PES UNIVERSITY

ALGORITHMS FOR INFORMATION RETRIEVAL AND INTELLIGENT WEB

Question Answering, Neural Models for IR

Surabhi Narayan

Department of Computer Science Engineering

Slides collated by:

Gaurav Mahajan (TA VIII sem)

Anshula Aithal (TA VIII sem)

ALGORITHMS FOR INFORMATION RETRIEVAL AND INTELLIGENT WEB

PES UNIVERSITY

Neural Models for IR, QA as an Information Retrieval task - Factoid QA - 1

Surabhi Narayan

Department of Computer Science Engineering

Neural models for IR

The idea is that huge pretrained language models have read a lot of facts in their pre-training data, presumably including the location of the Louvre, and have encoded this information in their parameters.

For some general factoid questions this can be a useful approach and is used in practice. But prompting a large language model is not yet a solution for question answering.

One way to employ large pretrained language models, utilizing their exposure to extensive training data. The approach involves prompting these models with questions and relying on their ability to generate responses based on their encoded knowledge.

Neural models for IR

However, a significant challenge arises as large language models often produce incorrect or "hallucinated" answers. Hallucinations refer to responses that deviate from factual information available in the real world. This limitation highlights that while this approach can be useful for certain factoid questions, it is not a foolproof solution.

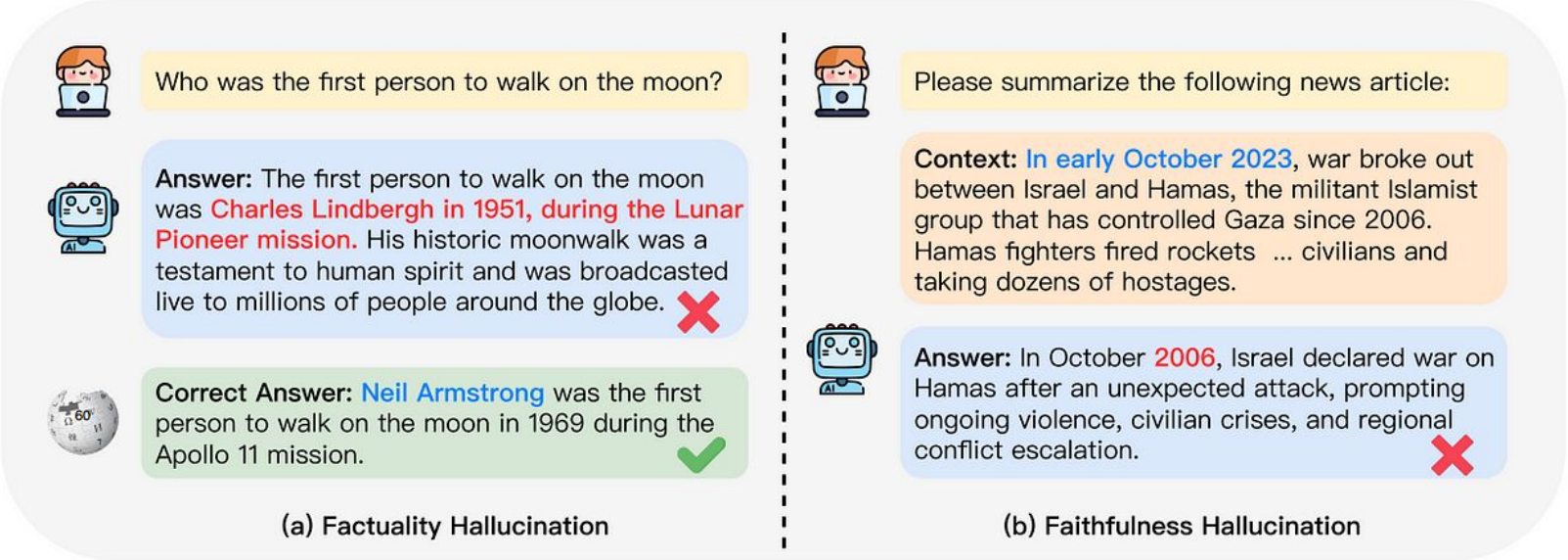


Figure 1: An intuitive example of LLM hallucination.

Neural models for IR

- The recognition of inaccuracies in generated responses calls for further refinement in the training and fine-tuning processes of question answering systems.

- One issue with language models is the lack of well-calibrated confidence estimates in their answers. In a calibrated system, the confidence of the system in the correctness of its answer is closely linked to the probability of that answer being correct. Ideally, if the system is unsure or incorrect, it should express hesitation or suggest verifying information from another source.

Neural models for IR

- However, language models often lack this calibration, leading to situations where they provide very wrong answers with a high degree of certainty.

- A second problem is that simply prompting a large language model doesn't allow us to ask questions about proprietary data, ie, data that is owned and controlled by an individual, an organization or a group. A common use of question-answering is to query private data, like asking an assistant about our email or private documents, or asking a question about our own medical records.

Neural models for IR

- internal datasets, or even the web itself, can be especially useful for rapidly changing or dynamic information; by contrast, large language models are often only released at long increments of many months and so may not have up-to-date information.
- Due to these downfalls, the retriever/reader model is the current dominated solution.
- In a retriever/reader model, information retrieval techniques are employed initially to retrieve documents that are likely to contain relevant information for answering a given question. Subsequently, the model may either extract an answer from specific spans of text within these documents or utilize large language models to generate an answer, a process sometimes referred to as retrieval-augmented generation.

Neural models for IR

- This approach addresses some of the challenges associated with using simple prompting methods for question answering. By basing answers on retrieved documents, it ensures that the response is grounded in facts sourced from a curated dataset. Additionally, the answer is provided in conjunction with the context of the passage or document from which it was extracted.

- Utilizing retrieval techniques offers several advantages. It enhances user confidence in the accuracy of the answer by providing context, allowing users to assess the reliability of the information. Moreover, this methodology can be applied to proprietary datasets, enabling the system to retrieve information from specialized domains such as legal or medical data for specific applications.

IR - with Dense Vectors

- The classic tf-idf or BM25 algorithms for IR have long been known to have a conceptual flaw: they work only if there is exact overlap of words between the query and document. In other words, the user posing a query (or asking a question) needs to guess exactly what words the writer of the answer might have used, an issue called the vocabulary mismatch problem.

- The solution to this problem is to use an approach that can handle synonymy: instead of (sparse) word-count vectors, using (dense) embeddings. This idea is implemented in modern times via encoders like BERT.

IR - with Dense Vectors

PES UNIVERSITY

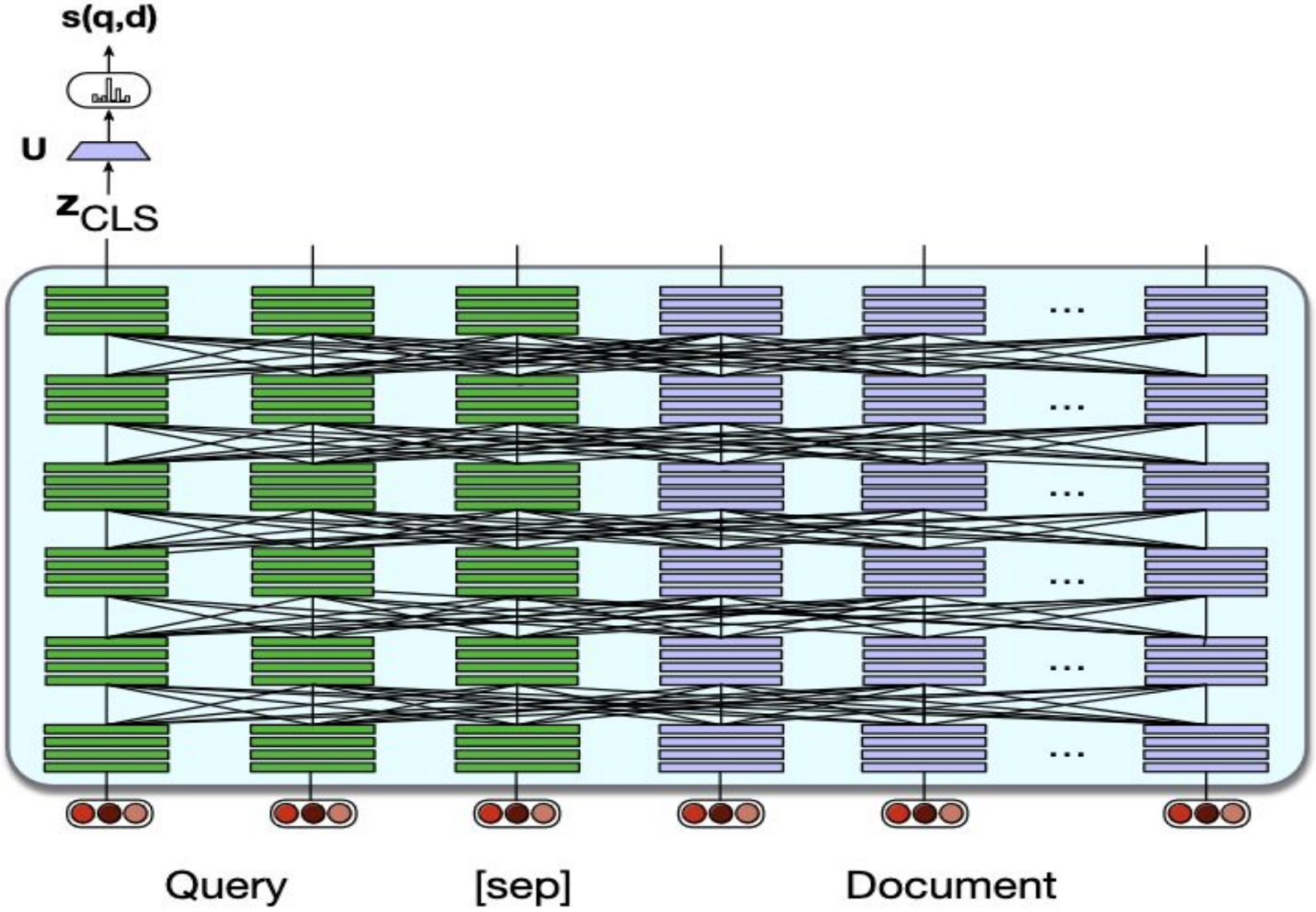
- The most powerful approach is to present both the query and the document to a single encoder. This enables the transformer's self-attention mechanism to see all the tokens of both the query and the document, and thus building a representation that is sensitive to the meanings of both query and document. Then a linear layer can be put on top of the [CLS](classification) token to predict a similarity score for the query/document tuple:

$$\mathbf{z} = \text{BERT}(\text{q}; [\text{SEP}]; \text{d})[\text{CLS}]$$

$$\text{score}(q, d) = \text{softmax}(\mathbf{U}(\mathbf{z}))$$

IR - with Dense Vectors

- This architecture is shown in the diagram.

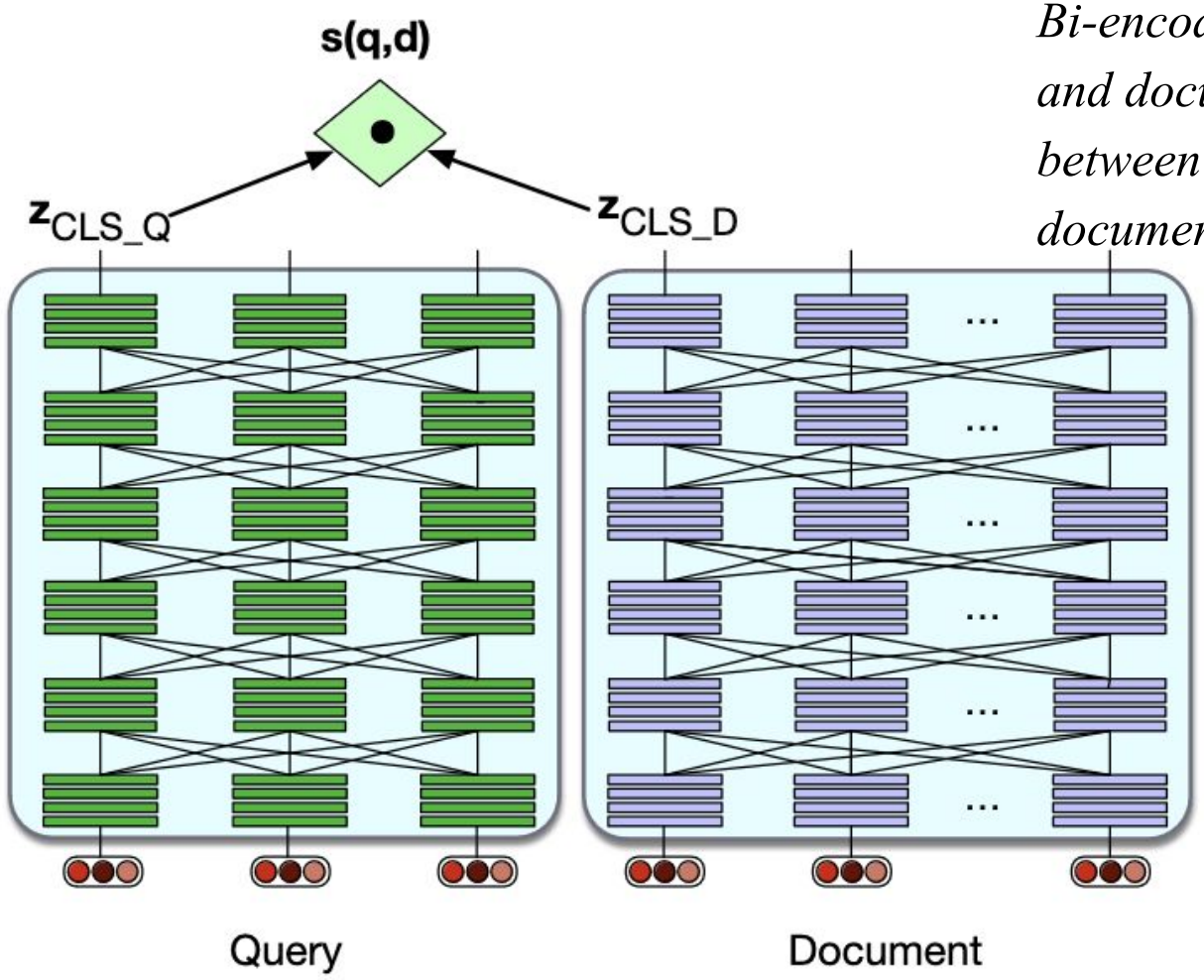


Query/ Document encoder: Use a single encoder to jointly encode query and document and finetune to produce a relevance score with a linear layer over the CLS token

