2018 학년도『KU 학부연구생 4 기』최종 결과보고서

연구주제	기계학습 기법을 이용한 오픈소스 기반 문서 요약 시스템에 관한 연구		
성명	임혜수	**학번**	2017320160
단과대학	정보대학	**학부(과)**	컴퓨터학과
지도교수 면담 일시	2018 년 11 월 13 일(화) 19:00 ~ 19:30 / 라이시움 309 호		

<최종 결과 보고서 작성 유의사항>

1. 글자크기는 10 point / 줄간격 160% 또는 1.0
2. 최종 결과 보고서 분량은 최소 10 page 이상
3. 구체적이고 논리적인 문장으로 기술
4. 논문 작성 틀(표, 참고문헌)은 APA 논문 작성법 또는 해당 계열 논문 작성법 양식에 맞춰 기술

연구주제	기계학습 기법을 이용한 오픈소스 기반 문서 요약 시스템에 관한 연구

Ⅰ. 초 록(Abstract)

온라인 상에 존재하는 데이터의 양은 방대하다. 점점 더 빠른 속도로 데이터는 생성되는데 많은 양의 데이터에서 원하는 정보를 빠르고 정확하게 얻기 위해서는 긴 글에 대한 간결한 요약이 필요하다. 문서 요약 시스템은 긴 문서로부터 짧은 요약문을 자동으로 만들어주는 시스템이다. 크게 추출 요약법과 추상 요약법으로 나뉘며 사람이 만들어 내는 것과 같이 간결하고 의미 있는 요약문을 만들어 내기 위해서는 아직 더 많은 연구가 필요하다. 이전까지는 자연어 생성 기술을 필요로 하지 않는 추출 요약법이 훨씬 더 좋은 성능을 보였으나 추상 요약법에서 딥러닝을 활용하면서 추상 요약법의 성능이 크게 향상되었다. 본 연구에서는 Nallapati et al.(2016) 에서 발표한 RNN 을 활용하여 시퀀스-투-시퀀스 주목 모델을 이용한 문서 요약 시스템인 Textsum 모델을 분석, 실행함으로써 추상 요약법에 대한 연구를 진행한다.

Ⅱ. 서 론

가. 이론적 배경

최근 온라인 상에 존재하는 데이터의 양이 방대해지고 있다. DOMO 에서 발표한 sixth edition of DOMO's report 에 따르면 매일 2.5 조 바이트 이상의 데이터가 생성되고 있으며 2020 년에는 초당 모든 사람이 1.7MB 의 데이터를 생성할 것이다.

데이터의 양이 방대할수록 사용자가 원하는 정보를 찾기 위해 투자해야 하는 시간은 길어졌다. 이에 따라 사용자에게 의미 있는 정보에 빠르고 정확하게 접근하여 처리할 수 있는 능력에 대한 필요성이 커지고 있다. 긴 문서에 대한 의미 있고 간결한 요약을 만들어내는 문서 요약 시스템의 중요성은 그에 따라 점점 커지고 있다.

문서 요약 시스템의 활용

문서 요약 시스템은 다양한 분야에서 활용될 수 있다. 문서 요약 시스템이 활용될 수 있는 분야에 대한 논문인 Khargharia et al. (2018)에 따르면 문서 요약 시스템은 미디어에 공개된 정보를 요약하여 관광에 관련된 데이터를 만들 수 있고, 법률 문서를 요약하는 데에도 사용될 수 있다. 또, 책, 뉴스, 의학 문서, 생물 의학 문서 등을 요약하는 등 문서 요약 시스템이 다양한 분야에서 활용될 수 있다. 문서 요약 시스템은 많은 양의 문서를 모두 읽지 않고도 그 문서가 말하고자 하는 내용을 파악할 수 있도록 하는데, 이는 방대한 양의 정보를 필요로 하는 법률과 의학 분야에 큰 도움을 줄 수 있을 것으로 보인다.

문서 요약 시스템의 이론적 배경

문서 요약 시스템은 주어진 문서로부터 가장 중요한 정보, 즉 전체 문서에 관련되어 있는 의미 있는 정보를 찾아 짧게 요약하는 시스템이다. 문서 요약 시스템의 대표적인 두가지 접근법으로는 추출 요약법 (Extraction-based summarization)과 추상 요약법 (Abstraction-based summarization)이 있다.

추출 요약 방법은 요약문을 만들기 위해 문서 내에 존재하는 단어, 구문, 문장을 선택한다. 이때 선택되는 문서의 일부는 TF-IDF[1]와 같은 기준에 의해서 중요하다고 판단된 단어, 구문, 문장이다. 다시 말해 기존의 데이터를 수정하지 않고 그 문서에서 가장 중요하다고 생각되는 문장을 복사해서 요약문에 붙이는 과정으로 진행된다. 기존 문서에서 사용되는 어휘를 그대로 사용하여 요약문을 만들기 때문에 사람이 작성한 요약문에 비해 어색한 경우가 많다.

반면 추상 요약 방법은 기존 문서로부터 추출한 어휘와 문장에 국한하지 않고 기존 문서(source document)로부터 paraphrase 하는 과정이 포함되어 있다. 기존 문서에는 등장하지 않는 단어를 이용하여 더 의미 있는 요약문을 만들어내기 때문에 일반적으로 추상 요약 방법이 추출 요약 방법보다 더 강력한 형태의 요약문을 만들어낼 수 있다. 자연어 생성 기술을 필요로하기 때문에 추출 요약 방법을 개발하는 것 보다는 어렵지만, 추출 요약 방법 보다는 사람이 표현한 것과 유사하게 요약문을 만들어낼 수 있다.

