2024.12.11

정보 추출

1. 정보 추출
   비정형 text로부터 구조화된 정보 자동 추출
   구조화된 정보 => 관계형 튜플 형태로 표현 (entitiy1, relation, entity2)

   비정형 텍스트 -> 구조화된 정보로 자동 추출
   구조화된 정보는 관계형 튜플 형태로 표현 (충북대, in, 청주)

- 관계 추출
  개체간 관계를 식별하고 추출하는 작업
  개체 유형과 둘 사이 관계 유형 추출
- 이벤트 추출 (여러개의 관계로 구성)
  하나의 이벤트 중심으로 주체, 대상, 장소 등을 추출
  여러개의 관계로 구성
  - 뉴진스의 8월 3일 롤라팔루자 공연(EVENT)에는 약 7만명의 관객이~ 특히 버니즈라 불리는~.

- 정보 추출 목적
  - 비정형 데이터로 정형 데이터 추출
  - 관계를 논리적 표현으로 변형하여 논리 연산함
  - -> 논리 연산 이용하여 새로운 추론 가능
  - 의미적 관계를 이용해서 질의 응답 (A→B , B→C ⇒ A→C)

- 정보 추출 방법 분류
  - 지정관계 정보 추출 (주어진 정보에 대해서만) => 말뭉치 기반
    1. 규칙 기반 2. 지도학습 기반 3. 준지도학습 기반 4. 간접지도학습 기반
  - 개방 정보 추출 (임의 관계 모두 추출)
    비지도학습 기반

- 딥러닝 기반 정보 추출

  기본 모델 또는 선행 학습 모델을 말뭉치 활용해 학습
  - 딥러닝 기반 개방 정보 추출
    - sequence labeling
      1. BIO 태깅 2. Span 선택
         - BIO 태깅: BiLSTM
         - Span 선택: Span 단위로 분류 1) predicate(서술어) 학습 2) 1)조건으로 argument 학습
    - sequence generation
      1. sequence generation 2. sequence decoding
  - sequence 생성 방법 문제점
    - 특정 관계 출력 대한 조건문이 없음

        - beam search 시 매 입력마다 고정 개수의 관계 추출
        - 만약 gold 관계가 많다면 유사 관계가 여러번 출력 -> recall 저하 (output 반복이 많음)
    - sequence 디코딩 방법 (위의 유사관계 한번만 출력되도록 함)
      출력된 관계 튜플을 다시 인코더에 추가 입력해 중복이 적고 새로운 관계 추출

기계 번역

- 규칙 기반
    - 종류 (3가지)
        1. **직접 번역 (Direct) 2. 전달 번역(Transfer) 3. 피봇 번역 (Pivot, InterIngua)**
            - N개의 언어 번역 복잡도: D(직접)/T(전달) = N * N, I(피봇) = 2N
            - 직접 번역

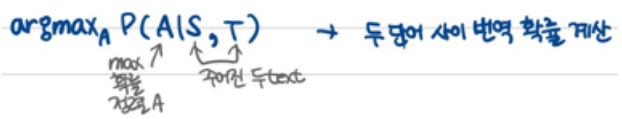
              언어 구조 변형X 거의 1:1 매핑 번역 (한국어-일본어)
            - 전달 번역
                - 구문 전달: 구조가 다른 언어 사이 번역. 구조 변형 후 각 요소 번역
                - 의미 전달: 상대 언어의 의미에 맞게 번역 (영어의 원어, 번역어)
            - 피봇 번역

              원어 분석해 의미표현(ex: 1. semantic frame, 2. 딥러닝 잠재벡터)으로 변경

              → 이를 단계적으로 번역어로 번역

              N개 언어 번역 시, n*(n-1) 경우의 수 생성 ⇒ 피봇 번역 시 2n 개만 번역하면 됨
    - 장점: 성공 시 문법적 정확한 번역 가능
    - 단점: 언어학적 지식 많이 필요, 문법적 예외 존재 → 오류 발생확률 높아짐, 1:1 대응 번역 → 단어나 구 사이 연결 자연스럽지 못함, 잘못된 분석의 오류 전파 → 번역 성능 저하
- 통계 기반

