CMPE 259, INSTRUCTOR: JORJETA JETCHEVA

QUESTION ANSWERING

INTRODUCTION

- Question Answering (QA) is an approach humans use when searching for information, from other humans and from digital entities
- Question answering systems typically focus on factoid questions:
  - Answers are facts and usually short
  - Examples: “What is the capital of France?”, “What is the average height of people in the US?”
- The major question answering paradigms are:
  - Information-retrieval (IR) based QA (aka Open Domain QA)
  - Knowledge-based QA

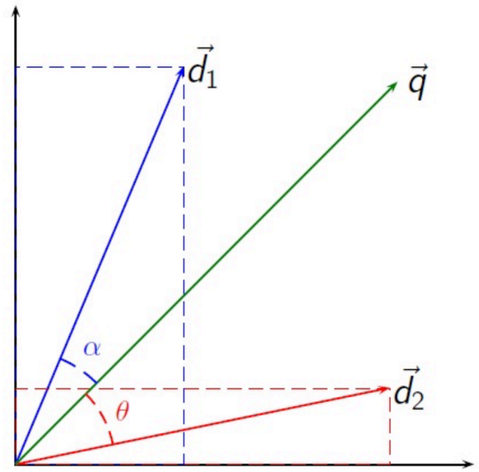
INFORMATION RETRIEVAL OVERVIEW

WHAT IS INFORMATION RETRIEVAL?

- Information Retrieval (IR) refers to retrieving any format media (text, images, etc.)
- An IR system is typically called a Search Engine
- Ad Hoc Retrieval: An IR task where the user poses a query, and the IR system returns an ordered/ranked set of documents from a document collection
  - A query is a specification of the user's information needs, e.g., a set of terms (words or phrases)
  - Collection is the set of documents the IR system uses to answer queries
  - A document is the unit that an IR system uses, web pages, news articles, scientific papers

BASIC IR ARCHITECTURE

- Queries and documents are mapped to vectors using unigram weighted word counts, typically tf-idf or the more powerful BM25
- Cosine similarity between the query and each of the document vectors is used to rank the documents
- Some IR systems use stop word removal, most do not (since term weighting already discounts stop words, and modern IR systems can handle the computation)



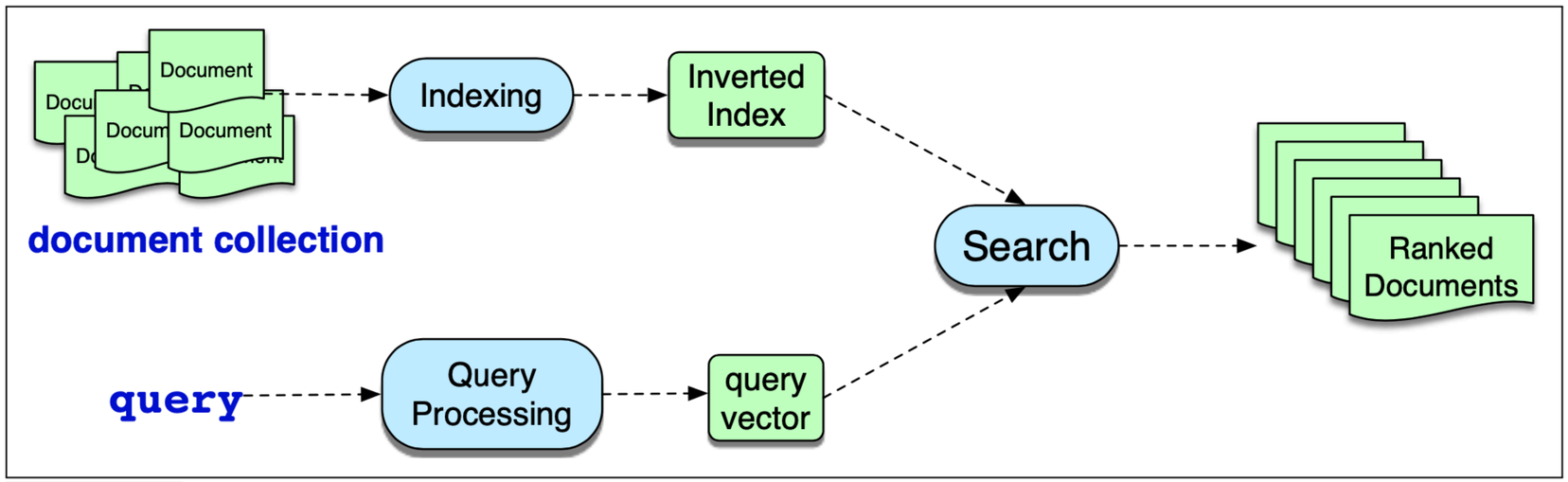


Figure 23.1 The architecture of an ad hoc IR system.

