

1. 자연어 처리의 시작

날짜 @2023년 12월 30일

▼ 목차

자연어 처리 개요

1 자연어 처리 활용 분야와 트렌드

 [Natural Language Processing, NLP](#)

 [Different Tasks of NLP](#)

 [Academic Disciplines related to NLP](#)

 [Trends of NLP](#)

 [Transformer](#)


2 기존의 자연어 처리 기법

 [Bag-of-Words Representation](#)

자연어 처리와 벡터

1 Word Embedding - (1)Word2Vec

 [Word Embedding](#)

 [Idea of Word Embedding](#)

 [Word2Vector](#)

 [How Word2Vec Algorithm Works](#)

Sliding window

[Neural Network의 구조](#)

[Neural Network의 학습](#)

[Embedding Layer](#)

[Logic 값](#)

 [Intrusion Detection](#)

 [Application of Word2Vec](#)

2 Word Embedding - (2)GloVe

 [GloVe: Global Vectors for Word Representation](#)

출석퀴즈 오답노트

자연어 처리 개요

1 자연어 처리 활용 분야와 트렌드

- ◆ 자연어 처리는 크게 다음 두 종류의 task로 구성됨

- Natural Language Understanding(NLU): 컴퓨터가 주어진 단어, 문장, 또는 문단이나 글을 이해
- Natural Language Generation(NLG): 자연어를 상황에 따라 적절히 생성

◆ 자연어 처리와 관련된 딥러닝 기술 / 적용 분야는 다음과 같음

- Language Modeling (주어진 문장/문단 일부를 보고 다음 단어를 예측)
- Machine Translation
- Question Answering
- Document Classification

Natural Language Processing, NLP

- 컴퓨터 비전과 더불어 인공지능 및 딥러닝 기술이 가장 활발히 적용되고 발전하는 분야
- 대표적으로 ACL, EMNLP, NAACL 등의 학회에서 최신 기술 및 연구 결과가 발표됨

Different Tasks of NLP

- Low-level parsing
 - **Tokenization**: 문장을 정보 단위인 token으로 쪼개 나가는 과정
 - e.g., `I study math` 라는 문장이 있을 때 이를 이해하기 위해 각 단어(`I`, `study`, `math`)로 쪼개어 보는 것
 - 문장은 이러한 token들이 특정 순서로 이루어진 하나의 sequence로 볼 수 있음
 - **Stemming**: 단어의 근본적인 의미만을 보존하는 어근을 추출하는 과정
 - e.g., 영어의 경우 `study - studying, studied` 등, 한글의 경우 `읽다 - 읽는데, 읽지만, 읽고` 등과 같이 어미가 다양하게 변화할 수 있음
 - 위와 같은 어미의 변화와 관계없이 이 단어들이 모두 같은 의미를 나타냄을 컴퓨터가 이해할 수 있어야 함
- Word and phrase level
 - **Named Entity Recognition(NER)**: 단일 혹은 여러 단어로 이루어진 고유 명사 인식
 - e.g., `New York Times` 를 개별적인 세 단어가 아닌 하나의 고유 명사로 인식