Textsum 의 이론적 배경

Textsum 은 2016 년 8 월에 Google Brain team 에서 공개한 텐서플로우 모델이다. 이 모델은 추상 요약법을 사용하고 Annotated English Gigaword [2]의 헤드라인을 만들어내는 모델이다. 이 모델을 공개할 당시에는 요약문을 평가하는데 흔하게 사용되는 기준인 ROUGE[3]로 평가하였을 때 가장 좋은 성능을 보였다.

Textsum 에서는 시퀀스-투-시퀀스 (sequence-to-sequence) 모델로 요약문 생성을 학습한다. 뉴스 기사의 첫 몇 줄만을 보고 헤드라인을 학습하는 과정을 거친다. 뉴스 데이터에 대해서는 문서의 일부만을 모델의 input 으로 해도 좋은 요약문을 생성할 수 있으나 문서의 모든 내용을 읽어야만 하는 데이터에 대해서는 좋은 성능을 보이지 못한다.

추출 요약법에 비해 추상 요약법은 자연어 생성 기술을 필요로 하기때문에 좋은 성능을 내기 어렵다. 그러나 문서 요약 시스템에서의 궁극적인 목적은 사람이 만들어내는 것과 유사한 요약문을 생성하는 것이므로 추상 요약법의 성능을 좋게 하는 방법이 필요하다. RNN 등의 딥러닝을 문서 요약 시스템에 활용하면서 추상 요약법 또한 좋은 성능을 보일 수 있게 되었다. Nallapati et al.(2016)에서는 시퀀스-투-시퀀스 주목 모델을 이용하여 추상 요약 모델을 만들었고 ROUGE 를 기준으로 하여 가장 좋은 성능을 보였다. Github 에 공개된 오픈소스는 Tensorflow 와 python 을 이용한다. Textsum 모델을 통해 추상 요약법의 성능을 높이는데 기여한 RNN 의 문서 요약 시스템에서의 활용을 잘 분석할 수 있을 것이라 판단되어 이 모델과 함께 공개된 데이터와 다른 데이터를 이용해 Textsum 모델을 분석하고자 한다.

---

[1] TF-IDF(Term Frequency - Inverse Document Frequency)는 정보 검색과 텍스트 마이닝에서 이용하는 가중치로, 여러 문서로 이루어진 문서군이 있을 때 어떤 단어가 특정 문서 내에서 얼마나 중요한 것인지를 나타내는 통계적 수치이다.

[2]Annotated English Gigaword 는 John's Hopkins University 에서 개발한 데이터로 영어 뉴스 기사 데이터이다. 다른 자동 요약 연구에도 많이 사용되는 데이터이다.

[3] ROUGE 는 자동 요약이나 번역하는 자연어 처리 소프트웨어를 평가하는 기준 중 하나이다.

Ⅲ. 연구내용

가. 연구 대상 (모델 설명)

Nallapati et al.(2016) 에 따르면 Textsum 은 시퀀스-투-시퀀스 주목 모델 (sequence-to-sequence attention model)을 사용한다. 이 모델은 크게 인코더 (Encoder), 디코더 (Decoder), 주의 분포 (Attention Distribution) 와 문맥 벡터(Context Vector), 어휘 분포(Vocabulary Distribution) 네 부분으로 구성된다.

