Question Answering

Reference:
- D. Jurafsky and J. Martin, “Speech and Language Processing”

Introduction

- By early 60s, two major paradigms of question answering (QA)
  - Information retrieval (IR) based
  - Knowledge based
- In 2011, IBM's Watson QA system won the TV game-show Jeopardy
  - Surpassing humans at answering questions like:

  "William Wilkinson's "An Account of the Principalities of Wallachia and Moldovia" inspired this author's most famous novel."

Introduction

- Factoid questions – questions that can be answered with simple facts expressed in short texts such as:

  "Where is the Louvre Museum located?"

- Some paradigms for factoid question answering:
  - IR based QA (also called open domain question QA)
  - Knowledge-based QA
  - Pretrained language model based QA
- We focus on factoid QA, but there are other QA tasks such as long-form QA (long answers), and community QA

IR with Dense Vectors

- The traditional IR algorithms have known to have a limitation: they work only if there is exact overlap of words between the query and document
- Vocabulary mismatch problem
  - E.g. User might decide to search for “a tragic love story”, but Shakespeare writes instead about “star-crossed lovers”
- The solution is to use an approach that can handle synonymy
- Instead of sparse word-count vectors, we use dense embedding
- Modern methods all make use of encoders like BERT

IR-based Factoid Question Answering

- The goal of IR-based QA is to answer a question by finding short text segments from a text collection
  - E.g. In figure 23.9, some sample factoid questions and their answers are listed

Question	Answer
Where is the Louvre Museum located?	in Paris, France
What are the names of Odin's ravens?	Huginn and Muninn
What kind of nuts are used in marzipan?	almonds
What instrument did Max Roach play?	drums
What's the official language of Algeria?	Arabic

Figure 23.9 Some factoid questions and their answers.

IR-based Factoid Question Answering

- The dominant paradigm for IR-based QA is the retrieve and read model
  - In figure 23.10, IR-based QA has two stages: retrieve relevant passages from a text collection, and reading passes over each passage and finds spans that are likely to answer the question

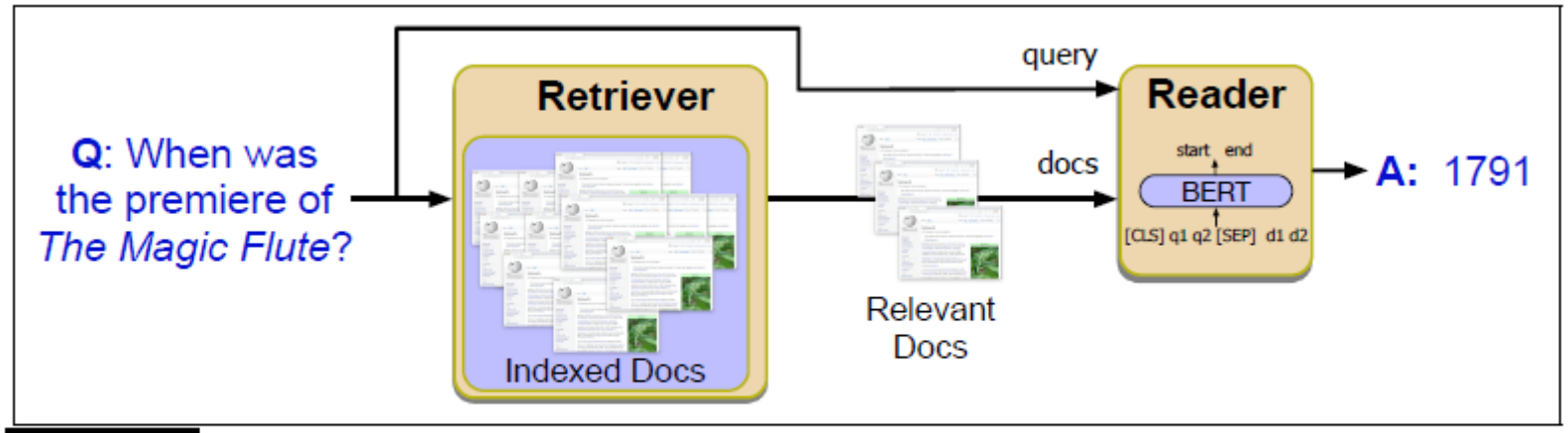

Figure 23.10 IR-based factoid question answering has two stages: **retrieval**, which returns relevant documents from the collection, and **reading**, in which a neural reading comprehension system extracts answer spans.