  병렬 말뭉치 (원문:번역문=1:1쌍) → 두 언어 사이 상관관계를 통계적으로 분석하여 번역
    - 텍스트 정렬: 병렬 text에서 원어와 번역어를 연결시키는 것

    - 통계기반 기계번역 모델

    - 성능 : 언어 종류 비슷, 충분한 크기의 병렬 말뭉치 존재 시 우수한 성능

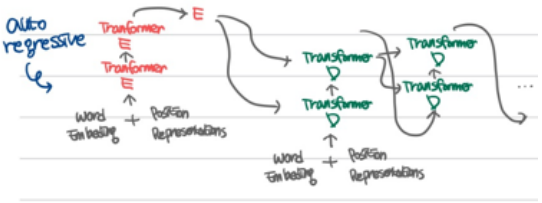
- 신경망 기반
  - 추세 : RNN 인코더-디코더 → 단순 RNN or LSTM 기반 (통계기반과 성능 비슷) → Attention 메커니즘 (유의미성능) → transformer
  - RNN 기반

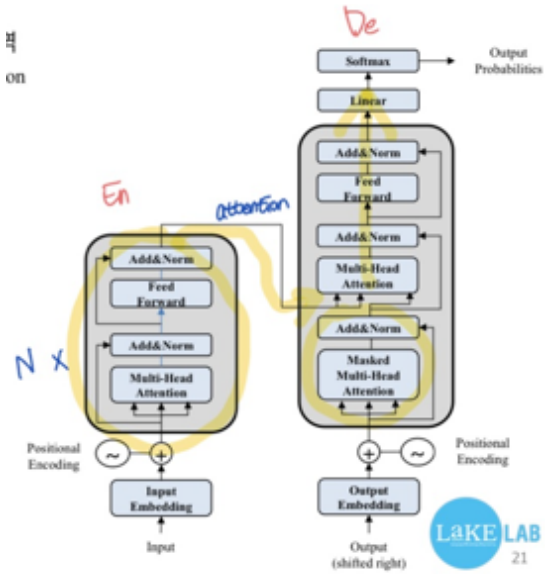
    n-m의 seq2seq 문제로 처리.

    한계: 인코더-디코더 원거리 정보 소실 → 정확한 번역 어려움

    노출 편향 → 성능 저하
  - attention (기본 RNN + attention 기능 추가)

    원본 언어의 단어 문맥 참조 → 정확도 상승
  - transformer (only attention 사용 → 인코더-디코더 구조 만듦)





단어 벡터 (=단어 임베딩)

단어를 벡터(숫자들의 배열)로 변환하는 방법

- 목표: 원시 말뭉치를 활용 → 비지도 학습, 범용적으로 사용하도록 단어 벡터로 만든 것
- VS one-hot encoding (학습 데이터 존재하는 단어로 단어집 구축)
  - 매우 큰 sparse vector 필요, 단어의 의미 파악 불가, 단어 매칭 or 비교 문제

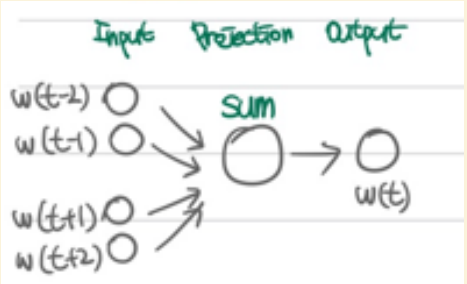
    ⇒ 이런 걸 피하기 위해 단어 벡터 사용
- 분포 가설 (끼리끼리) : 유사 단어들 문장에서 사용 분포(문맥)이 비슷
- 단어 벡터 관계 연산 ⇒ 관계 유추 가능
  - 의미적(왕-남+여=여왕), 문법적(worked-work+run=ran), 지식적 관계(대한민국-서울+마드리드=스페인)
- 평가방법
  - 내재평가: 평가데이터를 구축하영 활용.