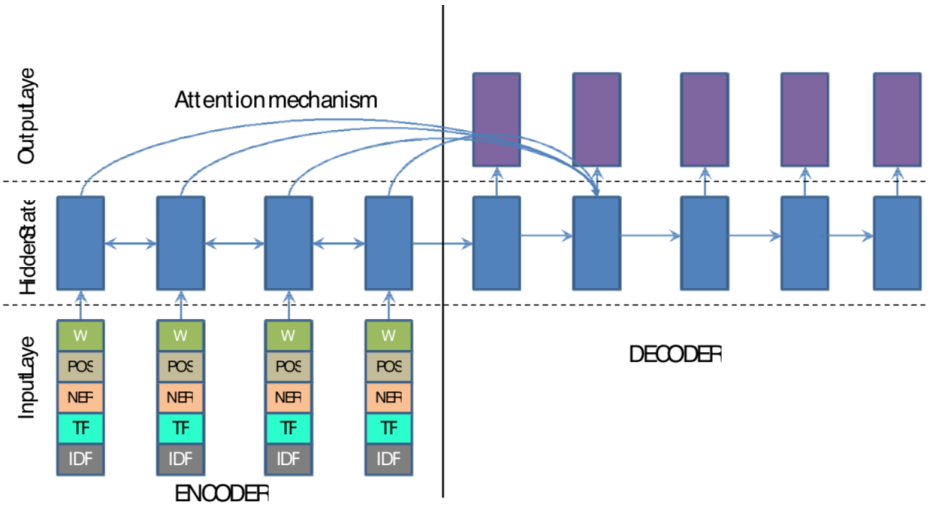
인코더와 디코더

Textsum 모델의 기준이 되었던 모델은 Bahdanau et al.(2014) 에서 사용되었던 뉴럴 기계 번역 모델이다. 이 모델은 양방향의 인코더와 단방향의 디코더, 주의 메커니즘 (attention mechanism) 과 소프트 맥스 (soft-max) 층으로 구성된다. 이 모델을 기본으로 하여 Textsum 모델은 Jean et al. (2014) 에서 설명되어 있는 Large Vocabulary Trick (LVT)라고 불리는 요약 문제를 해결하였다. 이 문제를 해결하기 위해서 디코더 어휘를 소스 문서에서 사용된 어휘로 제한하였다. 또한 디코더 어휘가 일정 크기에 도달하지 못할 경우에는 디코더 어휘가 일정 크기에 도달할 때까지 디코더 어휘에 타겟 사전에서 가장 빈도수가 높은 단어를 추가한다. 이는 계산하는데 가장 큰 병목이었던 소프트 맥스 층의 크기를 감소시키기 위함이다. 이에 더불어 주어진 샘플에서 중요한 부분에 집중함으로써 Textsum 모델은 계산 속도를 향상시킨다. 요약문을 만들기 위한 중요한 부분은 소스 문서로부터 가져오는 경우가 많기 때문에 이 기술은 문서 요약에 적합한 기술이다.

Feature-rich Encoder 를 이용한 핵심 단어 파악

요약문을 만들기 위해 가장 중요한 것은 소스 문서에서 핵심 주제가 무엇인지, 중요한 부분이 어디인지 파악하는 것이다. 이 목표를 성취하기 위해서는 소스 문서의 단어 임베딩 표현[4]에 더불어 parts-of-speech (품사), named-entity-tags (이름), TF-IDF 통계 등의 부가적인 언어적 특징을 파악해야 한다. 따라서 단어 임베딩과 유사하기 어휘 태그 유형에 부가적인 임베딩 행렬을 추가한다. TF-IDF 와 같은 연속적인 성질에 대해서는 이산화하여 one-hot 인코딩 과정을 거침으로써 해당하는 태그 유형에 대응시킨다. 마지막으로는 [그림 1]에서 볼 수 있는 것과 같이 소스 문서에 있는 단어 임베딩과 태그를 붙여서 하나의 긴 벡터로 만든다.

[4] 단어 임베딩 표현 (word-embedding) 은 word2vec 과 같이 단어를 수치화하는 방법의 일종이다.



[그림 1] : Feagrue-rich encoder : 단어 임베딩과 POS 태그, NER 태그, 이산화된 TF-IDF 값을 붙여 만들어진 하나의 임베딩을 사용한다.

생성-포인터 전환을 이용한 모델링

종종 요약을 할 때 테스트 과정에서 만들어진 요약의 핵심 어휘나 named-entity 가 훈련 과정에서 사용되지 않았거나 등장하지 않는 경우가 있다. 디코더 어휘는 훈련 과정에서 정해지기 때문에 훈련 과정에서 등장하지 않은 단어일 경우 디코더가 처리할 수 없다. 이러한 단어를 Out-of-vocabulary (OOV) 라고 부르는데, 이를 처리하는 가장 흔한 방법은 OOV 를 unknown 을 의미하는 'UNK'로 표현하는 것이다. 하지만 이 방법은 의미있는 요약문을 만들어내지 않는다. 요약문에서 OOV 를 처리할 수 있는 또 다른 방법으로는 테스트에 사용되는 소스 문서에서 OOV 가 등장하는 위치를 가리키는 것이다.

Textsum 모델에서는 [그림 2] 에 나와있는 switching decoder/pointer 구조를 사용한다. 디코더는 스위치와 결합되어 있는데 이 스위치는 생성기 (generator) 를 사용할 것인지 포인터 (pointer) 를 사용할 것인지 결정한다. 스위치가 켜져 있는 경우에는 디코더는 일반적인 방법으로 타겟 어휘로부터 요약문에 사용될 단어를 만들어낸다. 반면 스위치가 꺼져 있는 경우에는 요약문에 사용될 단어의 위치를 소스 문서에 찾아 가리키는 포인터를 사용한다. 이 경우 디코더는 포인터가 가리키는 위치에 있는 단어를 복사하여 요약문을 만들 때 사용한다. 스위치는 선형 레이어에서 시그모이드 활성화 함수 (sigmoid activation function) 로 모델링되어 있다.

$$P(s_i = 1) = \sigma(v^s \cdot (W_h^s h_i + W_e^s E[o_{i-1}] + W_c^s c_i + b^s))$$

위 식에서$P(s_i = 1)$은 디코더에서 i번째 스텝에 스위치가 켜질 확률, h_i는 은닉 상태 (hidden state), $E[o_{i-1}]$ 은 이전 스텝에서 나오는 임베딩 벡터, c_i 는 주의 가중치가 부여된 문맥 벡터, W_h^s, W_e^s, W_c^s, b^s, v^s 는 스위치 인자(parameter) 를 의미한다. 포인터를 샘플링하는 분포로는 문서의 단어 위치에 대한 주의 분포를 사용한다.

$$P_i^a(j) \propto exp(v^a \cdot (W_h^a h_{i-1} + W_e^a E[o_{i-1}] + W_c^a h_j^d + b^a)),$$

$$p_i = argmax_j(P_i^a(j)) \; for \; j \in \{1, \ldots, N_d\}.$$

위 등식 에서 p_i는 요약문 안에서 i번째 단어 위치에 대한 포인터 값을 의미한다. 이 포인터 p_i는 주의 분포인 P_i^a 로 부터 샘플링된 값이다. 문서에서의 단어 위치인 $j \in \{1, \ldots, N_d\}$에 대하여 $P_i^a(j)$는 디코더의 i번째 스텝에서 문서의 j번째 위치를 가리킬 확률이고 h_j^a는 인코더의 j번째 위치에 있는 은닉 상태를 의미한다.

