KoQuality: 한국어 언어 모델을 위한 고품질 명령어 데이터 큐레이션

나요한°[1*], 김다혜[2*], 채동규[1]
[1]한양대학교 컴퓨터소프트웨어학과
[2]한양대학교 인공지능학과
nayohan@hanyang.ac.kr, ekgp7812@gmail.com, dongkyu@hanyang.ac.kr

KoQuality: Curation of High-quality Instruction Data for Korean Language Models

Yohan Na°[1*], Dahye Kim[2*], Dong-Kyu Chae[1]
[1]Dept. of Computer Science, Hanyang University
[2] Dept. of Artificial Intelligence, Hanyang University

요 약

최근 생성형 언어모델에 명령어 튜닝을 적용하여 사람의 명령을 잘 이해하고, 대답의 성능을 향상시키는 연구가 활발히 수행되고 있으며, 이 과정에서 다양한 명령어 튜닝 데이터셋이 등장하고 있다. 하지만 많은 데이터셋들 중에서 어떤 것을 선택해서 활용하지가 불분명하기 때문에, 현존하는 연구들에서는 단순히 데이터셋을 모두 활용하는 방식으로 명령어 튜닝이 진행되고 있다. 하지만 최근 연구들에서 고품질의 적은 데이터셋으로도 명령어 튜닝을 하기에 충분하다는 결과들이 보고되고 있는 만큼, 많은 명령어 데이터셋에서 고품질의 명령어를 선별할 필요성이 커지고 있다. 이에 따라 본 논문에서는 한국어 데이터셋에서도 명령어 튜닝 데이터셋의 품질을 향상시키기 위해, 기존의 데이터셋들에서 데이터를 큐레이션하여 확보된 적은 양의 고품질의 명령어 데이터셋인 KoQuality를 제안한다. 또한 KoQuality를 활용하여 한국어 언어모델에 명령어 튜닝을 진행하였으며, 이를 통해 자연어 이해 성능을 높일 수 있음을 보인다. 특히 제로샷 상황에서 KoBEST 벤치마크에서 기존의 모델들보다 높은 성능 향상을 보였다.

주제어: 명령어 튜닝 데이터셋, 데이터 큐레이션, 한국어 언어모델

1. 서론

생성형 언어모델이 등장한 이후로 모델의 크기에 따라 자연어 이해 능력이 상승하는 결과들이 발표되었다. 이에 따라 GPT3, PaLM과 같이 거대 언어모델에 대한 연구가 활발하게 수행되어 왔다. 하지만 모델의 크기가 커질수록 이를 활용하기 위해 필요한 컴퓨팅 자원이 상당하며, 이를 구축하는 데 큰 비용이 소요된다. 이에 따라 LLAMA와 같이 비교적 작은 크기의 모델을 많은 양의 데이터로 학습시켜 기존의 거대 언어모델과 비슷한 수준의 성능을 보인 연구도 각광을 받았다.

특히 거대 언어모델들은 미세조정 (fine-tuning) 없이도 few-shot 상황에서 높은 성능을 보인다. 심지어 zero-shot 상황에서도 사용자의 질문에 대해 유연하게 답변하는 결과를 볼 수 있다. 즉, 하나의 task에 대해서 문제를 풀던 기존의 언어모델과 달리, 처음보는 문장에 관한 질문에 대한 답변도 그럴 듯하게 할 수 있을 정도로 거대 언어모델의 자연어 이해 수준이 높아지고 있다.

최근, 거대 언어모델들에 대해 사람들이 주는 질문과 같이 여러 입력에 대한 이해도를 향상시키기 위한 명령어 튜닝 방법론 [1] [2]들이 제안되고 있다. 이러한 방법론들을 기반으로 적은 학습으로도 언어모델의 자연어 이해 성능 및 답변 능력의 향상을 이루기위한 시도가 지속되고 있으며, 이를 위한 많은 명령어 튜닝 데이터셋이 제안되었다 [2] [3] [4]. 이러한 결과로 인해 다양한 명령어를 포함하는 데이터셋들이 등장하게 되었다.

하지만 데이터셋의 양이 많아진 만큼, 수집된 데이터셋에서 중복되는 명령어들이 발생하게 되었다. 이로 인해 다양한 데이터셋을 하나로 병합하여 학습하게 될 때 중복 및 품질이 좋지 않은 데이터가 포함될 가능성이 높아진다. 기존의 task들은 이러한 문제를 큰 배치사이즈로 해결하였으나, 컴퓨팅 자원이 한정적인 환경에서는 활용되기 어려운 문제가 있다.

