



Carnegie Mellon University
Language
Technologies
Institute

11-411/11-611 Natural Language Processing

Information Retrieval and Question Answering

David R. Mortensen

January 19, 2023

Language Technologies Institute

Information Retrieval

An Example of Information Retrieval

You run the customer support division of a software company, Glonk. You are deluged by support requests because your software is really bad. You have written up answers to many common questions (thousands of answers) and you want customers to be able to find the answers that address their problems.

You want to build a search engine for technical support documents.

query Why does Glonk Pro crash when I use accented letters?

- results**
1. **Glonk Pro support for unicode.** Glonk Pro does not support special characters other than those ...
 2. **Glonk Pro crashes on unexpected input.** A recent update fixes a behavior of Glonk Pro in which...
 3. **Glonk Lite support for non-English characters.** Glonk Lite is not designed for use with non-English text...
 4. ...

Information Retrieval

The task of information retrieval (IR):

- Given
 - A collection of documents
 - A query
- Return
 - A ranking of the set of documents

Overview of IR

Here is a simple approach to IR:

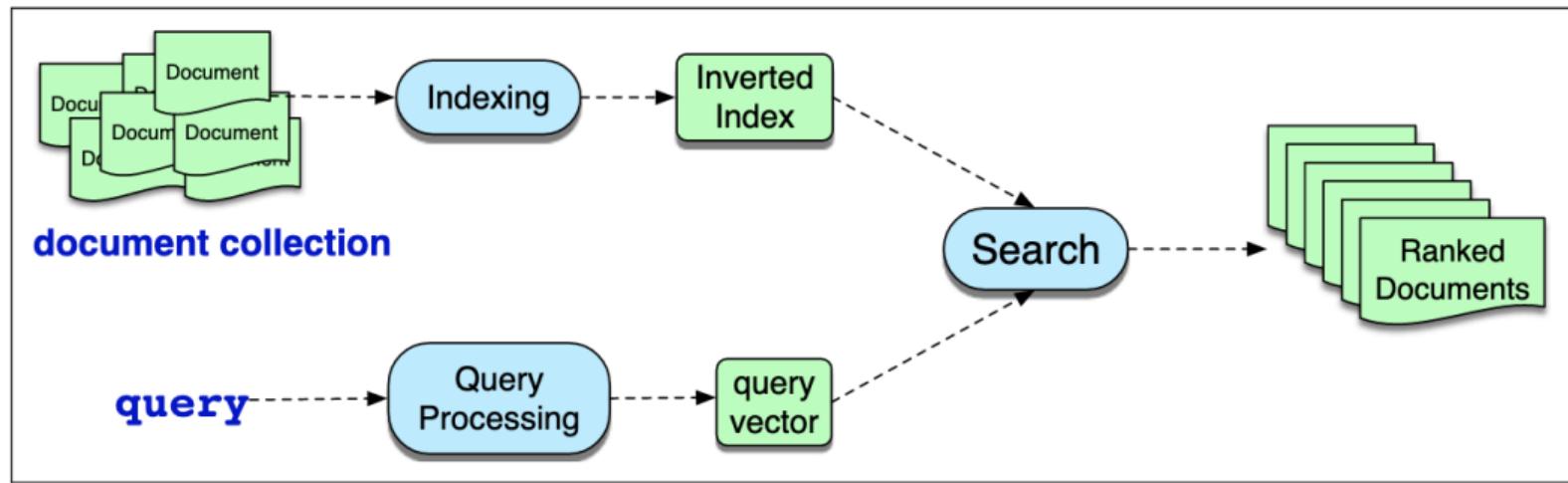


Figure 23.1 The architecture of an ad hoc IR system.

Documents Can Be Represented as Bags of Words

A BAG OF WORDS is a BAG or MULTISET of the words in a document.

- Like a set, except that identical elements can appear multiple times
- You could also think of it like a `Counter` object in Python

```
bow = {'and': 23502,  
       'or': 12342,  
       'the': 54939508,  
       ...  
       'hippopotamus': 1}
```

- If you take out sequencing information, any document can be viewed as a bag of words.

