언어 정보를 반영한 문장 점수 측정 기반의 문장 압축

이준범, 김소언, 박성배
경희대학교 컴퓨터공학과
{jblee9410, sekim0211, sbpark71}@khu.ac.kr

Sentence Compression based on Sentence Scoring Reflecting Linguistic Information

Jun-Beom Lee, So-Eon Kim, Seong-Bae Park
Dept. of Computer Science and Engineering, Kyung Hee University

요 약

문장 압축은 원본 문장의 중요한 의미를 보존하는 짧은 길이의 압축 문장을 생성하는 자연어처리 태스크이다. 문장 압축은 사용자가 텍스트로부터 필요한 정보를 빠르게 획득할 수 있도록 도울 수 있어 활발히 연구되고 있지만, 기존 연구들은 사람이 직접 정의한 압축 규칙이 필요하거나, 모델 학습을 위해 대량의 데이터셋이 필요하다는 문제점이 존재한다. 사전 학습된 언어 모델을 통한 perplexity 기반의 문장 점수 측정을 통해 문장을 압축하여 압축 규칙과 모델 학습을 위한 데이터셋이 필요하지 않은 연구 또한 존재하지만, 문장 점수 측정에 문장에 속한 단어들의 의미적 중요도를 반영하지 못하여 중요한 단어가 삭제되는 문제점이 존재한다. 본 논문은 언어 정보 중 품사 정보, 의존관계 정보, 개체명 정보의 중요도를 수치화하여 perplexity 기반의 문장 점수 측정에 반영하는 방법을 제안한다. 또한 제안한 문장 점수 측정 방법을 활용하였을 때 문장 점수 측정 기반 문장 압축 모델의 문장 압축 성능이 향상됨을 확인하였으며, 이를 통해 문장에 속한 단어의 언어 정보를 문장 점수 측정에 반영하는 것이 의미적으로 적절한 압축 문장을 생성하는 데 도움이 될 수 있음을 보였다.

1. 서론

문장 압축은 문장에서 중요하지 않은 정보를 삭제하여 문장의 길이를 축소하는 자연어처리 태스크이다. 문장 압축은 자동 문서요약기의 성능을 향상시키기 위해 사용[1,2]될 뿐 아니라 모바일 환경을 위한 이메일 내용 압축 연구[3]와 영상의 자막 생성을 위한 문장 압축 연구[4]와 같이 제한된 공간에서 사용자에게 정보를 효과적으로 전달하기 위해 사용된다. 최근 웹, 모바일 콘텐츠가 폭발적으로 증가하고 있으며 사용자들은 대부분 모바일 기기를 통해 콘텐츠를 소비하기 때문에, 작은 화면을 통해 사용자가 중요한 정보를 빠르게 획득할 수 있도록 텍스트 정보를 압축해주는 문장 압축의 중요성이 커지고 있다.

문장 압축 연구는 원본 문장에서 삭제하고자 하는 단어를 어떻게 결정할 것이냐에 따라 규칙 기반의 문장 압축 방법과 딥러닝 기반의 문장 압축 방법으로 나누어진다. 규칙 기반의 문장 압축 연구는 트리 트리밍을 통한 문장 압축[5]이 대표적으로, 문장의 파스 트리 정보를 바탕으로 사용자가 정의한 규칙에 따라 단어들을 삭제하여 압축된 문장을 생성한다. 규칙 기반의 문장 압축 방법은 사람이 정의한 규칙을 사용하기 때문에 문법적으로 적절한 압축 문장을 생성할 수 있다는 장점이 있지만, 데이터셋에 따라 새로운 압축 규칙을 정의해야 한다는 문제점이 있다. 딥러닝 기반의 문장 압축 연구는 LSTM 기반의 문장 압축[6]이 대표적으로, LSTM 모델을 사용해 입력 문장의 각 단어가 삭제되어야 할지, 삭제되지 않아야 할지를 이진 분류한 뒤 삭제되지 않아야 한다고 분류된 단어들을 결합하여 압축 문장을 생성한다. LSTM 기반 문장 압축 모델은 사람이 직접 정의한 규칙을 요구하지 않는다는 장점이 있지만, 모델 학습을 위해 많은 데이터를 요구한다는 문제점이 있다.