- **Part-of-Speech(PoS) Tagging:** 문장 내 단어의 품사 또는 성분 판별
 - e.g., 주어, 본동사, 목적어, 부사구, 형용사구 등을 판별하고, 어떤 형용사가 어떤 명사를 꾸며주는지 등을 인식
- **Sentence level**
 - **감정 분석(Sentiment analysis):** 주어진 문장이 긍정 혹은 부정 어조인지 예측
 - e.g., 주어진 문장이 **I love this movie** 이면 긍정 어조, **I hate this movie** 이면 부정 어조로 분류
 - **기계 번역(Machine translation)**
 - 주어진 영어 문장을 전체적으로 이해한 뒤, 적절한 외국어 단어를 대응시키고 문법과 어순을 고려하여 번역을 수행해야 함
- **Multi-sentence and paragraph level**
 - **Entailment prediction:** 문장 간의 논리적 내포 혹은 모순 관계 예측
 - e.g., **어제 존이 결혼을 했다** 라는 문장이 주어졌을 때,
 - **어제 최소한 1명은 결혼을 했다** 라는 문장은 자동으로 참이 되는 반면
 - **어제는 한 명도 결혼하지 않았다** 라는 문장은 첫 문장과 양립할 수 없는 논리적 모순 관계를 갖게 됨
 - **독해 기반의 질의응답(Question answering)**
 - e.g., 구글 검색창에 문장 형태로 검색했을 때, 그 문장의 키워드가 들어간 웹사이트를 리스트로 단순히 나열하는 것이 아니라 질문을 정확히 이해하고 답에 해당하는 정보를 검색 결과의 맨 위에 나타내주는 것
 - 질문에 들어간 키워드가 포함된 문서를 먼저 검색한 뒤, 독해를 통해 주어진 질문에 대한 정확한 정답을 알아내어 사용자에게 직접적으로 제시해 주는 과정을 거치게 됨
 - **Dialog system:** 챗봇과 같이 대화를 수행할 수 있는 자연어 처리 기술
 - **Summarization:** 주어진 문서를 자동으로 요약

Academic Disciplines related to NLP

- **Text mining**
 - 주로 빅데이터 분석과 관련됨
 - e.g.,

- 근 1년간의 뉴스 기사를 모았을 때 나타난 특정 키워드의 빈도를 시간 순으로 분석하여 트렌드를 알아내는 것
- 어떤 회사에서 특정 상품을 출시했을 때 이와 관련하여 사람들이 얘기한 내용을 수집하고 키워드의 빈도를 분석함으로써 해당 상품에 대한 소비자 반응을 얻어내는 것
 - 어떤 상품에 대해 사람들이 주로 논의하는 세부적인 요소(가성비, 내구성, 애프터 서비스 등)를 알아내고 이에 대한 의견은 어떠한지 등의 유용한 정보를 쉽게 얻을 수 있음
- 비슷한 의미를 갖는 키워드들을 그룹핑하여 분석할 필요가 있음
 - Topic modeling 등의 문서 군집화 기술로 수행할 수 있음
- 빅데이터 분석에 기반한 사회과학(Computational social science)과 밀접한 관련이 있음
 - e.g., 소셜미디어 데이터를 수집해 분석함으로써 사람들이 어떤 신조어를 많이 쓰는지, 이것이 현대의 어떠한 사회 현상과 관련이 있는지 등 사회과학적인 인사이트를 발견하는 데 활용할 수 있음
- 대표적으로 KDD, WWW, WSDM, CIKM, ICWSM 등의 학회에서 텍스트 마이닝 분야의 최신 기술 및 연구 결과가 발표됨
- **정보 검색(Information retrieval)**
 - 구글, 네이버 등에서 사용되는 검색 기술을 연구하는 분야
 - 검색 시스템의 검색 성능이 점차 고도화되며 검색 기술은 어느 정도 성숙한 상태에 이르렀으며, 이 때문에 자연어 처리나 텍스트 마이닝 분야에 비해 상대적으로 발전이 느림
 - 정보 검색의 세부 분야 중 추천 시스템 기술의 발전이 활발함
 - 음악 스트리밍 서비스에서 어떤 노래를 들었을 때 그 비슷한 노래들을 자동으로 추천해 주거나, 유튜브에서 어떤 영상을 봤을 때 그 정보를 바탕으로 그 사람이 관심 있을 법한 또 다른 영상을 자동으로 추천해 주는 기술
 - 사용자가 일일이 수동으로 검색어를 입력하지 않아도 사용자가 좋아할 가능성이 높은 것을 자동으로 제공해주는 측면에서 보다 적극적이고 자동화된 검색 시스템의 새로운 형태라고 할 수 있음
 - 상업적으로 굉장히 큰 임팩트를 가지는 분야로서, 음악과 영상뿐만 아니라 개인화된 광고나 상품 추천에 이르기까지 다양한 분야에서 활발히 활용됨



