

[목차 & 핵심내용] - 자연어처리 바이블 (기다연)

Part ① 자연어처리 핵심이론

Ch 1. 자연어 처리의 기본

Ch 2. 자연어 처리를 위한 수학

Ch 3. 언어학의 기본 원리

Ch 4. 텍스트의 전처리

Ch 5. 어휘 분석 (lexical analysis) 규칙 - 통계 - 딥러닝 기반

Ch 6. 구문 분석 (syntax analysis)

Ch 7. 의미 분석 (sense analysis)

Part ② 자연어 처리 응용시스템

Ch 8. 개체명 인식 (named entity recognition)

Ch 9. 언어 모델 (language model)

Ch 10. 정보 추출 (information extraction)

Ch 11. 질의 응답 (question & answering)

Ch 12. 기계 번역 (machine translation)

Ch 13. 자연어 생성

Ch 14. 대화 시스템 (dialog system)

Ch 15. 문서 요약 (text summarization)

Ch 16. 텍스트 분류 (text categorization)

Part ③ 딥러닝 기반 자연어처리 [실행 설명 포함]

Ch 17. 딥러닝 소개

Ch 18. 단어 임베딩 (word embedding)

Ch 19. 합성곱 신경망 (CNN)

Ch 20. 순환신경망 (RNN)

Ch 21. DL 기반 형태소 분석 & 품사 태깅

Ch 22. DL 기반 단어의미 분석

Ch 23. DL 기반 개체명 인식 (NER)

Ch 24. DL 기반 질의응답

Ch 25. DL 기반 기계번역

Ch 26. DL 기반 문장생성

Ch 27. DL 기반 문서 요약

Ch 28. DL 기반 대화 시스템

Ch 29. DL 기반 SNS 분석

Ch 30. 응용: 이미지 학습 생성

자연어 처리 (NLP)의 의미, 어려움, NLP 패러다임 (규칙 → 통계 → 딥러닝 기반)

학습분포, 조건부 확률, 다양한 분포의 형태, MLE & MAP, 정방향 & 엔트로피, KL-Divergence, Perplexity

음절 / 형태소 / 이전 / 품사, 한국어의 9동사 (W. 계산, 수식언, 특별언, 용언, 관계언), 구구조 & 의존 구조, 의미론 & 활용론
띄어쓰기 교정, 철자 / 맞춤법 교정

형태소 분석 (morphs) - 형태소 분리 / 원형 찾기, 품사 태깅 (pos tagging)

구문 분석, 구구조 구문 분석 (전이 기반 파싱), 의존 구문 분석 (그래프 기반 파싱), 구문 분석방법의 장/단점, 구조 의존 관계 유형

단어의 중의성 (어휘적, 구조적) 해석 기법 - WordNet 사용 활용, 분류기 (NB, KNN, SVM), 의미론의 종류, 추상의미 표현 (AMR)

SVM (Support Vector Machine), NER 평가식도 - F1-score, BIO 태깅 (Begin, Inside, Outside)

통계적 언어모델 - 아르코프 가정 기반, n-gram 언어모델, 회소성 문제 해소 기법 - 스무딩 (라플라스), 빈간법, 백오프, Perplexity 계산
관계 추출 (RE), 규칙 기반 관계 추출, 기계학습 기반 관계 추출 - 시도학습, 비지도학습

정방향식 기반 QA - 질문처리단계, 응답처리단계 (Boolean 모델, 벡터공간모델 - Jaccard index, TF-IDF), 정답 처리단계
규칙기반 기계번역, 통계기반 기계번역 (조건부 확률, 베이즈 정리 활용)

n-gram 기반 자연어 생성, AI 기반 자연어생성 - 지도학습 (최대우도측정, RNN(LM)), 강화학습 (PG-BLEU, 강화학습), 적대학습 (GAN, seq-GAN)
규칙기반 - FSA, Frame 기반, 데이터 기반 - 강화학습 (MDP), 대화시스템 평가 (slot error rate, 중간판 evaluation)

입력문서, 목적, 출력문서에 따른 문서 요약, LSA (장재의미분석), 불정형수, ROUGE 평가지표

감정분석, 특징값 추출 (Dict, Count, Tfidf, HashingVectorizer), 텍스트 분류 알고리즘 (NB, SVM, KNN, SGD, DT), 텍스트 군집화 (k-means)

자동적 계층적 자질 추출, 퍼셉트론, 활성화 함수 종류, 미니Batch 학습방법

사전 학습된 단어 임베딩 기법, 분포 기반, 단어 단위 임베딩 (CBOW, skip-gram, Glove), 문장 단위 임베팅 (ELMo, BERT)

Convolution 연산, Pooling 연산, FC-layer, CNN 기반 문장 분류 (word embedding-convolutional-pooling-FC)

BPTT, 기울기 소실 문제점, 장단기 메모리 (LSTM), 케이트 순환 유닛 (GRU), RNN 기반 자연어 생성 (seq2seq, attention + seq2seq)

DL 이전 - HMM, CRF, DL 이후 - FFNN, seq2seq, character-level-bi LSTM-CRF, CNN

단어 중의성 해석 (labeled data - Bi-GRU / BERT-kNN, 지식상호보완, zero-shot 기반)

단어 단위 - Bi-LSTM, 문자 단위, 단어+문자 단위

상 (쌍둥이) 네트워크, attention 기반 LSTM 네트워크, compare-aggregate 네트워크, 시각 질의응답 (VQA)

RBTT, SBTT, NMT (딥러닝 기반) - seq2seq, RNN, Attention, Transformer (self-attention, multi-head attention)

LSTM 기반 문장 생성, attention 기반 문장 생성 (GPT-2)

RNN 기반 문서 요약 (attention 분포), pointer-generator 네트워크, coverage

특직지향 - 라이프나인 (NLU, DM, NLG), 중간간 학습, 비특직지향 - 정식 기반, 생성기반, hybrid

워드클라우드, 네트워크 분석 (중심성), 감정분석

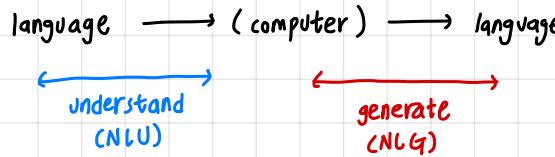
CNN 이미지 파악 추출, LSTM 번역기, 이미지 캐プ션 생성 모델 (Show & Tell)

Part 1. 자연어처리 학설이론

ch1. 자연어처리의 기본 : 인식 과정

· 자연어 처리 (NLP)

⇒ 컴퓨터가 자연어를 이해하거나 생성할 수 있도록 함
(자연어를 입·출력으로 사용하는 컴퓨터를 처리하는 과정)



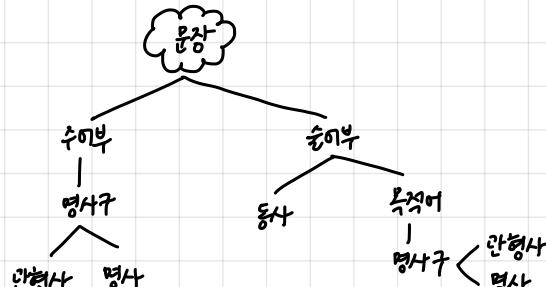
(NLV NLG) 입출력 반대 & 처리방법 다르지만 함께 흔용해서 쓰임

(자연어처리 응용분야)

- ① 기계번역
- ② 문장 요약
- ③ 음성인식 (Speech-to-Text, STT)
 - 음성이 의식 처리 ⇒ 연관성이 높은 단어로 추출
- ④ 개인비서 서비스

Q. 자연어처리가 어려운 이유?

- ① 언어의 종의성
 - : 맥락에 따라 해석의 의지가 달라짐
- ② 규칙의 제외
 - 형태론: 언어의 규칙 연구
 - 속어 → 여러 단어가 있어 대표하는 뜻이 달라짐
 - ③ 언어의 유연성 & 규칙성
 - : 유한한 소리+문자 → 만들 수 있는 문장의 수 길이가 무한함



< NLP 패러다임 >

① 규칙 기반

: 문법적인 규칙 사전에 의의 → 기반해서 자연어 처리
- 사전에 규칙을 등재하는 작업의 어려움

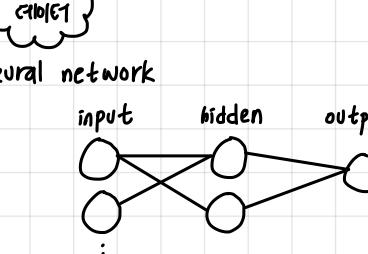
② 통계 기반

: 이미 등장한 단어 이후 다음에 나올 단어의 확률이 가장 큰 경우
↳ 전후부학률 사용
 $P(A|B) = B \text{ 가 일어났다는 가정 하에 } A \text{ 가 일어날 확률}$

③ 딥러닝 기반

1) 기계학습 (ML)

: 구조적인 알고리즘 개발 X, 문제 해결을 위한 프로그램 개발
↳ 데이터 (훈련) → 가중치 매개변수 갱신



2) 딥러닝 (DL)

: 신경망 구조에서 뉴런의 층 수 다층화
- 학습적인 분석 가능

< 딥러닝 + NLP > - 학습과정

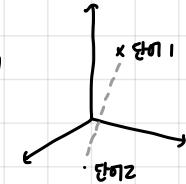
- 1) 어떤 특징으로 NLP 도입할 것인가?
- 2) 특징과 관련된 데이터셋 / corpus 구축
- 3) 학습시킬 모델 정의
- 4) 모델을 corpus로 학습
- 5) 학습모델 검증
- 6) 완성된 모델 실전에 투입

① 단어 임베딩 (word embedding)

: 자연어로 되어 있는 문장 전처리 과정

[단어 / 형태소] → 벡터 변환
(단어의 분산표현)

단어를 벡터 차원에 배치,
(의미 비슷한 단어들 = 가깝게)
의미 다른 단어들 = 멀게



② corpus (말뭉치)

- 특징과 맞는 corpus 정돈

ch 2. 자연스러운 위한 수학

2.1 확률분포

- 확률: 어떤 사건이 발생할 수 있는 가능성
- 표본공간 (sample space)

$$P(X=4\text{번}) = \text{확률}$$

/
확률변수

→ 확률 질량 함수 (Σ)

- ① 이산확률변수 (discrete random variable) (이항분포 (binomial))
: 변수 X 가 취할 수 있는 값들이 이산적인 경우 다항분포 (multinomial)

- ② 연속확률변수 (continuous)
→ 확률밀도 함수 (f)

- : 변수 X 가 취할 수 있는 값들이 연속인 경우

2.2 조건부 확률

- : 사건 A가 일어났을 때, 사건 B가 일어난 확률

$$P(B|A) = \frac{P(A \cap B)}{P(A)}$$

$$P(A \cap B) = P(A)P(B|A) = P(B)P(A|B) - \text{사건 2개}$$

$$P(A \cap B \cap C) = P(A)P(B|A)P(C|A \cap B) - \text{사건 3개}$$

↳ 연쇄 규칙 (chain rule) 기반

$$\Rightarrow P(A) = \sum_{i=1}^n P(B_i)P(A|B_i)$$

· 베이즈 정리

$$P(B_k|A) = \frac{P(A \cap B_k)}{P(A)}$$

2.3 이항분포, 다항분포, 정규분포

① 이항분포 (binomial)

- : 확률 p 인 베르누이 시행을 n 번 반복수행하는 경우

$$f(p) = {}_nC_k p^k (1-p)^{n-k} \quad \left({}_nC_k = \frac{n!}{k!(n-k)!} \right)$$

↳ 일반화

② 다항분포 (multinomial): Multi(P), $P = \langle p_1, \dots, p_k \rangle$

$$f(p_1, \dots, p_k) = \frac{n!}{x_1! \cdots x_k!} p_1^{x_1} \cdots p_k^{x_k}$$

③ 정규분포 (normal)

$$f(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \sim N(\mu, \sigma^2)$$

2.4 MLE & MAP

① Maximum Likelihood Estimate (MLE)

$$P(H) = \theta$$

$$P(T) = 1 - \theta \text{ 인 경우,}$$

$$P(D|\theta) = \theta^{\alpha_H} (1-\theta)^{\alpha_T}$$

: θ 가 확률일 때 0과 1 사이 확률

$$\hat{\theta} = \operatorname{argmax}_{\theta} P(D|\theta)$$

↳ θ 가 주어졌을 때 0과 1 사이 확률

최적의 $\hat{\theta}$ 를 찾기 위해서,

$$\begin{aligned} \hat{\theta} &= \operatorname{argmax}_{\theta} \ln P(D|\theta) \rightarrow \text{계산 편의 위해} \\ &= \operatorname{argmax}_{\theta} \ln (\theta^{\alpha_H} (1-\theta)^{\alpha_T}) \quad \text{연속증가함수 ln 사용} \\ &= \operatorname{argmax} (\alpha_H \ln \theta + \alpha_T \ln (1-\theta)) \end{aligned}$$

$$\begin{aligned} \frac{\partial}{\partial \theta} (\alpha_H \ln \theta + \alpha_T \ln (1-\theta)) &= 0 \\ \text{이분} \quad \text{max 찾기} \quad \frac{\alpha_H}{\theta} - \frac{\alpha_T}{1-\theta} &= 0 \\ \therefore \hat{\theta} &= \frac{\alpha_H}{\alpha_H + \alpha_T} \end{aligned}$$

② Maximum a Posteriori Estimation (MAP)

데이터가 주어졌을 때 4정인 확률:

$$P(\theta|D) = \frac{P(D|\theta)P(\theta)}{P(D)} \quad \begin{array}{l} \text{주어지는} \\ \text{데이터 } P(\theta) \\ \text{= 사건 지식} \end{array}$$

$$\text{Posterior} = \frac{\text{Likelihood} \times \text{Prior Knowledge}}{\text{Normalizing constant}}$$

· $P(\theta)$ = 사건 지식

· $P(D)$ = normalizing constant

- 데이터가 적은 경우에 MAP가 더 유용함

2.5 정보 이론 & 엔트로피 (entropy)

① 정보량 (quality of information)

정보량에서 하는 2가지 관점:

1) 중요성 (significance)

: 사건이 발생할 가능성이 높을수록 많은 정보를 지닌다

$$P(x_1) > P(x_2) \Rightarrow I(x_1) < I(x_2)$$

2) 가법성 (additivity)

: 두 사건 x_1, x_2 가 독립적이면

$$I(x_1, x_2) = I(x_1) + I(x_2)$$

→ 정보량의 연산 가능

$$I(x) = \frac{1}{P(x)}$$

· 2개 독립사건의 확률 = $\frac{1}{P}$ 사용

· 2개 사건의 결합된 사건 = 가법성 사용

$$\text{예) } P(H) = \frac{1}{2} \rightarrow I(H) = -\log_2(\frac{1}{2}) = 1$$

$$P(T) = \frac{1}{4} \rightarrow I(T) = -\log_2(\frac{1}{4}) = 2$$

$$I(HT) = -\log_2 P(HT)$$

$$= -\log_2(\frac{1}{8}) = 3$$

$$= I(H) + I(T) = 1+2 = 3$$

2.6 엔트로피 (entropy)

: 확률변수 X 의 표본공간에서 나타나는 모든 사건들의 정보량의 평균적인 기댓값

↓?
정보의 불확실성 ↑?

- entropy 값이 클수록 불확실성 ↑

$$H(X) = E(I(X)) \rightarrow E는 정보량의 expectation$$

$$= E(-\log_2 P(X))$$

$$= - \sum_x P(X=x) \log_2 P(X=x)$$

↳ 더 정확하게 불확실성을 측정하기 위해.

조건부 엔트로피 (conditional entropy) 사용

$$H(Y|X) = \sum_x p(X=x) \log_2 H(Y|X)$$

$$= \sum_x p(X=x) \left(- \sum_y p(Y=y|X=x) \log_2 p(Y=y|X=x) \right)$$

· Joint Entropy (joint entropy)

: 2개의 확률변수에 관한 불확실성 측정 (A면 x_i, y_j)

$$H(X, Y) = - \sum_{i=1}^N \sum_{j=1}^M p(x_i, y_j) \log(p(x_i, y_j))$$

: entropy ⇒ 2델이 적합한 정보를 지니고 있는지?

문제: 같은지 얼마나 잘 부합하는지? 측정

2.6 KL-Divergence & Perplexity

① Kullback - Leibler Divergence (KL-Divergence)

: 둘째 P 와 Q 가 서로 얼마나 일치하는지 측정

$$KL(P||Q) = \sum_i p(i) \ln \left(\frac{p(i)}{q(i)} \right)$$

- 값이 낮을수록 두 둘째가 일치하는 정도 높음

- 학습시 어떤 두 둘째를 일치하도록 할지 학습지표로 사용

② Perplexity (PPL)

: 2개의 언어 모델을 비교하는 평가지표

↳ 단어의 수는 정규화된 테스트 데이터에 대한 확률의 역수

$$PPL(w) = p(w_1, \dots, w_N)^{-\frac{1}{N}}$$

/
테스트 데이터

ch3. 언어학의 기본 원리

③ 음절, 형태소, 어미, 품사

① 음절 (Syllable)

: 언어를 듣고 말할 때 가장 작은 빌드 단위 (1음자 단위)

한국어
한성 - consonant (C)

중성 - vowel (V)

총성

〈음절의 구성〉

· V 아, ㅓ, ㅣ

· V+C 약, 일

· C+V 다, 리, 우

· C+V+C 달, 깃, 금, 성

② 형태소 (morpheme)

: 언어에서 의미를 가지는 가장 작은 단위

① 실질적인 의미의 유우 (설립 형태소 (기획 형태소) 형식 형태소 (운법 형태소))

↳ 운법적 기능

ex) 명사, 동사, 부사, 형용사

조사 - 어미의 구성

② 자립성의 유우 (자립 형태소 - 홀로 사용 의존 형태소 - 결합해 사용)

· 유일 / 복수 / 특이 형태소

: 적은 수의 형태소와만 결합 가능한 형태소

ex) 가방 - 바

요술 - 일

· 이형태 (allomorph)

: 서로 다른 모습의 형태가 하나의 형태소에 속하는 경우

ex) -을, -을 / -가, -이 / , -는, -은

· 기본형 이형태 (basic allomorph)

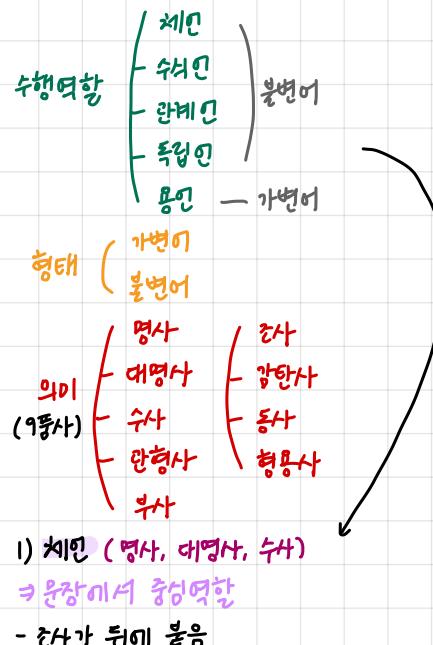
: 이형태들 중 본래의 형태와 가까운 것

③ 어절

: 1개 이상의 형태소가 묶여 구성된 단위

- 말할 때 끊는 단위 + 쓸 때 익어쓰는 단위

④ 품사



〈용언의 활용〉

(어간) (하나의 이상의 어근이 결합 + 어미)
 접사에 의해 발생되는 어간

5

는 어미에 따라 '활용' 등

활용 ↳ 서법, 높임법, 시제 등 문법적 의미 나타냄

1) 어말 어미 (증결 어미 - 문장 서술 끝맺음)

연결 어미 - 다음 말에 연결

선성 어미 - 단어를 다른 둘째는 연결

2) 선이말 어미 - 어말 어미 앞

3) 높임 선이말 어미 - 어간 뒤

4) 시제 선이말 어미

(어간) (높임 선이말 어미) (선이말 어미) (어말 어미)
 (시제 선이말 어미)

< 한국어의 9동사 >

[체언]

① 명사 : 사람 / 사물의 이름

- (보통 명사 - 구름, 꽃)
- (고유 명사 - 서울, 한라산)
- (자명 명사)
- (의존 명사 - 것, 뿐)

② 대명사 : 어떠한 것의 이름을 대신

- (인칭 대명사 - 그대, 이분, 누구)
- (지시 대명사 - 이, 그것)
- (의문 대명사 - 누구, 무엇, 이다)

③ 수사 : 수량 / 순서 나타내는 단어

- (양수사 - 하나, 둘, 삼)
- (시수사 - 첫째, 서둘째)

[수식언]

① 관형사 : 체언 앞에서 체언 꾸며주는 역할

- (성상 관형사 - 새, 옛, 현)
- (지시 관형사 - 이, 그, 다른)
- (수 관형사 - 한, 두, 모든, 몇)
- 활용형 X → 어떤 - 이미로 나누지 않으며
조사가 놓지 않고 서제에 없음

[관계언]

① 조사 : 체언 / 용언의 명사형 뒤에 붙어 말의 뜻 더해줌

- (주조사 (격을 나타냄) - -라, -과)
- (접속조사 (단어 - 단어 이어짐) - -과, -라)
- (보조사 (격조사 자리끼울 수 있음) - -는, -만, -까지도)

[부조사의 종류]

- 1) 주격 (체언이 서술어의 주제임 표시) - -이, -가
- 2) 서술격 (선행하는 체언과 결합 → 서술어로) - -이다
- 3) 목적격 (체언이 서술어의 목적지임 표시) - -을, -를
- 4) 양격 (" 봄 ") - -이, -가
- 5) 관형격
- 6) 부사격 (" 부사어 ") - -이가
- 7) 호格 (호칭) - -야, -아, -시여

[독립언]

① 감탄사

- 놀람 / 감정 - 아이고, 밀씨구
- 의지 - 떠나, 올지, 굳세
- 호응 - 그대

[용언]

어간 ⇒ 동사 / 형용사

① 동사 : 사물의 동작 / 작용 나타냄

- 목적이 필요? (자동사 - 목적이 필요X : 주체가 남이 하게 하는 동사)
 타동사 - 목적이 필요

행동의 자발성 (능동사)

피동사 : 본인이 당하는 동작

행동의 주체? (주동사)

사동사

쓰임 (본동사 - 단독으로 서술어 O)

변조동사 - 변조 형용사의 도움 받는 동사

활용 형태 (규칙 동사 - 어간 변화X, 이미 규칙적으로 활용)

불규칙 동사 - 어간 불규칙적 변화

3.2 구조 & 의존구조

① 품조

: 문장을 구성하는 요소들을 결합해서 형성되는 일정한 구조

구조요소 1

구조요소 2 (선행관계
지배관계)

수형도 (Parse-tree)
→ 명사구 (N)
→ 동사구 (V)

- 표면적으로는 같은 언어요소 + 같은 순서로 구성
- 뜻이 다른 2개 이상의 문장을 다는 구조로 기술

② 의존구조

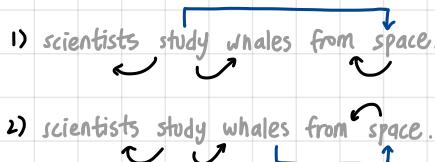
: 문장 내 단어들이 서로 의존관계를 통해 하나의 구문 형성

문장 [지배소 (head)
의존소 (modifier)]

↪ 의존구문 분석 : 의존 관계를 분석

- 한국어: 구조분석 < 의존구문 분석

의존트리 (Dependency tree)



→ 의존트리의 형식에 따라 단어들 간의 의존관계가 달라짐

의존구문 분석 (그래프 기반 (graph-based) - 비결정적)

(전이 기반 (transition-based))

↪ 선형적 탐색 - 근거리 의존관계 O

한 문장이 갖는 모든 의존 분석 결과 중에서

가장 높은 정수의 의존트리 선택

- 모든 문장을 훑기 때문에 속도 ↓

3.3 의미론 & 학용론

① 의미론

: 단어, 문장, 발화에서 표현이 실제로 가리키는 지시체 - 의미 관계 파악

< 의미를 파악하는 방법 >

① 의미역 (semantic roles / thematic relations)

→ 명사 단어의 다양한 역할 설명

- agent : 어떤 행동을 의도를 가지고 수행
- patient : 어떤 행동에 영향받는 개체
- instrument : 어떤 행동이 일어나도록 함
- location : 어떤 행동이 일어나는 장소

② 어휘적 관계 (lexical relation)

- | | |
|--------------------|--------------------|
| - 동의 관계 (synonymy) | - 동음이의어 |
| - 반의 관계 (antonymy) | - 다의어 (polysemy) |
| - 상하 관계 (hyponymy) | - 현유어 (metonymy) |
| - 원형 (prototype) | - 연어 (collocation) |

③ 학용론

: 언어 사용자 - 발화 맥락 (context) 연구

1) 적시 - 상대가 특정한 것을 가리켜 표현

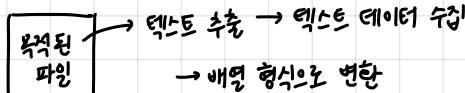
2) 하행 (적립 하행 예) 적설질문
(간접 하행 예) 제안

ch4. 텍스트 전처리

비정형 데이터 (unstructured data)

: 그림, 음서, 영상과 같이 형태가 구조가 다른, 구조화되지 않은 데이터

4.1 텍스트 문서의 변환



4.2 띄어쓰기 교정

- 띄어쓰기의 부재 ⇒ 의미 혼용

- 띄어쓰기 교정 방법: ① 규칙 기반
② 통계 기반
③ 딥러닝 기반

① 규칙 기반 (형태소 분석기)

- 직접 규칙 등을 설정, 규칙의 기준에 따라서 교정
- 무한한 수의 규칙을 직접 계획 생성하기에 어려움

② 통계 / 학습기반 (n-gram)

- 말뭉치 → 음절 n-gram 정보 활용
- 학습 데이터셋의 크기를 줄이기 성능 ↑ (Data sparseness 문제)
- 학습 말뭉치의 영향이 큼

예) "아버지가방에들어갔다."

$$\begin{aligned} P(\text{아버지}) & P(\text{방에} | \text{아버지}) \dots \\ P(\text{아버지가}) & P(\text{방에} | \text{아버지가}) \dots \end{aligned}$$

→ bigram 언어모델

⇒ 학습이 가장 큰 옵션 선택

4.3 철자 및 맞춤법 교정

- 텍스트 내 오류 감지
- 오류의 수정

<오류의 종류>

- 삽입 (insertion) - 추가적으로 문자 입력
- 생략 (deletion) - 없어야 하는 문자 생략
- 대체 (substitution) - 없어야 하는 문자 대신 다른 문자 대체
- 순열 (transposition) - 철자 순서 바꾸기