이러한 문제를 해결하기 위하여 문장 점수 측정 기반 문장 압축 모델인 Deleter[7]가 제안되었다. Deleter는 BERT를 활용한 Perplexity 기반의 문장 점수 측정 방법을 통해 문장에서 불필요한 단어를

삭제하고 최적의 문장 압축 경로를 찾아 문장을 압축한다. Deleter는 학습을 위한 별도의 데이터를 요구하지 않는다는 장점이 있으며 Perplexity 기반의 문장 점수 측정 방법을 사용하기 때문에 문법적으로 오류가 적은 압축 문장을 생성할 수 있다. 하지만 Deleter는 문장을 구성하고 있는 단어들의 의미적 중요도를 문장 점수에 반영하지 못하기 때문에 Deleter를 활용한 문장 압축에서 의미적으로 중요한 단어들이 삭제되기 쉽다.

본 논문에서는 이러한 문제점을 해결하기 위하여 언어 정보 점수를 포함한 Perplexity 기반의 문장 점수 측정 방법을 사용하는 LI-Deleter (Deleter with Linguistic Information) 모델을 제안한다. LI-Deleter는 문장에 속하는 단어들의 품사 정보 점수, 의존관계 정보 점수, 개체명 정보의 중요도를 수치화한 뒤, 세 정보의 중요도의 가중합을 단어의 언어 정보 점수로 사용한다. 가중합을 위한 세 정보의 가중치를 구하기 위하여 단어의 품사 정보 점수, 의존관계 정보 점수, 개체명 정보 점수를 입력으로 하고 단어의 삭제 여부를 출력으로 하는 한 개의 선형 레이어로 구성된 이진 분류 모델을 사용한다. LI-Deleter는 계산된 문장에 속한 단어들의 언어 정보 점수를 Perplexity 기반의 문장 점수 계산에 포함함으로써, 문법적으로 오류가 없으면서 원본 문장의 중요한 의미를 잃어버리지 않은 압축 문장을 생성할 수 있다.

실험을 위해 Google sentence compression dataset[8]을 사용하였으며, 문장 압축 성능 확인을 위해 F-1 Score와 평균 압축률을 측정하였다. 실험 결과 LI-Deleter는 기존 Deleter보다 F-1 Score가 2.44만큼 향상되었으며, 2.46% 낮은 평균 압축률을 보였다. 이를 통해 언어 정보를 반영한 문장 점수를 사용하는 것이 문장 점수 측정 방식의 문장 압축 모델의 성능을 향상시킬 수 있음을 확인하였다.

2. Deleter

2-1 Deleter의 동작

Deleter의 동작은 압축 후보 문장 생성, BERT를 사용한 후보 문장들의 점수 측정, 압축 문장 결정의 세 단계의 반복으로 이루어진다. 압축 후보 문장 생성 단계에서는 입력 문장에서 한 개 이상의 토큰을 삭제한 후보 문장들을 생성하는데, 후보 문장의 점수 측정 단계에서 BERT를 사용하기 때문에 토큰의 단위는 BERT의 학습 단위인 subword이다. 후보 문장 점수 측정 단계에서는 사전 학습된 언어 모델 BERT를 사용해 모든 후보 문장의 Average Perplexity Score (AvgPPL)를 계산한다. AvgPPL은 Perplexity 기반의 문장 점수이기 때문에 값이 낮을수록 문법적으로 적절한 문장임을 나타낸다. 최종적으로 압축 문장 결정 단계에서는 AvgPPL이 가장 낮은 후보 문장을 압축 문장으로 결정하며 종료 조건을 만족할 때까지 세 단계의 압축 단계를 반복한다. 2-2장과 2-3장에서는 각각 문장 점수 계산 방법 및 문장 압축 종료 조건에 대해 자세히 설명한다.

