Reading Wikipedia to Answer Open-Domain Questions

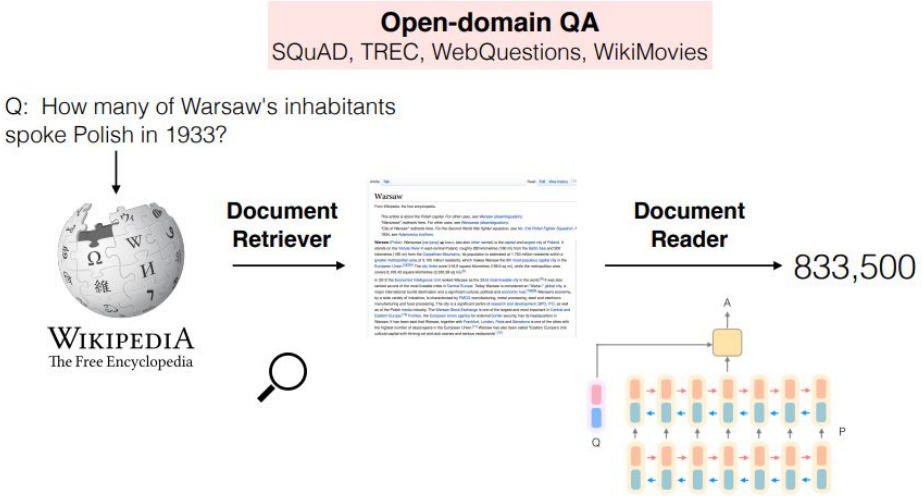
집현전 3기 5조 - 신원지, 한다솜, 김택현, 한나연

목차

Open Domain Question Answering(ODQA)

- 백과사전에서 정보를 찾듯, 엄청나게 많은 정보를 포함하는 **대량의 문서** (ex: Wikipedia) 에서 질문에 대한 답변을 찾는 Task.
- MRS : “Machine Reading at scale”
  - Document Retriever : 정보 검색
  - Document Reader : 정보 추출

open-domain QA and of machine comprehension of text. In order to answer any question, one must first retrieve the few relevant articles among more than 5 million items, and then scan them carefully to identify the answer. We term this setting, *machine reading at scale* (MRS). Our work treats



1. Introduction

Types of questions

1) Factoid type questions [what, which, when, who, how]

Q. 대한민국의 수도는 어디인가? → A. 서울 [Named Entity]

2) List type questions

Q. 집현전 3기에서는 어떤 논문을 리뷰했는가? → A. Attention, BERT, XLNet, …

3) Confirmation questions [yes or no]

Q. 오늘은 일요일인가? → A. 네

4) Causal questions [why or how]

Q. 김대리는 왜 회사에 늦었는가? → A. 교통사고가 나 차가 막혔기 때문이다.

5) Hypothetical questions [no specific answers]

Q. 만약 남한과 북한이 통일되면 어떻게 될까? (what would happen if ~ 로 시작하는 질문) → A. ???

6) Complex questions

Q. 부동산 가격 상승의 원인은 무엇인가? → A. ???

(내 독해)

Abstract

This paper proposes to tackle open-domain question answering using Wikipedia as the unique knowledge source: the answer to any factoid question is a text span in a Wikipedia article.

1. Introduction

Wikipedia 란?

- 누구나 자유롭게 편집할 수 있는 백과사전으로 계속 갱신됨
- 사람이 관심있어하는 내용이 담겨있으며, 컴퓨터보다는 사람이 읽기 편한 구조로 되어있음

Wikipedia를 knowledge source로 사용

1) 내부의 그래프 구조는 제외하고 **텍스트 데이터**만 사용

   → KB(Knowledge Base)를 다른 data로 변환할 수 있는 general model

2) Only Wikipedia for KB

   → IBM의 DeepQA와 같은 large-scale QA 시스템은 위키피디아, knowledge base(지식 베이스), 사전, 뉴스, 책 등 다양한 종류의 데이터셋을 사용하여 정보의 중복성을 활용함.