IR - with Dense Vectors

- Here the retrieval step is not done on an entire document. Instead documents are broken up into smaller passages, such as non-overlapping fixed-length chunks of say 100 tokens, and the retriever encodes and retrieves these passages rather than entire documents
- The query and document have to be made to fit in the BERT 512-token window, for example by truncating the query to 64 tokens and truncating the document if necessary so that it, the query, [CLS], and [SEP](separator) fit in 512 tokens.
- The BERT system together with the linear layer U can then be fine-tuned for the relevance task by gathering a tuning dataset of relevant and non-relevant passages.
- The problem with this architecture is the expense in computation and time. With this architecture, every time we get a query, we have to pass every single document in our entire collection through a BERT encoder jointly with the new query. This enormous use of resources is impractical for real cases.

IR - with Dense Vectors



Bi-encoder: Use separate encoders for query and document, and use the dot product between CLS token outputs for the query and document as the score.

- At the opposite end of the computational spectrum is a more efficient architecture known as the bi-encoder. In this architecture we can encode the documents in the collection only one time by using two separate encoder models, one to encode the query and one to encode the document.

IR - with Dense Vectors

- We encode each document, and store all the encoded document vectors in advance. When a query comes in, we encode just this query and then use the dot product between the query vector and the precomputed document vectors as the score for each candidate document.
- For example, if we used BERT, we would have two encoders BERT_Q and BERT_D and we could represent the query and document as the [CLS] token of the respective encoders:

$$\mathbf{z}_q = \text{BERT}_Q(\text{q})\,[\text{CLS}]$$
$$\mathbf{z}_d = \text{BERT}_D(\text{d})\,[\text{CLS}]$$
$$\text{score}(q,d) = \mathbf{z}_q \cdot \mathbf{z}_d$$

- The bi-encoder is much cheaper than a full query/document encoder, but is also less accurate, since its relevance decision can't take full advantage of all the possible meaning interactions between all the tokens in the query and the tokens in the document.

QA as an Information Retrieval task - Factoid QA

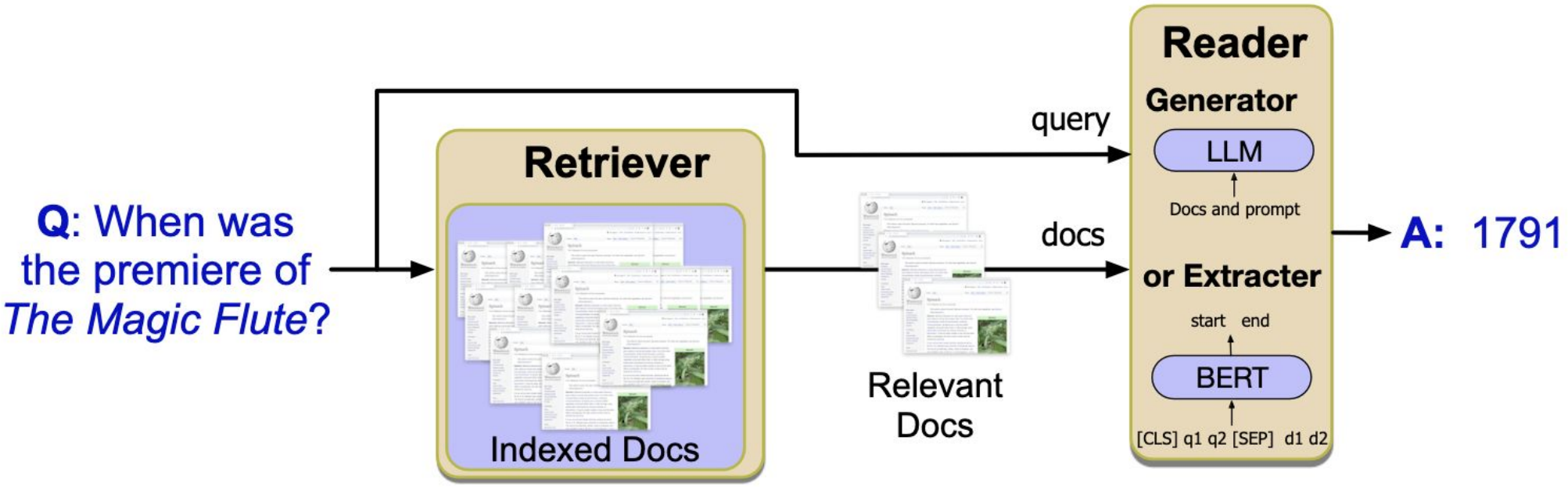
- The goal of retrieval-based Question Answering (sometimes referred to as open-domain QA) is to answer a user's question by either identifying short text segments from the web or another extensive collection of documents or by generating an answer based on the information retrieved from these sources.
- Some examples of Factoid questions are:

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

QA as an Information Retrieval task - Factoid QA

- The dominant paradigm for retrieval-based QA is sometimes called the retrieve and read model. In the first stage of this 2-stage model we retrieve relevant passages from a text collection, for example using the dense retrievers. The second stage, called the reader, is commonly implemented as either an extractor or a generator.
- The first method is span extraction, using a neural reading comprehension algorithm that passes over each passage and is trained to find spans of text that answer the question. The second method is also known as retrieval-augmented generation: we take a large pretrained language model, give it some set of retrieved passages and other text as its prompt, and autoregressively generate a new answer token by token.

QA as an Information Retrieval task - Factoid QA



QA as an Information Retrieval task - Factoid QA

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"

Fig: A passage from the SQuAD 2.0 dataset with 3 sample questions and the labeled answer spans.

THANK YOU

Surabhi Narayan

Department of Computer Science Engineering

PES UNIVERSITY

ALGORITHMS FOR INFORMATION RETRIEVAL AND INTELLIGENT WEB

Question Answering, Neural Models for IR

Surabhi Narayan

Department of Computer Science Engineering

Slides collated by:

Gaurav Mahajan (TA VIII sem)

Anshula Aithal (TA VIII sem)

ALGORITHMS FOR INFORMATION RETRIEVAL AND INTELLIGENT WEB

PES UNIVERSITY

Neural Models for IR, QA as an Information Retrieval task - Factoid QA - 2

Surabhi Narayan

Department of Computer Science Engineering

QA as an Information Retrieval task - Factoid QA

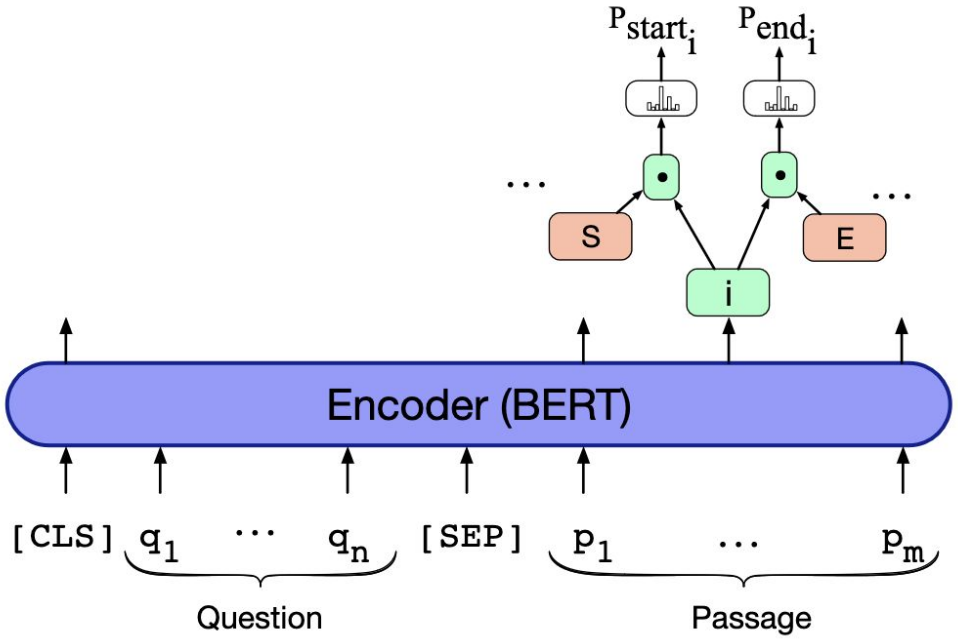
Reader Algorithms : 1. **Answer Span Extraction**

- The job of the reader is to take a passage as input and produce the answer. Here we introduce the span extraction style of reader, in which the answer is a span of text in the passage. For example given a question like “How tall is Mt. Everest?” and a passage that contains the clause Reaching 29,029 feet at its summit, a reader will output 29,029 feet.

- The answer extraction task is commonly modeled by span labeling: identifying in the passage a span (a continuous string of text) that constitutes an answer. Neuralspan algorithms for reading comprehension are given a question q of n tokens q1, ..., qn and a passage p of m tokens p1, ..., pm. Their goal is thus to compute the probability P(a|q, p) that each possible span a is the answer.

QA as an Information Retrieval task - Factoid QA

- If each span a starts at position as and ends at position a_e, we make the simplifying assumption that this probability can be estimated as $P(a|q, p) = P_{start}(a_s|q, p)P_{end}(a_e|q, p)$.
  Thus for for each token p_i in the passage we'll compute two probabilities: $p_{start}(i)$ that p_i is the start of the answer span, and $p_{end}(i)$ that pi is the end of the answer



QA as an Information Retrieval task - Factoid QA

- A standard baseline algorithm for reading comprehension is to pass the question and passage to any encoder like BERT, as strings separated with a [SEP] token, resulting in an encoding token embedding for every passage token p_i.

- For span-based question answering, we represent the question as the first sequence and the passage as the second sequence. We'll also need to add a linear layer that will be trained in the fine-tuning phase to predict the start and end position of the span.

QA as an Information Retrieval task - Factoid QA

- Many datasets (like SQuAD 2.0 and Natural Questions) also contain (question, passage) pairs in which the answer is not contained in the passage. We thus also need a way to estimate the probability that the answer to a question is not in the document. This is standardly done by treating questions with no answer as having the [CLS] token as the answer, and hence the answer span start and end index will point at [CLS]

- For datasets where annotated documents or passages exceed the maximum 512 input tokens allowed by BERT, such as Natural Questions with complete Wikipedia pages, a solution involves creating multiple pseudo-passage observations from the labeled Wikipedia page. Each observation is constructed by concatenating [CLS], the question, [SEP], and tokens from the document.

QA as an Information Retrieval task - Factoid QA

- To manage longer documents, a sliding window of size 512 (or 512 minus the question length and special tokens) is utilized to create successive pseudo-passages. This process involves walking through the document, packing each window of tokens into the next pseudo-passage. The answer span for each observation is either marked as [CLS] (indicating no answer in that specific window) or the gold-labeled span.
- This approach can also be applied during inference by breaking up each retrieved document into separate observation passages and labeling each one. The answer is then chosen as the span with the highest probability (or nil if no span surpasses the probability of [CLS]).

