Reading Wikipedia to Answer Open-domain Questions

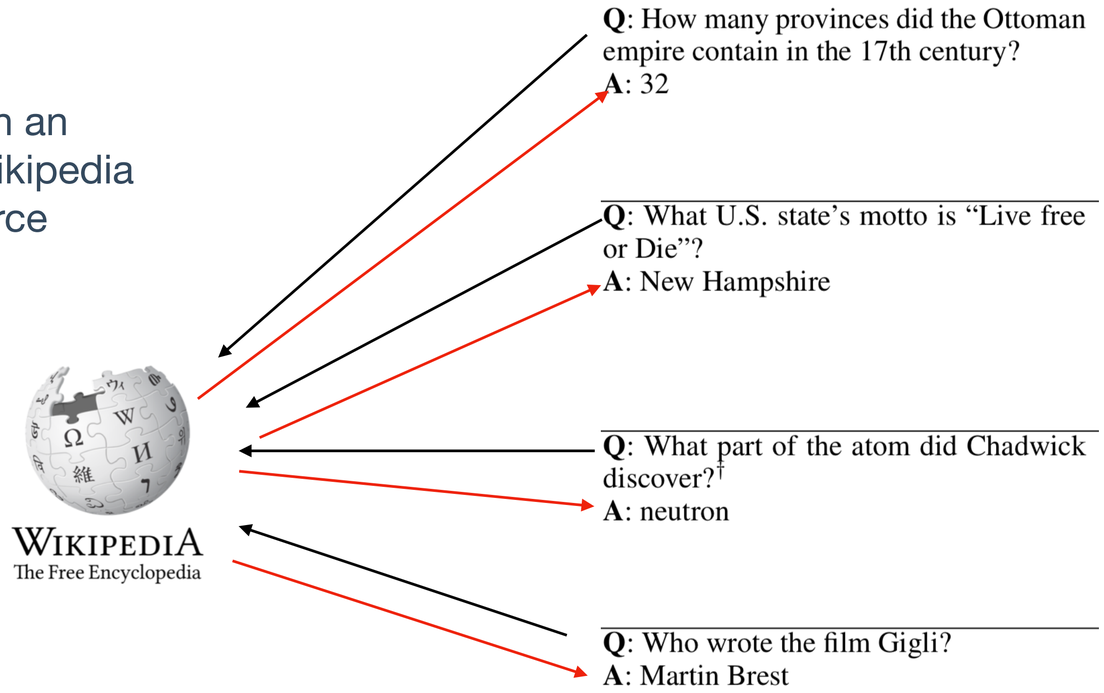
ACL 2017

Danqi Chen
Stanford University
Adam Fisch, **Jason Weston** & **Antoine Bordes**
Facebook AI Research

idejie
2019.05.05

Object

answering factoid questions in an open-domain setting using Wikipedia as the unique knowledge source



Challenges

1.document retrieval (finding the relevant articles)

2.machine comprehension of text (identifying the answer spans from those articles)

Why Wikipedia

WIKIPEDIA
The Free Encyclopedia

1. up-to-date knowledge that humans are interested in.
2. a constantly evolving source of detailed information

How to use it

treats Wikipedia as **a collection of articles** and **does not** rely on its **internal graph structure**.

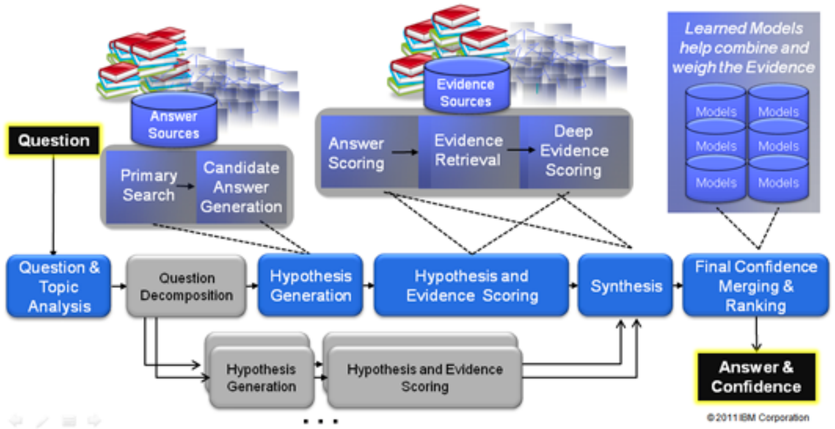
So it is generic and could be switched to other **collections of documents**, **books**, or even daily updated **newspapers**.

Related Works

Knowledge Bases KBs

Pro: easier for computers to process
Con: too sparsely populated for open-domain question answering

Large-scale QA



rely on multiple sources to answer

Wikipedia, KBs, dictionaries, news articles, books, etc.

Pro: large-scale
Con: information redundancy

assume that a short piece of relevant text is already identified and given to the model

Related Works

Origin

QA from unstructured documents

TREC competitions

KBs

QA from KBs

Freebase KB

WebQuestions

SimpleQuestions

QA extracted from KBs

OpenIE triples and NELL

Deep learning architectures

Past

Now

Past:

combine article content with multiple other answer matching modules based on different types of semi-structured knowledge such as infoboxes, article structure, category structure, and definitions.