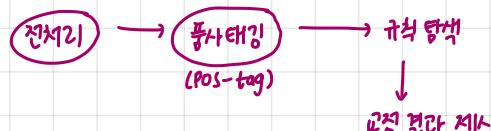
0	1	2	3	4	5
---	---	---	---	---	---

 → 문자 삽입 / 생략 error

↓
<문자>

오류 수정:

① 규칙 기반



- 입력 문장이 들어오면 태그셋으로 품사 태깅!
- 있는 것이 없다면 통신 예절
- 통신 예절 = 주사 / 어미 등의 문법 형태소의 음절 정보 /
대규모 말뭉치에서의 음절간 결합 정보
- 맞춤법 교정
- mecab-ko-dic 품사 태그 label

② 통계 / 학습 기반 (Bayesian inference model)

- 주어진 단어로부터 오타가 일어난 학습 문서

$$\hat{c} = \underset{c \in C}{\operatorname{argmax}} p(t|c) p(c)$$

선택된
맞춤법 수정결과
단어 c 사용 확률
P(오타 t | 움은 단어 c) 뱍 학습
율은 단어 후보군

- 이것이 가장 큰 것으로 대체해서 맞춤법 교정

ch 5. 어휘 분석 (lexical analysis)

: 한 단어의 구조 분석 + 분석 → 의미 + 품사 단어 수준 연구

어휘의 의미를 판단, [1) 형태소 분석

2) 품사 태깅 (part of speech, POS)

5.1 형태소 분석 (Morphological analysis)

: 최소한의 의미를 갖는 형태소를 이용해 자연어의 제약 + 조건에 맞게 분석

(형태소 분석 철학)

- 단어에서 핵심 의미 포함하는 형태소 분리
- 형태론적 변형이 일어난 단어의 원형 찾기
- 단어 속 사전들 사이의 조합 조건에 따라 옳은 분석 후보 선택

① 형태소 분리

· 이질 / 단일 ⇒ 하나 이상의 형태소 결합 (concatenate)된 형태
= 형태소열 (sequence of morphemes)

- 형태소 = 1개 이상의 품사를 가질 수 있음
→ 1개 이상의 형태소로 품사의 쌍으로 이루어짐

· 형태소 품사쌍 (morpheme-tag pair)

= 형태소 + 품사의 쌍

- 이질 = 1개 이상의 형태소 품사쌍으로 이루어짐

예) 나 (대명사) + 는 (조사)

날 (동사, 날다) + 는 (조사)

:

② 형태소 원형 찾기

· 한국어 ⇒ 형태소 변형 (morphological transformation)

이 일어나기 전에 형태소 원형 찾기

5.2 영어 형태소 분석

영단어 (stemming (기간 추출)
lemmatization (표제어 추출))

- 일반적인 영어의 형태소 = 접두사

접두사 (접두사 - unsure
접미사 - careful)

5.3 한국어 형태소 분석

KoNLPy (한국어 (Hannanum)
- 고모란 (Komoran)
- 미캡 (mecab)
- 고꼬마 (Kkhma)
트위터 (Twitter)
- Khaiii (카카오))

5.4 품사 태깅 (POS tagging)

(품사의 가능성과 종류)

- 주체 ⇒ 체언 (명사, 대명사, 수사)
- 활용 ⇒ 용언 (동사, 형용사)
- 수식 ⇒ 수식언 (관형사, 부사)
- 독립 ⇒ 독립언 (장단사)
- 관계 ⇒ 관계언 (조사)

⇒ 품사 태깅을 통해 "형태론적 중의성 해결"
(morphological disambiguation)

자동 품사태깅

1) 자식 기반 POS-tagging

- 문맥틀 (context frame) 형식 규칙 기술
- 제약 문법 이용 규칙 기술
- 말뭉치 → 빈도 높은 중의적 단어 처리
- 휴리스틱 규칙
- 비문맥 규칙 기술
- 패턴 - 처리 형태의 부정지식
- finite-state intersection grammar

2) 통계 기반 POS-tagging

- 변형 마르코프 기법
- 통계적 결정 트리
- 최대 entropy 모형
- 베이지안 추론
- 페지망
- 반복 알고리즘 → labelling 기법
- 분별 학습

① 규칙 기반 POS-tagging

: 언어 정본에서 생성되는 규칙 기반 태깅

· 긍정 정보 ([O or O])

: 문장에서 선호되는 어휘 태그에 대한 언어 지식

[가 or 나] → 가 [다 or 라]

↑
가,나에 중의성

· 부정 정보 (O ? O)

: 특정 문장에서 배제되는 언어 지식

가? 나 → not 다

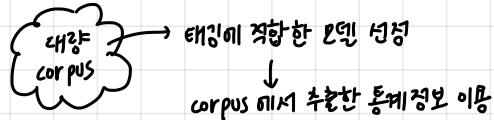
· 수정 정보 (O : O)

: 다른 태깅 방법 유발 시에 오류 수정하기 위해 사용
(= 오류 교정)

A: 가 → 나

(잘못된 정본가 들이왔을 때 가 → 나로 수정)

② 통계 기반 POS-tagging



<은하 마코프 모델 (HMM: Hidden Markov Model)>

↳ 은하 확률만 이용

· 문장의 길이 = N , $w_{1:N} = w_1, w_2, \dots, w_N$

(w_i = 문장에서 i 번째에 나타나는 단어)

· 가장 확률 높은 풍사 태깅 $c_{1:N} = c_1, c_2, \dots, c_N$

(c_i = i 번째 단어에 할당되는 풍사)

$$T(w_{1:N}) \stackrel{\text{def}}{=} \underset{c_{1:N}}{\operatorname{argmax}} P(c_{1:N} | w_{1:N})$$

$$= \underset{c_{1:N}}{\operatorname{argmax}} \frac{P(c_{1:N}, w_{1:N})}{P(w_{1:N})}$$

$$= \underset{c_{1:N}}{\operatorname{argmax}} P(c_{1:N}, w_{1:N})$$

chain rule 적용해
개별 단어의 확률 둘로 변형

① $P(c_i)$ 먼저 분리

$$P(c_{1:N} | w_{1:N}) = P(c_1) P(w_1 | c_1)$$

$$= P(c_2 | w_1, c_1) P(w_2 | w_1, c_1, c_2)$$

:

$$\text{확률 끝} = P(c_N | w_{1:N-1}, c_{1:N-1}) P(w_N | w_{1:N-1}, c_{1:N})$$

$$= \prod_{i=1}^N P(c_i | w_{1:i-1}, c_{1:i-1}) P(w_i | w_{1:i-1}, c_{1:i})$$

② $P(w_i)$ 먼저 분리

$$P(c_{1:N}, w_{1:N}) = P(w_1) P(c_1 | w_1)$$

$$= P(w_2 | c_1, w_1) P(c_2 | c_1, w_{1:2})$$

:

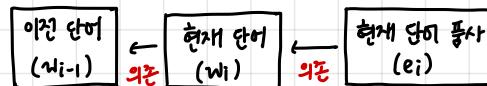
$$= \prod_{i=1}^N P(w_i | c_{1:i-1}, w_{1:i-1}) P(c_i | c_{1:i-1}, w_{1:i})$$

· 마르코프 가정 (Markov assumption)

⇒ 풍사 태깅 모델 유도

$$P(c_i | w_{1:i-1}, c_{1:i-1}) \approx P(c_i | c_{i-1})$$

$$P(w_i | w_{1:i-1}, c_i) \approx P(w_i | c_i)$$



⇒ 현재 풍사의 발생은 바로 이전 풍사에만 의존한다.

↳ 풍사 태깅 모델 개발

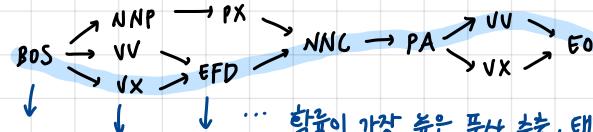
$$T(w_{1:N}) = \underset{c_{1:N}}{\operatorname{argmax}} \prod_{i=1}^N P(c_i | c_{i-1}) P(w_i | c_i)$$

= 이후 발생 정보 + 풍사 유역 정보 사용

· $P(c_i | c_{i-1})$ = 상태 전이 확률 (state transition probability)

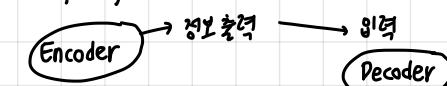
· $P(w_i | c_i)$ = 관측 심볼 확률 (observation symbol //)

∴ 마르코프 가정 기반 + HMM 모델 기반의 풍사 태깅 방법.



③ 딥러닝 기반 POS-tagging

예) seq2seq



예) 세종 → 문자 단위 데코더 → 양방향 LSTM (Bi-LSTM)

· 문자 단위 = 초기자음 + 모음

(초기자음 + 모음 + 마지막 자음)

- "문자 융합 철자 규칙" ⇒ 문자 형태 변환 가능

(이전 문자) ————— (다음 문자)
 마지막 자음 ————— 초기 자음

↓
결과 조합 문자 변환

① KEEP: 수정되지 않는 문자

(B. 시작)

② NOOP: 사라지는 문자

I: 내부

③ MOD: 수정되는 문자

⇒ 변화하는 문자 태그 + 풍사 태그

→ Bi-LSTM (양방향 LSTM) + CRF

· CRF (Conditional Random Field)

: 주변 충격에 대한 가능성 고려 가능

→ 주어진 input에 대해 가장 높은 output 출력

Ch6. 구문분석

- 자연어 문장에서 구조 요소들의 유법적 구조 분석
- ⇒ 유법적 구조 정의 적용 추출

① 규칙 기반 분석 : 인간이 직접 정의한 문법 규칙 적용

② 통계 기반 분석 : 학습적 유법 규칙 통계적으로 계산

③ 딥러닝 기반 분석 :  → 딥러닝 템 험
데이터셋

6.1 구문 문법 (construction grammar)

- 유법적 구조 요소들 → 문장 생성 +
문장 → 구조 요소들로부터 생성

1) 구조 문법

(phrase structure grammar)



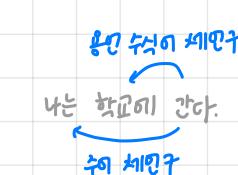
- 절 \neq 단어들 간의 계층적 관계

- 구조 관계 기반

- 트리 구조 분석

2) 의존 문법

(dependency grammar)



- 문장 구성 단어들 간의 계층적 관계

- 의존 관계 기반

6.2 구문 중의성 (syntax ambiguity)

구문 중의성 해소를 위해서 ⇒ 의미 / 문맥 등 추가적 정보 필요

6.3 구조 구문 분석 (phrase structure phrasing)

: 단어들 + 단어들이 2인 철의 계층적 관계 따라 분석

① 규칙 기반 구조 분석

1) 구조 문법 규칙 미리 정의

- 구조 문법 : 문장을 하위 구성을로 나눔으로써 문장 구조 나타냄
(구성소 : 1개의 단위로 가능한하는 일련의 단어들)

$$A \rightarrow BC$$

(구성소 A → 하위구성소 B,C)

2) 규칙에 따라 어휘 사전 (lexicon) 내의 단어들 대입

② 통계 기반 구조 분석

1) 통계적으로 학습적 구조 문법 계산

- 학습적 구조 문법 : 각 규칙 조건부 확률로 정의

$$A \rightarrow BC [P]$$

(P: 구성소 A가 하위구성소 B,C로 나뉨 조건부 확률)

2-1) 인간이 tagging 한 corpus로부터 조건부 확률 계산

$$P(\alpha \rightarrow B | \alpha) = \frac{\text{Count}(\alpha \rightarrow B)}{\sum_n \text{Count}(\alpha \rightarrow n)} = \frac{\text{Count}(\alpha \rightarrow B)}{\text{Count}(\alpha)}$$

(Count($\alpha \rightarrow B$) : corpus에서 $\alpha \rightarrow B$ 가 나타난 횟수)

2-2) tagging되지 않은 문장들을 구조 구문 분석 → 조건부 확률 조정

Inside - outside 알고리즘

- 유법 규칙들이 동일한 학률을 가지고 있다고 가정,

- 문장의 구문 분석 결과 학률 계산 → 유법 규칙들의 조건부 확률 조정

$$P(T|S) = \prod_i P(\alpha_i \rightarrow B_i | \alpha)$$

P(T|S) = 자연어 문장 S가 주어졌을 때 T의 조건부 확률

$$T' = \operatorname{argmax}_T P(T|S)$$

↳ 학률 가장 높은 구문 분석 규칙 사용

③ 딥러닝 기반 구조 분석

- 딥러닝 모델 ⇒ 자연어 문장의 표준 정보 +
의미적 정보 입력
→ 구성을의 구조 예측

<전이 기반 파싱>
(transition-based parsing)

: 자연어 문장을 한 단어씩 읽으며 →
현재 단계에서 수행할 action 선택

↳ 예) 이동-감축 파싱 (shift-reduce parsing)

- ① 이동 (shift)
- 각 전이 단계에서 선택
- ② 단항 감축 (unary)
- ③ 이항 감축 (binary)

① 이동 (shift)

: 자연어 문장에 포함된 단어를 스택에 이동시키는 연산

② 감축 (reduce)

: 스택에 포함된 1개 / 2개의 구성소를 꺼내 상위 구성소로 감축,
상위 구성소를 다시 스택으로 이동



여기 action을 할 것인가? = Oracle (다리를)이 결정

(딥러닝 전이 기반 파싱 = Oracle이 RNN Encoder로

제한한 문장, 단어 특성 벡터, 스택의 현재 상태 input)

· 정점 :

- 입력된 자연어 문장에 포함된 단어 수에 선형적인
전이 action 가능

· 단점 :

- 각 전이 action 선택시 문장의 전체적 구조 고려하기 어려움

⇒ 거내가 먼 단어들의 문법 구조 고려하기 어려움

6.4 의존구문 분석

: 문장 내 단어들의 의존 관계 분석

$\text{node} = \text{각 단어}$

$\text{edge} = \text{의존 관계} \rightarrow \text{의존 분석 트리 구축}$

$\text{tree} = \text{node} + \text{edge}$

① 규칙 기반 의존 분석

(지배소 (head)) : 의존 관계 표현에서 절의 중심

(의존소 (modifier)) : 절 내 지배소에 의존하는 단어

"안다"의 의존노드 (node로 표현)

↑
나는 학교에 안다
절의 중심 = 지배소
(root node로 표현)

절의 중심 = 지배소
(root node로 표현)

(CG (Constraint Grammar) 알고리즘

⇒ 문맥 의존 규칙 정의 → 규칙 기반 의존 분석 시행

② 딥러닝 기반 의존 분석

(선이 기반 파싱)

(그래프 기반 파싱)

1) 선이 기반 파싱 (transition-based parsing)

: 단기 하사식의 의존 분석 트리에 포함

- 계속 이동 연산으로 스택에 단어들을 채운 다음

- 강화 연산으로 의존 분석 트리 구성

- 지배소인 "안다"는 2번 강화 연산 ("나는"
"학교에")

2) 그래프 기반 파싱 (graph-based parsing)

: 문장 내 단어간의 모든 의존관계 정수화

→ 가장 높은 정수의 의존 분석 트리 선택

(상점 : 문장 전체의 문법 구조 고려)

(단점 : 시간복잡도 ↑ ⇒ 비효율적)

6.5 구문 분석 방법의 장단점

6.3 구구조 분석 (규칙 기반)

(통계 기반
딥러닝 기반)

6.4 의존 분석 (규칙 및 통계 기반)

(딥러닝 기반)

⇒ 구문 분석에서 각 방법의 장단점?

① 규칙 기반

장점	단점
- 미리 정의된 규칙 사용	- 문법 규칙 미리 정의
⇒ 성학한 의존 분석	- 시간 + 비용 문제 - 자연어 문장의 중의성 해소 X

② 통계 기반

장점	단점
- 조건부 학습 기반 계산	- 문장 전체의 문법 구조 활용 X
⇒ 구문 중의성 해소 가능	- 하위 병주화 정보 활용 X

③ 딥러닝 기반

장점	단점
- 자연어 문장내 여러 정보 활용	- 실수 파라미터 사용
(선체 구조 성보 하위 병주화 정보)	⇒ 문법 규칙 해석 불가
↓	특정 백터 계산

6.6 구조 의존 관계 유형

: 문장 내 각 부분이 어떤 문법적 구조 가능하는지?

(구의 유형 (구구조 분석))

(의존관계 유형 (의존 분석))

① 구구조 분석 - 구의 유형 (제3층 알뭉치)

: 구문 표지 - 가능 표지 형태로 출력

1) 구문 표지

NP - 체언구

VP - 동언구

VNP - 궁정지정사구

AP - 부사구

DP - 관형사구

IP - 강한사구

X - 의사구

L/R - 부호 (왼쪽, 오른쪽)

2) 가능 표지

SBJ - 주어

OBJ - 목적어

CMP - 보어

MOD - 체언수식어

AJT - 용언수식어

CNJ - 접속어

INT - 특립어

② 의존 분석 - 의존관계 유형 (Universal dependencies 셋)

acl - 형용사절

advcl - 부사절 한정어

advmod - 부사 한정어

amod - 형용사 한정어

appos - 동격 한정어

aux - 조동사

case - 격 표시

cc - 등위 접속사

ccomp - 접속절

cfl - 문류사

compound - 합성어

conj - 접속

cop - 연결동사

csubj - 명사절

dep - 명시되지 않은 의존관계

det - 한정사

discourse - 달화 표지

expl - 허사

fixed - 문법적 단위 표현

flat - 단단히 표현

iobj - 간접 목적어

list - 목록

mark - 표지

nummod - 수 한정어

obj - 목적어

parataxis - 병렬

Ch 7. 의미 분석

7.1 단어 - 단어 사이의 중의성



7.2 중의성 해소 방법

단어 중의성 해소 방법 (word sense disambiguation)

= 문장 내 중의성을 갖는 단어 — 사전에 정의된 의미와 비교

- 지식기반 방법
- 지도학습 방법
- 심층학습 방법 (지식정보 + 단어의 의미가 labeling된 데이터 같이 이용)

① 지식 기반 방법

: 문장에 등장한 단어들 \Rightarrow 정의된 지식을 활용해서 예측

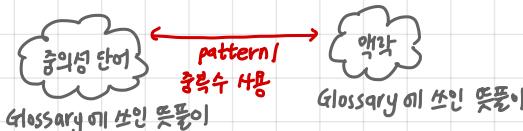
예) WordNet, ConceptNet, BabelNet, Freebase



단어의 각 의미들의 상하위어 + 반의어 + 동의어 개념 포함

\Rightarrow 활용해 (사전 정의 기반 방법 : 사전에 정의된 뜻풀이 기반 의미 추론
그래프 기반 방법 : 단어의 의미관계 정보 (단어 의미망 사전)으로 의미 추론

예) 사전 정의 기반 방법 - 레스크 (lesk) 알고리즘



예) 그래프 기반 방법 - DFS

: 문장 내 단어들의 연결성 추출

\rightarrow 각 의미 간에 연결이 가장 많이 된 의미 선택

② 지도학습 방법

: 문장 내 단어의 의미가 labeling된 데이터 이용

\rightarrow 기계학습 모델에 학습 \rightarrow 단어 의미 예측

예) - 나이브 베이즈 분류기 (Naive Bayes classifier) - NB

- kNN (k-Nearest Neighbors classifier)

- 서지벡터기 (Support Vector Machine) - SVM

1) NB 분류기

중의적 단어 \rightarrow 주변 맥락

$$c = \underset{c_k}{\operatorname{argmax}} P(c_k | \vec{x}) \rightarrow \text{가장 빈도수 높은 } c \text{ 찾기}$$

$$= \underset{c_k}{\operatorname{argmax}} \frac{P(\vec{x} | c_k)}{P(\vec{x})} \cdot P(c_k) \quad) \log \text{ 합수화}$$

$$= \underset{c_k}{\operatorname{argmax}} (\log P(\vec{x} | c_k) + \log P(c_k))$$

$$= \underset{c_k}{\operatorname{argmax}} (\sum_{v_j \in \vec{x}} \log P(v_j | c_k) + \log P(c_k))$$

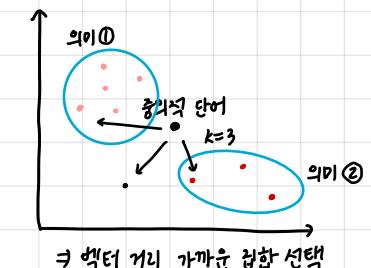
\Rightarrow 주변 맥락 고려해서 중의적 단어의 의미 결정

2) kNN 분류기

: 벡터 공간에 표현된 자료들 \Rightarrow 정해진 k값에 따라서 가장 많이 묶이는 자료들 \rightarrow 의미 클래스에 선택

\hookrightarrow 벡터 거리 계산 \Rightarrow ① 유clidean 거리 (Euclidean distance)

② 코사인 유사도 (cos similarity)

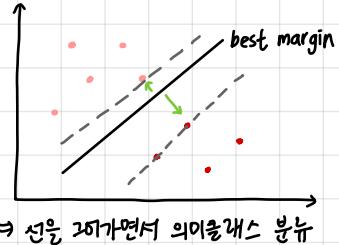


$$\text{Euclidean distance} = \sqrt{\sum_{i=1}^n (p_i - q_i)^2} \quad \cdot \text{ 두 점 } P, Q \\ (P = (p_1, \dots, p_n) \\ Q = (q_1, \dots, q_n))$$

$$\text{cosine similarity} = \frac{A \cdot B}{\|A\| \|B\|} = \frac{\sum_{i=1}^n A_i \times B_i}{\sqrt{\sum_{i=1}^n (A_i)^2} \sqrt{\sum_{i=1}^n (B_i)^2}}$$

3) SVM 분류기

- 벡터 공간에 표현된 차집으로부터 의미 클래스 분류하기 위해 의미 클래스간 가장 넓은 길이 사용하는 방향으로 선을 그어 의미 분류
- 합집 차집 / 차집 결합 커도 robust



①.3 차집 선택 / 차집 결합

① 차집 선택

- 좋은 분류성능의 차집 선택 + 안좋은 분류성능의 차집 제외
- 예) 단어의 출현빈도

② 차집 결합

- 스테밍(stemming): 동일한 단어들의 접사 / 어미 제거
- 용어 클러스터링
- 잠재 의미 색인(Latent Semantic Indexing, LSI)

②.4 의미역 (semantic role) 분석

- 의미를 해석하기 위해 서술어가 수식하는 대상의 의미관계 파악
→ 역할 분류
- 논항: 수식을 받는 대상
↳ 문장 구조가 바뀌어도 행위주 / 피동주 바뀌지 않음

"영희가 밥을 먹는다" "철수가 택배를 보낸다"
⇒ 의미역의 의미 같음(행위를 하는人)

〈의미역 종류〉

① 필수적 의미역

- 서술어의 의미를 구성하는데 필수적으로 요구됨
 - 행동주(agent) ⇒ 행위를 하는 주체 (고의성/의도성 O)
철수가 돈을 뺀다.
 - 도구(instrument) ⇒ 행위, 이동의 의미를 표현하는 통사 수단
철수가 앞치로 돈을 뺀다.
 - 피동주/수용자(patient) ⇒ 동사가 행위 표현시 행위에 영향 받거나 상태 변화를 끊는 것
철수가 인희를 사랑했다.
 - 경험자(experiencer) ⇒ 현상의 경험 주체가 되는 내적 상태에 영향을 받은 것
영희가 사랑에 빠졌다.
 - 수혜자(benefactive) ⇒ 행위가 수행될 때 이익 받는 개체
내가 철수에게 밥을 사줬다.
 - 출처/근원(source) ⇒ 행동의 동기/이유의 출처, 장소
나는 식당에서 밥을 주문했다.
 - 도달점/목표(goal) ⇒ 공간적/상정적/주관적 목적지
나는 책을 서랍에 보관했다.

② 수의적 의미역

• 서술어의 의미 보충

- 장소/위치(locative) - 경로(path)
- 이유(reason) - 시간(time)
- 목적(purpose) - 방법(manner)

〈지도학습 기반 의미역 분석〉

- 기계학습 기반의 의미역 분석

⇒ CRF/SVM 등의 분류기 사용 (입력 = 다양한 차집)

개선방안

- 1) 일반 명사에 대한 등형이의어 정보 + 개체명 인식 정보 추가
- 2) 사건에 있는 단어들까지 확장해서 사용

②.5 의미표현

↳ 추상 의미표현 (abstract meaning representation, AMR)

- 1개의 의미분석 대상 X, 여러 의미분석 결과는 결합해서 문장의 의미표현

- 그래프 기반 표현
- 프레임셋(frameset) + 일반 이휘들을 활용해 개별 - 개별, 개별 - 는항관계 표현
- 필수역, 수의역 + 명령, 소유, 양보 등과 같은 다양한 관계 표현
(: name - 개체명
: wiki - 텍스트에서 정의되지 않은 개체명)

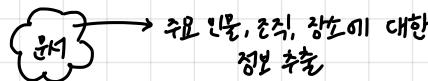
Part 2. 자연어처리 응용시스템

ch 8. 개체명 인식 (named entity recognition) - NER

: 텍스트에서 유한 의미의 개체를 인식하는 기법

⇒ 사방 (PS), 장소 (LC), 기관 (OG), 날짜 (DT)

라 같은 명명된 개체를 텍스트로 식별



예) 대첩은 1597년 (DT) 9월 16일 (DT) 때 이순신 (PC)이
지휘하는 수군 13척이 명량 (LC)에서 공격했다.