본 논문에서는 명령어 데이터셋에 대한 효율적인 학습을 위해, 기존의 명령어 데이터셋을 통합하고 그 중 고품질의 명령어 데이터셋을 추출하는 큐레이션 방법을 제안한다. 데이터셋을 큐레이션하기 위해서 길이 기반으로 문장 임베딩들을 클러스터링하며, 각 클러스터 그룹을 기반으로 데이터셋을 제작하고자 한다. 또한 생성형 언어모델의 성능 지표 중 하나인 Perplexity를 활용하여 샘플링을 진행하였다. 이렇게 큐레이션된 데이터셋을 KoQuality[1][2]로 명명하였으며, 이를 활용한 결과 기존의 데이터셋의 1% 크기만으로도 한국어 자연어 이해 task인 KoBEST [5]에서 높은 성능을 달성할 수 있음을 확인하였다.

2. 관련 연구

2.1 한국어 언어모델

한국어 생성형 언어모델에는 SKT의 KoGPT2, Kakaobrain에서 개발한 KoGPT등이 있으며, Eleuther AI의 그룹에서 만든 한국어 특화 생성형 언어모델인 Polyglot-Ko [6]도 있다.

*공동제1저자

[1]https://github.com/nayohan/KoQuality
[2]https://huggingface.co/datasets/DILAB-HYU/KoQuality

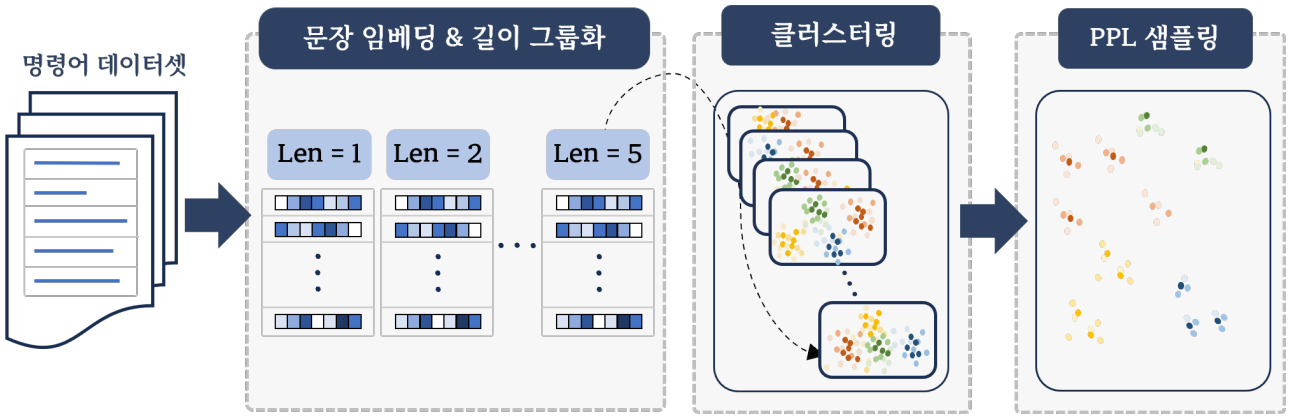

그림 1. 제안하는 KoQuality 구축 프로세스.

Polyglot-Ko는 다양한 분야에서 수집된 1.2TB의 한국어 텍스트 데이터를 활용하여 학습된 언어모델이다. 모델의 크기는 각각 1.3B, 3.8B, 5.8B, 12.8B의 크기로 구성되어 있으며, 주요 데이터 구성은 블로그 포스트, 뉴스, 모두의 말뭉치, 특허 데이터셋, QA 데이터셋으로 구성되어 있다.

2.2 한국어 명령어 데이터셋

KoAlpaca는 Stanford Alpaca [1]에서 제안한 LLAMA에 Alpaca 명령어 데이터셋을 학습시킨 것을 모방해, 같은 방식으로 제작되었다. KoAlpaca v1.0은 Instruction과 Input을 DeepL API로 번역을 진행하였고, 답변은 OpenAI ChatGPT API를 활용하여 생성되었다. 추가 생성한 KoAlpaca V1.1은 네이버 지식인 베스트 질문을 크롤링한 데이터를 질문과 채택된 답변을 기반으로 데이터셋을 생성하였다. 이를 활용하여 Polyglot-Ko에 명령어 튜닝을 진행하여 KoAlpaca-Polyglot-5.8B와 모델을 만들어 공개했다. OIG-small-chip2-ko 데이터셋은 The Open Instruction Generalist(OIG)라는 오픈소스 데이터 제작 프로젝트를 기반으로 만들어졌다. 43M개의 명령어를 포함하고 있는 거대한 데이터셋 중 MLM을 활용하여 비교적 고품질의 명령어 데이터셋을 추출해 OIG-small-chip2를 구글 번역 API로 번역하여 생성된 데이터셋이다. KULLMv2 [7]은 기존의 데이터셋 중 Alpaca, Vicuna, Databricks-Dolly 데이터셋을 병합하여 구성하였으며, DeepL API을 이용하여 번역하였다.