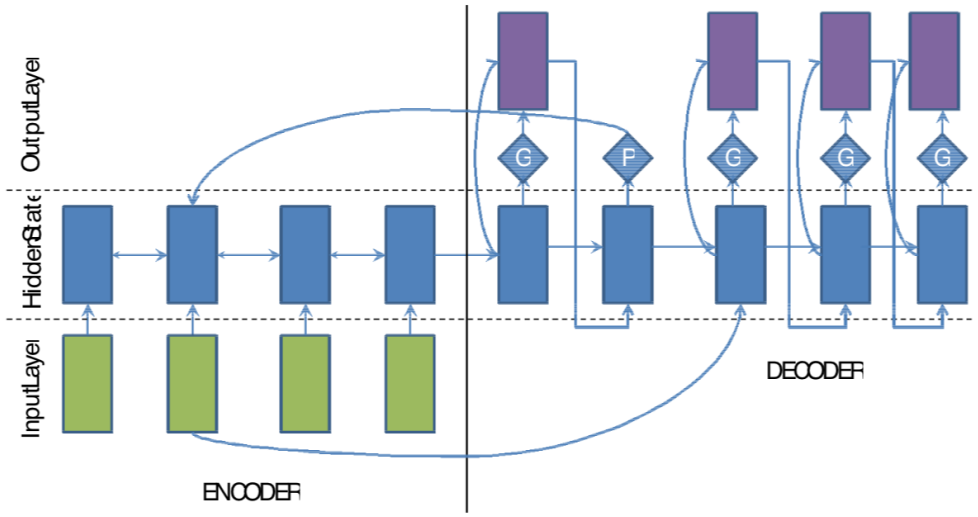
훈련 과정에서 요약문에 사용되는 단어가 타겟 어휘에 없을 경우에는 명시적인 포인터 정보를 모델에게 제공한다. 만약 oov 단어가 여러 문서에서 발견된다면 처음으로 발견되는 문서를 선택한다. 훈련 과정에서 조건부 확률에 로그 취한 것을 부가적인 정규화 패널티를 부여하여 최적화한다.

$$logP(Y|X) = \sum_i \quad (g_i log\{P(y_i|Y_{-i}, X)P(s_i)\} + (1 - g_i) log\{P(p(i)|Y_{-i}, X)(1 - P(s_i))\}$$

Y, X는 요약문과 문서 단어 집합, g_i는 지시 함수 (indicator function) 를 의미한다. g_i는 요약문의 i번째 위치의 단어가 OOV 일 경우 0 이다.

테스트 과정에서 모델은 스위치가 켜질 확률인 $P(s_i)$를 기준으로 모든 스텝마다 생성기를 이용할지 포인터를 이용할지 결정한다. 모든 스텝마다 가장 좋은 아웃풋(output)을 만들어 내기 위하여 생성기를 사용할 것인지 포인터를 사용할 것인지에 대한 사후확률에 대해 $argmax$를 이용한다.

포인터 메커니즘은 문서의 어떤 단어를 가리킬 것인지 결정하기 때문에 인코더의 은닉 상태를 사용한다. 은닉 상태는 단어의 전체 문맥에 의존하기 때문에 모델은 타겟 어휘에 없는 단어이더라도 정확하게 가리킬 수 있다. 단어가 소스 어휘에도 없는 경우 포인터 모델은 RNN 에 의해서 인코딩된 'UNK' 토큰의 문맥적 표현을 고려하기 때문에 소스에서 단어의 정확한 위치를 알 수 있다. 단어의 정확한 위치를 알게되면 훈련 과정에서 소스 어휘와 타겟 어휘 모두에 발견되지 않는 단어여도 요약문에 사용될 수 있다.



[그림 2] : 생성기/포인터 스위칭 모델 : 스위치가 'G'를 나타내면, 소프트맥스 계층을 가지고 있는 생성기가 단어를 만들어낸다. 스위치가 'P'를 나타내면 포인터 네트워크가 활성화되어 소스 문서에서 단어를 복사한다. 그림에서 인코더에서 디코더로 향하는 화살표에서 볼 수 있듯이 포인터가 활성화되면 소스에 있는 임베딩이 다음 스텝의 입력으로 들어간다.

