테이블 구조 정보를 활용한 헤더 텍스트 생성

정해민, 심묘섭, 민경구, 최주영, 박민준, 최정규

LG AI 연구원
{haemin.jung, myoseop.sim, kyungkoo.min, jooyoung.choi, minjun.park, stanleyjk.choi}@lgresearch.ai

Header Text Generation based on Structural Information of Table

Haemin Jung, Myoseop Sim, Kyungkoo Min, Jooyoung Choi, Minjun Park, Stanley Jungkyu Choi
LG AI Research

요 약

테이블 데이터는 일반적으로 헤더와 데이터로 구성되며, 헤더는 데이터의 구조와 내용을 이해하는데 중요한 역할을 한다. 하지만 웹 스크래핑 등을 통해 얻은 데이터와 같이 다양한 상황에서 헤더 정보가 누락될 수 있다. 수동으로 헤더를 생성하는 것은 시간이 많이 걸리고 비효율적이기 때문에, 본 논문에서는 자동으로 헤더를 생성하는 태스크를 정의하고 이를 해결하기 위한 모델을 제안한다. 이 모델은 BART를 기반으로 각 열을 구성하는 텍스트와 열 간의 관계를 분석하여 헤더 텍스트를 생성한다. 이 과정을 통해 테이블 데이터의 구성요소 간의 관계에 대해 이해하고, 테이블 데이터의 헤더를 생성하여 다양한 애플리케이션에서의 활용할 수 있다. 실험을 통해 그 성능을 평가한 결과, 테이블 구조 정보를 종합적으로 활용하는 것이 더 높은 성능을 보임을 확인하였다.

주제어: Text Generation, Table Header Generation, BART

1. 서론

현대 사회에서 데이터는 매우 다양한 형태로 구성된다. 특히 테이블 형태의 데이터는 그 구조화된 형태와 높은 표현력 때문에 다양한 분석과 모델링 분야에서 널리 사용된다. 테이블 형태의 데이터는 그 종류가 매우 다양하지만, 일반적으로 헤더(header)와 데이터 행(row)로 구성된다고 볼 수 있다. 헤더는 각 열이 가진 의미를 명시하므로 테이블 데이터를 이해하고 분석하는 데 중요한 역할을 한다.

그러나, 데이터가 다양한 원천으로부터 추출되거나 생성되는 과정에서 헤더 정보가 결여된 데이터셋이 생길 수 있고, 이는 데이터 처리 및 분석의 복잡도를 증가시킨다. 대표적인 예로는 웹 스크래핑(web scraping)을 통해 수집한 데이터가 있다. 헤더가 결여된 테이블에서 각 열이 가진 정보를 수동으로 식별하고 정의하는 작업은 테이블을 구성하는 열의 수가 많을수록 많은 시간이 소요되고, 효율성이 떨어진다.

이러한 문제를 해결하기 위해, 본 연구는 테이블 데이터의 헤더를 자동으로 생성하는 태스크를 정의하고, 이를 해결하기 위한 방법론을 제시한다. 제안하는 모델은 테이블을 구성하는 각 열의 데이터 특성을 분석하여 헤더 텍스트를 창출한다. 이 과정에서, 개별 열의 데이터 특성 뿐만 아니라, 테이블 내 다른 열들과의 상호 관계성 또한 고려한다.

본 연구에서 소개하는 모델은 다수의 모듈로 구성되어 있으며, 헤더 예측 과정에서 두 가지 추가적인 정보를 적극 활용한다. 첫째, 다른 열들에서 예측된 헤더 정보를 활용하여 특정 열의 헤더를 더욱 정밀하게 예측하며, 둘째, 테이블 전체가 대표하고 있는 개념 또는 주제에 대한 정보를 활용한다. 이러한 접근 방식은 테이블 데이터의 맥락을 더욱 정확히 파악하고, 더욱 적절한 헤더 텍스트를 생성하는 데 도움을 준다.

예를 들어, 그림 1에서와 같은 테이블에서 첫 번째 열의 헤더를 생성한다고 하자. 해당 열의 원소들만 본다면, 헤더가 될 수 있는 대안은 '지역', '구', '자치구' 등 무수히 다양하다. 그런데, 표에서 두 번째 열은 '정당'을 나타내고 있다. 따라서, 첫 번째 열의 헤더는 이를 고려하여 '선거구' 등과 같이 보다 데이터의 문맥이 고려된 헤더로 특정되는 것이 타당하다.