QA as an Information Retrieval task - Factoid QA

PES UNIVERSITY

Retrieval-Augmented Generation

- The second standard reader algorithm is to generate from a large language model, conditioned on the retrieved passages. This method is known as retrieval-augmented generation, or RAG.
- In simple conditional generation, the task of question answering can be framed as word prediction. By providing a language model with a question and a token like "A:" to signal that an answer should follow, the model is prompted as follows:

```
Q: Who wrote the book 'The Origin of Species'? A:
```

QA as an Information Retrieval task - Factoid QA

- Subsequently, autoregressive generation is performed conditioned on this input text. The model generates responses based on the provided context, progressing sequentially to construct a coherent answer to the posed question.
- More formally the simple autoregressive language modeling computes the probability of a string from the previous tokens:

$$p(x_1, \ldots, x_n) = \prod_{i=1}^{n} p(x_i | x_{<i})$$

QA as an Information Retrieval task - Factoid QA

- And simple conditional generation for question answering adds a prompt like Q: , followed by a query q , and A:, all concatenated:

$$p(x_1, \ldots, x_n) = \prod_{i=1}^{n} p(\texttt{[Q:]}\ ;\ q\ ;\ \texttt{[A:]}\ ;\ x_{<i})$$

QA as an Information Retrieval task - Factoid QA

- The advantage of using a large language model is the enormous amount of knowledge encoded in its parameters from the text it was pretrained on. But, while this kind of simple prompted generation can work fine for many simple factoid questions, it is not a general solution for QA, because it leads to hallucination and is unable to answer questions from proprietary data.

- The idea of retrieval-augmented generation is to address these problems by conditioning on the retrieved passages as part of the prefix, perhaps with some prompt text like “Based on these texts, answer this question:”.

QA as an Information Retrieval task - Factoid QA

- Datasets for IR-based QA are most commonly created by first developing reading comprehension datasets containing tuples of (passage, question, answer).
- Reading comprehension systems can use the datasets to train a reader that is given a passage and a question, and predicts a span in the passage as the answer.
- Including the passage from which the answer is to be extracted eliminates the need for reading comprehension systems to deal with IR.

QA as an Information Retrieval task - Factoid QA

PES UNIVERSITY

SQuAD Dataset : The Stanford Question Answering Dataset (SQuAD) consists of passages from Wikipedia and associated questions whose answers are spans from the passage.

HotpotQA Dataset: was created by showing crowd workers multiple context documents and asked to come up with questions that require reasoning about all of the documents.

The TriviaQA Dataset: contains 94K questions written by trivia enthusiasts, together with supporting documents from Wikipedia and the web resulting in 650K question-answer-evidence triples.

Natural Questions Dataset: incorporates real anonymized queries to the Google search engine

TyDi QA Dataset: contains 204K question-answer pairs from 11 typologically diverse languages, including Arabic, Bengali, Kiswahili, Russian, and Thai.

THANK YOU

Surabhi Narayan

Department of Computer Science Engineering

PES UNIVERSITY

ALGORITHMS FOR INFORMATION RETRIEVAL AND INTELLIGENT WEB

Question Answering, Neural Models for IR

Surabhi Narayan

Department of Computer Science Engineering

Slides collated by:

Gaurav Mahajan (TA VIII sem)

Anshula Aithal (TA VIII sem)

ALGORITHMS FOR INFORMATION RETRIEVAL AND INTELLIGENT WEB

Pretrained Models for QA

Surabhi Narayan

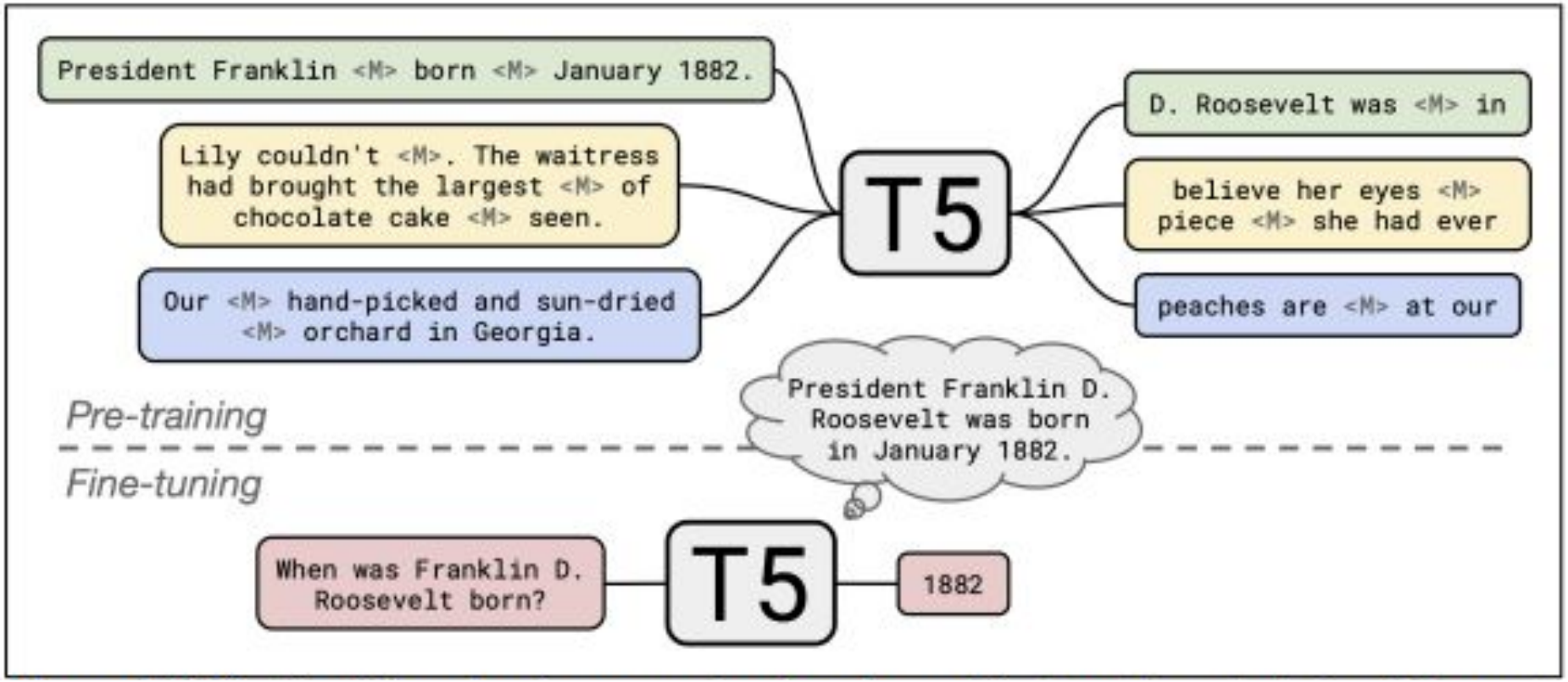
Department of Computer Science Engineering

Pretrained Models for QA

- An alternative approach to doing QA is to query a pre-trained language model, forcing a model to answer a question solely from information stored in its parameters.

- For example the T5 language model, which is an encoder decoder architecture pretrained to fill in masked spans

Question Answering



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

Question Answering

- Roberts et al. (2020) then finetune the T5 system to the question answering task, by giving it a question, and training it to output the answer text in the decoder. Using the largest 11-billion-parameter T5 model does competitively, although not quite as well as systems designed specifically for question answering.

- Language modeling is not yet a complete solution for question answering; for example in addition to not working quite as well, they suffer from poor interpretability (unlike standard QA systems, for example, they currently can't give users more context by telling them what passage the answer came from). Nonetheless, the study of extracting answers from language models is an intriguing area for future question answer research.

Watson DeepQA system

PES UNIVERSITY

Watson DeepQA system from IBM won the Jeopardy! challenge in 2011

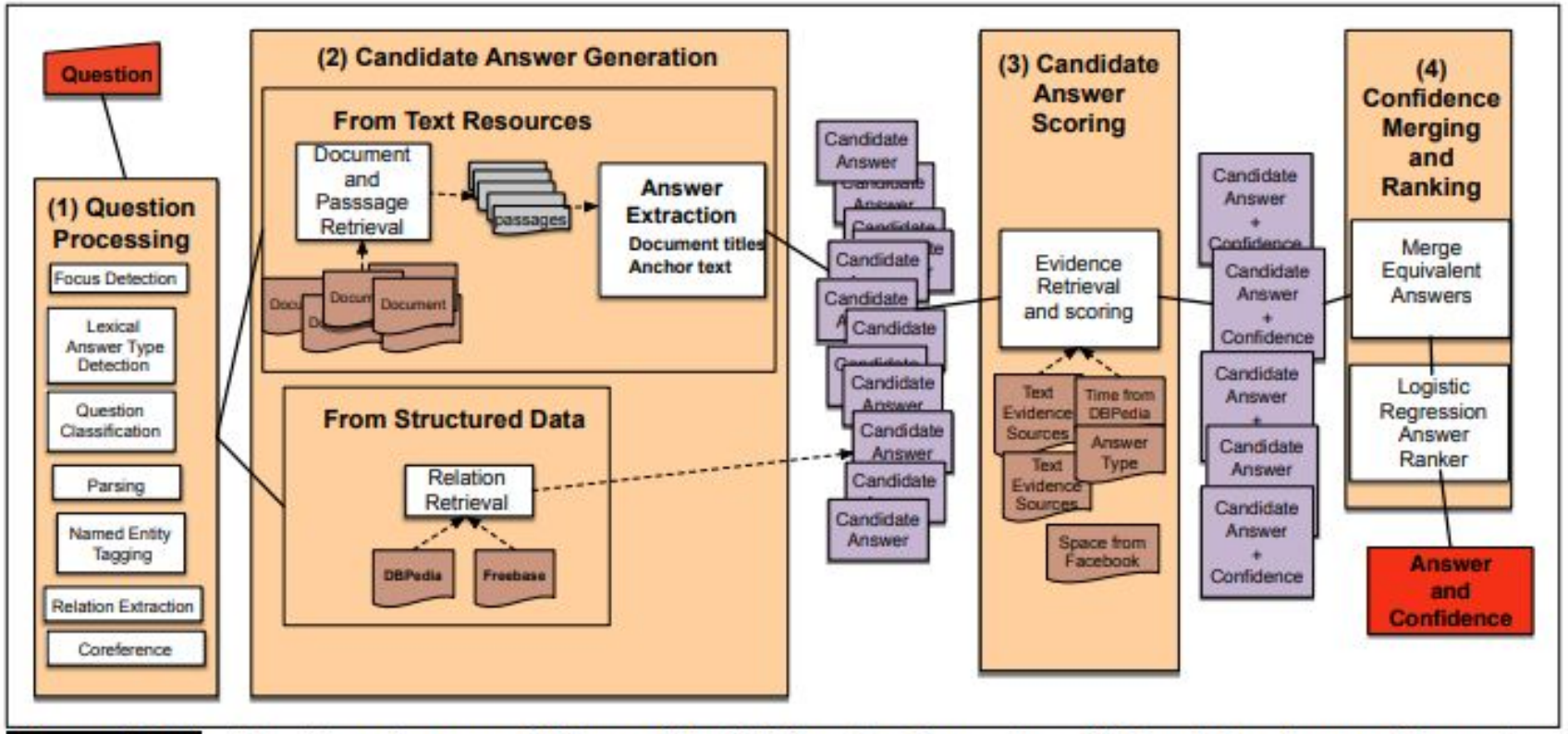


Figure 14.17 The 4 broad stages of Watson QA: (1) Question Processing, (2) Candidate Answer Generation, (3) Candidate Answer Scoring, and (4) Answer Merging and Confidence Scoring.

Watson DeepQA system

Let's consider how it handles these Jeopardy! examples, each 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.

Watson DeepQA System

PES UNIVERSITY

1. Question Processing

In this stage 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).

The question focus, shown in bold in both examples, is extracted. The focus is the string of words in the question that corefers with the answer. It is likely to be replaced by the answer in any answer string found and so can be used to align with a supporting passage. In DeepQA 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 the determiner “this” as in the Conan Doyle example, and rules extracting pronouns like she, he, hers, him, as in the poet example.

Watson DeepQA System

The lexical answer type (shown in blue above) is a word or words which tell lexical answer type us something about the semantic type of the answer. Because of the wide variety of questions in Jeopardy!, DeepQA chooses a wide variety of words to be answer types, rather than a small set of named entities. 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

Watson DeepQA System

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. These answer types will be used in the later ‘candidate answer scoring’ phase as a source of evidence for each candidate.

Watson DeepQA System

Relations like the following are also extracted:

authorof(focus,“Songs of a sourdough”)

publish (e1, he, “Songs of a sourdough”)

in (e2, e1, 1907)

temporallink(publish(...), 1907)

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

Watson DeepQA System

PES UNIVERSITY

2. Candidate Answer Generation

Next we combine the processed question with external documents and other knowledge sources to suggest many candidate answers from both text documents and structured knowledge bases.

To extract answers from text DeepQA uses simple versions of Retrieve and Read. For example for the IR stage, DeepQA generates a query from the question by eliminating stop words, and then upweighting any terms which occur in any relation with the focus.

Watson DeepQA System

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 passed to a standard IR system: (2.0 Robert Redford) (2.0 Paul Newman) star depression era grifter (1.5 flick)

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.

Watson DeepQA System

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 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.

Watson DeepQA System

PES UNIVERSITY

3.Candidate Answer Scoring

Next DeepQA uses many sources of evidence to score each candidate.
This includes a classifier that scores whether the candidate answer can be interpreted as a subclass or instance of the potential 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 hyponymy, or synonymy link, in this case finding hyponymy between “Symptom” and “Condition”.

