CHAPTER

25 Question Answering

The quest for knowledge is deeply human, and so it is not surprising that practically as soon as there were computers we were asking them questions. By the early 1960s, systems used the two major paradigms of question answering—**information-retrieval-based** and **knowledge-based**—to answer questions about baseball statistics or scientific facts. Even imaginary computers got into the act. Deep Thought, the computer that Douglas Adams invented in *The Hitchhiker's Guide to the Galaxy*, managed to answer "the Ultimate Question Of Life, The Universe, and Everything".[1] In 2011, IBM's Watson question-answering system won the TV game-show *Jeopardy!* using a hybrid architecture that surpassed humans at answering questions like

> WILLIAM WILKINSON'S "AN ACCOUNT OF THE PRINCIPALITIES OF WALLACHIA AND MOLDOVIA" INSPIRED THIS AUTHOR'S MOST FAMOUS NOVEL[2]

Most question answering systems focus on **factoid questions**, questions that can be answered with simple facts expressed in short texts. The answers to the questions below can be expressed by a personal name, temporal expression, or location:

(25.1) Who founded Virgin Airlines?

(25.2) What is the average age of the onset of autism?

(25.3) Where is Apple Computer based?

In this chapter we describe the two major paradigms for factoid question answering. Information-retrieval or **IR-based question answering** relies on the vast quantities of textual information on the web or in collections like PubMed. Given a user question, information retrieval techniques first find relevant documents and passages. Then systems (feature-based, neural, or both) use **reading comprehension** algorithms to read these retrieved documents or passages and draw an answer directly from **spans of text**.

In the second paradigm, **knowledge-based question answering**, a system instead builds a semantic representation of the query, mapping *What states border Texas?* to the logical representation: $\lambda x.state(x) \wedge borders(x, texas)$, or *When was Ada Lovelace born?* to the gapped relation: `birth-year (Ada Lovelace, ?x)`. These meaning representations are then used to query databases of facts.

Finally, large industrial systems like the DeepQA system in IBM's Watson are often hybrids, using both text datasets and structured knowledge bases to answer questions. DeepQA finds many candidate answers in both knowledge bases and in textual sources, and then scores each candidate answer using knowledge sources like geospatial databases, taxonomical classification, or other textual sources.

We describe IR-based approaches (including neural reading comprehension systems) in the next section, followed by sections on knowledge-based systems, on Watson Deep QA, and a discussion of evaluation.

[1] The answer was 42, but unfortunately the details of the question were never revealed.

[2] The answer, of course, is 'Who is Bram Stoker', and the novel was *Dracula*.

25.1 IR-based Factoid Question Answering

The goal of information retrieval based question answering is to answer a user's question by finding short text segments on the web or some other collection of documents. Figure 25.1 shows some sample factoid questions and their answers.

Question	Answer
Where is the Louvre Museum located?	in Paris, France
What's the abbreviation for limited partnership?	L.P.
What are the names of Odin's ravens?	Huginn and Muninn
What currency is used in China?	the yuan
What kind of nuts are used in marzipan?	almonds
What instrument does Max Roach play?	drums
What's the official language of Algeria?	Arabic
How many pounds are there in a stone?	14

Figure 25.1 Some sample factoid questions and their answers.

Figure 25.2 shows the three phases of an IR-based factoid question-answering system: question processing, passage retrieval and ranking, and answer extraction.

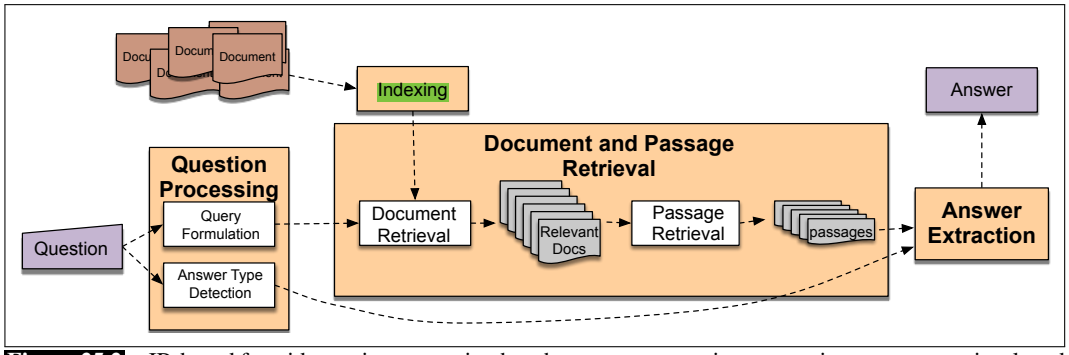


Figure 25.2 IR-based factoid question answering has three stages: question processing, passage retrieval, and answer processing.

25.1.1 Question Processing

The main goal of the question-processing phase is to extract the **query**: the keywords passed to the IR system to match potential documents. Some systems additionally extract further information such as:

- **answer type**: the entity type (person, location, time, etc.) of the answer.
- **focus**: the string of words in the question that is likely to be replaced by the answer in any answer string found.
- **question type**: is this a definition question, a math question, a list question?

For example, for the question *Which US state capital has the largest population?* the query processing might produce:

query: "US state capital has the largest population"
answer type: city
focus: state capital

In the next two sections we summarize the two most commonly used tasks, query formulation and answer type detection.

25.1.2 Query Formulation

Query formulation is the task of creating a query—a list of tokens— to send to an information retrieval system to retrieve documents that might contain answer strings.

For question answering from the web, we can simply pass the entire question to the web search engine, at most perhaps leaving out the question word (*where*, *when*, etc.). For question answering from smaller sets of documents like corporate information pages or Wikipedia, we still use an IR engine to index and search our documents, generally using standard tf-idf cosine matching, but we might need to do more processing. For example, for searching Wikipedia, it helps to compute tf-idf over bigrams rather than unigrams in the query and document (Chen et al., 2017). Or we might need to do query expansion, since while on the web the answer to a question might appear in many different forms, one of which will probably match the question, in smaller document sets an answer might appear only once. Query expansion methods can add query terms in hopes of matching the particular form of the answer as it appears, like adding morphological variants of the content words in the question, or synonyms from a thesaurus.

query reformulation

A query formulation approach that is sometimes used for questioning the web is to apply **query reformulation** rules to the query. The rules rephrase the question to make it look like a substring of possible declarative answers. The question *"when was the laser invented?"* might be reformulated as *"the laser was invented"*; the question *"where is the Valley of the Kings?"* as *"the Valley of the Kings is located in"*. Here are some sample handwritten reformulation rules from Lin (2007):

(25.4) *wh-word* did A *verb* B → ... A *verb*+ed B

(25.5) Where is A → A is located in

25.1.3 Answer Types

question classification
answer type

Some systems make use of **question classification**, the task of finding the **answer type**, the named-entity categorizing the answer. A question like *"Who founded Virgin Airlines?"* expects an answer of type PERSON. A question like *"What Canadian city has the largest population?"* expects an answer of type CITY. If we know that the answer type for a question is a person, we can avoid examining every sentence in the document collection, instead focusing on sentences mentioning people.

answer type taxonomy

While answer types might just be the named entities like PERSON, LOCATION, and ORGANIZATION described in Chapter 18, we can also use a larger hierarchical set of answer types called an **answer type taxonomy**. Such taxonomies can be built automatically, from resources like WordNet (Harabagiu et al. 2000, Pasca 2003), or they can be designed by hand. Figure 25.4 shows one such hand-built ontology, the Li and Roth (2005) tagset; a subset is also shown in Fig. 25.3. In this hierarchical tagset, each question can be labeled with a coarse-grained tag like HUMAN or a fine-grained tag like HUMAN:DESCRIPTION, HUMAN:GROUP, HUMAN:IND, and so on. The HUMAN:DESCRIPTION type is often called a BIOGRAPHY question because the answer is required to give a brief biography of the person rather than just a name.

