Learning to Filter Context for Retrieval-Augmented Generation

Zhiruo Wang[♠] **Jun Araki**[♦] **Zhengbao Jiang**[♠]
Md Rizwan Parvez[♦] **Graham Neubig**[♠]
[♠]Carnegie Mellon University [♦]Bosch Research
{zhiruow,zhengbaj,gneubig}@cs.cmu.edu

Venue : arXiv preprint 2023

발제자 : 이다현 (hyundai@soongsil.ac.kr)
HUMANE Lab

2024-07-15

Introduction

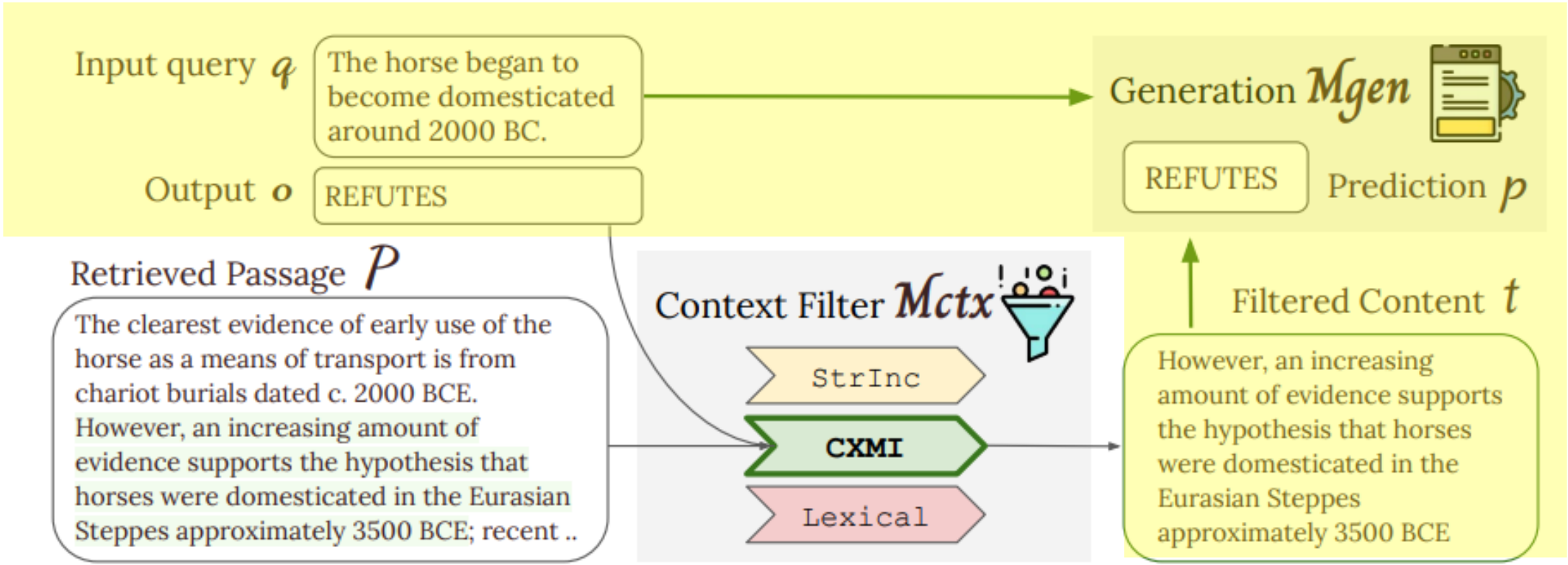
- 모델에 내재된 Knowledge만을 이용하는 한계 → Retrieval-augmented 접근 방식
- 외부 Knowledge(e.g. Wikipedia)를 탐색해 가져옴
  - Retriever에 의해 반환된 Top-k개 Passage들을 Generator에게 무분별하게 입력으로 제공
- Retriever의 불완전성
  - 종종 Query와 무관하거나 오히려 정답 생성에 방해가 되는 정보를 Generator에게 제공
  - 정답과 관련된 Passage를 잘 Retrieve 해왔을지라도, 해당 Passage 내에는 정답에 방해가 되는 문장들이 존재할 수 있음