Trends of NLP

- 머신러닝과 딥러닝 기술은 일반적으로 입력과 출력에 숫자로 이루어진 데이터를 필요로 하기 때문에, 자연어 처리 문제에 이러한 기술을 적용하기 위해서는 먼저 주어진 텍스트 데이터를 단어 단위로 분리한 뒤 각 단어를 특정 차원의 벡터로 표현하는 과정을 거치게 됨
 - 단어를 어떤 벡터 공간의 한 점으로 나타낸다는 뜻에서 이를 **Word embedding**이라고 부름
- 문장 내의 각 단어를 벡터로 나타낼 때, 이 문장은 해당 벡터들이 특정한 순서로 주어지는 하나의 시퀀스로 볼 수 있음
- 이러한 시퀀스 데이터를 처리하는 데 특화된 모델 구조로 **Recurrent Neural Network(RNN)**라고 하는 딥러닝 모델이 자연어 처리의 핵심 모델로 자리잡게 됨
 - 이러한 RNN 계열의 모델 중 **Long Short-Term Memory(LSTM)**와 이를 보다 단 순화시켜 계산 속도를 빠르게 한 **Gated Recurrent Unit(GRU)** 등의 모델이 주로 사용되어 옴
- 2017년, **Attention is all you need**라는 논문이 나오며 기존의 RNN 기반 자연어 처리 모델 구조를 소위 self attention이라고 불리는 모듈로 완전히 대체 가능한 **Transformer**라는 새로운 모델이 등장했고, 이는 다양한 자연어 처리 분야에서 큰 성능 향상을 가져오게 됨
- 현재 대부분의 자연어 처리를 위한 딥러닝 모델은 이 Transformer 모델 구조를 기본으로 함

Transformer

- 원래 기계 번역 task를 위해 처음 제안됨
 - 딥러닝이 나오기 전 기계 번역에서는 언어 간 문장 구조와 어순을 전문가가 일일이 고려하여 만든 특정한 언어학적 룰을 기반으로 번역을 수행하는 기법들이 주로 사용됨
 - 그러나 다양한 예외 상황과 언어의 사용 패턴에 일일이 대응하는 것이 불가능했기 때문에, 영어 문장과 이를 번역한 한글 문장의 쌍을 모아 이를 학습 데이터로 사용하고 특정 룰이 필요하지 않은 RNN 기반 딥러닝 모델이 월등히 좋은 번역 성능을 보임
 - Google Translate, Naver Papago 등의 상용화된 번역 서비스를 가능케 한 기술
- Transformer 모델은 자연어 처리 분야뿐만 아니라 영상 처리, 시계열 예측, 신약 또는 신물질 개발 등의 다양한 분야에 활발히 적용되고 있음
 - 특정한 task에 특화되어 만들어진 것이 아님

- 핵심이 되는 self attention 모듈을 여러 겹 쌓아 모델 크기를 키운 뒤 이를 대규모 텍스트 데이터를 통해 자가 지도 학습을 시킨 것(사전 학습 모델, Pre-trained model)을 큰 구조 변화 없이 다양한 task에 적용 가능하며, 이는 특화된 모델보다도 뛰어난 성능을 보임 (전이 학습, Transfer learning)
- 자가 지도 학습(Self-supervised learning)은 별도의 label이 필요 없는 범용적 task를 활용하여 모델이 스스로 학습하도록 하는 것임
 - e.g., I study math라는 문장에서 study라는 단어를 가리고 앞뒤 문맥을 통해 모델이 이를 맞히게 함
 - 이를 통해 딥러닝 모델은 언어의 문법적, 의미론적 지식을 학습함
- 이러한 모델은 특정한 task만을 수행할 수 있는 제한적인 인공지능에서 한 발 더 나아간 범용 인공지능 기술(Artificial general intelligence)로, 현대 기술이 한 단계 더 발전한 것으로 볼 수 있음
- 다만 자가 지도 학습을 통해 사전 학습 모델을 설계하고 학습시키기 위해서는 대규모의 데이터와 GPU 리소스가 필요하므로, 최근 자연어 처리 기술의 발전은 Google, Facebook, OpenAI 등 막강한 자본력과 데이터가 뒷받침되는 일부 소수 기관에서 주도하게 됨