본 연구에서는 한국어 테이블 데이터셋을 기반으로 실험을 수행하였으며, 제안하는 모델의 세 가지 파생 형태들의 성능을 종합적으로 비교함으로써 테이블의 구조적 정보를 활용하는 것이 헤더 생성 작업에서 어떠한 긍정적인 효과를 가져오는지 깊이 있게 분석하였다. 이러한 분석은 헤더 텍스트 생성을 넘어서, 테이블 데이터의 구조적 정보를 활용하는 방식에 대한 통찰을 제공한다.

그림 1. 테이블의 헤더 텍스트 생성 작업 예시

2. 관련 연구

2.1 시퀀스-투-시퀀스

본 연구에서는 테이블을 구성하는 텍스트들을 이용하여 열의 헤더 텍스트를 생성하는 태스크를 다룬다. 이는 시퀀스-투-시퀀스(Sequence to Sequence) 모델[1]로 대표되는 언어 모델(language model)로 해결할 수 있다. 언어 모델은 확률에 기반을 두고 토큰(token)을 연속적으로 예측할 수 있는데, 시퀀스-투-시퀀스 모델은 입력 시퀀스에 대한 인코딩(encoding)과 디코딩(decoding)을 통해 태스크 목적에 맞는 출력 시퀀스로 변형하는 모델이다. 전통적으로는 기계 번역을 비롯해, 최근에는 관련된 각종 모델의 발전으로 자연어 요약이나 생성, 질의응답 등 다양한 목적으로 활용되고 있다.

최근 자연어 처리 분야에서는 특히 트랜스포머(Transformer) 모델[2]을 기반으로 하는 시퀀스-투-시퀀스 모델들이 널리 활용되고 있다. 이들은 일반적인 텍스트에 대해 학습을 마친 사전 학습 언어 모델(pre-trained language model) 형태로 배포되고 있다. 사전 학습 언어 모델들은 수행하고자 하는 태스크에 맞게 데이터를 추가적으로 학습시키는 미세 조정(fine-tuning)을 거쳐 사용된다.

2.2 테이블 데이터에 대한 연구

테이블 데이터는 대표적인 '구조화된(structured) 데이터'이다. 테이블 데이터에 담긴 정보를 활용하여 질의응답을 하는 분야 [3,4], 테이블을 텍스트 문서의 형태 중 하나로 보고 기계 독해를 하는 분야 [5] 등에서 주로 연구가 이루어지고 있으며, 테이블 데이터를 처리하는 여러 방식에 대해 연구가 진행되고 있다. 다만, 헤더 텍스트 생성 태스크는 그 필요성에도 불구하고 구체적으로 정의되어 있지 않다. 여러 텍스트들의 내용을 종합하는 태스크의 특성을 보면, 일종의 요약 태스크로도 볼 수도 있다.

3. 제안하는 방법

본 연구에서는 테이블의 구조적 특성을 반영한 헤더 텍스트 생성 모델을 제안한다.

3.1 문제 정의

방법론을 설명하기에 앞서, 본 연구에서 다루고자 하는 '테이블'을 명확하게 한정하고자 한다. 테이블은 그림 2과 같이 매우 다양한 형태로 구성할 수 있으나, 본 연구에서는 열마다 헤더 텍스트를 갖는 테이블(그림 2(a))만으로 대상을 한정하였다. 이는 관계형 데이터베이스를 구성하는 테이블의 형태로서, 가장 일반적인 모양으로 간주할 수 있다. 행마다 헤더 텍스트를 갖는 경우(그림 2(b)), 열과 행이 모두 헤더로서 기능하는 경우(그림 2(c))는 고려하지 않았다. 또한, 셀의 병합이 있는 경우(그림 2(d)) 및 다중 헤더를 갖는 경우(그림 2(e)) 역시 복잡도에 따른 분석 난이도를 고려하여 본 연구의 대상에서 제외하였다. 또한, 테이블을 구성하는 항목들은 모두 문자열 텍스트로, 수치형 텍스트가 있는 열은 제외한다.

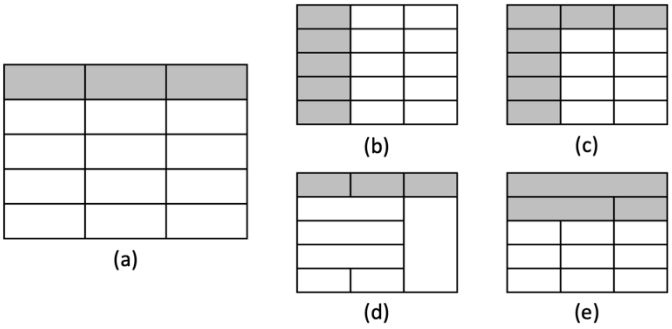

그림 2. 테이블 형태

다음으로, '테이블 헤더 텍스트 생성' 태스크를 구체적으로 정의하고자 한다. 헤더와 $m \times n$개 데이터로 구성된 테이블 T는 n개 열의 집합으로 나타낼 수 있으며, i번째 열 C_i는 $\{h^i, c_1^i, \dots, c_j^i, \dots, c_n^i\}$으로 나타낼 수 있다. 이 때 h^i는 i번째 열의 헤더 텍스트이며, c_j^i은 i번째 열을 구성하는 j번째 행의 텍스트이다. 따라서 테이블 T는 다음과 같이 표현할 수 있다.