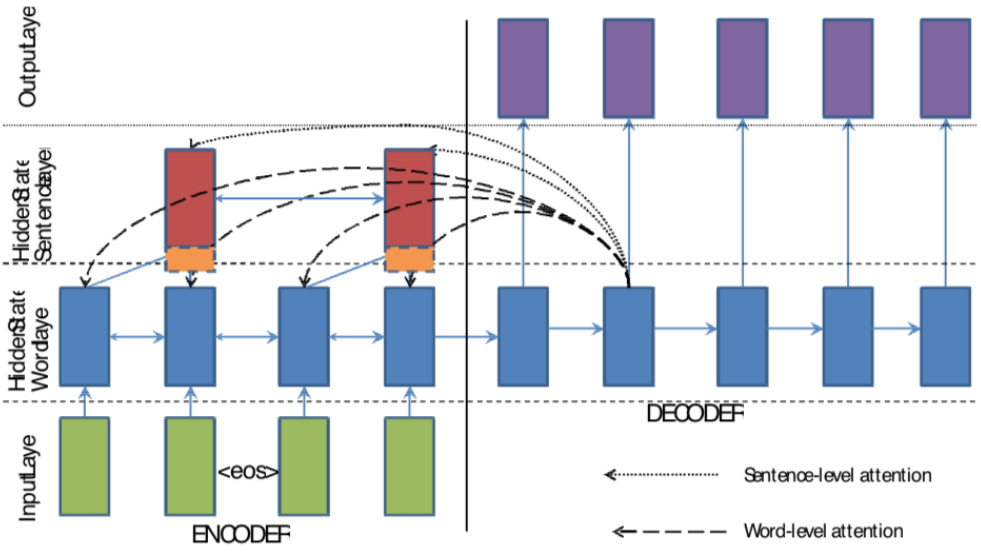
계층적 주의(Hierarchical Attention)를 이용한 계층적 문서 구조 파악

소스 문서가 매우 긴 데이터 셋에 대해서는 문서의 핵심 어휘를 파악하는 것 이외에도 요약문을 만들 때 사용할 수 있는 핵심 문장을 파악하는 것이 중요하다. 이 모델은 소스 부분에서 두 개의 양방향 RNN(단어, 문장)을 사용하여 단어와 문장 두 가지 중요성 레벨의 개념(notion)을 파악하는 것을 목표로 한다. 주의 메커니즘은 두 개의 레벨에 대해서 동시에 동작하는데 단어 레벨 주의(word-level-attention)는 해당하는 문장 레벨 주의 (sentence-level- attention) 를 보고 다시 가중치가 부여되고 다시 정규화된다.

$$P^a(j) = \frac{P_w^a(j)P_s^a(s(j))}{\sum_{k=1}^{N_d} P_w^a(k)P_s^a(s(k))}$$

$P_w^a(j)$는 소스 문서에서 j 번째 위치에서 단어 레벨 주의의 가중치, $s(j)$는 j 번째 위치의 단어가 있는 문장의 ID, $P_s^a(l)$은 소스 문서에서 l 번째 위치에서 문장 레벨 주의의 가중치, N_d는 소스 문서에서 사용되는 단어의 수, $P^a(j)$는 j 번째 단어 위치에서 다시 조정된 주의를 의미한다.

이렇게 조정된 주의 (attention) 는 디코더의 은닉 상태의 입력인 주의 가중치 문맥 벡터 (attention-weighted context vector) 를 계산하기 위해 사용된다. 또한 문서에서 문장의 위치적 중요성을 모델링하기 위해 문장 레벨 RNN 의 은닉 상태에 부가적인 위치 임베딩을 붙여 넣는다. 따라서 이 구조는 핵심 문장을 그 문장 안에 있는 핵심 어휘와 함께 모델링한다.



[그림 3] : 계층적 주의(Hierarchical attention)를 이용한 계층적 인코더 : 파선 화살표로 표현되어 있는 단어 레벨에서의 주의 가중치는 해당하는 문장 레벨 주의 가중치에 의해 다시 조정된다. 문장 레벨 주의 가중치는 점선 화살표로 표현되어 있다. 최상위 계층 RNN 의 아래에 표시되어 있는 파선 사각형은 해당 은닉 상태와 함께 붙여진 문장 레벨의 위치 임베딩을 의미한다.

나. 실행과정 (Experiment)

tensorflow/models/research/textsum [5] 에 위치하는 python 코드와 데이터를 이용해 Textsum 모델을 실행시킬 수 있다. 본 연구에서는 우분투 16.04 LTS 환경에서 python 3.5.2 버전을 이용하였다. python 버전 차이 등의 원인으로 발생하는 오류를 수정해가며 오픈 소스를 이용해 train, evaluation, decode 과정을 거쳤다. Textsum 모델은 Gigaword 데이터셋에 대하여 진행된 연구이므로 bath size 등 모델에서 필요로 하는 상수는 Gigaword 에 가장 적합한 값으로 설정되어있다. 이 모델은 training, eval, decode 3 개의 모드로 실행된다.

Toy dataset

오픈 소스와 함께 공개되어 있는 데이터는 제목과 기사 내용이 담긴 data 파일과 이 파일에서 사용된 어휘인 data 파일로 구성된다. 이 데이터는 textsum 모델 실행을 위한 예시 데이터이다.

이 데이터를 이용하여 훈련과정을 실행하니 [그림 4]와 같은 오류가 발생하였다.

```
limhyesu@limhyesu-desktop:~/models/research/textsumtrain$ bazel-bin/textsum/seq2seq_attention \
>      --mode=train \
>      --article_key=article \
>      --abstract_key=abstract \
>      --data_path=data/data \
>      --vocab_path=data/vocab \
>      --log_root=textsum/log_root \
>      --train_dir=textsum/log_root/train
Exception in thread Thread-8:
Traceback (most recent call last):
  File "/home/limhyesu/anaconda3/lib/python3.5/threading.py", line 914, in _bootstrap_inner
    self.run()
  File "/home/limhyesu/anaconda3/lib/python3.5/threading.py", line 862, in run
    self._target(*self._args, **self._kwargs)
  File "/home/limhyesu/models/research/textsumtrain/textsum/batch_reader.py", line 139, in _FillInputQueue
    data.ToSentences(article, include_token=False)]
  File "/home/limhyesu/models/research/textsumtrain/textsum/data.py", line 215, in ToSentences
    return [s for s in s_gen]
  File "/home/limhyesu/models/research/textsumtrain/textsum/data.py", line 215, in <listcomp>
    return [s for s in s_gen]
  File "/home/limhyesu/models/research/textsumtrain/textsum/data.py", line 189, in SnippetGen
    start_p = text.index(start_tok, cur)
TypeError: a bytes-like object is required, not 'str'
```

