한국어 임베딩

- 1장 서론 -

2019. 10. 29 Eden

1장에서 다루는 내용

- 1.1 임베딩이란
- 1.2 임베딩의 역할
  - 1.2.1 단어/문장 간 관련도 계산
  - 1.2.2 의미/문법 정보 함축
  - 1.2.3 전이 학습
- 1.3 임베딩 기법의 역사와 종류
  - 1.3.1 통계 기반에서 뉴럴 네트워크 기반으로
  - 1.3.2 단어 수준에서 문장 수준으로
  - 1.3.3 룰 -> 엔드투엔드 -> 프리트레인/파인 튜닝
  - 1.3.4 임베딩의 종류와 성능
- 1.4 개발 환경
  - 1.4.1 환경 소개
  - 1.4.2 AWS 구성
  - 1.4.3 코드 실행
  - 1.4.4 버그 리포트 및 Q&A
  - 1.4.5 이 책이 도움받고 있는 오픈소스들
- 1.5 이 책이 다루는 데이터와 주요 용어
- 1.6 이 장의 요약
- 1.7 참고 문헌

1.1 임베딩이란

1.1 임베딩이란

- 자연어 처리 분야에서 임베딩이란,
  - 사람이 쓰는 자연어를 기계가 이해할 수 있는 숫자의 나열인 벡터로 바꾼 결과 혹은 그 일련의 과정 전체를 의미

- 임베딩 이름 유래
  - 단어나 문장 각각을 벡터로 변환해 벡터 공간으로 '끼워 넣는다(embed)'는 의미

1.2 임베딩의 역할

1.2 임베딩의 역할

(1) 단어/문장 간 관련도 계산

(2) 의미적/문법적 정보 함축

(3) 전이 학습

1.2 임베딩의 역할 (1) 단어/문장 간 관련도 계산

- 단어를 벡터로 임베딩하는 순간 단어 벡터들 사이의 유사도(similarity)를 계산하는 일이 가능해짐
- 예) L1 distance, L2 distance, 코사인 유사도 등등

표 1-2 쿼리 단어들의 코사인 유사도 기준 상위 5개 단어 목록

희망	절망	학교	학생	가족	자동차
소망	체념	초등	대학생	아이	승용차
행복	고뇌	중학교	대학원생	부모	상용차
희망찬	절망감	고등학교	고학생	편부모	트럭
꿈	상실감	야학교	교직원	고달픈	대형트럭
열망	번민	중학	학부모	사랑	모터사이클

- 각 쿼리 단어별로 벡터 간 유사도 측정 기법의 일종인 코사인 유사도(cosine similarity) 기준 상위 5개를 나열한 것

1.2 임베딩의 역할 (2) 의미/문법 정보 함축

- 임베딩은 벡터인 만큼 사칙연산이 가능
- 단어 벡터 간 덧셈/뺄셈을 통해 단어들 사이의 의미적, 문법적 관계를 도출할 수 있음

- 단어 유추 평가 (word analogy) :
  - (a 단어 벡터) + (b 단어 벡터) - (c 단어 벡터) 를 계산
  - Ex 1) '아들 - 딸 + 소녀 = 소년' 이 성립하면 성공적인 임베딩
  - Ex 2) '신랑 - 신부 + 왕 = 여왕' 이 성립하면 성공적인 임베딩

1.2 임베딩의 역할 (3) 전이 학습

- 전이 학습 (transfer learning)
  - 임베딩을 다른 딥럽닝 모델의 입력값으로 쓰는 기법
- HOW?
  - 대규모 말뭉치를 활용해 임베딩을 미리 만들어 놓음
  - 임베딩에는 의미적, 문법적 정보 등이 녹아 있음
  - pretrain / fine tuning
- 임베딩이 중요한 이유
  - 임베딩의 품질이 좋으면, 수행하려는 task의 성능이 올라감
  - 임베딩의 품질이 좋으면, 모델의 수렴(converge)이 빨라짐

1.3 임베딩 기법의 역사와 종류

1.3 임베딩 기법의 역사와 종류

- 임베딩 기법의 역사
  - 통계 기반 -> 뉴럴 네트워크 기반
  - 단어 수준 -> 문장 수준
  - 룰 -> 엔드투엔드 -> 프리트레인/파인 튜닝

- 임베딩의 종류와 성능

1.3.1 통계 기반 -> 뉴럴 네트워크 기반

- 초기 임베딩 기법은 대부분 말뭉치의 통계량을 직접적으로 활용하는 경향
  - 잠재 의미 분석 (Latent Semantic Analysis, LSA)
  - 단어 사용 빈도 등 말뭉치의 통계량 정보가 들어 있는 커다란 행렬에 특이값 분해 등 수학적 기법을 적용해 행렬에 속한 벡터들의 차원을 축소하는 방법 -> 4,5장