   → 문제는, 문서에 답변에 대한 evidence가 한번만 나타난 경우, 정확하게 정답을 찾아내기 어려웠음

Challenges

- **Large-scale open-domain QA**
- **Machine comprehension**

2. Related Work

Open-domain QA

- 대량의 **비정형 데이터**에서 질문에 대한 정답을 찾는 것
- **Text REtrieval Conference(TREC)** 에서 시작된 task
    - QA main

Machine Comprehension 분야의 발전

- 딥러닝 구조: Attention-based, memory augmented neural networks)
- 다양한 데이터셋 등장: QuizBowl(정보가 주어지면 4가지 선택지 중 하나를 선택-사지선다 문제), CNN/Daily Mail(기사), CBT(어린이 책), SQuAD(위키피디아)
- Ryu et al. 위키피디아와 semi-structured data를 결합해 사용

2. Related Work

다양한 종류의 QA pipeline

- Microsoft's AskMSR: 검색 엔진 기반 QA 시스템으로 질문과 정답에 대한 언어학적 해석보다는 데이터 중복(data redundancy)에 기반
- IBM's DeepQA: unstructured data(텍스트)와 structured data(데이터베이스, 온톨로지) 모두 사용
- YodaQA: DeepQA 이후에 등장한 오픈소스 factoid QA 시스템으로 웹사이트, 데이터베이스, 위키피디아에서 사용 가능

Multitask learning

- 데이터를 확보하기 어려운 경우 여러 task를 동시에 학습시키는 방법으로 연관있는 task를 동시에 학습시킴.
- 모든 task의 성능 향상을 목표로 하며, target task와 source task가 구분되어 target task의 학습을 목표로 하는 transfer learning(전이 학습)과는 다름

Task transfer:

- Task A에서 학습한 가중치를 Task B로 transfer
- 예를 들어, image classification과 image segmentation은 둘 다 visual task이므로 segmentation 데이터셋이 부족하면 image classification으로 학습된 가중치를 공유하는 idea

3. Our System : DrQA

Document Retriever

- The Document Retriever module for finding relevant articles
- 관련 **문서 검색** 모듈
- **Using bi-gram hashing and TF-IDF matching**

Document Reader

- A machine comprehension model, Document Reader, for extracting answers
- **답변 추출** 모듈
- **Multi-layer RNN (LSTM)**

3.1. Document Retriever

- **TF-IDF + N-gram**
  - Articles와 Question을 TF-IDF weighted BoW로 연관 문서를 검색
- **Bi-gram**에서 가장 좋은 성능을 보임
  - 단어 순서를 고려한 선택
- 각 질문으로부터 **5개의 wikipedia articles**를 return하도록 셋팅
- 데이터 처리에 Murmur3 hash 알고리즘을 활용하여 속도와 메모리 효율성을 보존

3.2. Document Reader

- (논문 출판 당시) 기계 이해 task에서 neural network model의 성공적인 결과에서 영감을 얻었다.
- **Multi-layer RNN**
- A Question : l개의 토큰들로 구성
  - $\{q_1, \ldots, q_l\}$
- A Paragraph : m개의 토큰들로 구성
  - $\{p_1, \ldots, p_m\}$
- n개의 paragraph(문단)로 된 문서 집합이 주어졌을 때, 각각의 paragraph를 차례로 적용한 다음 궁극적으로 예측된 답과 결합하는 RNN 모델을 개발한다.

3.2. Document Reader

1. Paragraph Encoding
    a. Word embeddings
    b. Exact match
    c. Token features
    d. Aligned question embedding
2. Question Encoding
3. Prediction

3.2. Document Reader - Paragraph Encoding

- 문단(Paragraph) 내 모든 토큰을 feature vector로 변환
- Feature vector을 RNN의 Input으로하여 주변 context 정보를 담은 $\tilde{\mathbf{p}}_i$를 얻음
- Multi-layer bidirectional LSTM

$$\{\mathbf{p}_1, \ldots, \mathbf{p}_m\} = \mathrm{RNN}(\{\tilde{\mathbf{p}}_1, \ldots, \tilde{\mathbf{p}}_m\}),$$

3.2. Document Reader - Paragraph Encoding

$\tilde{\mathbf{p}}_i$ 는 다음 4가지 부분으로 이루어짐

- 1. Word Embedding
  - $f_{emb}(p_i) = \mathbf{E}(p_i)$
  - 300차원의 GloVe : 840B Web crawl data로 훈련
  - 기존의 word embedding은 고정, 가장 자주 묻는 **1000개의 질문** 단어에 대해서만 fine tuning
    - “What, how, which, many” 와 같은 단어는 QA 시스템에서 핵심적인 표현임.
- 2. Exact Match
  - $f_{exact\_match}(p_i) = \mathbb{I}(p_i \in q)$
  - Question에 등장한 단어 q와 paragraph의 p 토큰의 매칭 여부의 binary feature을 활용
  - Original, lower-case, lemma form 세가지 경우의 exactly matching 여부
  - Extremely helpful 했다고 함.