https://en.wikipedia.org/wiki/Vector_space_model

TF-IDF - REVIEW

- TF = Term Frequency:
  - The number of times a word (term **t**) appears in a given document (**d**): $tf_{t,d} = log_{10}(count(t,d)+1)$
  - Or can be the number of times term appears divided by the number of words in the document
  - Or can be binary, e.g., the word appears or does not appear in a document
- IDF = Inverse Document Frequency: $idf_t = log_{10}\frac{N}{df_t}$
  - Number of documents in the corpus (**N**) divided by the number of documents in which the particular word appears (df_t)
  - Reflects how much information the word provides (how rare or common it is across the corpus)
- TF-IDF: $tf\_idf(t,d) = tf(t,d) \times idf(t,d)$

TF-IDF INTUITION – REVIEW

IDF goes down as the ratio inside the log goes up $idf_t = log_{10}\frac{N}{df_t}$

$$tf\_idf(t,d) = tf(t,d) \times idf(t,d)$$

10 docs out of 100 contain the word => log(100/10) = 1 (word is rare)

100 docs out of 100 contain the word => log(100/100) = 0 (word is common)

- A high weight in tf–idf is reached by a high term frequency in the given document (high tf value), and a low frequency of the term across the whole collection of documents (high idf value)
  - TF-IDF emphasizes rare words that are strong indicators of a particular document type (e.g., medical content (“patient”) vs. space exploration content (e.g., “astronaut”))
  - And de-emphasizes common words that don’t help differentiate between documents (e.g., the word “can” will not help us differentiate between medical and space exploration content)

Note: Often +1 is added to the denominator in the idf computation to avoid the tf-idf being 0 for words that appear in all docs; other formulations exist as well.

DOCUMENT SCORING USING TF-IDF: OVERVIEW

- The score of a document d is computed as the cosine similarity between the query q and the document d:

$$score(q,d) = cos(\mathbf{q},\mathbf{d}) = \frac{\mathbf{q}\cdot\mathbf{d}}{|\mathbf{q}||\mathbf{d}|}$$

  which can be expressed the dot product of their normalized vectors:

$$score(q,d) = cos(\mathbf{q},\mathbf{d}) = \frac{\mathbf{q}}{|\mathbf{q}|}\cdot\frac{\mathbf{d}}{|\mathbf{d}|}$$

- The tf-idf version of this formula where the dot product is expressed as a summation is shown to the right
- Usually some simplifying assumptions are made:
  - The scores for all documents are divided by |q|, so removing it from the denominator does not change the results of comparing the scores of different documents
  - Each word in the query typically has a frequency of 1 and so the tf-idf computation for the query is typically ignored

$$score(q,d) = cos(\mathbf{q},\mathbf{d}) = \frac{\mathbf{q}}{|\mathbf{q}|}\cdot\frac{\mathbf{d}}{|\mathbf{d}|}$$

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

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

DOCUMENT SCORING USING TF-IDF: EXAMPLE

Query:	sweet love
Doc 1:	Sweet sweet nurse! Love?
Doc 2:	Sweet sorrow
Doc 3:	How sweet is love?
Doc 4:	Nurse!

	Document 1					Document 2				
word	**count**	**tf**	**df**	**idf**	**tf-idf**	**count**	**tf**	**df**	**idf**	**tf-idf**
love	1	0.301	2	0.301	0.091	0	0	2	0.301	0
sweet	2	0.477	3	0.125	0.060	1	0.301	3	0.125	0.038
sorrow	0	0	1	0.602	0	1	0.301	1	0.602	0.181
how	0	0	1	0.602	0	0	0	1	0.602	0
nurse	1	0.301	2	0.301	0.091	0	0	2	0.301	0
is	0	0	1	0.602	0	0	0	1	0.602	0
$\lvert d_1 \rvert = \sqrt{.091^2 + .060^2 + .091^2} = .141$						$\lvert d_2 \rvert = \sqrt{.038^2 + .181^2} = .185$				

Figure 23.2 Computation of tf-idf for nano-documents 1 and 2, using Eq. 23.3, Eq. 23.4, and Eq. 23.5.