Question classifiers can be built by hand-writing rules like the following rule from (Hovy et al., 2002) for detecting the answer type BIOGRAPHY:

(25.6) who {is | was | are | were} PERSON

Most question classifiers, however, are based on supervised learning, trained on databases of questions that have been hand-labeled with an answer type (Li and Roth, 2002). Either feature-based or neural methods can be used. Feature based

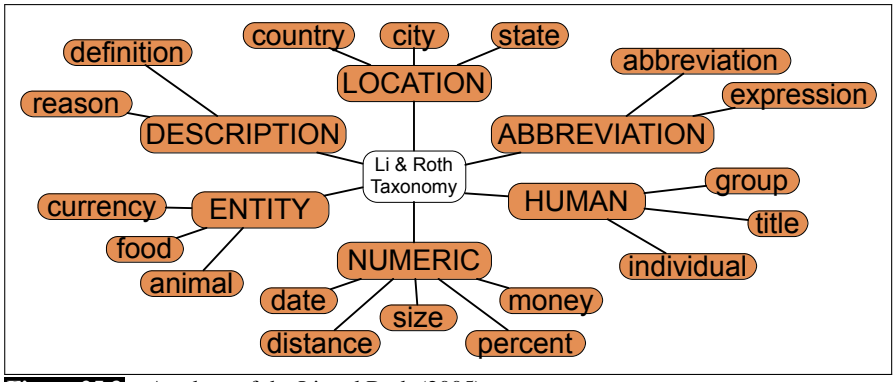


Figure 25.3 A subset of the Li and Roth (2005) answer types.

methods rely on words in the questions and their embeddings, the part-of-speech of each word, and named entities in the questions. Often, a single word in the question gives extra information about the answer type, and its identity is used as a feature. This word is sometimes called the **answer type word** or **question headword**, and may be defined as the headword of the first NP after the question's *wh-word*; headwords are indicated in boldface in the following examples:

(25.7) Which **city** in China has the largest number of foreign financial companies?
(25.8) What is the state **flower** of California?

In general, question classification accuracies are relatively high on easy question types like PERSON, LOCATION, and TIME questions; detecting REASON and DESCRIPTION questions can be much harder.

25.1.4 Document and Passage Retrieval

The IR query produced from the question processing stage is sent to an IR engine, resulting in a set of documents ranked by their relevance to the query. Because most answer-extraction methods are designed to apply to smaller regions such as paragraphs, QA systems next divide the top n documents into smaller **passages** such as sections, paragraphs, or sentences. These might be already segmented in the source document or we might need to run a paragraph segmentation algorithm.

passages

passage retrieval

The simplest form of **passage retrieval** is then to simply pass along every passage to the answer extraction stage. A more sophisticated variant is to filter the passages by running a named entity or answer type classification on the retrieved passages, discarding passages that don't contain the answer type of the question. It's also possible to use supervised learning to fully rank the remaining passages, using features like:

- The number of **named entities** of the right type in the passage
- The number of **question keywords** in the passage
- The longest exact sequence of question keywords that occurs in the passage
- The rank of the document from which the passage was extracted
- The **proximity** of the keywords from the original query to each other (Pasca 2003, Monz 2004).
- The number of n**-grams** that **overlap** between the passage and the question (Brill et al., 2002).

snippets

For question answering from the web we can instead take **snippets** from a Web search engine as the passages.

Tag	Example
ABBREVIATION	
abb	What's the abbreviation for limited partnership?
exp	What does the "c" stand for in the equation E=mc2?
DESCRIPTION	
definition	What are tannins?
description	What are the words to the Canadian National anthem?
manner	How can you get rust stains out of clothing?
reason	What caused the Titanic to sink?
ENTITY	
animal	What are the names of Odin's ravens?
body	What part of your body contains the corpus callosum?
color	What colors make up a rainbow?
creative	In what book can I find the story of Aladdin?
currency	What currency is used in China?
disease/medicine	What does Salk vaccine prevent?
event	What war involved the battle of Chapultepec?
food	What kind of nuts are used in marzipan?
instrument	What instrument does Max Roach play?
lang	What's the official language of Algeria?
letter	What letter appears on the cold-water tap in Spain?
other	What is the name of King Arthur's sword?
plant	What are some fragrant white climbing roses?
product	What is the fastest computer?
religion	What religion has the most members?
sport	What was the name of the ball game played by the Mayans?
substance	What fuel do airplanes use?
symbol	What is the chemical symbol for nitrogen?
technique	What is the best way to remove wallpaper?
term	How do you say " Grandma" in Irish?
vehicle	What was the name of Captain Bligh's ship?
word	What's the singular of dice?
HUMAN	
description	Who was Confucius?
group	What are the major companies that are part of Dow Jones?
ind	Who was the first Russian astronaut to do a spacewalk?
title	What was Queen Victoria's title regarding India?
LOCATION	
city	What's the oldest capital city in the Americas?
country	What country borders the most others?
mountain	What is the highest peak in Africa?
other	What river runs through Liverpool?
state	What states do not have state income tax?
NUMERIC	
code	What is the telephone number for the University of Colorado?
count	About how many soldiers died in World War II?
date	What is the date of Boxing Day?
distance	How long was Mao's 1930s Long March?
money	How much did a McDonald's hamburger cost in 1963?
order	Where does Shanghai rank among world cities in population?
other	What is the population of Mexico?
period	What was the average life expectancy during the Stone Age?
percent	What fraction of a beaver's life is spent swimming?
temp	How hot should the oven be when making Peachy Oat Muffins?
speed	How fast must a spacecraft travel to escape Earth's gravity?
size	What is the size of Argentina?
weight	How many pounds are there in a stone?

Figure 25.4 Question typology from Li and Roth (2002), (2005). Example sentences are from their corpus of 5500 labeled questions. A question can be labeled either with a coarse-grained tag like HUMAN or NUMERIC or with a fine-grained tag like HUMAN:DESCRIPTION, HUMAN:GROUP, HUMAN:IND, and so on.

25.1.5 Answer Extraction

The final stage of question answering is to extract a specific answer from the passage, for example responding *29,029 feet* to a question like *"How tall is Mt. Everest?"*. This task is commonly modeled by **span labeling**: given a passage, identifying the **span** of text which constitutes an answer.

span

A simple baseline algorithm for answer extraction is to run a named entity tagger on the candidate passage and return whatever span in the passage is the correct answer type. Thus, in the following examples, the underlined named entities would be extracted from the passages as the answer to the HUMAN and DISTANCE-QUANTITY questions:

"Who is the prime minister of India?"
Manmohan Singh, Prime Minister of India, had told left leaders that the deal would not be renegotiated.

"How tall is Mt. Everest?"
The official height of Mount Everest is 29029 feet

Unfortunately, the answers to many questions, such as DEFINITION questions, don't tend to be of a particular named entity type. For this reason modern work on answer extraction uses more sophisticated algorithms, generally based on supervised learning. The next section introduces a simple feature-based classifier, after which we turn to modern neural algorithms.

25.1.6 Feature-based Answer Extraction

Supervised learning approaches to answer extraction train classifiers to decide if a span or a sentence contains an answer. One obviously useful feature is the answer type feature of the above baseline algorithm. Hand-written regular expression patterns also play a role, such as the sample patterns for definition questions in Fig. 25.5.

Pattern	Question	Answer
<AP> such as <QP>	What is autism?	", developmental disorders such as autism"
<QP>, a <AP>	What is a caldera?	"the Long Valley caldera, a volcanic crater 19 miles long"

Figure 25.5 Some answer-extraction patterns using the answer phrase (AP) and question phrase (QP) for definition questions (Pasca, 2003).

Other features in such classifiers include:

Answer type match: True if the candidate answer contains a phrase with the correct answer type.

Pattern match: The identity of a pattern that matches the candidate answer.

Number of matched question keywords: How many question keywords are contained in the candidate answer.

Keyword distance: The distance between the candidate answer and query keywords.

Novelty factor: True if at least one word in the candidate answer is novel, that is, not in the query.

Apposition features: True if the candidate answer is an appositive to a phrase containing many question terms. Can be approximated by the number of question terms separated from the candidate answer through at most three words and one comma (Pasca, 2003).

Punctuation location: True if the candidate answer is immediately followed by a comma, period, quotation marks, semicolon, or exclamation mark.

Sequences of question terms: The length of the longest sequence of question terms that occurs in the candidate answer.

25.1.7 N-gram tiling answer extraction

n-gram tiling

An alternative approach to answer extraction, used solely in Web search, is based on **n-gram tiling**, an approach that relies on the **redundancy** of the web (Brill et al. 2002, Lin 2007). This simplified method begins with the snippets returned from the Web search engine, produced by a reformulated query. In the first step, n-gram mining, every unigram, bigram, and trigram occurring in the snippet is extracted and weighted. The weight is a function of the number of snippets in which the n-gram occurred, and the weight of the query reformulation pattern that returned it. In the n-gram filtering step, n-grams are scored by how well they match the predicted answer type. These scores are computed by handwritten filters built for each answer type. Finally, an n-gram tiling algorithm concatenates overlapping n-gram fragments into longer answers. A standard greedy method is to start with the highest-scoring candidate and try to tile each other candidate with this candidate. The best-scoring concatenation is added to the set of candidates, the lower-scoring candidate is removed, and the process continues until a single answer is built.