2 기존의 자연어 처리 기법

Bag-of-Words Representation

- ◆ “John really really loves this movie”, “Jane really likes this song”
- 다음과 같은 2개 문장으로 이루어진 텍스트 데이터셋이 주어졌을 때,
 - 1단계로 이러한 텍스트 데이터셋에서 unique한 단어를 모아 vocabulary를 구성함
 - 중복되는 단어는 vocabulary에 한 번만 등록됨
 - 각 단어를 범주형 변수(Categorical variable)로 볼 수 있으며, 이 데이터에 머신러닝/딥러닝 모델을 적용하기 위해 이를 One-hot vector 형태로 나타내어야 함
 - 위 경우 가능한 단어가 총 8개이므로 차원이 8인 좌표 공간을 설정한 뒤 이 공간에 존재하는 8개의 dimension을 vocabulary의 각 단어에 매핑하고, 해당하는 dimension의 값만을 1로, 나머지는 모두 0을 할당하여 특정한 단어를 8-dimension 벡터로 표현함

- 워드 임베딩 기법과는 대비되는 방식으로, 모든 단어 쌍에 대하여 8차원 좌표 공간상의 Euclidean distance는 모두 $\sqrt{2}$ 로, Cosine similarity는 모두 0으로 동일하게 계산됨
 - 단어의 벡터 표현형을 의미에 상관없이 모두가 동일한 관계를 갖도록 설정한 것
- 이러한 방식을 통해 각 단어로 구성된 문장 혹은 다수의 문장으로 구성된 문서 전체를 One-hot vector를 확장시켜 나타낼 수 있음
- 각 문장이나 문서에 포함된 단어들의 One-hot vector를 모두 더한 벡터로 표현하며, 이를 Bag-of-words vector라고 함
 - vocabulary상에 존재하는 각 단어별로 가방을 준비해, 특정 문장에서 나타난 단어들을 순차적으로 그에 해당하는 가방에 넣어준 후 그 수를 세서 최종 벡터로 나타내는 것
- Bag-of-words vector로 나타낸 문서를 특정 category(class)로 분류하는 대표적인 방법으로 Naive Bayes Classifier가 있음

NaiveBayes Classifier for Document Classification

Bag-of-Words

Bayes' Rule Applied to Documents and Classes

- For a document d and a class c

$$\begin{aligned}
 c_{MAP} &= \operatorname{argmax}_{c \in C} P(c|d) && \text{MAP is "maximum a posteriori" = most likely class} \\
 &= \operatorname{argmax}_{c \in C} \frac{P(d|c)P(c)}{P(d)} && \text{Bayes Rule} \\
 &= \operatorname{argmax}_{c \in C} P(d|c)P(c) && \text{Dropping the denominator}
 \end{aligned}$$

boostcamp ai&tech

© NAVER Connect Foundation

14

- C 개의 class가 있다고 할 때, 주어진 문서 d 가 C 개의 각 class에 속할 확률은 $P(c|d)$ 로 표현할 수 있음
- 가장 높은 확률을 갖는 class c 를 택하는 방식으로 문서 분류를 수행하게 되며, 이를 Maximum A Posteriori(MAP)라고 함
- Bayes rule에 의해 두 번째 식과 같이 표현됨

- d 는 고정된 하나의 문서이므로, 특정 문서 d 가 뽑힐 확률인 $P(d)$ 를 상수로 볼 수 있고, 따라서 이를 무시할 수 있어 마지막 수식이 도출됨