Watson DeepQA System

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

4. Answer Merging and Scoring

DeepQA finally merges equivalent candidate answers. 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, for example using the anchor dictionaries described above for entity linking, which will list many synonyms for Wikipedia titles (e.g., JFK, John F. Kennedy, Senator John F. Kennedy, President Kennedy, Jack Kennedy). We then merge the evidence for each variant, combining the scoring feature vectors for the merged candidates into a single vector.

Watson DeepQA System

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.

Watson DeepQA System

DeepQA's fundamental intuition is thus to propose a very 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.

THANK YOU

Surabhi Narayan

Department of Computer Science Engineering

ALGORITHMS FOR INFORMATION RETRIEVAL

Entity Linking Models

Surabhi Narayan
Department of Computer Science Engineering

Slides collated by:
Gaurav Mahajan (TA VIII sem)
Anshula Aithal (TA VIII sem)

ALGORITHMS FOR INFORMATION RETRIEVAL

PES UNIVERSITY

Entity Linking Models

Dr. Surabhi Narayan
Department of Computer Science Engineering

Entity Linking Models

- Before we turn to the second major paradigm for question answering, knowledge-based question answering, we introduce the important core technology of entity linking, since it is required for any knowledge-based QA algorithm.

- Entity linking is the task of associating a mention in text with the representation of some real-world entity in an ontology.

- Entity linking is done in (roughly) two stages: **mention detection** and **mention disambiguation**.

Entity Linking Models - Linking based on Anchor Dictionaries and Web Graph

- As a simple baseline we introduce the TAGME linker for Wikipedia, which itself draws on earlier algorithms.
- Wikification algorithms define the set of entities as the set of Wikipedia pages, so we'll refer to each Wikipedia page as a unique entity e.
- TAGME first creates a catalog of all entities (i.e. all Wikipedia pages, removing some disambiguation and other meta-pages) and indexes them in a standard IR engine like Lucene.
- For each page e, the algorithm computes an in-link count in(e): the total number of in-links from other Wikipedia pages that point to e.
- These counts can be derived from Wikipedia dumps. Finally, the algorithm requires an anchor dictionary.

Entity Linking Models - Linking based on Anchor Dictionaries and Web Graph

- An anchor dictionary lists for each Wikipedia page, its anchor texts: the hyperlinked spans of text on other pages that point to it.
- For example, the web page for Stanford University, http://www.stanford.edu, might be pointed to from another page using anchor texts like Stanford or Stanford University:
  Stanford University
- We compute a Wikipedia anchor dictionary by including, for each Wikipedia page e, e's title as well as all the anchor texts from all Wikipedia pages that point to e.
- For each anchor string a we'll also compute its total frequency freq(a) in Wikipedia (including non-anchor uses), the number of times a occurs as a link (which we'll call link(a)), and its link probability linkprob(a) = link(a)/freq(a).

Entity Linking Models - Mention Detection

Given a question (or other text we are trying to link), TAGME detects mentions by querying the anchor dictionary for each token sequence up to 6 words. This large set of sequences is pruned with some simple heuristics (for example pruning substrings if they have small linkprobs). The question

When was Ada Lovelace born?

might give rise to the anchor Ada Lovelace and possibly Ada, but substrings spans like Lovelace might be pruned as having too low a linkprob, and but spans like born have such a low linkprob that they would not be in the anchor dictionary at all.

Entity Linking Models - Mention Disambiguation

- If a mention span is unambiguous (points to only one entity/Wikipedia page), we are done with entity linking!
- However, many spans are ambiguous, matching anchors for multiple Wikipedia entities/pages.
- The TAGME algorithm uses two factors for disambiguating ambiguous spans, which have been referred to as prior probability and relatedness/coherence. The first factor is **p(e|a)**, the probability with which the span refers to a particular entity. For each page **e ∈ E(a)**, the probability p(e|a) that anchor a points to e, is the ratio of the number of links into e with anchor text a to the total number of occurrences of a as an anchor:

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

Entity Linking Models - Mention Disambiguation

Let's see how that factor works in linking entities in the following question:

What Chinese Dynasty came before the Yuan?

- The most common association for the span Yuan in the anchor dictionary is the name of the Chinese currency, i.e., the probability p(Yuan currency| yuan) is very high.
- Rarer Wikipedia associations for Yuan include the common Chinese last name, a language spoken in Thailand, and the correct entity in this case, the name of the Chinese dynasty.
- So if we chose based only on p(e|a) , we would make the wrong disambiguation and miss the correct link, Yuan dynasty.

Entity Linking Models - Mention Disambiguation

- To help in just this sort of case, TAGME uses a second factor, the relatedness of this entity to other entities in the input question. In our example, the fact that the question also contains the span Chinese Dynasty, which has a high probability link to the page Dynasties in Chinese history, ought to help match Yuan dynasty.

- Given a question q, for each candidate anchors span a detected in q, we assign a relatedness score to each possible entity $e \in E(a)$ of a. The relatedness score of the link $a \rightarrow e$ is the weighted average relatedness between e and all other entities in q. Two entities are considered related to the extent their Wikipedia pages share many in-links

Entity Linking Models - Mention Disambiguation

The relatedness between two entities A and B is computed as

$$\text{rel}(A,B) = \frac{\log(\max(|\text{in}(A)|,|\text{in}(B)|)) - \log(|\text{in}(A) \cap \text{in}(B)|)}{\log(|W|) - \log(\min(|\text{in}(A)|,|\text{in}(B)|))}$$

The vote given by anchor b to the candidate annotation a $\longrightarrow$ X is the average, over all the possible entities of b, of their relatedness to X, weighted by their prior probability:

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

Entity Linking Models - Mention Disambiguation

The total relatedness score for a → X is the sum of the votes of all the other anchors detected in q:

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

To score a → X, we combine relatedness and prior by choosing the entity X that has the highest relatedness(a → X), finding other entities within a small of this value, and from this set, choosing the entity with the highest prior P(X|a). The result of this step is a single entity assigned to each span in q.

Entity Linking Models - Mention Disambiguation

The TAGME algorithm has one further step of pruning spurious anchor/entity pairs, assigning a score averaging link probability with the coherence.

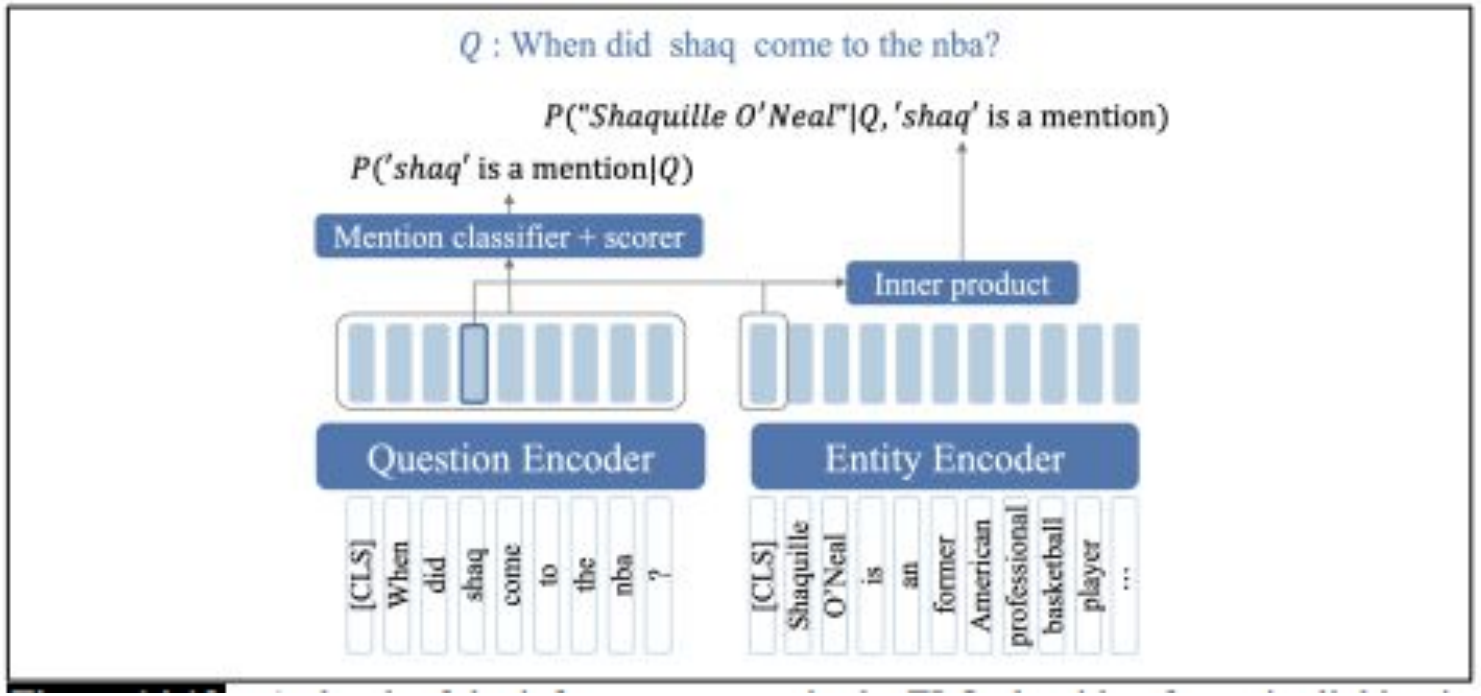
$$\text{coherence}(a \to X) = \frac{1}{|S| - 1} \sum_{B \in \mathcal{S} \setminus X} \text{rel}(B, X)$$

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

Finally, pairs are pruned if score(a $\rightarrow$ X) < λ, 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. This allows embeddings for all the entities in the knowledge base to be precomputed and cached.



Let's sketch the ELQ linking algorithm, which is given a question q and a set of candidate entities from Wikipedia with associated Wikipedia text, and outputs tuples (e,ms,me) of entity id, mention start, and mention end

Neural Graph-based linking

Entity Mention Detection To get an h-dimensional embedding for each question token, the algorithm runs the question through BERT in the normal way:

$$[\mathbf{q}_1 \cdots \mathbf{q}_n] = \text{BERT}([\text{CLS}]q_1 \cdots q_n[\text{SEP}])$$

It then computes the likelihood of each span [i, j] in q being an entity mention, in a way similar to the span-based algorithm we saw for the reader above. First we compute the score for i/ j being the start/end of a mention:

$$s_{\text{start}}(i) = \mathbf{w}_{\text{start}} \cdot \mathbf{q}_i, \qquad s_{\text{end}}(j) = \mathbf{w}_{\text{end}} \cdot \mathbf{q}_j,$$

Neural Graph-based linking

where $\mathbf{w}_{\text{start}}$ and $\mathbf{w}_{\text{end}}$ are vectors learned during training. Next, another trainable embedding, $\mathbf{w}_{\text{mention}}$ is used to compute a score for each token being part of a mention:

$$s_{\text{mention}}(t) = \mathbf{w}_{\text{mention}} \cdot \mathbf{q}_t$$

Mention probabilities are then computed by combining these three scores:

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

Neural Graph-based linking

To link mentions to entities, we next compute embeddings for each entity in the set E = e1,⋯ , ei ,⋯ , ew of all Wikipedia entities. For each entity ei we'll get text from the entity's Wikipedia page, the title t(ei) and the first 128 tokens of the Wikipedia page which we'll call the description d(ei). This is again run through BERT, taking the output of the CLS token BERT[CLS] as the entity representation:

$$\mathbf{x}_e = \text{BERT}_{[\text{CLS}]}([\text{CLS}]t(e_i)[\text{ENT}]d(e_i)[\text{SEP}])$$

Neural Graph-based Linking

Mention spans can be linked to entities by computing, for each entity e and span [i, j], the dot product similarity between the span encoding (the average of the token embeddings) and the entity encoding.

$$\mathbf{y}_{i,j} = \frac{1}{(j-i+1)} \sum_{t-i}^{j} \mathbf{q}_t$$

$$s(e,[i,j]) = \mathbf{x}_e \mathbf{y}_{i,j}$$

Finally, we take a softmax to get a distribution over entities for each span:

$$p(e|[i,j]) = \frac{\exp(s(e,[i,j]))}{\sum_{e' \in \mathcal{E}} \exp(s(e',[i,j]))}$$

The ELQ mention detection and entity linking algorithm is fully supervised

THANK YOU

Surabhi Narayan

Department of Computer Science Engineering

ALGORITHMS FOR INFORMATION RETRIEVAL AND INTELLIGENT WEB

Coreference Resolution

Surabhi Narayan
Department of Computer Science Engineering

Slides collated by:
Gaurav Mahajan (TA VIII sem)
Anshula Aithal (TA VIII sem)

ALGORITHMS FOR INFORMATION RETRIEVAL AND INTELLIGENT WEB

PES UNIVERSITY

Coreference Resolution

Dr. Surabhi Narayan
Department of Computer Science Engineering

Coreference Resolution

- An essential aspect of language processing involves determining the entity being discussed in a text. Take, for example, the following passage:

  "Victoria Chen, CFO of Megabucks Banking, saw her pay jump to $2.3 million, as the 38-year-old became the company's president. It is widely known that she came to Megabucks from rival Lotsabucks."