IR-based QA: Reader

- The reader's job is to take a passage as input and produce s span of text in the passage as the answer
  - E.g. A question like "How tall is Mt. Everest?", and a passage that contains the clause Reaching 29,029 feet, the reader will output 29,029 feet
- The answer extraction task is commonly modelled by span labelling: identifying a span in the passage that constitutes an answer
  - Given a question q of m tokens $q_1, \dots, q_m$ and a passage p of n tokens $p_1, \dots, p_m$, the job is to compute the probability $q_1, \dots, q_m$ that each possible span
  - If each span a starts at position a_s and ends at position a_e, the probability can be estimated as

$$\mathrm{P}(a|q,p) = P_{\text{start}}(a_s|\mathrm{q},p)P_{end}(a_e|q,p)$$

IR-based QA: Reader

- A standard baseline algorithm for Reader is to pass the question q_i and passage p_i to any encoder like BERT

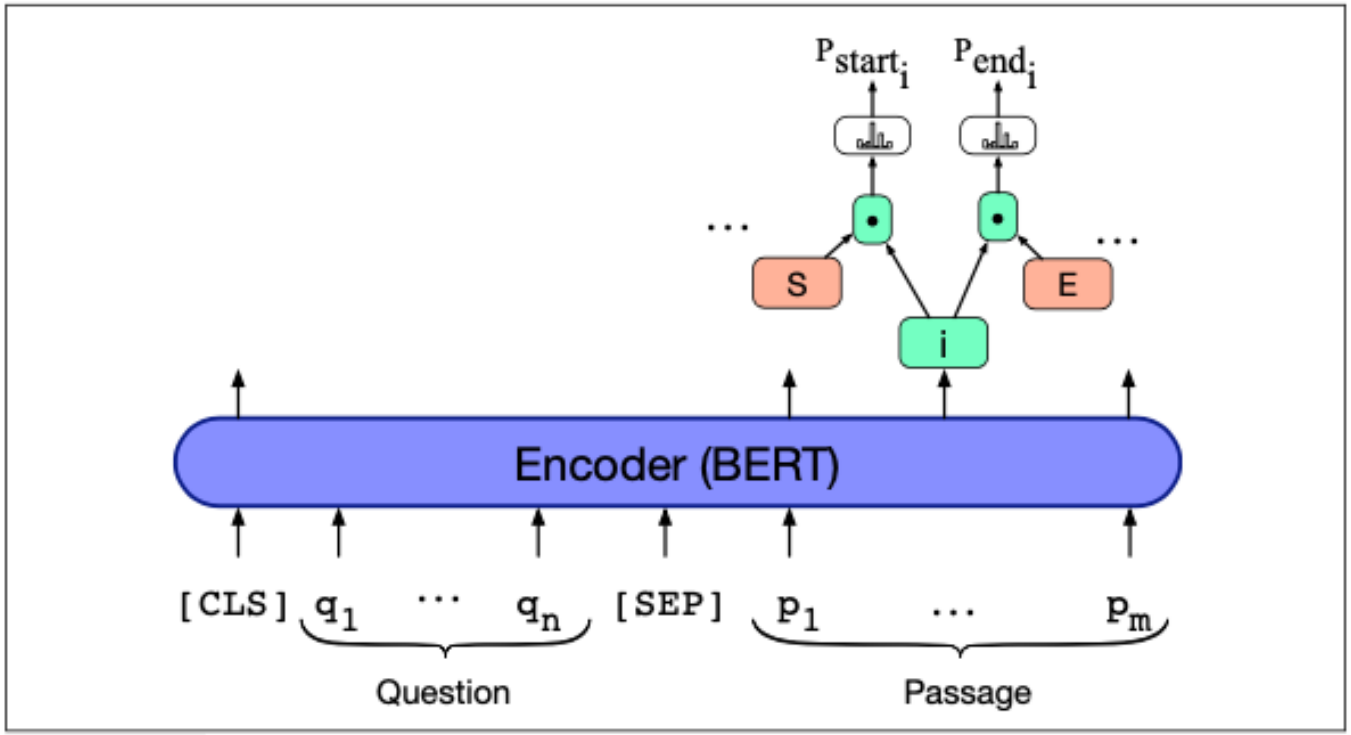

Figure 23.12 An encoder model (using BERT) for span-based question answering from reading-comprehension-based question answering tasks.

IR-based QA: Reader

- Represent the question q_i and passage p_i as the sequence
- Add a linear layer to predict the start and end position of the span
  - A span-start embedding s and a span-end embedding E will be learned in fine-tuning
  - A span-start probability for each output token p_i'

$$P_{start_i} = \frac{\exp(S \cdot p_i')}{\Sigma_j \exp(\mathrm{S} \cdot p_j')}$$

  - A span-end probability $P_{end_i} = \frac{\exp(\mathrm{E} \cdot p_i')}{\Sigma_j \exp(\mathrm{E} \cdot p_j')}$
  - The score of a candidate span from position i to j is $s \cdot p_i' + E \cdot p_j'$
- The training loss is the negative sum of the log-likelihoods of the correct start and end positions for each instance

$$L = -\log P_{start_j} - \log P_{end_j}$$

Entity Linking

- Entity linking is the task of associating a mention in text with the representation of some real-world entity in an ontology
  - The most common ontology for factoid question-answering is Wikipedia
- There are two algorithms
  - One baseline is to use anchor dictionaries and information from the Wikipedia graph structure