NaiveBayes Classifier for Document Classification

Bag-of-Words

- **Bayes' Rule Applied to Documents and Classes**
 - For a document d , which consists of a sequence of words w_i and a class c
 - The probability of a document can be represented by multiplying the probability of each word appearing
 - $P(d|c)P(c) = P(w_1, w_2, \dots, w_n|c)P(c) \rightarrow P(c) \prod_{w_i \in W} P(w_i|c)$ (by conditional independence assumption)
- $P(d|c)$ 는 class c 가 고정되어 있을 때 문서 d 가 나타날 확률
 - 이때 d 는 각 단어 w_1, w_2, \dots, w_n 가 동시에 나타나는 하나의 사건으로 볼 수 있음
 - 각 단어가 등장할 확률($P(w_i|c)$)이 서로 독립이라고 가정한다면 $P(d|c)$ 는 이들의 곱으로 나타낼 수 있음
- $P(c)$ 는 문서 d 가 주어지기 이전에 각 class가 나타날 확률
- 따라서 Naive Bayes Classifier에서 필요로 하는 모든 parameter들의 값은 $P(c) \prod_{w_i \in W} P(w_i|c)$ 를 통해 추정할 수 있음
- 예제

- **Example**

- For a document d , which consists of sequence of words w , and a class c

	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	?

- $P(c_{CV}) = \frac{2}{4} = \frac{1}{2}$
- $P(c_{NLP}) = \frac{2}{4} = \frac{1}{2}$

- **Example**

- For each word w_i , we can calculate conditional probability for class c
 - $P(w_k | c_i) = \frac{n_k}{n}$, where n_k is occurrences of w_k in documents of topic c_i

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

- For a test document $d_5 = \text{"Classification task uses transformer"}$

- We calculate the conditional probability of the document for each class
- We can choose a class that has the highest probability for the document

$$- P(c_{CV}|d_5) = P(c_{CV}) \prod_{w \in W} P(w|c_{CV}) = \frac{1}{2} \times \frac{1}{14} \times \frac{1}{14} \times \frac{1}{14} \times \frac{1}{14}$$

$$- P(c_{NLP}|d_5) = P(c_{NLP}) \prod_{w \in W} P(w|c_{NLP}) = \frac{1}{2} \times \frac{1}{10} \times \frac{2}{10} \times \frac{1}{10} \times \frac{1}{10}$$

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

boostcamp

© NAVER Cloud Platform

18

- 특정 class 내에서 학습 데이터의 단어가 전혀 발견되지 않았을 경우, 해당 단어가 나타날 확률이 0으로 추정되어 이 단어가 포함된 문장이 주어졌을 때 해당 class로 분류될 확률 값이 무조건 0으로 계산되는 문제가 있음
 - 다른 단어들이 아무리 해당 class와 밀접한 관련이 있더라도 해당 class로 분류되는 것이 완전히 불가능해지게 됨
 - 이러한 문제를 해결하기 위해 추가적인 regularization 기법을 활용할 수 있음
- 이러한 parameter 추정 방식은 Maximum Likelihood Estimation(MLE)를 통해 도출됨

자연어 처리와 벡터

1 Word Embedding - (1)Word2Vec

Word Embedding

- 각 단어를 특정 벡터로 표현할 수 있는 워드 임베딩
 - 자연어가 정보의 기본 단위(e.g., 단어)들의 시퀀스라고 볼 때 각 단어들을 특정 차원으로 이루어진 공간상의 한 점, 혹은 그 점의 좌표를 나타내는 벡터로 변환해주는 기법
- 워드 임베딩은 머신러닝/딥러닝 기술로, 학습 데이터는 텍스트 데이터셋이며 이 좌표공간의 차원 수를 사전에 미리 정의하여 워드 임베딩 알고리즘에 입력으로 주면 학습이 이루어짐

