16주차

날짜	@2024년 6월 25일
과제	강의 요약 질문 출석 퀴즈 캐글 필사
세부내용	[자연어 처리의 모든 것] 1. 자연어 처리의 시작 2. 자연어 처리와 딥러닝
자료	[Week16] 출석퀴즈 [Week16] 캐글필사 [프로젝트] 아이디에이션

1. 자연어 처리의 시작

자연어 처리 활용 분야와 트렌드

자연어 처리 분야 - NLP

- major conferences : ACL, EMNLP, NAACL
- Low-level parsing
  - Tokenization : 주어진 문장을 단어 단위(Token)로 쪼개는 것 → 문장은 token의 sequence
  - stemming : 단어의 어근을 추출하는 것 → 어미의 변화에도 같은 의미로 해석
- Word and Phrase level
  - Named Entity Recognition(NER) : 고유 명사를 인식하는 태스크
  - part-of-speech(POS) tagging : 단어 품사를 알아내는 태스크
  - noun-phrase chunking, dependency parsing, coreference resolution
- Sentence level
  - Sentiment analysis : 감정분석(긍정/부정 분류 태스크)
  - machine translation : 기계번역
- Multi-sentence and Paragraph level
  - Entailment prediction : 두 문장 간 모순/논리 관계를 예측
  - Question answering : 질의응답
  - Dialog systems (챗봇), summarization (한줄요약)

자연어 처리 분야 - 텍스트마이닝

- major conference : KDD, The WebConf, WSDM, CIKM, ICWSM
- 문서에서 유용한 정보 추출
- document clustering → 키워드를 가지고 주제 그룹핑
- computational social science와 밀접한 관련

자연어 처리 분야 - Information retrieval(정보 검색)

- major conferences : SIGIR, WSDM, CIKM, RecSys
- 정보 검색 기술 연구 → 어느 정도 성숙화된 상태
- 추천시스템

자연어처리 발전 과정

- 인공지능, 딥러닝 기술 가장 활발히 적용되는 분야
- 워드 임베딩 : 각 단어를 벡터로 나타내는 방법 → 순서 정보 포함
- RNN-family models(LSTMs, GRUs) 사용되어 옴
- Transformer models 등장 → 성능 향상
- NLP tasks 별로 모듈이 다양했으나,
  transformer models을 이용해 범용적 태스크로 학습하고, 전이 학습 형태로 적용하게 됨

기존의 자연어 처리 기법

Bag-of-Words

- 예 : “Jone really really loves this movie”, “Jane really likes this song”

  ① 유니크한 단어 사전을 구축
  - vocabulary : {”John”, “really”, “loves”, “this”, “movie”, “Jane”, “likes”, “song”}
  - 단어 중복을 허용하지 않아야 함

  ② 카테고리형 변수인 각각의 단어들을 원-핫 벡터로 표현
  - 단어의 수(8개)만큼을 벡터 차원으로 설정
  - 각 단어에 해당하는 부분은 1로 나머지는 0으로 매핑
  - John : [1 0 0 0 0 0 0 0]
  - really : [0 1 0 0 0 0 0 0]
  - song : [0 0 0 0 0 0 0 1]
  - 모든 단어들 간의 유클리디안 거리는 $\sqrt{2}$, 코사인 유사도는 0으로 계산됨

    → 단어 의미와 상관없이 모두가 동일한 관계를 가지도록 벡터를 표현

  ③ 문장/문서를 원-핫 벡터로 표현
  - 문장1 : [1 2 1 1 1 0 0 0]
  - 문장2: [0 1 0 1 0 1 1 1]

NaiveBayes Classifier for Document Classification

- 특정 문서 d가 C개의 클래스로 분류될 수 있다고 가정
- Maximum a Posteriori 방법

$$C_{MAP} = \arg\max_{c \in C} P(c|d) = \arg\max_{c \in C} \frac{P(d|c)P(c)}{P(d)} = \arg\max_{c \in C} P(d|c)P(c)$$

  - $P(c|d)$: 문서 d가 클래스 c 분류될 확률
  - $P(d)$: 문서 d가 뽑힐 확률 → 상수값이므로 무시
  - $P(d|c)P(c) = P(w_1, w_2, \cdots, w_n|c)P(c) = P(c)\Pi_{w_i \in W} P(w_i|c)$

    → 문서 d는 단어 w1부터 wn이 동시에 나타나는 사건과 같음

    → c가 고정된 경우 각 단어가 나타날 확률은 서로 독립이라고 가정

- 예

Data	Doc(d)	Document (words, w)	Class (c)
Training	1	Image recognition uses convolutional neural networks	CV

Data	Doc(d)	Document (words, w)	Class (c)
	2	Transformers 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	?