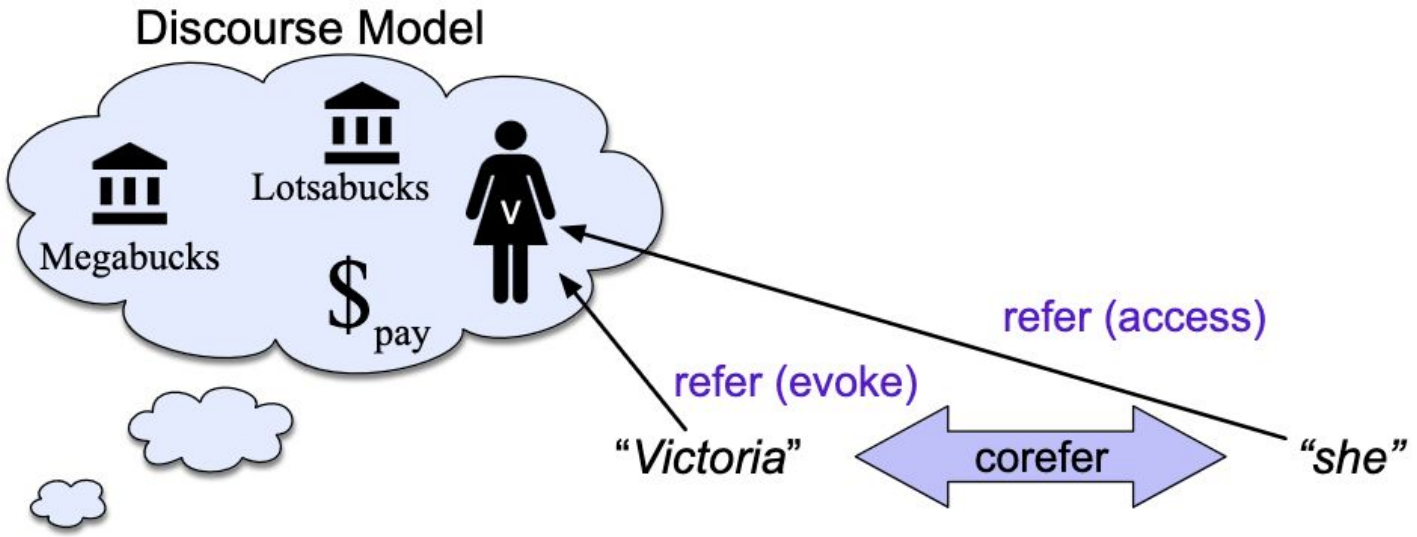
- In this passage, the underlined phrases serve as referring expressions used by the writer to identify the person named Victoria Chen. These linguistic expressions, such as "her" or "Victoria Chen," are referred to as mentions or referring expressions. The actual person being referred to (Victoria Chen) is called the referent, and to differentiate between referring expressions and their referents, the former is typically italicized.

Coreference Resolution

- When two or more referring expressions are employed to denote the same entity in a discourse, it is termed coreference. In this passage, "Victoria Chen" and "she" are examples of referring expressions that corefer to the same individual. This understanding of coreference is crucial in comprehending and analyzing the relationships between different elements within a given text.
- Natural language processing systems (and humans) interpret linguistic expressions with respect to a discourse model. A discourse model is a mental model that the understander builds incrementally when interpreting a text, containing representations of the entities referred to in the text, as well as properties of the entities and relations among them.

Coreference Resolution

- When a referent is first mentioned in a discourse, we say that a representation for it is evoked into the evoked model. Upon subsequent mention, this representation is accessed from the model.



Coreference Resolution

- Reference in a text to an entity that has been previously introduced into the discourse is called **anaphora**, and the referring expression used is said to be **anaphor, or anaphoric.**
- The anaphor corefers with a prior mention (in this case Victoria Chen) that is called the **antecedent**. Not every referring expression is an antecedent. An entity that has only a single mention in a text is called a **singleton**.
- Coreference resolution is the task of determining whether two mentions corefer, meaning they refer to the same entity in the discourse model, also known as the same discourse entity. The collection of coreferring expressions is commonly referred to as a coreference chain or a cluster.

Coreference Resolution

- In the context of the provided example in processing , a coreference resolution algorithm would be tasked with identifying at least four coreference chains, corresponding to the four entities in the discourse model:

1. {Victoria Chen, her, the 38-year-old, She}
2. {Megabucks Banking, the company, Megabucks}
3. {her pay}
4. {Lotsabucks}

Note: her is syntactically part of another mention, her pay, referring to a completely different discourse entity.

Coreference Resolution

- Coreference resolution thus comprises two tasks (although they are often performed jointly):
  (1) identifying the mentions
  (2) clustering them into coreference chains/discourse entities.
  Beyond coreference resolution, there is often a need to delve deeper into determining which real-world entity is associated with a given discourse entity. For instance, the mention of "Washington" could refer to the U.S. state, the capital city, or the historical figure George Washington. The interpretation of the sentence varies significantly depending on the specific real-world entity being referenced.

Coreference Resolution

- The formulation of the coreference resolution task is as follows: When provided with a text T, the objective is to identify all entities within the text and determine the coreference links between these entities. The evaluation of the system's performance is conducted by comparing the generated coreference links produced by the system with those present in manually created gold coreference annotations for the same text T. Eg:

$[\text{Victoria Chen}]_a^1$, CFO of $[\text{Megabucks Banking}]_a^2$, saw $[[\text{her}]_b^1 \text{ pay}]_a^3$ jump to \$2.3 million, as $[\text{the 38-year-old}]_c^1$ also became $[[\text{the company}]_b^2$'s president. It is widely known that $[\text{she}]_d^1$ came to $[\text{Megabucks}]_c^2$ from rival $[\text{Lotsabucks}]_a^4$.

Coreference Resolution

- Solving this task involves addressing challenges such as resolving pronominal anaphora (identifying that "her" refers to Victoria Chen), filtering out non-referential pronouns like the pleonastic "It" in phrases like "It has been ten years," handling definite noun phrases to establish coreference relationships (e.g., recognizing that "the 38-year-old" is coreferent with Victoria Chen), and deciphering references within names (e.g., realizing that "Megabucks" is the same entity as "Megabucks Banking"). This multifaceted process is essential for accurately identifying and linking mentions into coherent coreference clusters.

Coreference Resolution

- Exactly what counts as a mention and what links are annotated differs from task to task and dataset to dataset. For example some coreference datasets do not label singletons, making the task much simpler. Resolvers can achieve much higher scores on corpora without singletons, since singletons constitute the majority of mentions in running text, and they are often hard to distinguish from non-referential NPs.
- A few popular coreference datasets include OntoNotes, ISNotes, LitBank coreference corpus, ARRAU corpus, etc.

THANK YOU

Surabhi Narayan

Department of Computer Science Engineering

ALGORITHMS FOR INFORMATION RETRIEVAL AND INTELLIGENT WEB

Mention Detection

Surabhi Narayan
Department of Computer Science Engineering

Slides collated by:
Gaurav Mahajan (TA VIII sem)
Anshula Aithal (TA VIII sem)

ALGORITHMS FOR INFORMATION RETRIEVAL AND INTELLIGENT WEB

PES UNIVERSITY

Mention Detection

Dr. Surabhi Narayan
Department of Computer Science Engineering

Mention Detection

- The first stage of coreference is mention detection: finding the spans of text that constitute each mention.
- Mention detection algorithms typically take a liberal approach by proposing a wide range of candidate mentions, prioritizing high recall. The emphasis at this stage is on inclusivity, capturing a broad set of potential mentions.
- Subsequent stages in the coreference resolution process involve more refined filtering to enhance precision.

Mention Detection

- Using the previous sample sentence:
  Victoria Chen, CFO of Megabucks Banking, saw her pay jump to $2.3 million, as the 38-year-old also became the company's president. It is widely known that she came to Megabucks from rival Lotsabucks.
  It might result in the following list of 13 potential mentions:

Victoria Chen	$2.3 million	she
CFO of Megabucks Banking	the 38-year-old	Megabucks
Megabucks Banking	the company	Lotsabucks
her	the company's president	
her pay	It	

Mention Detection

- The span-based algorithm described first extracts all n-gram spans of words up to N=10. However, it's essential to note that many noun phrases (NPs) and the majority of random n-gram spans are not referring expressions. Consequently, all such mention detection systems need to eventually filter out pleonastic/expletive pronouns like "It," appositives such as "CFO of Megabucks Banking Inc," or predicate nominals like "the company's president" or "$2.3 million."

Mention Detection

- Mention-detection rules are sometimes designed specifically for particular evaluation campaigns. For OntoNotes, for example, mentions are not embedded within larger mentions, and while numeric quantities are annotated, they are rarely coreferential. Thus for OntoNotes tasks like CoNLL 2012 , a common first pass rule-based mention detection algorithm is:

1. Take all NPs, possessive pronouns, and named entities.
2. Remove numeric quantities (100 dollars, 8%), mentions embedded in larger mentions, adjectival forms of nations, and stop words (like *there*).
3. Remove pleonastic *it* based on regular expression patterns.

THANK YOU

Surabhi Narayan

Department of Computer Science Engineering

PES UNIVERSITY

ALGORITHMS FOR INFORMATION RETRIEVAL AND INTELLIGENT WEB

Question Answering, Neural Models for IR

Surabhi Narayan

Department of Computer Science Engineering

Slides collated by:

Gaurav Mahajan (TA VIII sem)

Anshula Aithal (TA VIII sem)

ALGORITHMS FOR INFORMATION RETRIEVAL AND INTELLIGENT WEB

PES UNIVERSITY

Question Answering - Introduction

Surabhi Narayan

Department of Computer Science Engineering

Question Answering

- Question Answering is a computer science discipline within the fields of information retrieval and natural language processing, which focuses on building systems that automatically answer questions posed by humans in a natural language.

- Question answering systems are designed to meet human information needs in contexts like virtual assistants, chatbots, search engines, or database queries. They primarily target factoid questions, seeking concise factual answers.
  Eg: Where is the Louvre Museum located?

Question Answering

- 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.

- We will learn the two major paradigms for factoid question answering. Information-retrieval (IR) based QA, sometimes called open domain QA, relies on the vast amount of text on the web or in collections of scientific papers like PubMed.

Question Answering

- QA systems can be based on various techniques, including information retrieval, knowledge-based, generative, and rule-based approaches. Each method has its strengths and weaknesses, and the choice of method depends on the project's specific needs.

- QA systems can be used in many places, like customer service, search engines, healthcare, education, finance, e-commerce, voice assistants, chatbots, and virtual assistants.

Question Answering - Typical flow

A natural language question-answering (QA) system is a computer program that automatically answers questions using NLP. The basic process of a natural language QA system includes the following steps:

1. **Text pre-processing**: The question is pre-processed to remove irrelevant information and standardise the text's format. This step includes tokenization, lemmatization, and stop-word removal, among others.

2. **Question understanding**: The pre-processed question is analysed to extract the relevant entities and concepts and to identify the type of question being asked. This step can be done using natural language processing (NLP) techniques such as named entity recognition, dependency parsing, and part-of-speech tagging.

Question Answering

3. Information retrieval: The question is used to search a database or corpus of text to retrieve the most relevant information. This can be done using information retrieval techniques such as keyword search or semantic search.

4. Answer generation: The retrieved information is analysed to extract the specific answer to the question. This can be done using various techniques, such as machine learning algorithms, rule-based systems, or a combination.

5. Ranking: The extracted answers are ranked based on relevance and confidence score.

Question Answering

- Given a user question, information retrieval is used to find relevant passages. Then neural reading comprehension algorithms read these retrieved 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, such as mapping
  What states border Texas? to the logical representation: λx.state(x) ∧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.

Question Answering - Types

PES UNIVERSITY

Information retrieval-based QA

This approach uses information retrieval techniques, such as keyword or semantic search, to identify the documents or passages most likely to hold the answer to a given question.

Knowledge-based QA

Knowledge-based question answering (QA) automatically answers questions using a knowledge base, such as a database or ontology, to retrieve the relevant information.

Question Answering – Types

Generative QA

Generative question answering (QA) automatically answers questions using a generative model, such as a neural network, to generate a natural language answer to a given question.