2.3 명령어 데이터셋 큐레이션

명령어 튜닝을 위해 생성된 대량의 데이터셋에서 데이터를 큐레이션하고자 하는 논문들이 최근 등장하고 있다. Instruction Mining은 데이터셋을 큐레이션하기 위한 지표들에 대해 탐색하고, 데이터 품질과 지표 간의 관계를 조사하는 실험을 통해 고품질의 데이터를 선택할 수 있는 것을 보였다. 또한, 데이터셋을 큐레이션을 통해 적은 수의 데이터셋으로도 좋은 명령어 튜닝 성능을 내는 방법에 대한 논문들이 등장하고 있다 [8, 9].

3. 제안하는 방법

본 논문에서는 거대 언어모델의 학습 데이터셋인 명령어 데이터의 품질을 분석하고, 지표 기반 계층적 분류 체계를 가진 데이터 큐레이션 방법론을 제안한다. 그림1은 본 논문에서 제안하는 데이터 큐레이션 방법의 흐름도이다. 데이터셋을 명령어의 길이 별로 그룹화하고, 각 그룹 내의 문장을 KoSimCSE를 활용하여 임베딩한다. 각 길이 그룹 내 임베딩을 K-means Clustering을 통해 의미가 유사한 그룹으로 분류하고, length와 perplexity가 고려된 그룹에서 perplexity를 기준으로 최종적인 샘플링을 진행한다. 제안하는 방법론을 통해 명령어 데이터셋의 다양성을 확보하면서도 높은 품질의 데이터를 선정하는 것을 목표로 한다. 이를 위해 명령어 데이터셋 큐레이션 기준을 문장 길이, K-means 클러스터링, Perplexity로 선정하였고, 각각의 지표에 대한 분석을 진행하였다.

3.1 문장 길이 기반 그룹화

문장의 길이는 텍스트에 대한 분석 및 이해에 중요한 역할을 한다. 문장의 길이는 정보양과 정보의 상세성, 가독성 및 문장 작성자의 목적을 구분할 수 있는 가장 단순한 지표이다. 따라서 명령어 관점에서 보면 문장의 길이에 따라서 지시하고자 하는 사용자의 의도가 달라질 수 있다.

문장의 길이를 기반으로 간단히 나눠보면, 길이 10 이하의 짧은 문장은 “다음”, “계속”, “코드에서 제안”, “다른 예제 제공” 등 주로 이전의 명령에 대해서 추가적으로 명령하는 문장이거나, “리만 가설이란?”과 같은 간결한 키워드에 대한 질의 형태의 문장인 경우가 대다수이다. 이와 달리 길이 10-50의 문장은 “소설 위대한 개츠비를 쓴 사람은 누구입니까?” 와 같은 문장 구성을 가지고 있으며, 주로 우리가 일상생활에서 흔히 사용하는 질문의 형태를 띄고 있다. 길이 50 이상의 문장은 “주어진 진술이 왜 참인지 몇 마디로 설명하세요. 비디오 게임을 사용하면 아이들이 기술과 전략을 개발하는 데 도움이 될 수 있습니다.”와 같은 형태로 명령어와 함께 추가적인 설명이 포

함된 문장들이 대다수를 이루고 있다. 이와 같이 문장 길이에 따라서 명령어의 특성이 달라지고 이는 데이터를 선정하는데 큰 영향을 미치므로, 길이가 고려된 상황에서의 데이터 선택이 필수적이라고 할 수 있다. 따라서 제안하는 방법론은 문장의 길이 별로 데이터셋을 그룹화한 후, 같은 그룹 내에서 클러스터링을 수행한다. 이러한 계층적 분류를 통해 데이터셋의 의미적 다양성을 확보할 수 있을 것으로 기대한다.

3.2 문장 임베딩 및 클러스터링

최근 양질의 문장 임베딩 생성을 위해 대조학습 (contrastive learning) 이 큰 주목을 받고 있다. 대표적인 대조학습 기반 문장 임베딩 프레임워크로는 SimCSE [10]가 있다. SimCSE는 대조적 학습을 활용하여 문장 간의 유사성을 학습한 모델로, 비슷한 문장은 비슷한 임베딩 공간에 매핑되고, 다른 문장은 다른 공간에 매핑되도록 학습하는 프레임워크이다. 본 논문에서는 SimCSE 프레임워크를 한국어 데이터셋으로 학습시킨 KoSimCSE를 활용하여 문장 임베딩을 진행하였다. 앞서 기술하였듯이 문장 임베딩 생성 시 문장의 길이 그룹별로 클러스터링하는 것이 매우 중요하다. 특히 최근 대조학습이 길이에 대해 취약하다는 연구 결과가 있으며 [11], 비슷한 의미의 문장이어도 길이가 다른 문장은 다른 임베딩을 가질 수 있다는 결과가 발표되었다. 따라서 문장 길이를 비율 별로 범위를 나눠서 클러스터링을 하는 것이 필수적이다. 이에 생성된 임베딩을 길이 그룹 별로 나눈 뒤에, K-means 클러스터링 알고리즘을 기반으로 벡터의 유사도에 따른 군집을 생성하였다.