25.1.8 Neural Answer Extraction

Neural network approaches to answer extraction draw on the intuition that a question and its answer are semantically similar in some appropriate way. As we'll see, this intuition can be fleshed out by computing an embedding for the question and an embedding for each token of the passage, and then selecting passage spans whose embeddings are closest to the question embedding.

Reading Comprehension

reading comprehension

Neural answer extractors are often designed in the context of the **reading comprehension** task. It was Hirschman et al. (1999) who first proposed to take children's reading comprehension tests—pedagogical instruments in which a child is given a passage to read and must answer questions about it—and use them to evaluate machine text comprehension algorithm. They acquired a corpus of 120 passages with 5 questions each designed for 3rd-6th grade children, built an answer extraction system, and measured how well the answers given by their system corresponded to the answer key from the test's publisher.

Since then reading comprehension has become both a task in itself, as a useful way to measure natural language understanding performance, but also as (sometimes called the **reader** component of question answerers).

Reading Comprehension Datasets. Modern reading comprehension systems tend to use collections of questions that are designed specifically for NLP, and so are large enough for training supervised learning systems. For example the Stanford Question Answering Dataset (**SQuAD**) consists of passages from Wikipedia and associated questions whose answers are spans from the passage, as well as some questions that are designed to be unanswerable (Rajpurkar et al. 2016, Rajpurkar et al. 2018); a total of just over 150,000 questions. Fig. 25.6 shows a (shortened) excerpt from a SQUAD 2.0 passage together with three questions and their answer spans.

SQuAD

Beyoncé Giselle Knowles-Carter (born September 4, 1981) is an American singer, songwriter, record producer and actress. Born and raised in Houston, Texas, she performed in various singing and dancing competitions as a child, and rose to fame in the late 1990s as lead singer of R&B girl-group Destiny's Child. Managed by her father, Mathew Knowles, the group became one of the world's best-selling girl groups of all time. Their hiatus saw the release of Beyoncé's debut album, Dangerously in Love (2003), which established her as a solo artist worldwide, earned five Grammy Awards and featured the Billboard Hot 100 number-one singles "Crazy in Love" and "Baby Boy".
Q: "In what city and state did Beyoncé grow up?" A: "Houston, Texas"
Q: "What areas did Beyoncé compete in when she was growing up?" A: "singing and dancing"
Q: "When did Beyoncé release Dangerously in Love?" A: "2003"

Figure 25.6 A (Wikipedia) passage from the SQuAD 2.0 dataset (Rajpurkar et al., 2018) with 3 sample questions and the labeled answer spans.

SQuAD was built by having humans write questions for a given Wikipedia passage and choose the answer span. Other datasets used similar techniques; the NewsQA dataset consists of 100,000 question-answer pairs from CNN news articles. For other datasets like WikiQA the span is the entire sentence containing the answer (Yang et al., 2015); the task of choosing a sentence rather than a smaller answer span is sometimes called the **sentence selection** task.

sentence selection

25.1.9 A bi-LSTM-based Reading Comprehension Algorithm

Neural algorithms for reading comprehension are given a question q of l tokens $q_1,...,q_l$ and a passage p of m tokens $p_1,...,p_m$. Their goal is to compute, for each token p_i the probability $p_{\text{start}}(i)$ that p_i is the start of the answer span, and the probability $p_{\text{end}}(i)$ that p_i is the end of the answer span.

Fig. 25.7 shows the architecture of the Document Reader component of the DrQA system of Chen et al. (2017). Like most such systems, DrQA builds an embedding for the question, builds an embedding for each token in the passage, computes a similarity function between the question and each passage word in context, and then uses the question-passage similarity scores to decide where the answer span starts and ends.

Let's consider the algorithm in detail, following closely the description in Chen et al. (2017). The question is represented by a single embedding $\mathbf{q}$, which is a weighted sum of representations for each question word q_i. It is computed by passing the series of embeddings $P\mathbf{E}(q_1),...,\mathbf{E}(q_l)$ of question words through an RNN (such as a bi-LSTM shown in Fig. 25.7). The resulting hidden representations $\{\mathbf{q}_1,...,\mathbf{q}_l\}$ are combined by a weighted sum

$$\mathbf{q} = \sum_j b_j \mathbf{q}_j \tag{25.9}$$

The weight b_j is a measure of the relevance of each question word, and relies on a learned weight vector $\mathbf{w}$:

$$b_j = \frac{exp(\mathbf{w}\cdot\mathbf{q}_j)}{\sum_{j'} exp(\mathbf{w}\cdot\mathbf{q}'_j)} \tag{25.10}$$

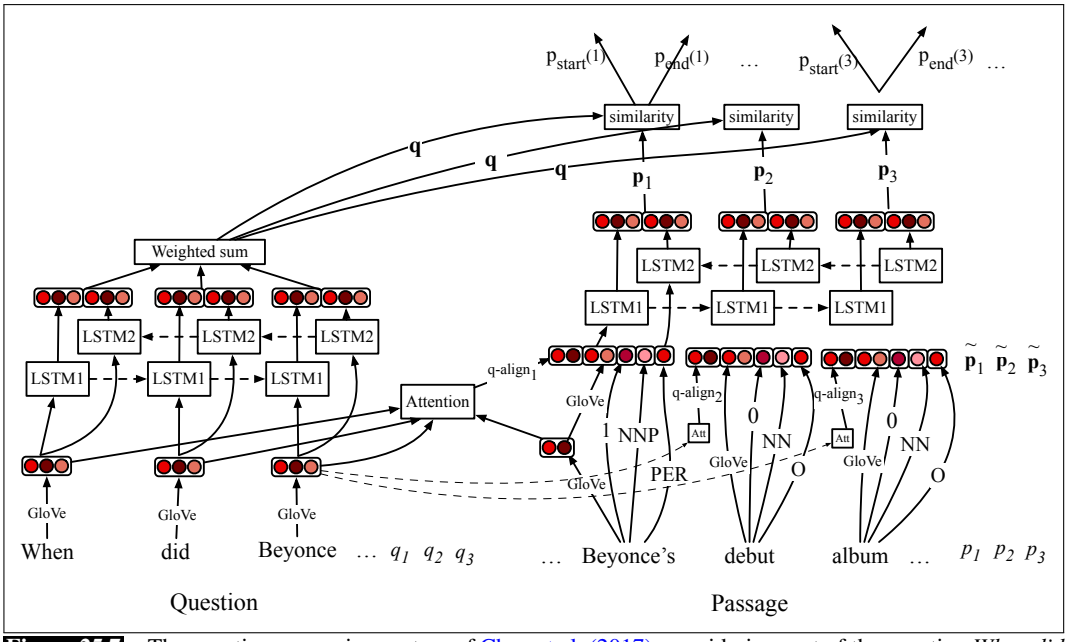


Figure 25.7 The question answering system of Chen et al. (2017), considering part of the question *When did Beyoncé release Dangerously in Love?* and the passage starting *Beyoncé's debut album, Dangerously in Love (2003)*.

To compute the passage embedding $\{\mathbf{p}_1, ..., \mathbf{p}_m\}$ we first form an input representation $\tilde{p} = \{\tilde{\mathbf{p}}_1, ..., \tilde{\mathbf{p}}_m\}$ by concatenating four components:

- An embedding for each word $\mathbf{E}(p_i)$ such as from GLoVE (Pennington et al., 2014).
- Token features like the part of speech of p_i, or the named entity tag of p_i, from running POS or NER taggers.
- Exact match features representing whether the passage word p_i occurred in the question: $\mathbb{1}(p_i \in q)$. Separate exact match features might be used for lemmatized or lower-cased versions of the tokens.
- Aligned question embedding: In addition to the exact match features, many QA systems use an attention mechanism to give a more sophisticated model of similarity between the passage and question words, such as similar but non-identical words like *release* and *singles*. For example a weighted similarity $\sum_j a_{i,j}\mathbf{E}(q_j)$ can be used, where the attention weight $a_{i,j}$ encodes the similarity between p_i and each question word q_j. This attention weight can be computed as the dot product between functions α of the word embeddings of the question and passage:

$$q_{i,j} = \frac{\exp(\alpha(\mathbf{E}(p_i)) \cdot \alpha(\mathbf{E}(q_j)))}{\sum_{j'} \exp(\alpha(\mathbf{E}(p_i)) \cdot \alpha(\mathbf{E}(q'_j)))} \tag{25.11}$$

$\alpha(\cdot)$ can be a simple feed forward network.