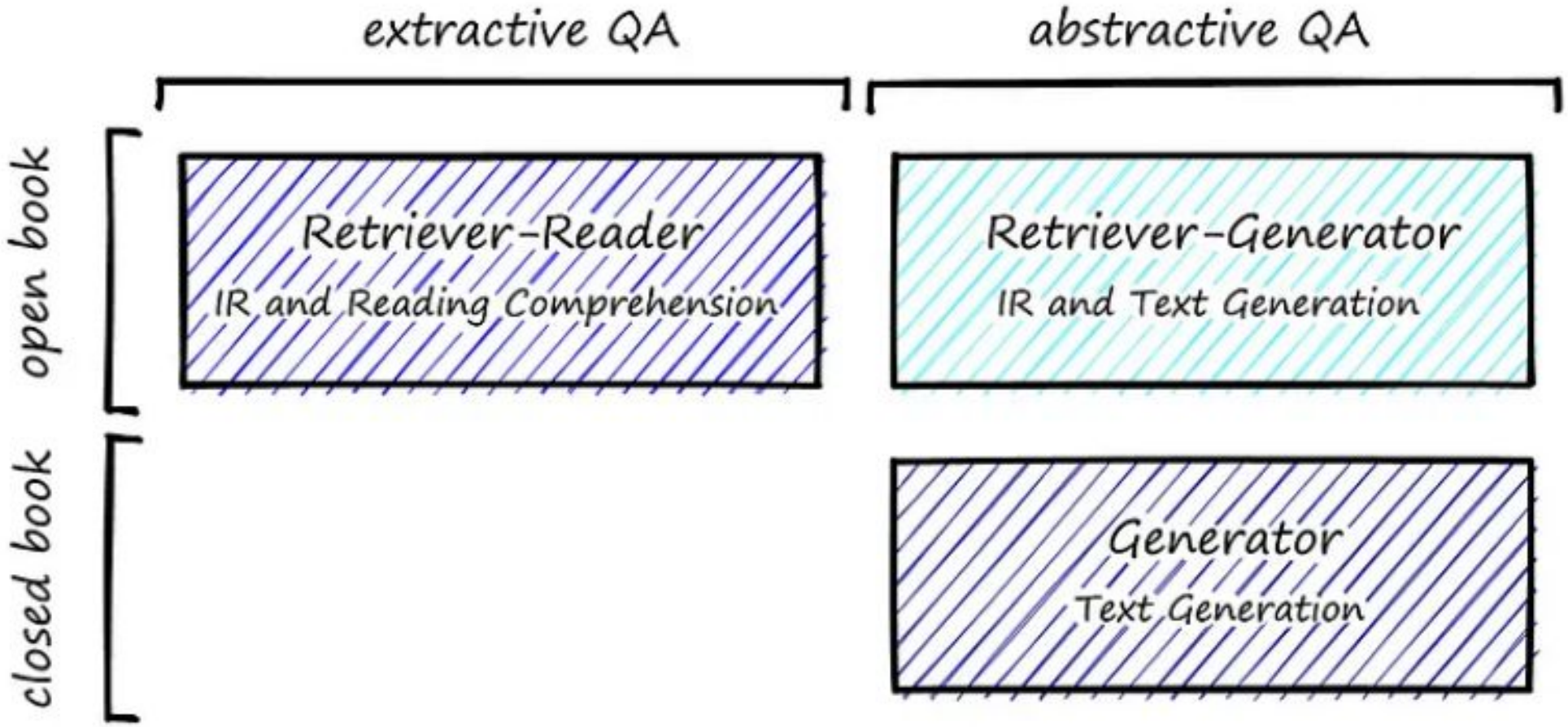
Hybrid QA

Hybrid question answering (QA) automatically answers questions by combining multiple QA approaches, such as information retrieval-based, knowledge-based, and generative QA.

Rule-based QA

Rule-based question answering (QA) automatically answers questions using a predefined set of rules based on keywords or patterns in the question.

Question Answering - Variants based on Answer creation methods



There are a few approaches to question answering (QA).

Question Answering

1. **Extractive QA**: In this approach, the model directly retrieves the answer from the provided context and presents it to the user. This type of QA is commonly solved using BERT-like models.
2. **Generative QA**: Contrasting with extractive QA, generative QA involves the model generating free-form text as the answer, directly based on the given context. This method relies on Text Generation models.

QA systems differ in the **source of their answers:**

1. **Open QA**: Here, the answer is extracted from a context, similar to extractive QA.
2. **Closed QA**: In contrast, closed QA does not receive any context for answering. The model generates the entire answer autonomously without external context.

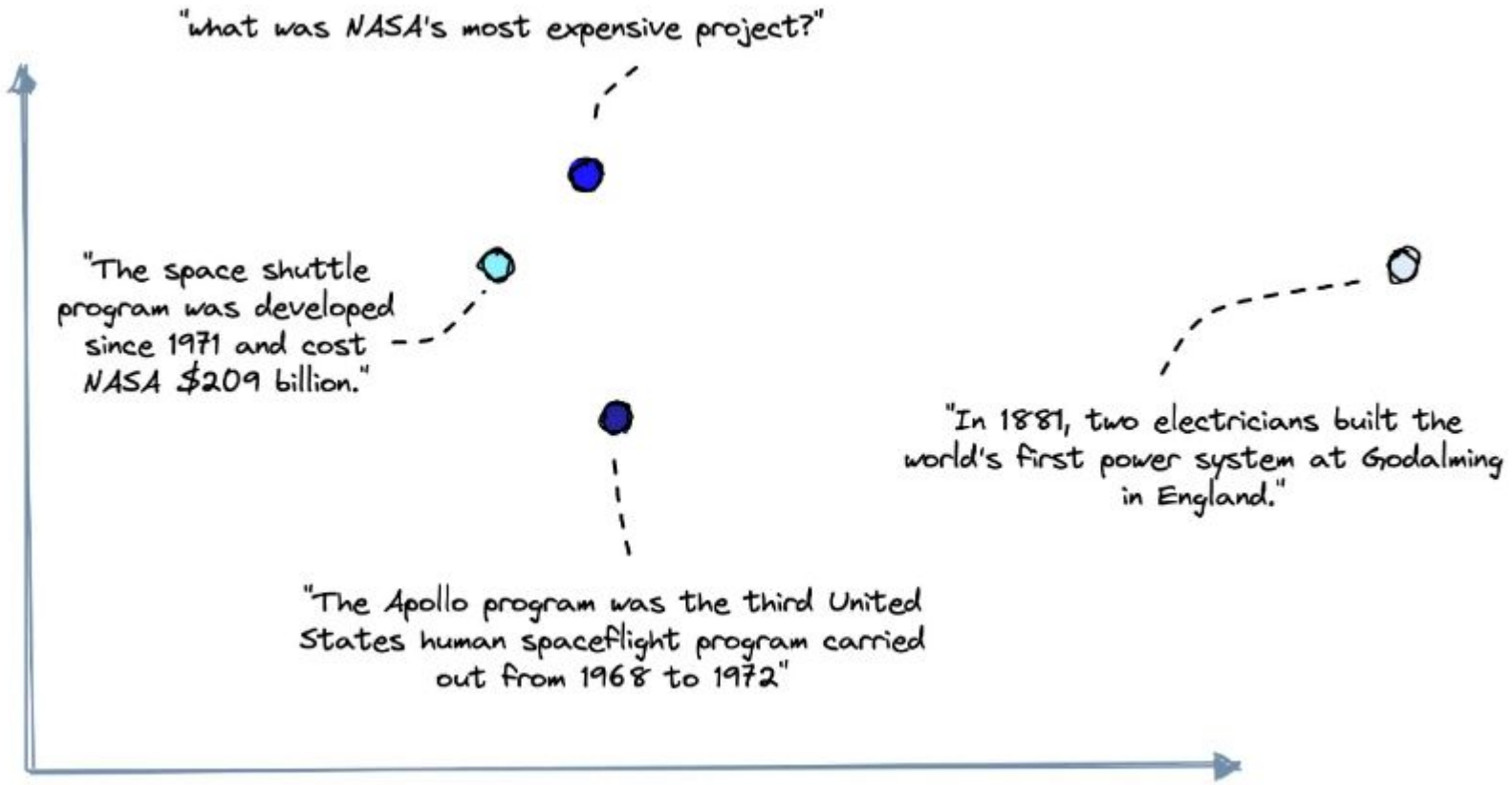
Question Answering - Components

Indexed data (document store/vector database) i.e. External Knowledge

Indexed data storage is a way to organize documents so that they can be quickly retrieved when a question is asked. This is done by creating indexes of the documents, which are essentially lists of the words and phrases that appear in the documents. When a question is asked, the model can use the indexes to quickly find documents that are likely to contain the answer to the question.Some popular indexed data storage systems include Elasticsearch, Solr, and MongoDB.

A **vector database** is a type of document store that stores documents as vectors. The vectors for different words and phrases are similar to each other if the words and phrases have similar meanings.Some popular vector databases include Pincone, Chrome, and Wiavate.

Question Answering - Components



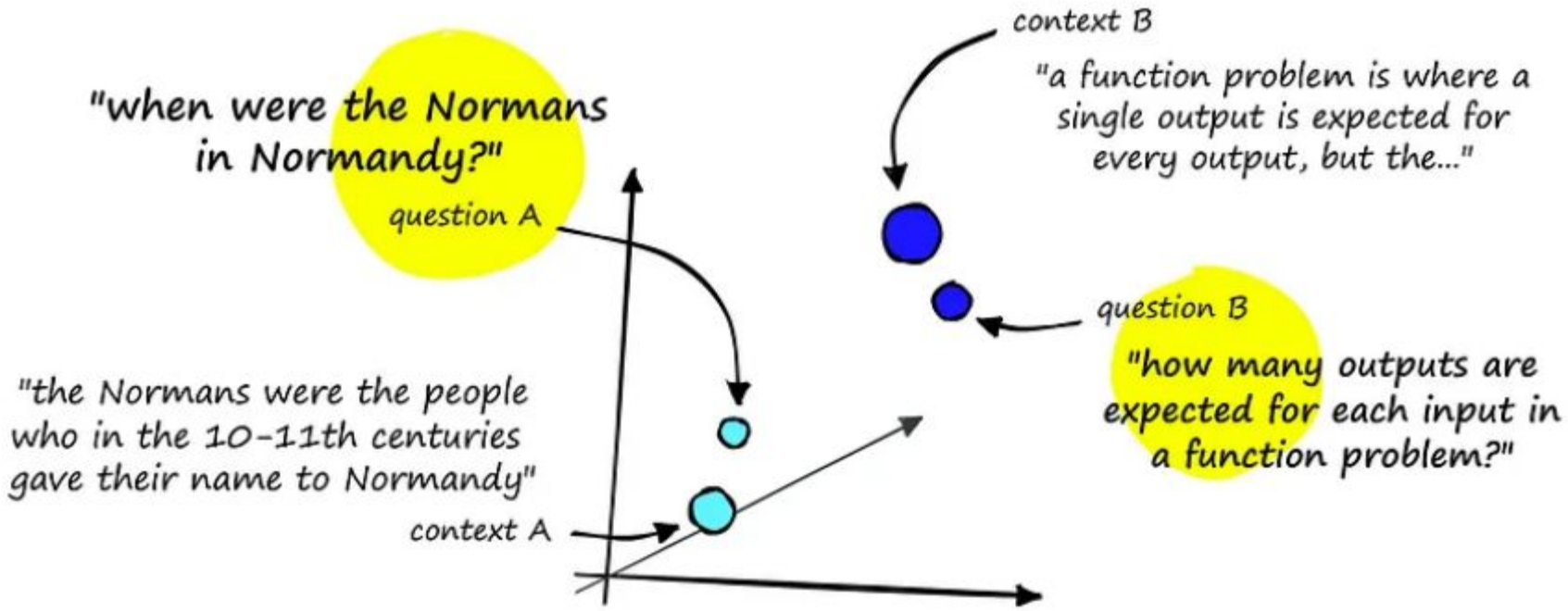
Queries and contexts with similar meaning are embedding into a similar vector space.