2-2 Average Perplexity Score (AvgPPL)

Deleter는 m개의 토큰으로 구성된 문장 $W=(w_1,w_2,\cdots,w_m)$으로부터 n개의 토큰 $D=(d_1,d_2,\cdots d_n)$이 삭제된 압축 후보 문장 $T=(t_1,t_2,\cdots,t_{m-n})$의 AvgPPL을 수식 1과 같이 계산한다.

$$AvgPPL(T)=\exp\left(\frac{1}{m}\left(-\sum_{i}^{m-n}\log p(T_i|T_{\setminus i})-\sum_{j}^{n}\log p(D_j|D_{\setminus j})\right)\right)$$

수식 1 AvgPPL의 계산식

이때 $T_{\setminus i}$는 압축 후보 문장 T에서 i번째 토큰인 T_i를 제외한 토큰들의 집합을 의미하며, $p(T_i|T_{\setminus i})$는 BERT를 통해 계산한 T_i의 likelihood를 의미한다. 또한 $D_{\setminus j}$는 압축 후보 문장 T를 생성하기 위해 삭제된 토큰들의 집합 D에서 j번째 토큰인 D_j를 제외한 토큰들의 집합을 의미하며, $p(D_j|D_{\setminus j})$는 BERT를 통해 계산한 D_j의 likelihood를 의미한다.

2-3 Deleter의 종료 조건

Deleter는 후보 문장 점수 측정 단계에서 모든 압축 후보 문장이 수식 2의 조건을 만족하지 못한다면 압축 후보 문장을 생성하기 위해 삭제하는 토큰의 개수를 한 개씩 증가시킨다. 만약 삭제하는 토큰의 수가 세 개가 되었음에도 모든 압축 후보 문장이 수식 2의 조건을 만족하지 못한다면, 문장 압축을 종료하며 현재 입력 문장을 최종 압축 문장으로 결정한다.

$$\frac{AvgPPL_{candidate}}{AvgPPL_{input}}>1+\lambda\log(L_{root})$$

수식 2 삭제 토큰 개수 증가 조건

수식 2에서 $AvgPPL_{input}$은 입력 문장의 AvgPPL을 의미하며, $AvgPPL_{candidate}$는 압축 후보 문장의 AvgPPL을 의미한다. 또한 λ는 토큰 삭제될 때마다

AvgPPL이 증가하는 비율을 조정하기 위한 하이퍼 파라미터이며, L_{root}는 최초 입력 문장의 토큰 개수를 의미한다.

3. LI-Deleter

Deleter는 사전 학습된 언어 모델 BERT를 사용하여 계산한 압축 후보 문장들의 점수를 바탕으로 문장을 압축한다. 이때 문장에 속한 단어들의 언어 정보가 문장의 점수에 반영되지 않기 때문에 중요한 언어 정보가 삭제된 압축 문장을 생성하는 경우가 발생한다. LI-Deleter는 Deleter의 압축 후보 문장 점수 측정 단계에서 압축 후보 문장이 되기 위해 입력 문장에서 삭제된 토큰의 의미적 중요도를 해당 후보 문장의 점수에 더하여 준다. 이를 통해 중요한 토큰이 삭제된 압축 후보 문장이 압축 문장으로 선택되는 빈도를 줄일 수 있다.

LI-Deleter의 후보 문장 점수 측정 단계에서 m개의 토큰으로 구성된 문장 $W=(w_1,w_2,\cdots,w_m)$으로부터 t번째 토큰 w_t이 삭제된 압축 후보 문장 $W_{\setminus t}=(w_1,w_2,\cdots,w_{t-1},w_{t+1},\cdots,w_m)$이 존재할 때, $W_{\setminus t}$의 문장 점수 $Score(W_{\setminus t})$는 수식 3과 같이 계산된다.

$$Score(W_{\setminus t})=AvgPPL(W_{\setminus t})+LI(w_t)$$

수식 3 LI-Deleter의 문장 점수 계산식

이때 $LI(w_t)$는 W의 t번째 토큰 w_t의 언어 정보 점수(Linguistic Information Score)를 의미하며 수식 4와 같이 계산된다.