Queries Can Also Be Represented as Bags of Words

- Natural language queries are like little documents
- “What is the name of the current king of France?”

```
query = {'the': 2,  
         'of': 2,  
         'what': 1,  
         'is': 1,  
         'name': 1,  
         'current': 1,  
         'king': 1,  
         'France': 1}
```

Bags of Words Can Be Represented as Sparse Vectors

- So far, we've represented bags of words as dense MAPS, but sometimes it is useful to represent them as SPARSE VECTORS
- Let V be the set of all words in the document collection D . Then our BoW vectors for documents in D will have $|V|$ dimensions.
- Each dimension corresponds to a word **type** and the value at this dimension corresponds to the number of TOKENS of that type in the document that the vector is representing

$$\mathbf{v} = [0, 0, 0, 1, 0, 0, 0, 0, 0, 2, 0, \dots, 0]$$

Queries and Documents are Similar if their Vector Representations are Similar

We represent documents and queries as vectors because it is easy to compute the similarity between them.

Goal Compute similarity between query vector and a document vector

Means Many possibilities

- Dot product
- Euclidean distance
- COSINE SIMILARITY
- Other distance metrics

Cosine Similarity between Vectors



- The Euclidean distance between the blue vector and the yellow vector is much shorter than the Euclidean distance between the blue vector and the red vector
- The angle θ between the blue vector and the yellow vector is much greater than the angle β between the blue vector and the red vector
- Cosine similarity cares about angles, not about Euclidean distance

Computing the Similarity between BoW Vectors

- DOT PRODUCT (simplest metric):

$$a \cdot b = \sum_{i=1}^n a_i b_i \quad (1)$$

It is sensitive to the magnitude of the vectors (e.g., absolute word counts)

- COSINE SIMILARITY (better for our purposes):

$$S_c(a, b) = \frac{a \cdot b}{\|a\| \|b\|} = \frac{\sum_{i=1}^n a_i b_i}{\sqrt{\sum_{i=1}^n a_i^2} \sqrt{\sum_{i=1}^n b_i^2}} \quad (2)$$

Dot product divided by the norms of the two vectors.

It captures only the similarity in the **angle** or **direction** of the vector.

But There Is a Problem

The

(and *a, of, to*, etc.)

Some words are very common in a given document because they are common across all documents (e.g., *the*). **They are not discriminative.**

tf-idf Controls for Frequent but Uninformative Words

tf-idf (product of term frequency and inverse document frequency) addresses this.
Term frequency is defined as:

$$tf_{t,d} = \text{count}(t, d) + 1 \quad (3)$$

Where t is a term and d is a document. Inverse document frequency is defined as:

$$idf_f = \log_{10} \frac{N}{df_t} \quad (4)$$

where N is the number of documents and df_t is the number of documents in which term t occurs. Combined, we get tf-idf:

$$\text{tf-idf}(t, d) = tf_{t,d} \cdot idf_t \quad (5)$$

Applying tf-idf to Query and Document Vectors

When applied to computing the similarity between a query vector q and a document vector d :

$$\text{score}(q, d) = \sum_{t \in q} \frac{\text{tf-idf}(t, q)}{\sqrt{\sum_{q_i \in q} \text{tf-idf}^2(q_i, q)}} \cdot \frac{\text{tf-idf}(t, d)}{\sqrt{\sum_{d_i \in d} \text{tf-idf}^2(d_i, d)}} \quad (6)$$

Applying Simplifying Assumptions to tf-idf Scoring

Mathematically, **tf-idf** is rather ill-founded, but it works. We can make it even less well-founded but computationally cheaper (without hurting performance much) by simplifying the scoring formula:

$$\text{score}(q, d) = \sum_{t \in q} \frac{\text{tf-idf}(t, d)}{|d|} \quad (7)$$

This is used in practice.

Some Factors are Missing from tf-idf

- Weight placed on each term
- Document length

Term Frequency Weight



VS.



Is a document that mentions hippos eight times more relevant to the query “hippopotamus” than one that mentions hippos four times?

Scaling by Document Length

Query: *perceptron*

Document A Short document in which *perceptron* occurs once

Document B Long document in which *perceptron* occurs once

How much more relevant is Document A than Document B?

BM25 Is More Powerful than tf-idf

BM25 incorporates these as hyperparameters. BM25 has the same basic structure as tf-idf, but adds k and b :

$$\text{BM25}(d, q) = \sum_{t \in q} \log\left(\frac{N}{\text{df}_t}\right) \frac{\text{tf}_{t,d}}{k(1 - b + b(\frac{|d|}{\text{avgdl}})) + \text{tf}_{t,d}} \quad (8)$$

Two hyperparameters k and b

- k : term frequency weight
- b : scaling by document length

avgdl is average document length.

Evaluating IR Systems with Precision and Recall

NLP works because of metrics that allow us to compare the performance of different systems.