- 최근에는 뉴럴 네트워크 기반의 임베딩 기법들이 주목받고 있음
  - 이전 단어들이 주어졌을 때 다음 단어가 뭐가 될지 예측하거나, 문장 내 일부분에 구멍을 뚫어 놓고(masking) 해당 단어가 무엇일지 맞추는 과정에서 학습됨
  - 5장

1.3.2 단어 수준 -> 문장 수준

- 단어 수준 임베딩 (NPLM, Word2Vec, GloVe, FastText, ...)
  - 각각의 벡터에 해당 단어의 문맥적 의미를 함축
  - 단점 : 동음이의어를 분간하기 어려움 (4장)

- 문장 수준 임베딩 (ELMo, BERT, GPT, ...)
  - 개별 단어가 아닌 단어 시퀀스 전체의 문맥적 의미를 함축하기 때문에 단어 임베딩 기법 보다 전이 학습 효과가 좋음
  - 동음이의어들을 분리해 이해할 수 있음 (5장)

1.3.2 룰 -> 엔드투엔드 -> 프리트레인/파인 튜닝

- 1990년대 : 자연어 처리 모델 대부분은 사람이 피처를 직접 뽑음
- 2000년대 중반 이후 : 자연어처리 분야에서도 딥러닝 모델이 주목받기 시작
  - End-to-end model : 데이터를 통째로 모델에 넣고 입출력 사이의 관계를 사람의 개입 없이 모델 스스로 처음부터 끝까지 이해하도록 유도
- 2018년 이후 : pretrain과 fine tuning 방식으로 발전 (ELMo 이후)
  - pretrain : 우선 대규모 말뭉치로 임베딩을 만듦
  - fine tuning : 임베딩을 입력으로 하는 새로운 딥러닝 모델을 만들고 우리가 풀고 싶은 구체적 문제에 맞는 소규모 데이터에 맞게 임베딩을 포함한 모델 전체를 업데이트함

1.3.3 룰 -> 엔드투엔드 -> 프리트레인/파인 튜닝

- Downstream task : 우리가 풀고 싶은 자연어 처리의 구체적 문제들
  - 예) 품사 판별, 개체명 인식, 의미역 분석

- Upstream task : 다운스트림 태스크에 앞서 해결해야 할 과제
  - 단어/문장 임베딩을 프리트레인 하는 작업

- 임베딩 품질이 좋아야 문제를 제대로 풀 수 있다!

1.3.4 임베딩 종류와 성능

- 행렬 분해
  - 말뭉치 정보가 들어있는 원래 행렬을 2개 이상의 작은 행렬로 분해한 이후 둘 중 하나의 행렬만 쓰거나 sum 하거나 concat 하여 사용하는 방식의 임베딩 기법 (4장)
- 예측
  - 어떤 단어 주변에 특정 단어가 나타날지 예측하거나, 이전 단어들이 주어졌을 때 다음 단어가 무엇일지 예측하거나, 문장 내 일부 단어를 지우고 해당 단어가 무엇일지 맞추는 과정에서 학습하는 방법 (4, 5장)
- 토픽 기반
  - 주어진 문서에 잠재된 주제를 추론하는 방식으로 임베딩을 수행 (5장)
  - 예) 잠재 디리클레 할당, LDA

1.3.4 임베딩 종류와 성능

- 임베딩 성능 평가 (4,6 장)
  - 다운스트림 태스크에 대한 임베딩 종류별 성능 분석
  - 다운스트림 태스크
    - 형태소 분석, 문장 성분 분석, 의존 관계 분석, 의미역 분석, 상호 참조 해결
  - 문장 임베딩 기법인 ELMo, GPT, BERT가 단어 임베딩 기법인 GloVe를 크게 앞섬

- 임베딩 품질이 각종 다운스트림 태스크 성능에 크게 영향을 주고 있음

1.4 환경 소개

모두 환경 설정 하셨나요?

저는 실패했어요..

1.5 이 책이 다루는 데이터와 주요 용어

- 이 책이 다루는 데이터는 text 형태의 자연어

- Corpus 말뭉치 : 임베딩 학습이라는 특정한 목적을 가지고 수집한 표본
- Collection : 말뭉치에 속한 각각의 집합
- Sentence 문장 : 이 책이 다루는 기본 단위
- Document 문서 : 생각이나 감정, 정보를 공유하는 문장 집합
- Paragraph 단락 : 책에서는 문서와 단락을 굳이 구분하지 않겠음

- Token : 이 책에서 다루는 가장 작은 단위. 문장은 여러 개의 토큰으로 구성됨
  - 문맥에 따라서는 word(단어), morpheme(형태소), subword 라고 부를 수 있음
- Tokenize : 문장을 토큰 시퀀스로 분석하는 과정
- Vocabulary 어휘 집합 : 말뭉치에 있는 모든 문서를 문장으로 나누고 여기에 토크나이즈를 실시한 후 중복을 제거한 토큰들의 집합
- Unknown word 미등록 단어 : 어휘 집합에 없는 토큰

퀴즈

- fine tunning 과 transfer learning의 차이가 무엇일까요?

감사합니다!