송문기 MoonGhi Song

NLP Machine Learning Engineer

| Contact 010 0000 0000 | Portfolio Link
| Notion Link | Github github.com/moon23k

Skill Set

- Pandas, Numpy, Matplotlib
- Shell Script, BeautifulSoup
- Scikit Learn, TensorFlow, Pytorch, HuggingFace
- Django, Flask, HTML

프로젝트

1. 자연어 생성 과제를 위한 세 가지 트랜스포머 모델의 성능 결과 비교

상세 설명
보러 가기

1) Standard, 2) Recurrent, 3) Evolved 트랜스포머 모델을 직접 구현
각 모델을 세 가지 자연어 생성 과제(번역, 대화 생성, 요약)에서 비교 분석

코드 구현
보러 가기

- Evolved Transformer: 각 과제별 00%, 00%, 00% 성능 기록
- Recurrent Transformer: 각 과제별 00%, 00%, 00%성능 기록
  (Inductive Bias가 중요한 소규모 훈련환경에 적합)

TBD: 각 트랜스포머 모델로 BERT 구현 후 성능 비교

2. 문서 요약을 위한 BERT 활용 방법론 별 성능 결과 비교

상세 설명
보러 가기

BERT의 입력 시퀀스 제한사항 해결을 위해 BERT Sum의 방법론과,
디코더 연결을 위한 세 가지 방법론(Simple, Fused, Generative)을 통합적으로 고려한,
총 3가지 모델을 구현하고, 각 모델의 문서 요약 성능을 Base Line Model과 비교

코드 구현
보러 가기

- BERTSum + Simple Model: 00%의 성능 개선
- BERTSum + Fused Model: 00%의 성능 개선
- BERTSum + Generative Model: Base Line 대비 00%의 성능 개선

3. 대화 생성 모델의 개성 부여를 위한 SeqGAN 의 활용

상세 설명
보러 가기

모델의 예측 분포를 레이블 데이터 분포에 근사하는 학습 방식에서 기인하는
일반적이고 재미없는 대화 생성의 문제를 SeqGAN 기법으로 해결하고,
인기 시트콤 *How I Met Your Mother* 등장 인물의 개성을 선택적으로 부여

코드 구현
보러 가기

Image will Attached

4. 의미적으로 다양한 대화 생성을 위한 학습 방법론 구현

상세 설명
보러 가기

대화 생성 모델의 적절하고 다양한 발화 가능성 제고를 위해 일반적 생성적 학습 방법론과
의미 단위 구분을 통한 생성적 학습방법론을 각각 구현하고, 성능 비교

코드 구현
보러 가기

- 단순 글자 단위의 상이함이 아닌, 의미론적인 다양성 제고 확인

5. 컴퓨팅 자원의 효율적 사용을 위한 학습 방법론 별 성능 비교

상세 설명
보러 가기

일반적인 사용자의 입장에서 딥러닝 모델 사이즈 및 데이터 볼륨을 최대한 유지하며,
연산 자원을 효율적으로 사용하기 위한 5가지 학습 기법 적용에 따른 메모리 및 성능 비교

코드 구현
보러 가기

- 학습 방법론: Gradient Checkpoint, AMP, Accumulative Update
- 최대 메모리 사용량 00% 감소 성능 00 기록(향상 or 하락)

6. 데이터 부족 해결을 위한 BackTranslation 조건 별 성능 비교

상세 설명
보러 가기

BART 사전학습 모델을 통해 SRC Text 를 생성하고, 데이터의 사이즈, 디코딩 전략, 노이즈
삽입이라는 변인에 따른 번역 성능 결과 비교

코드 구현
보러 가기

- 생성 데이터의 사이즈가 가장 큰 변인으로 작용
- 학습 데이터 생성의 측면에서 None MAP 디코딩 방식의 우수성 확인
- 언어 특성에 따라 상이한 노이징 전략 사용시 최대 00% 성능 향상

자기 소개

함께 일하고 싶습니다!

기계 번역 품질의 향상을 위한 BackTranslation 및 생성적 학습법 구현, 다양한 발화 가능성의 제고를 위해 의미 유사도 클러스터링을 활용한 학습법 및 개성부여 방법 구현, 그리고 다양한 사전학습 모델을 활용해 요약업무에 발전가능성을 제고시키기 위한 6 가지 방법론의 비교 실험 등 스스로 문제를 찾아, 해결 방안을 위한 새로운 지식을 습득하고, 코드 구현으로 결과를 도출하는 일련의 과정을 즐깁니다. 하지만 저의 지식과 결과물은 세상 사람들과는 단절된 채, 좁은 울타리 안에서만 머물고 있습니다. 이제는 실제 서비스 구현을 통해 많은 사람들과 나눌 수 있는 개발자가 되어 보려 합니다. (지원 기업별 서비스에 따라 적용 가능 방안을 추가해서 어필할 예정)

스스로의 장단점을 정확히 인지하고, 단점을 극복합니다!

똥인지 된장인지 찍어먹어 봐야 압니다. 새로운 지식에 대한 직관적 이해도가 부족하다는 단점을 경험주의적으로 극복합니다. 일례로 트랜스포머를 공부하며, 그 대단함과 대략적인 작동원리를 접했습니다. 하지만 쿼리, 키, 밸류 간 내적의 의미부터 레이어의 연결방식까지 글로는 도저히 이해되지 않았습니다. 때문에 트랜스포머를 처음부터 끝까지 직접 구현하며 결과값을 확인하고 나서야 트랜스포머를 이해하고 사용할 수 있게 되었습니다. 덕분에 트랜스포머에 대한 깊은 이해도를 갖게 되어, 트랜스포머의 개선 모델을 추가로 구현하며, 의미 있는 실험 결과까지 도출할 수 있었습니다.

실패를 통해 더 크게 성장합니다!

개발은 지속적인 실패 속에서도 다음 과정으로 나아가며, 결국 뜻했던 바를 달성해내는 과정이라고 생각합니다. 저는 단일 문장을 넘어, 문서 수준의 데이터를 제대로 처리하기 위한 사전학습 모델의 활용이라는 주제의 프로젝트 진행한 경험이 있습니다. 개별 문장의 정확한 의미 포착을 위한 반복문의 활용, 필요 시점마다 입력 값의 차원을 조절하는 방식 등 다양한 접근방식을 취했지만 모두 실패로 돌아갔습니다. 하지만 앞선 실패 경험으로 접근방식에 대한 갈피를 잡고 성공적인 결과를 만들어냈으며, 나아가 멀티턴 대화 생성 모델을 구축하는데 큰 자양분으로 삼을 수 있었습니다.

교육 이수

- 동국대학교 영어통번역, 법학 전공
- Code States AI BootCamp 수료

오픈 소스 기여

- Paperswithcode 에 게제 예정(TBD)