3.2. Document Reader - Paragraph Encoding

- 3. Token features
  - $f_{token}(p_i) = (\mathrm{POS}(p_i), \mathrm{NER}(p_i), \mathrm{TF}(p_i))$
  - 품사(Part-of-speech, POS), 개체명 인식(NER) 태그와 TF(Term Frequency) 속성을 추가
- 4. Aligned question embedding

$$f_{align}(p_i) = \sum_j a_{i,j}\mathbf{E}(q_j)$$

$$a_{i,j} = \frac{\exp\left(\alpha(\mathbf{E}(p_i)) \cdot \alpha(\mathbf{E}(q_j))\right)}{\sum_{j'} \exp\left(\alpha(\mathbf{E}(p_i)) \cdot \alpha(\mathbf{E}(q_{j'}))\right)},$$

an aligned question embedding $f_{align}(p_i) = \sum_j a_{i,j}\mathbf{E}(q_j)$, where the attention score $a_{i,j}$ captures the similarity between p_i and each question words q_j. Specifically, $a_{i,j}$ is computed by the dot products between nonlinear

  - Exact match와 다른 측면으로, 유사어(e.g., car and vehicle) 에 대한 **soft-alignments**을 가능하게 한다.
    - c.f. soft-alignments : 사람의 word 2 word 라벨링 없이 기계가 직접 alignment(순서)를 학습
    - ↔ hard alignments : 사람에 의해 학습. ex) I / am / hungry -> 나 / 는 / 배고프다

3.2. Document Reader - Question Encoding

- A question : $\{q_1, \ldots, q_l\}$
- Combine the resulting hidden units into One single vector

$$\mathbf{q} = \sum_j b_j \mathbf{q}_j \qquad \{\mathbf{q}_1, \ldots, \mathbf{q}_l\} \rightarrow \mathbf{q}.$$

$$b_j = \frac{\exp(\mathbf{w} \cdot \mathbf{q}_j)}{\sum_{j'} \exp(\mathbf{w} \cdot \mathbf{q}_{j'})},$$

w is a weight vector to learn.

- b_j : question word의 각 중요도를 encoding

3.2. Document Reader - Prediction

- Paragraph들과 question의 유사성을 포착하고 **answer 범위의 start, end 두 끝을 예측**하기 위해 2개의 classifier 활용함

1. Paragraph vector와 Question vector를 입력받는다. (from **document retriever**)

2. Predict start and end positions

$$P_{start}(i) \propto \exp(\mathbf{p}_i \mathbf{W}_s \mathbf{q})$$
$$P_{end}(i) \propto \exp(\mathbf{p}_i \mathbf{W}_e \mathbf{q})$$

3. score 가 최대화 될 수 있는 부분(best span)을 선택

$$P_{start}(i) \times P_{end}(i')$$

4. Data

- 3 types of data
  - Wikipedia
  - SQuAD
  - QA datasets(CuratedTREC, WebQuestions, and WikiMovies)

4.1 Wikipedia(Knowledge Source)

- 질의 응답을 위한 전체규모 실험을 위해 영어 위키피디아의 2016-12-21 dump 사용
- 각 페이지는 plain text만 추출
- 모든 구조화된 데이터(목록, 그림 등) 제거
- 5,075,182개의 기사들은 9,008,962개의 unique uncased token type으로 구성

4.2 SQuAD

- Stanford Question Answering Dataset(SQuAD)
- 위키피디아 기반의 기계이해를 위한 dataset
  - Training set : 87k
  - Development set : 10k
  - Large hidden set : SQuAD creator만 접근 가능
- 위키피디아 글에서 추출된 단락으로 구성
- 답변은 항상 단락 길이
- 예측답변이 맞으면 model에게 credit 부여
- 사용 metrics : string match(EM), F1 score(token level에서의 precision & recall 가중평균 측정