We then pass $\tilde{p}$ through a biLSTM:

$$\{\mathbf{p}_1, ..., \mathbf{p}_m\}) = RNN(\{\tilde{\mathbf{p}}_1, ..., \tilde{\mathbf{p}}_m\} \tag{25.12}$$

The result of the previous two steps is a single question embedding $\mathbf{q}$ and a representation for each word in the passage $\{\mathbf{p}_1, ..., \mathbf{p}_m\}$. In order to find the answer

span, we can train two separate classifiers, one to compute for each p_i the probability $p_{start}(i)$ that p_i is the start of the answer span, and one to compute the probability $p_{end}(i)$. While the classifiers could just take the dot product between the passage and question embeddings as input, it turns out to work better to learn a more sophisticated similarity function, like a bilinear attention layer $\mathbf{W}$:

$$p_{start}(i) \propto exp(\mathbf{p}_i \mathbf{W}_s \mathbf{q})$$
$$p_{end}(i) \propto exp(\mathbf{p}_i \mathbf{W}_e \mathbf{q}) \tag{25.13}$$

These neural answer extractors can be trained end-to-end by using datasets like SQuAD.

25.1.10 BERT-based Question Answering

The power of contextual embeddings allow question answering models based on BERT contextual embeddings and the transformer architecture to achieve even higher accuracy (Fig. 25.8).

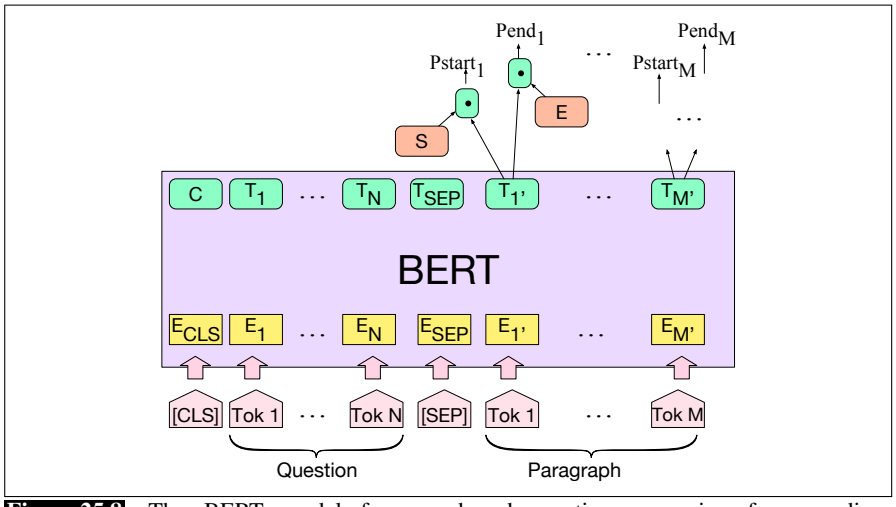


Figure 25.8 The BERT model for span-based question answering from reading-comprehension-based question answering tasks. Figure after Devlin et al. (2019).

Recall from Chapter 10 that BERT represents two input strings as a sequence of wordpiece tokens separated with a [SEP] token. The pre-trained BERT model will produce an output token embedding T_i' for every paragraph token i'. For span-based question answering, we represent the question as the first sequence and, the paragraph as the second sequence. We'll also need to add some structure to the output head that will be trained in the fine-tuning phase. We'll add two new embeddings: a span-start embedding S and a span-end embedding E. To get a span-start probability for each output token T_i', we compute the dot product between S and T_i' and then normalize over all tokens T_i' in the paragraph:

$$Pstart_i = \frac{e^{S \cdot T_i}}{\sum_j e^{S \cdot T_j}} \tag{25.14}$$

We do the analogous thing to compute a span-end probability:

$$Pend_i = \frac{e^{E \cdot T_i}}{\sum_j e^{E \cdot T_j}} \tag{25.15}$$

The score of a candidate span from position i to j is $S \cdot T_i + E \cdot T_j$, and the highest scoring span in which $j \geq i$ is chosen is the model prediction. The training objective for fine-tuning is the sum of the log-likelihoods of the correct start and end positions for each observation.

25.2 Knowledge-based Question Answering

While an enormous amount of information is encoded in the vast amount of text on the web, information obviously also exists in more structured forms. We use the term **knowledge-based question answering** for the idea of answering a natural language question by mapping it to a query over a structured database. Like the text-based paradigm for question answering, this approach dates back to the earliest days of natural language processing, with systems like BASEBALL (Green et al., 1961) that answered questions from a structured database of baseball games and stats.

Systems for mapping from a text string to any logical form are called **semantic parsers**. Semantic parsers for question answering usually map either to some version of predicate calculus or a query language like SQL or SPARQL, as in the examples in Fig. 25.9.

Question	Logical form
When was Ada Lovelace born?	`birth-year (Ada Lovelace, ?x)`
What states border Texas?	λ x.state(x) $\wedge$ borders(x,texas)
What is the largest state	argmax(λx.state(x), λx.size(x))
How many people survived the sinking of the Titanic	`(count (!fb:event.disaster.survivors fb:en.sinking_of_the_titanic))`

Figure 25.9 Sample logical forms produced by a semantic parser for question answering. These range from simple relations like `birth-year`, or relations normalized to databases like Freebase, to full predicate calculus.

The logical form of the question is thus either in the form of a query or can easily be converted into one. The database can be a full relational database, or simpler structured databases like sets of **RDF triples**. Recall from Chapter 18 that an RDF triple is a 3-tuple, a predicate with two arguments, expressing some simple relation or proposition. Popular ontologies like Freebase (Bollacker et al., 2008) or DBpedia (Bizer et al., 2009) have large numbers of triples derived from Wikipedia **infoboxes**, the structured tables associated with certain Wikipedia articles.

The simplest formation of the knowledge-based question answering task is to answer factoid questions that ask about one of the missing arguments in a triple. Consider an RDF triple like the following:

subject	predicate	object
Ada Lovelace	birth-year	1815

This triple can be used to answer text questions like 'When was Ada Lovelace born?' or 'Who was born in 1815?'. Question answering in this paradigm requires mapping from textual strings like "When was ... born" to canonical relations in the knowledge base like *birth-year*. We might sketch this task as:

"When was Ada Lovelace born?" → `birth-year (Ada Lovelace, ?x)`
"What is the capital of England?" → `capital-city(?x, England)`

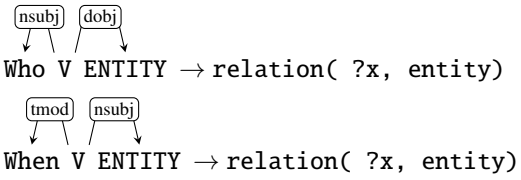
25.2.1 Rule-based Methods

For relations that are very frequent, it may be worthwhile to write handwritten rules to extract relations from the question, just as we saw in Section **??**. For example, to extract the birth-year relation, we could write patterns that search for the question word *When*, a main verb like *born*, and then extract the named entity argument of the verb.

25.2.2 Supervised Methods

In some cases we have supervised data, consisting of a set of questions paired with their correct logical form like the examples in Fig. 25.9. The task is then to take those pairs of training tuples and produce a system that maps from new questions to their logical forms.

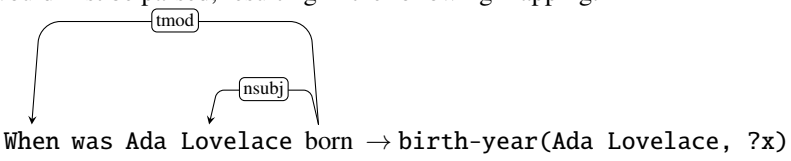
Most supervised algorithms for learning to answer these simple questions about relations first parse the questions and then align the parse trees to the logical form. Generally these systems bootstrap by having a small set of rules for building this mapping, and an initial lexicon as well. For example, a system might have built-in strings for each of the entities in the system (`Texas`, `Ada Lovelace`), and then have simple default rules mapping fragments of the question parse tree to particular relations:



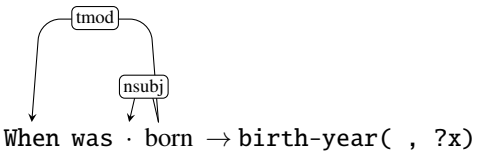
Then given these rules and the lexicon, a training tuple like the following:

"When was Ada Lovelace born?" → `birth-year (Ada Lovelace, ?x)`

would first be parsed, resulting in the following mapping.