$$T\{C_i \in C | C \text{ is a set of columns}\}$$
$$= C_1 \cup C_2 \cup \dots \cup C_n$$
$$= \{h^1, \dots, h^n, c_1^1, \dots, c_m^n\}$$

본 연구에서 수행하고자 하는 '테이블 헤더 텍스트 생성' 태스크는 주어진 테이블 T에 대하여, 그 원소 c_j^i들의 정보를 바탕으로 합리적인 헤더 텍스트 $\{\hat{h}^1, \dots, \hat{h}^n\}$를 생성하는 것이다.

3.2 모델 구조

헤더 텍스트 생성 모델의 전체적인 구조는 그림 3과 같다. 좌측에 녹색으로 표시된 부분은 테이블의 정답 헤더 부분이고, 회색으로 표시된 부분이 테이블을 구성하는 텍스트 데이터이다. 이 모델은 헤더를 생성하는 과정에서 세 가지를 고려한다. 즉, (1) 열을 구성하는 데이터와 헤더의 관계, (2) 다른 헤더들과의 상호 관계, 그리고 (3) 테이블의 전체적인 맥락을 모두 고려한다. 먼저, 테이블의 각 열 데이터는 트랜스포머 모델을 거쳐 노란색으로 표시된 헤더를 생성해낸다. 이 헤더는 최종 예측 모듈에서 테이블 전체 텍스트의 정보를 담고 있는 임베딩과 결합됨으로써 테이블 전체 및 다른 열과의 관계성을 고려한 헤더로 최종 변환된다. 학습 과정에서, 최종적으로 생성된 헤더가 정답 헤더와 가까워지도록 학습된다.

이제 모델을 구성하는 세부 모듈들에 대해 설명한다.