Now:

use Wikipedia text documents as the sole resource in order to emphasize the task of machine reading at scale, as described in the introduction.

Modules

Open-domain QA

SQuAD, TREC, WebQuestions, WikiMovies

Q: How many of Warsaw's inhabitants spoke Polish in 1933?

WIKIPEDIA
The Free Encyclopedia

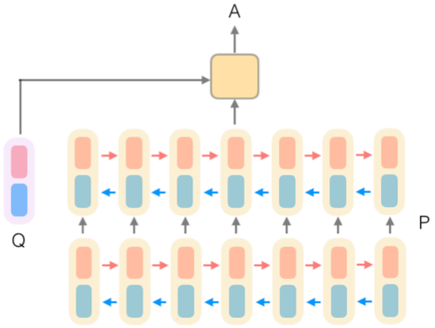
Document Retriever

using bigram hashing and TF-IDF matching designed to, given a question, efficiently return a subset of relevant articles

Document Reader

multi-layer RNN machine comprehension model trained to detect answer spans in those few returned documents.

833,500



Both modules are competitive

1.Document Retriever

Question
Wikipedia Data

Return Top5 articles
For Document Reader

1.narrow our search space
2.focus on reading only articles that are likely to be relevant

A simple inverted index lookup followed by term vector model scoring performs quite well on this task for many question types, compared to the built-in ElasticSearch based Wikipedia Search API

Articles and questions are compared as TF-IDF weighted bag-of- word vectors.

Our **best** performing system uses **bigram** counts while preserving speed and memory efficiency by using the hashing of (Weinberger et al., 2009) to map the bigrams to 2^{24} bins with an **unsigned *murmur3* hash**.

Question
Top5 Articles from Docment Retriever

2.Document Reader

Answer

Assuming

a question q consisting of l tokens

$$\{q_1, \ldots, q_l\}$$

a single paragraph **p** consists of m tokens

$$\{p_1, \ldots, p_m\}$$

Structure Graph

Paragraph Encoding

Vector p

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

Question Encoding

Vector q

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

Prediction

Argmax

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

RNN

The feature vector $\hat{\mathbf{p}}_\mathbf{i}$ is comprised of the fol- lowing parts:

- ***Word embeddings*** : $f_{emb}(p_i) = \mathbf{E}(p_i)$. We use the 300-dimensional Glove word em- beddings trained from 840B Web crawl data (Pennington et al., 2014). We keep most of the pre-trained word embeddings fixed and only fine-tune the 1000 most frequent ques- tion words because the representations of some key words such as *what* , *how* , *which* , *many* could be crucial for QA systems.
- ***Exact match*** : $f_{exact\_match}(p_i) = I(p_i \in q)$. We use three simple binary features, indicating whether p_i can be exactly matched to one question word in q, either in its original, lowercase or lemma form. These simple features turn out to be extremely helpful, as we will show in Section 5.
- ***Token features*** :
  $f_{token}(pi) = (\mathrm{POS}(p_i), \mathrm{NER}(p_i), \mathrm{TF}(p_i))$. We also add a few manual features which reflect some properties of token p_i in its con- text, which include its part-of-speech (POS) and named entity recognition (NER) tags and its (normalized) term frequency (TF).
- ***Aligned question embedding*** :
  Following (Lee et al., 2016) and other re- cent works, the last part we incorporate is 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 mappings of word embeddings:

$$a_{i,j} = \frac{\exp(\quad \alpha(\mathbf{E}(p_i)) \cdot \alpha(\mathbf{E}(qj)))}{\sum_{j'} exp(\quad \alpha(\mathbf{E}(pi)) \cdot \alpha(\mathbf{E}(q_{j'}))\quad)}$$

and $\alpha(\cdot)$ is a single dense layer with ReLU nonlinearity. Compared to the *exact match* features, these features add soft alignments between similar but non-identical words (e.g., *car* and *vehicle*).

Experiment

Data

(1)Wikipedia that serves as our knowledge source for finding answers,

(2)the SQuAD dataset which is our main resource to train Document Reader

(3)three more QA datasets (CuratedTREC, We- bQuestions and WikiMovies) that in addition to SQuAD, are used to test the open-domain QA abilities of our full system

(4)to evaluate the ability of our model to learn from multitask learning and distant supervision.

Processing

1.run Document Retriever on the question to retrieve the top 5 Wikipedia articles

Retrieve Rules:
1.All paragraphs from those articles without an exact match of the known answer are directly discarded.
2.All para- graphs shorter than 25 or longer than 1500 characters are also filtered out.
3.If any named entities are detected in the question, we remove any paragraph that does not contain them at all.
4.If there is no paragraph with non-zero overlap, the example is discarded.

2.use 3-layer bidirec- tional LSTMs with h = 128 hidden units for both paragraph and question encoding.

comprehension benchmark

Compare with counters

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

Compare without different features

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)

Full QA system

Three versions

- SQuAD: A single Document Reader model is trained on the SQuAD training set only and used on all evaluation sets.
- Fine-tune (DS): A Document Reader model is pre-trained on SQuAD and then fine-tuned for each dataset independently using its distant supervision (DS) training set.
- Multitask (DS): A single Document Reader model is jointly trained on the SQuAD training set and *all* the DS sources.

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