4.2 SQuAD

- 관련 단락에 주어진 standard machine comprehension task를 위한 Document Reader를 학습과 평가하는데 이용
- SQuAD development set은 QA pair에만 사용
- 모델이 해당 질문에 대답하는 Resource로서 위키피디아 전체 사용
- 시스템은 연관 단락 접근 없이 올바른 answer span을 알려줌

4.3 Open-domain QA Evaluation Resources

- SQuAD 데이터는 사람이 직접 보고 답변하는 과정을 거친 데이터, 명확한 분포를 가짐
- 다른 방식으로 구축된 dataset에서의 학습과 평가
- Datasets
  - CuratedTREC
  - WebQuestions
  - WikiMovies

4.3 Open-domain QA Evaluation Resources

- CuratedTREC
  - TREC QA task에서의 benchmark 기반
  - 2180개의 질문 포함
- WebQuestions
  - Freebase KB에서 질의 응답을 위해 제작
  - Google Suggest API로 크롤링한 질문으로 생성
  - Entity name을 이용해 각 응답을 text로 변환
  - Dataset은 Freebase ID를 참조하지 않고 plain text의 질의 응답 쌍으로 구성
- WikiMovies
  - 영화 도메인의 96k 질의 응답 쌍
  - Knowledge source로써의 Wikipedia의 부분집합으로 사용될 수 있음

4.3 Open-domain QA Evaluation Resources

- QA dataset에서 training data 예시

Dataset	Example	Article / Paragraph
SQuAD	**Q**: How many provinces did the Ottoman empire contain in the 17th century? **A**: 32	**Article**: Ottoman Empire **Paragraph**: ... At the beginning of the 17th century the empire contained 32 provinces and numerous vassal states. Some of these were later absorbed into the Ottoman Empire, while others were granted various types of autonomy during the course of centuries.
CuratedTREC	**Q**: What U.S. state's motto is "Live free or Die"? **A**: New Hampshire	**Article**: Live Free or Die **Paragraph**: "Live Free or Die" is the official motto of the U.S. state of New Hampshire, adopted by the state in 1945. It is possibly the best-known of all state mottos, partly because it conveys an assertive independence historically found in American political philosophy and partly because of its contrast to the milder sentiments found in other state mottos.
WebQuestions	**Q**: What part of the atom did Chadwick discover?† **A**: neutron	**Article**: Atom **Paragraph**: ... The atomic mass of these isotopes varied by integer amounts, called the whole number rule. The explanation for these different isotopes awaited the discovery of the neutron, an uncharged particle with a mass similar to the proton, by the physicist James Chadwick in 1932. ...
WikiMovies	**Q**: Who wrote the film Gigli? **A**: Martin Brest	**Article**: Gigli **Paragraph**: Gigli is a 2003 American romantic comedy film written and directed by Martin Brest and starring Ben Affleck, Jennifer Lopez, Justin Bartha, Al Pacino, Christopher Walken, and Lainie Kazan.

4.3 Open-domain QA Evaluation Resources

- 각 데이터셋에 사용된 질문 숫자

Dataset	Train		Test
	Plain	DS	
SQuAD	87,599	71,231	10,570†
CuratedTREC	1,486*	3,464	694
WebQuestions	3,778*	4,602	2,032
WikiMovies	96,185*	36,301	9,952

- Document retrieval results

Dataset	Wiki Search	Doc. Retriever	
		plain	+bigrams
SQuAD	62.7	76.1	**77.8**
CuratedTREC	81.0	85.2	**86.0**
WebQuestions	73.7	**75.5**	74.4
WikiMovies	61.7	54.4	**70.3**

4.4. Distantly Supervised Data

- QA datasets은 질의 응답 쌍만 포함하고 SQuAD에 관련이 없어서 training Document Reader로 직접 사용될 수 없음
- Training set 제작 프로세스
  - Top 5 Wikipedia article에 대한 retrieve하는 질문에 대해 Document Retriever 실행
  - 정답과 exact match가 없는 article에서 나온 모든 단락 폐기
  - 25글자 미만, 1,500자 초과 단락 폐기
  - Named entity가 질문에서 나오면, named entity가 없는 단락 폐기
  - 각 retrieved page에서 모든 남은 단락에 대해 질문과 20 token window사이의 unigram & bigram 을 이용한 답변과 맞는 모든 위치에 점수 부여, 가장 많이 겹친 top5 단락 까지 남김
  - Non-zero overlap이 있는 단락이 없는 예시 폐기
  - 이에 맞지 않는 쌍을 distant supervision(DS) training dataset에 추가