  - Another one is modern neural algorithms

Entity based on Anchor Dictionaries and Web Graph

- A classic baseline is the TAGME algorithm, using anchor dictionaries and information from the Wikipedia graph structure
- The TAGME algorithm defines the set of entities as the set of Wikipedia pages
  - Each Wikipedia page as a unique entity e
  - The total number of in-links $in(e)$ from other Wikipedia pages that point to e
  - An anchor dictionary lists its anchor texts a for each Wikipedia page e

The TAGME algorithm

- The first stage is entity mention detection
  - Given a question, TAGME detections by querying the anchor dictionary for each token; the large set of sequences is pruned with some simple heuristics
  - E.g. A question like “When was Ada Lovelace born?”, might cause the anchor Ada Lovelace or Ada, but substrings spans like Lovelace might be pruned as having too low a linkprob
- The second stage is entity mention disambiguation
  - TAGME uses prior probability and relatedness/coherence for disambiguating ambiguous spans

The TAGME algorithm – mention disambiguation

- The first factor is prior probability $p(e|a)$
  - The probability that anchor a points to each page $e \in \varepsilon(a)$, is ratio of the number of links into e with anchor text a to the total number of occurrences of a as an anchor

$$prior(a \rightarrow e) = p(e|a) = \frac{count(a \rightarrow e)}{link(a)}$$

The TAGME algorithm – mention disambiguation

- The second factor is relatedness of the entity e to all other entities in the input question q
  - The relatedness score of the anchor $a \rightarrow \mathrm{X}$ is to combine relatedness and prior by choosing the entity X that has the highest relatedness$(a \rightarrow X)$, and from the set, choosing the entity with the highest prior $p(x|a)$

$$\text{relatedness}(a \rightarrow X) = \sum_{b \in \chi_q \setminus a} vote(b, X)$$

$$\text{vote}(b, X) = \frac{1}{|\varepsilon(b)|} \sum_{Y \in \varepsilon(b)} rel(X, Y) p(Y|b)$$

$$rel(\mathrm{X}, Y) = \frac{\log\big(max(|in(X)|, |in(Y)|)\big) - \log(|in(X) \cap in(Y)|)}{\log(|W|) - \log\big(min(|in(X)|, |\mathrm{in}(Y)|)\big)}$$