From many pairs like this, we could induce mappings between pieces of parse fragment, such as the mapping between the parse fragment on the left and the relation on the right:



A supervised system would thus parse each tuple in the training set and induce a bigger set of such specific rules, allowing it to map unseen examples of "When was

X born?" questions to the `birth-year` relation. Rules can furthermore be associated with counts based on the number of times the rule is used to parse the training data. Like rule counts for probabilistic grammars, these can be normalized into probabilities. The probabilities can then be used to choose the highest probability parse for sentences with multiple semantic interpretations.

The supervised approach can be extended to deal with more complex questions that are not just about single relations. Consider the question *What is the biggest state bordering Texas?* —taken from the GeoQuery database of questions on U.S. Geography (Zelle and Mooney, 1996)—with the semantic form: $\text{argmax}(\lambda x.state(x) \wedge borders(x,texas), \lambda x.size(x))$ This question has much more complex structures than the simple single-relation questions we considered above, such as the argmax function, the mapping of the word *biggest* to *size* and so on. Zettlemoyer and Collins (2005) shows how more complex default rules (along with richer syntactic structures) can be used to learn to map from text sentences to more complex logical forms. The rules take the training set's pairings of sentence and meaning as above and use the complex rules to break each training example down into smaller tuples that can then be recombined to parse new sentences.

25.2.3 Dealing with Variation: Semi-Supervised Methods

Because it is difficult to create training sets with questions labeled with their meaning representation, supervised datasets can't cover the wide variety of forms that even simple factoid questions can take. For this reason most techniques for mapping factoid questions to the canonical relations or other structures in knowledge bases find some way to make use of textual redundancy.

The most common source of redundancy, of course, is the web, which contains vast numbers of textual variants expressing any relation. For this reason, most methods make some use of web text, either via semi-supervised methods like **distant supervision** or unsupervised methods like **open information extraction**, both introduced in Chapter 18. For example the REVERB open information extractor (Fader et al., 2011) extracts billions of (subject, relation, object) triples of strings from the web, such as ("Ada Lovelace","was born in", "1815"). By **aligning** these strings with a canonical knowledge source like Wikipedia, we create new relations that can be queried while simultaneously learning to map between the words in question and canonical relations.

To align a REVERB triple with a canonical knowledge source we first align the arguments and then the predicate. Recall from Chapter 22 that linking a string like "Ada Lovelace" with a Wikipedia page is called **entity linking**; we thus represent the concept 'Ada Lovelace' by a unique identifier of a Wikipedia page. If this subject string is not associated with a unique page on Wikipedia, we can disambiguate which page is being sought, for example by using the cosine distance between the triple string ('Ada Lovelace was born in 1815') and each candidate Wikipedia page. Date strings like '1815' can be turned into a normalized form using standard tools for temporal normalization like SUTime (Chang and Manning, 2012). Once we've aligned the arguments, we align the predicates. Given the Freebase relation `people.person.birthdate(ada lovelace,1815)` and the string 'Ada Lovelace was born in 1815', having linked Ada Lovelace and normalized 1815, we learn the mapping between the string 'was born in' and the relation `people.person.birthdate`. In the simplest case, this can be done by aligning the relation with the string of words in between the arguments; more complex alignment algorithms like IBM Model 1 (Chapter 11) can be used. Then if a phrase aligns with a predicate across many

entities, it can be extracted into a lexicon for mapping questions to relations.

Here are some examples from such a resulting lexicon, produced by Berant et al. (2013), giving many variants of phrases that align with the Freebase relation `country.capital` between a country and its capital city:

capital of	capital city of	become capital of
capitol of	national capital of	official capital of
political capital of	administrative capital of	beautiful capital of
capitol city of	remain capital of	make capital of
political center of	bustling capital of	capital city in
cosmopolitan capital of	move its capital to	modern capital of
federal capital of	beautiful capital city of	administrative capital city of

Figure 25.10 Some phrases that align with the Freebase relation `country.capital` from Berant et al. (2013).

Other useful sources of linguistic redundancy are paraphrase databases. For example the site `wikianswers.com` contains millions of pairs of questions that users have tagged as having the same meaning, 18 million of which have been collected in the PARALEX corpus (Fader et al., 2013). Here's an example:

Q: What are the green blobs in plant cells?

Lemmatized synonyms from PARALEX:

what be the green blob in plant cell?
what be green part in plant cell?
what be the green part of a plant cell?
what be the green substance in plant cell?
what be the part of plant cell that give it green color?
what cell part do plant have that enable the plant to be give a green color?
what part of the plant cell turn it green?
part of the plant cell where the cell get it green color?
the green part in a plant be call?
the part of the plant cell that make the plant green be call?

The resulting millions of pairs of question paraphrases can be aligned to each other using MT alignment approaches to create an MT-style phrase table for translating from question phrases to synonymous phrases. These can be used by question answering algorithms to generate all paraphrases of a question as part of the process of finding an answer (Fader et al. 2013, Berant and Liang 2014).

25.3 Using multiple information sources: IBM's Watson

Of course there is no reason to limit ourselves to just text-based or knowledge-based resources for question answering. The Watson system from IBM that won the Jeopardy! challenge in 2011 is an example of a system that relies on a wide variety of resources to answer questions.

Figure 25.11 shows the 4 stages of the DeepQA system that is the question answering component of Watson.

The first stage is **question processing**. The DeepQA system runs parsing, named entity tagging, and relation extraction on the question. Then, like the text-based systems in Section 25.1, the DeepQA system extracts the **focus**, the **answer type**

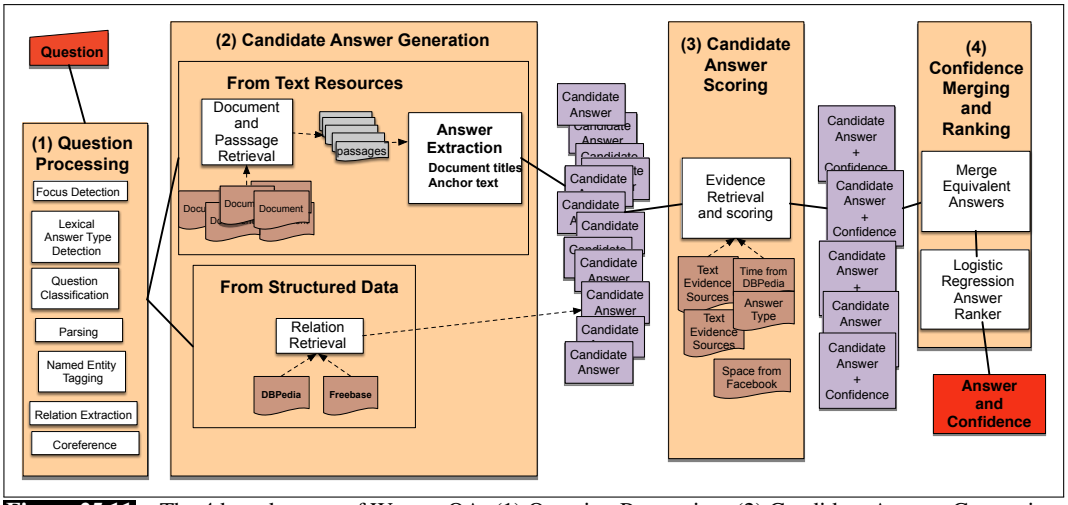


Figure 25.11 The 4 broad stages of Watson QA: (1) Question Processing, (2) Candidate Answer Generation, (3) Candidate Answer Scoring, and (4) Answer Merging and Confidence Scoring.

(also called the **lexical answer type** or **LAT**), and performs **question classification** and **question sectioning**.

Consider these Jeopardy! examples, with a category followed by a question:

> Poets and Poetry: **He** was a bank clerk in the Yukon before he published "Songs of a Sourdough" in 1907.
> THEATRE: A new play based on **this Sir Arthur Conan Doyle canine classic** opened on the London stage in 2007.