3.3 Perplexity 기반 샘플링

끝으로 적은 수의 양질의 명령어 데이터셋을 구축하기 위한 샘플링 방법에 대해서 논의한다. 언어모델의 경우, 다음 단어를 생성하기 위해 확률 기반 예측을 수행하기 때문에 Perplexity가 중요한 지표로 작용한다. Perplexity란 이전 단어로 다음 단어를 예측할 때 몇 개의 단어 후보를 고려하는지를 의미하며, 단어 w로 구성된 문장의 Perplexity는 다음과 같은 수식으로 계산할 수 있다.

$$\text{Perplexity} = \sqrt[N]{\frac{1}{\prod_{i=1}^{N} P(w_i \mid w_1, w_2, w_3, ..., w_{i-1})}} \quad (1)$$

위 식에서 분모는 문장에 대한 생성 확률을 의미하고, 전체 수식의 의미는 생성 확률의 역수를 단어의 수인 N으로 정규화한 것이다. Perplexity는 텍스트 생성 언어 모델의 성능 평가 지표로 사용되며, Perplexity가 낮을수록 해당 정답 샘플에 대한 발생 확률이 높은 것을 의미한다. 이를 다르게 해석하면, Perplexity가 낮은 문장은 적은 개수의 단어 후보를 고려하기 때문에 생성하기 쉬운 문장으로, Perplexity가 높은 문장은 반대로 어려운 문장이라고 볼 수 있다.

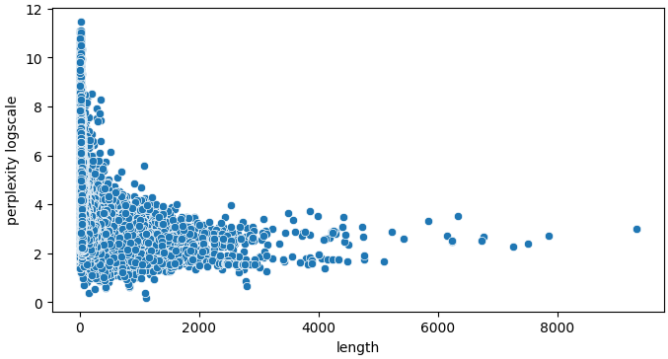

그림 2. 문장 길이와 Perplexity와의 관계 비교.

이러한 특성을 활용하여서, 본 논문에서는 Perplexity (PPL) 를 기반으로 한 샘플링 전략을 제안하며 이를 $PPL_{Sampling}$ 으로 명명한다. 우선 클러스터 내의 문장들을 Perplexity순으로 정렬하여, 일정 비율 만큼 추출한다. 즉, 언어모델이 다음 단어를 예측하기 쉬운 명령어부터 어려운 명령어까지 고르게 데이터셋에 포함한다. 또한 비교를 위한 단순한 베이스라인 샘플링 방법으로서 PPL_{High} 를 추가로 설계하였다. 이는 단순히 Perplexity가 높은 문장들을 샘플링하는 것으로, 모델이 예측하기 어려운 문장들을 선별하는 의미가 있다.

한편, Perplexity는 문장의 길이의 영향을 쉽게 받는다는 연구 결과가 있다 [12]. 그림 2 에서도 볼 수 있듯이 문장이 길수록 Perplexity는 낮아지는 경향성이 있다. 이러한 경향성을 배제하기 위해서, 앞서 언급한 두 가지 샘플링 방법 모두 사전에 구성된 문장 길이 별 그룹 내에서 Perplexity를 기준으로 샘플링을 수행한다. 앞서 나온 그림 1은 이와 같이 문장의 길이, K-means 클러스터링, Perplexity를 반영한 샘플링 방식의 흐름도를 나타낸다.