Where $in(X)$ is the set of Wikipedia pages pointing to x and W is the set of all Wikipedia pages in the collection

The TAGME algorithm – mention disambiguation

- The TAGME algorithm has one further step to prune spurious anchor/entity pairs, assigning a score averaging link probability with the coherence

$$coherence(a \to X) = \frac{1}{|S| - 1} \sum_{B \in \mathcal{S} \setminus X} rel(B, X)$$

$$score(a \to X) = \frac{coherence(a \to X) + linkprob(a)}{2}$$

Finally, pairs are pruned if $score(a \to X) < \lambda$, where the threshold λ is set on a held-out set

Neural Graph-based linking

- More recent entity linking models are based on biencoders, encoding a candidate mention span, encoding an entity, and computing the dot product between the encodings
  - In figure 23.13, ELQ linking algorithm encodes each Wikipedia entity, each mention span from the question, and computes their similarity

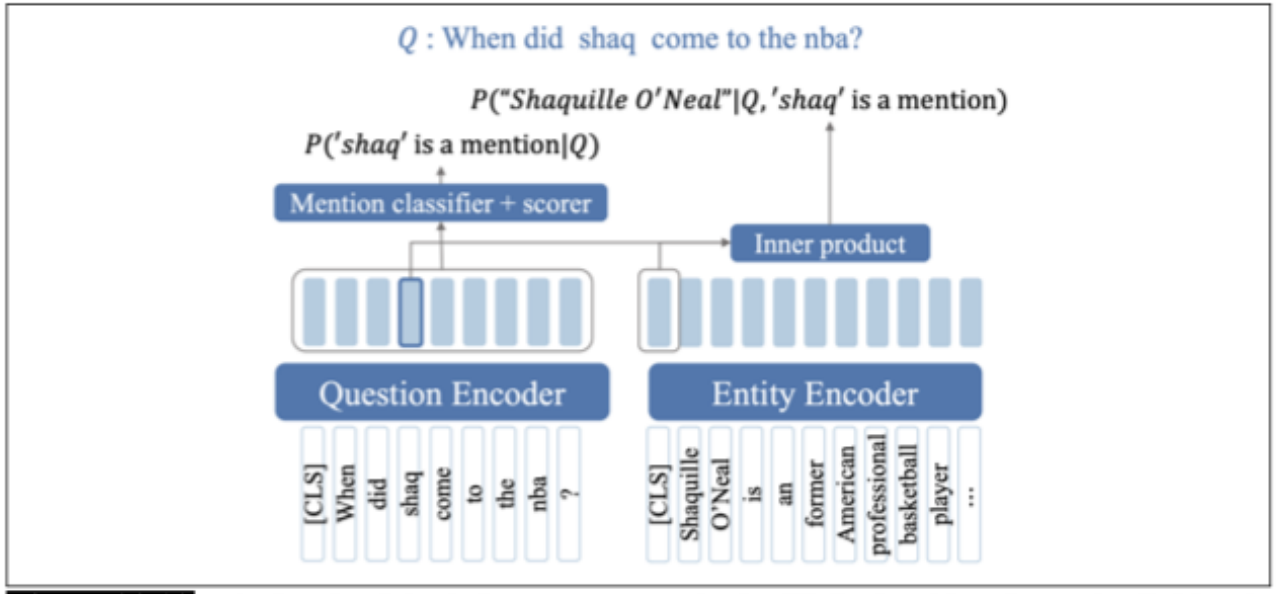

Figure 23.13 A sketch of the inference process in the ELQ algorithm for entity linking in questions (Li et al., 2020). Each candidate question mention span and candidate entity are separately encoded, and then scored by the entity/span dot product.