The questions are parsed, named entities are extracted (*Sir Arthur Conan Doyle* identified as a PERSON, Yukon as a GEOPOLITICAL ENTITY, "Songs of a Sourdough" as a COMPOSITION), coreference is run (*he* is linked with *clerk*) and relations like the following are extracted:

> authorof(focus,"Songs of a sourdough")
> publish (e1, he, "Songs of a sourdough")
> in (e2, e1, 1907)
> temporallink(publish(...), 1907)

focus

Next DeepQA extracts the question **focus**, shown in bold in both examples. The focus is the part of the question that co-refers with the answer, used for example to align with a supporting passage. The focus is extracted by handwritten rules—made possible by the relatively stylized syntax of Jeopardy! questions—such as a rule extracting any noun phrase with determiner "this" as in the Conan Doyle example, and rules extracting pronouns like *she*, *he*, *hers*, *him*, as in the poet example.

lexical answer type

The **lexical answer type** (shown in blue above) is a word or words which tell us something about the semantic type of the answer. Because of the wide variety of questions in Jeopardy!, Jeopardy! uses a far larger set of answer types than the sets for standard factoid algorithms like the one shown in Fig. 25.4. Even a large set of named entity tags is insufficient to define a set of answer types. The DeepQA team investigated a set of 20,000 questions and found that a named entity tagger with over 100 named entity types covered less than half the types in these questions. Thus DeepQA extracts a wide variety of words to be answer types; roughly 5,000 lexical answer types occurred in the 20,000 questions they investigated, often with multiple answer types in each question.

These lexical answer types are again extracted by rules: the default rule is to choose the syntactic headword of the focus. Other rules improve this default choice. For example additional lexical answer types can be words in the question that are coreferent with or have a particular syntactic relation with the focus, such as headwords of appositives or predicative nominatives of the focus. In some cases even the Jeopardy! category can act as a lexical answer type, if it refers to a type of entity that is compatible with the other lexical answer types. Thus in the first case above, *he*, *poet*, and *clerk* are all lexical answer types. In addition to using the rules directly as a classifier, they can instead be used as features in a logistic regression classifier that can return a probability as well as a lexical answer type.

Note that answer types function quite differently in DeepQA than the purely IR-based factoid question answerers. In the algorithm described in Section 25.1, we determine the answer type, and then use a strict filtering algorithm only considering text strings that have exactly that type. In DeepQA, by contrast, we extract lots of answers, unconstrained by answer type, and a set of answer types, and then in the later 'candidate answer scoring' phase, we simply score how well each answer fits the answer types as one of many sources of evidence.

Finally the question is classified by type (definition question, multiple-choice, puzzle, fill-in-the-blank). This is generally done by writing pattern-matching regular expressions over words or parse trees.

In the second **candidate answer generation** stage, we combine the processed question with external documents and other knowledge sources to suggest many candidate answers. These candidate answers can either be extracted from text documents or from structured knowledge bases.

For structured resources like DBpedia, IMDB, or the triples produced by Open Information Extraction, we can just query these stores with the relation and the known entity, just as we saw in Section 25.2. Thus if we have extracted the relation `authorof(focus,"Songs of a sourdough")`, we can query a triple store with `authorof(?x,"Songs of a sourdough")` to return the correct author.

The method for extracting answers from text depends on the type of text documents. To extract answers from normal text documents we can do passage search just as we did in Section 25.1. As we did in that section, we need to generate a query from the question; for DeepQA this is generally done by eliminating stop words, and then upweighting any terms which occur in any relation with the focus. For example from this query:

> MOVIE-"ING": Robert Redford and Paul Newman starred in this depression-era grifter flick. *(Answer: "The Sting")*

the following weighted query might be extracted:

> (2.0 Robert Redford) (2.0 Paul Newman) star depression era grifter (1.5 flick)

The query can now be passed to a standard IR system. DeepQA also makes use of the convenient fact that the vast majority of Jeopardy! answers are the title of a Wikipedia document. To find these titles, we can do a second text retrieval pass specifically on Wikipedia documents. Then instead of extracting passages from the retrieved Wikipedia document, we directly return the titles of the highly ranked retrieved documents as the possible answers.

Once we have a set of passages, we need to extract candidate answers. If the document happens to be a Wikipedia page, we can just take the title, but for other texts, like news documents, we need other approaches. Two common approaches are to extract all **anchor texts** in the document (anchor text is the text between <a>

anchor texts

and used to point to a URL in an HTML page), or to extract all noun phrases in the passage that are Wikipedia document titles.

The third **candidate answer scoring** stage uses many sources of evidence to score the candidates. One of the most important is the lexical answer type. DeepQA includes a system that takes a candidate answer and a lexical answer type and returns a score indicating whether the candidate answer can be interpreted as a subclass or instance of the answer type. Consider the candidate "difficulty swallowing" and the lexical answer type "manifestation". DeepQA first matches each of these words with possible entities in ontologies like DBpedia and WordNet. Thus the candidate "difficulty swallowing" is matched with the DBpedia entity "Dysphagia", and then that instance is mapped to the WordNet type "Symptom". The answer type "manifestation" is mapped to the WordNet type "Condition". The system looks for a link of hyponymy, instance-of or synonymy between these two types; in this case a hyponymy relation is found between "Symptom" and "Condition".

Other scorers are based on using time and space relations extracted from DBpedia or other structured databases. For example, we can extract temporal properties of the entity (when was a person born, when died) and then compare to time expressions in the question. If a time expression in the question occurs chronologically before a person was born, that would be evidence against this person being the answer to the question.

Finally, we can use text retrieval to help retrieve evidence supporting a candidate answer. We can retrieve passages with terms matching the question, then replace the focus in the question with the candidate answer and measure the overlapping words or ordering of the passage with the modified question.

The output of this stage is a set of candidate answers, each with a vector of scoring features.

The final **answer merging and scoring** step first merges candidate answers that are equivalent. Thus if we had extracted two candidate answers *J.F.K.* and *John F. Kennedy*, this stage would merge the two into a single candidate. Synonym dictionaries are a useful resource that are created by listing all anchor text strings that point to the same Wikipedia page; such dictionaries give large numbers of synonyms for each Wikipedia title — e.g., *JFK, John F. Kennedy, John Fitzgerald Kennedy, Senator John F. Kennedy, President Kennedy, Jack Kennedy*, etc. (Spitkovsky and Chang, 2012). For common nouns, we can use morphological parsing to merge candidates which are morphological variants.

We then merge the evidence for each variant, combining the scoring feature vectors for the merged candidates into a single vector.

Now we have a set of candidates, each with a feature vector. A classifier takes each feature vector and assigns a confidence value to this candidate answer. The classifier is trained on thousands of candidate answers, each labeled for whether it is correct or incorrect, together with their feature vectors, and learns to predict a probability of being a correct answer. Since, in training, there are far more incorrect answers than correct answers, we need to use one of the standard techniques for dealing with very imbalanced data. DeepQA uses *instance weighting*, assigning an instance weight of .5 for each incorrect answer example in training. The candidate answers are then sorted by this confidence value, resulting in a single best answer.[3]

In summary, we've seen in the four stages of DeepQA that it draws on the intu-

[3] The merging and ranking is actually run iteratively; first the candidates are ranked by the classifier, giving a rough first value for each candidate answer, then that value is used to decide which of the variants of a name to select as the merged answer, then the merged answers are re-ranked.

itions of both the IR-based and knowledge-based paradigms. Indeed, Watson's architectural innovation is its reliance on proposing a very large number of candidate answers from both text-based and knowledge-based sources and then developing a wide variety of evidence features for scoring these candidates—again both text-based and knowledge-based. See the papers mentioned at the end of the chapter for more details.