$$LI(w_t)=\alpha PI(w_t)+\beta DI(w_t)+\gamma EI(w_t)$$

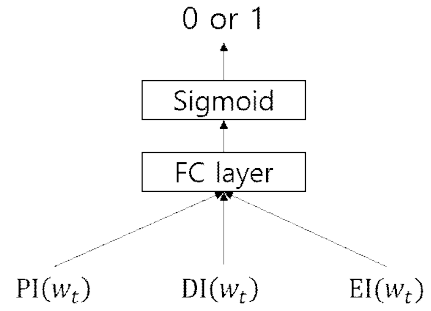
$$PI(w_t)=\frac{count_{target}(Part\ of\ Speech\ Tag(w_t))}{count_{source}(Part\ of\ Speech\ Tag(w_t))}$$

$$DI(w_t)=\frac{count_{target}(Dependency\ Tag(w_t))}{count_{source}(Dependency\ Tag(w_t))}$$

$$EI(w_t)=\frac{count_{target}(Entity\ Tag(w_t))}{count_{source}(Entity\ Tag(w_t))}$$

수식 4 언어 정보 점수의 계산식

이때 $PI(w_t)$, $DI(w_t)$, $EI(w_t)$는 각각 w_t의 품사 정보 점수(Part Of Speech Information Score), 의존관계 정보 점수(Dependency Information Score), 개체명 정보 점수(Entity Information Score)를 의미하며 α,β,γ는 세 점수의 가중치를 의미한다. 또한 $count_{source}$와 $count_{target}$는 각각 전체 데이터셋의 원



(그림 1) 토큰 삭제 여부 이진 분류 모델

본 문장에 각 태그가 등장한 횟수, 전체 데이터셋의 압축 문장에 각 태그가 등장한 횟수를 의미한다.

수식 4의 품사 정보 점수, 의존관계 정보 점수, 개체명 정보 점수의 가중치인 α,β,γ를 결정하기 위하여 토큰의 품사 정보 점수, 의존관계 정보 점수, 개체명 정보 점수를 입력으로 받아 토큰의 삭제 여부를 출력하는 그림 1과 같은 이진 분류 모델을 사용하였다. 이진 분류 모델에는 세 정보의 점수를 입력으로 받아 가중합을 출력하는 선형 레이어가 존재하며, 학습된 선형 레이어의 가중치를 세 언어 정보 점수의 가중치 α,β,γ로 사용하였다.

4. 실험

실험을 위해 Google Sentence Compression Dataset을 사용하였다. Google Sentence Compression Dataset은 20만 쌍의 학습 데이터와 1만 쌍의 테스트 데이터로 구성되어있으며, 20만 쌍의 학습 데이터를 19만 쌍, 5천 쌍, 5천 쌍으로 나누어 각각 토큰 삭제 여부 이진 분류 모델의 train, valid, test를 위해 사용하였다. 또한 Deleter의 실험 세팅과 동일하게 1만 쌍의 테스트 데이터 중 첫 1,000쌍의 데이터를 사용하여 LI-Deleter의 문장 압축 성능을 측정하였다. 또한 Google Sentence Compression Dataset에서 제공하는 텍스트의 품사 태그, 의존관계 태그, 개체명 태그 정보를 본 논문의 언어 정보로 사용하였다.

토큰 삭제 여부 이진 분류 모델은 66.9%의 정확도를 보였으며, 학습된 토큰 삭제 여부 이진 분류 모델의 품사 정보, 의존관계 정보, 개체명 정보의 가중치를 사용한 LI-Deleter와 비교 모델인 Deleter의 문장 압축 성능은 표 1과 같다.

이때 평균 압축률은 모델이 압축한 문장의 토큰

<표 1> Deleter와 LI-Deleter의 문장 압축 성능 비교

모델	F-1 Score	평균 압축률(%)
Deleter	38.06	40.05
LI-Deleter	**40.50**	**38.59**

개수를 원본 문장의 토큰 개수로 나눈 압축률의 평균으로, 모델이 문장의 길이를 얼마나 많이 압축시켰는가를 확인할 수 있는 지표이다.

실험 결과 LI-Deleter는 Deleter보다 2.44만큼 향상된 F-1 Score를 보였으며, 2.46%만큼 낮은 평균 압축률을 보였다. 이를 통해 언어 정보의 중요도가 반영된 문장 점수를 활용한 문장 압축 모델인 LI-Deleter가 Perplexity 기반의 문장 점수만을 활용하는 Deleter보다 더 짧으면서도 정답에 포함된 중요한 단어들을 잘 포함하는 압축 문장을 생성함을 확인할 수 있다.