Introduction

- 이상 : 모델이 정확한 출력을 생성하기 위해 정답을 정확히 지원하는 콘텐츠에 기반해야 함
- 현실 : 완벽한 Retriever가 없기에 Retriever 단독으로는 해결이 어려움
- 선행연구
  - Reranking 과정 추가
    - 2차 필터링: Initial Retrieval결과 Top-N개 → Reranker를 이용해 Top-K개로 추림
  - 생성 모델이 도움이 필요할 때만 Retrieve
  - Evidential Passage만 선택
    - Answer Generation과 Evidentiality Prediction을 동시에 학습
- 그러나 이는 상당한 인간 주석 작업을 필요로함
- 또한 관련 Passage에서도 방해가 되는 콘텐츠는 여전히 발생할 수 있음

Introduction

- **Fil**ter **Co**ntext
- 목표
  - Retrieved Passage 중에서도 Output 생성에 필요한 문장을 골라내는 것

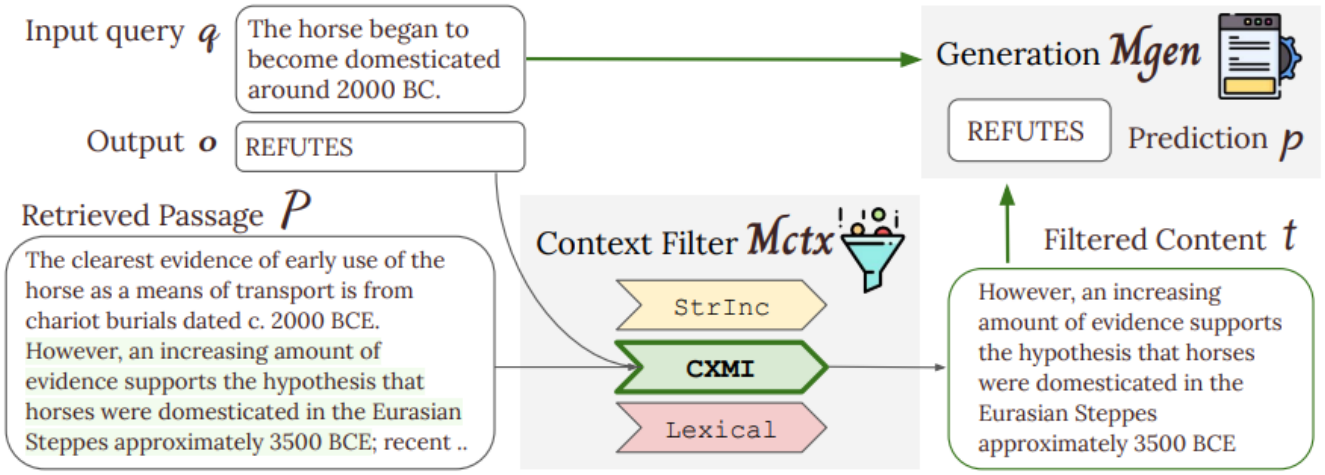


Method

- Test Time
  - Context Filtering Model M_{ctx} 이용
- Training Time
  - String Inclusion: 정답 Output 포함 여부
  - Lexical Overlap: 정답 Output과 문장 간 텍스트 겹침 정도 (Uni-gram)
  - Conditional Cross-Mutual Information(CXMI): 문장이 제공됐을 때 Generator가 정답 Output을 생성할 확률

Method

- q : 입력 Query
- o : 정답 Output
- e : 예시 한 세트, $e = \{q, o\}$
  - M_{ctx} : Context Filtering Model
    - 검색된 Passages P 는 주어진 상황
    - Retrieve된 Passage P의 내용을 필터링해 Filtered Content t를 반환
    - 학습을 위한 데이터
      - Input: $\{q, P\}$
      - Output: t
  - M_{gen} : 생성 모델
    - Filtered Content t와 Input Query q를 입력으로 받아 답변 p를 생성