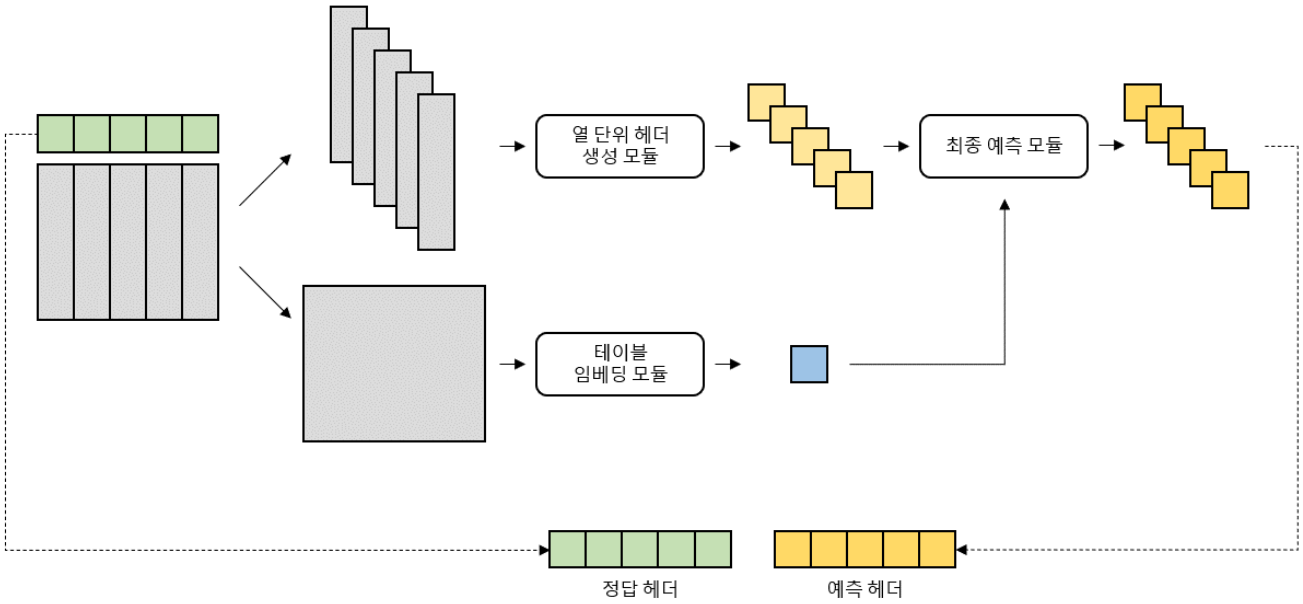

그림 3. 모델 구조와 학습 과정

3.2.1 열 단위 헤더 생성 모듈

열 단위 헤더 생성 모듈은 각 열을 구성하는 텍스트들을 바탕으로 헤더를 예측한다. 즉, 각 i번째 열에 대하여 c_1^i부터 c_n^i까지의 텍스트를 인코딩한 다음, 다시 디코딩하여 헤더 텍스트 $\tilde{h}^i$를 만든다.

이 모듈은 그 자체로 가장 단순한 형태의 헤더 텍스트 생성 모델이다. 따라서, 사전 학습된 시퀀스-투-시퀀스 모델을 가져와 각 열의 텍스트를 바탕으로 헤더를 생성하도록 미세 조정한 다음, 실험 단계에서 베이스라인으로 활용할 수 있다.

3.2.2 테이블 임베딩 모듈

열 단위 헤더 생성 모듈에서 열 단위로 데이터 정보가 반영되어 헤더가 생성되었으나, 이것 만으로 헤더를 생성하기에는 정보가 부족하다. 즉, 올바른 예측을 위해서는 테이블 전체의 맥락을 반영하는 것 역시 필요하다. 이를 위해 전체 테이블의 텍스트를 펼쳐 테이블 인코딩 모듈에 입력으로 집어넣는다. 이를 통해 테이블 컨텍스트 벡터 v_t를 얻을 수 있다.

3.2.3 최종 예측 모듈

최종 예측 모듈은 열 단위 헤더 생성 모델과 마찬가지로 시퀀스-투-시퀀스 모델이지만, 헤더 간의 관계성을 반영하게 된다. 열 단위 헤더 생성 모듈의 결과로 생성한 헤더들의 집합을 $\tilde{H}$라고 했을 때, 최종 예측 모듈은 $\tilde{H}$와 테이블 컨텍스트 벡터 v_t를 종합적으로 고려하여 최종 헤더 집합인 $\hat{H}$을 만들어낸다. 보다 구체적으로는 인코더 부분의 결과에 추가 정보로 v_t를 반영한 다음 디코더에 입력으로 넣어 학습시킨다.

4. 실험

4.1 데이터셋

실험을 위한 데이터셋으로는 KorWikiTable 데이터[6]를 사용하였다. 이 데이터셋은 한국어 위키피디아에 있는 테이블 정보를 담고 있으며, 1,196,306건의 테이블 중, 행 방향으로 헤더가 붙어있는 정보 상자(info box) 형태의 데이터는 제외하고 열 방향으로 헤더가 붙어 있는 데이터만 선별하여 591,415건의 데이터를 얻었다.

데이터셋에서 일부 테이블은 행마다 셀의 개수가 다르다. 이런 경우는 온전한 열을 추출하기 어려우므로 제외하였다. 이렇게 선별된 테이블의 개수는 262,515개이다.

또한, 테이블의 열 중 숫자로만 구성된 것들은 제외하였다. 대표적인 경우로는 정수형과 실수형(백분율) 데이터 열이 있다. 이 경우 숫자만으로는 의미 있는 헤더를 예측하는 것은 불가능하기 때문에 제외하였다. 또한, 열 데이터의 빈칸이 20% 이상인 경우는 제외하였다.

그 외에 각 테이블의 하단에는 통계치를 제시하기 위한 셀들이 포함되어 있는 경우가 많았다. (예: 합계를 나타내기 위한 행 등) 이 부분은 노이즈로 작용할 수 있기 때문에 끝에서 3줄을 단순히 제외시키는 규칙 기반 전처리를 적용하였다.

최종적으로 427,954개의 열 데이터를 얻었으며, 이 데이터를 8:1:1 비율로 학습, 검증, 테스트 세트로 분할하여 실험을 진행하였다.

4.2 실험 설계

열 단위 헤더 생성 모듈과 최종 예측 모듈에는 사전 학습된 언어모델로 KoBART[7]를 사용하였다. KoBART는 40GB 이상의 한국어 텍스트를 BART[8] 모델 구조에 사전

학습시킨 모델이며, 124백만 개의 파라미터를 가진다. 사전 학습된 KoBART에 수집한 열 데이터 중 학습 세트를 학습시켜 모델을 미세 조정하였으며, 학습 알고리즘으로는 Adam[9]을 사용하였다. 배치 사이즈는 32, GPU 자원은 A100 40GB를 사용하였다.