5. Experiments

- Document Retriever 와 Document Reader module을 분리해서 평가
- 전체 위키피디아에 대한 open-domain QA(DrQA)를 위한 모듈의 조합에 대한 테스트

5.1 Finding Relevant Articles

5.2 Reader Evaluation on SQuAD

5.3 Full Wikipedia Question Answering

5.1 Finding Relevant Articles

Dataset	Wiki Search	Doc. Retriever	
		plain	+bigrams
SQuAD	62.7	76.1	**77.8**
CuratedTREC	81.0	85.2	**86.0**
WebQuestions	73.7	**75.5**	74.4
WikiMovies	61.7	54.4	**70.3**

Table 3: Document retrieval results. % of questions for which the answer segment appears in one of the top 5 pages returned by the method.

- 질문과 연관 Top-5페이지에 최소 1번 이상 정답이 나타나는 영역(text span)의 비율
- 기존 Wikipedia Search(ElasticSearch 기반) 보다 성능 향상
- bigram hashing을 이용했을 때 더 좋은 성능을 보임
- Okapi BM25사용 retrieval과 word embedding space에서의 코사인 거리를 사용한 것보다 좋은 성능

5.2 Reader Evaluation on SQuAD

- SQuAD 테스트셋에 대한 성능 평가

  -> 70%(Exact Match), 79.0%(F1 score)

Method	Dev		Test	
	EM	F1	EM	F1
Dynamic Coattention Networks (Xiong et al., 2016)	65.4	75.6	66.2	75.9
Multi-Perspective Matching (Wang et al., 2016)†	66.1	75.8	65.5	75.1
BiDAF (Seo et al., 2016)	67.7	77.3	68.0	77.3
R-net†	n/a	n/a	71.3	79.7
DrQA (Our model, Document Reader Only)	**69.5**	**78.8**	70.0	79.0

5.2 Reader Evaluation on SQuAD

- paragraph encoding feature에 aligned question embedding과 exact match에 관한 정보모두를 제거했을 때 성능 저하가 가장 큼

Features	**F1**
Full	78.8
No f_{token}	78.0 (-0.8)
No f_{exact_match}	77.3 (-1.5)
No $f_{aligned}$	77.3 (-1.5)
No $f_{aligned}$ and f_{exact_match}	59.4 (-19.4)

Table 5: Feature ablation analysis of the paragraph representations of our Document Reader. Results are reported on the SQuAD development set.

5.3 Full Wikipedia Question Answering

- 세 가지 버전의 DrQA를 비교
  - SQuAD: SQuAD 훈련 데이터로 학습
  - Fine-tune(DS): SQuAD 데이터로 사전학습 후 각 데이터셋의 훈련 데이터로 파인튜닝(fine-tuning)
  - Multitask(DS): SQuAD 훈련 데이터 + DS로 만든 훈련 데이터로 학습
- 각 수치는 Top-1 exact-match accuracy를 나타냄

Dataset	YodaQA	DrQA		
		SQuAD	+Fine-tune (DS)	+Multitask (DS)
SQuAD *(All Wikipedia)*	n/a	27.1	28.4	29.8
CuratedTREC	31.3	19.7	25.7	25.4
WebQuestions	39.8	11.8	19.5	20.7
WikiMovies	n/a	24.5	34.3	36.5

6. Conclusions

- 핵심 도전과제 : Machine reading at scale(MRS)
- Wikipedia를 unique knowledge source로 이용
- Search, distant supervision, multitask learning 들을 통합해 machine reading comprehension system을 보완하고 task를 해결

부록: Document Reader Implementation Details

- 3-layer bidirectional LSTM
- h=128 hidden units for both paragraph and question encoding
- Stanford CoreNLP toolkit을 이용한 토큰화, 표제어 추출, 품사 태깅, 개체명 인식
- batch_size = 32
- dropout with p = 0.3

감 사 합 니 다