Method

- Context Filtering 모델을 학습시키기 위한 Oracle Filtered Content 선별
- Filtering 함수 $f(\ )$ 사용
  - Query q, 정답 Output o, Passage P를 입력
  - Best Span T를 반환
    1. Passage P를 spaCy Tokenizer를 이용해 문장 단위로 Split
    2. 모든 문장들 중 $f(\ )$ 값을 최대로 하는 Span 하나를 Best Span T로 반환
- 사용된 Filtering 함수 $f(\ )$ 종류
  - String Inclusion
  - Lexical Overlap
  - Conditional Cross-Mutual Information(CXMI)

Method

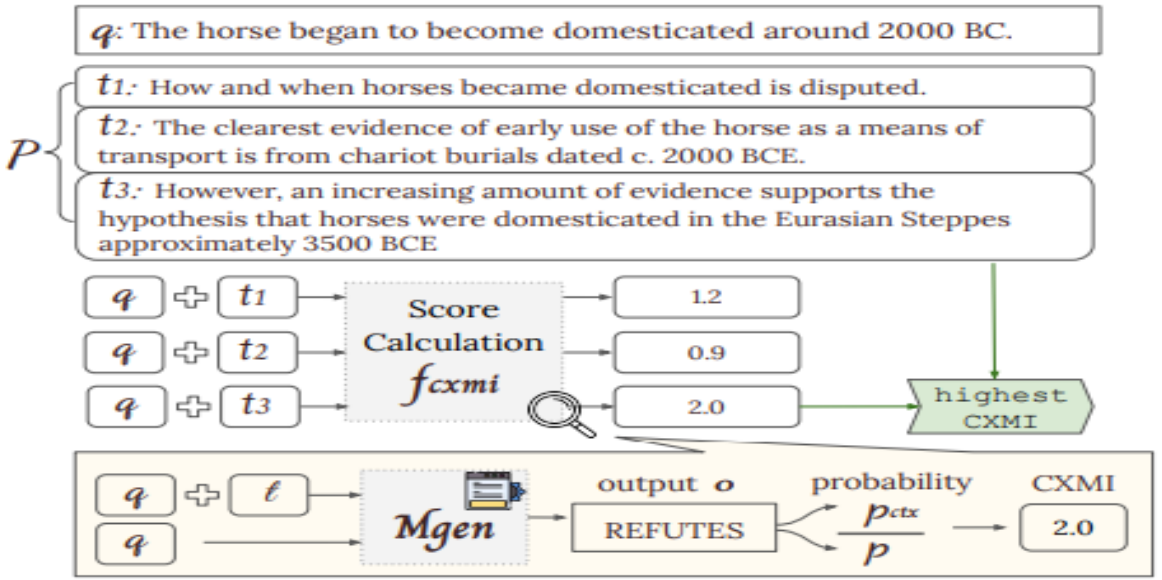
- String Inclusion
  - $f_{inc}(t, o) \in \{0, 1\}$
  - Text Span t가 Output o를 텍스트 그대로 포함하고 있는지 여부
    - Output o를 포함하는 첫 번째 Text Span t를 Best Span T로 반환
  - 단점
    - Passage 내에 정답 Output 텍스트를 그대로 포함하고 있는 경우에만 사용 가능
    - Abstractive Task에는 적용 불가

Method

- Lexical Overlap
  - $f_{uf1} \in [0, 1]$
  - Example $e = \{q, o\}$와 Text Span t 간의 Unigram F1 Score를 측정
  - Task에 따라 Filtering 함수 입력값이 다름
    - Knowledge-grounded Response Generation: $f_{uf1}(\mathrm{t}, \mathrm{o}) \in [0, 1]$
    - Fact Verification: $f_{uf1}(\mathrm{t}, \mathrm{q}) \in [0, 1]$
  - Unigram F1 Score가 임계값 $\lambda = 0.5$ 이상인 값들 중 가장 큰 값인 Text Span t를 Best Span T로 반환
  - 단점
    - Query 자체가 factually incorrect할 경우(예: fact verification)
    - Nonfactual한 Span을 BestSpan으로 반환해 부정확한 생성 결과를 유도할 수 있음