5. 관련 연구

문장 압축 태스크를 수행하기 위해 사람이 정의한 규칙을 사용하는 연구와 딥러닝 기반의 연구가 존재했다. 트리 트리밍 기반 문장 압축 방법[5]은 문장의 파스 트리 정보로부터 사람이 정의한 압축 규칙에 따라 문장을 압축하여 문법적으로 오류가 적은 압축 문장을 생성하였으나, 데이터셋에 따라 새로운 규칙을 새로 정의해야한다는 문제점이 존재했다. 딥러닝 기반의 연구는 LSTM 기반의 이진 분류 모델을 통해 문장에서 삭제할 단어를 분류하는 방법[6]이 좋은 성능을 보였지만, 모델 학습을 위해 대량의 데이터셋을 필요로 한다는 문제점이 존재했다.

Deleter[7]는 사전 학습된 언어 모델 BERT을 사용한 문장 점수 측정을 기반으로 순차적으로 문장의 단어를 삭제하는 모델로, 사람이 정의한 압축 규칙과 모델 학습이 필요하지 않다는 장점을 보였지만 문장 압축에 문장에 속한 단어들의 언어 정보를 활용하지 않기 때문에 문장에서 중요한 정보가 삭제되는 문제점이 존재했다.

LI-Deleter는 문장을 구성하고 있는 단어들의 언어 정보 중 품사 정보, 의존관계 정보, 개체명 정보의 중요도를 수치화한 뒤 Deleter의 문장 점수 측정에 반영하여 문장의 중요한 정보가 삭제되지 않는 압축 문장을 생성할 수 있다.

6. 결론

본 논문은 문장 점수 측정 기반의 문장 압축 모델인 Deleter의 문장 점수 계산에 문장을 구성하고 있는 토큰들의 언어적 중요도를 반영하는 방법을 제안한다. 언어적 중요도를 수치화하기 위하여 품사 정보 점수, 의존관계 정보 점수, 개체명 정보 점수를 가중합한 언어 정보 점수를 사용하였으며, 실험을 통해 언어 정보 점수를 반영한 문장 점수 측정 방법을 사용하였을 때 Deleter의 문장 압축 성능이 향상됨을 확인하였다.

사사

이 논문은 2021년도 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원(No. 2020R1A4A1018607)과 정보통신기획평가원의 지원(No.2013-0-00109,WiseKB: 빅데이터 이해 기반 자가학습형 지식베이스 및 추론 기술 개발)을 받아 수행된 연구임

참고문헌

[1] K. Knight and M. Daniel, “Statistics-based Summarization-Step One: Sentence Compression,” *In Proceedings of the 17th National Conference on Artificial Intelligence and 12th Conference on Innovative Applications of Artificial Intelligence*, Austin, 2000, 703-710.

[2] H. Jing, "Sentence Reduction for Automatic Text Summarization," *In Proceedings of the 6th Applied Natural Language Processing Conference*, Seattle 2000, 310-315.

[3] S. Corston-Oliver, "Text Compaction for Display on Very Small Screens," *In Proceedings of the North American Chapter of the Association for Computational Linguistics workshop on Automatic Summarization*, Pittsburgh, 2001, 89-98.

[4] V. Vandeghinste, and K. Sang, “Using a Parallel Transcript/Subtitle Corpus for Sentence Compression,” *In Proceedings of the 4th International Conference on Language Resources and Evaluation*, Lisbon, 2004, 231-234.

[5] T. Berg-Kirkpatrick, D. Gillick, and D. Klein, “Jointly learning to extract and compress,” *In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologie*s, Portland, 2011, 481-490.

[6] K. Filippova, E. Alfonseca, C. Colmenares, L. Kaiser, O. Vinyals, “Sentence Compression by Deletion with LSTMs,” *In Proceedings of the Conference on Empirical Methods in Natural Language Processing*, Lisbon, 2015, 360-368.

[7] T. Niu, C. Xiong, and R. Socher. “Deleter: Leveraging BERT to Perform Unsupervised Successive Text Compression,” *arXiv preprint arXiv:1909.03223*, 2019.

[8] K. Filippova, and Y. Altun, “Overcoming the lack of parallel data in sentence compression,“ *In Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, Seattle, 2013, 1481-1491.