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

Document 1 example: $(0.060 + 0.091)/0.141 = 1.07$

Doc	$\lvert d \rvert$	**tf-idf**(*sweet*)	**tf-idf**(*love*)	**score**
1	.141	.060	.091	1.07
3	.274	.038	.091	0.471
2	.185	.038	0	0.205
4	.090	0	0	0

Figure 23.3 Ranking documents by Eq. 23.9.

The results of this kind of unigram approach reveal lack of actual comprehension

BM25

- BM25 aka Okapi BM25
  - BM stands for Best Matching
  - Okapi was the first system to use BM25

BMI score of document ***d*** *given query* ***q***

$$\sum_{t \in q} \overbrace{\log\left(\frac{N}{df_t}\right)}^{\text{IDF}} \overbrace{\frac{tf_{t,d}}{k\left(1-b+b\left(\frac{|d|}{|d_{\text{avg}}|}\right)\right)+tf_{t,d}}}^{\text{weighted tf}}$$

$|d_{avg}|$ *is the length of the average document*

- BM25 is a modification of TF-IDF, where
  - k adjusts the balance between TF and IDF
    - When k is large, we approach raw term frequency (and ignoring idf) and when $k = 0$, results in ignoring term frequency
  - b controls the importance of document length normalization
    - We set $b = 1$ to normalize by the whole doc length; $b = 0$ means no scaling
  - Empirically suggested values that work well : $k = [1.2,2]$ and $b = 0.75$

INVERTED INDEX

- Basic search problem in IR: find all documents $d \in C$ that contain a term $q \in Q$
- To efficiently perform this search, we use an inverted index
- Given a query term, the inverted index outputs a list of documents that contain the term
- The inverted index has the following elements:
  - Dictionary: a list of terms, each with a pointer to a list of postings
    - Could include document frequency for the terms
  - Postings list: a list of document IDs associated with each term
    - Could include the term frequency and positions of the term within the respective document
- There are many other alternatives to the inverted index approach

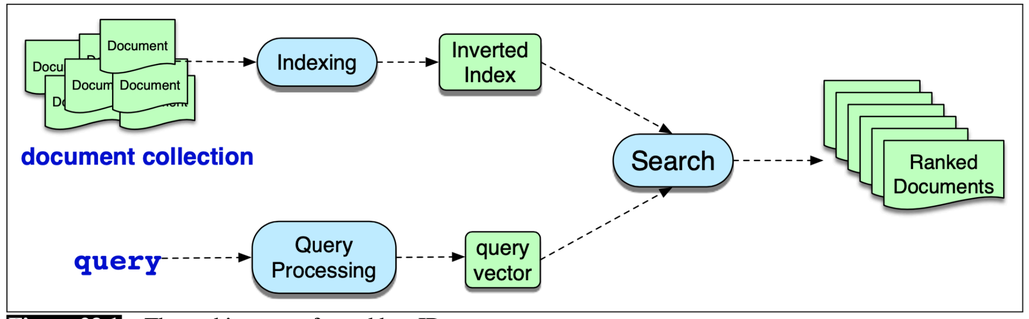
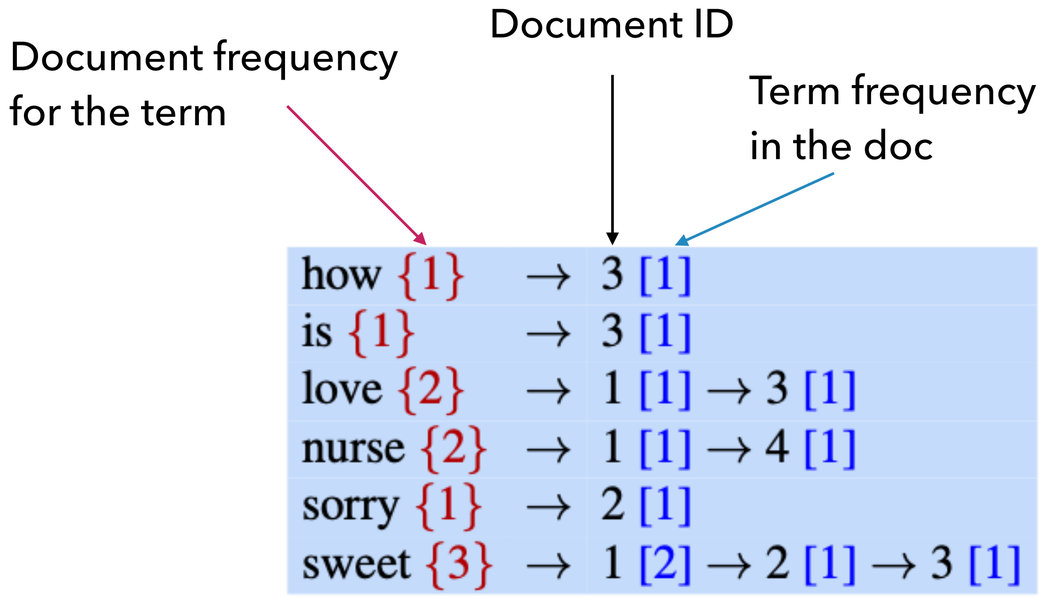