Method

- Conditional Cross-Mutual Information(CXMI)
- $f_{\mathrm{cxmi}}(\mathrm{t},\mathrm{e}) = \frac{\mathrm{M_{gen}}(\mathrm{o}|\mathrm{t}\oplus\mathrm{q})}{\mathrm{M_{gen}}(\mathrm{o}|\mathrm{t})} \in \mathbb{R}$
- $\mathrm{M_{gen}}$ Input에 Text span이 있는 경우와 없는 경우에 대한 정답 Output 생성 확률의 차이 측정
- CXMI Score가 Threshold λ =1.0 보다 크고, 가장 값이 큰 Text Span t 를 Bext Span T로 반환
- Lexical한 방법인 String Inclusion과 Lexical Overlap의 한계 극복 & 모든 Task에 적용 가능
- 단점
  - Computational Cost가 큼



Method

- Context Filtering Model
- Train: Oracle Filtered Content를 이용해 학습
  - 세 Filtering 방법을 이용해 각 학습 데이터 인스턴스에 대한 Filtered Content t_{silver} 확보
  - Context Filtering Model M_{ctx} 학습
    - Input: [Query q ; Retrieved Passage P]
    - Output: t_{pred} ↔ Label: t_{silver}
- Inference: 학습된 Context Filtering Model 이용해 Span 반환
  - Context Filtering Model 을 통해 Filtering된 Span을 Query와 함께 Generation Model의 Input으로 사용

Method

- Generation Model
- Train: Oracle Filtered Content를 이용해 학습
  - Input: [t_{silver} ; Query $\mathbf{q}$]
  - Output: 정답 $\mathbf{o}$
- Inference: 학습된 Context Filtering Model의 예측 결과를 이용해 추론
  - Input: [t_{pred} ; Query $\mathbf{q}$]
  - Output: 정답 $\mathbf{o}$

  - 모든 Retrieved Text Span을 Prepend 하는 대신, 일부 Span만을 사용
  - 모델 학습, 추론 시점에서 연산량 감소

Experiments & Analysis

- Knowledge-IntensiveTasks
  - Open-DomainQuestionAnswering(ODQA)
    - NaturalQuestions(NQ),TriviaQA(TQA):정답 Output이 Extractive한 형태: ExactMatch 이용 평가
- Multi-hopQA
  - HotpotQA :정답 Output이 Abstractive: UnigramF1 이용 평가
- Long-formQA
  - ELI5:정답 Output이 Abstractive하고 김: UnigramF1 이용 평가
- FactVerification
  - FEVER: Accuracy 이용 평가
- Knowledge-groundedDialogGeneration
  - WoW: UnigramF1이용 평가

Dataset	# Examples (thousands) train	dev	test	Evaluation Metric
NQ	79.2	8.7	3.6	EM
TQA	78.8	8.8	11.3	EM
HotpotQA	88.9	5.6	5.6	F_1
ELI5	273.0	1.5	0.6	F_1
FEVER	105.0	10.4	10.1	Accuracy
WoW	63.7	3.1	2.9	F_1

Experiments & Analysis

- Experiment Setting
- Base Model
  - Faln-T5 XL (3B)
  - LLAMA2 7B (LoRA Tuning)
    - 각 모델을 Context Filtering Model M_{ctx}와 Generation Model M_{gen}로 사용

- Baseline
  - Augmenting with Full Passages(FULL)
    - 모든 Passage를 다 사용
  - Passage-Wise Filtering(PSG)
    - Passage 단위로 Filtering