[그림 4] : toy dataset 으로 실행한 결과 훈련과정에서 발생한 오류 화면

이 오류는 python 2.x 와 python 3.x 사이의 버전 차이로 인해 발생하는 오류였다. batch_reader.py 의 _GetExFeatureText 함수 정의에서 반환 값에 utf-8 로 디코드하는 코드를 추가하였다.

변경 전	변경 후
def _GetExFeatureText(self, ex, key): """Extract text for a feature from td.Example. Args: ex: tf.Example. key: key of the feature to be extracted. Returns: feature: a feature text extracted. """ return ex.features.feature[key].bytes_list.value[0]	def _GetExFeatureText(self, ex, key): """Extract text for a feature from td.Example. Args: ex: tf.Example. key: key of the feature to be extracted. Returns: feature: a feature text extracted. """ return ex.features.feature[key].bytes_list.value[0].decode('utf-8')

[표 1] : batch_reader.py 의 _GetExFeatureText 함수 정의 부분 수정

[5] https://github.com/tensorflow/models/tree/master/research/textsum

training 모드는 자동으로 멈추지 않기 때문에 수동으로 멈춰야 한다. 이 데이터로 training 을 한 결과, running_avg_loss 가 금방 0.000079 등의 0 에 가까운 값으로 수렴하는 것을 볼 수 있었다. 이는 데이터의 크기가 너무 작아 인스턴스가 부족하여 과적합(overfit)된 것으로 결론을 내렸다.

toy data 에는 train 을 위한 데이터 이외에 다른 데이터가 제공되어 있지 않기 때문에 train 과정에서 사용한 데이터를 그대로 사용하여 eval 모드와 decode 모드를 실행한 결과이다.

모델을 통해 생성된 헤드라인	정확한 헤드라인
output=to to <UNK> <UNK> <UNK> <UNK> to <UNK> .	output=argentine to extradite nazi ss captain to italy .
output=<UNK> <UNK> for <UNK>	output=injury worry for sampras .
output=<UNK> <UNK> <UNK> <UNK> as <UNK> <UNK> .	output=sri lanka closes schools as war escalates .
output=<UNK> <UNK> <UNK> <UNK> <UNK> .	output=crimean parliament adopts new constitution .
output=<UNK> <UNK> <UNK> <UNK> <UNK> <UNK> <UNK> .	output=arafat doubts gaza attack will affect redeployment .
output=<UNK> <UNK> <UNK> <UNK> <UNK> .	output=six voted us athletics honor .

[표 2] : toy dataset 을 이용한 학습 결과

train 과정에서 사용한 데이터와 같은 데이터를 사용하여 모델이 과적합되었음에도 불구하고 첫 번째 열에서 볼 수 있듯이 모델을 통해 생성된 헤드라인은 유의미한 단어보다는 <UNK> 토큰만 등장한다. 이는 toy dataset 의 사이즈가 작고, 어휘가 충분치 않기 때문에 좋지 않은 결과가 나온 것으로 보인다.

CNN dataset

Toy dataset 을 이용한 결과 보다는 의미 있는 결과를 내기 위하여 문서 요약 연구에서 많이 사용되는 공개 데이터인 CNN 데이터를 이용하여 모델을 실행시켰다. CNN 기사를 다운 받은 후 모델에 사용하기 전에 전처리 과정을 거쳐야 한다. See et al. (2017) 에서 CNN 데이터와 DailyMail 데이터를 기준으로 연구하였는데 이들은 라이선스가 필요한 Gigaword 와 달리 공개 데이터이다[6]. CNN 데이터셋은 92570 개의 기사로 구성된다.

데이터를 전처리하는 과정에서는 Stanford CoreNLP 를 이용하여 CNN 데이터를 토큰화 (tokenize)

[6] https://github.com/becxer/cnn-dailymail/ 에서 CNN 데이터를 다운받아 전처리하였다.

하였다. 이후 textsum_data_convert.py[7]를 이용하여 CNN 문서를 이용해 어휘 파일을 만들고 텍스트를 utf-8 로 디코딩된 바이너리 파일로 변환하였다. 어휘 파일은 잘 만들어졌으나 텍스트 파일을 바이너리 파일로 변환하는 과정에서 다음과 같은 오류가 발생하였다.

```
AttributeError: 'str' object has no attribute 'decode'
```

이를 해결하기 위해 textsum_data_convert.py 파일의 _convert_files_to_binary 함수 정의 부분을 수정하였다.