- 이후 해당하는 좌표공간 상에서 학습 데이터에서 나타난 각각의 단어들의 최적의 좌표값, 그에 해당하는 벡터 표현형을 출력으로 내어주게 됨

Idea of Word Embedding



워드 임베딩의 기본 아이디어는, 비슷한 의미를 갖는 단어가 좌표공간상에서도 비슷한 위치의 점으로 매핑되도록 함으로써 단어들의 의미상의 유사도를 잘 반영한 벡터 표현을 제공하는 것

- **cat** 이라는 단어와 **kitty** 라는 단어는 의미가 비슷하므로 좌표공간상에서 가까운 위치로, **hamburger** 는 의미가 많이 다르므로 상대적으로 두 단어와 떨어진 위치로 좌표값을 부여받음
- 워드 임베딩을 통해서 의미를 잘 반영한 벡터를 학습해두면 이를 입력으로 받아 다양한 자연어 처리 태스크를 수행할 때 보다 쉽게 성능을 올릴 수 있는 여건을 제공
 - e.g., 감정 분석 태스크를 수행하는 경우, 학습 데이터로 긍정 어조의 label을 가진 **I love this movie** 라는 문장이 주어지고 테스트 데이터로는 **i like this movie** 가 주어졌을 때 **love** 와 **like** 의 의미가 유사하므로 워드 임베딩에 의해 비슷한 벡터 표현을 얻게 됨
 - 이 벡터 표현을 감정 분석 분류기에 넣는다면 **love** 와 **like** 라는 단어의 유사도가 반영되어 테스트 데이터를 긍정으로 잘 분류할 수 있을 것임

Word2Vector

- 같은 문장에서 나타난 인접한 단어들은 의미가 비슷할 것이라는 가정을 사용
 - **The cat purrs** , **This cat hunts mice** 라는 문장이 있는 경우 각 문장 내에서 **cat** 이라는 단어를 중심으로 앞과 뒤쪽에 나타난 **The** , **This** , **purrs** , **hunts** , **mice** 등의 단어들이 이 **cat** 이라는 단어와 의미적으로 관련성이 높은 것으로 생각할 수 있음
 - **The** 와 **This** 의 경우는 이 **cat** 을 꾸며주는 단어로서의 관계가 존재하며, **purrs** 와 **hunts** 는 **cat** 이 할 만한 행동이라는 관계를 내포하고 있고, **mice** 라는 것은 **cat** 이 **hunts** 라는 행동을 할 때의 대상으로서 **cat** 과 의미적 관계를 가짐
- Word2Vec은 주변에 등장하는 단어들을 통해 특정 한 단어의 의미를 알 수 있다는 사실에 착안함
- 주어진 학습 데이터를 바탕으로 해당 단어 주변에 나타나는 다른 단어들의 확률 분포를 예측하게 됨

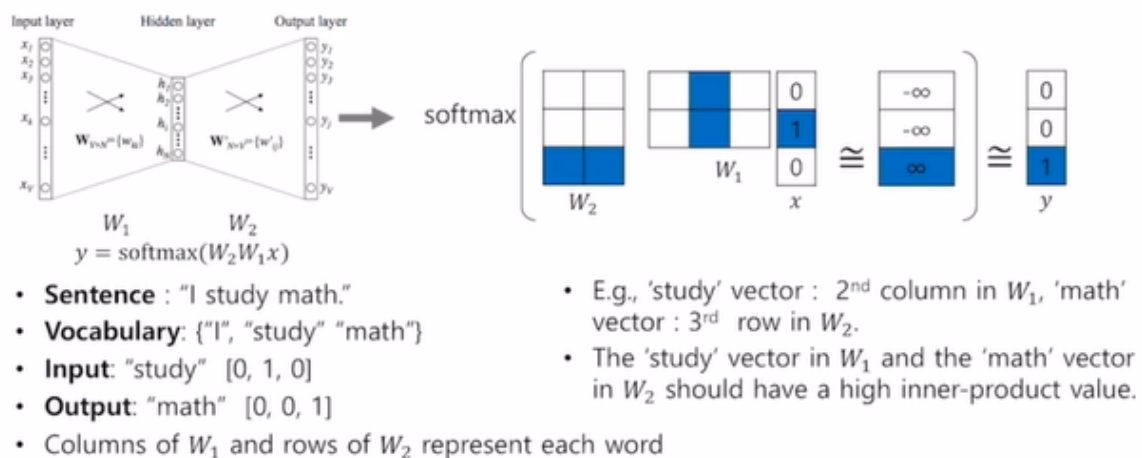
- 구체적으로는 **cat** 이라는 단어를 입력으로 주고 주변 단어를 숨긴 채 이를 예측하도록 하는 방식으로 Word2Vec 모델의 학습이 진행됨
- **cat** 이라는 단어가 나오면 그 주변에 나타나는 단어들 중 **meow**, **fleas**, **pet** 등의 단어가 높은 확률을 가진다는 것을 모델이 학습함