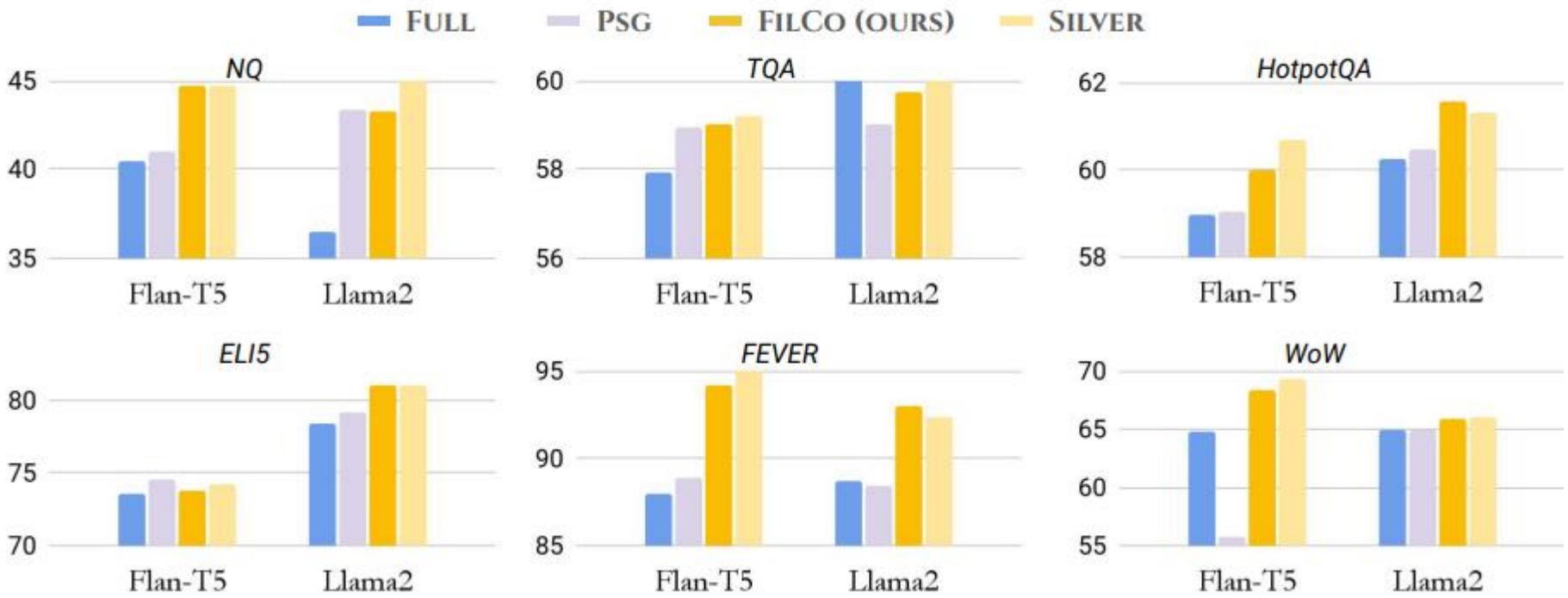
Experiments & Analysis

- Experiment Setting
- FilCo
  - Dataset 별 t_{silver} 산출 위해 적용한 Filtering 방법
  - NQ,TQA:StringInclusion
  - FEVER:LexicalOverlap
  - Wow,HotpotQA,ELI5:CXMI
  - Top-1 Passage로부터 t_{silver}를 반환하도록 M_{ctx} 학습
  - t_{silver}를 이용해 정답 $\mathbf{Output\ o}$를 생성하도록 M_{gen} 학습

  - Upper bound를 확인하기 위한 Additional Experiment
    - M_{ctx}의 예측 결과가 아닌 Oracle Filtered Content인 t_{silver}를M_{gen}의 input으로 사용