    - 각 클래스가 나타날 확률 : $P(c_{cv}) = P(c_{nlp}) = \frac{2}{4} = \frac{1}{2}$
    - 각 단어가 나타날 확률 :

Word	Prob	Word	Prob
$P(w_{classification}\|c_{cv})$	$\frac{1}{14}$	$P(w_{classification}\|c_{nlp})$	$\frac{1}{10}$
$P(w_{task}\|c_{cv})$	$\frac{1}{14}$	$P(w_{task}\|c_{nlp})$	$\frac{2}{10}$
$P(w_{uses}\|c_{cv})$	$\frac{1}{14}$	$P(w_{uses}\|c_{nlp})$	$\frac{1}{10}$
$P(w_{transformer}\|c_{cv})$	$\frac{1}{14}$	$P(w_{transformer}\|c_{nlp})$	$\frac{1}{10}$

    - $P(c_{cv}|d_5) = P(c_{cv})\Pi_{w \in W} P(w|c_{cv}) = \frac{1}{2} \times \frac{1}{14} \times \frac{1}{14} \times \frac{1}{14} \times \frac{1}{14}$
    - $P(c_{nlp}|d_5) = P(c_{nlp})\Pi_{w \in W} P(w|c_{nlp}) = \frac{1}{2} \times \frac{1}{10} \times \frac{2}{10} \times \frac{1}{10} \times \frac{1}{10}$
- 단점 : 단어가 1번이라도 등장하지 않으면 모든 단어들의 확률 곱으로 인해 0으로 수렴함
- 파라미터 추정 방식 : 최대우도법(MLE) 기반

Word Embedding - (1) Word2Vec

Word Embedding 이란?

- 각 단어를 좌표 공간에 최적의 벡터로 표현하는 기법
- 텍스트 데이터를 입력으로 주면 워드임베딩이 학습한 후 최적의 좌표값을 출력
- 비슷한 의미를 가지는 단어가 비슷한 공간에 매핑 → 단어 간 유사도를 나타냄

Word2Vec ?

- 같은 문장에서 나타난 인접한 단어들 간 의미는 비슷할 것이라고 가정
- “cat”이라는 단어를 주어졌을 때 주변 단어가 나타날 확률을 계산
- 알고리즘 작동 원리
    - sentence : “I study math”
    - vocabulary : {”I”, “study”, “math”} → 각 단어는 3차원의 원-핫 벡터로 표현
    - sliding window 적용 (w=3) : (I, study), (study, I), (study, math), (math, study)
    - 예측을 수행하는 네트워크를 구성 : 입력노드와 출력노드는 벡터 차원의 수와 같음
    - input : “study” [0 1 0] → W1(2×3) * x(3×1) → W2(3×2) * a(2×1) → softmax → output : [0 0 1] “math”
      ⇒ W 행렬에서 해당 원소와 곱해지는 부분의 컬럼을 뽑아오는 것과 같음 → 임베딩 레이어

Application of Word2Vec

- 기계번역, 고유명사 인식, 감정분석, POS tagging, 이미지 캡셔닝 등

Word Embedding - (2) GloVe

GloVe : Global Vectors for Word Representation

- 사전에 미리 각 단어들의 동시 등장 빈도수를 계산
- 단어 간 내적값과 사전에 계산된 값의 차이를 줄여가는 형태로 학습

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

- Word2Vec : 빈번하게 자주 등장하는 단어가 반복 연산됨

  Glove : 미리 동시에 계산된 빈도수를 계산하여 중복된 계산을 줄일 수 있음 → 빠른 학습
- 더 적은 데이터에서도 잘 동작

사전학습된 GloVe 모델

- https://github.com/stanfordnlp/GloVe
- uncased : 대소문자를 구분하지 않음 (↔ cased : 대소문자를 구분)

