이버 등에서 사용되는 검색기술을 연구하는 분야로, 세부분야로 추천시스템도 포함되어 있습니다.

2) Bag-of-Words

Bag-of-Words 는 텍스트 마이닝 분야에서 딥러닝 기술이 적용되기 이전에 활용되던 단어 및 문서를 숫자로 바꾸는 방법입니다.
먼저 텍스트 데이터에서 중복된 단어를 제거해 unique 한 단어들을 가져와 vocabulary 를 만들어줍니다. 그 다음 unique 한 단어들을 one-hot vector 로 표현해 줍니다. 이때, one-hot vector 의 차원은 vocabulary 의 개수와 동일하게 설정하며 이 경우, 어떤 단어쌍이던 유클리디안 거리가 $\sqrt{2}$ 로 표현되고 코사인 유사도는 0 으로 표현됩니다. 이렇게 구해진 one-hot vector 들을 더해서 문장이나 문서를 나타낼 수 있으며 이를 Bag-of-words vector 라 부릅니다.

NaiveBayes Classifier for Document Classification

NaiveBayes Classifier 는 Bag-of-words vector 로 나타내어진 문서를 분류하는 방법입니다.
d 는 document, c 는 class 라고 했을 때, 아래와 같습니다.

$$c_{MAP} = \underset{c \in C}{\operatorname{argmax}}\ P(c|d)$$

MAP is "maximum a posteriori" = most likely class

$$= \underset{c \in C}{\operatorname{argmax}}\ \frac{P(d|c)P(c)}{P(d)}$$

Bayes Rule

$$= \underset{c \in C}{\operatorname{argmax}}\ P(d|c)P(c)$$

Dropping the denominator

MAP 는 최대 사후 확률을 말하는 것으로 가장 비슷한 클래스를 표현하는 것과 동일하며 가장 위의 식에 Bayes Rule 을 적용하면 두번째 식이 되고, 여기서 P(d) 는 상수이므로 제거해 줍니다.

$$P(d|c)\mathrm{P}(c) = P(w_1, w_2, \ldots, w_n|c)\mathrm{P}(c) \rightarrow \mathrm{P}(c)\prod_{w_i \in W} P(\mathrm{w}_i|c)$$

위의 식에서 P(d|c)는 class 가 고정되었을 때 문서 d 가 나타날 확률이며 문서 d 는 단어 W_1 부터 마지막 단어 W_n 까지 동시에 나타나는 동시사건으로 볼 수 있습니다. 각 단어가 등장할 확률이 c 가 고정되어있는 경우, 서로 독립이라고 가정할 수 있다면 P(d|c)는 각 단어가 나타날 수 있는 확률을 모두 곱한 형태로 나타낼 수 있습니다.

위 식의 예시를 들어보면 다음과 같습니다.

	Doc(d)	Document (words, w)	Class (c)
Training	1	Image recognition uses convolutional neural networks	CV
	2	Transformer can be used for image classification task	CV
	3	Language modeling uses transformer	NLP
	4	Document classification task is language task	NLP
Test	5	Classification task uses transformer	?

여기서 $P(C_{CV})=12$ 이며 $P(C_{NLP})= 12$ 입니다. 이때 각 단어에 대한 조건부 확률을 구하면 다음과 같습니다.

Word	Prob	Word	Prob
$P(w_{\text{"classification"}}\|c_{\text{CV}})$	$\frac{1}{14}$	$P(w_{\text{"classification"}}\|c_{\text{NLP}})$	$\frac{1}{10}$
$P(w_{\text{"task"}}\|c_{\text{CV}})$	$\frac{1}{14}$	$P(w_{\text{"task"}}\|c_{\text{NLP}})$	$\frac{2}{10}$
$P(w_{\text{"uses"}}\|c_{\text{CV}})$	$\frac{1}{14}$	$P(w_{\text{"uses"}}\|c_{\text{NLP}})$	$\frac{1}{10}$
$P(w_{\text{"transformer"}}\|c_{\text{CV}})$	$\frac{1}{14}$	$P(w_{\text{"transformer"}}\|c_{\text{NLP}})$	$\frac{1}{10}$

위의 값들을 활용해 d_5 에 대한 예측값을 구해보면 아래와 같이 나옵니다.

$P(C_{CV}|d_5)=12\times114\times114\times114\times114$

$P(c_{NLP}|d_5)=12\times110\times210\times110\times110$

위의 결과를 보면 $P(C_{NLP}|d_5)$의 값이 더 크기 때문에 NLP 로 예측하게 됩니다. 이때, 만약 d_5 의 단어가 NLP 로 분류된 문장에 포함되어 있지 않으면 해당 단어는 0 의 확률을 갖게 되어 다른 단어가 아무리 연관성이 높아도 해당 클래스로 분류할 수 없습니다. 이를 막기 위해 Regularization(상수를 더해주는 방법)이 필요합니다.