Three of the most important metrics are Precision, Recall, and (their harmonic mean) F1:

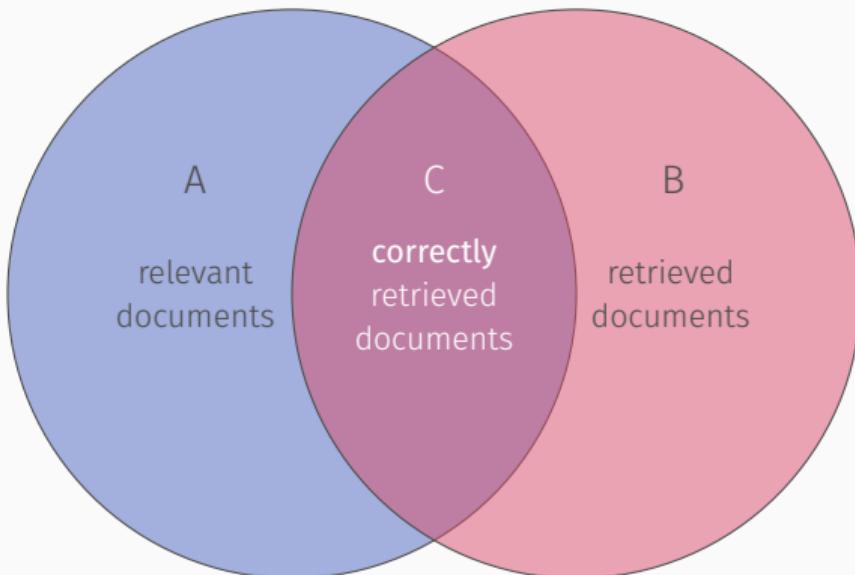
$$\text{precision} = \frac{TP}{TP + FP} \quad (9)$$

$$\text{recall} = \frac{TP}{TP + FN} \quad (10)$$

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \quad (11)$$

where TP = true positives, FP = false positives, and FN = false negatives

Precision and Recall Illustrated



A Venn diagram can simplify precision and recall considerably:

- A documents that are relevant to the query
- B documents that are retrieved
- C is the intersection between these two sets

$$\text{precision} = \frac{|C|}{|B|} \quad (12)$$

$$\text{recall} = \frac{|C|}{|A|} \quad (13)$$

A Starting Point for Evaluation: Precision@K

A very simple metric for evaluating ranked retrieval systems is precision@ K :
How many relevant items are found in your top- K results?

We can write this as precision_r or precision_{@ K}

By itself, Precision@K is a very unstable metric.

Mean Average Precision (MAP) Can Be Used to Evaluate Ranked Retrieval

For ranked retrieval, we need a different metric. First, consider average precision (*AP*):

$$AP = \frac{1}{|R_r|} \sum_{d \in R_r} \text{precision}_r(d) \quad (14)$$

Where R_r is the set of relevant documents at or above r . We can then compute Mean Average Precision (*MAP*):

$$MAP = \frac{1}{|Q|} \sum_{q \in Q} AP(q) \quad (15)$$

where Q is the set of queries. This allows us to evaluate our rankings of the documents—not just whether the documents we have retrieved are relevant—by rewarding us for having more relevant documents at the top of the ranking

Discounted Cumulative Gain is an Insufficient Metric for Ranked Retrieval

Gain G is the relevance score for each item recommended. Cumulative gain at K is the sum of the gains of the first K items:

$$CG_{@K} = \sum_{i=1}^K G_i \quad (16)$$

Discounted cumulative gain weights each relevance score based on its position:

$$DCG_{@K} = \sum_{i=1}^K \frac{G_i}{\log_2(i + 1)} \quad (17)$$

Normalized Discounted Cumulative Gain is Another Metric for Ranked Retrieval

Unfortunately, this score depends on the number of items recommended. Instead, we use Normalized Discounted Cumulative Gain (nDCG):

$$NDCG_{@K} = \frac{DCG_{@K}}{IDCG_{@K}} \quad (18)$$

where IDCG is the DCG for the ground truth:

$$IDCG_{@K} = \sum_{i=1}^{K^{\text{ideal}}} \frac{G_i^{\text{ideal}}}{\log_2(i+1)} \quad (19)$$

IR with Dense Vectors

Sparse vectors may have been fine in 1980, but we can do better now by representing queries and documents as dense vectors, using self-supervised contextual language models like BERT (You will learn more about BERT in Module 4)