8.1 NER 시스템

① 지도학습기반

: 머신러닝 모델에 샘플 데이터 학습 → NER 예측값

- 은닉 마르코프 모델 (HMM)

- SVM

- 전부 우작위장 (RF)

② 지식기반

- 이후 자동/도메인별 지식에 의존 → 도메인 전문가의 필요성

③ 비지도 / 부트스트랩 기반

(bootstrap: 데이터셋 내 문맥이 고르지 않은 경우

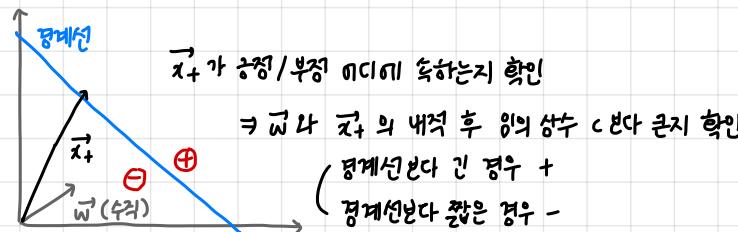
⇨ 랜덤 샘플링으로 training data 수 늘리기)

SVM (Support Vector Machine)

: 서로 다른 2개의 그룹 나누는 기법

⇒ 결정 경계를 그어서 그룹으로 분류하는 역할

↳ 결정 경계를 정하는 규칙 필요

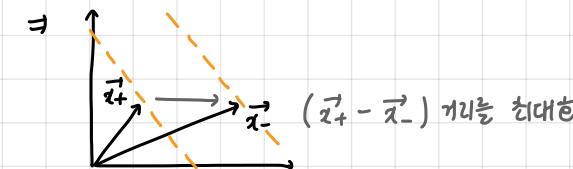


$$\Rightarrow (\begin{array}{l} \vec{w} \cdot \vec{x}_+ + b \geq 1 \\ \vec{w} \cdot \vec{x}_- + b \leq -1 \end{array} \text{ 어디에 속하는지 확인})$$

↳ 양 변에 y_i 곱해주기

$$y_i(\vec{w} \cdot \vec{x}_i + b) - 1 \geq 0$$

↳ 목표: 최대 거리를 갖는 결정경계를 정하는 것



8.2 NER 평가척도

(텍스트에 따라 엔티티와 상관없이 예측된 레이블이 맞는지

" 레이블과 상관없이 예측된 엔티티가 맞는지

⇒ F1 score 사용

$$F_1 = \left(\frac{2}{\text{recall} + \text{precision}} \right) = \text{recall} \text{과 precision의 조화평균}$$

① 정밀도 (precision)

$$= \frac{\text{시스템이 올바르게 예측한 엔티티 수}}{\text{시스템이 예측한 수}}$$

② 재현율 (recall)

$$= \frac{\text{시스템이 올바르게 예측한 개체 수}}{\text{사방이 식별한 수}}$$

8.3 BIO Tagging Scheme (태깅 기법)

: 개체명을 텍스트로부터 인식하는 기법

- 정보 추출 (information extraction)에 사용

(B (Begin) 개체명 중 시작을 나타내는 단어에 태그

 I (Inside) B/I 개체명 뒤에 오는 단어 태그

 O (Outside) 개체명이 아닌 나머지 단어에 태그

New York → New (B-LOC), York (I-LOC)

8.4 학습 corpus

- CoNLL 2002 / CoNLL 2003 ⇒ 뉴스와이어 기사에 개체명 태깅
- SemEval (Drug NER shared task) ⇒ 의약품 개체명 태깅
- CHEMDNER ⇒ 화학 분야의 개체명 정보
- Twitter

Ch 9. 언어 모델 (language model)

: 언어를 이루는 구성 요소 (글자, 형태소, 단어열) 등에 학습값 부여
→ 다음 구성 요소 예측 / 생성하는 모델

① 통계적 언어 모델 (statistical LM, SLM)

: 학습에 기초

② 딥러닝 언어 모델 (DNN LM)

: 인공신경망에 기초

9.1 통계적 언어 모델

: 단어들이 가지는 확률 분포를 기반으로 각 단어의 조합 예측

· 목표: 실제로 흔하게 사용하는 단어열 (문장)의 분포를 정확하게 근사

예) 자동완성 기능

↓
'전부 학습' 기반

$$P(A, B, C, D) = P(A) P(B|A) P(C|A, B) P(D|A, B, C)$$

∴

단어 $A, B, C, D \Rightarrow$ 단어의 조합 예측

이를 이용해서 n 개의 단어 (w_1, \dots, w_n) 결합 확률:

$$P(w_1, \dots, w_n) = \prod_{i=1}^n P(w_i | w_1, \dots, w_{i-1})$$

(문장의 확률 = 이전 단어들이 주어졌을 때 다음 단어로 등장할 확률의 곱)

모델 훈련시, corpus 내 각 단어들의 조합이 나오는 횟수 기반 확률 계산

$$P(\text{먹는다} | \text{나는, 사과를}) = \frac{\text{count}(\text{나는, 사과를, 먹는다})}{\text{count}(\text{나는, 사과를})}$$

↳ 모든 계산을 진행하기는 비효율적, 마르코프 가정 사용

(마르코프 가정: 미래 사건에 대한 조건부

= 과거에 대해서는 특별, 현재 사건에만 영향 받음)

$$P(w_1, \dots, w_n) \approx P(w_1) P(w_2 | w_1) \cdots P(w_n | w_{n-1})$$

9.2 n-gram 언어 모델

단어의 등장 확률 = 바로 이전 단어와만 연관되었다고 하면

멀리 떨어진 단어와의 의존관계 (= 장기적 의존성) X

⇨ 특정 단어는 주변 n 개의 단어와 연관된다고 가정

(N 개의 단어열 = N -gram)

예) "The boy is looking at a girl"

1. 1-gram (유니그램, unigram) - The, boy, is, ...
2. 2-gram (바이그램, bigram) - The boy, boy is, ...
3. 3-gram (트라이그램, trigram) - The boy is, ...
4. 4-gram - The boy is looking, ...

∴ 특정 단어가 등장하는 확률 계산시 이전 $N-1$ 개의 단어 등장 확률 고려
(= $N-1$ 차 마르코프 가정)

1-gram (0개, 시로 특별적)

$$P(w_1, \dots, w_n) = \prod_{i=1}^n P(w_i)$$

2-gram (전부 1차)

$$P(w_1, \dots, w_n) = \prod_{i=1}^n P(w_i | w_{i-1})$$

:

$$N\text{-gram } P(w_1, \dots, w_n) = \prod_{i=1}^n P(w_i | w_{i-N}, \dots, w_{i-1})$$

N 값에 따라서, (작은 N : 부자연스러운, 세로운 문장 생성)

(큰 N : 자연스러운, 이미 존재할 확률이 높은 문장 생성)

↳ 리소스가 많아지며 과부하

- 가장 편한은 $N=3$

- $N=5$ 이상은 예측효과 ↓, 리소스 ↑ (Perplexity 값 증가)

- SLM (통계적 언어모델) + 큰 N 의 N -gram 언어모델

⇨ 희소성 문제 (sparsity problem)

↳ 새로운 문장이 생성되지 않는 단점

⇨ 딥러닝 기반 언어모델 사용 / SLM에는 일반화 (generalization) 적용

9.3 난그 확률(log probabilities)

(언어 모델에서 확률 계산시 대부분 log를 취한 값 사용)

→ 언더플로우(underflow)를 피하기 위함

· 언더플로우 : 부동 소수점 연산에서 컴퓨터가 표현 가능한
값보다 작은 값 발생

N-gram LM식:

$$\log P(w_1, \dots, w_n) = \sum_{i=1}^n \log P(w_i | w_1, \dots, w_{i-1})$$

9.4 일반화(Generalization)

→ 단어를 하나도 없으면 0이 되는 문제
즉, 통계적 언어 모델의 '회소성 문제'를 해결하는 방안

(corpus의 크기를 키우는 것만이 해결방안은 아님,
한정된 corpus 내에서 모델의 일반화 능력 향상)

① 스무싱(Smoothing)

: corpus에 존재하지 않는 단어 조합에 특정 값(α , 데스并不是很) 부여,
확률 분포에 변화를 주는 기법

→ 모든 단어를 일정한 값 가짐

→ corpus에 없는 문장의 확률 0이 되는 문제 해결

$$P(w_i | w_{<i}) = \frac{\text{count}(w_{<i}, w_i) + \alpha}{\text{count}(w_{<i}) + \alpha V}$$

· $w_{<i}$: i번째 단어 이전에 등장하는 모든 단어

· V : 어휘 사이즈 (=corpus의 단일 단어의 개수)

<라플라스 스무싱> (Laplace smoothing)

: 0값을 1로 치환, 모든 단어를 1을 더하는 방법

- 제1노 레이터(corpus에 없는 단어들)이 적은 경우 유용

- 제1노 레이터에 특정값(α)을 더한다고 일반화 문제 해소 X

↳

Interpolation(보간법) / Backoff(백오프) 사용

9.5 Interpolation & Back off

① Interpolation(보간법)

. 특정 N-gram 확률 + 이전 N-gram 확률과 섞어서 사용

ex) 3-gram 언어 모델 → 이전 2개의 단어 +

$$\left(\begin{array}{l} 2\text{-gram (1개 단어)} \\ 1\text{-gram (특별적)} \end{array} \right) \times \text{일정 가중치 } \lambda \text{로 계산}$$

$$\hat{P}(w_n | w_{n-1}, w_{n-2}) = \lambda_1 P(w_n | w_{n-1}, w_{n-2}) + \lambda_2 P(w_n | w_{n-1}) + \lambda_3 P(w_n)$$

- 라플라스 스무싱 = 모든 제1노 레이터에 똑같은 빈도 수 부여

- 보간법 = 제1노 레이터의 N-gram 정본에 따라 다른 빈도 수 부여

② Back off(백오프)

: corpus에 3-gram 확률 있다 → 3-gram

3-gram 없고 2-gram 확률 있다 → 2-gram

1-gram 확률만 있다 → 1-gram

$$\hat{P}(w_i | w_{i-2}, w_{i-1}) = \begin{cases} P(w_i | w_{i-2}, w_{i-1}), & \text{if 3-gram 빈도수 } \neq 0 \\ \alpha_1 P(w_i | w_{i-1}), & \text{if 2-gram 빈도수 } \neq 0 \\ \alpha_2 P(w_i), & \text{otherwise} \end{cases}$$

9.6 모델 평가 & Perplexity

모델이 test dataset에 얼마나 잘 적합하는지 평가척도 계산

① 모델간 비교

⇒ Perplexity (PPL) (헷갈리는 정도)

: 모델이 test data에 대해 확률 분포를 얼마나 정확하게 예측할 수 있는지에 대한 내부 측정 지표

- 낮을수록 성능 ↑

<퍼플렉시티 계산>

= 모델이 선택할 수 있는 경우의 수 (분기계수)

(branching factor)

$$PPL(W) = P(w_1, \dots, w_N)^{\frac{1}{N}}$$

(후본군에 대한 확률의 역수를 후본군 개수로 정규화)

· W: 문장 ↗

· N: N개의 단어

$$= \left(\frac{1}{P(w_1, \dots, w_N)} \right)^{\frac{1}{N}}$$

$$= \left(\prod_{i=1}^N \frac{1}{P(w_i | w_1, \dots, w_{i-1})} \right)^{\frac{1}{N}}$$

↑ 확률 계산시 로그 확률 사용

예) 2-gram 언어 모델의 PPL

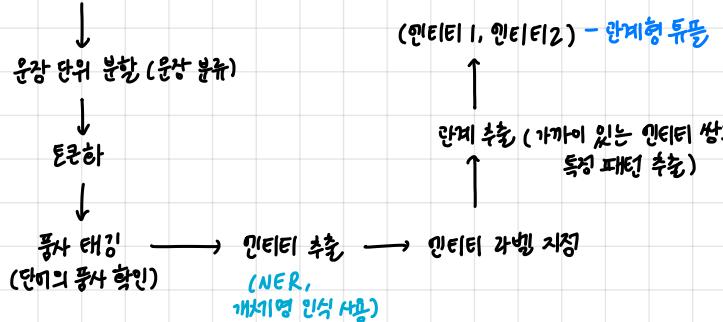
$$PPL(W) = \left(\frac{1}{\prod_{i=1}^N P(w_i | w_{i-1})} \right)^{\frac{1}{N}}$$

ch 10. 정보 추출 (information extraction)

- 비정형 데이터로부터 유용한 정보를 자동으로 추출
- 엔티티간의 의미적 관계 \Rightarrow 관계형 투플 표현
(엔티티1, 엔티티2)

10.1 정보추출 학습방법

텍스트 데이터 입력



예) 서울에 있는 고려대학교: ([ORG: '고려대학교'], 'in', [LOC: '서울'])

〈정보추출 학습작업〉

① NER (개체명 인식)

: 단지된 엔티티의 이름, 장소, 숫자 등 특정 유형의 개체에 대한 참조 식별

② co-reference Resolution (상호 참조)

: 이전 단계에서 추출된 엔티티 유형 기반 \rightarrow 엔티티간 상호 참조와 관련된 용어 찾기

③ Relation Extraction (관계 추출)

: 텍스트 내 엔티티간 관계 식별 \rightarrow 구조화된 정보로 나타냄

10.2 관계추출 (RE)

: 각각의 엔티티에 연결하는 엔티티 유형 함께 끌어 \rightarrow 관계 유형 추출

- 컨퍼스에 있는 관계추출 방법론 사용

(TACRED \Rightarrow TAC KBP (Tac Knowledge Base Population))

(관계 유형 4개 per: schools-attended
(정의된 관계가 없는 경우 no_relation))

- 관계 투플 형식 사용

(엔티티1, 관계, 엔티티2)

10.3 정보 추출 접근법

1. 규칙기반

: 수작업으로 전문 도메인 지식 활용 \rightarrow 규칙 생성 (관계/정보 추출)

2. 기계학습 (ML) 기반

: 기계학습 알고리즘을 이용해서 규칙 수동 추출 (도메인 전문가 필요)

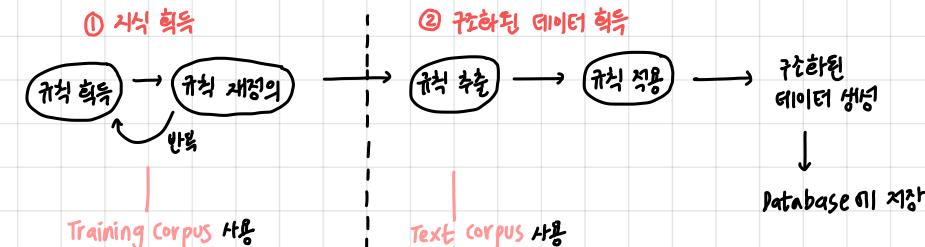
① 규칙 기반 접근법

: 사방이 규칙 생성 \rightarrow 관계 투플 (규칙) 데이터에서 정의

\rightarrow 정의된 규칙 사용해서 비정형 데이터에서 관계 투플 추출

단점: - 유지 보수 작업 어려움 (비효율적)

< 반자동화 규칙기반 접근법 프로세스 >



1. 규칙 학습

: 도메인 전문가에 의해 공식화된 규칙 학습 (규칙 일반화)

2. 규칙 재정의

: 주어진 규칙 사용 \rightarrow 조정 + 세분화 과정

3. 규칙 적용

: ① 지식 학습 단계에서 정의한 규칙 기반 \rightarrow 구조화된 데이터 생성

- 속성-값 형식으로 생성 (attribute-value)

4. 구조화된 데이터 생성

: 속성-값 쌍이 키-값 output

② 기계학습(ML) 기반

장점: 패턴을 기계가 학습해 자동적으로 데이터 추출

⇒ 효율적

(ML 기반 접근법 프로세스)

1. 주석 달린 데이터 학습
2. 기계학습 알고리듬 기반 학습

(HMM, 최대 엔트로피 크델, 조건부 벤딩 필드(CRF), Naïve Bayes, Decision Tree)

기계학습 접근법 (① 감독 학습(supervised learning))

(② 비지도 학습(unsupervised learning))

① 지도 감독 학습(SL)

: 입력 데이터 - 이어나는 텍스트를 포함하는 학습 데이터 셋에서 정답 예측으로 구축

⇒ 알고리듬이 입력 문서에서 분류하는 능력을 학습

주제 단어 추출 기반의 패턴 추출:

① 출역에 대해 가능성 있는 모든 텍스트 세그먼트 감지 (identify text segment)

② 후보 텍스트 세그먼트에서 정본추출 때 가장 유용한 세그먼트 선택 (select relevant information)

예) The hurricane Aomi in Korea hurt 2,000 people.

↳ ① identify text segment

The hurricane Aomi in KOREA hurt 20,000 people.

↳ ② select relevant information (정규표현식 사용)

(Event place : KOREA)

(Affected people : 20,000)

· 관계추출을 위해 (관련성 있는지)

(관련성 있는지 분류 (지도 학습 기반))

- 후보 텍스트 세그먼트 ⇒ 이후 문맥 (lexical context)에 따라 분류

⇒ 관련성 없는 텍스트 세그먼트 삭제 &

관련성 있는 텍스트 세그먼트 주요 정보로 추출

· CRYSTAL ⇒ 전문가가 학습한 라벨링 문서 학습에 사용

→ 이전에 경험된 가장 유사한 쌍 / 규칙 기반 정본(관계) 추출

② 비지도 학습(USL)

: 라벨 없는 데이터에서 중개된 구조 찾고 입력 때면 생성방법 학습

Ch 11. 질의응답 (Question & Answering)

- 분야: 무인 상담 시스템, 자동 응답 chatbot, 기계 통해

III.1 정보검색 기반 질의응답

(IR based question answering)

① 질문처리단계 (question processing)

: 질문 분석 (question analysis)을 통해
 (질문유형 분류
 정답유형 분류)

② 문서처리단계 (document retrieval)

: 정답을 포함하고 있는 문서/문서 검색
 → 관련성 높은 문서 추출

③ 정답처리단계

(정답후보추출 (answer candidate extraction): 정답 후보 개체 추출

(정답순위화 (answer ranking): 추출된 정답후보 풍차)

→

- 문장의 의미적 유사도 (semantic similarity)
- 정답유형 - 후보의 의미적 연관성 (semantic relation)
- 키워드 충복성 (co-occurrence keyword)

iii. 기반해서 순위 매김

(전체 질의응답 프로세스)

질문처리단계 —————→ 문서처리단계 —————→ 정답처리단계

- 질문유형 분류
- 정답유형 분류
- 질의재생성

- 문서 검색
- 정답 후보 추출
- 정답 순위화



[각 단계별]

① 질문처리단계 (question processing)

- 질문유형 분류 ⇒ 의문사에 기반해서 구분
- 각 질문유형에 따라 정답유형은 제약된다

의문사	매번	정답유형
누구		인물/경의
몇	몇 + N	날짜/시간
무슨	무슨 + N	N 의미에 의존
어느	어느 + N	"
무엇	시+무엇	"
어디	무엇 + N	위치/경의
언제		날짜
얼마		숫자
왜		이유
하여/여지하		방법/경의/이유

- 정답유형 분류 ⇒ 질문유형에서의 의문사 정보 +
 개체명 태그 + 형태소 분석 결과 활용

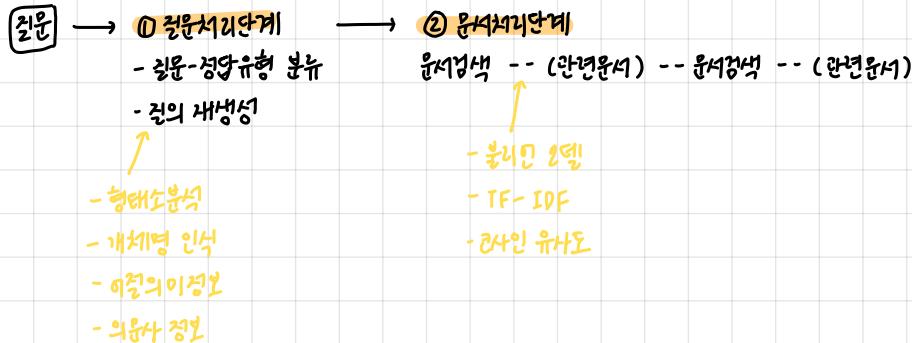
② 질의재생성 (query reformulation)

- 정답이 포함되어 있는 문서를 정색하기 위해 질문 내
 주요 키워드 / 매번 인식 → 질의 생성

(문서처리단계에서 문서 검색이 내용됨)

- 즉, 형태소 분석, 개체명 인식(NER), 의문사 정본,
 어절정보, 어휘의미 정보
 ⇒ 질문유형 후 정답유형 분류에 사용됨

② 문서처리단계 (document processing)



문서검색에 활용되는 오델 :

① 불리언 (boolean) 오델

: 각 단어가 수집된 문서에서 출현했는지 여부를 boolean 형으로 나타냄
(나타남=1, 나타나지 않음=0)

- document term matrix 를 출현여부 이/1로 표시

나 기다연 네이밍 차운어

문서 | 0 1 1 0

- AND / OR 를 사용해 특정 단어가 출현하는 문서 검색

- 어떤 문서가 더 중요한지 등의 순위 측정 불가

↓

② 벡터공간 오델 (vector space model)

: 각 문서에 나타나는 단어들에 가중치 부여

→ 가중치에 따라 질의어 - 문서의 유사도 측정 → 문서 순위 부여

· 유사도 측정방법:

 (자카드 유사도 (Jaccard similarity))

 (코사인 유사도)

 TF-IDF

① 자카드 (Jaccard) 유사도

: 두集合 A, B 의 합집합에서 교집합의 비율 이용

(0: 두집합의 공통 원소 없음)

(1: 두집합이 동일함)

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}$$

문장과 비교할 문서는 (Question, Docu, 이면)

$$J(Question, Docu) = \frac{Question \cap Docu}{Question \cup Docu}$$

- 간단하지만 단어빈도수 (TF), 문서 빈도수 (DF)를 고려하지 않음

② TF-IDF

1. TF (Text Frequency)

: 단어가 문서에 나타난 횟수 (단어의 중요도 파악 가능)

2. DF (Document Frequency)

: 해당 단어가 나타난 문서 수

(DF가 높은 단어 = 인동을 유기에서 나타남 → 문서검색시 중요도 높음)

IDF (Inverse DF)

$$\log \left(\frac{\text{전체 단어 수}}{\text{해당 단어 DF}} \right)$$

(범위를 축하는 Log 역할)

(IDF가 높은 단어 → 중요도 높음)

$$TF-IDF = TF + IDF \text{의 합}$$

(낮으면 중요도 낮음)

$$tf(d, t) = \text{특정 문서 } d \text{에서 특정 단어 } t \text{의 등장 횟수}$$

$$idf(d, t) = \log \left(\frac{n}{1 + df(t)} \right)$$

∴ 문서 검색에서 Boolean / vector space 오델 활용 ⇒ 관련 있는 문장 / 문서 추출

③ 정답 처리 단계



① 정답 후보 추출

질문 처리 단계에서의 정답 유형 정보



정답 유형 개체명에 따라

문서 처리 단계에서 관련 문장 / 문서 탐색

질문

정답 유형

관련 문서 → 형태소 분석 / 개체명 태그 분석 진행

정답 후보



정답 유형과 같은 개체를 갖는 것을 정답 후보로 추출

② 정답 후보 순위화

- 1) 정답 후보가 정답 유형과 일치하는가?
- 2) 질문 분석에서 얻은 키워드가 정답 후보에 존재하는가?
- 3) 정답 후보 - 재생성된 질의와의 유사도?

동의 기준을 갖고 순위화

→ 순위 가장 높은 후보 출력

ch 12. 기계번역 (machine translation)

12.1 규칙기반 기계번역

- 원본 문장의 형태소/ 구문 분석 → 핵심 단어를 목표로 인어 번역
→ 문법적 규칙에 맞게 언어 대응기
- Vauquois Triangle에 따라서 규칙기반 기계번역 과정 설명 가능



: 분석 수준에 따라서
 ↗ 차집 번역
 ↗ 통사구조 전달
 ↗ 의미구조 전달은 불가능

- 단점:**
 - 시스템 생성 전 필요한 언어학적 지식의 양이 많음 (복잡성↑)
 - 운행에 맞춰 1:1 대응을 통해 문장 번역 → 단어/구의 연결이 자연스럽지 않음
- 장점:**
 - 문법적인 부분에 충실 → 전통 번역 문장의 정확성
(하지만 문장 전체 번역하는데는 부분을 강조하는 일이 어려움)

12.2 통계 기반 기계번역

: 대량의 예제들 (코퍼스)을 바탕으로 2개의 언어사이의 상관관계를 통계적으로 분석한 모델 생성 → 주어진 문장 번역

<규칙기반 기계번역과의 차이점>

① 코퍼스(발용치) 필요함

- 코퍼스를 이용해서 기계학습 진행
- 2개의 언어가 쌍으로 연결 (언어1 - 언어2)
- 코퍼스의 크기가 클수록 Bleu score ↑ (번역 평가 높음)

② 모델

- 기계학습 기반의 모델 알고리즘 등장

<통계기반 번역의 수학적 원리>

- 입력 언어 (Source) = 영어 (E) ↗ E-F의 병렬 코퍼스 사용
- 출력 언어 (Target) = 프랑스어 (F)

① 조건부 확률

$$P(F|E) = \frac{P(F \cap E)}{P(E)} = \frac{\text{count}(F, E)}{\text{count}(E)}$$

: 영어 문장 (E)에 대한 프랑스어 문장 (F)의 확률

→ 모든 문장별로 조건부 확률 계산

→ 확률이 가장 높은 프랑스어 문장 선택

· 하지만 무한한 문장에 대해 확률을 계산할 수 없음



문장 단위의 확률도 계산 (= 단어 확률들의 곱)

$$P(F|E) = \prod_{E_j} \text{argmax}_{F_i} P(F_i|E_j)$$

: 입력 영어 문장의 각 단어마다 확률이 가장 높은 (argmax) 프랑스어 문장 선택

→ 기존 문장 단위 > 단어 단위로 이루어지므로

(0의 확률 피하기 좋음)

(단어 단위로 더 정확한 확률계산 가능)

② 번역의 절 지표화

문서 만든 조건부 학습을 기반 기계번역도 쉽지 않다

- 단어·의미는 항상 1:1 매칭이 아님
- 인접 단어의 번역 (의용사/ 관계형) 모두 다름
- 인접별 미순이 다름

↓
조건부 학습 + 베이즈 정리 활용

$$\begin{aligned} F_{\text{best}} &= \underset{F}{\operatorname{argmax}} P(F|E) \\ &= \underset{F}{\operatorname{argmax}} \frac{P(F) P(E|F)}{P(E)} \quad) \text{ 베이즈 정리 사용} \\ &= \underset{F}{\operatorname{argmax}} P(F) P(E|F) \end{aligned}$$

12.3 + 기반 번역

(Phrase-based Model) \Rightarrow 단어 단위 X, + 단위로 번역하는 방법

Ch 13. 자연어 생성

· 자연어 생성이 어려운 이유?