Question Answering - Components

PES UNIVERSITY

Retriever

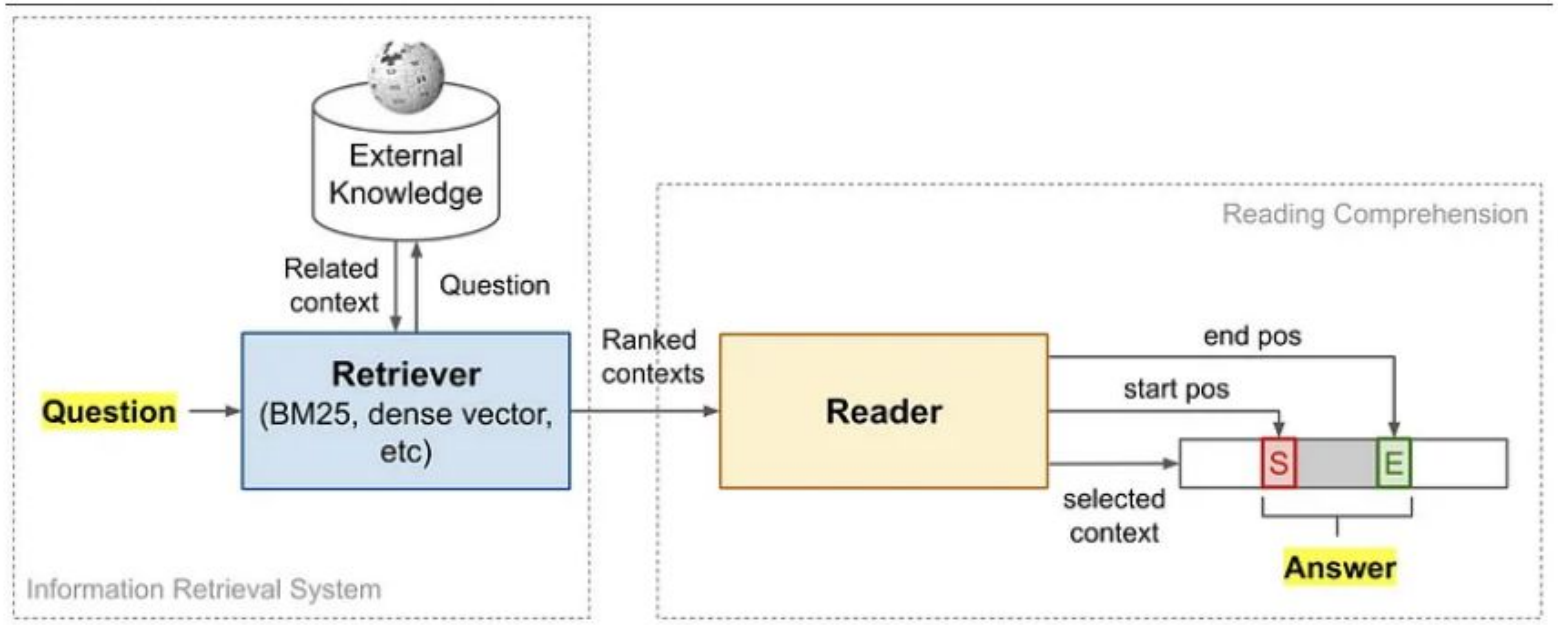
A retriever is responsible for retrieving the most relevant documents from a knowledge base that are likely to contain the answer to a given question i.e. it retrieves a set of contexts for our QA model to extract answers.



Question Answering

Reader

A reader is responsible for extracting the answer from the retrieved documents/set of Contexts. Using the Reader's model alone solves the reading comprehension (RC) QA task, but if it is used with the Retriever model, we have the Open-Domain QA model.

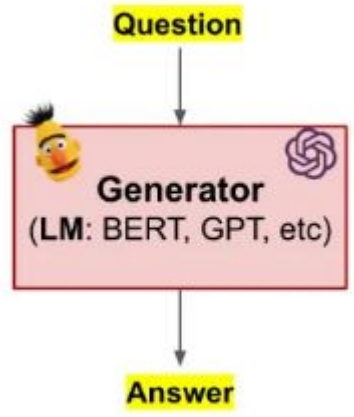


Question Answering - Components

PES UNIVERSITY

Generator

Generator does not extract the answer to a question from a context. Instead, it generates text directly to answer the question. This is done by using a language model, such as GPT-3. The language model is trained on a large dataset of text, and it can be used to generate text that is relevant to the question.The context can be optionally provided to the model. If the context is provided, the model will use the context to generate a more informative and accurate answer. However, the model can also generate an answer without the context.

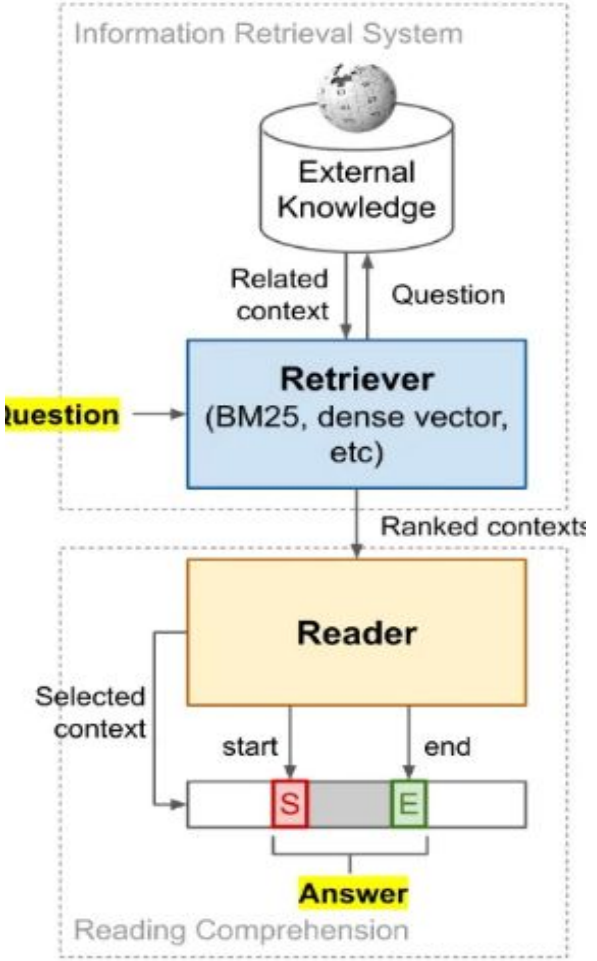


Generator

Question Answering - Combining Components to form variants

Retriever — Reader (Open-book Extractive QA)

The retriever takes the question and context, which is indexed in the database. It creates a query vector(vector representation) that is compared against all indexed context vectors in the database and returns the (n) most similar contexts. The reader model then takes in the most similar contexts and the question to provide a span selection of the answer.

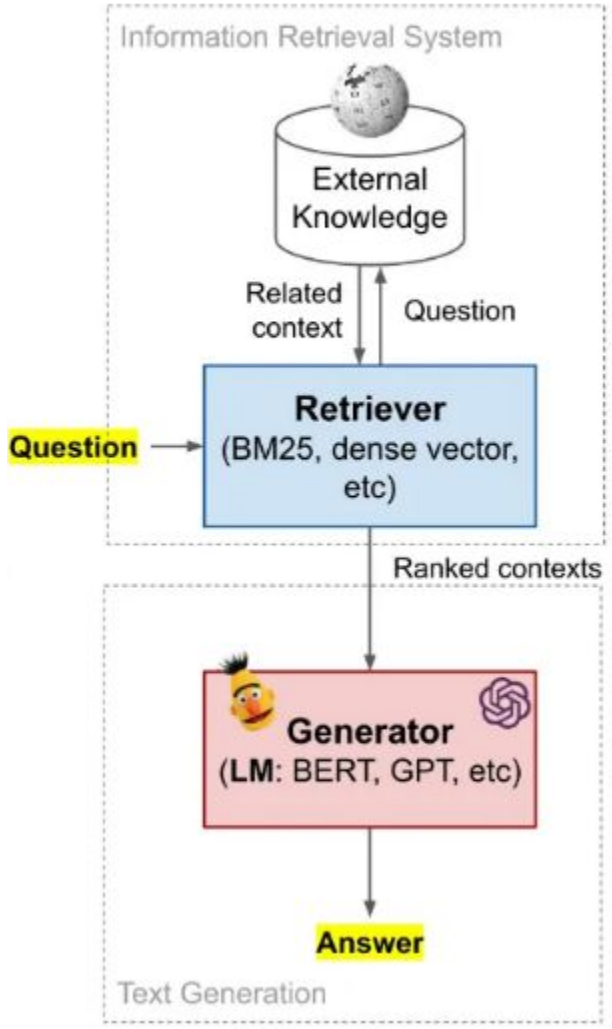


Retriever-Reader

Question Answering

Retriever — Generator (Open-book Abstractive QA)

In this system, reader model is replaced with a generator model. The generator model utilizes the question, along with the provided contexts and knowledge, to produce answers. This approach is referred to as abstractive question-answering and is termed “open-book” because the data source is external.The system is employed to address more conceptual questions that do not demand precise responses. Consequently, the answers it provides are more in the form of suggestions or opinions rather than direct, verbatim answers.

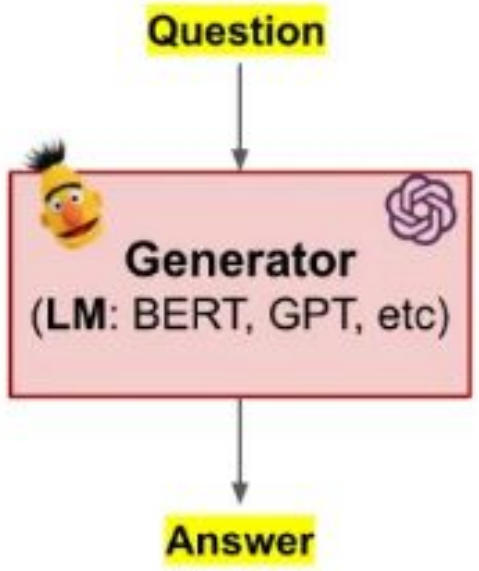


Retriever-Generator

Question Answering

Generator (Closed-book Abstractive QA)

This architecture is known as closed-book abstractive QA since it relies solely on the model's internal knowledge to generate answers. The retriever and reader models are removed, and the generative model handles the answers.



Generator

References

"Speech and Language Processing", Third Edition, Daniel Jurafsky, James H.Martin, Chapter 14: Question Answering and Information Retrieval, 2023.

https://iprathore71.medium.com/q-a-model-76957da40e07

https://spotintelligence.com/2023/01/20/question-answering-qa-system-nlp/

PES UNIVERSITY

THANK YOU

Surabhi Narayan

Department of Computer Science Engineering

PES UNIVERSITY

ALGORITHMS FOR INFORMATION RETRIEVAL AND INTELLIGENT WEB

Question Answering, Neural Models for IR

Surabhi Narayan

Department of Computer Science Engineering

Slides collated by:

Gaurav Mahajan (TA VIII sem)

Anshula Aithal (TA VIII sem)

ALGORITHMS FOR INFORMATION RETRIEVAL AND INTELLIGENT WEB

Knowledge Based QA

Surabhi Narayan
Department of Computer Science Engineering

Knowledge Based QA

- While an enormous amount of information is encoded in the vast amount of text on the web, information 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 that answered questions from a structured database of baseball games and stats.

Knowledge Based QA

- There are two common paradigms are used for knowledge-based QA.
- The first, **graph-based QA**, models the knowledge base as a graph, often with entities as nodes and relations or propositions as edges between nodes.
- The second, QA by **semantic parsing**

Knowledge Based QA - Graph Based using RDF triple stores

- Let’s introduce the components of a simple knowledge-based QA system after entity linking has been performed.
- We’ll focus on the very simplest case of graph-based QA, in which the dataset is a set of factoids in the form of RDF triples, and the task is to answer questions about one of the missing arguments.
- An RDF triple is a 3-tuple, a predicate with two arguments, expressing some simple relation or proposition.

Knowledge Based QA - Graph Based using RDF triple stores

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?”.

Knowledge Based QA - Graph Based using RDF triple stores

A number of such question datasets exist.

SimpleQuestions contains 100K questions written by annotators based on triples from Freebase. For example, the question ”What American cartoonist is the creator of Andy Lippincott?”. was written based on the triple (andy Lippincott, character created by, garry trudeau).

FreebaseQA, aligns the trivia questions from TriviaQA and other sources with triples in Freebase, aligning, for example, the trivia question “Which 18th century author wrote Clarissa (or The Character History of a Young Lady), said to be the longest novel in the English language?” with the triple (Clarissa, book.written-work.author, Samuel Richardson).

Knowledge Based QA - Graph Based using RDF triple stores

Another such family of datasets starts from WEBQUESTIONS, which contains 5,810 questions asked by web users, each beginning with a wh-word, containing exactly one entity, and paired with handwritten answers drawn from the Freebase page of the question's entity.

WEBQUESTIONSSP 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.

Knowledge Based QA - Graph Based using RDF triple stores

Let's assume we've already done the stage of entity linking introduced in the prior section. Thus we've mapped already from a textual mention like Ada Lovelace to the canonical entity ID in the knowledge base. For simple triple relation question answering, the next step is to determine which relation is being asked about, mapping from a string like "When was ... born" to canonical relations in the knowledge base like birth-year. We might sketch the combined task as:

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

"What is the capital of England?" → `capital-city(?x, England)`

Knowledge Based QA - Graph Based using RDF triple stores

The next step is relation detection and linking. For simple questions, where we assume the question has only a single relation, relation detection and linking can be done in a way resembling the neural entity linking models: computing similarity (generally by dot product) between the encoding of the question text and an encoding for each possible relation.

Knowledge Based QA - Graph Based using RDF triple stores

Ranking of answers : Most algorithms have a final stage which takes the top j entities and the top k relations returned by the entity and relation inference steps, searches the knowledge base for triples containing those entities and relations, and then ranks those triples.

This ranking can be heuristic, for example scoring each entity/relation pairs based on the string similarity between the mention span and the entities text aliases, or favoring entities that have a high in-degree (are linked to by many relations).

Or the ranking can be done by training a classifier to take the concatenated entity/relation encodings and predict a probability.