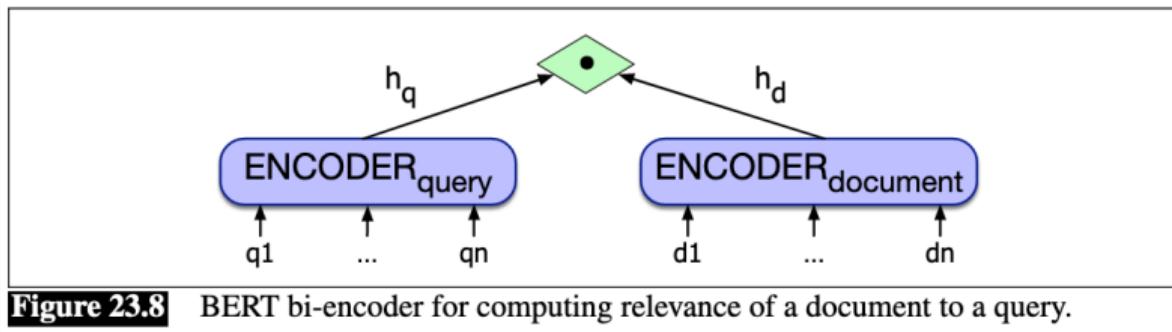


Figure 23.8 BERT bi-encoder for computing relevance of a document to a query.

This solves the **vocabulary mismatch problem**: queries and documents about the same topic sometimes use different, synonymous words and words may have different meanings depending on their context. More about this in a moment.

Question Answering

The Task of Question Answering

- Given:
 - A collection of documents
 - A question from a user
- Return:
 - A short passage containing the answer, or
 - A set of short passages ranked according to how likely they are to contain the answer

Question	Answer
What is the second largest city in Pennsylvania?	Pittsburgh
After whom is Pittsburgh named?	Sir William Pitt
What is the highest-ranked university in Pittsburgh?	Carnegie Mellon University
When was Fort Pitt founded?	November 27, 1758

Question Answering Can Be Seen as an IR Problem

Given a question and a corpus of documents

- Retrieve a set of documents relevant to the question
- Retrieve the set of passages from these documents relevant to the question
- Return answers extracted from these passages

Traditional IR-based QA Pipeline

This is how we did it in the old days:

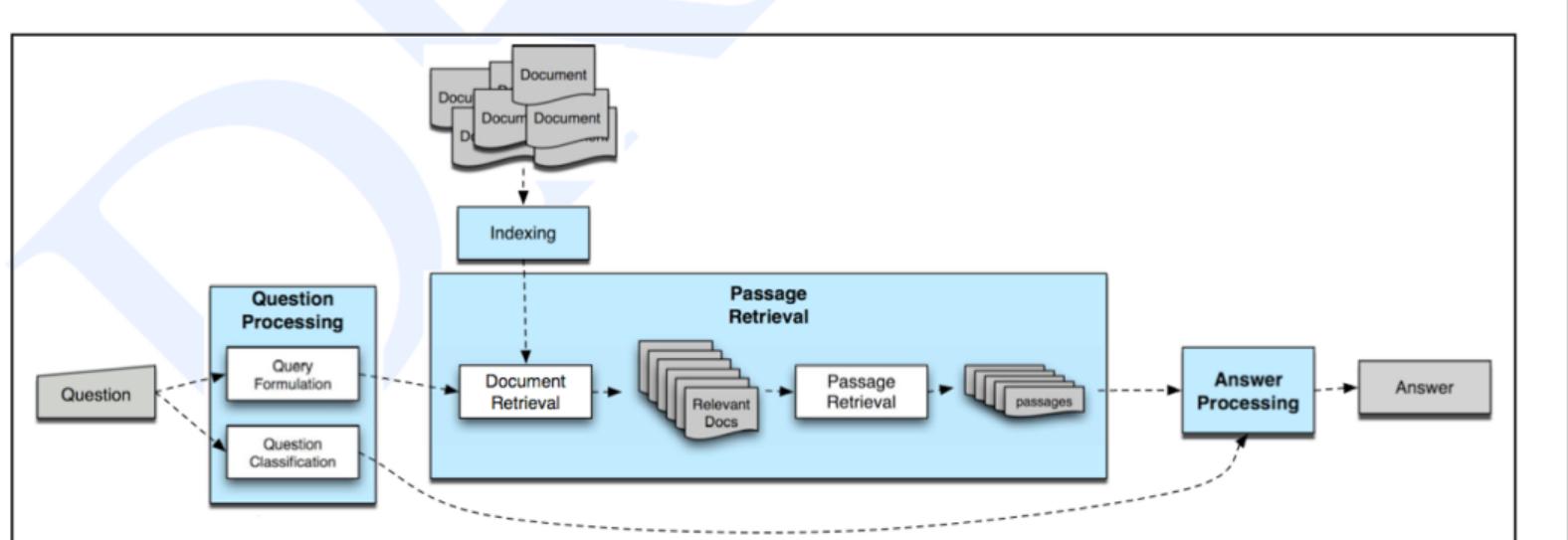


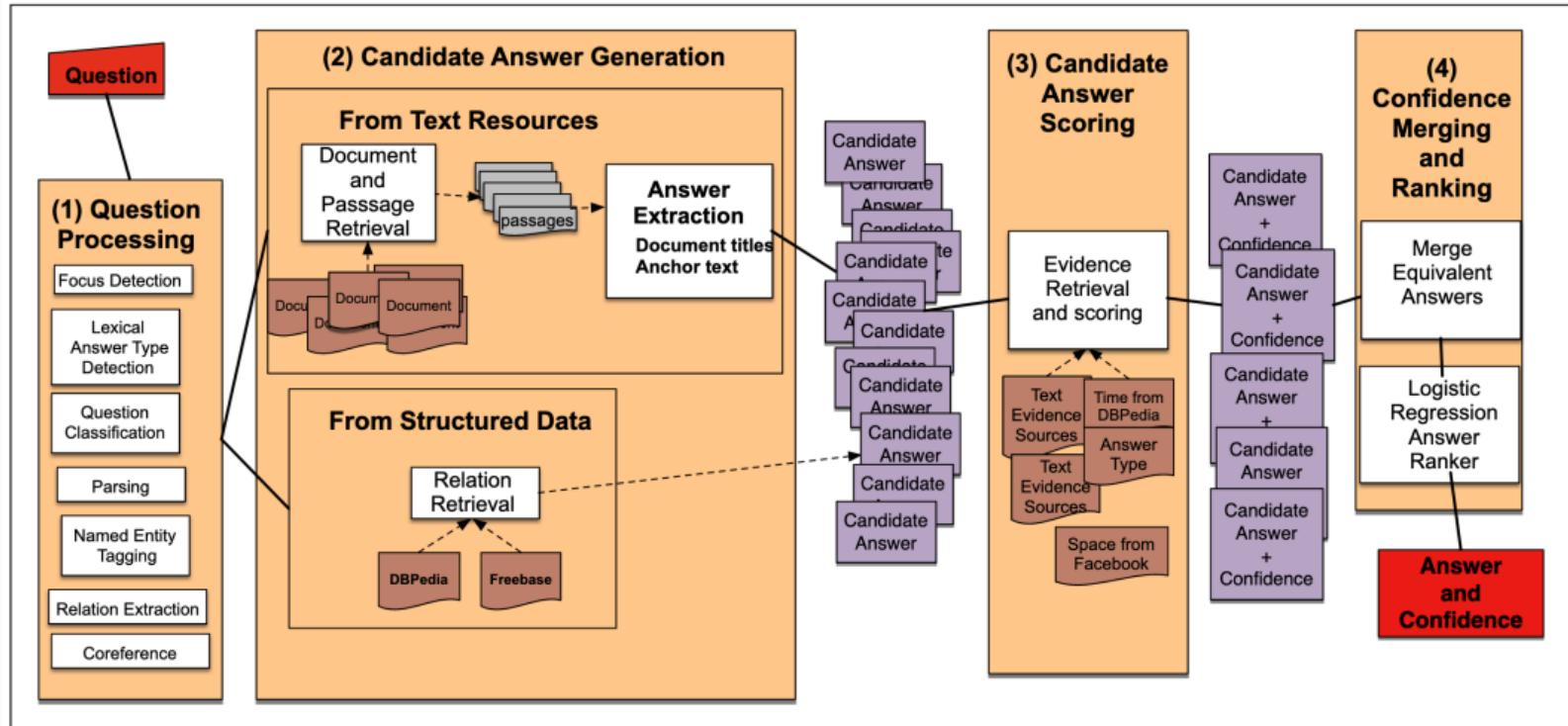
Figure 23.8 The 3 stages of a generic question answering system: question processing, passage retrieval, and answer processing..