① 입력 ≠ 자연어인 경우 (image인 경우)

→ 입력과 출력 사이의 정보 불균형

② 운방적 특성

< N-gram 기반 자연어 생성 >

: 이전 단어가 주어졌을 때 다음에 오는 단어 예측하는 대 할용

예) 동해물과, 백두산이, 아르고,

$$\prod_{k=1}^n P(x_t \mid \text{context}) = P(x_t \mid x_{t-n+1}, x_{t-n+2}, \dots, x_{t-1})$$

: 다음에 오는 단어 = '동해물'

< 인용신경망 기반 자연어 생성 > (NNLM)

$$\hat{P}(x_t \mid \text{context}) = P(x_t \mid x_{t-n+1}, x_{t-n+2}, \dots, x_{t-1})$$

/ 다음에 등장할 단어

지도 학습 (supervised learning)

강화 학습 (reinforcement learning)

복대 학습 (adversarial learning) 활용

[13.1] 지도 학습 기반 자연어 생성

① 최대 우도 추정 (maximum likelihood estimation, MLE)

- 순차적 다중 레이블 분류 (sequential multi-label classification) 활용

→ 다중 레이블 크로스 엔트로피 최적화

$$J_\theta(s_n) = -\sum_{t=0}^{n-1} \log \hat{P}(x_t \mid s_t)$$

/ s_0 : 빈 문장

- 장기적 의존성 (long-term dependency) 피하기 불가

② 순환 신경망 언어 모델 (RNNLM)

- 장기 의존성 문제 해결

- 가변적 길이의 이전 입력값 = 은닉 (hidden) 벡터로

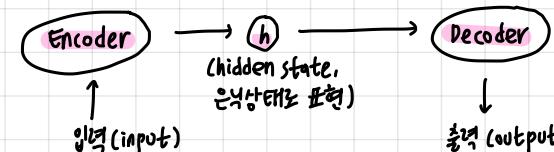
자동회귀적 (auto-regressive)으로 사용 → 다음 단어 예측

$$\hat{P}(x_t \mid \text{context}) = P(x_t \mid \text{RNN}(x_0, \dots, x_{t-1}))$$

/
기존 NNLM과 다르게 RNN 구조 사용

$$J_\theta(s_n) = -\sum_{t=0}^{n-1} \log \hat{P}(x_t \mid \text{RNN}(s_t))$$

/
우적함수 MLE ⇒ RNN 구조 추가



(Encoder: 입력운영 → 은닉 상태 (hidden state)로 표현)

(Decoder: 은닉 상태 → 문장 (자연어) 생성)

- RNNLM 학습 MLE 우적함수를 통해서 입력 레이어 풍포 학습

⇒ 노출 편향 (exposure bias) 발생

(노출 편향: 자동회귀적 생성 모델이 학습 과정에서 정답 (ground-truth) 문맥 정보 전달 받음

추운 대정에서 자신이 생성한 정보 참조

⇒ 학습 환경 - 추운 환경 간의 불일치 현상)

- 생성하는데는 유통이 길어질수록 MLE의 effectiveness ↓, 노출 편향 ↑

- 경사도 소실 (gradient vanishing) 문제 발생 → 장기 의존성 문제

(경사도 소실: RNN에서 tanh 힘소를 사용해 기울기가 지수적으로 감소,

일정 수준 이하로 감소 시 매개변수가 갱신되지 않음)

\downarrow
노출 편향 + 기울기 소실 문제는 해소하려는 시도

① 스케줄링 (schedule sampling)

② 트랜스포머 (transformer)

① 스케줄 샘플링 (schedule sampling), SS

↳ 노출 편향 문제 해소

- 자동화적 생성 모델 (RNNLM)이 학습 과정에서도
추운 과정을 미리 경험해 학습 - 추운 과정의 차이를 줄이는 방법

· 샘플링 (sampling)

- 자신이 생성한 샘플 → 다음 샘플의 입력값으로 제공

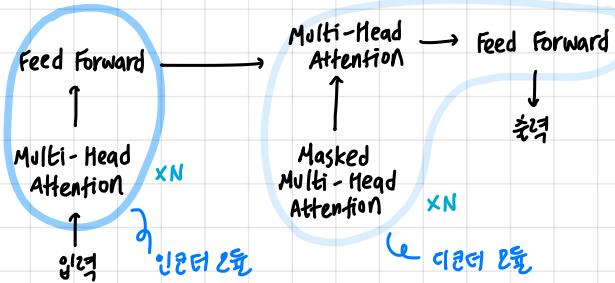
학습

정답 정보 제공받음 통해 학습 진행

(샘플링 학습 = 학습이 진행됨에 따라 점차 감소)

② 트랜스포머 (transformer)

- 셀프 어텐션 (self attention) 모델 \Rightarrow 입력 병렬 처리해 n 표현
 \leftrightarrow RNN: 순환 처리



13.2 강화 학습 기반 자연어 생성

① PG-BLEU

(강화 학습: 이산 (discrete) 행동 후 최고의 보상을 주는 행동을
에이전트 (agent)가 선택하는 학습방법)

· PG (Policy-gradient)

= REINFORCE 같은 강화 학습 정책-기울기 알고리즘

+

BLEU 지표를 이용해 대표되는 기본 불가능한 평가 지표 추적화

(PG 알고리즘)

: 정책을 직접 모델링 + 가치학습 목표로 함

(정책 = $\pi_\theta(\alpha|s)$)

- 보상 (reward) 함수값은 정책에 의존적
- 최고의 보상을 만드는 θ (정책 파라미터) 학습화

< BLEU score > (Bilingual Evaluation Understudy) - 평가 지표

$$\text{BLEU} = \min\left(1, \frac{\text{output length}}{\text{reference length}}\right) \left(\prod_{i=1}^N \text{precision}\right)^{-N}$$

$$N\text{-gram precision} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}}$$

기반으로 계산

(N-gram 시퀀스 중 정답 문장에 포함된 시퀀스 비율)

$$\text{precision} \times (\text{문장 길이에 대한 패널티}) = \text{BLEU score}$$

↳ 너무 짧은 문장에 대해서 너무 높은 BLEU score X

하지만, PG-BLEU만을 사용하면 손발에 학습이 잘 진행되지 않는 문제점 발생

\Rightarrow 강화 학습 + 가치학습을 융합한 필요성 존재

\Rightarrow 모방 학습 (imitation learning)

② 강화 학습 (imitation learning)

- 강화 학습이 사용되는 한반기 지도 학습 결합
- ⇒ 강화 학습의 약점을 지도 학습의 강점으로 보완
(사용되는 지도 학습 방법 = MLE)

$$MLE \Rightarrow J_\theta(\hat{s}_n) = - \sum_{t=0}^{n-1} \log \hat{p}(x_t | s_t)$$

MLE를 통해 선형 학습 진행 시에 봄난 날뛰는 역할 수행

13.3 적대 학습 기반 자연어 생성

① 생성적 적대 네트워크 (Generative Adversarial Network), GAN

- 2개의 경쟁양이 서로 경쟁 관계에서 학습하는 기계학습 모델

△ 판별기 (discriminator)

생성기 (generator)

· 생성기 목표

- 실제 데이터와 흡동될 정도로 사실적 출력을 인위적으로 제조

· 판별기 목표

- 입력값이 실제 데이터인지 / 생성기에서 만든 것인 데이터인지 판별

⇒ GAN의 핵심은 판별기의 성능을 따라가며, (생성기) - 더 사실적인 데이터 생성
(판별기) - 데이터 전위 판단에 더 능숙

GAN 프로세스



(학습 = 판별기 성능으로 구해진 손실에 따라 생성기 & 판별기의 기울기 전달)

1) 생성기 ⇒ 생성된 데이터가 판별기에 의해 구분된 기댓값 최소화

$$\min V(G) = E(\log(1 - D(G(z)))$$

2) 판별기 ⇒ 실제 데이터 구분에 대한 기댓값 + 생성된 데이터 구분에 대한 기댓값

$$\max V(D) = E(\log D(x)) + E(\log(1 - D(G(z))))$$

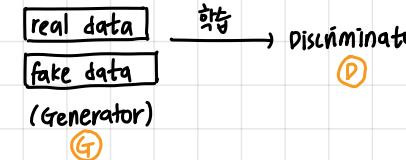
- GAN은 연속적인 데이터를 처리, 이산적 데이터에 잘 작동하지 않음

② seqGAN (Sequence Generative Adversarial Networks)

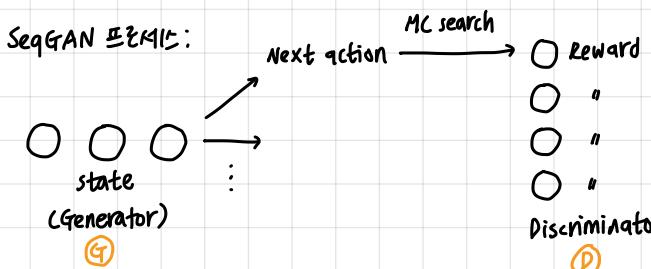
- 이산적 데이터 후보군 중 최고의 번장을 원도록 학습하는 강화 학습 기법

< seqGAN 프로세스 >

① 기본의 GAN 프로세스:



② SeqGAN 프로세스:



· 생성기 = LSTM 모델 사용

· 판별기 = CNN 모델 사용

· 생성기 & 판별기의 선형 학습 (pre-training) 실행

(생성기: MLE

판별기: 실제 데이터와 생성기로 만든 데이터 (negative sample)

구분하도록 크로스 엔트로피 손실 계산

→ 생성기는 만든 문장 행동 - 가치 함수 (action-value function) 계산

→ 정책 경사도 (policy gradient)는 생성기 파라미터 θ 업데이트

→ 판별기의 파라미터 업데이트

13.4 GAN의 문제점 및 해결방안

① 경사도 소멸 (gradient vanishing)

· SeqGAN에서 생성기의 비해 판별기의 성능이 강력한 경우 생성기 학습 X

⇒ (판별기 경수 scaling 하는 MalIGAN 모델

(경수를 순위에 따른 값으로 변환하는 RankGAN 모델)

② 모드 붕괴 (mode collapse)

· 생성기가 판별기를 두터 가중 구운 output만 반복적으로 생성 → 다양성 저하

⇒ 장기의 훈련 파악할 수 있도록 정본 유도하는 leakGAN 모델

Ch 14. 대화 시스템 (Dialog system)

- (사용자 주도 대화시스템)
- (시스템 주도 대화 시스템)

14.1 대화 시스템 프로세스

(시작)

Speech → 음성 인식 (speech recognition) → 자연어 이해 (NLU)
 텍스트화 ← 의의 분석

(끝)

Speech ← 음성 합성 (text-to-speech synthesis) ← 자연어 생성 (NLG)
 텍스트화 ← 의의 분석

① 자연어 이해 시스템 (NL Understanding)

: 입력된 텍스트 데이터 내용에 따라 슬롯 (slot)을 만들어 채우는 형식

② 대화 관리 (Dialog Manager)

: 대화의 문맥, 흐름, 적용 모델, 생략 부록, 중의성 해석, 예의처리 담당
 ↳ 대화의 문맥 분석 + 사용자가 원하는 것 파악

③ 자연어 생성 시스템 (NL Generation)

: 목적에 따라 자연어 생성 (=답변 생성)

④ 음성 합성 시스템 (Text-to-Speech synthesis)

: 사방과 유사하게, 듣기 편한 음성으로 변환하여 전달

〈대화 관리 시스템 방식〉

- (규칙 기반 접근법)
- (아이터 기반 접근법)

14.2 규칙 기반 대화 관리 시스템

① FSA (Finite State Automata)

: 모든 경우의 수에 대해 규칙 정하기 → 대화의 문맥, 순서 표시

→ 대화 (speech)가 들어오면 미리 정해둔 규칙에 따라 질문 수집 → 사용자에게 서비스 제공하는 방식

② Frame 기반

: frame이라는 규칙 + 규칙간의 우선순위 존재

→ 사용자에게 질문하며 문맥을 이해하는 형식

예) 사용자 주도형 Frame 기반 방식

시스템 : 기다려면 → Frame
 사용자 : 기다려면 정보 표시
 인물 정보 : 기다려면

시스템 : 나이 → Frame
 사용자 : 기다려면의 나이 표시
 인물 정보 : 기다려면
 요청 항목 : 나이

14.3 아이터 기반 대화관리 시스템

① 강화학습 기반

강화학습? : 순차적인 학습 과정을 위한 문제 해결 과정
 = Markov Decision Process (MDP)

MDP의 구성요소:

- 에이전트가 관찰 가능한 상태
- 에이전트가 특정 상태에서 할 수 있는 행동
- 환경이 에이전트에게 주는 정보
- 에이전트가 어떤 상태에서 어떤 행동을 해서 다음 상태에 도달할 확률
- 보상을 구분할 수 있게 해주는 강가율

∴ 강화학습은 시행착오 (trial & error) 등 통해 보상을 최대화하는 방향으로 행동

14.4 대화시스템 평가

① Slot Error Rate

: 문장이 슬롯 filling을 잘 했는지 평가하는 지표

$$\text{Slot Error Rate} = \frac{\# \text{ of inserted/deleted/substituted slots}}{\# \text{ of total reference slots}}$$

② End-to-end evaluation (Task success)

: 대화가 잘 이루어졌는지 평가하는 지표

예) "Make appointment with Zoey at 11:00 a.m. in Room 101."

Slot	Filler
Person	Zoey ✓
Time	12:00 a.m. ✗
Room	101 ✓

① Slot error rate = 1/3

② End-to-end evaluation = 마지막에 성확한 답변이 전송되었는지?
(= 최종적으로 이루어진 task를 success 했는지?)

14.5 대화시스템 분류 (목적에 따른 분류)

(기능대화
일상대화)

① 기능대화 (task-oriented dialog)

: 사용자가 원하는 goal + 시스템과의 대화를 통해 목적이 결과로 나타나는 바탕
= Task completion

- 대화의 목적은 상황에 따라 다이나믹하게 변화

- domain-specific 한 특징

(대화주제에 따른 도메인이 명확하게 구분됨)

- 명확한 답이 반드시 존재함

② 일상 대화 (social conversation)

: 대화의 '대화' 자체가 목적

- open-domain 한 특징

< 기능대화 VS. 일상대화 >

기능대화	일상대화
목적이 분명함	대화 자체가 목적임
domain-specific	open-domain
1개의 답	여러 개의 답

ch15. 문서 요약 (Text summarization)

- 텍스트의 의미는 유지하면서 텍스트의 내용을 간략하게 줄여주는 것
(source text → summary text)

문서요약

- I (interpretation)**: 문서 텍스트 (source)를 해석해 컴퓨터가 이해할 수 있도록 표현
(문장 표현 sentence representation) → 문서 표현 (document))
- T (transformation)**: 문서 표현을 요약문으로 표현될 수 있도록 변형 / 가공하는 것
- G (generation)**: 요약문에 대한 표현으로부터 최종 요약문 생성

15.1 문서요약 방법

입력 문서, 목적, 출력 문서에 따라 분류 가능:

입력문서	목적	출력문서
1) 사이즈 (단일 문서 다중 문서)	1) 사용자 (일반적인 질의 의향적인)	1) 타겟 (우호 축정)
2) 도메인 (domain-specific general)	2) 사용 흥도 (새로워 유익한)	2) 평판 (충정 평가가 필요한)
3) 형태 (구조, 규모, 매체, 장르)	3) 학제성 (배경 지식 최신 주제)	3) 형식 (고정된 유동적인)

① 입력 문서

① 문서의 크기 (=사이즈)

- 단일 문서 요약: 핫뉴스나 다른 리소스는 사용하더라도 하나의 입력문서만 처리
- 다중 문서 요약: 다수의 문서를 입력으로 사용

↳ 같은 주제에 대해 고통적인 문서 사용시 충복성 문제 야기

② 문서의 도메인 (domain)

- 특정 도메인에 대한 요약
- 일반적인 도메인에 대한 요약
- 입력 문서와 동일한 도메인의 요약 생성하는 경우.
 - 용어에 대한 오해 (ambiguity)
 - 문법 사용, 세미 채마 사용 줄일 수 있음

예) 바이오메디컬 분야의 텍스트 요약

③ 형태

구조 (structure), 규모 (scale), 매체 (medium), 장르 (genre)

기반 다양한 형태를 가질 수 있음

1) 구조 (영시적인 구조)

예) 문서를 서론-본론-결론으로 나누어서 문서 요약 진행 ⇒ 성능 ↑

- Let Summ 시스템
 - 법률 주제에 따른 구조에 의해 요약 생성
 - 각 구조마다의 중요 기여율 계산 → 조합해서 최종 요약문 생성
 - HTML 문서 양쪽 문서로 사용
 - 각 파트에 대해 추출 → 문장 구성 → 정수화 (score) → 요약 생성

2) 규모 (입력 텍스트의 길이)

- 큰 규모: 뉴스 기사거리, 책
- 작은 규모: 트윗 (tweet), 손전기의 블로그

3) 매체

- 요약 시스템에 정보를 전달하는 역할
- 예) 텍스트, 비디오, 음성 등

4) 장르

- 예) 뉴스, 인터뷰, 리뷰, 소설 등
- 각 장르마다 알맞은 요소 추정

② 목적

① 사용자

- 일반적인 요약 시스템 (generic): 문서 작성자가 유도한 흐름에 맞춘 시스템
- 질의 의향적 (query-oriented) 시스템: 사용자의 의향을 고려

예) 사용자가 문서 요약시에 '특정 사람'에 집중하고 싶다면 해당 사람을 기준으로 흐름을 이끌어야 함

↳ 유사한 문장끼리 클러스터 (cluster) 구성

→ 질의에 대한 문장간 정수 계산

→ 높은 정수 갖는 문장으로 요약문 생성

② 사용 용도

- (사용자의 관심 분야를 결정하기 위해 도와주는 추단 (사사적))
 - (원본 문서를 대표하는 요약문으로 대체 (유용적))

1) 유용적 요약문 (informative)

: 원본 문서의 필수적 정보 담고 있음

- 예) 저널, 연구 논문, 학회 논문에서의 요약 (abstract)

2) 사사적 요약문 (indicative)

: 유익한 정보 X, 원본 문서의 전반적인 설명만 포함

- 문서의 목적, 범위, 연구 방법 포함 → 원본 문서를 참고할지 알지 못함

- 예) 책에서 어떤 주제에 대한 요약, 네포트

③ 확장성 (expansiveness)

- (원본 문서의 배경 (background) 까지 포함)
 - (일부 과거 문서와 비교해 허신 소식 제공)

1) 배경 요약

: 상황에 대한 장소, 시간, 관련된 행위자에 대한 설명 포함

2) 허신소식 제공

: 새로운 문서에서의 새로운 주제를 포함

- 이전 문서에 나타난 내용인지 구별하기 위해 '사전 지식 (prior knowledge)' 필요

④ 축약 문서

① 파생 (derivation)

- (추출 (extraction)): 통계적 특징 고려 → 추출 문장 정렬 (가장 두드러진 문장 찾기)
- (추상 (abstraction)): 같은 의미의 다른 표현 (paraphrase) / 새로운 문장 생성

② 편파성 (partiality)

: 충실히 판단 / 의견 포함하지 않고 요약

- 평가가 필요한: (특서적 판단 = 특정 주제만 포함)
 - (명사적 판단 = 적정적 판례)

③ 형식 (conventionality)

(고정)

유동적

(자동요약시스템 개발)

15.2 통계적 접근법

- 문서의 주요 주제 / 사용자의 요청에 의거한 일부 자료 이용해 텍스트간 연관성 측정 계산
 - 척도 측정의 문장들을 요약문에 사용

① 풍이 빈도 (term frequency)

TF-IDF (term frequency-inverse document frequency) 사용

$$idf(t) = \log \frac{1}{|d/tIND| + 1}$$

$|d|$: 커퍼스 D에서의 문서들의 개수
 $|d/tIND|$: t를 포함하고 있는 문서들의 개수

② 텍스트의 위치

- 1번째 / 마지막에 등장하는 문장이 다른 문장들보다 중요도가 높음 (Luhn score를 통해서 증명)

$$(\text{sentence})$$

$$(\begin{bmatrix} * & * \\ 1 & 2 \\ 3 & 4 \\ 5 & 6 \\ 7 \end{bmatrix})$$

⇒ 그중으로 나누어서 진행

*: 중요한 (Luhn score가 높은) 문장

- 단어의 위치 = 단어 끝을 앞에 나타날수록 더 유익함

⇒ 전체 문장 내에서의 단어 위치를 파악할 수 있음

⇒ 4가지 정의된 함수 사용 (DP, IP, GS, BF)

1) Direct Proportion (DP)

: 처음 출현 : 1점

(마지막에 출현 : $1/n$ 점 (n : 문장 내 단어 개수))

2) Inverse Proportion (IP)

$$= 1/i \quad (i: 위치 번호)$$

- 작은 위치에서 정도가 빠르게 감소 (선행되는 문장 위치가 유리)

3) Geometric sequence (GS)

: 단어에 대한 점수

$$= 모든 단어에 대해 $(1/2)^{i-1}$ 들의 합$$

4) Binary Function (BF)

: 첫 등장 : 1점

: 다른 경우: $\lambda < 1$ 점

- 첫 등장에 중요한 가중치 부여

이렇게 계통적으로 계산시,

$$\text{Score}(s) = \sum_{w_i \in s} \frac{\log(\text{freq}(w_i)) * \text{pos}(w_i)}{|s|}$$

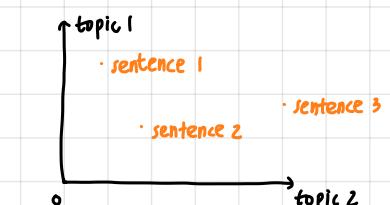
• $\text{pos}(w_i)$ = 4가지 함수 중 하나

• $\text{freq}(w_i)$ = 문장 s 내 단어 w_i 의 빈도

• $|s|$ = 문장 s의 길이

③ 잠재 의미 분석 (latent semantic analysis, LSA)

: 문서를 각 단어들에 포함된 용어들 간의 관계성 탐색 (Topic) (sentence)



<LSA 알고리즘>

① 행렬 A 생성

($A =$ 문서내 용어들을 m 개의 행, n 개의 열로 나눈 행렬, $m \times n$ 행렬)

② 행렬 A의 특이값 분해 (SVD)

$$A = U \Sigma V^T$$

- $U = m \times n$ 의 직교 대각행렬의 열 (column) \downarrow

= 원쪽 특이 벡터

- $\Sigma = \text{diag}(z_1, \dots, z_n)$

= $n \times n$ 대각 행렬 (diagonal matrix)의 대각 원소

- 특이값 벡터로 구성

- $V = n \times n$ 직교 행렬의 열

= 오른쪽 특이 벡터

“특이값?”

① 고유값 & 고유벡터

$$\begin{bmatrix} A \\ \vdots \\ A \end{bmatrix} \begin{bmatrix} v \\ \vdots \\ v \end{bmatrix} = \begin{bmatrix} M \\ \vdots \\ M \end{bmatrix}$$

영화 행렬

$$Av = \lambda v$$

(λ = 고유값, 벡터 v = 고유벡터)

- 고유값을 알면 $(A - \lambda I)v = 0$ 를 이용해 고유벡터 v 를 구할 수 있음

② 특이값

A 가 정방행렬 ($m=n$)인 경우: $A = v \lambda v^{-1}$

A 가 정방 행렬이 아닌 경우 ($m \neq n$): $A = U \Sigma V^T$

특이값 분해: 행렬 A 를 3개의 행렬로 표시하는 방법

③ $A = U \Sigma V^T$

$$\begin{bmatrix} A \\ \vdots \\ A \end{bmatrix}_{m \times n} = U_{n \times n} \times \Sigma_{n \times m} \times V^T_{m \times m}$$

〃

④ 문장 k 의 중요도 계산

$$s_k = \sqrt{\sum_{i=1}^n v^2 k_i \cdot z^2}$$

[15.3] 기계학습 접근법

① 물질 향수

: 특정 용장이 고약문이 포함될지 안될지 결정하는 함수

예) 베이지안 분류기 \rightarrow 확률값을 구해서 분류 문제로 구분 (포함 O
포함 X)

[15.4] 평가

① ROUGE (recall-oriented understudy of Grading Evaluation)

: 정답요약 (reference summary) & 모델이 생성한 요약 비교

- 모델이 생성한 요약

- 정답요약

사용

- recall값에 대한 단위 (n-gram) 개수

- 고려하는 n-gram의 길이 = N일 때

ROUGE-N 을 할 수 있음

$$\text{ROUGE}-N = \frac{\text{후반 요약, 정답 요약에서 찾은 n-gram 개수}}{\text{정답 요약에 있는 n-gram 개수}}$$

\downarrow
n-gram의 사이즈

$$= \frac{\text{Count match (n-gram)}}{\text{Count (n-gram)}}$$

ch 16. 텍스트 분류 (Text Categorization)

: 문서를 입력받아 사전에 정의된 클래스 중 어떤 분류하거나
구집화하는 과정

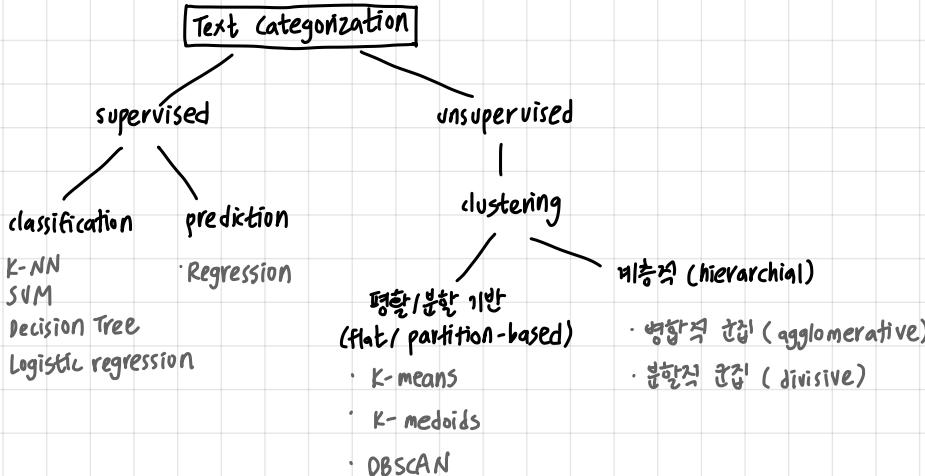
① 분류 (classification) = 지도학습

: 항목에 맞춰 정수화하는 작업

② 군집화 (clustering) = 비지도학습

: 항목 간 유사한지에 따라 스스로 분류되는 작업

(ML에서의 텍스트 분류 방법들)



16.1 감정분석 (sentimental analysis) (= 오피니언 아이디)

: 텍스트에 나타난 사람들의 태도, 의견, 성향과 같은 감정 분석

예) 소셜 미디어 분석, 영학평론 분석

- CNN, RNN ⇒ 감정 분석 모델

- NSMC ⇒ 네이버 영화 감정 분석 라이브 데이터

· 감정 사전을 이용 구축해놓는 것이 중요함

16.2 텍스트 분류 메커니즘

① 카테고리 / 의도 분류

(카테고리 / 도메인 분류 = 글이 어떤 카테고리에 속하는지?)

(의도 분류 = 어떤 의도를 가지고 하는 것인지?)