The ELQ linking algorithm

- In the ELQ mention detection phases
  - Get an h-dim embedding for each question token q through BERT: $[q_1 \dots q_n] = BERT([CLS]q_1 \cdots q_n[SEP])$
  - Get mention probabilities by combining the score for each span $[i, j]$ being the start $s_{start}(i)$ and the end $s_{end}(j)$, and the score for each token being part of a mention $s_{mention}(t)$

$$p([i,j]) = \sigma(S_{start}(i) + S_{end}(j) + \sum_{t=i}^{j} S_{mention}(t))$$

$$S_{start}(i) = w_{start} \cdot q_i,\ S_{end}(j) = w_{end} \cdot q_j$$

$$S_{mention}(i) = w_{mention} \cdot q_t$$

The ELQ linking algorithm

- In the ELQ entity linking phases
  - Get each entity representation e_i of all Wikipedia entities:

$$X_e = B_{[CLS]}([CLS]t(e_i)[ENT]d(e_i)[SEP])$$

  - Get a distribution over entities for each mention span:

$$p(e|[i,j]) = \frac{\exp\big(s(e,[i,j])\big)}{\sum_{e' \epsilon \varepsilon} \exp\big(s(e',[i,j])\big)}$$

$$s(e,[i,j]) = X_e y_{i,j},\ y_{i,j} = \frac{1}{(j-i+1)} \sum_{t-i}^{j} q_t$$

The ELQ linking algorithm

- The ELQ mention detection and entity linking phases can be fully supervised
  - Given a training set, the ELQ mention detection and entity linking phases are trained jointly, optimizing the sum of their losses
  - The mention detection loss is a binary cross-entropy loss:

$$\mathcal{L}_{MD} = -\frac{1}{N}\sum_{1\leq i\leq j\leq min(i+L-1,n)} \left(y_{[i,j]} \log p([i,j]) + \left(1\right.\right.$$

Knowledge-based Question Answering

- The idea is to map a language question to a query over a structured database
- There are two common paradigms
  - Graph-based QA
  - QA by semantic parsing

Knowledge-based QA from RDF triple stores

- A knowledge-based QA system
  - The task is to answer questions about one of the missing arguments, given the dataset in the form of RDF triples
    - E.g. The text question “When was Ada Lovelace born?”, an RDF triple like the following can be used to answer:

subject	**predicate**	**object**
Ada Lovelace	birth-year	1815

Knowledge-based QA from RDF triple stores

- The components of a knowledge-based QA system:
  - In entity linking stage, a textual mention like *Ada Lovelace* is mapped to the canonical entity ID in the knowledge base
  - Relation detection and linking stage is to determine which relation is being asked about
    - E.g. “When was Ada Lovelace born?” -> birth-year (Ada Lovelace, ?x)
    - This can be done using the neural entity linking models like BERT

Knowledge-based QA from RDF triple stores

- The neural entity linking models like BERT are used in elation detection and linking stage
  - E.g. BERT model can be used to represent the question span for the purposes of relation detection, and a separate vector is trained for each relation γ_i
  - The probability of a particular relation γ_i is then computed by softmax over dot products:

$$\boldsymbol{m_\gamma} = BERT_{CLS}([CLS]q_1 \cdots q_n[SEP])$$

$$s(\boldsymbol{m_r}, r_i) = \boldsymbol{m_r} \cdot \boldsymbol{w_{r_i}}$$

$$p(r_i|q_1, \ldots, q_n) = \frac{\exp\big(s(\boldsymbol{m_r}, r_i)\big)}{\sum_{k=1} N_R \exp\big(s(\boldsymbol{m_r}, r_k)\big)}$$

- The final stage is to rank the triples returned by the entity and relation inference steps

Knowledge-based QA by Semantic Parsing

- QA uses a semantic parser to map the question to a structured program to return an answer. The logical forms of the questions can be a query language like SQL or SPARQL, or some other executable program like the samples in figure 23.14