4. 실험 결과

4.1 데이터셋

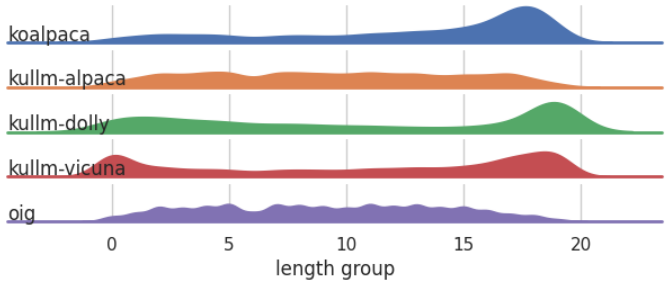

그림 3. 명령어 데이터셋 별 길이 분포.

소량의 고품질 명령어 데이터셋을 제작하기 위해, 본 연구에서는 2.2절에서 소개한 KoAlpaca, KULLM, OIG 세 가지 데이터셋들을 활용하였다. 각각은 서로 다른 길이 분포를 가지고 있으며, 웹 크롤링, 번역기, 오픈소스 제작 프로젝트 등을

통해 생성된 데이터셋이다. KoAlpaca 데이터셋은 지식인 베스트 질문을 크롤링하여 가장 고품질의 데이터셋을 가지고 있다. 그림 3은 한국어 명령어 데이터셋 별 길이 분포를 나타낸 것이다. KoAlpaca 데이터셋은 명령어의 길이가 긴 데이터의 비율이 높고, KULLM 데이터셋은 긴 명령어 뿐만 아니라 길이가 짧은 명령어도 다수 포함이 되어있다. OIG 데이터셋은 전반적으로 고르게 분포된 것을 확인할 수 있다. 이렇듯 각 데이터셋 마다 특성이 다르기에, 길이 그룹마다의 특성을 고르게 반영할 수 있도록 문장 길이 기반 큐레이션 방법론을 설계하였다.

4.2 실험 세팅

데이터셋 큐레이션의 지표인 문장길이, 클러스터링, Perplexity 간의 관계를 파악하여, 각 요소가 미치는 영향을 분석해보고자 했다. 길이 그룹별 클러스터링을 진행하기 위해, 길이 그룹의 개수를 $L : \{1, 5, 10, 20\}$중 하나로, 클러스터링 개수를 $K : \{10, 20, 50, 100\}$ 중 하나로 지정하였다. PPL 방법론은 $PPL_{High}, PPL_{Sampling}$ 을 비교하며, 샘플링 비율은 $N : \{0.1, 0.05, 0.01\}$ 중 하나로 설정하여 데이터셋을 생성하였다. 아래의 그림 4는 데이터셋 3개에 포함된 모든 문장들의 길이 분포를 시각화하였으며 (왼쪽), 문장 길이 그룹 5개로 나누었을 경우를 보여주고 있다 (오른쪽).

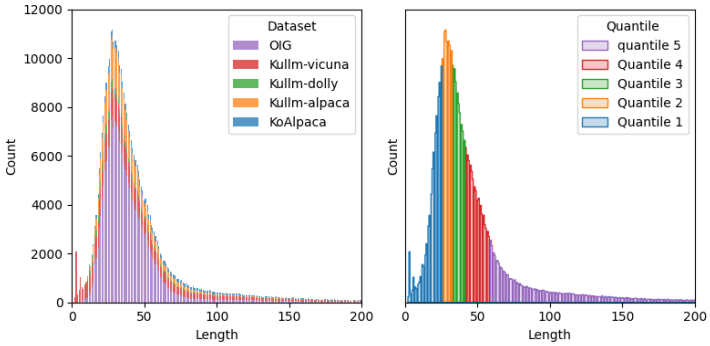

그림 4. 명령어 데이터셋을 통합하여 산출한 문장 길이 분포.

표 1. 각 클러스터 별 문장의 예시.

Len	C	문장 예시
Q1	K0	GMO는 먹어도 안전한가요? 프렌치 토스트를 만드는 단계는 무엇입니까?
	K1	VPN(가상 사설망)은 어떻게 설정합니까? PDF 파일을 어떻게 편집할 수 있습니까?
Q2	K0	쿼른 다진 돼지고기와 프로티엔 30g으로 만든 타코 레시피를 알려주세요. 비건 음식을 만드는 데 사용할 수 있는 일반적인 재료 목록을 생성합니다.
	K1	숫자 벡터 'x = [5, 4, 3]'이 주어졌을 때 합계를 계산합니다. 동일한 스크립트를 제공하되 xlsx 라이브러리를 다른 것으로 대체하세요.

데이터셋 생성을 위해 KoSimCSE를 활용하여 임베딩을 생성하였고, PPL계산을 위해 Polyglot-Ko-1.3B모델을 활용하였다. 생성한 명령어 데이터셋을 활용하여 Polyglot-Ko의 5.8B 모델에 학습시킨 후, KoBEST 벤치마크 중 COPA, BoolQ, Hellaswag 데이터셋에서의 few-shot 성능을 측정하였다. 모든 실험은 2에폭, 배치사이즈 4, Gradient Accumulation 4, 학습률 5e-5로 NVIDIA A100 80GB 4기를 사용하여 진행되었다.