Components of Traditional QA Pipeline (Baseline)

- Question Processing
 - Classify question according to type (for answer processing)
 - Convert question to vector
- Indexing
 - Represent documents as indexed vectors
- Passage Retrieval
 - Retrieve documents based on similarity to question (e.g. tf-idf)
 - Segment retrieved documents into passages
 - Rank passages according to similarity to question vector
- Answer Processing
 - Extract answers from passages, using question type information

This is your primary baseline. You must improve on this approach in some way.



IR-Based QA with Dense Vectors

BERT-like models make much simpler pipelines possible:

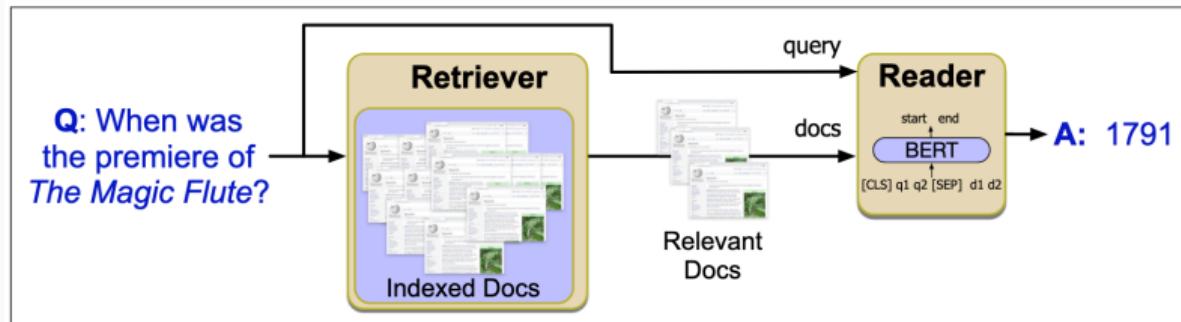


Figure 23.10 IR-based factoid question answering has two stages: **retrieval**, which returns relevant documents from the collection, and **reading**, in which a neural reading comprehension system extracts answer spans.

- A BERT-based retriever (see dense vector IR above) retrieves and ranks the relevant documents
- A BERT-based “reader” extracts the answers from the documents

BERT-Based Reader for QA

In extractive QA, the model identifies a span of a passage that corresponds to the answer. In modern QA systems, this is done with a Reader:

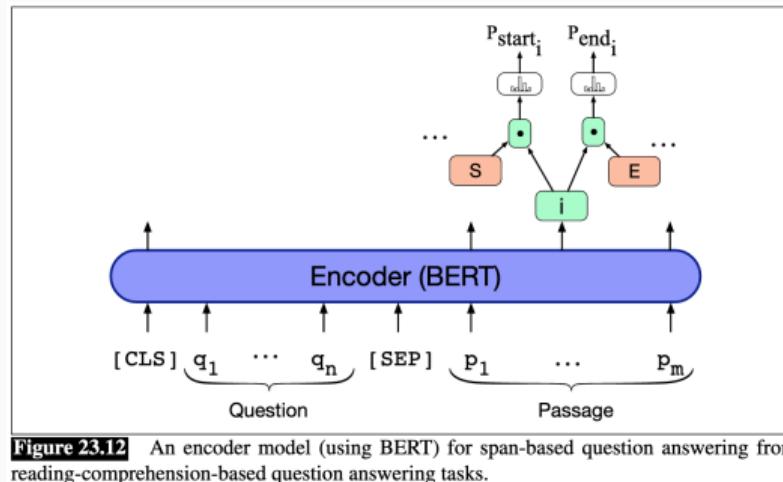


Figure 23.12 An encoder model (using BERT) for span-based question answering from reading-comprehension-based question answering tasks.

It consists of an encoder (e.g., BERT) with a linear layer added. The linear layer (more about this in Modules 3–4) has been fine-tuned to identify the start and end tokens of an answer span.

- Stanford Question Answering Dataset
- Questions from Wikipedia
- Answers are spans of Wikipedia articles
- In SQuAD 2.0, some questions are unanswerable
- About 150,000 questions