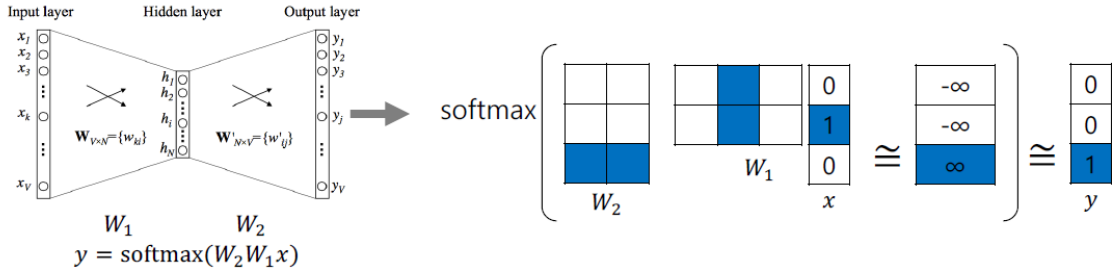
Word Embedding

1) Word Embedding 이란?

Word Embedding 이란 단어들을 특정한 차원으로 이루어진 공간상의 한 점, 혹은 그 점의 좌표를 나타내는 vector 로 변환해주는 기법입니다. 비슷한 의미를 가지는 단어가 좌표 공간상에 비슷한 위치의 점으로 매핑되도록 함으로써 단어들의 의미상 유사도를 잘 반영한 벡터 표현을 자연어처리 알고리즘에 제공해주는 역할을 합니다.

2) Word2Vec

Word2Vec 은 같은 문장에서 나타난 인접한 단어간의 의미가 비슷할 것이라는 가정을 사용합니다.



- **Sentence** : "I study math."
- **Vocabulary**: {"I", "study" "math"}
- **Input**: "study" [0, 1, 0]
- **Output**: "math" [0, 0, 1]
- Columns of W_1 and rows of W_2 represent each word
- E.g., 'study' vector : 2nd column in W_1, 'math' vector : 3rd row in W_2.
- The 'study' vector in W_1 and the 'math' vector in W_2 should have a high inner-product value.

위의 그림에서 주어진 학습데이터는 Sentence 하나이고, tokenization 과정을 시행해 unique 한 단어를 모은 사전이 Vocabulary 입니다. 사전의 각 단어는 사전의 사이즈만큼의 차원을 가지는 원 핫 벡터의 형태로 나타내어지며 sliding window 라는 기법을 적용해서 한 단어를 중심으로 앞 뒤로 나타난 단어와 입출력 단어 쌍을 구성합니다. 앞 뒤로 보는 단어의 수는 window size 로 설정합니다.

이러한 입출력 쌍을 가지고 예측을 하는 two layer Neural Network 를 만들었을 때, 각 단어가 vocabulary 의 사이즈인 3 차원으로 나타나기 때문에 입출력 레이어의 노드 수는 3 개가 되며 hidden layer 의 노드 수는 사용자가 정하는 하이퍼 파라미터로 Word Embedding 을 수행하는 좌표공간의 차원 수와 동일하게 설정합니다.

임베딩 차원을 2 로 설정하면 linear transform matrix 인 W_1 은 3 차원의 입력을 받아 2 차원의 출력 벡터를 내보내기 때문에 해당 행렬의 사이즈는 2x3 이 됩니다. 같은 원리로 W_2 는 3x2 가 됩니다. 이렇게 나온 output 을 softmax 에 통과시켜 3 차원 벡터가 특정한 확률분포값을 나타내도록 바꿔줍니다.

단어의 원 핫 벡터 x 와 첫 번째 선형변환 matrix 를 곱하는 것은 원 핫 벡터의 자리에 해당하는 컬럼 벡터를 W_1 에서 뽑아오는 과정으로 이를 임베딩 레이어라 부릅니다. W_1 과 W_2 의 내적을 벡터의 유사도를 나타내는 것으로 생각한다면 주어진 W_1 과 W_2 의 내적의 유사도가 최대한 커지도록 해야하며 주어진 출력단어가 아닌 다른 단어들의 W_2 상에서의 벡터들의 내적에 기반한 유사도는 작아지도록 학습시켜줍니다.

3) GloVe

GloVe 는 Word2Vec 와 다르게 각 입력 및 출력 단어쌍에 대해 같은 윈도우 내에서 얼마나 동시에 나타났는지를 사전에 계산합니다. 입력벡터 u_i 와 출력벡터 v_j 간의 내적 값이 두 벡터가 한 윈도우 내에서 동시에 몇 번 나타나는가에 대한 값 P_{ij} 에 log 를 취해준 값에 얼마나 가까워질 수 있도록 하는 loss function $J(\theta)$를 사용합니다.

$$J(\theta) = \frac{1}{2} \sum_{i,j=1}^{W} f(P_{ij})(u_i^T v_j - \log P_{ij})^2$$

Word2Vec 에서는 특정 입출력 단어쌍이 자주 등장한 경우, 단어쌍의 조합이 여러번에 걸쳐 학습됨으로써 두 벡터의 내적 값이 비례해 커지도록 하는 학습방식을 사용했다면 GloVe 에서는 단어쌍이 동시에 등장한 횟수를 미리 계산하고 이에 log 취한 값을 두 단어간의 내적값의 ground truth 로 사용하여 학습을 진행해 중복되는 계산을 줄여주는 장점이 있어 Word2Vec 보다 학습이 빨리 진행되고 더 적은 데이터에 대해서도 학습이 더 잘 됩니다.