4.3 실험 결과

표 2. $PPL_{Sampling}$ 방법의 평균 few-shot 성능 수치.

Length	L=5	L=10	L=20	Avg.
K=10	59.23	60.18	60.28	59.90
K=20	60.71	60.53	59.20	60.15
K=50	60.65	60.51	60.43	60.53
K=100	**60.88**	60.59	60.78	**60.75**
Avg.	60.37	**60.45**	60.17	

우선 $PPL_{Sampling}$ 샘플링 방법으로 생성한 데이터셋에 대한 실험 결과는 표 2 에서 확인할 수 있다. 표에서 각 셀의 의미는 KoBEST 벤치마크 데이터셋들에서 측정한 few-shot 성능들의 평균치를 의미한다. 각 few-shot 실험은 k가 0, 1, 2, 5, 10일 경우 총 5개의 시나리오 (예를 들면, k가 0일 경우 zero-shot을 의미함) 에서 모두 측정해보았다 (아래의 그림 5는 각 시나리오에서의 결과를 보여준다). 실험 결과 길이 그룹의 개수 (L)가 10일 때의 평균 few-shot accuracy가 60.45으로 가장 높았고, 클러스터 개수에 대해서는 $K = 100$일 때가 60.75로 가장 높았다. 개별적으로 보았을 때는 $L = 5$, $K = 100$일 때 가장 높은 성능을 보였다.

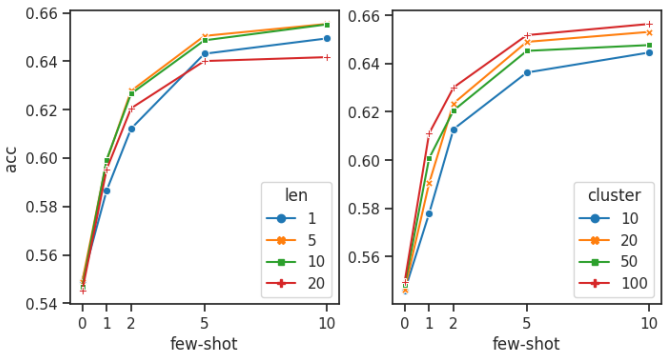

그림 5. 5가지 few-shot 시나리오에서의 결과.

Length	L=1	L=5	L=10	L=20	Avg.
Random	58.23	59.98	60.57	60.21	59.75
PPL_{High}	59.73	59.03	59.00	58.41	59.04
$PPL_{Sampling}$	59.73	59.15	60.24	60.98	60.02

표 3. $K = 1$, $N = 0.01$ 일때 길이 그룹 (L) 별 성능비교.

다음으로 몇 가지의 데이터셋 샘플링 비율 (N) 및 클러스터의 수 (K) 에 따른 각 샘플링 방안들의 결과를 확인해보았다.

Cluster	K=10	K=20	K=50	K=100	Avg.
Random	59.96	60.95	60.27	58.48	59.90
PPL_{High}	58.27	58.48	59.78	58.91	58.86
$PPL_{Sampling}$	59.34	60.22	60.94	60.12	60.15

표 4. $L = 1$, $N = 0.01$ 일때 클러스터링 개수 (K) 별 성능비교

Sampling	N=0.1	N=0.05	N=0.01	Avg.
$PPL_{Sampling}$	59.40	60.86	60.88	60.38

표 5. $L = 5$, $K = 100$ 일때 샘플링 비율 (N) 별 성능비교

표 3은 $K = 1$, $N = 0.01$ 일때 각 샘플링 방법의 성능을 측정한 결과이다. 실험 결과 길이 그룹의 개수가 늘어날수록 성능이 올라가는 경향을 보였으며, $PPL_{Sampling}$이 PPL_{High}보다 더 나은 성능을 보였다. 표 4는 $L = 1$, $N = 0.01$ 일 때 클러스터 개수에 따른 성능 변화를 측정한 결과이다. 클러스터의 개수가 50일 때 가장 좋은 성능을 보였으며, 표 3와 마찬가지로 $PPL_{Sampling}$을 적용했을 때의 성능이 가장 높았다. 표 5를 보면 N=0.01인 작지만 고품질인 데이터셋에서 평균적으로 더 높은 성능을 보였다.