Knowledge Based QA - Semantic Parsing

The second kind of knowledge-based QA uses a semantic parser to map the question to a structured program to produce an answer. These logical forms can take the form of some version of predicate calculus, a query language like SQL or SPARQL, or some other executable program

The logical form of the question is thus either in the form of a query or can easily be converted into one (predicate calculus can be converted to SQL, for example). The database can be a full relational database, or some other structured knowledge store

Knowledge Based QA - Semantic Parsing

Semantic parsing algorithms can be supervised fully with questions paired with a hand-built logical form, or can be weakly supervised by questions paired with an answer (the denotation), in which the logical form is modeled only as a latent variable.

For the fully supervised case, we can get a set of questions paired with their correct logical form from datasets like the GEOQUERY dataset of questions about US geography (Zelle and Mooney, 1996), the DROP dataset of complex questions (on history and football games) that require reasoning, or the ATIS dataset of flight queries, all of which have versions with SQL or other logical forms.

Knowledge Based QA - Semantic Parsing

The task is then to take those pairs of training tuples and produce a system that maps from new questions to their logical forms. A common baseline algorithm is a simple sequence-to-sequence model, for example using BERT to represent question tokens, passing them to an encoder-decoder.

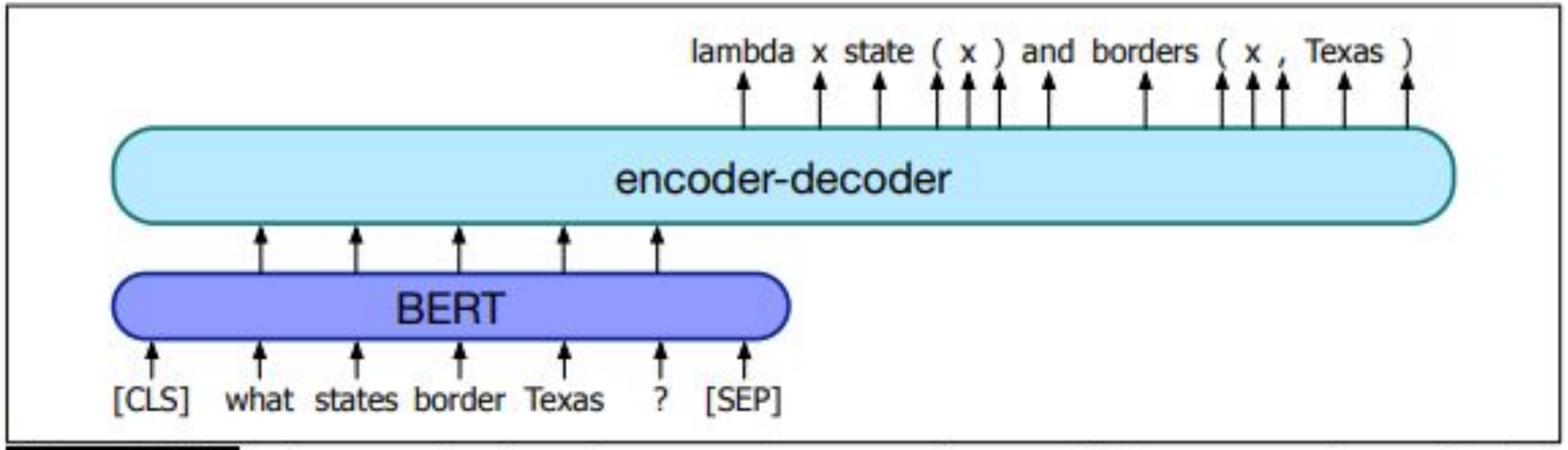


Figure 14.15 An encoder-decoder semantic parser for translating a question to logical form, with a BERT pre-encoder followed by an encoder-decoder (biLSTM or Transformer).

Knowledge Based QA - Semantic Parsing

PES UNIVERSITY

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))))

Sample logical forms produced by a semantic parser for question answering,

THANK YOU

Surabhi Narayan

Department of Computer Science Engineering

PES UNIVERSITY

ALGORITHMS FOR INFORMATION RETRIEVAL AND INTELLIGENT WEB

Architecture for Coreference Algorithms

Surabhi Narayan

Department of Computer Science Engineering

Slides collated by:

Gaurav Mahajan (TA VIII sem)

Anshula Aithal (TA VIII sem)

ALGORITHMS FOR INFORMATION RETRIEVAL

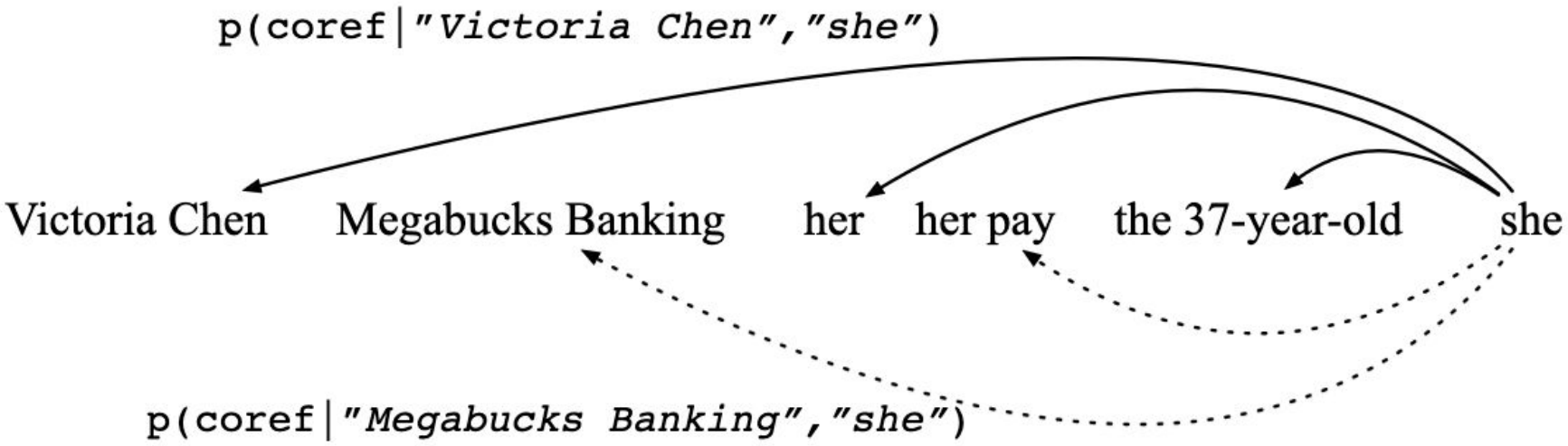
PES UNIVERSITY

Architecture for Coreference Algorithms

Dr. Surabhi Narayan
Department of Computer Science Engineering

Mention-Pair Algorithm

- The mention-pair architecture is based around a classifier that— as its name suggests—is given a pair of mentions, a candidate anaphor and a candidate antecedent, and makes a binary classification decision: coreferring or not.
- Lets refer to the previous example for the pronoun ‘she’:



Mention-Pair Algorithm

- For each prior mention (Victoria Chen, Megabucks Banking, her, etc.), the binary classifier computes a probability: whether or not the mention is the antecedent of she. We want this probability to be high for actual antecedents (Victoria Chen, her, the 38-year-old) and low for non-antecedents (Megabucks Banking, her pay).
- For training, we need a heuristic for selecting training samples; since most pairs of mentions in a document are not coreferent, selecting every pair would lead to a massive overabundance of negative samples. The most common heuristic, from is to choose the closest antecedent as a positive example, and all pairs in between as the negative examples.

Mention-Pair Algorithm

- More formally, for each anaphor mention m_i we create:
  - one positive instance (m_i, m_j) where m_j is the closest antecedent to m_i, and a negative instance (m_i, m_k) for each m_k between m_j and m_i
- Thus for the anaphor she, we would choose (she, her) as the positive example and no negative examples. Similarly, for the anaphor the company we would choose (the company, Megabucks) as the positive example and (the company, she) (the company, the 38-year-old) (the company, her pay) and (the company, her) as negative examples.

Mention-Pair Algorithm

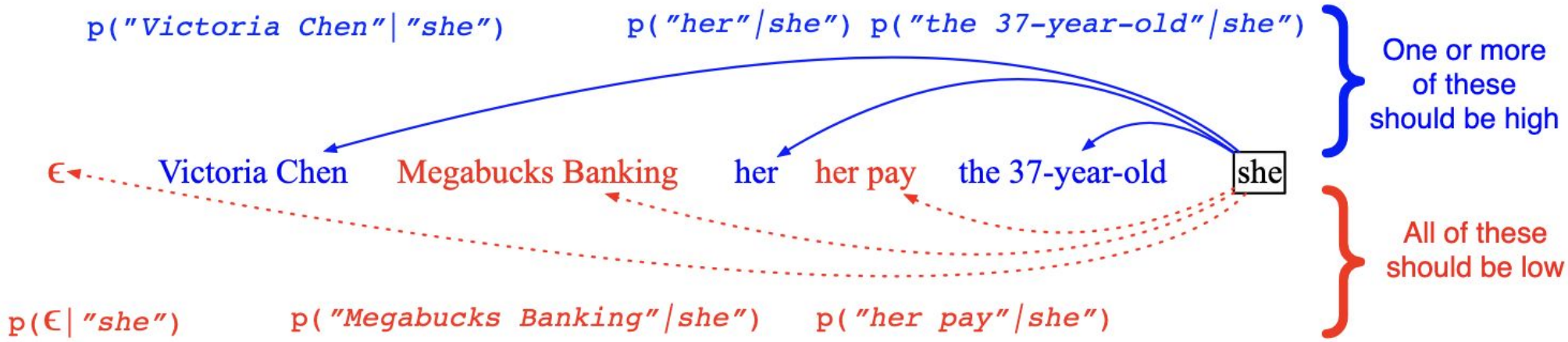
- While the mention-pair model has the advantage of simplicity, it has two main problems. First, the classifier doesn't directly compare candidate antecedents to each other, so it's not trained to decide, between two likely antecedents, which one is in fact better.
- Second, it ignores the discourse model, looking only at mentions, not entities. Each classifier decision is made completely locally to the pair, without being able to take into account other mentions of the same entity.

Mention Rank Algorithm

- The mention ranking model directly compares candidate antecedents to each other, choosing the highest-scoring antecedent for each anaphor.
- In early formulations, for a mention indexed as i, the classifier was responsible for determining which of the prior mentions {1, ..., i − 1} served as the antecedent . However, consider the case where mention i is not actually anaphoric, and none of the antecedents should be selected. In such situations, a model would need to run a separate anaphoricity classifier on i.
- Instead, a more effective approach is to jointly learn anaphoricity detection and coreference together with a single loss. This integrated approach avoids the need for explicit separate classifiers, allowing the model to simultaneously capture anaphoricity and coreference information.

Mention Rank Algorithm

- In modern mention-ranking systems, for the ith mention (anaphor), we have an associated random variable y_i ranging over the values Y (i) = {1, ..., i − 1, ϵ}. The value ϵ is a special dummy mention meaning that i does not have an antecedent (i.e., is either discourse-new and starts a new coref chain, or is non-anaphoric).



Mention Rank Algorithm

- At test time, for a given mention i the model computes one softmax over all the antecedents (plus ϵ) giving a probability for each candidate antecedent (or none).
- Once the antecedent is classified for each anaphor, transitive closure can be run over the pairwise decisions to get a complete clustering.
- Training in the mention-ranking model poses challenges compared to the mention-pair model. In this model, determining which of all possible gold antecedents to use for training each anaphor is complex. The best antecedent for each mention is latent, meaning that for each mention, there exists an entire cluster of legal gold antecedents to choose from.

Mention Rank Algorithm

- Early approaches utilized heuristics to select an antecedent. For instance, one heuristic involved choosing the closest antecedent as the gold antecedent, and considering all non-antecedents within a window of two sentences as negative examples.
- The simplest approach involves giving credit to any legal antecedent by summing over all of them, employing a loss function that optimizes the likelihood of all correct antecedents from the gold clustering.

Entity-Based Model

- Both the mention-pair and mention-ranking models make their decisions about mentions. By contrast, entity-based models link each mention not to a previous mention but to a previous discourse entity (cluster of mentions).
- A mention-ranking model can be turned into an entity-ranking model simply by having the classifier make its decisions over clusters of mentions rather than individual mentions
- For traditional feature-based models, this can be done by extracting features over clusters. The size of a cluster is a useful feature, as is its ‘shape’, which is the list of types of the mentions in the cluster i.e., sequences of the tokens (P)roper, (D)efinite, (I)ndefinite, (Pr)onoun, so that a cluster composed of {Victoria, her, the 38-year-old} would have the shape P-Pr-D

Entity-Based Method

- An entity-based model that includes a mention-pair classifier can use as features aggregates of mention-pair probabilities, for example computing the average probability of coreference over all mention-pairs in the two clusters.
- Neural models can learn representations of clusters automatically, for example by using an RNN over the sequence of cluster mentions to encode a state corresponding to a cluster representation, or by learning distributed representations for pairs of clusters by pooling over learned representations of mention pairs.
- However, although entity-based models are more expressive, the use of cluster-level information in practice has not led to large gains in performance, so mention-ranking models are still more commonly used.

THANK YOU

Surabhi Narayan

Department of Computer Science Engineering