의미 유추 관계 평가, 구문적 유추 관계 평가

- 외재평가: 다른 응용프로그램 입력을 활용 → 성능변화 측정,

  감정 분석, 품사 태깅, 개체명 인식 등 평가

- 단어 벡터 구축 방법
  - Word2vec : 언어 문맥이 가진 양방향 의존성 특징을 사용
    - CBOW(Continuous Bag-of-Words): 특정 단어 중심으로 n개와 이후 n개 단어 주어질 때 중심단어 예측



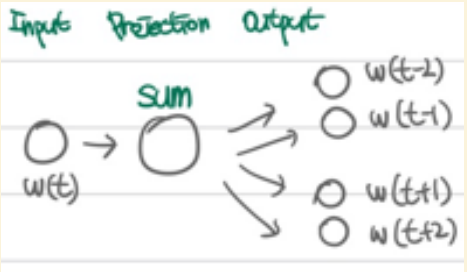
      - 원리
        - 인코딩 : ont-hot 벡터 x W = 인코딩 → 각 문맥 평균 구함
        - 디코딩 : 인코딩된 베게터를 x W` → 확장 → 대상 벡터 예측

          계산 결과 값 없이 softmax → 대상 벡터와 비교

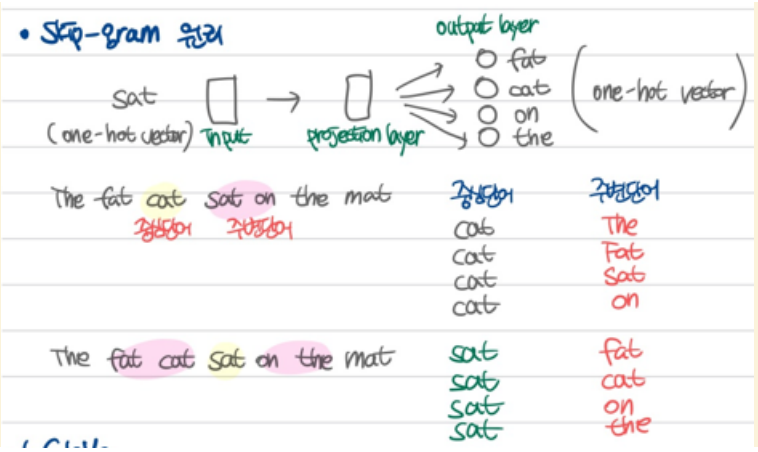
          이 손실값 계산하여 backpropagation
        - 손실함수 :

$$H(\hat{y}, y) = -\sum_{j=1}^{|V|} y_j \log(\hat{y}_j)$$

    - Skip-gram: 중심단어 주어졌을 때 이전 단어 n개와 이후 단어 n개 예측



      - 원리



- GloVe

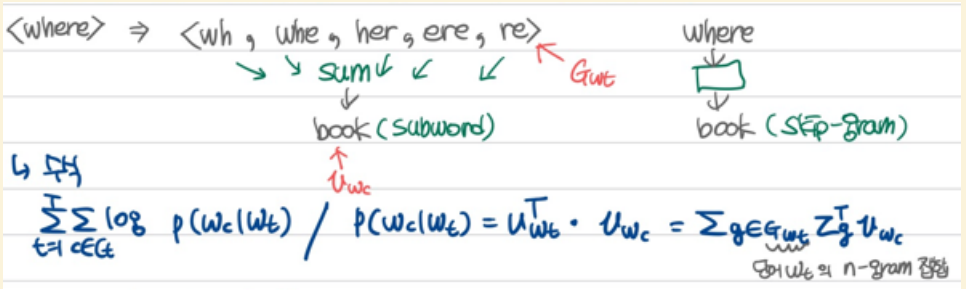
  학습 말뭉치에서 단어 간 동시 발생 빈도 사용

  (단어 간 동시 발생 : 비율이 벡터 공간 상 차이 되도록 임베딩 수행)