25.4 Evaluation of Factoid Answers

mean reciprocal rank
MRR

A common evaluation metric for factoid question answering, introduced in the TREC Q/A track in 1999, is **mean reciprocal rank**, or **MRR**. MRR assumes a test set of questions that have been human-labeled with correct answers. MRR also assumes that systems are returning a short **ranked** list of answers or passages containing answers. Each question is then scored according to the reciprocal of the **rank** of the first correct answer. For example if the system returned five answers but the first three are wrong and hence the highest-ranked correct answer is ranked fourth, the reciprocal rank score for that question would be $\frac{1}{4}$. Questions with return sets that do not contain any correct answers are assigned a zero. The score of a system is then the average of the score for each question in the set. More formally, for an evaluation of a system returning a set of ranked answers for a test set consisting of N questions, the MRR is defined as

$$\text{MRR} = \frac{1}{N}\sum_{i=1 \text{ s.t. } rank_i \neq 0}^{N}\frac{1}{rank_i} \tag{25.16}$$

Reading comprehension systems on datasets like SQuAD are often evaluated using two metrics, both ignoring punctuation and articles (*a, an, the*) (Rajpurkar et al., 2016):

- Exact match: The percentage of predicted answers that match the gold answer exactly.
- F_1 score: The average overlap between predicted and gold answers. Treat the prediction and gold as a bag of tokens, and compute F_1, averaging the F_1 over all questions.

A number of test sets are available for question answering. Early systems used the TREC QA dataset; questions and handwritten answers for TREC competitions from 1999 to 2004 are publicly available. TriviaQA (Joshi et al., 2017) has 650K question-answer evidence triples, from 95K hand-created question-answer pairs together with on average six supporting evidence documents collected retrospectively from Wikipedia and the Web.

Another family of datasets starts from WEBQUESTIONS (Berant et al., 2013), which contains 5,810 questions asked by web users, each beginning with a wh-word and containing exactly one entity. Questions are paired with handwritten answers drawn from the Freebase page of the question's entity. WEBQUESTIONSSP (Yih et al., 2016) augments WEBQUESTIONS with human-created semantic parses (SPARQL queries) for those questions answerable using Freebase. COMPLEXWEBQUESTIONS augments the dataset with compositional and other kinds of complex questions, resulting in 34,689 questions, along with answers, web snippets, and SPARQL queries. (Talmor and Berant, 2018).

There are a wide variety of datasets for training and testing reading comprehension/answer extraction in addition to the SQuAD (Rajpurkar et al., 2016) and WikiQA (Yang et al., 2015) datasets discussed on page 8. The NarrativeQA (Kočiský et al., 2018) dataset, for example, has questions based on entire long documents like books or movie scripts, while the Question Answering in Context (QuAC) dataset (Choi et al., 2018) has 100K questions created by two crowd workers who are asking and answering questions about a hidden Wikipedia text.

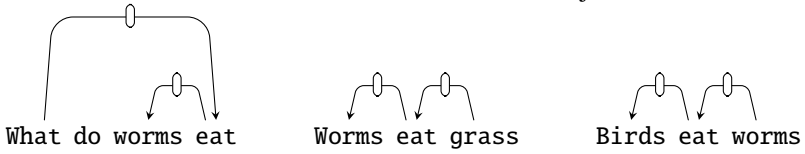
Others take their structure from the fact that reading comprehension tasks designed for children tend to be multiple choice, with the task being to choose among the given answers. The MCTest dataset uses this structure, with 500 fictional short stories created by crowd workers with questions and multiple choice answers (Richardson et al., 2013). The AI2 Reasoning Challenge (ARC) (Clark et al., 2018), has questions that are designed to be hard to answer from simple lexical methods:

> Which property of a mineral can be determined just by looking at it?
> (A) luster [correct] (B) mass (C) weight (D) hardness

This ARC example is difficult because the correct answer *luster* is unlikely to cooccur frequently on the web with phrases like *looking at it*, while the word *mineral* is highly associated with the incorrect answer *hardness*.

Bibliographical and Historical Notes

Question answering was one of the earliest NLP tasks, and early versions of the text-based and knowledge-based paradigms were developed by the very early 1960s. The text-based algorithms generally relied on simple parsing of the question and of the sentences in the document, and then looking for matches. This approach was used very early on (Phillips, 1960) but perhaps the most complete early system, and one that strikingly prefigures modern relation-based systems, was the Protosynthex system of Simmons et al. (1964). Given a question, Protosynthex first formed a query from the content words in the question, and then retrieved candidate answer sentences in the document, ranked by their frequency-weighted term overlap with the question. The query and each retrieved sentence were then parsed with dependency parsers, and the sentence whose structure best matches the question structure selected. Thus the question *What do worms eat?* would match *worms eat grass*: both have the subject *worms* as a dependent of *eat*, in the version of dependency grammar used at the time, while *birds eat worms* has *birds* as the subject:



The alternative knowledge-based paradigm was implemented in the BASEBALL system (Green et al., 1961). This system answered questions about baseball games like "Where did the Red Sox play on July 7" by querying a structured database of game information. The database was stored as a kind of attribute-value matrix with values for attributes of each game:

```
Month = July
   Place = Boston
```

```
Day  = 7
Game Serial No.  = 96
(Team = Red Sox, Score = 5)
(Team = Yankees, Score = 3)
```

Each question was constituency-parsed using the algorithm of Zellig Harris's TDAP project at the University of Pennsylvania, essentially a cascade of finite-state transducers (see the historical discussion in Joshi and Hopely 1999 and Karttunen 1999). Then in a content analysis phase each word or phrase was associated with a program that computed parts of its meaning. Thus the phrase 'Where' had code to assign the semantics `Place = ?`, with the result that the question "Where did the Red Sox play on July 7" was assigned the meaning

```
Place = ?
Team = Red Sox
Month = July
Day = 7
```

The question is then matched against the database to return the answer. Simmons (1965) summarizes other early QA systems.

Another important progenitor of the knowledge-based paradigm for question-answering is work that used predicate calculus as the meaning representation language. The **LUNAR** system (Woods et al. 1972, Woods 1978) was designed to be a natural language interface to a database of chemical facts about lunar geology. It could answer questions like *Do any samples have greater than 13 percent aluminum* by parsing them into a logical form

LUNAR

(TEST (FOR SOME X16 / (SEQ SAMPLES) : T ; (CONTAIN' X16
(NPR* X17 / (QUOTE AL203)) (GREATERTHAN 13 PCT))))

The rise of the web brought the information-retrieval paradigm for question answering to the forefront with the TREC QA track beginning in 1999, leading to a wide variety of factoid and non-factoid systems competing in annual evaluations.

At the same time, Hirschman et al. (1999) introduced the idea of using children's reading comprehension tests to evaluate machine text comprehension algorithms. They acquired a corpus of 120 passages with 5 questions each designed for 3rd-6th grade children, built an answer extraction system, and measured how well the answers given by their system corresponded to the answer key from the test's publisher. Their algorithm focused on word overlap as a feature; later algorithms added named entity features and more complex similarity between the question and the answer span (Riloff and Thelen 2000, Ng et al. 2000).

Neural reading comprehension systems drew on the insight of these early systems that answer finding should focus on question-passage similarity. Many of the architectural outlines of modern systems were laid out in the AttentiveReader (Hermann et al., 2015). The idea of using passage-aligned question embeddings in the passage computation was introduced by Lee et al. (2017). Seo et al. (2017) achieves high performance by introducing bi-directional attention flow. Chen et al. (2017) and Clark and Gardner (2018) show how to extract answers from entire documents.

The DeepQA component of the Watson system that won the Jeopardy! challenge is described in a series of papers in volume 56 of the IBM Journal of Research and Development; see for example Ferrucci (2012), Lally et al. (2012), Chu-Carroll et al. (2012), Murdock et al. (2012b), Murdock et al. (2012a), Kalyanpur et al. (2012), and Gondek et al. (2012).

Other question-answering tasks include Quiz Bowl, which has timing considerations since the question can be interrupted (Boyd-Graber et al., 2018). Question answering is also an important function of modern personal assistant dialog systems; see Chapter 26 for more.

Exercises

Berant, J., Chou, A., Frostig, R., and Liang, P. (2013). Semantic parsing on freebase from question-answer pairs. In *EMNLP 2013*.

Berant, J. and Liang, P. (2014). Semantic parsing via paraphrasing. In *ACL 2014*.

Bizer, C., Lehmann, J., Kobilarov, G., Auer, S., Becker, C., Cyganiak, R., and Hellmann, S. (2009). DBpedia—A crystallization point for the Web of Data. *Web Semantics: science, services and agents on the world wide web*, *7*(3), 154–165.

Bollacker, K., Evans, C., Paritosh, P., Sturge, T., and Taylor, J. (2008). Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD 2008*, 1247–1250.

Boyd-Graber, J., Feng, S., and Rodriguez, P. (2018). Human-computer question answering: The case for quizbowl. In Escalera, S. and Weimer, M. (Eds.), *The NIPS '17 Competition: Building Intelligent Systems*. Springer.