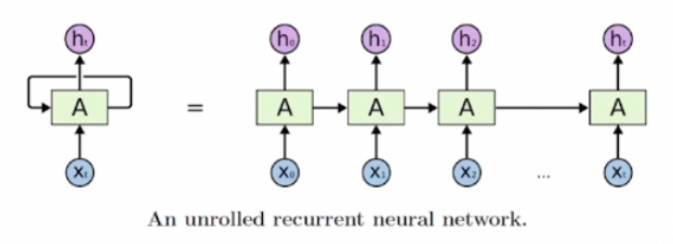
2. 자연어 처리와 딥러닝

Recurrent Neural Network (RNN)

RNN

- sequence 데이터가 입력/출력으로 주어짐
- 각 time step에서 입력된 x_t와 그 전 time에서 계산된 h_{t-1}을 받아서,

  현재 time step에서 h_t를 계산하여 내보내는 구조
- 동일한 A 모듈이 재귀적으로 사용되는 형태
- 기본 구조



An unrolled recurrent neural network.

- 구성요소
  - x_t : input vector at some time step
  - h_{t-1} : old hidden-state vector
  - h_t : new hidden-state vector
  - f_w : RNN function with parameters W
  - y_t : output vector at time step t → 매 step마다 계산하는 경우 / 마지막에만 계산하는 경우

⇒ 매 타임스텝마다 동일한 파라미터를 가진 모듈을 사용

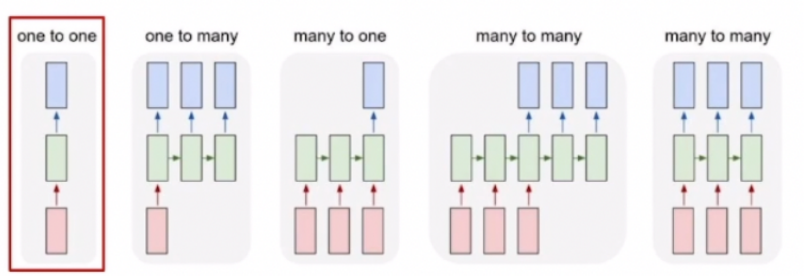
RNN 함수 정의

- 가정 : x_t → 3차원 벡터, h_{t-1} → 2차원 벡터

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$
$$y_t = W_{hy}h_t$$

- h의 차원 수는 하이퍼파라미터
- 차원 : (2x1) = (2x2) * (2x1) + (2x3) * (3x1)
- W_{hh} : $h_{t-1} \rightarrow h_t$ 변환을 담당
- W_{xh} : $x_t \rightarrow h_t$ 변환을 담당
- W_{hy} : $h_t \rightarrow y_t$ 변환을 담당

RNN의 종류

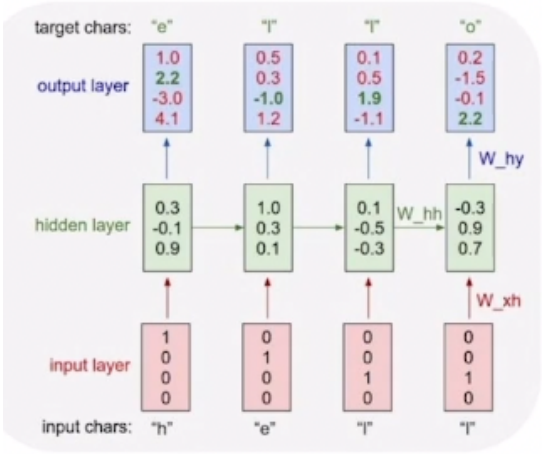


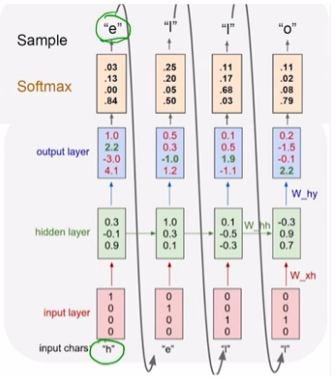
- one-to-one : [키, 몸무게, 나이] → 저혈압/고혈압 분류
- one-to-many : image captioning → 첫번째 step에서만 진짜 입력값이 들어감
- many-to-one : 감성분석 → 하나의 출력값만 존재
- many-to-many (1) : 기계번역 → 입/출력이 모두 sequence data
- many-to-many (2) : 비디오 분류, POS tagging