- FastText (Subword + Skip-gram)

  skip-gram을 사용. unknown 단어(어휘사전에 없는 단어) 문제 어느정도 해결 가능

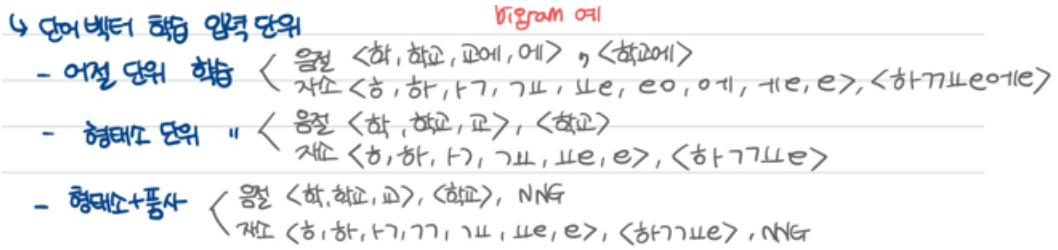
  - subword 모델: 단어 벡터들을 n-gram 벡터의 합으로 나타냄



- 한국어 단어 분리

  - 분리 단위 (6)ㅂ

    어절, 형태소, 음절, 자소, 천지인, 품사포함



n-그램 언어 모델

- 통계적 언어 모델
  - 주어진 단어를 기반으로 확률이 가장 높은 다음 단어 예측

- 조건부 확률의 연쇄법칙 사용 문장 등장 확률

$$\prod_{i=1}^{n} P(w_i|w_1, w_2, \cdots, w_{i-1})$$

- 마르코프 가정 사용 연쇄법칙 복잡성 해결, 간소화

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

- n-gram 언어 모델

  특정 단어가 주변 존재하는 몇 단어(n-gram)와 연관된다고 가정

예시 : The boy is looking at a pretty girl
- 1-gram(unigram)

The | boy | is | looking | at | a | pretty | girl
- 2-gram(bigram)

The boy | boy is | is looking | looking at | at a | a pretty | pretty girl
- 3-gram(trigram)

The boy is | boy is looking | is looking at | looking at a | at a pretty | a pretty girl

  - 확률 계산

- unigram : $\prod_{i=1}^{n} P(w_i)$
- bi : $P(w_i|w_{i-1})$

❖ **bigram 언어 모델의 계산 예**

<s> I eat an apple </s>
<s> an apple I eat </s>
<s> I like cheese cake </s>

bigram:
(<s>, I), (I, eat), (eat, an), (an, apple), (apple, </s>)
(<s>, an), (an, apple), (apple, I), (I, eat), (eat, </s>)
(<s>, I), (I, like), (like, cheese), (cheese, cake), (cake, </s>)

unigram count:
<s>: 3
I: 3
eat: 2
an: 2
:

bigram count:
(<s> I): 2
(<s> an): 1
(I eat): 2
(eat an): 1
:

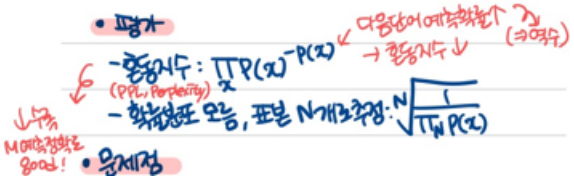
$$P(I|<s>) = \frac{cnt(<s>,I)}{cnt(<s>)} = \frac{2}{3} = 0.67$$

$$P(an|<s>) = \frac{cnt(<s>,an)}{cnt(<s>)} = \frac{1}{3} = 0.33$$

$$P(eat|I) = \frac{cnt(I,eat)}{cnt(I)} = \frac{2}{3} = 0.67$$

$$P(an|eat) = \frac{cnt(eat,an)}{cnt(eat)} = \frac{1}{2} = 0.5$$

  - 평가



  - 문제점
    - 한계

      생성 문장 부자연스러움, 학습코퍼스따라 확률값 다름, 방대한 코퍼스 필요
    - 희소성 문제

새로운 데이터셋 제대로된 예측 불가, 모든 언어 조합 말뭉치 구축 힘듦

- 마르코프 가정 문제

  바로 전 단어와만 연관있다는 가정, 장기의존성 간과

  - 장기의존성: 멀리 떨어진 단어와의 의존 관계를 가지는 언어 특성

- 평탄화 (단어 희소성 문제 해결 방법)