② 스팸 행 분류

: 메일이 스팸인지 아닌지 분류

16.3 텍스트 분류 프로세스

<학습 과정>



<분류 과정>



① 라이터 준비

② 라이터 전처리 (NLTK 패키지 사용)

- 불용어 제거

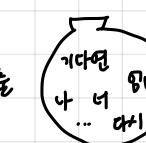
- 특수기호 제거

- 정규화 (normalization)

③ 특징 값 추출

1) Bag of words

: 단어의 순서와 관계없이 충현빈도를 추출



2) TF-IDF vectorizer

: 단어의 빈도, 즉 문서 빈도 사용 → 각 단어마다 중요도 가중치로 부여

예) - 문서 유사도 구하기

- 검색 시스템에서 검색 결과 중요도 정하기

- 문서 내 특정 단어 중요도 구하기

$$idf(d, t) = \log \left(\frac{n}{1 + df(t)} \right)$$

tf(d, t) : 문서 d에서 단어 t의 개수
df(t) : t가 등장한 문서의 수
idf(d, t) : df(t)의 반비례

3) 추가적인 특징값 추출방법

· Dict Vectorizer

: 각 단어 수를 세어놓은 사전에서 BOW 생성

· Count Vectorizer

: 문서 집합에서 단어 벡터 생성 → 각 단어의 수만큼 BOW 벡터 생성

· Tfidf Vectorizer

: TF-IDF를 가중치로 향후 BOW 벡터 생성

· Hashing Vectorizer

: 해시 함수(hash function)으로 BOW 벡터 생성 (=작은 메모리 + 빠른 시간)

④ 텍스트 분류 알고리즘으로 학습

⑤ 모델 평가

(Precision, Recall, F-1 score)

16.4 텍스트 분류 알고리즘

① Naive Bayes 알고리즘

: 베이즈 정리: 모든 특성을 사이에 독립성 가정

$$P(y|X) = \frac{P(X|y) P(y)}{P(X)}$$

(X 가 발생했을 때 y 가 발생할 확률 구할 수 있음)

(y = 클래스 변수)

(X = 애개 변수 = (x_1, \dots, x_n) — 특성)

$$P(y|x_1, \dots, x_n) = \frac{P(x_1|y) \dots P(x_n|y) P(y)}{P(x_1) \dots P(x_n)}$$

$$\approx P(y) \prod_{i=1}^n P(x_i|y)$$

→ 최대 확률을 구하기 (argmax)

$$y = \operatorname{argmax}_y P(y) \prod_{i=1}^n P(x_i|y)$$

: 다중분포 NB 분류기 = 문서 분류 문제 (스포츠, 정치, 기술 등의 문서 범주)

: 베르누이 NB 분류기 = 예측하는 변수가 boolean 변수 (True / False)

- 텍스트가 단어가 나타나는지 True / False로 표시

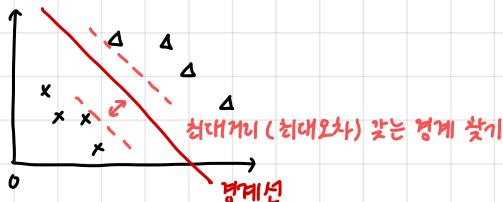
: 가우시안 NB 분류기 = 예측 변수가 연속적인 경우

② SVM (Support Vector Machine)

: 주어진 데이터 집합에서 새로운 데이터가 어떤 카테고리에 속하지 판단하는 비학습적 이진 선형 분류 모델 생성

→ 가장 큰 틈을 갖는 경계를 찾음

(가능한 최대 오차(margin)는 두 개의 클래스 간의 거리)



③ KNN (K-Nearest Neighbor)

: 차연학습 알고리즘 → 분류하는 동안 모든 데이터 사용

(비교수적 학습 알고리즘 → 기본 데이터에 아무런 가정도 하지 않음)

④ 결정 트리 (Decision tree)

: 의사결정 규칙과 트리구조 찾는 방식

- 데이터 마이닝에서 자주 사용

⑤ 경사 하강법 (Stochastic Gradient Descent, SGD)

: 매개변수 증가, 감소 조정 → 손실함수 최소화하는 방향으로 학습

⑥ The Random Forest Algorithm

: 여러 개의 결정 트리를 임의적으로 학습하는 양상을 학습방법

16.5 텍스트 군집화 알고리즘

① k-평균 (k-means)

: 데이터를 k개의 클러스터로 묶는 알고리즘

→ 각 클러스터의 거리 차이 분산을 최소화하는 방식

- 분할법 (partition-based) 군집화 방법

- 비용함수: 각 그룹의 중심 (centroid)과 그룹 내 데이터의 거리의 제곱합

→ 최소화하는 방향으로 진행

- 목적함수:

$$V = \sum_{i=1}^k \sum_{j \in C_i} \|x_j - m_i\|^2$$

↳ 데이터 — 중심의 제곱합

대부분의 기계학습 기반 데이터 모델을 다루기 위해선 scikit-learn 라이브러리 사용:

① 차원 학습 - NB, DT, SVM 등

② 비지도 학습 - 군집화, 이상치 탐지 등

③ 모델 선택 / 평가 - 교차검증, 과이프라인 등

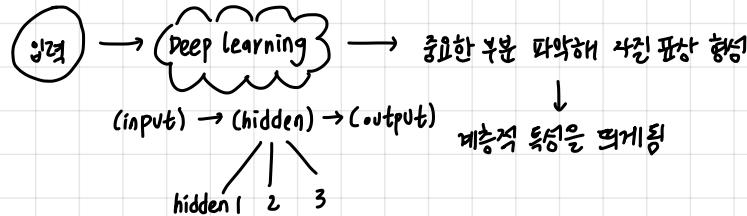
Part 3. 딥러닝 기반 자연어 처리

Ch 19. 딥러닝 소개

- 입력 + 출력에 대한 이상적인 출력 정의 → 입력을 출력으로 자동적으로 변형하는 방법 등등
- = 통한간 학습 (end-to-end learning)

19.1 차원적 계층적 자료 표상 습득 (feature representation)

- 자질 표상
- 특성을 잘 수행할 수 있도록 도우는 입력의 특징들



19.2 데이터 및 모델의 구조

- 학습에 사용되는 데이터
 - ① 지도학습을 위한 것
 - ② 비지도학습을 위한 것
 - auto-encoder (입력과 출력 동일)
 - language modeling (입력의 상대적 위치 이용)
 - 유통 순서 예측

데이터가 학습 된 이후에는 입력, 출력의 특성에 맞게 모델 설정

- 1) 입력, 출력 = 단일 (정점)
 - ⇒ 완전연결층 (affine) / 합성곱 신경망 (CNN)
- 2) 입력, 출력 = 가변
 - ⇒ 순환신경망 (RNN)

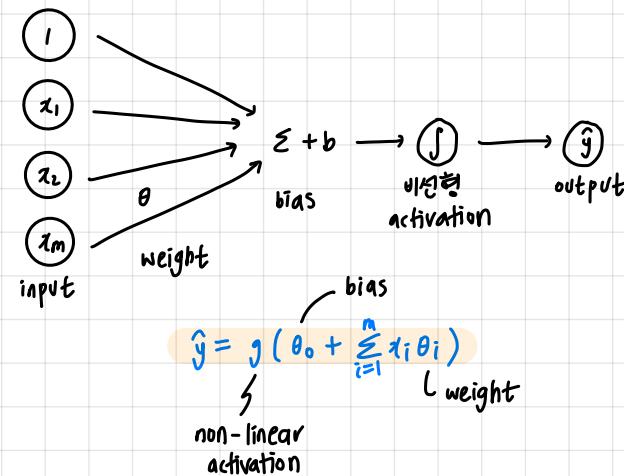
19.3 퍼셉트론 (perceptron)

(입력) \rightarrow 각각의 입력값 \times 가중치 + 편향 \rightarrow 1차 출력

(m:n)

↓
 비선형 활성화 함수
 (non-linear activation fn)

출력 \leftarrow



19.4 활성화 함수

① sigmoid (시그모이드) 함수

$$g(z) = \frac{1}{1+e^{-z}}$$

- 출력 0~1 사이의 값

② tanh (하이퍼탄젠트) 함수

$$g(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$

- 출력 -1~1 사이의 값

③ ReLU (Rectified Linear Unit) 함수

$$g(z) = \max(0, z)$$

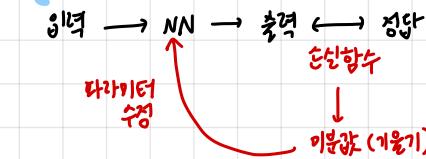
$$g'(z) = \begin{cases} 1, & z > 0 \\ 0, & \text{otherwise} \end{cases}$$

- 계층의 출력값에 따른 통제는 sigmoid / tanh 같다. 멀티층
- 이론값이 계산을 천할하게 수행하도록 도와줌 → 가장 널리 쓰임

19.5 딥러닝 모델 학습

- 목표: 최적의 파라미터 값을 찾는 과정
 feed-forward

← back-propagation



19.6 훈련 함수 (loss function)

- 모델의 예측과 정답 사이 수치화하는 함수
- 각각의 파라미터에 대해 편미분
- gradient 계산
- 계산된 gradient에 따라 파라미터 조정
- 반복 적용

gradient 계산방식?

(편미분 특성상 입력층의 gradient는 한번에 계산불가)

⇒ 출력층 ~ 입력층 순차적으로 gradient 계산

⇒ '역전파법' (back-propagation)



딥러닝 학습에 - 미분값 (gradient)

- 활성화 함수의 영향 높음

18.4 문장 단위 임베딩

: 단의이 구별 문제를 해결하기 위해,

같은 단어가 입력으로 세너하더라도 문맥에 따라 다른 벡터로 임베딩

① ELMo (Embedding from Language Models)

: 문장을 경방향으로 예측하는 LM로 학습해서 얻은 벡터

+

문장을 역방향으로 예측하는 LM로 학습해서 얻은 벡터

- 입력 = 단어 단위 X

문자 단위 → 학습시 없던 단어에 대해서도 임베딩 가능

- 단어 임베딩 모델의 역할만 수행

(단어 벡터만 만들뿐, 실제 자연어 처리는 다른 모델이 수행)

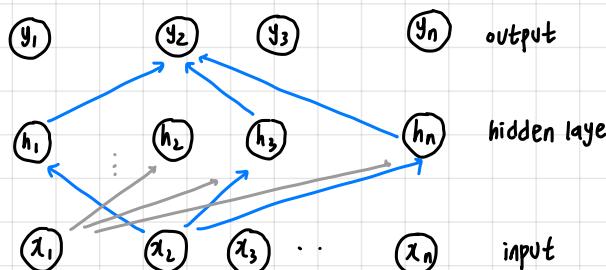
② BERT (Bidirectional Encoder Representations from Transformers)

: 전제: 필요지식은 대부분 배운 바탕을 통해 학습



학습 완료된 BERT 모델 → 최소한의 파라미터 (단전연결계층 × 2) 추가

→ 자동학습 시 파라미터 포함된 BERT 모델 추가 학습



- 새로운 파라미터가 적은 경우 = 자동학습에 필요한 데이터 양 ↓

↪ BERT의 데이터 효율 ↑

(ELMo: 단방향 모델 2개 (정방향, 역방향) 결합 → 양방향 의존성 해결

BERT: 단일모델로 양방향 의존성 모두 학습

단일 모델로 양방향을 학습하는 경우,

입력으로 주어지는 것 = 예측하는 것의 문제 발생

⇒ 1) masked LM

2) permutation LM 사용

① masked LM

: 입력 문장 중 일부 mask → 원래 문장 예측

예) 나는 [?] 의 수도, 서울 출신이다.

→ ? 예측

② permutation LM

: 단어들을 무작위로 섞어 예측

(같은 단어를 예측하더라도 섞인 순서에 따라 다른 문맥정보 제공)

예) XLNet

: BERT보다 더 넓은 범위가 문맥에 포함

18.5 단어 임베딩 입력 단위

: 토큰 (token)

: 자연어처리 시스템의 입력 단위

- 입력 단위에 따라 모델 복잡도 증가

∴ 입력의 토큰 단위 선택 = 언어가 표현할 수 있는 토큰들의 상합 선택

(토큰의 수 ↓ = 한 마을을 표현하는데 더 많은 토큰 필요)

↪ 토큰의 수가 증가하면 성능이 저하하기에 적당한 토큰의 수 선택이 중요

< BPE (Byte Pair Encoding) >

: 통계 기반 토큰화 방법

(BERT에서 사용되고 있음)

: 문자 단위 축약 → 학습 corpus에서 등장빈도 높은 n-gram을 1개의 토큰으로 유기

→ 자동으로 토큰화 방법 학습

영어 (=analytic language)는 비교적 토큰화가 자유롭지만, 한국어 = 형태소 분석 우선 진행
형태소 분석이 제대로 진행되지 않는 경우: 오류 전파 (error propagation) 문제 발생

[18-1] 설습

: CNN을 이용한 문장 분류 모델 + Glove 단어 임베딩 모델 적용

· **Glove**: 단어 단위의 임베딩 기법 → 단어의 의미를 표현

(단어간의 동사 발생 비율 벡터공간에 표시)

- CNN ⇒ 문장을 여러 카테고리로 하나로 표현

① 품스 (politeness) 레이터 불러오기

(Stanford에서 제공하는 데이터)

② 데이터 불러오기 (load-data)

· read-csv로 데이터 파일 읽기

· 공손한 레이터 →
 (polite = 상위 25% (normalized score = 1)
 (impolite = 하위 25% (normalized score = 0)
 neutral (중립적) (normalized score = 2)

· 이후 '공손' = 1, '비공손' = 0으로만 바꿔하고 나머지는 제거

③ 사전 구성 (make-vocab)

· nltk.word_tokenize + .lower ⇒ 소문자는 바꾸기 + tokenize (토큰화)

· collections.Counter ⇒ 전체 데이터에서 각 토큰 등장 빈도 확인

word-counter = Counter()

word-counter.update (토큰화된 문장)

↳ 각 토큰을 count해서 몇 번 등장하는지 확인

· counter 중 most_common () ⇒ 등장 빈도 중 등장 빈도 높은 단어를
 vocab_size (5000) 만큼 선택

- 10개만 출력해서 확인해보기

· 각 단어에 고유번호 부여 ⇒ (0: PAD (哑字))

(1: OOV (out-of-vocabulary, 사전에 없는 단어))

④ 토큰 → 숫자 (인덱싱)

→ text-to-index

숫자 → 토큰으로 변환하는 사전 생성 → index-to-text

(text > index : word-index 함수 사용 ⇒ make-vocab에서 인덱스만 추출

index > text : word-inverted-index 함수 사용

*- 신경망 (CNN)이 입력에서 카테고리 분류를 하기 위해서는 텍스트 그대로 쓰면 안되고

숫자 (index)로 바꿔야 사용해야 하기에 이 작업 필수

④ 텍스트 문장 토큰화 → 숫자로 변경해보기,

학습 데이터로 사용 가능

4-1. 동일 길이 되도록 padding 기법 / truncating 기법

· tensorflow, python, keras, preprocessing, sequence ⇒ pad-sequence

- 8은 문장 sentence-size = 200으로 지정

(padding = 'post' (뒤에 pad 넣기)

truncating = 'post' (뒤에서 자름)

⑤ 훈련-테스트 데이터 분리 (90% = 훈련 data)

(10% = 테스트 data)

⑥ (레이터 전처리)

레이터셋 분리 했으니 이제 카테고리 분류에 쓰임 CNN 모델 구성 (model)

- tensorflow or keras 사용

1) keras.Sequential()

2) keras.layers.Embedding → 단어 임베딩 (공간 상태 표시)

3) keras.layers.Conv1D → 1D convolutional 계층

(activation = tf.nn.relu, ReLU function)

4) keras.layers.GlobalMaxPool1D → 1D 맥스 풀링 계층

5) keras.layers.Dense

(activation = tf.nn.softmax, 소프트맥스)

⑦ 2번 학습 (model.compile)

history = model.fit

- optimizer = Adam 사용

- 다른 validation data 구성하지 않고 바로 test data 사용

⑧ 학습 결과 시각화 (plot-loss), 성능 평가 (eval-loss)

plot-loss (val-loss = 훈련 데이터를 계산하는 loss)

(loss = 훈련 데이터를 계산하는 loss)

eval-loss ⇒ model.evaluate 사용해서 accuracy 계산

⑨ Glove 벡터 → 임베딩 행렬 초기화 (load-glove-embeddings)

- Glove ⇒ 각 단어별로 단어 (1번째 토큰)
(벡터를 이루는 숫자 (2번째 이후 토큰들) 끝나)

Glove의 내부모습:

the	0.1	0.2	0.3	0.4
a	0.02	0.04	0.01	0.04

단어 벡터를 이루는 숫자 → :

단어 벡터를 이루는 숫자 → numpy 행렬 모양으로 변환

1) 단어-벡터 이루는 숫자들간의 dictionary 구조 생성

embedding = {} (빈 dictionary 생성)

→ 각 단어별로 {단어 : 벡터 이루는 숫자} 구조

2) 단어별로 임베딩 행렬 무작위로 생성

- Stanford 공식 학습 레이블에서 glove (사전 학습)된 단어만 대체,
나머지는 그대로

(num_loaded = glove 사전 학습 벡터는 대체된 경우,

vocab_size = 전체 vocabulary 사이즈)

⑩ 기존의 단어 임베딩 모듈 (keras.layers.Embedding) 7/14

기존, 위에서 초기화한 임베딩 행렬을 사용

- keras.initializers.Constant (임베딩 행렬)

→ keras.layers.Embedding (embeddings_initializer = "I") 사용

∴ 기존에 glove 사전 학습된 임베딩 모듈을 사용하지 않는 경우다

사용하면 (NN모듈을 활용한 차례고리 분류가 더 학습이 잘됨)

[18-2] 설습

: BERT를 이용한 문장 분류

(Glove: 단어 단위 임베딩, BERT: 문장 단위 임베딩)

- BERT ⇒ 대부분의 지식은 바이트 학습을 통해서 얻을 수 있음
- 바이트 학습으로 사전 학습 완료된 토큰 + 천수한의 파라미터
→ 지도 학습으로 추가 학습

① 금속향 데이터 다운로드

(18-1 실습과 동일한 데이터)

- bert-tensorflow ⇒ BERT 사용에 필요한 허브 사용 가능

② sentiment treebank 데이터 다운로드

(Stanford 제공 데이터)

③ 토큰 생성 (create_tokenizer, from_hub_module)

⇒ Hub module에서 vocab_file (BERT의 사전학습된 단어) 가져오기

bert-module = hub.module(bert-model-hub)

- 둘 신자는 변경

- 토큰화(tokenize) 진행

④ 특징 추출 (make_features)

[label column] max-seq-length (최대 길이 한정)

[data column]
[]
[data column]
[]
[label column]

⑤ BERT 문장 분류 템플릿 생성 (create_model)

module 설정 (bert-module)

input 설정 (bert-inputs)

output 설정 (bert-outputs)
[문장 단위: pooled-output
[토큰 단위: sequence-outputs]]

→ model_fn_builder 허브는 실제 model 허브 생성

(기존에 정의한 데이터 없음 사용)

⑥ 성능 평가 (metric_fn)

⇒ 다양한 evaluation metrics (지표) 정의

tf.metrics.f1_score

tf.metrics.recall

tf.metrics.precision

tf.metrics.true_positive ~ false_negative

⑦ 학습하는 동안의 parameter 조정 + 학습기록

(estimator-builder)

→ run-on-dfs ⇒ 실제 data dataframe에 적용

⑧ tensorboard를 이용한 학습 결과 시각화

⇒ output + 허브 설정에 대한 다양한 시각화 자료 제공

ch 19. 합성곱 신경망 (CNN)

: 이미지 인식 / 분류 분야에서 주로 사용되는 기술

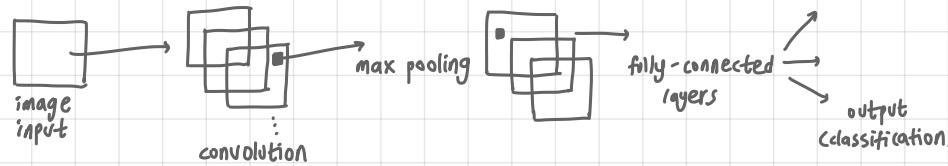
CNN의 구조?

① convolutional 계층

: 이미지의 특징 / 패턴을 잘 포착할 수 있도록 학습

② pooling 계층

③ 완전연결 (fully-connected) 계층



19.1 convolution 연산

: RGB 3개의 채널로 이루어진 이미지 → 디제털화된 이미지에 중요 요소 포함

주요 특징 / 거친 추출하는 작업

① 패딩 & 스트라이드 (padding, stride)

· 패딩 : 입력 데이터 주변을 특정 패턴으로 채우는 것 (대부분 0)

→ 출력의 크기 (output size) 조정할 목적, 정반대 압축되는 것 막음

· 스트라이드 : 전통적인 연산 진행시, 이미지를 2차원

방향으로 일자만족의 단계를 갖고 연산을 수행하지 않음

예) (이미지 입력 크기 = $N \times N$)

(필터 크기 = F)

→ 스트라이드에 따른 출력 크기 = $\{(N - F) / \text{stride}\} + 1$

- 완벽하게 나누어 떨어지지 않는 경우 = 스트라이드의 배수로 재조정

19.2 Pooling 연산

Convolution 연산을 통해 얻은 이미지 차집 → 풀링 과정

(추출된 모든 특징을 고려하지 못하는 단점 타파)

① max pooling

② average pooling

① max pooling (최대 풀링)

: 최댓값을 추출하는 풀링기법

② average pooling (평균 풀링)

: 평균값을 추출

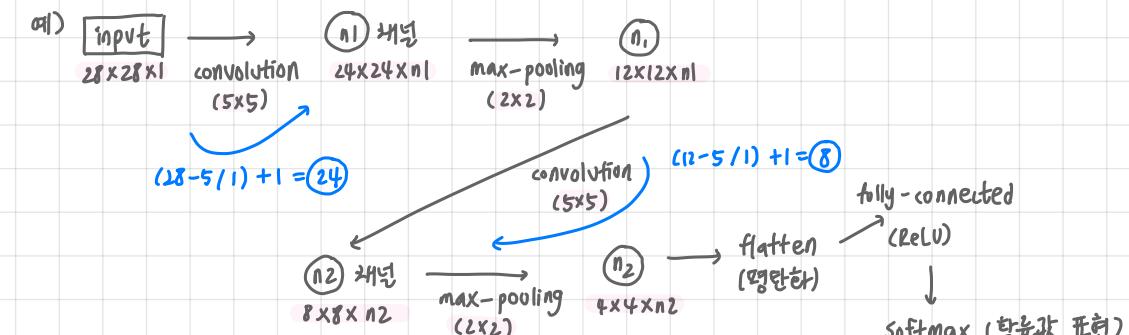
- 대부분의 경우 최대 풀링 > 평균 풀링 (이상치에 영향 X)

19.3 Fully-connected 계층

: Convolutional - Pooling 이후에 압축한 자료들을 fully-connected에 입력

→ 소프트맥스 교차 엔트로피 손실함수를 이용해 학습진행

· 전통적인 계층의 관계를 더 추가해서 구현할 수 있다.



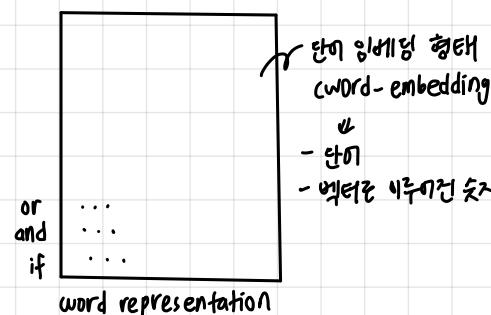
$\Rightarrow 28 \times 28$ 이미지 픽셀 → fully-connected 전 $4 \times 4 \times$ (필터의 개수)로 압축 가능

→ flatten (1차원 배열화) → 분류학습 진행

19.4 CNN을 이용한 문장 분류

CNN은 이미지 분류뿐만 아니라 순차정보가 있는 자연어 분류에도 사용할 수 있다.