🚀 How Word2Vec Algorithm Works

- ◆ Word2Vector 모델의 구조와 학습 방식을 구체적으로 알아보자.

How Word2Vec Algorithm Works

Word Embedding: Word2Vec, GloVe



Distributed Representations of Words and Phrases and their Compositionality, NeurIPS'13

boostcamp

© NAVER Connect Foundation

24

- 주어진 학습 데이터가 **I study math** 라는 단 하나의 문장으로 구성된 상황을 가정
- Word2Vec에서는 주어진 학습 데이터를 워드별로 분리하는 tokenization 과정을 먼저 수행하며, 이중 unique한 단어들을 모아 vocabulary를 구축함
- vocabulary 내 각 단어는 전체 vocabulary size만큼의 dimension을 갖는 one-hot vector의 형태로 나타내어짐

Sliding window

- 이 기법을 적용하여 vocabulary에서 어떤 한 단어를 중심으로 앞뒤로 나타난 워드 각각과의 단어 쌍을 구성, 입출력 데이터를 만들
 - 윈도우의 사이즈가 3인 경우 (앞뒤로 한 단어씩만을 보는 상황)

- **I**를 중심 단어로 했을 때 앞뒤로 이 윈도우를 적용하면, 왼쪽에는 아무 단어가 없고 오른쪽에는 **study** 라는 단어가 존재하므로 **(I, study)** 라는 단어 쌍을 구성할 수 있음
- 여기서 이 윈도우를 오른쪽으로 한 칸 옮기면(sliding) 중심 단어는 **study**로 바뀜
- 왼쪽의 **I**로 인해 **(study, I)** 라는 단어 쌍을 얻으며, 오른쪽에 있는 **math** 라는 단어가 발견되어 **(study, math)** 라는 단어 쌍 또한 얻게 됨
- 주어진 학습 데이터에 각 문장별로 이러한 슬라이딩 윈도우를 적용하고 중심 단어와 주변 단어 각각의 단어 쌍을 구성함으로써 Word2Vec의 학습 데이터를 구성할 수 있음

Neural Network의 구조

- 위와 같은 방식으로 만들어진 단어 쌍들을 신경망에 입력으로 넣어 예측을 수행함
 - 각 단어가 vocabulary의 크기만큼의 차원을 갖는 one-hot vector(3차원)로 나타나기 때문에, 이 경우 입력과 출력 레이어의 노드 수는 모두 3개가 됨
- Hidden layer의 노드 수는 사용자가 정하는 hyperparameter로, Word embedding을 수행하는 좌표공간의 차원 수를 나타냄(이 경우 2개)