Character-level Language Model

Character-level Language Model

- 주어진 문자열/단어 순서를 바탕으로 다음에 올 단어를 예측하는 태스크
- 예) “hello”





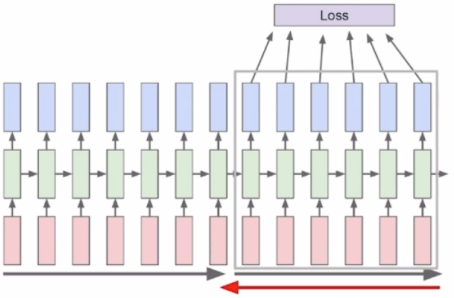
- vocabulary : [h, e, l, o]
- h = [1 0 0 0], e = [0 1 0 0], l = [0 0 1 0], o = [0 0 0 1]

    - h가 주어졌을 때 e를 예측 → he가 주어졌을 때 → l을 예측 → ...
    - logit = $W_{hy}h_t + b$ → output vector (4-dim) → softmax → 최종 출력값
- 문단에 대한 학습 → 많은 학습이 진행될수록 완전한 형태의 문장 출력
- 과거 주식 데이터로 다음날 주식값 예측

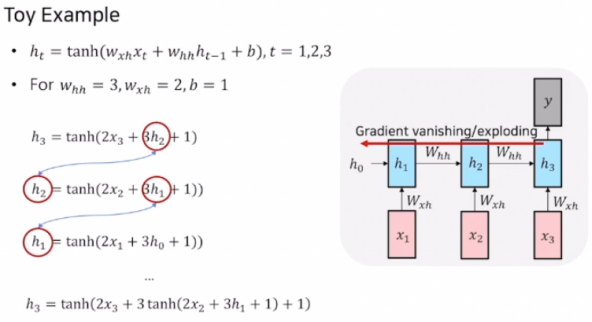
Backpropagation through time and Long-Term-Dependency

BPTT

- BPTT : 타임스텝마다 계산된 weight를 역전파로 학습하는 방식
- sequence가 너무 길어지면 학습과 역전파에 한계가 존재
- Truncation : 한 번에 학습할 수 있는 sequence를 조절하는 방법



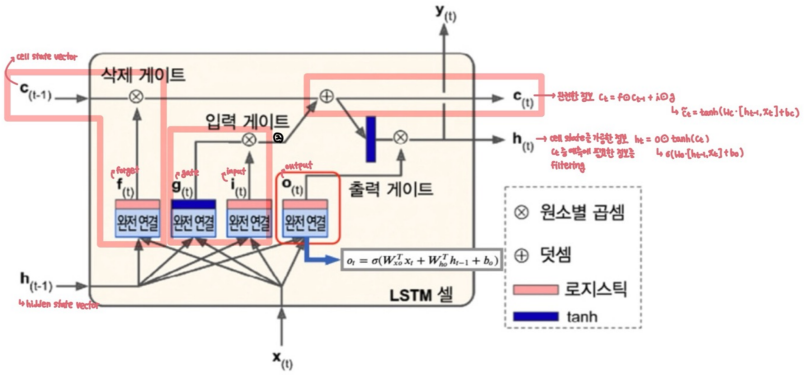
- 기존 Vanila RNN(org.RNN) : 기울기 소실/폭주로 Long-Term-Dependency 문제 발생
  ↔ LSTM : Long-Term-Dependency 문제 해결한 모델
- RNN BPTT 과정



Long Short-Term Momory (LSTM)

LSTM

- 단기기억으로 저장해두었다가 경우에 따라 꺼내 사용하여 오래 기억할 수 있도록 개선
- $\{c_t, h_t\} = \mathrm{LSTM}(x_t, c_{t-1}, h_{t-1})$



- i : input gate, cell에 사용할지 말지를 0~1로 결정하는 게이트
- f : forget gate, 정보를 어느정도 잊을지를 0~1 사이 값으로 결정하는 게이트
- o : output gate, cell 정보를 어느정도 사용할지를 0~1 사이 값으로 결정하는 게이트
- g : gate gate, 어느정도로 cell state에 반영할지를 -1~1 사이 값으로 결정하는 게이트

GRU

- LSTM에 비해 적은 메모리, 빠른 계산시간
- cell state과 hidden state를 일원화
- 이전의 은닉 상태와 현재의 은닉 상태의 가중 평균값을 계산