CNN을 이용한 문장 구조



문장의 임베딩을 이용한 문장

행렬에서 수식 (J) 방향으로

→ convolution 차원 감소 → 차원 축소

↓
max-pooling → FC-layer →



2-class classification
(이진분류)

CNN 기반 문장 분류

① 단어 임베딩 (word embedding)

- 대부분 word2vec / glove와 같은 시전 학습된 단어 임베딩 사용
- 해서 초기화 → 웨이트 학습 시에 미세 조정 (fine-tuning)
- 단어 임베딩 2차원 행렬 = (전체 단어 개수, 임베딩 크기)
(언더스가 너무 크면 빈도가 높은 단어들이 대해서만 진행)
- (18-1 숫자 \Rightarrow 5000개의 고빈드 단어만 있어서 사용함)
 - 나머지 고빈드 = UNK (unknown)으로 대체해서 진행

CNN 모델의 입력

- = 문장의 단어 토큰들을 바탕해 이를 위한 단어 표현 (word representation) 행렬 사용
- 문장 [번째 토큰의 임베딩] = x_i

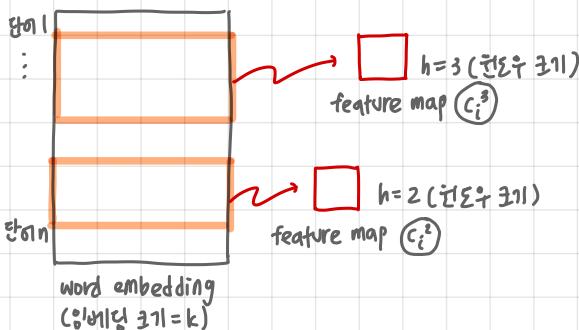
$$x_{1:n} = x_1 + x_2 + \dots + x_n$$

② 전결수선 (convolution) 연산

- 각 단어의 임베딩 방향 고정
- 우측 방향으로만 차원에 대해 슬라이싱 진행

$$c_i^h = \tanh(W \cdot x_{i:i+h-1} + b)$$

- 원도우 크기 = h
- 임베딩 크기 = k \rightarrow 차원 (c_i^h) 추출
- 전결수선 필터 = W
- 비선형 활성화 함수 tanh 사용



③ 패딩 (padding) 연산

- 목적: 문장에서 처음, 끝에 위치하는 단어들에 대해 전결수선하는 횟수를 다른 단어들과 동일하게 하기 위함
(처음, 마지막에 위치 \Rightarrow 상대적으로 연산에 포함될 확률이 적음)
- 원도우 크기 커우면, 연산을 하는 단어의 수 = $2 \times (\text{원도우 크기}-1)$ 만큼 늘어

④ 최대 풀링 & 원천 연산

convolution 계층에서 추출한 자질 \rightarrow 전결수선 필터별 최대 풀링 진행

$$c^h = (c_1^h, \dots, c_{n-h+1}^h)$$

$$\hat{c}_h = \max(c^h) - 최대값만 뽑아오기$$

원천연산 계층의 입력으로 들어가게됨

- 각 가중치 행렬 (weights), 편향 (bias)을 통해 분류 클래스 만큼
- 크기는 나운스팅 사용 (projection) 진행

$$\hat{y} = b (W \cdot \hat{c} + b)$$

비선형 함수 / 최대 풀링한 자질

예시처럼 이진 분류 (binary) 하는 경우, 목적함수 = 교차 엔트로피는 지정

$$\text{Loss} = - \sum_{i=1}^N y \log(\hat{y}) + (1-y) \log(1-\hat{y})$$

CNN을 통한 문장 분류: (장점)

- 병렬 처리 가능 \rightarrow 속도 개선
- RNN 사용시 이전 성능 다음에 넘김으로써 문장 길어지는 경우
문장 중간 성과 흐려지는 long-term dependency 문제
 \rightarrow CNN 사용시 문장의 순차성과 반영 + 부분 자질 (local feature) 반영 가능

[19-1] 실습

· 허깅풀신경망(CNN)을 이용한 이진 감정분석

- 아카운티뷰 레이터 사용

① 레이터 읽기 (read_dataset)

{
label = 부정 0, 긍정 1의 이진 감정 label
sentences = 나쁜 레이터 문장
tokenized_sentences = 토큰화된 각별 레이터

② 학습- 테스트 레이터 분리 (9:1)

(학습 = 레이터 원본에서 train.txt 사용
테스트 = 레이터 원본에서 test.txt 사용

③ 레이터 전처리

1) tokenize \Rightarrow CNN 모델에 넣기 전에 각 토큰 텍스트 \rightarrow 숫자화

- tokenizer.texts_to_sequences 사용

2) 모든 문장 같은 길이로 처리 (padding)

- tf.keras.preprocessing.sequence의 pad_sequences 사용

- padding = 'post' & maximum 길이에 일관 맞추기

④ CNN 모델 생성 (create_embedding_matrix)

· 19-1 실습처럼 Glove 사전 학습된 단어 임베딩 사용

- 학습 컴퓨터에 있는 단어들 중 Glove에 있는 단어들 = 초기화

CNN 모델 선언

(Embedding 계층 \Rightarrow 규제와 같은 embedding_matrix 사용

filter = [2, 3, 4, 5] 다양하게 사용

* filter의 크기를 다양하게 사용시 (local 성분

global 정보 모두 고려해 학습 가능

⑤ 학습 결과 시각화 + loss 계산 (plot-history)

(train loss, test (val) loss

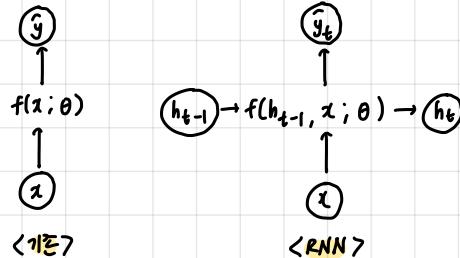
train acc, test (val) acc

Ch 20. 수학 신경망 (RNN)

: 드간 연결이 시간 순서에 따라 방향 그래프 (directed graph) 형성하는 신경망

〈기존 NN과의 차이점〉

(기준: 입력값 x , 파라미터 $\theta \rightarrow$ 출력값 y)
 RNN: 출력값 y 를 다음 연산 위해 h_{t-1} 전달, h_t 반환



RNN의 특징 :

- 1) 짧은 파라미터 수, 순환적 구조
 ↳ 길이 제한 없이 정복 처리
 - 2) 관계의 정복 기억하는 채로 현재 입력값 처리 → 문제 정복 방지
 ↳ 유익한 의존성 0

20.1 RNN 학습과정

\Rightarrow BPTT (back-propagation through time) 학습 방식 활용

$$\hat{y}_t = h_t = f(h_{t-1}, x_t; \theta) \\ = \tanh(W_{hh}h_{t-1} + W_{xb}x_t + b_h)$$

$$y_t = w_h \cdot x_t + b_h$$

$$\text{음수예측 } h_{t-1} := w_{ht} h_{t-1} + b_{ht}$$

$$\cdot b_h = b_{1h} + b_{2h}$$

→ [\[자료보기\]](#)

→ 활성화 함수 \tanh → 은닉 상태 h_t 와 \hat{y}_t 출력

→ 역전파 (BP) 를 통해서 RNN 학습

- BPTT: 손실에 대한 미분을 통해 역전파 수행시, 출력 부분까지 모든 time step이

적용되는 기초기 등수 계산

→ RNN 구조 특성상 입력 time step 수 = 같은 feedforward (FNN)이 가지는 step 수

∴ FNN과 같은 방식으로 학습

하지만, BPPT 방식으로 학습하는 경우 기울기 소실 (gradient vanishing) 문제 발생 (BPPT 과정에서 모든 신경망 학습시 tanh / sigmoid 함수 사용하기 때문)

$$\tanh(x) = \frac{1-e^{-x}}{1+e^{-x}} \quad \text{sigmoid}(x) = \frac{1}{1+e^{-x}}$$

→ 기울기 모두 같다 죠다 같다
선정망 계승을 거쳐면서 matmul 연산 반복시 전달값 정차 강도

∴ 기울기 소실 문제의 RNN을 해결하기 위한 방안 고안:

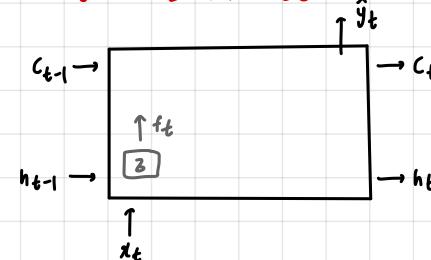
20.2 장단기 예측(LSTM)

- 장기 의존성 학습 가능한 memory model
 - 은닉 상태를 관리하는 셀 상태 (cell state, C) 존재

LSTM (tanh = 주된 정보 통과
 sigmoid = 주된 정보를 조정하기 위한 값 (0~1사이)
→ 입력값을 얼마나 통과시킬지 결정 (0: 통과 X)

LSTM 예시

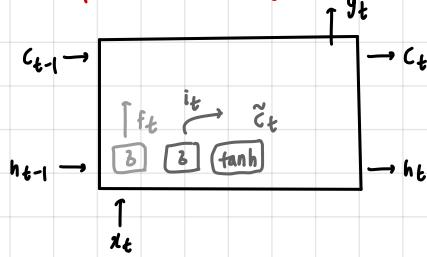
① 암각 캐이트에서 3 연산



- 양각 케이트 = 선단받은 정보를 셀 상태로부터 제거할 것인지 결정
 $x_t, h_{t-1} \rightarrow$ 합역 $\rightarrow (t-1)$ 에 대해 0이 있는지 안한 (sigmoid)

$$f_t = \sigma(W_f \cdot (h_{t-1} \cdot \pi_t) + b_f)$$

② input 케이트에서 sigmoid, tanh 연산



· 입력 케이트 = 셀 상태에 새로운 정보 추가한지 결정

(sigmoid(b) \Rightarrow 어떤 값 허락이트할지 결정)

($\tanh h \Rightarrow$ 새로운 후보 셀 상태 (\tilde{c}_t) 생성)

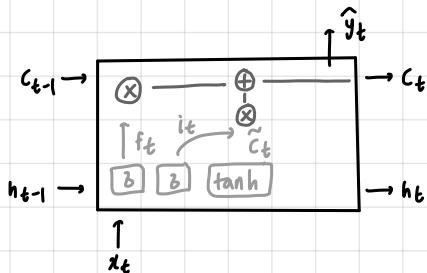
\Rightarrow 2개 연산의 결과로 상태에 대한 허락이트가 만들어짐

$$i_t = b(w_i \cdot (h_{t-1} \cdot x_t) + b_i) \quad - b는 어떤 값 허락이트할지?$$

$$\tilde{c}_t = \tanh (w_c \cdot (h_{t-1} \cdot x_t) + b_c) \quad - \tanh 는 새로운 후보 cell 생성 (\tilde{c}_t)$$

③ $c_{t-1} \rightarrow c_t$ 상태은 업데이트

: 양각 케이트, 입력 케이트에서 모든 값 활용



1) c_{t-1} 에 양각 케이트 층 출력값 f_t 곱 \otimes \Rightarrow 불필요 성보 양각

2) 입력 정보를 얼마나 봤자들일지 반영된 입력 케이트 층 출력값 i_t

곱 \otimes 후보 $\tilde{c}_t \rightarrow$ 셀 상태에 \oplus

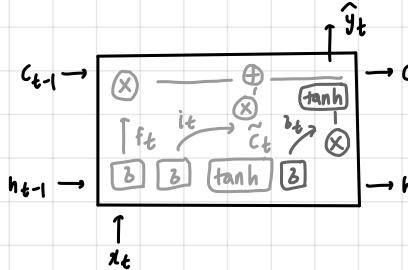
$$c_t = f_t \cdot c_{t-1} + i_t \times \tilde{c}_t$$

④ output 케이트에서 출력 결정

· 셀 상태 (c_t), 현재 입력값 (x_t) 함께 반영된 값 반환

1) $\tanh \rightarrow -1 \sim 1$ 사이값으로 변환

2) sigmoid (b) \rightarrow 셀 상태 어떤 부분 출역?



$$b_t = b(w_o \cdot (h_{t-1} \cdot x_t) + b_o)$$

$$h_t = b_t \times \tanh(c_t) \quad \rightarrow \text{현재의 은닉 상태}$$

2.3 케이트 순환 유닛 (Gated Recurrent Unit), GRU

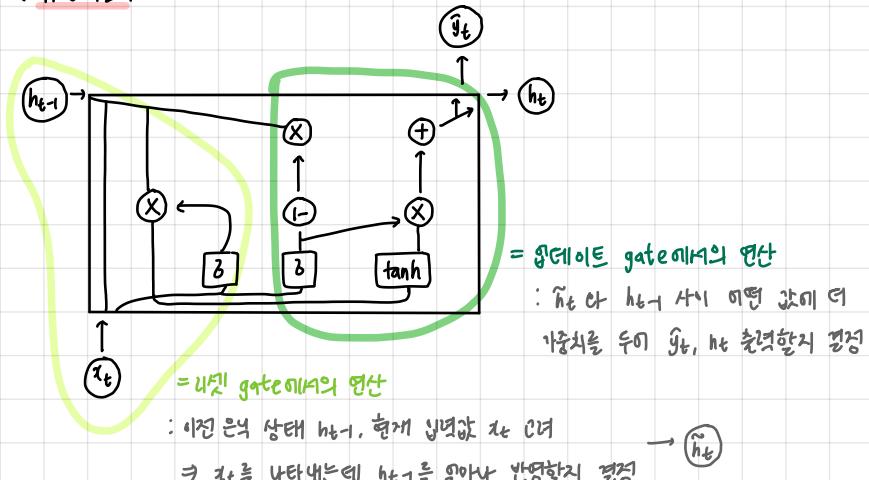
2개의 셀 gate 가짐 (리셋 케이트 (reset gate))

업데이트 케이트 (update gate)

① 리셋 gate : 현재 입력값에 이전 은닉 상태 얼마나 반영할지 결정

② 업데이트 gate (컨트롤러) · 양각 케이트, 입력 케이트 조정

<GRU 구조>



20.4 RNN 기반 자연어 생성

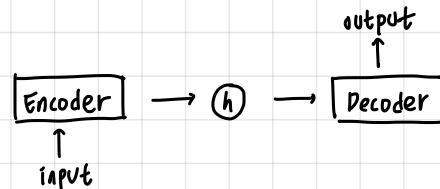
① seq2seq

: 입력된 순차 레이터(Sequence) → 다른 순차 레이터(Sequence) 출력하는 모델

- 자연어 생성 작업이 필요한 분야에 사용

(인코더(encoder) : 입력 레이터 $\rightarrow h$ (장치 표현)으로 인코딩

디코더(decoder) : 인코딩된 장치 표현(h) \rightarrow 출력 레이터 생성



- 인코더 + GRU / LSTM (장기 의존성 문제 없는) 네트워크 사용

- 디코더 \Rightarrow 인코딩된 장치 표현 + 스스로 이전에 생성한 단어 총합

\rightarrow 특정 시점에 가장 적절한 단어 반환

(맨 처음 단어 생성 (이전 단어X) $\Rightarrow <\text{SOS}>$ 토큰
맨 마지막 단어 생성 (이후 단어X) $\Rightarrow <\text{EOS}>$ 토큰

- 하지만 문장이 길어질수록 장기 의존성 문제를 완벽히 피해갈 수 X

\Rightarrow 어텐션(attention) 네트워크 + seq2seq

20.5 Attention + seq2seq

· RNN + seq2seq : 입력 문장에 대한 정보 \rightarrow 인코딩된 장치 표현(h)으로만 표현

(디코더가 출력시 어떤 정보 활용해야 할지 불분명

문장 길이로 정보 유실 문제↑

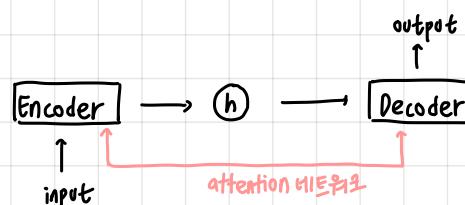
⇒ Attention + seq2seq

· attention : 현재 요청 (query)에 대해 주어진 값 중 어떤 값에 집중해야 하는지?

\rightarrow encoder ~ decoder 사이 연결

\Rightarrow 디코더는 입력단어에 대해 파악 가능, 어떤 단어에 집중해야 하는지 파악 가능

\therefore 장기 의존성 문제 해결



Ch 21. 딥러닝 기반 한글이 형태소 분석 후 품사 태깅 (POS-tagging)

1) 형태소 분석 : 입력 문자열 → 형태소열

나 + 는 + 기 + 다 + 연 + 이 + 다

2) 품사 태깅 : 형태소 분석 결과 T대로 태그를 붙이는 것

나 / NP + 는 / JX + ...

· 한글 = 고학년이기엔 1개의 어근 - 1개 이상의 접사가 올음

⇒ 어떤 단위의 형태소 분석이 불가함

< 형태소분석 / 품사 태깅 접근법 >

① 규칙기반

; 다양한 규칙 생성해 이를 따라 형태소 분석 → 품사 태깅

② 통계기반

; (1) 유의문장의 가능한 모든 형태소 후보 생성
(각 후보에 대해 품사 태깅 → 최적 품사 태깅 방법 선택
- KoNLPy 패키지)

KoNLPy ⇒ 5가지 형태소 분석기 제공

- 한나눔 (Hannanum)
- Mecab
- 꼬꼬마 (kkma)
- okt (= Twitter)
- 코모란 (Komoran)

2.1 딥러닝 이전 방법

① HMM : Hidden Markov Model

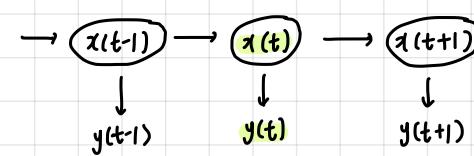
; 어떠한 결과를 예측하는 원인 = 은닉 상태인 이전의 여러 연속된 사건들

- $x(t)$: 시간 t 에서의 은닉 상태
- $y(t)$: 시간 t 에서의 관측값
- 각 상태 ⇒ 조건부 의존성 (conditional dependency) ⇒ 이어짐

$x(t)$ = 오직 이전 상태 $x(t-1)$ 의 독립적인 영향 받음

(이전 상태들 = 독립적 X, 의존적 Y)

⇒ 결과값에는 이전 상태들이 반영됨



$x(t-1), x(t)$ ⇒ 독립적 연관성 O

· 우리가 관측하는 $y(t-1), y(t)$ ⇒ 연관성 X,

형태소 분석에 미려움이 따른다

② CRF : Conditional Random Field

; 어떠한 배열을 입력했을 때 같은 길이의 결과를 반환하는 시퀀스 라벨링에 사용
(sequence labeling)

- 특징 함수 (feature function)을 정의해 사용

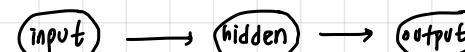
- 특징 함수 : 문장, 문장을 구성하는 단어들의 위치 + 레이블 정보 입력
→ 어떤 네이트이 얼마나 적합한지 계산

(HMM = 이전 정보 간접적으로 알 수 있음,

CRF = 이전 정보/상태 직접적으로 확인 가능)

2.2 딥러닝 이후 방법

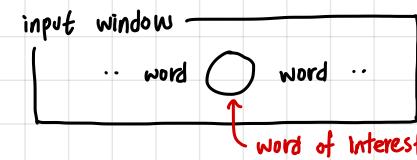
① FFNN : Feed Forward Neural Network



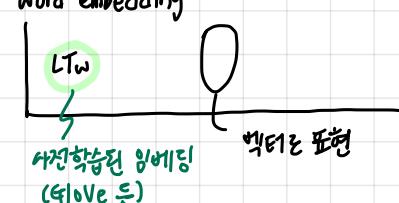
한 방향으로만 전파하는 NN

<FFNN 프로세스>

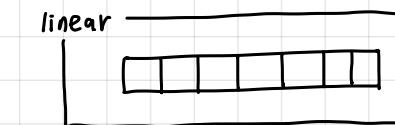
① 테그할 형태소 중심 주변 N개 형태소 워드 임베딩



② word embedding



③ 2개의 FFNN layer 통과 (닉네임, 출력층)



④ argmax 계산

⇒ 주어진 입력문장의 문어 중 가장 높은 점수의 품사 선택

$$\text{argmax} \left(\sum_{t=1}^T f(t, y_t) + A(y_{t-1}, y_t) \right)$$

② sequence-to-sequence

- 임의 길이의 한 흐름의 sequence → 다른 흐름의 sequence로 변환하는 학습 모델
- 추가 정보 사용 X, 직접 end-to-end 방식으로 학습
- attention 기반 encoder-decoder 모델 사용

input 4는 <NP> 기다린다. (음절 단위 분리된 문장)

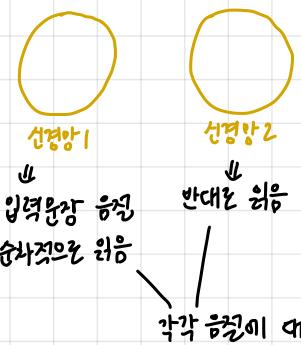


encoder-decoder

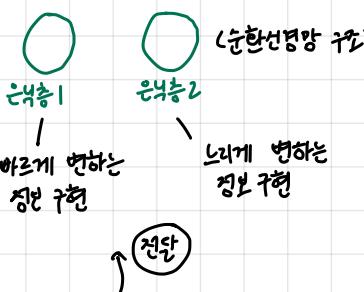


output 4<NP>는<JX>... (음절 + 풍자 태그)

1) 인코더 (encoder)

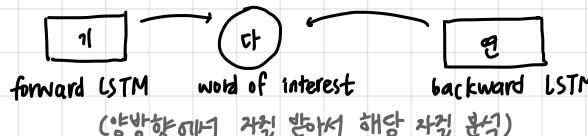


2) 디코더 (decoder)



③ character-level - bidirectional LSTM-CRF

- 의미쓰기가 제대로 되지 않으면 문장의 경우, 단어 단위가 아닌 음절 단위 형태소 분석 시험
- = 음절 단위 + (의미쓰기 같은 보조적 특징 (auxiliary attribute))
- concatenate 한 형태
- 양방향의 연속적 자질 추출



input
기 다 연

input unit auxiliary attribute

↓
forward LSTM (앞 음절)

↓
backward LSTM (뒤 음절)

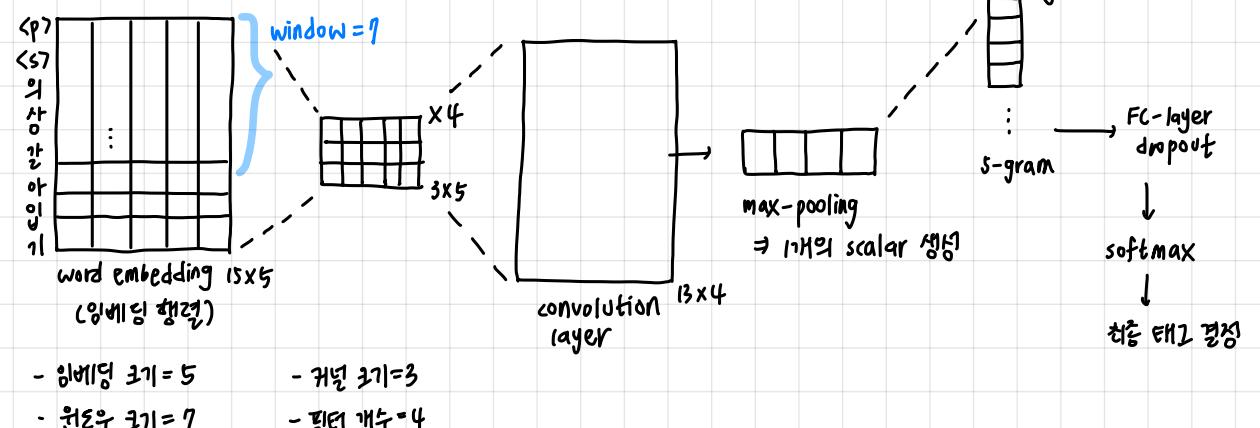
↓
stage-1 output (B-(이전) I-(이후)) 태그로 형태소 구분

↓
형태소별로 음절 같은 뒤 Bi-LSTM layer 통과

↓
output (VV VF EF) : 각 형태소(음절)에 대한 태깅 결과

④ CNN

- CNN 이용시 LSTM/RNN 이용하는 경우보다 빠른 속도
- input = 단어 단위 X, 음절 기반
- window size 만큼 좌우로 확장한 문맥 사용



Ch 22. 딥러닝 기반 한의성 단어의미 분석



LSTM이 단어의미를 분석함에 있어서 '구운 분석 정보'를 가장 중요하게 사용한다.

(구운 분석 정보 = 옆-느낌 사이의 의존관계 정보 포함)

· 단점: 구운 분석 사용시,

- 구운 분석 단계에서의 오류의 전파 (error propagation)
- 의미의 분석 전에 구운 분석이 진행되어야 한다는 단점

⇒ 해결하기 위해 형태소 분석 특징들만 사용하는 방법

<LSTM 모델의 발전>

의미의 분석을 향해 있어서 LSTM 모델이 다음과 같은 순서로 발전했다:

- ① 일반 LSTM 모델
- ② LSTM + CRFs (output layer에 추가)
(인접 주변 정보 활용 → 현재 네이밍 추출)
- ③ Stacked Bidirectional LSTM - CRFs
: hidden layer를 한 층 더 쌓아 1번째 hidden layer의 양방향으로 나온 h가 합쳐져서 2번째 hidden layer의 input으로 사용
- ④ BERT - LSTM - CRFs
 - BERT ⇒ (base 모델 (12개 트랜스포머로 구성)
large 모델 (24개 " "))
 - 트랜스포머 encoder ⇒ multi head attention + pairwise FFNN

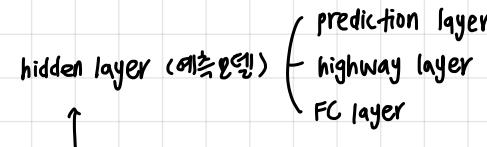
22.11 심층학습 기반 단어 중의성 해석

① labeled 데이터 이용

- 단어 중의성 해석을 위해 맥락 벡터 계산
(맥락 벡터 계산 위해 Bi-GRU 모델 사용)

⇒ Bi-GRU가 중의성 해석을 위해 순차 데이터 내 현재 정보 클래스 분류 위해 각각 정보 + 미래 정보 포함한 문맥 벡터 표현

(Bi-GRU 모델)



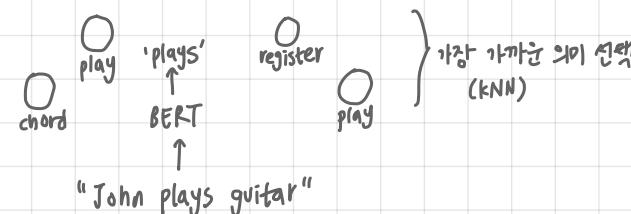
multi head attention ⇒ 중의성 단어가 문장에起到하는 의미를
↑ 분류하기 위해 문맥 어떤 단어와
contextual encoding (GRU) 이어지는지 파악
↑
word embedding layer

<BERT-KNN 모델>

- BERT의 2가지 학습 과정 (MLM (Marked LM)
NSP (Next Sentence Prediction))

을 통해 문맥 벡터 표현

- 문장의 labeled 데이터 입력 → BERT → 결과 출력
- KNN으로 가장 의미가 가까운 벡터 선택



② 지식 상호 연관

- 다양한 지식을 서로 상호 보완해서 중의적 단어의 의미 파악
→ 중의성 해석

- 2원
- context : 중의적 단어가 갖는 문장을 벡터화로 표상해 입력문장 생성
 - Gloss : 중의적 단어의 뜻풀이 (glossary) → Bi-LSTM으로 표현
 - Memory : context, Gloss의 2개의 벡터 관계 파악 → Attention 메커니즘
 - Scoring : (context
Memory에서의 attention 결과값 사용해 중의적 단어 의미 선택)

- Gloss 단어에서 'Gloss Expansion' 사용

: 상위, 하위어의 뜻풀이까지 정복해서 context + context 표현
(상위어 뜻) (중의적 단어 뜻) (하위어 뜻)

③ Zero-shot 기반

: 학습 데이터(1)에 존재하지 않는 대상의 의미도 예측

(unlabeled data 까지 가능)

1) Attentive Context Encoder - 문맥 벡터 표현

2) Definition Encoder - 의미 벡터 표현

3) Graph Embedding - "상위어의 의미정보" 임베딩

: 2) 의미 벡터와 1) 문맥 벡터의 내적으로

입력문장의 중의성 해석

[22-1] 실습

- : ELMo (사전 학습된 문장 단위 임베딩 기법) 시각화를 이용한 단어 의미의 풍미성 해석
- allenlp \Rightarrow Elmo Embedder() 클래스 제공 라이브러리
↳ 사전 학습된 모델 풍미 포함

① ELMo 클래스 선언 (get_elmo_vector)

- 단어 문장 단위로 토큰화
- \rightarrow 벡터로 저장하여 array 형태로 저장

② PCA 기반 차원축소 (dim_reduction)

- PCA (주성분분석)
 \Rightarrow n개의 합성으로 데이터 feature들을 얻는 feature_selection 방법
(회귀 분산 설명 = explained variance ratio 확대)

③ 데이터 차원 공간 상 표현 (plot)

- word of interest (중요한 단어)의 변화 형태 출력
(word+s, word+ing, word+ed 형태)

④ main 함수

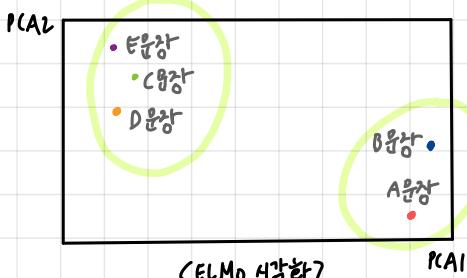
model = Elmo () 선언

- 1) dictionary 구조에 원하는 단어 다 넣기
{ "A 문장", "B 문장", ... }

- 2) word of interest 와 각 단어 비교

- 3) Elmo layer 수만큼 단어 출력
(= 차원 축소는 (dim-reduction) 축소된 차원의 크기만큼 차원 공간 상)

- 4) dim-reduction (n=2) \rightarrow 2차원 벡터 공간 상에서 표현



Ch 23. 딥러닝 기반 개체명 인식 (NER)

· 개체명 인식 (Named Entity Recognition)

: 텍스트 내에서 고유한 entity를 인식하는 작업

문자(텍스트)를 표현하는 방식에 있어서 차별 헤더 사용 → 단어 임베딩으로 대체

· 단어 임베딩 (word embedding)

: 단어를 유한 차수의 벡터로 표현하는 방법



"표현단위"에 따라 분류가능

23.1 단어 단위 구조

Bi-LSTM 브렐 사용

: 문장 → 단어 단위 임베딩으로 input

→ 단어 임베딩은 (정방향

역방향으로 1번씩 들어감

(양방향의 성분 포함시 더 높은 성능)

NER label (개체 라벨링) B-LOC I-LOC O O ..



word LSTM -B (역방향)



word LSTM -F (정방향)



word embedding (단어 단위 임베딩)



words seoul is the capital of Korea



23.2 문자 단위 구조

: 문자 단위의 연속적인 템포적 input

→ 문자 label 예측 → 문자 label을 단어 label로 변환

NER label (개체 라벨링) B-LOC I-LOC O O ..



word LSTM -B (역방향)



word LSTM -F (정방향)



word embedding (단어 단위 임베딩)



words s e o u l i s + h e ..