변경 전	`body = document_parts[1].decode('utf8').replace('\n', ' ').replace('\t', ' ')`
	`sentences = sent_tokenize(body)`
	`body = '<d><p>' + ' '.join(['<s>' + sentence + '</s>' for sentence in sentences]) + '</p></d>'`
	`body = body.encode('utf8')`

변경 후	`body = document_parts[1].replace('\n', ' ').replace('\t', ' ')`
	`sentences = sent_tokenize(body)`
	`body = '<d><p>' + ' '.join(['<s>' + sentence + '</s>' for sentence in sentences]) + '</p></d>'`
	`body = body.encode('utf8')`

[표 3] : textsum_data_convert.py 의 _convert_files_to_binary 함수 정의 부분 수정

이후에 또 다른 오류가 발생하였다.

```
TypeError: '<d><p><s>(CNN) -- Ronaldinho rounded on critics who said he was finished after he helped lead Brazi has type str, but expected one of: bytes
```

[그림 5] : CNN dataset 변환 과정 발생 오류

이 또한 textsum_data_convert.py 파일의 _convert_files_to_binary 함수 정의 부분에서 생긴 오류였다. 배포된 CNN 데이터에는 기사의 헤드라인이 정해져있지 않다. 기사의 일부와 헤드라인을 이용해 supervised 학습을 하는 textsum 모델을 이용하기 위해서 textsum_data_convert 에서는 데이터를 title 과 body 로 구분하고 있다. 문서의 첫 문장을 title 로 지정하고 나머지를 body 로 지정한다. 이때 모델에 필요한 데이터 형식을 맞춰 주면서 utf-8 로 인코드하는 과정을 필요로 한다. 위와 같은 오류는 title 을 인코드하는 코드가 빠져 생긴 오류였다. 따라서 아래와 같이 코드를 수정하였다.

변경 전	`title = '<d><p><s>' + document_parts[0] + '</s></p></d>'`
변경 후	`title = '<d><p><s>' + document_parts[0] + '</s></p></d>'` `title = title.encode('utf8')`

[표 4] : textsum_data_convert.py 의 _convert_files_to_binary 함수 정의 부분 수정

textsum_data_convert.py 에서 텍스트 파일에서 사용되는 어휘와 빈도수를 확인하여 어휘 파일을

[7] https://github.com/surmenok/TextSum/blob/master/textsum_data_convert.py 에서 공개한 데이터 변환 코드이다.

만드는 모드인 text_to_vocabulary 를 실행하여 어휘 파일인 vocab 파일이 생성하였다. 텍스트 파일을 바이너리 파일로 변환하는 모드인 text_to_binary 실행하여 전체 데이터셋의 80%, 15%, 5% 비율로 train, validation, test 을 위한 데이터 파일이 각각 생성하였다. 이렇게 생성된 데이터를 이용하여 Textsum 모델을 실행시킬 수 있다.

cnn-train.bin 에는 92570 개의 기사 중 80%인 74056 개의 기사가 있다. 이 데이터를 이용해 48 시간 동안 textsum 모델의 train 모드를 실행한 결과 running_avg_loss 가 1.x 에 수렴하는 결과를 볼 수 있었다.

다음은 13885 개의 기사가 있는 cnn-validation.bin 을 이용해 eval 모드를 실행하고 4628 개의 기사가 있는 cnn-test.bin 을 이용해 decode 모드를 실행한 결과이다. 그러나 모델이 생성한 헤드라인과 정확한 헤드라인을 비교해본 결과 decode 과정에서 test 데이터를 잘 읽고 있지 않은 것과 같은 결과가 나왔다. 입력으로 하고자 하는 파일이 입력으로 잘 들어가지 않는 것으로 보인다.

모델을 통해 생성된 헤드라인	정확한 헤드라인
output=(CNN Student News) -- October 28, 2013	output=(CNN) -- Looking like a tourist can cost you money and pride.
output=Los Angeles (CNN) -- The Los Angeles	output=(CNN) -- While English has long been the de facto language of international business, more multinational companies are now mandating that employees communicate only in English.
output=(CNN Student News) -- October 14, 2013 2013	output=There have been Christians in the Middle East since the time of, well, Christ.
output=(CNN Student News) -- August 14, 2013	output=(CNN) -- When two continents collide, there's inevitably going to be some chaos.
output=(CNN) -- Remember the third thing in <UNK>	output=Creigh Deeds remembers turning his back just before his son attacked him, stabbing the Virginia state senator multiple times.
output=Washington (CNN) -- President Barack Obama said Thursday for State with President Barack Obama -- President Barack Obama appears to <UNK> <UNK>	output=India has an unlikely new public health hero: a giant, anthropomorphic stool that chases people to squat in toilets.

[표 5] : CNN dataset 을 이용한 학습 결과

Ⅳ. 연구 결과

toy dataset 으로 textsum 모델을 통해 학습한 결과 <UNK> 토큰 이 많이 발생하는 등 좋지 않은 결과를 보여주고 있다. 이는 textsum 모델이 Rush et al.(2015) 에 설명되어 있는 Gigaword 에 적합한 상수를 이용하였기 때문에 이 상수가 toy dataset 에는 적합하지 않아 발생한 결과로 판단하였다. 또한 Gigaword 는 400 만 개의 기사가 있는 반면 toy dataset 은 그에 비해 크기가 작고, 어휘 파일 또한 크기가 작아 Gigaword 데이터로 학습한 결과에 비해 좋지 않은 성능을 보이고 있다.

CNN dataset 으로 textsum 모델을 통해 학습한 결과에서는 원하는 데이터에 대해서 요약문을 만들어내고 있지 않았다. 이는 Textsum 모델과 함께 공개된 파일이 아닌 textsum_data_convert.py 를 사용하였기 때문에 이 과정에서 문제가 생겼을 수 있고, 충분하지 못한 training, eval 시간이 원인으로 작용하였을 수 있다. 이러한 문제점은 향후 연구에서 정확한 원인을 찾아내어 해결할 필요가 있다.