실험을 통해 다음의 3가지 케이스에 대해 성능을 비교하였다.

1. koBART: 열 데이터만을 사용했을 때. 열 단위 헤더 생성 모듈의 결과를 사용한다.
2. koBART+koBART: 다른 헤더 정보와의 관계성을 복합적으로 사용했을 때.
3. koBART+koBART^{T}: 다른 헤더와의 관계를 고려하면서 테이블을 임베딩한 컨텍스트 벡터를 추가하여 테이블의 모든 정보를 복합적으로 사용했을 때.

이 실험을 통해 테이블의 구성 요소들을 활용하는 것이 어떤 의미를 가지는지에 대해 알아보고자 하였다. 성능 평가를 위한 지표로는 정확도를 사용하였다.

4.3 실험 결과

표 1은 성능 평가 결과를 나타낸다. 단순히 열 단위로 예측했을 때보다, 다른 열의 이름을 어텐션으로 반영한 경우 더 높은 성능을 보였다. 또한, 전체 테이블의 임베딩 정보를 추가해줌으로써 추가적인 성능 향상을 보였다. 이는 일차적으로 테이블 데이터가 어떤 내용에 대해 구성되었는지에 대한 정보를 추가적으로 알려준다면, 헤더 텍스트를 더 잘 예측할 수 있다는 것을 의미한다. 추가적으로는 테이블의 구조 정보, 즉, 각각의 열과, 다른 열과의 관계성, 전체 테이블이 나타내고자 하는 개념과의 관계성을 종합적으로 고려하여야 언어 모델이 테이블 데이터에 대한 이해도가 높아질 수 있음을 시사한다.

표 1. 성능 평가 결과

	Accuracy
koBART	0.859
koBART+koBART	0.868
koBART+koBART^{T}	0.870

5. 결론

본 연구에서는 테이블을 구성하는 각 열의 헤더를 예측하는 태스크를 제시하고, 이 태스크를 수행하기 위해 테이블이 가진 구조 정보를 어떤 식으로 활용할 것인지 탐색하였다. 각 열을 구성하는 원소들의 텍스트를 넣고, 헤더를 출력하도록 딥러닝 모델을 학습시킬 때, 다른 열과의 관계성과 테이블 전체의 맥락을 함께 고려하였다. 최종적으로는 성능 비교를 통해 테이블의 정보를 활용하는 방식에 대한 인사이트를 제공한다는 점에서 의의가 있다. 다만, 헤더 텍스트 생성 성능을 극대화하기 위한 모델 구조에 대해서는 추가적인 연구가 필요하다.

또한, 본 연구에서는 헤더 텍스트 생성 태스크에 대해서만 다뤘지만, 이 문제는 테이블 완성 태스크로 확장될 여지가 있다. 예를 들어, 테이블의 정보를 바탕으로 그 타이틀을 생성할 수도 있을 것이며, 특정 셀이 비어 있을 때 그 안에 들어갈 내용을 테이블의 다른 내용을 바탕으로 생성할 수도 있을 것이다.

또한, 하나의 열에 대해 다수의 대안이 존재하는 만큼, 단순히 정확도만을 이용한 평가에는 한계가 있다. 데이터에 존재하는 정답은 아니어도, 합리적인 답이라면 성능 지표를 계산할 때 고려해야 할 필요성이 있으므로, 이에 대한 추가적인 실험 역시 필요하다.

참고문헌

[1] Sutskever, Ilya, Oriol Vinyals, and Quoc V. Le. "Sequence to sequence learning with neural networks." Advances in neural information processing systems 27 (2014).

[2] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, “Attention is all you need,” Advances in Neural Information Processing Systems, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., Vol. 30, 2017.

[3] F. Zhu, et al. “TAT-QA: A question answering benchmark on a hybrid of tabular and textual content in finance,” Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics 2021, pp.3277-3287, 2021

[4] Tom B. Brown et al. "Language models are few-shot learners", arXiv preprint arXiv:2005.14165, 2020.

[5] W. Chen, et al. "Open question answering over tables and text," arXiv preprint arXiv:2010.10439, 2020.

[6] C. Jun, et al., “Korean-specific dataset for table question answering,” The 13th edition International Conference on Language Resources and Evaluation 2022, pp.6114-6120. 2022.

[7] https://github.com/SKT-AI/KoBART

[8] LEWIS, Mike, et al. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. arXiv preprint arXiv:1910.13461, 2019.

[9] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” ICLR (Poster), 2015. Available: http://arxiv.org/abs/1412.6980