↳ 문자 단위로 입력

23.3 단어 + 문자 단위 구조

① 단어 + convolution 조합의 문자 하나를 표현

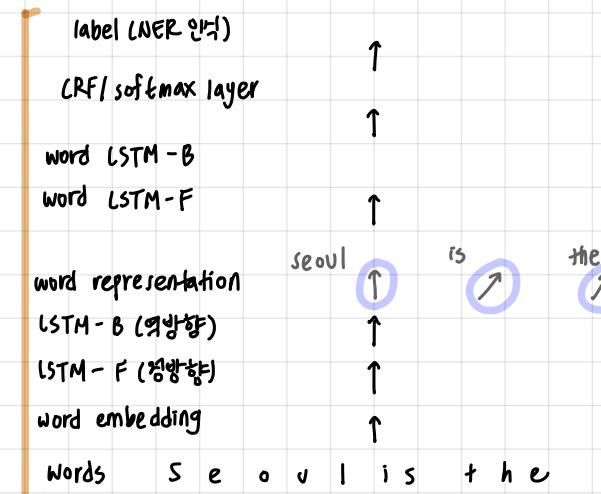
→ Bi-LSTM layer

→ 마지막에 softmax / CRF layer 사용

② 각 단어에 해당하는 문자 LSTM으로 단어 임베딩 연결

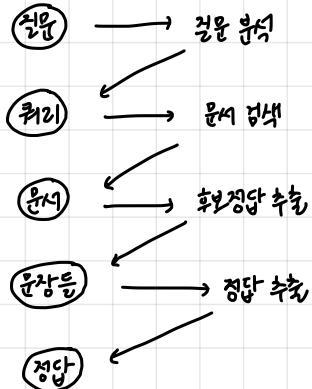
→ 해당 문자 다른 문장에 Bi-LSTM으로 전달

→ 마지막에 softmax / CRF layer 사용



Ch 24. 딥러닝 기반 질의응답 (Question & Answering)

(전통적) 질의응답 처리과정



기존 질의응답 과정:

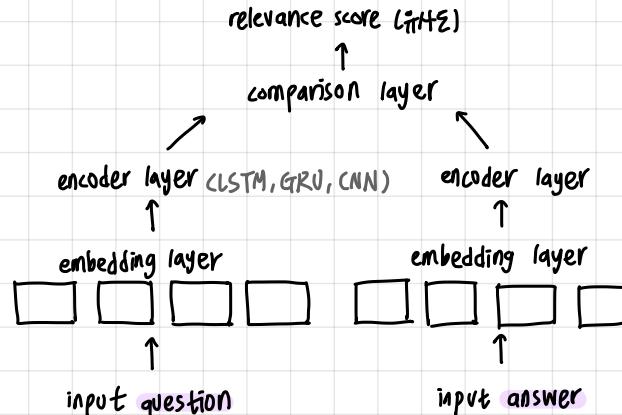
- 1) feature engineering, 형태소 분석, NER 등 같은 언어학적 지식에 의존적
- 2) 외부 리스스에 의존적

딥러닝 기반 질의응답 등장

24.1 딥러닝 기반 질의응답 모델

① 쌍 네트워크 (쌍둥이 네트워크)

- 문장 의미 유사도, paraphrase 식별
- 여러 문장의 쌍 모델링 작업에 사용



① encoder layer

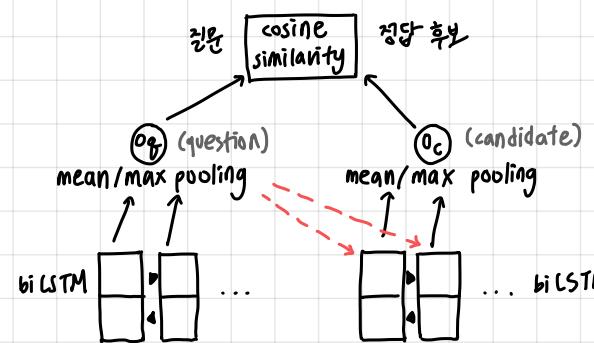
- 각각 입력 문장과 같은 구조, vector representation으로 변환
- input answer, question 인코딩 각각 = 서로 상호작용 X

② comparison layer

- 내적인 유사도, element-wise 연산, 신경망 기반 가중치 → 질문 - 정답 후보 간의 연관성 점수 계산

- 단점: - 질문, 정답 후보 간의 상호작용 X
- 정답 / 질문에 관계없이 질문 / 정답 동일한 벡터로 mapping
- 단순한 8진
- 장점: - 단순한 8진

② Attention 기반 LSTM 네트워크



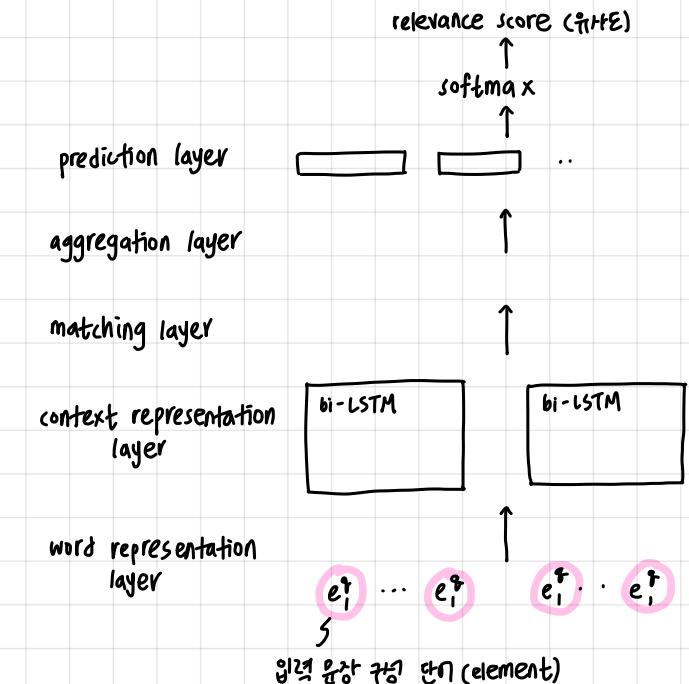
- 1) bi-LSTM + pooling layer → 질문 표현 o_q 생성
- 2) 질문 표현에서 생성된 가중치가 응해지면서 o_c 생성
 $\Rightarrow o_q$ 와 o_c 간의 상호작용 관계

Attention 메커니즘

- 정답 후보 문장에 존재하는 단어 중 질문과 관련된 단어에 가중치를 더 가하는 방식

③ Compare - Aggregate 네트워크

\Rightarrow Bilateral multi-perspective matching (BiNPM) 네트워크



① word representation layer

- 입력 문장 구성 단어 → 일정 차원의 벡터로 형성
 $\quad (=$ 단어 임베딩)

- character-composed embedding 사용
 \quad 사전 학습된 Glove, word2vec 등 사용

② context representation layer

- 질문, 정답 후보 → 원래 정보를 지닌 word-level representation 생성

③ matching layer

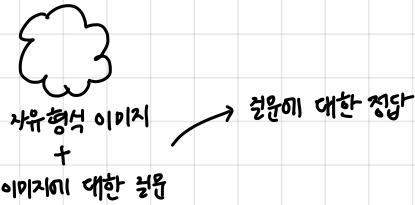
- 질문, 정답 후보의 representation 비교
 \Rightarrow 문맥 정보의 일치 정도 파악
 \rightarrow 불라ляр 청취, bi-LSTM 등과

④ aggregation layer

- : bi-LSTM으로 matching layer의 결과값 결합,
새로운 벡터 생성

⑤ softmax 계산 → relevance score

24.2 시각 질의응답 (Visual QA, VQA)



· VQA = 개방형 (open-ended) + 다수선택형 (multiple choice)

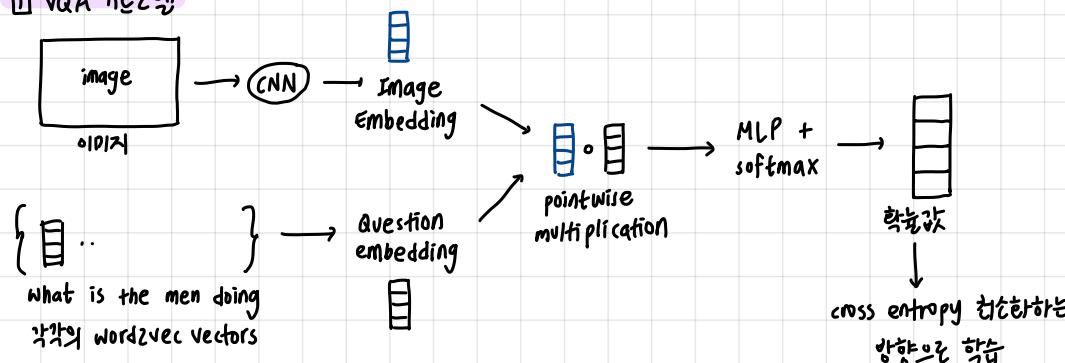
<질문 종류>

- 세밀한 인식 (fine-grained recognition)
- 물체 인식 (object detection)
- 행동 인식 (activity detection)
- 지식기반 추론 (knowledge based reasoning)
- 일반상식추론 (commonsense reasoning)



VQA의 기본 모델 = (이미지 정본 추출 모델
텍스트 정본 추출 모델)

II VQA 기본모델



1) 이미지 → ResNet / VGG 같은 CNN 통과 → 이미지 정본 담고 있는

이미지 임베딩 획득 (image embedding)

2) 텍스트 → word2vec 모델은 텍스트 임베딩 획득 + elementwise 곱셈

3) 이미지, 텍스트 서로 다른 차원이므로

FC layer → 같은 차원의 임베딩으로 변환

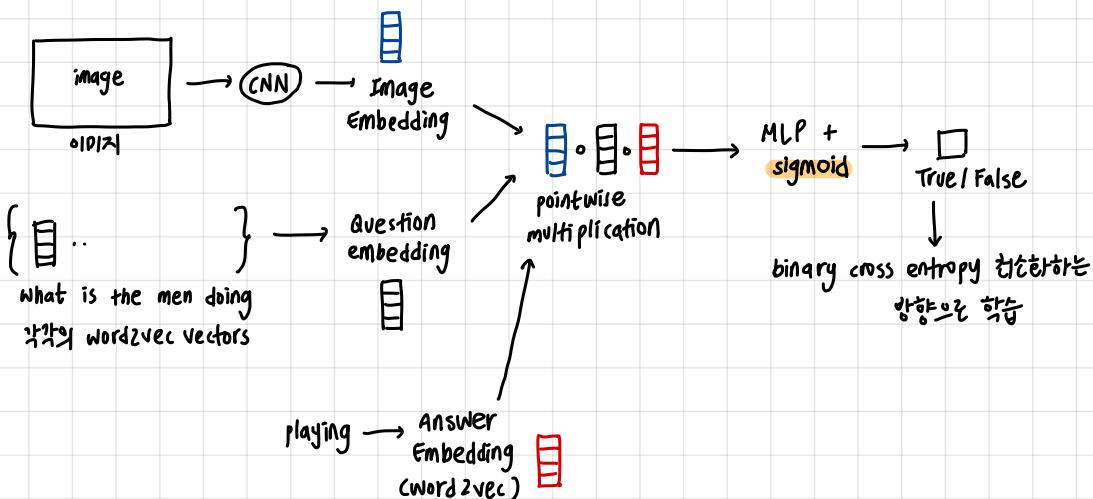
4) elementwise 곱 → 이미지 + 텍스트 결합

5) MLP, softmax → 다수선택 K개 (정답 후보)에 대한 확률, 획득

6) 실제 정답 → 생성된 정답의 cross entropy 최소화시키는 방향으로 진행

② 다수선택형 문제 VQA 모델 (이전 분류학)

- softmax 대신 마지막에 sigmoid 사용



· 이미지 레이어를 다운시 사전학습된 ResNet / VGG 사용시 학습 속도 향상

24.3 VQA 모델의 응용

1) 이미지 레이어 - AlexNet

- GoogleNet

2) 텍스트 레이어 - Glove

(word2vec)

- LSTM 네트워크

3) element-wise 곱 - 단순 concatenate

ch 25. 딥러닝 기반 기계번역

· 기계번역: 소스문장 (source sentence) → 타겟문장 (target sentence)

<기계번역 접근법>

① 주석기반 기계번역 (RBTT)

: 소스문장의 구조 / 형태소 분석

→ 타겟언어의 문법적 규칙에 맞게 변형

② 통계기반 기계번역 (SBTT)

: 대용량 커퍼스 → 학습된 통계정보 사용 (학습기반)

(번역모델: source text의 alignment 추출)

(번역언어: target text의 학습 생성)

③ 딥러닝 기반 기계번역 (NMT)

입력

출력

(= end-to-end 번역시스템)

25.1 NMT (딥러닝 기반 기계번역) 흐름

<발전흐름>

word-based
SMT (단어 단위) → phrase-based
SMT (구조 단위) → hierarchical phrase-based
SMT (계층적 구조 단위)

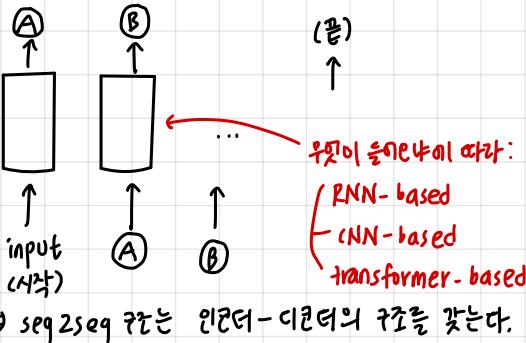
Transformer-based ← CNN-based ← RNN-based
NMT NMT NMT

: 현재는 트랜스포머 기반 딥러닝 기계번역의 성능이 가장 높음

25.2 Seq2seq: 인코더 & 디코더

· NMT의 구조 = sequence to sequence (문장 to 문장)

⇒ 병렬 커퍼스 (parallel corpus)



① 인코더 (encoder)

: 컴퓨터가 알아들을 수 있는 언어를 바꾸는 작업

→ 고려원의 벡터 (language representation)

② 디코더 (decoder)

: 이전 time 까지 번역된 문장을 바탕으로 현재 time
의 번역 단위를 생성하는 작업

⇒ end-to-end의 방식

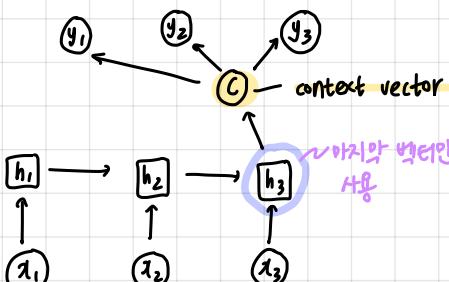
25.3 RNN 기반 NMT

: encoder ⇒ 입력의 각 아이템 처리, 정렬 추출해서

벡터 생성 (= context vector)

decoder ⇒ encoder에서 생성한 context vector를 받아

출현할 아이템 선택



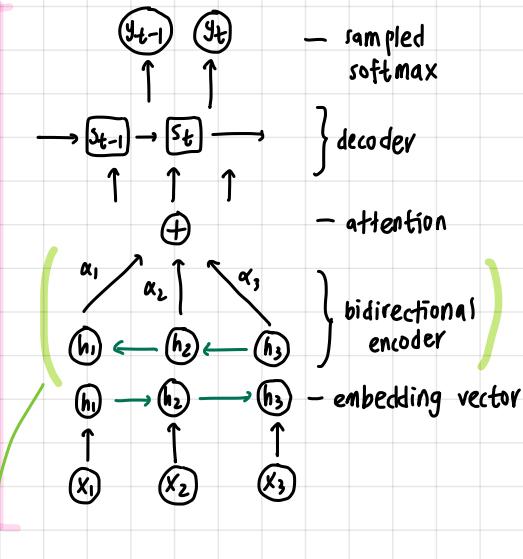
하지만, RNN 기반 NMT = 마지막 벡터 (만 사용)
 $(x_1 \sim x_n)$ 까지의 모든 입력이 context vector (하나로만 표현됨)
 ⇒ 긴 문장 / 학습 데이터에 있는 단어에 대한 학습 X
 $(=$ 고정된 벡터이기에 긴 문장에 대한 학습을 하지 못함
 ∵ Attention의 등장

25.4 Attention

: timestep마다 동적으로 더 정밀한 부분에 대해서 context vector가 변화

- 모든 hidden state → decoder에 전송

- attention으로 각 decoding step에서 입력 문장 중
어디에 집중해야 하는지 파악



① bidirectional encoder

⇒ (feed forward

backward)により 각각 hidden vector 생성

각 단어별로 1개의 vector 생성

1개의 벡터 생성

⇒ Annotations



↑ step의 annotations (1개의 벡터 합)

+

다음의 이전 hidden state vector

⇒ Feed Forward NN을 거쳐 energy (e_{ij}) 생성

$$e_{ij} = \alpha(s_{i-1}, h_j)$$

↑ 이전 step hidden vector
 ↓ 각 step의 annotations

② energy (e_{ij})

= 원증의 점수

→ softmax를 계산

" " i는 시점이 얼마나 흥미롭게 해야 할지" 라고 대해 계산

↓

energy \times 각 state의 annotations

→ context vector (c_i) 계산 가능

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum \exp(e_{ij})}$$

$$c_i = \sum \alpha_{ij} h_j$$

③ context vector (c_i)

: 디코더의 이전 hidden state vector (s_{i-1}),

이전 출력 (y_{i-1}), context vector (c_i)

⇒ 다음 출력 생성 (y_i)

$$s_i = f(s_{i-1}, y_{i-1}, c_i)$$

$$p(y_i | X) = g(y_{i-1}, s_i, c_i)$$

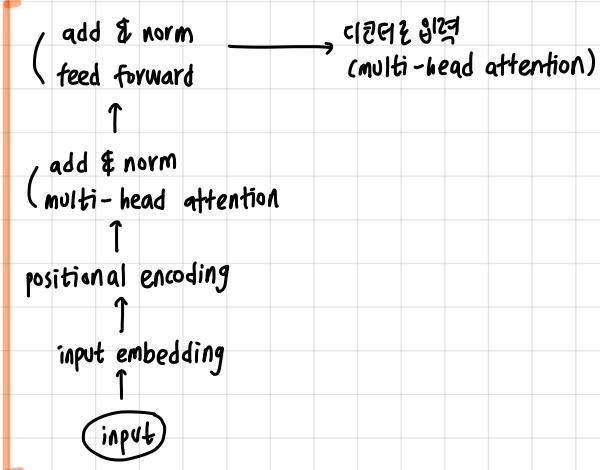
↳ 현재의 출력

여기서 더 나아가 Transformer = RNN + self-Attention, Attention만 사용

25.4 Transformer

: 인코더 - 디코더의 구조

① Transformer 인코더



- 학습 등도 가능
- 병렬화 가능

1) input으로 들어온 텍스트 데이터 → multi-head attention

· multi-head attention

: encoder가 input에 특정한 단어를 encode하기 위해서는 그것에 모든 다른 단어들 간의 관계를 살펴봄

- head의 개수만큼 self-attention 수행

(self-attention) : input 단어들, 자기 자신들 간의 attention)

2) self-attention × head 연산 후 feed forward로 출력

⇒ 각 위치의 단어마다 독립적으로 적용해 출력 형성

↓
Transformer 구조를 개별 설명

25.5 self-attention

① encoder에 입력된 벡터들 → 각 3개의 벡터 생성

⇒ Query 벡터, Key 벡터, Value 벡터

→Query 벡터에 대해 3개의 학습 가능한 행렬 각각 생성

embedding

$$\boxed{x_1}, \boxed{x_2}, \dots, \boxed{x_n}$$

Queries

$$x_1, \boxed{W^Q} \rightarrow \boxed{w^Q}$$

Keys

$$" \quad \Rightarrow \quad \boxed{w^K}$$

Values

$$\Rightarrow 3개의 벡터 생성 \quad \Rightarrow \quad \boxed{w^V}$$

· weight matrix (x_i) $\times W^Q = q_i$ (query 벡터)

· $x_i \times W^K = k_i$ (key 벡터)

· $x_i \times W^V = v_i$ (value 벡터)

② 정수 계산

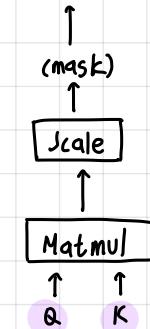
: A라는 단어, 입력 문장 속 모든 다른 단어들에 대한 정수계산

⇒ 정수 : 현재 위치의 단어를 encode할 시 다른

단어들이 얼마나 흥미롭게 해야 할지 결정

= query vector o key vector (내적)

Softmax



③ 정수/8

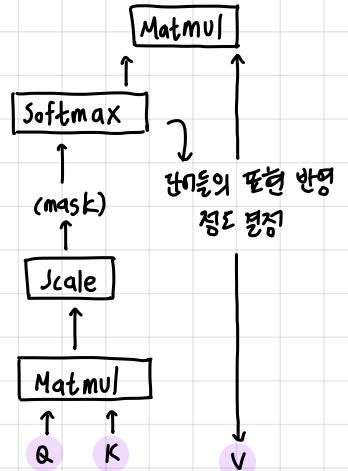
(대 8로 나누는가?)

⇒ key vector의 사이즈인 64의 제곱근)

- 정수/8으로 나누어줌으로써 더 안정적인 gradient 있음

→ softmax 통과 (양수화, 합=1)

- softmax: 현재 위치의 단어의 encoding이 있어 얼마나 각 단어들의 표현이 들판 같지 결정



④ 입력의 각 단어들의 value 벡터에 정수 곱하기

⇒ 집중하고 싶은 관련있는 단어들 그대로 놓겨짐

(관련 없는 단어들 ⇒ 0.001 같은 작은 정수를 곱해 없앰)

⑤ 정수가 곱해진 weighted value 벡터 합하기



output: 현재 위치에 대한 self-attention layer의 출력

∴ self-attention layer를 출력하기 위한 프로세스:

'scale dot product attention'

25.6 Multi-head attention

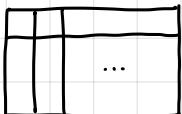
(head의 개수만큼 25.5의 self-attention 연산 수행)

⇒ self attention layer + "multi-head"

= 여러 개의
query 벡터
value 벡터
key 벡터 학습

· 여러 개의 query, value, key weight 행렬을 가질 시,
각 맵터들을 가지 다른 representation 공간으로 표현

\times (입력 벡터의 모음)



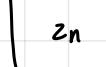
↓ 각 head에서 다르게 self-attention 연산 수행

attention
head #0



...

attention
head #n



: self-attention을 n번 수행해서 각각 다르게 표현된 n개의 attention head 출력

하지만, 이를 바로 feed forward layer로 보낼 수는 없다.

feed-forward layer

: 한 위치에 대해 오직 1개의 input만 받을 수 있음

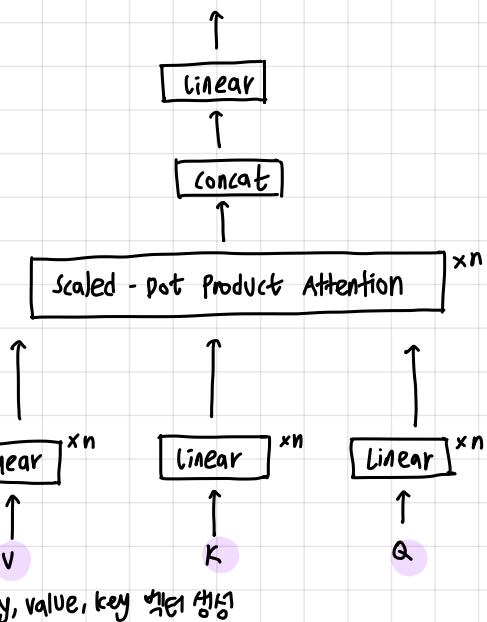


n개의 attention head 행렬을 이어붙여서 1개의 행렬로 만들기

→ 1개의 다른 weight Wo 곱하기

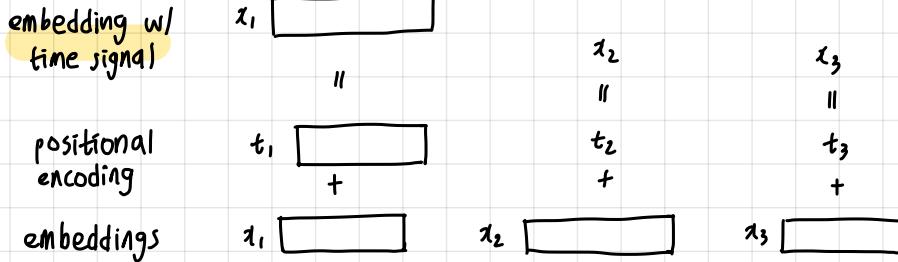
∴ 1개의 multi-head attention 값으로 표현

Multi-head attention



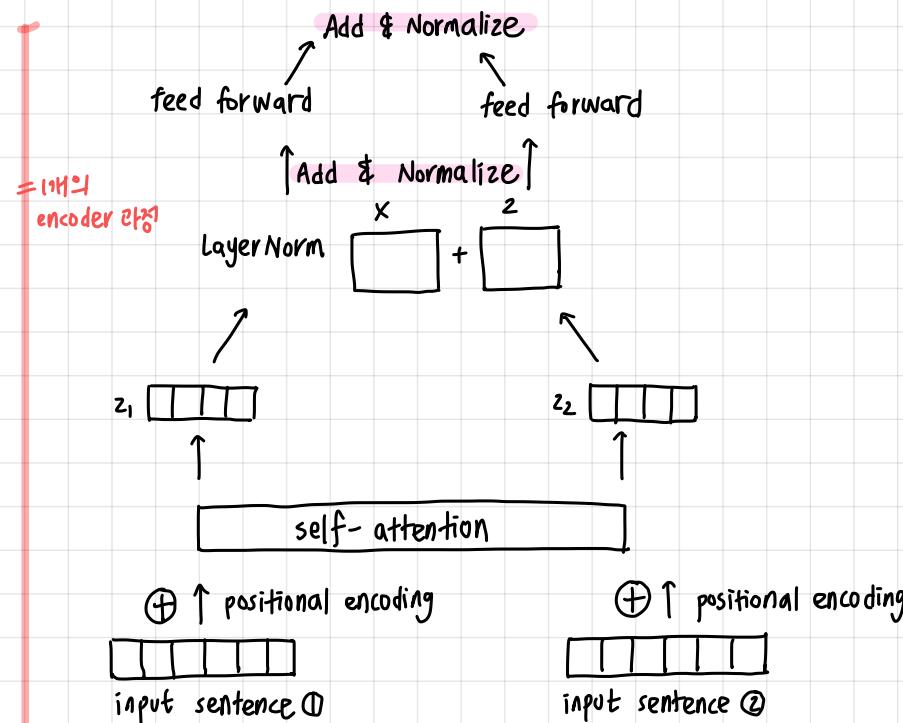
25.7 Positional Encoding

- transformer 템플릿 내 입력 문장 내 단어들의 순서 고려 필요
- 각각의 입력 embedding에 positional encoding 벡터 추가
- ⇒ 2열이 각 단어의 위치 + 시퀀스내 다른 단어간 위치 차이에 대한 정보 알려줌