Question	Logical form
What states border Texas?	$\lambda x.\text{state}(x) \wedge \text{borders}(x, \text{texas})$
What is the largest state?	$\text{argmax}(\lambda x.\text{state}(x), \lambda x.\text{size}(x))$
I'd like to book a flight from San Diego to Toronto	SELECT DISTINCT f1.flight_id FROM flight f1, airport_service a1, city c1, airport_service a2, city c2 WHERE f1.from_airport=a1.airport_code AND a1.city_code=c1.city_code AND c1.city_name= 'san diego' AND f1.to_airport=a2.airport_code AND a2.city_code=c2.city_code AND c2.city_name= 'toronto'
How many people survived the sinking of the Titanic?	(count (!fb:event.disaster.survivors fb:en.sinking_of_the_titanic))
How many yards longer was Johnson's longest touchdown compared to his shortest touchdown of the first quarter?	ARITHMETIC diff(SELECT num(ARGMAX(SELECT)) SELECT num(ARGMIN(FILTER(SELECT))))

Knowledge-based QA by Semantic Parsing

- Semantic parsing algorithms are supervised with questions paired with their logical forms
- The task then is to take those pairs of training tuples to map from new questions to their logical forms; a common baseline is a sequence-to-sequence model, for example using BERT to represent question tokens, passing to an encoder-decoder, as shown in figure 23.15

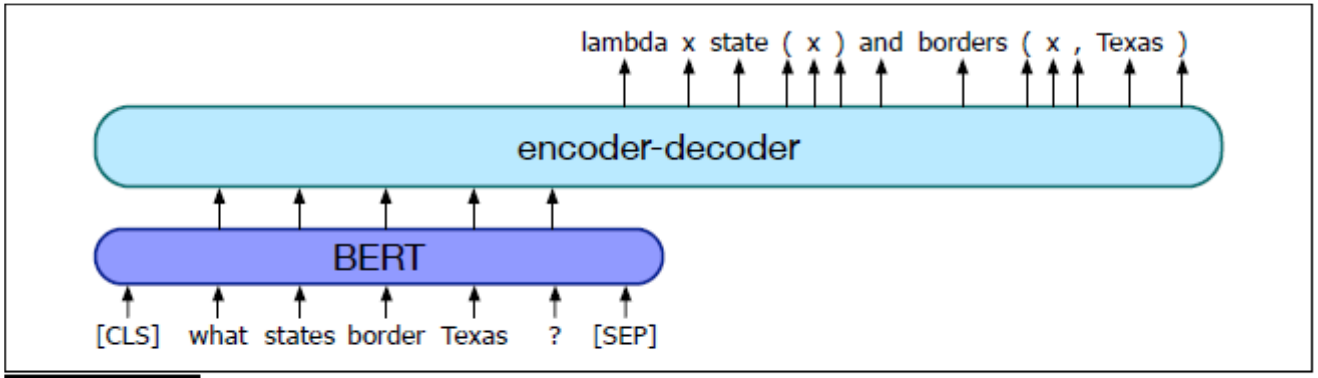

Figure 23.15 An encoder-decoder semantic parser for translating a question to logical form, with a BERT pre-encoder followed by an encoder-decoder (biLSTM or Transformer).

QA by a Pretrained Language Model

- Use a language model to answer a question solely from information stored in its parameters
  - In figure 23.16, the T5 model is an encoder-decoder architecture pretrained to fill in masked spans of task; then it is fine-tuned to the QA task by giving it a question, and training it to output the answer text in the decoder