  학습 말뭉치가 없을 경우 단어열로 인한 전체 문장 확률이 0 되는 것 방지

  - 스무딩: 등장 없던 단어 조합에 작은 확률 값 부여
  - 보간법: n-gram 확률을 이전 n-gram 확률과 혼합해 사용
  - 백오프: 현재 n-gram 확률값이 = 0 → 작은 n-gram 확률 사용

- 신경망 언어

  단어를 벡터로 표현. 벡터를 신경망에 넣고 계산해 다음 단어 예측

  - 장점: 저장공간 불필요, 데이터 평탄화 유리
  - 단점: 문맥 한계가 여전히 존재

- RNN

  은닉층 출력을 다시 입력으로 사용, sequence length 단계 만큼 시간 필요

  - 장점: n-gram 한계 해결
  - 단점: 먼거리 정보 상실(지역성), 제한된 문맥 (왼↔오 한방향), 병렬처리 제한→대용량데이터 계산 한계

- attention
  - 장점: 먼거리 정보 처리 가능, 정보 병목 현상 해결
  - 단점: 직렬처리 → 속도 저하

트랜스포머

- self-attention

  각 단어 벡터를 질의어로 사용(→ 모든 단어는 이전 계층의 모든 단어 주의 집중)

  각 값들 적절히 합. 집합연산이며 순서 필요 없음

  O(n^2) 계산 복잡도

- 트랜스포머 모델 구축 위한 개선
  - 입력 단어 순서 부여 → 위치 정보 벡터를 인덱스 위치 키에 추가 (언어는 순서가 중요하므로)
  - 비선형 계산을 위한 feed forward 계층 추가 (ReLU 같은것 추가)
  - 미래 단어 볼 수 없게 mask 처리 → 디코더 학습 시 attention 값 최소화 해 mask 처리
  - multi-headed attention (MHA) → 쿼리 분할 ⇒ softmax

    - 잔차 연결 (출력을 그대로 입력 + @ ⇒ 더 좋음)
    - 층 정규화 학습 시 속도 향상
    - scaled dot product : softmax값 낮을수록 gradient 값 높게 학습
- 특징
    - self-attention → 모든 단어를 주의 집중하여 장거리 정보 참조 가능, 정보 병목 현상 해결 (디코더 Q로 인코더 K, V 연결)
    - 병렬 처리 가능 → 속도 향상
    - RNN 없이 attention 만 사용 → 인코더-디코더 만듦 (N층 구축 가능)
    - RNN 보다 훨씬 빨리 학습 → 성능 대폭 향상
- 구조 (E or D or E+D)
    - E: MHA과 feed forward 연결로 구성
    - D: MHA를 mask 하여 q 구함

      → 인코더에서 온 k, v와 합함

      → 다시 MHA + FF 구조 입력으로 사용

거대 언어 포델 (LLM)

- 토큰화
    - 필요성
        - 모든 단어 어휘 벡터 표현 어려움
        - 새로운 단어 표기법 필요
        - 입력 벡터 크기 최적화
    - 효과
        - 단어를 통계에 기반, 부분 단어(토큰)로 분리 → 미등록어 최소화
        - 최적화된 입출력 단위 처리
    - 방법

      단어를 토큰으로 분리해 처리 / 압축 기술 사용 → 최적 토큰으로 분리
    - 입출력 제한

      실제 트랜스포머 입력 크기는 토큰수로 제한 (단어수x) 출력 후 다시 단어로 결합. 출력 단어수오ㅏ 다름
- Byte Pair Encoding