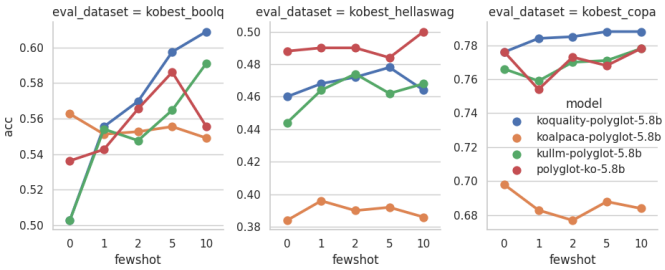

그림 6. 명령어 튜닝된 언어모델의 KoBEST 벤치마크 결과.

마지막으로 그림 6은 명령어 튜닝 데이터셋들로 각각 학습한 Polyglot-Ko-5.8B 모델의 few-shot 예측 성능을 나타낸다. 본 연구에서 제안하는 KoQuality-Polyglot-5.8b 모델[3]의 경우 $L = 5$, $K = 100$, $N = 0.01$ 의 하이퍼 파라미터를 활용하여 학습되었다. 대체적으로 다른 데이터셋에 비해 규모가 매우 작음에도 불구하고 좋은 성능을 보이고 있다.

5. 분석

본 장에서는 앞선 4장에서 보고한 실험결과들에 이어서, 본 연구에서 제안하는 데이터 큐레이션 방법에 대한 깊이있는 실험적 분석을 수행하였다.

5.1 클러스터링 유무 분석

그림 7은 클러스터 유무에 따른 few-shot 성능을 나타낸 결과이다. 클러스터 개수가 1일 때와 100일 때의 성능향상을 비

[3]https://huggingface.co/DILAB-HYU/KoQuality-Polyglot-5.8b

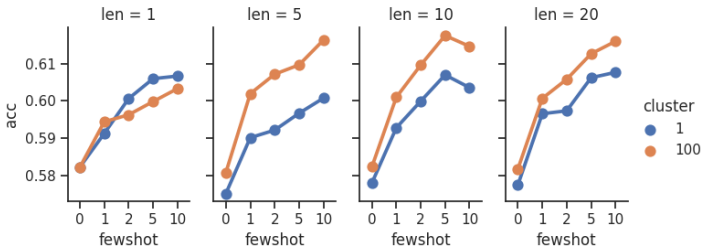

그림 7. 클러스터 유무에 따른 길이 그룹 개수 (L) 별 성능.

교해보면, 클러스터의 개수가 100일 때 few-shot 성능의 향상폭이 큰 것을 확인할 수 있다. 반대로 클러스터링을 하지 않았을 때에는 길이 그룹 개수가 늘어날 때 유의미한 성능 향상을 보이지 않았다.

5.2 길이 그룹 유무 분석

그림 8. 길이 그룹 유무에 따른 few-shot 성능.

다음으로 그림 8은 길이그룹 유무에 따른 few-shot 성능을 나타낸 결과이다. 길이 그룹이 없을 때보다 길이 그룹의 개수가 10일때 더 높은 성능 향상을 보였다. 길이에 대한 그룹화를 한 후에 클러스터링을 진행하는 것이 KoSimCSE모델 임베딩이 가지는 길이 취약성을 완화하면서, 클러스터링 진행시 유의미한 성능 향상을 가져오는 것으로 유추해볼 수 있다.

5.3 PPL 샘플링 방법 분석

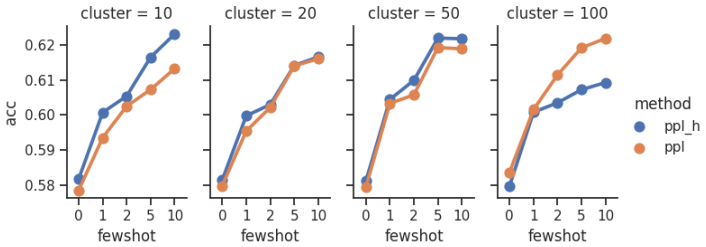

그림 9. PPL 샘플링 방법과 클러스터링 간의 상관관계.

그림 9와 그림 10은 각각 다른 PPL 샘플링 방법을 적용했을 때의 few-shot 성능을 비교한 결과이다. 먼저 그림 9에서 클러스터의 개수를 다르게 지정한 결과, PPL_{High} 는 $PPL_{Sampling}$과 비교하여 성능 하락을 보였다. 그림 10은 PPL 방법의 길이별 성능을 나타낸 결과이다. 길이 별 그룹화를 진행하지 않은 len=1의 경우, PPL_{High} 가 상대적으로 낮은 성능을 보였다.