Neural Network의 학습

- Hidden layer는 3차원 벡터를 받아 2차원 벡터를 출력하기 때문에, 선형 변환 행렬 W_1 의 형태는 **2×3**이라고 생각할 수 있음
- Output layer는 2차원 벡터를 받아 3차원 벡터를 출력하기 때문에, 선형 변환 행렬 W_2 의 형태는 **3×2**라고 생각할 수 있음
- Output layer로부터 출력된 3차원 벡터를 Softmax layer에 통과시켜, 이 벡터가 특정한 확률 분포 값을 나타내도록 바꿔주게 됨
- Softmax layer를 통과시켜 나온 확률 분포 벡터와 Ground truth에 해당하는 확률 분포 벡터와의 거리가 최대한 가까워지도록, Softmax loss를 계산하고 W_1 과 W_2 를 업데이트시키는 과정을 반복함

Embedding Layer

- W_1 와 입력 x 의 행렬곱을 살펴보면, x 는 one-hot vector이므로 x 에서 값이 1인 부분에 해당하는 W_1 의 column의 값들만이 남게 되며 이를 Embedding layer라고 함
 - 실제로 행렬곱을 수행하지 않고 one-hot vector의 값이 1인 자리에 해당하는 인덱스와, 그 인덱스에 위치한 column vector만을 W_1 에서 뽑아오는 식으로 계산이 이루어짐

Logic 값

- Softmax layer의 입력값, 즉 Softmax에 통과시키기 전의 값을 logic 값이라고 함
- 이 logic 값은 Ground truth인 word에 해당하는 것은 ∞ , 아닌 것은 $-\infty$ 로 나왔을 때 Softmax를 통과시킨 값이 Grount truth 확률 분포와 최대한 가까워짐

Intrusion Detection

- ◆ 여러 단어들이 주어져 있을 때 이 중 나머지 단어와 의미가 가장 상이한 단어 하나를 찾아내는 task
- 각 단어에 대하여 나머지 각각의 단어와의 Euclidean 거리를 계산해 합한 뒤 평균을 취함
 - 이 평균 거리가 가장 큰 단어가 주어진 단어들 중 그 의미가 가장 상이한 단어라고 볼 수 있음

Application of Word2Vec

- Word2Vec은 다양한 자연어 처리 task에서 널리 활용됨
 - 기계 번역
 - 서로 다른 언어 간 같은 의미를 갖는 단어들의 embedding vector가 쉽게 정합 (align)되도록 함으로써 번역의 성능을 높여줌
 - PoS Tagging
 - 고유명사 인식
 - 감정 분석
 - 벡터 표현형을 통해 각 단어의 긍/부정의 의미를 보다 용이하게 파악할 수 있도록 함
- Image captioning에서도 활용이 가능함
 - 어떤 이미지의 상황을 이해하고 그에 대한 설명을 자연어 형태로 생성하는 task

Word Embedding - (2)GloVe

GloVe: Global Vectors for Word Representation

- Word2Vec과 더불어 많이 쓰이는 또다른 Word embedding 기법

- 각 단어 쌍들에 대하여, 학습 데이터에서 그 두 단어가 한 window 내 동시 등장한 횟수(P_{ij})를 사전에 미리 계산하고 이를 loss function에 사용하는 것이 차이점

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

- Word2Vec의 경우 특정 단어 쌍이 자주 등장하면 학습이 진행될수록 해당 두 embedding vector 간의 내적 값이 점점 커짐 (학습 과정에 자연스럽게 반영됨)
- 반면 GloVe에서는 특정 단어쌍이 동시 등장한 횟수를 미리 계산하고 이를 직접적으로 학습에 사용함으로써, 중복되는 계산을 줄여준다는 장점이 있음
 - 때문에 Word2Vec에 비해 상대적으로 학습이 빠르며, 더 적은 데이터에 대해서도 잘 동작함
- ◆ Word2Vec과 GloVe 모두 주어진 학습 데이터 텍스트 데이터에 기반하여 Word embedding을 학습하는 알고리즘으로, 다양한 task에서 비슷한 성능을 보인다는 것이 중요함

출석퀴즈 오답노트