Ⅴ. 논 의

Textsum 모델은 뉴스 기사의 일부분을 보고 헤드라인을 만들어 내는 모델이다. 이러한 모델의 특성상 문서의 모든 부분을 읽고 요약문을 생성해야 하는 데이터에 대해서는 좋은 요약문을 만들어 낼 수 없을 것이라 판단된다. 따라서 긴 문서에 대한 요약문을 만들어 낼 수 있는 모델에 대한 연구가 필요하다. 또한 Nallapati et al. (2016) 에 따르면 Textsum 모델은 생성된 요약문은 같은 문장을 반복하거나 요약문 안에서 사용되는 단어를 반복해서 사용하는 한계점을 갖고 있다. See et al.(2017) 에서 볼 수 있듯이 이러한 문제점은 기존 문서에서 이미 살펴본 부분을 확인함으로써 그 문제점을 해결할 수 있다. 또한 'UNK' 토큰이 많이 발생하는 문제점은 디코드 어휘는 훈련 과정에서 정해지지만 요약문을 만들어 낼 때 필요로 하는 단어가 디코더 어휘에 없을 수 있기 때문에 발생한다. 따라서 생성기와 포인터를 번갈아 사용함으로써 이러한 문제를 해결할 수 있다.

본 연구에서는 training 결과를 시각화하는 도구를 따로 사용하지 않았기 때문에 결과를 분석함에 있어서 어려운 점이 있었다. Tensorboard 와 같은 시각화 도구를 사용한다면 결과 분석에서 생기는 어려움을 해결할 수 있을 것이라 판단된다.

VI. 참고문헌

권영대, 김누리, & 이지형. (2017). 문장 수반 관계를 고려한 문서 요약. *정보과학회논문지*, *44*(2), 179-185.

김형석. (2017, 10 월 31 일). 검색일 11 월 13 일, 2018 년, 출처 : https://brunch.co.kr/@kakao-it/139

Khargharia, D., Newar, N., & Baruah, N. (2018). APPLICATIONS OF TEXT SUMMARIZATION. *International Journal of Advanced Research in Computer Science*, *9*(3).

Mehta, P. (2016). From extractive to abstractive summarization: a journey. In *Proceedings of the ACL 2016 Student Research Workshop* (pp. 100-106).

Nallapati, R., Zhou, B., Gulcehre, C., & Xiang, B. (2016). Abstractive text summarization using sequence-to-sequence rnns and beyond. *arXiv preprint arXiv:1602.06023*.

Rush, A. M., Chopra, S., & Weston, J. (2015). A neural attention model for abstractive sentence summarization. *arXiv preprint arXiv:1509.00685*.

See, A., Liu, P. J., & Manning, C. D. (2017). Get to the point: Summarization with pointer-generator networks. *arXiv preprint arXiv:1704.04368*.

Annotated English Gigaword. (n. d.). Retrieved July 5, 2018, from https://catalog.ldc.upenn.edu/LDC2012T21

Domo Releases Sixth Annual "Data never sleeps" Infographic. (2018, June 5). Retrieved November 7, 2018, URL https://www.domo.com/news/press/domo-releases-sixth-annual-data-never-sleeps-infographic

Guilaume, G. (2017, November 8). Seq2Seq with Attention and Beam Search. Retrieved September 27, 2018, URL https://guillaumegenthial.github.io/sequence-to-sequence.html

How to Run Textsum. (2016, September 10). Retrieved September 10, 2018, URL https://hanveiga.github.io/post/Running%20Textsum%20with%20all%20details%20explained/

Peter, L., & Xin, P. (2016, August 24). Text summarization with Tensorflow. Retrieved July 1, 2018, URL https://ai.googleblog.com/2016/08/text-summarization-with-tensorflow.html

Surmenok, P. (2016, October 15). How to run text summarization with tensorflow. Retrieved September 27, 2018, URL http://pavel.surmenok.com/2016/10/15/how-to-run-text-summarization-with -tensorflow/

표절 검사 결과

문서보기 창

Turnitin 독창성 보고서

처리일자: 22-11월-2018 13:28 KST
ID: 1043439087
단어 수: 2818
제출됨: 1

기계학습 기법을 이용한 오픈소스 기반 문서 요약 시스템에 관한 연구 저자: 임혜수

유사성 지표	출처별 유사성	
4%	Internet Sources:	4%
	출판물:	1%
	학생 보고서:	2%

인용 리스트 제외 서지 리스트 제외 사소한 일치 제외 다운로드 프린트

모드: 보고서 빠른보기(클래식)

1% match (출처: 인터넷 2014년 07월 06일)
http://rodatus.net

1% match (출처: 인터넷 2018년 10월 18일)
https://ko.wikipedia.org/wiki/Tf-idf

1% match (출처: 인터넷 2012년 07월 28일)
http://drucillachanofsky5221.professionalliabilityinsurancepros.com

1% match (출처: 인터넷 2014년 01월 28일)
http://www.coconutbreeze.com

<1% match (출처: 인터넷 2014년 01월 17일)
http://www.xdytrans.cn

<1% match (출처: 인터넷 2010년 11월 08일)
http://mephist.egloos.com

<1% match (출처: 인터넷 2018년 09월 12일)
http://lijiancheng0614.github.io