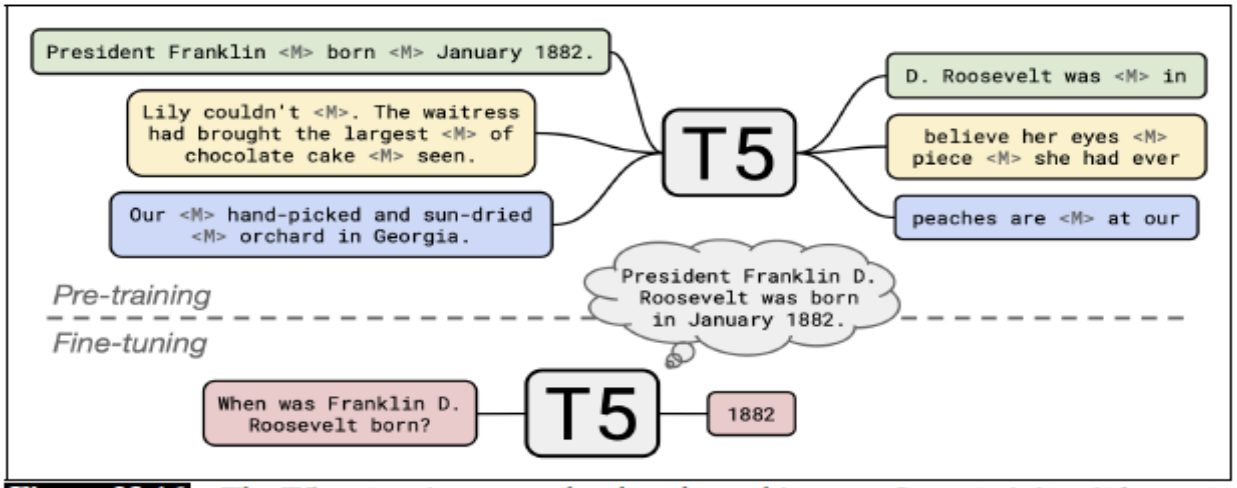

Figure 23.16 The T5 system is an encoder-decoder architecture. In pretraining, it learns to fill in masked spans of task (marked by <M>) by generating the missing spans (separated by <M>) in the decoder. It is then fine-tuned on QA datasets, given the question, without adding any additional context or passages. Figure from Roberts et al. (2020).

QA by a Pretrained Language Model

- Language modelling is not yet a complete solution for QA; for example there are poor interpretability, in addition to not working quite as well

Classic QA Models

- Pre-neural architectures using hybrids of rules and feature-based classifiers can achieve higher performance
- One classic system, the Watson DeepQA system from IBM, won the Jeopardy! Challenge in 2011 (Figure 23.17)