Brill, E., Dumais, S. T., and Banko, M. (2002). An analysis of the AskMSR question-answering system. In *EMNLP 2002*, 257–264.

Chang, A. X. and Manning, C. D. (2012). SUTime: A library for recognizing and normalizing time expressions.. In *LREC-12*, 3735–3740.

Chen, D., Fisch, A., Weston, J., and Bordes, A. (2017). Reading wikipedia to answer open-domain questions. In *ACL 2017*.

Choi, E., He, H., Iyyer, M., Yatskar, M., Yih, W.-t., Choi, Y., Liang, P., and Zettlemoyer, L. (2018). Quac: Question answering in context. In *EMNLP 2018*.

Chu-Carroll, J., Fan, J., Boguraev, B. K., Carmel, D., Sheinwald, D., and Welty, C. (2012). Finding needles in the haystack: Search and candidate generation. *IBM Journal of Research and Development*, *56*(3/4), 6:1–6:12.

Clark, C. and Gardner, M. (2018). Simple and effective multi-paragraph reading comprehension. In *ACL 2018*.

Clark, P., Cowhey, I., Etzioni, O., Khot, T., Sabharwal, A., Schoenick, C., and Tafjord, O. (2018). Think you have solved question answering? Try ARC, the AI2 reasoning challenge.. arXiv preprint arXiv:1803.05457.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL HLT 2019*, 4171–4186.

Fader, A., Soderland, S., and Etzioni, O. (2011). Identifying relations for open information extraction. In *EMNLP-11*, 1535–1545.

Fader, A., Zettlemoyer, L., and Etzioni, O. (2013). Paraphrase-driven learning for open question answering. In *ACL 2013*, 1608–1618.

Ferrucci, D. A. (2012). Introduction to "This is Watson". *IBM Journal of Research and Development*, *56*(3/4), 1:1–1:15.

Gondek, D. C., Lally, A., Kalyanpur, A., Murdock, J. W., Duboué, P. A., Zhang, L., Pan, Y., Qiu, Z. M., and Welty, C. (2012). A framework for merging and ranking of answers in deepqa. *IBM Journal of Research and Development*, *56*(3/4), 14:1–14:12.

Green, B. F., Wolf, A. K., Chomsky, C., and Laughery, K. (1961). Baseball: An automatic question answerer. In *Proceedings of the Western Joint Computer Conference 19*, 219–224. Reprinted in Grosz et al. (1986).

Harabagiu, S., Pasca, M., and Maiorano, S. (2000). Experiments with open-domain textual question answering. In *COLING-00*.

Hermann, K. M., Kocisky, T., Grefenstette, E., Espeholt, L., Kay, W., Suleyman, M., and Blunsom, P. (2015). Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, 1693–1701.

Hirschman, L., Light, M., Breck, E., and Burger, J. D. (1999). Deep Read: A reading comprehension system. In *ACL-99*, 325–332.

Hovy, E. H., Hermjakob, U., and Ravichandran, D. (2002). A question/answer typology with surface text patterns. In *HLT-01*.

Joshi, A. K. and Hopely, P. (1999). A parser from antiquity. In Kornai, A. (Ed.), *Extended Finite State Models of Language*, 6–15. Cambridge University Press.

Joshi, M., Choi, E., Weld, D. S., and Zettlemoyer, L. (2017). Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *ACL 2017*.

Kalyanpur, A., Boguraev, B. K., Patwardhan, S., Murdock, J. W., Lally, A., Welty, C., Prager, J. M., Coppola, B., Fokoue-Nkoutche, A., Zhang, L., Pan, Y., and Qiu, Z. M. (2012). Structured data and inference in deepqa. *IBM Journal of Research and Development*, *56*(3/4), 10:1–10:14.

Karttunen, L. (1999). Comments on Joshi. In Kornai, A. (Ed.), *Extended Finite State Models of Language*, 16–18. Cambridge University Press.

Kočiský, T., Schwarz, J., Blunsom, P., Dyer, C., Hermann, K. M., Melis, G., and Grefenstette, E. (2018). The NarrativeQA reading comprehension challenge. *TACL*, *6*, 317–328.

Lally, A., Prager, J. M., McCord, M. C., Boguraev, B. K., Patwardhan, S., Fan, J., Fodor, P., and Chu-Carroll, J. (2012). Question analysis: How Watson reads a clue. *IBM Journal of Research and Development*, *56*(3/4), 2:1–2:14.

Lee, K., Salant, S., Kwiatkowski, T., Parikh, A., Das, D., and Berant, J. (2017). Learning recurrent span representations for extractive question answering. In *arXiv 1611.01436*.

Li, X. and Roth, D. (2002). Learning question classifiers. In *COLING-02*, 556–562.

Li, X. and Roth, D. (2005). Learning question classifiers: The role of semantic information. *Journal of Natural Language Engineering*, *11*(4).

Lin, J. (2007). An exploration of the principles underlying redundancy-based factoid question answering. *ACM Transactions on Information Systems*, *25*(2).

Monz, C. (2004). Minimal span weighting retrieval for question answering. In *SIGIR Workshop on Information Retrieval for Question Answering*, 23–30.

Murdock, J. W., Fan, J., Lally, A., Shima, H., and Boguraev, B. K. (2012a). Textual evidence gathering and analysis. *IBM Journal of Research and Development*, *56*(3/4), 8:1–8:14.

Murdock, J. W., Kalyanpur, A., Welty, C., Fan, J., Ferrucci, D. A., Gondek, D. C., Zhang, L., and Kanayama, H. (2012b). Typing candidate answers using type coercion. *IBM Journal of Research and Development*, *56*(3/4), 7:1–7:13.

Ng, H. T., Teo, L. H., and Kwan, J. L. P. (2000). A machine learning approach to answering questions for reading comprehension tests. In *EMNLP 2000*, 124–132.

Pasca, M. (2003). *Open-Domain Question Answering from Large Text Collections*. CSLI.

Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *EMNLP 2014*, 1532–1543.

Phillips, A. V. (1960). A question-answering routine. Tech. rep. 16, MIT AI Lab.

Rajpurkar, P., Jia, R., and Liang, P. (2018). Know what you don't know: Unanswerable questions for SQuAD. In *ACL 2018*.

Rajpurkar, P., Zhang, J., Lopyrev, K., and Liang, P. (2016). SQuAD: 100,000+ questions for machine comprehension of text. In *EMNLP 2016*.

Richardson, M., Burges, C. J. C., and Renshaw, E. (2013). MCTest: A challenge dataset for the open-domain machine comprehension of text. In *EMNLP 2013*, 193–203.

Riloff, E. and Thelen, M. (2000). A rule-based question answering system for reading comprehension tests. In *Proceedings of ANLP/NAACL workshop on reading comprehension tests*, 13–19.

Seo, M., Kembhavi, A., Farhadi, A., and Hajishirzi, H. (2017). Bidirectional attention flow for machine comprehension. In *ICLR 2017*.

Simmons, R. F. (1965). Answering English questions by computer: A survey. *CACM*, *8*(1), 53–70.

Simmons, R. F., Klein, S., and McConlogue, K. (1964). Indexing and dependency logic for answering english questions. *American Documentation*, *15*(3), 196–204.

Spitkovsky, V. I. and Chang, A. X. (2012). A cross-lingual dictionary for English Wikipedia concepts. In *LREC-12*.

Talmor, A. and Berant, J. (2018). The web as a knowledge-base for answering complex questions. In *NAACL HLT 2018*.

Woods, W. A. (1978). Semantics and quantification in natural language question answering. In Yovits, M. (Ed.), *Advances in Computers*, 2–64. Academic.

Woods, W. A., Kaplan, R. M., and Nash-Webber, B. L. (1972). The lunar sciences natural language information system: Final report. Tech. rep. 2378, BBN.

Yang, Y., Yih, W.-t., and Meek, C. (2015). Wikiqa: A challenge dataset for open-domain question answering. In *EMNLP 2015*.

Yih, W.-t., Richardson, M., Meek, C., Chang, M.-W., and Suh, J. (2016). The value of semantic parse labeling for knowledge base question answering. In *ACL 2016*, 201–206.

Zelle, J. M. and Mooney, R. J. (1996). Learning to parse database queries using inductive logic programming. In *AAAI-96*, 1050–1055.

Zettlemoyer, L. and Collins, M. (2005). Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Uncertainty in Artificial Intelligence, UAI'05*, 658–666.