"The Joan B. Kroc Institute for International Peace Studies at the University of Notre Dame i...	"To whom was John B. Kroc married?"	{ "text": ["Ray Kroc"], "answer_start": [377] }
"The Joan B. Kroc Institute for International Peace Studies at the University of Notre Dame i...	"What company did Ray Kroc own?"	{ "text": ["McDonald's"], "answer_start": [360] }
"The library system of the university is divided between the main library and each of the..	"How many stories tall is the main library at Notre Dame?"	{ "text": ["14"], "answer_start": [136] }
"The library system of the university is divided between the main library and each of the..	"What is the name of the main library at Notre Dame?"	{ "text": ["Theodore M. Hesburgh Library"], "answer_start": [145] }
"The library system of the university is divided between the main library and each of the..	"In what year was the Theodore M. Hesburgh Library at Notre Dame finished?"	{ "text": ["1963"], "answer_start": [188] }
"The library system of the university is divided between the main library and each of the..	"Which artist created the mural on the Theodore M. Hesburgh Library?"	{ "text": ["Millard Sheets"], "answer_start": [344] }
"The library system of the university is divided between the main library and each of the..	"What is a common name to reference the mural created by Millard Sheets at Notre Dame?"	{ "text": ["Touchdown Jesus"], "answer_start": [394] }

- HotpotQA
- Questions are more difficult to answer than SQuAD
- Crowd workers were shown multiple context documents
- Asked to write question-answer pairs that required an understanding of all of the documents
- Suffers from SQuAD problem: annotators are likely to use words from the documents in the questions

- TriviaQA tries to address this problem by using questions that were not developed with the provided documents in mind
- Questions were developed by trivia enthusiasts
- Supporting documents were identified after the fact

Natural Questions Dataset

- Natural Questions
- Based on queries to the Google search engine
- Annotators are presented with a query and a Wikipedia page from the top search results
- They provide a paragraph-length answer and a short answer

- TyDi QA
- The aforementioned datasets are all in English
- TyDi QA has 204,000 questions from a diverse set of languages:
 - Arabic
 - Bengali
 - Kiswahili
 - Russian
 - Thai
 - etc.

Entity Linking

ENTITY LINKING is the task of associating mentions in text with the representations of real-world entities in an ontology. This ontology is often Wikipedia, in which case the term WIKIFICATION is sometimes used.

You will hear more about entity linking in Module 9

Knowledge-Based QA Is Useful for Structured Data

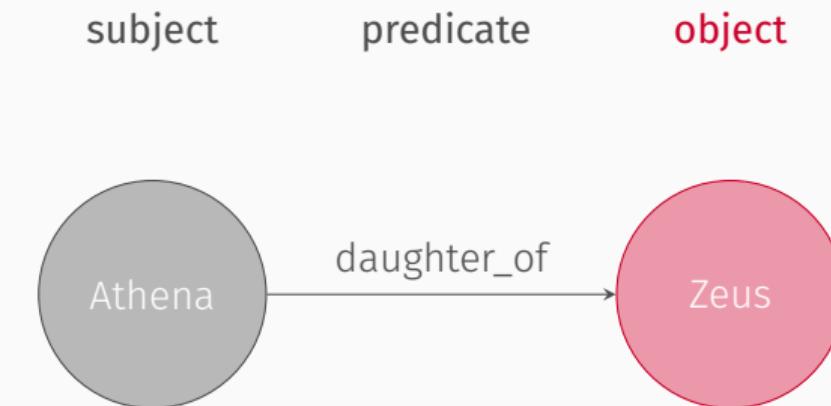
- For queries purely over natural language data, the IR-based methods described so far work well
- However, what if one also, or exclusively, wants to answer questions about structured databases (e.g., tabular data or knowledge graphs)?
- For these cases, knowledge-based QA is appropriate

Knowledge-Based QA Is not Extractive

- In IR-based QA, one typically tries to identify one or more spans in documents that contain answers
- It is **extractive**
- In KB-QA, answers are derived from structured knowledge (relational databases, triple stores, other kinds of graphs) and converted to natural language
- It is **abstractive** in a sense

What Is an RDF Triple?

- RDF triples consist of two entities and a relation that holds between them
- One such relation might be `daughter_of`
- Pairs that fall into such a relation:
 - $\langle \text{Athena}, \text{Zeus} \rangle$
 - $\langle \text{Leia}, \text{Anakin} \rangle$
 - $\langle \text{Liza Minnelli}, \text{Judy Garland} \rangle$
- The relation is stated as a PREDICATE
- The first entity is called the SUBJECT and the second entity is called the OBJECT
- A set of such triples forms a graph



Extracting RDF Triples

RDF triples can easily be extracted from structured data:

- Wikidata
- Freebase
- Relational databases

They can also be extracted from text data using RELATION EXTRACTION, about which more will be said in the semantics module

Using Triples to Answer Questions

Triples are useless unless their entities can be linked to the entities in questions. Thus, ENTITY LINKING is very important to KB-QA. Take the following question:

Who is the father of Athena?