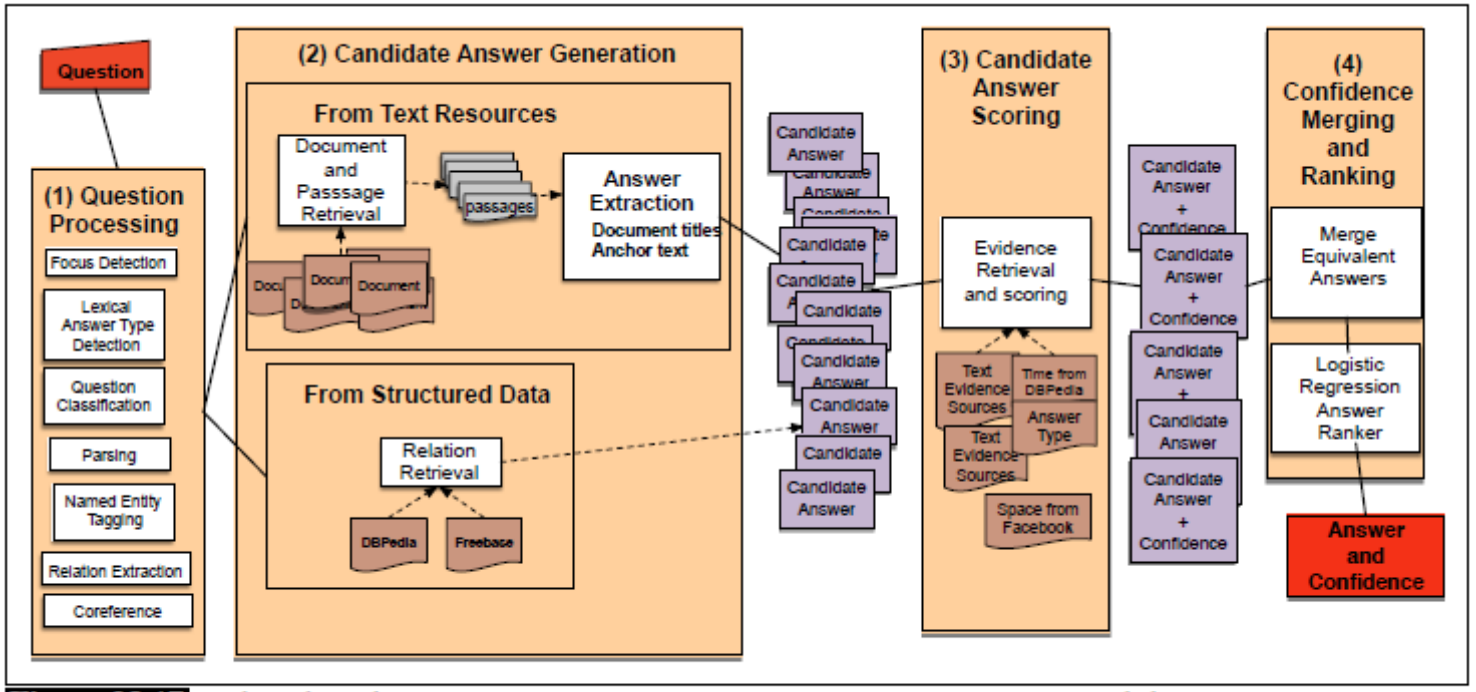

Figure 23.17 The 4 broad stages of Watson QA: (1) Question Processing, (2) Candidate Answer Generation, (3) Candidate Answer Scoring, and (4) Answer Merging and Confidence Scoring.

Classic QA Models

- DeepQA's intuition is to propose a large number of candidate answers from both text-based and knowledge-based sources and then use a rich variety of evidence features for scoring these candidates
- There are 4 broad stages of DeepQA. Give an example 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
  - Question Processing
  - Candidate Answer Generation
  - Candidate Answer Scoring
  - Answer Merging and Confidence Scoring

Question Processing

- In Question Processing stage
  - The question are parsed, and named entities like “Songs of a Sourdough” as a COMPOSITION are extracted
  - The question focus like “He” by handwritten rules is extracted
  - The lexical answer type like “He” are again extracted by rules: the syntactic headword of the focus by default, and other extra rules like additional lexical answer types or a particular syntactic relation with the focus
  - Finally the question is classified by type

Candidate Answer Generation

- In Candidate Answer Generation stage, the processed question is combined with external documents and structured knowledge to suggest candidate answers
  - Query structured sources with authorof(?x “Songs of a sourdough”) to return an author
  - Extract answers from text DeepQA using versions of Retrieve and Read

Candidate Answer Scoring

- In Candidate Answer Scoring stage, DeepQA uses many sources of evidence to score each candidate
  - Include a classifier to score whether the candidate can be interpreted as a subclass or instance of the potential answer type
  - Use time and space relations extracted from DBpedia or other structured databases
  - Use text retrieval to help retrieve evidence supporting a candidate answer
  - The output is a set of candidate answers with a vector of scoring feacures

Answer Merging and Scoring

- In this stage, a classifier takes each feature vector and assigns a confidence value to this candidate answer
  - The classifier is trained on many candidate answers, each labeled for whether it is correct or not, together with their feature vectors, and learns to predict a probability of being a correct answer.
  - The candidate answers are then sorted by the confidence value, resulting in a single best answer

Evaluation of Factoid Answers

- Factoid answers is commonly evaluated using mean reciprocal rank (MRR)
  - MRR returns a short ranked list of answers for each test set question, comparing against the correct answer
  - The MRR of a system is the average of the scores for each question in the test set Q; each test question is scored with the reciprocal of the rank of the first correct answer

$$MRR = \frac{1}{|Q|}\sum_{i=1}^{|Q|}\frac{1}{rank_i}$$

- Reading comprehension systems are evaluated via two metrics:
  - Exact match: The % of predicted answers that match the gold answer exactly
  - F_1 score: The average token overlap between predicted and gold answers