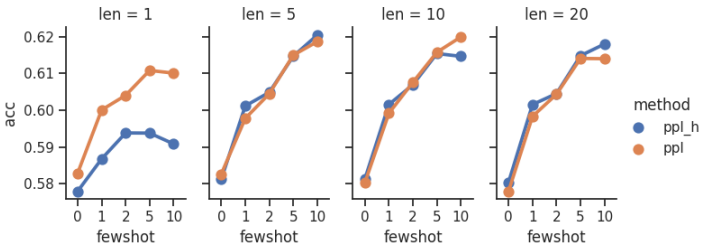

그림 10. PPL 샘플링 방법과 길이 그룹 간의 상관관계.

표 6. PPL 샘플링 방법 별 문장 예시.

Method	문장 예시	PPL
$PPL_{Sampling}$	녹차를 마시면 어떤 이점이 있습니까?	21.12
	당뇨병의 경고 신호는 무엇입니까?	23.91
	포도당의 분자식을 결정합니다.	83.58
PPL_{High}	6가지 요리 허브 나열	11768.52
	입력이 팔린드롬인지 여부 확인 레이스카	5131.08
	식사 계획 시스템 설계	2443.3

성능 하락의 원인을 분석해봤을 때, 길이 별 그룹화를 진행하지 않고 PPL_{High} 방법으로 샘플링을 했을 때 PPL이 매우 높은 문장들이 다수 샘플링 되는 것을 확인할 수 있었다. 특히 표 6의 문장 예시에서 확인할 수 있듯이 PPL이 2,000 이상인 명령어들은 의미적으로 부자연스러운 문장인 경우가 있으며, 영어 데이터셋을 한국어로 번역하는 과정에서 의미가 변형되는 경우도 존재하였다. 이로 인해 길이 그룹 개수가 1일 때의 PPL_{High}는 상대적으로 낮은 성능을 보인 것을 확인할 수 있었다.

6. 결론

본 논문은 한국어 언어모델의 자연어 이해 성능을 높이고자 소량의 고품질 명령어 데이터셋을 구축하는 방법을 제안하였고, 이를 통해 만들어진 데이터셋인 KoQuality를 제안하였다. 명령어 튜닝에 영향을 주는 지표를 분석하여 Length, Clustering, Perplexity를 큐레이션의 기준으로 선정하였다. 제안된 방법론은 기존의 한국어 언어모델과 명령어 튜닝된 한국어 모델들의 KoBEST 벤치마크에 대해서 few-shot 환경에서 높은 성능 향상을 보여주었다. 명령어 튜닝 데이터셋은 언어모델의 최종 학습을 위해 필수적으로 사용되는 만큼, 고품질의 데이터셋을 활용하는 것이 실제 언어모델의 성능을 높일 수 있는 것으로 분석되었다. 본 연구 결과가 앞으로 나올 한국어 거대 언어모델의 성능 향상에 기여할 수 있기를 기대한다.

감사의 글

이 논문은 2023년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원(No.2020-0-01373,인공지능대학원 지원(한양대학교))을 받아 수행되었습니다.

참고문헌

[1] R. Taori et al., "Stanford alpaca: An instruction-following llama model," https://github.com/tatsu-lab/stanford_alpaca, 2023.

[2] Y. Wang et al., "Self-instruct: Aligning language models with self-generated instructions," Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics, pp. 13 484–13 508, 2023.

[3] Y. Wang et al., "Super-naturalinstructions: Generalization via declarative instructions on 1600+ nlp tasks," Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, pp. 5085–5109, 2022.

[4] M. Conover et al. (2023) Free dolly: Introducing the world's first truly open instruction-tuned llm. [Online]. Available: https://www.databricks.com/blog/2023/04/12/dolly-first-open-commercially-viable-instruction-tuned-llm

[5] M. Jang et al., "KoBEST: Korean balanced evaluation of significant tasks," Proceedings of the 29th International Conference on Computational Linguistics, pp. 3697–3708, Oct. 2022.

[6] H. Ko et al., "A technical report for polyglot-ko: Open-source large-scale korean language models," 2023.

[7] N. . A. Lab and H.-I. A. research, "Kullm: Korea university large language model project," https://github.com/nlpai-lab/kullm, 2023.

[8] D. Yin et al., "Dynosaur: A dynamic growth paradigm for instruction-tuning data curation," arXiv preprint arXiv:2305.14327, 2023.

[9] H. Chen et al., "Maybe only 0.5% data is needed: A preliminary exploration of low training data instruction tuning," arXiv preprint arXiv:2305.09246, 2023.

[10] T. Gao et al., "SimCSE: Simple contrastive learning of sentence embeddings," Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pp. 6894–6910, Nov. 2021.

[11] C. Xiao et al., "Can text encoders be deceived by length attack?" 2023. [Online]. Available: https://openreview.net/forum?id=KPHbtTtCDw

[12] Y. Wang et al., "Perplexity from plm is unreliable for evaluating text quality," arXiv preprint arXiv:2210.05892, 2022.