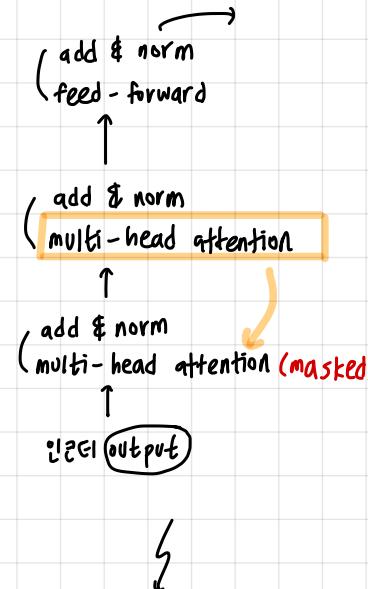


25.8 Residual & Layer normalization

- encoder 마다 각 sub-layer → residual connection으로 연결
- layer normalization (= add & norm) 과정 포함



25.8 Decoder



<decoder와 encoder의 차이점>

① self-attention layer

- ⇒ 현재 위치 + 이전 위치들에 대해서만 attention 진행
- softmax 이전에 현재 이후 위치들에 대해 masking 처리

② multi-head attention layer

- ⇒ (query 벡터 - query layer에서 가져옴)
- (key, value 벡터 - encoder의 출력에서 가져옴)

Transformer의 encoder → (positional encoding

multi-head attention

normalization) → Transformer의 decoder
(masked multi-head attention

multi-head attention)
(Linear layer ←
Softmax layer)

을 통해서 벡터를 1개의 단어로 표현

- ① linear layer ⇒ FC-신경망으로 decoder가 출력한 벡터의 각 셀을 그에 대응하는 단어의 정수로 표현
- ② softmax layer ⇒ 정수들을 양수의 확률값으로 변환

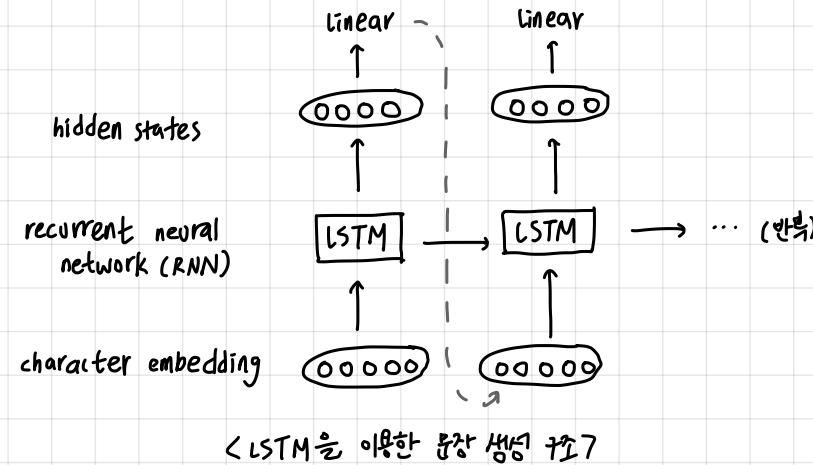
↓
output probabilities 중 확률이 가장 큰 단어는 번역

Ch 26. 뇌러닝 기반 문장생성

자연스러운 문장을 생성하기 위해 만족해야 하는 조건:

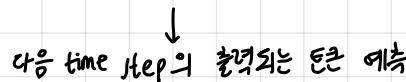
- ① 생성되는 텍스트가 들바른 통사 구조를 가짐
- ② 의미적으로 자연스러움
- ③ 문장 내 단어들 / 문장 내 문장들 → 통일된 문맥에서 구성됨

26.1 LSTM 기반 문장생성 모델



- 각 time step에서 이전 출력층의 값 = 다음 입력층의 값
 → LSTM의 은닉층 (hidden states)은 전달
 → 이전 time step의 정보를 고려해 가중치로 반영

∴ 현재 time step의 입력 토큰



- ① 레이터셋 전처리 - training / test 레이터셋 준비 + 전처리 진행
- ② 모델 설계 - batch, corpus, 임베딩 차원, RNN 디라미터 지정 + 모델 설계
- ③ 모델 훈련 - optimizer / loss function 정의, 문장 생성을 위한 checkpoint 저장
- ④ 문장 생성 - 각 time step 별로 다음 토큰 예측
 → 예측되는 토큰을 묶어서 문장 생성

26.2 Self-attention 기반 문장생성 모델

(open GPT-2 사용)

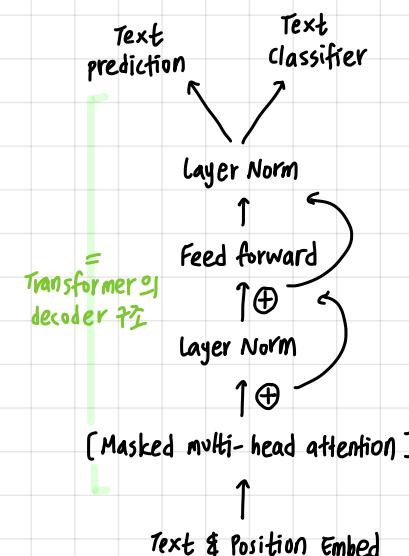
GPT-2

= 바이저드 학습 방식으로 훈련 + multi-task learning에 적용 가능한 모델

- 기존 \Rightarrow BERT / GloVe와 같이 사전학습된 언어 모델 사용 + fine-tuning
 → SOTA (state-of-the-art) 달성
- 지도 학습이기 때문에 대량의 라벨링된 데이터셋 필요

(GPT-2의 특징)

- ① 커뮤니티 사이트에서 web scrapping으로 길어온 기존 자료 사용해서 훈련 데이터로 투입
 → 바이저드 학습 방식으로 언어 모델 학습
- ② 별도로 fine-tuning 할 필요 없음
- ③ Transformer의 decoder 구조 활용
- ④ 방대한 양의 데이터셋을 이용해서 학습 진행 - 성능 향상의 주요 원인



Ch 27. 딥러닝 기반 문서 요약 (Text summarization)

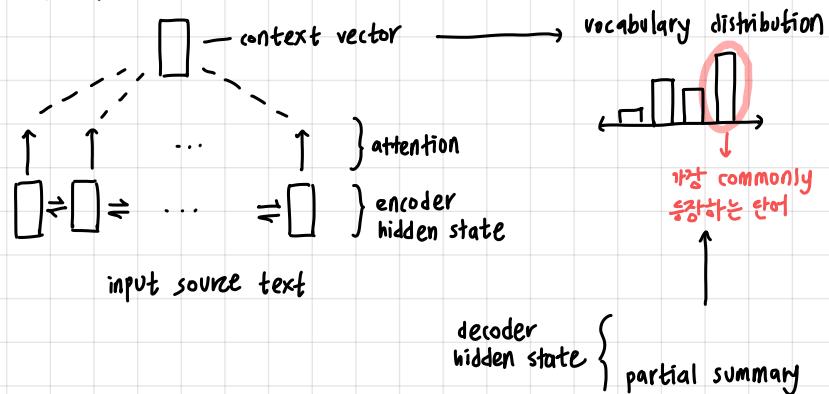
문서 요약

- (1) 추출 요약 (extractive summarization)
- (2) 추상 요약 (abstractive summarization)

27.1 딥러닝 기반 추상요약

① 순환신경망 (RNN)

⇒ seq2seq + Attention 기법을 추가한 모델



- 1) encoder RNN에서 input source text를 단어별로 읽어들임
- 2) encoder hidden state에서 sequence 형성
→ 전체 문장에 대한 알기 처리 완료
- 3) decoder RNN에서 요약문의 형태를 만들기 위해 단어들의 sequence 생성
: 요약문의 이전 단어 input → decoder hidden state 상태 업데이트
→ 문서 내 단어들의 확률 분포 = 이전 단어 분포 계산

· 어텐션 분포 (attention distribution)

: 다음에 나온 단어에 대해 네트워크가 어디로 집중해서 학습할지 알려주는 시표

- 4) 이전 단어 → encoder hidden state의 가장 높은 원액 벡터 생성
- 원액 벡터:
같은 단계의 decoder에서 문서 source text에서 읽어온 것이 무엇인지 나타냄
- 5) 원액 벡터 + decoder hidden state ⇒ 이전에 모든 단어들의 확률 분포인
‘사전 분포’(vocabulary distribution) 생성

(RNN을 이용한 문장 요약의 문제점)

- ① 요약문이 때때로 사실적 세부사항을 무시하거나 계산하는 경향 0

→ (사전에 있는 단어 (OOV)
혹은 단어 (rare word)인 경우)

- ② 요약문은 때때로 같은 단어끼리 재반복되는 경우 0

↳

원인

- ① input source text에서 나타나는 단어를 그대로 가져오기 때문에
⇒ seq2seq + attention 모델 내 여러 개의 layer를 거치면서 원본 단어를
그대로 복제하는 작화의 어려움
 - 희귀한 단어인 경우 극장한 문제
 - 계산상 하면서 임의적으로 다른 단어로 대체하는 경우 빈번

Ex) Grace → Anna
Seoul → Tokyo

- ② 디코더에서 long-term 정보를 저장하기보다 디코더의 입력 (= 이전에 나온 단어)
에 너무 의존적인 경우에 발생하는 문제
⇒ 반복되는 단일 단어 → 계획되는 반복 순환 유발

27.2 Combination (pointer-generator) 방법

순환신경망 (seq2seq + attention) 모델의 문제점 ①

= 부정확한 복사 (inaccurate copy) 때문

⇒ 이를 해결하기 위해 고정 규칙에서 단어 생성하는 기능 유지

+

원본 텍스트에서 포인팅으로 단어 그대로 가져오기
의 하이브리드 네트워크 사용

· p_{gen} (0~1 사이 scalar)의 생성 확률 (generation probability) 생성

: 원본 input text에서 사용한 단어 대 사전으로부터 생성한 단어의 확률값

$$p_{\text{final}}(w) = p_{\text{gen}} p_{\text{vocab}}(w) + (1 - p_{\text{gen}}) \sum a_i \quad \text{어텐션 분포}$$

최종분포 ↗
└ 생성 확률 (가중치 역할)

$$P_{final}(w) = \underbrace{p_{gen} P_{vocab}(w)}_{\text{생성된 단어 } w \text{의 확률}} + \underbrace{(1-p_{gen}) \sum q_i}_{\text{천본 문장 내 단어가 나타나는 }} \times$$

생성된 단어 w 의 확률
= 사전으로부터 단어가 생성될 확률 \rightarrow pointing → 확률 더하기

< seq2seq + attention VS. pointer-generator >

pointer-generator 의 장점 :

① 천본 텍스트로부터 단어를 그대로 가져오기 쉬움
→ 해당 단어에 대한 p_{gen} 가중치 크게 부과

② OOV (사전에 정의되지 않은 단어)에 대해서도 그대로 가져옴
- 적은 양의 사전 활용 시 이점
- 본이지 않는 단어 (unseen words) 처리 가능

③ 학습 반복이 적기에 학습 속도 향상

2.7.3 coverage

seq2seq + attention 모델의 문제점 ② 인 반복되는 단어 해결
⇒ coverage 기술 사용

coverage

: 이전의 문장을 사용해 단어들에 대해 cover 해있던 기록 추적 +
같은 부분에 접근 → penalty 부과

1) 디코더에서 각 단계 t 에서 coverage 벡터 $c^t =$ 모든 이전의 문장 a^t 의 합

$$c^t = \sum_{t=0}^{t-1} a^t$$

(모든 이전의 문장 내 단어들의 기록 추적)

∴ 천본에서 특정 단어의 coverage = attention에서 받은 값의 합

2) coverage 벡터 c^t , 이전의 문장 a^t 사이 중복되는 단어에 대해
penalty 부과

$$\text{covloss}_t = \sum_{i=0} \min(a^{ti}, c^{ti})$$

⇒ coverage 벡터와 이전의 문장 벡터 중에서 더 낮은 값의 합

ch 28. 딥러닝 기반 대화 시스템

① 목적 지향 대화 시스템

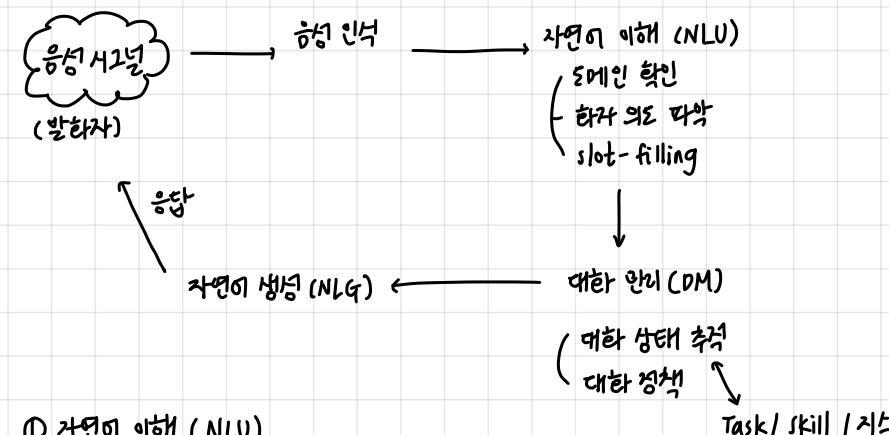
- 특정한 목적 / 작업을 수행하는 것을 목표로 함
(파이프라인 방식
(중간간 학습))

② 비목적 지향 대화 시스템

- 인간처럼 자연스럽고 일관되게 대화가 가능하도록 함
(검색 기반 방식
(생성 기반 방식))

28.1 목적 지향 대화시스템 (task-oriented)

① 파이프라인 (pipeline) 방식

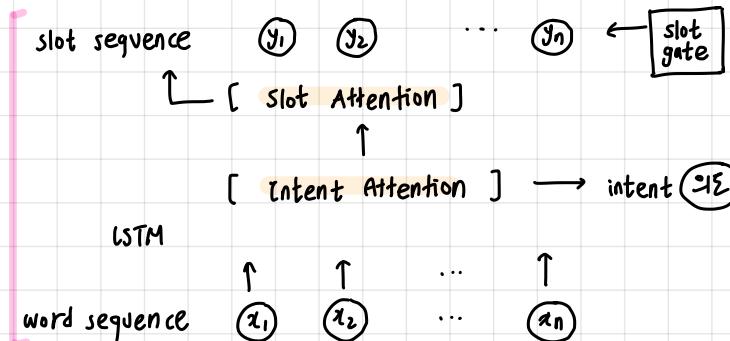


예) 다음주 미국행 항공권 예약 해줘

슬롯	B-date	B-desti	O	O	O
의도			항공권 예약		
도메인			예약		

⇒ 성능 향상을 위해 (의도 추출
슬롯 채우기 분야에서 쓰이는 오톨 엔진 활용)

: 의도 추출 + 슬롯 채우기를 동시에 하는 '동시학습' 방식 (joint learning)



⇒ 동시에 학습함으로써 2개의 작업의 성능을 모두 향상시킬 수 있음

$$\alpha_{i,j}^s = \frac{\exp(e_{i,j}^s)}{\sum \exp(e_{i,j}^s)}, \quad e_{i,j}^s = z(W \cdot h_i)$$

↓
 $c^s = \sum \alpha_{i,j}^s h_i \Rightarrow$ 槽門層 차집

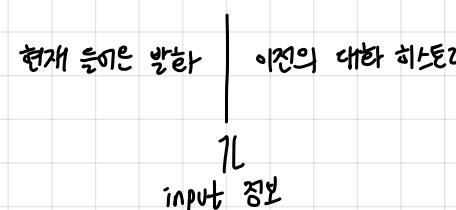
↳ softmax $\in y^s, y^I$ (slot, intention) 구하기 → cross entropy 목적함수 계산

1) 슬롯 추출: $y^s = \text{softmax}(W(h_i + c_i^s))$

2) 의도 파악: $y^I = \text{softmax}(W(h_i + c_i^I))$

② 대화 추적 (Dialogue Management)

: 이전의 대화 내용 + 현재의 발화 → 대화 추적 + 정책 학습 진행



⇒ 목표와 요청을 정확하게 추적해서 블바는 응답 제공

③ 자연어 생성 (NLG)

: 자연스러운 응답 생성

- Attention 모델에서 좋은 성능

② 통합간 학습 (end-to-end learning)

· 기존 파이프라인 방식:

- 도메인 관련 지식이 필요한 hand-craft 개발 필요
 - 새로운 도메인으로의 확장 어려움



통합간 학습 = - hand-craft 개발 필요 X

- 많은 양의 데이터셋 필요

28.2 비목적 시향 대화 시스템 (챗봇)

① 검색 기반 (retrieval-based)

: 다음 발화 예측시 기존 발화 저장소에서 가장 관련있는 발화
→ 응답으로 제공

- 훈련 데이터 = 대화 X, 접두 발화 X, 응답 발화 X
- <대화, 응답 발화, 접두라벨>의 triplet 임력 데이터로 사용

(검색 기반 대화 시스템 모델)

① 대화-응답 일치 모델

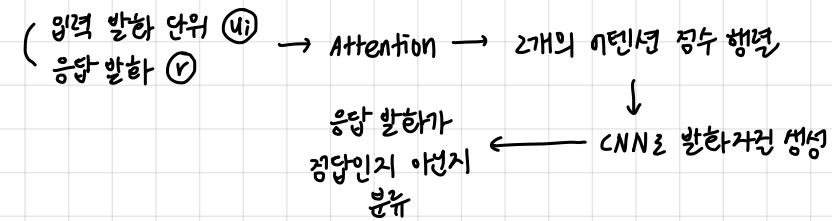
(대화 ⑥ → LSTM으로 추출 → (흐름값 ⑦
응답 발화 ⑧)

⇒ h_c 와 h_r 의 흐름 일치 확률 구하기

② 발화 단위 attention 모델

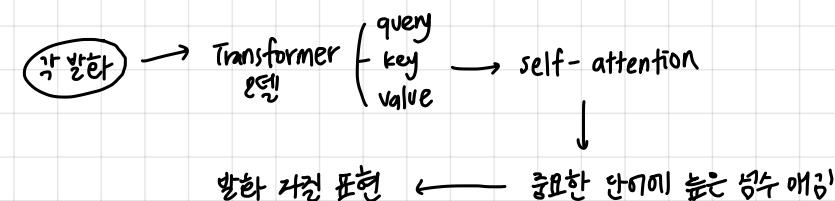
① = 대화를 1개의 문장으로 간주

② = 발화 별로 문맥 정보 추출 → 개별적으로 자질 구하기



- Attention을 통해서 학습되는 구조
- 응답 발화 - 대화 내 발화 관련성 학습 가능

③ Transformer 기반 모델



② 생성 기반 (generative)

- (장점: 검색 기반과 달리 발화 저장소 필요 X
단점: 정확도 ↓ (의미없는 문장 생성 / 같은 단어 반복 충격하는 문제)

· 기존: seq2seq 모델

- (인코더 임력 → 인코더의 마지막 은닉층을 디코더의 임력)
- 마지막 은닉층이 모든 대화 정보 포함
→ 길이 길어질수록 정보 손실의 문제

· 발전: 계층적 구조의 seq2seq 모델

- ⇒ 인코더의 은닉층을 다음 발화의 은닉층에 가중치 전달
⇒ 대화의 발화 계층을 두어 학습하는 방식

- 대화 히스토리 자질 사용 → 정보 손실 적음
- 긴 문장 처리 가능

③ Hybrid (검색-생성 혼합)

- (검색 기반: 정확 + 자연스럽지 X
생성 기반: 자연스러움 + 무의미함) ⇒ 혼합

Ch 29. 딥러닝을 이용한 SNS 분석

- SNS 분석이 학제적 이유?
 - 1) 다양한 데이터 형태 (정형 + 비정형)
 - 2) privacy issue → 적은 양의 공개 데이터

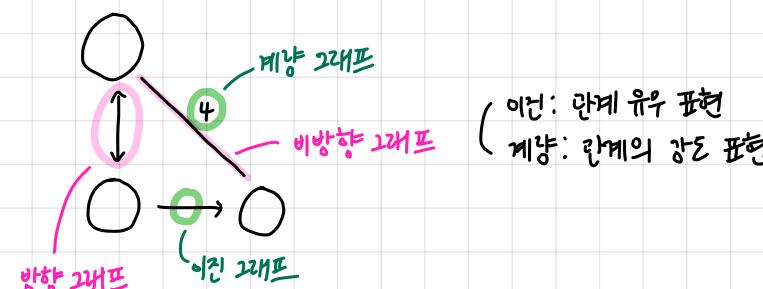
29.1 SNS 분석 기법

① 워드클라우드

: 유세) 문단에서 자주 사용되는 단어 번도 계산 → 구름 형태 시각화

② 네트워크 분석

- 데이터를 구조화된 선형망(연결망), 1개 이상의 관계 유형에 의해 연결된 구성원의 집합으로 표현
- 네트워크 = 액터 (actor)의 집합
- 소시오크램 → 노드 + 링크로 표현



<중심성> - centrality

: 액터의 구조적인 위치에 따른 상대적 중요성 측정 지표

① 연결정도 중심성 (degree)

= 한노드에 몇 개의 링크?

② 근접 중심성 (closeness)

= 간접적 연결까지 고려한 전체 네트워크 내 거리

③ 애개 중심성 (betweenness)

= 직접 연결되지 않은 액터간 관계 통제 / 중개하는 정도

④ 아이겐벡터 중심성 (eigenvalue)

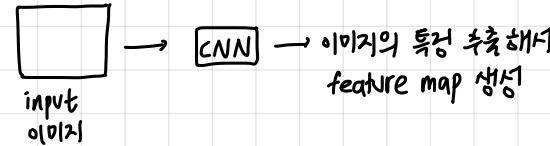
= 연결된 개수 + 중요도 함께 계산

∴ 특정 액터가 그래프 내 어떤 역할을 하고 있는지 파악

③ 감성분석 (sentiment analysis)

: 텍스트에 나타난 사람들의 태도, 의견, 성향 분석하는 기법
(= opinion mining)

Ch 30. 이미지 캡션 생성



30.1 필요 데이터셋

① ILSVRC ImageNet

= 이미지 + 레이블로 구성된 데이터셋

- 이미지 인코딩 추출시 쓰이는 CNN 훈련에 사용 적합

② MS-COCO

= 이미지 + 할당된 표지의 텍스트 문장으로 구성된 데이터셋

30.2 이미지 캡션 생성 과정

① CNN - 이미지 feature 추출

= convolutional layer - pooling layer의 반복으로 이미지에서 feature vector 생성 → 이미지 속성 학습 표현

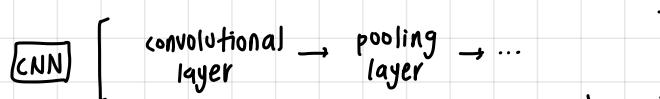
- pre-trained CNN (예: Inception v3)와 같은 모델 사용

- 시도한 CNN 모델 구성

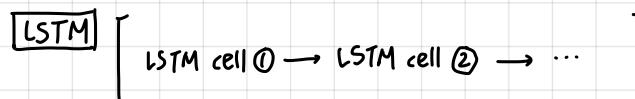
② LSTM - feature 번역

= 이미지 feature를 텍스트로 간주해서 이를 번역하는 과정

→ 이미지 캡션 생성



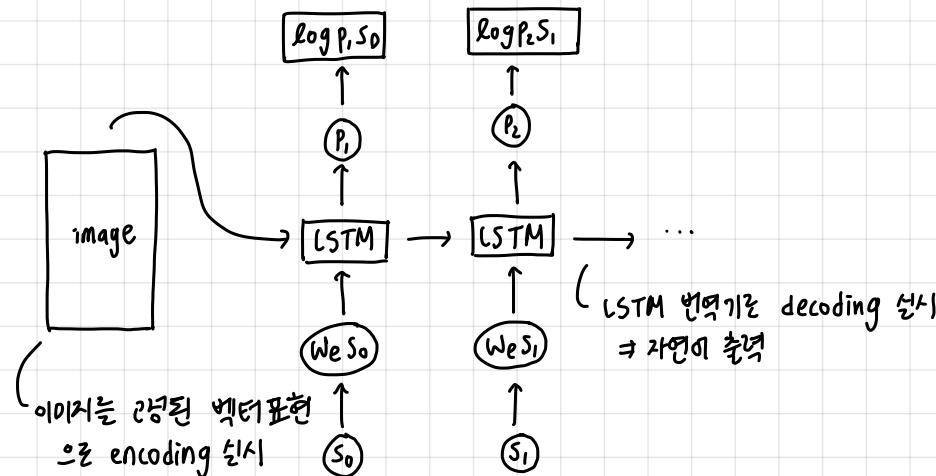
이미지 feature vector



Embedding Layer

30.3 이미지 캡션 생성 모델 (Show & Tell)

- CNN + LSTM 기반의 모델



- 단어 = $\{s_0, s_1, \dots\}$
- 단어의 워드 임베딩 벡터 = $\{w_{eS_0}, w_{eS_1}, \dots\}$
- LSTM 출력, 다음 단어 예측에 사용되는 확률분포 = $\{p_1, p_2, \dots\}$
- 올바른 단어가 나온 MLE = $\{\log p_1, s_0, \log p_2, s_1, \dots\}$

⇒ MLE의 부정합을 최소화 하는 것이 목적

- 훈련 횟수 (number of epochs) 늘 증가시킬수록 성능 향상