To which Athena, in the knowledge graph/triple store, does “Athena” refer?

- Athena Chu?
- Athena Gana?
- Athena Faris?

Knowing this, the question can be converted into

`daughter_of(Athena, ?x)`

Where `?x` is a variable. This query can be made to the database.



Relation Detection

But how do you get from

Who is the father of Athena?

to

`daughter_of(Athena, ?x)`

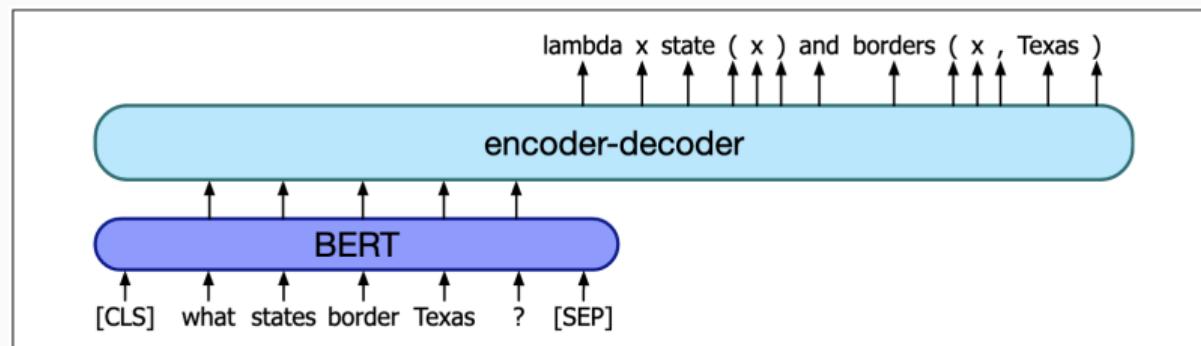
An important part of this is RELATION DETECTION. One common approach:

- You have an encoding of each possible relation (perhaps an embedding from a neural language model)
- You have an encoding of the question
- You compare these encodings, using the dot product, to find the best match

Semantic Parsing and Question Answering

Another kind of KB-QA: use a SEMANTIC PARSE (Module 8) that takes a question and returns a logical form:

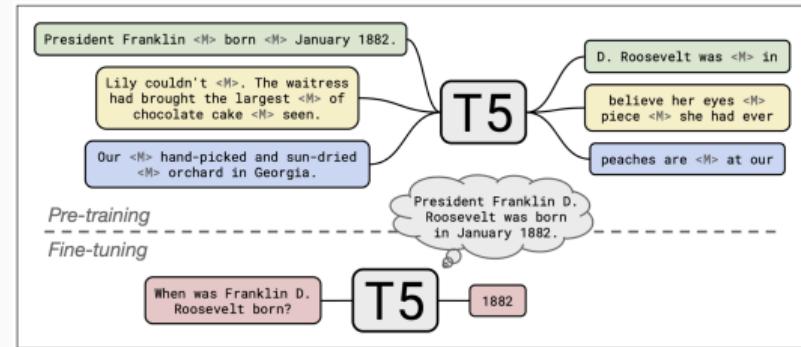
- First order logic (Module 8)
- SQL
- SPARQL



This logical form can then be used to query a structured database.

Second Baseline: Using LMs to do QA

- Large language models like T5, BERT, or GPT-3 store the answers to many questions in their parameters
- It is easy to build a QA system based solely upon this stored information
- T5, for example, is pre-trained to “fill in” deleted tokens (indicated by <M>)
- It can be fine-tuned to return an answer, given a question
- This type of model can be used as a baseline for your project but cannot be the basis of your QA system



Mean Reciprocal Rank is a Metric for QA

The most common metric for evaluating QA is Mean Reciprocal Rank, or MRR. It assumes that QA systems output a series of questions ranked according to confidence.

$$\text{MRR} = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{\text{rank}_i} \quad (20)$$

where Q is the set of questions in the test set and rank is the first correct answer to the i th question in Q .

MRR is the sum of the reciprocal of the ranks of the first correct answers normalized by the number of questions in the test set.

Metrics for Reading Comprehension

- **Exact match** For each answer, determine whether the HYPOTHEZED answer is string-identical to the REFERENCE answer. **This is a coarse-grained metric.**
- F_1 For each answer, identify the words that are present in the hypothesis, the words that are present in the reference, and the overlap between the two. Use these values to compute precision, recall, and F_1 . There are various ways of averaging the results, which we will discuss at greater length in Module 3. **This is a fine-grained metric.**

Questions?