Experiments & Analysis

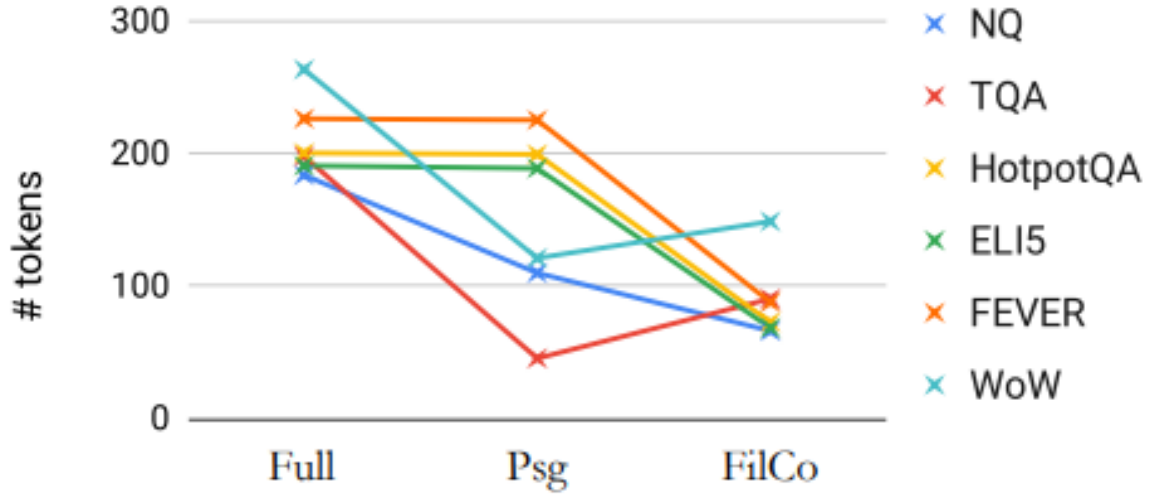
- Main Experiment



- 전체 Passage를 Input에 넣는 것보다 일부만을 넣는 경우가 더 높은 성능을 보임 → Filtering의 효과 존재
- 더 Fine-grained하게 Filtering하는 FilCo가 PSG보다 높은 성능 보임 → 단일 Passage 내에서도 문장 단위 필터링 효과 존재
- FilCo와 SILVER 유사한 성능 à ContextFilteringModel을 이용해 Filtering하는 것이 효과적

Experiments & Analysis

- Context Filtering 결과 추가 분석 : # of Input Tokens, Precision



Method	FULL	PSG	FILCO	SILVER
NQ	2.5	1.3	5.1	**7.3**
TQA	4.5	3.0	**8.4**	4.6
HOTPOTQA	2.6	2.6	10.8	**17.1**
ELI5	92.9	92.5	**98.8**	**98.8**
FEVER	1.2	1.2	**5.1**	4.4
WOW	10.8	35.5	62.9	**71.5**

- 모델 InputToken 수가 FilCo 적용 후 Full대비 44% ~ 64% 감소 → 모델 연산량 감소
- Context와 Output의 UnigramPrecision 측정 결과, Filtering 후에 Context가 모든 Task에서 더 높은 Precision
  - 필터링을 적용한 결과, Context 내에 정답 생성에 필요한 Token들이 더 많은 비율을 차지함
  - 정답 생성에 불필요한 정보를 걸러낸 덕분에 최종적으로 성능 향상까지 연결되었을 것

Experiments & Analysis

- Generation with Multiple Passages
- Top-5 Passage를 Input으로 사용

Measure	StrInc	Lexical	cxmi
Flan-T5			
NQ	**44.7**	30.0	39.9
TQA	**59.2**	39.0	45.3
HotpotQA	59.2	57.4	**60.0**
ELI5	73.6	73.9	**74.2**
FEVER	80.9	86.4	**95.8**
WoW	63.4	**69.3**	66.6
Llama 2			
NQ	**43.3**	35.2	41.8
TQA	**60.7**	57.1	**60.7**
HotpotQA	59.5	61.1	**61.3**
ELI5	78.6	**78.8**	72.8
FEVER	86.6	88.4	**92.3**
WoW	65.5	**66.0**	65.4

Conclusion

- Retrieve된 Passage 내에서도 정답 생성에 필요한 내용을 문장 단위로 필터링하는 방법론인 FilCo 제안
- FlanT5와 LLaMA2를 이용해 6개의 Knowledge-intensive데이터셋에 실험한 결과 성능이 모두 향상됨
- Relevant Passage일 확률이 높은 Top-1 Passage라 할지라도 특정 문장만을 필터링해 사용하는 것의 효과 확인
  - 전체 Passage를 사용하는 것 대비 성능 향상 뿐만 아니라 연산량 감소에도 효과적
  - 데이터셋의 특성에 따라 최적의 Filtering 방법이 다름을 확인

Open question

- 더 나은 Filtering 함수가 있을까?