Figure 23.1 The architecture of an ad hoc IR system.



EVALUATING AND COMPARING IR SYSTEMS

- Binary precision and recall can be used here where one class is “relevant” and the other is “not relevant” for the documents returned by the IR system
- However, when comparing 2 IR systems, we need to know which system has the relevant documents higher than the other, so we would use:
  - Precision as the fraction of relevant documents seen at a given rank
  - Recall as the fraction of relevant documents detected by a certain rank.

Rank	Judgment	Precision_{Rank}	Recall_{Rank}
1	R	1.0	.11
2	N	.50	.11
3	R	.66	.22
4	N	.50	.22
5	R	.60	.33
6	R	.66	.44
7	N	.57	.44
8	R	.63	.55
9	N	.55	.55
10	N	.50	.55

Assumption: there are 9 relevant documents

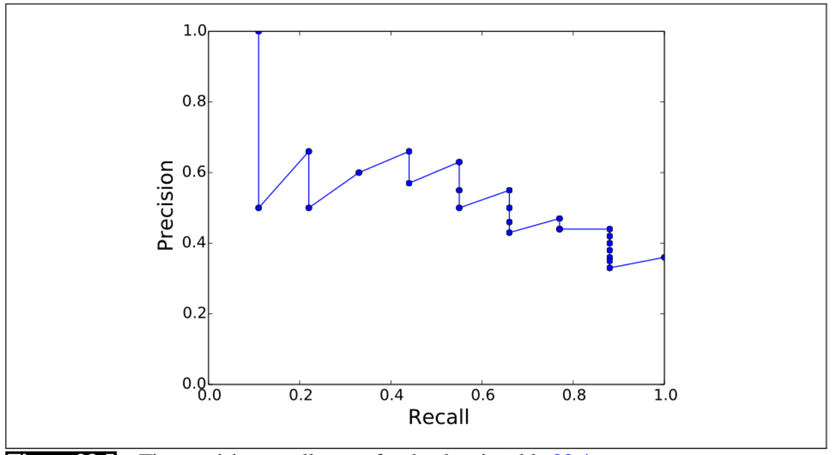


Figure 23.5 The precision recall curve for the data in table 23.4.

EVALUATING AND COMPARING IR SYSTEMS (CONT.)

- When comparing 2 IR systems across a set of test queries, we can interpolate the average precision (at each rank) at fixed recall thresholds from 0 to 100 (e.g., at intervals of 10) and plot the two curves
  - We interpolate by using the max precision across the current or previous ranks: $IntPrecision(r) = \max_{i \geq r} Precision(i)$

Rank	Judgment	$Precision_{Rank}$	$Recall_{Rank}$
1	R	1.0	.11
2	N	.50	.11
3	R	.66	.22
4	N	.50	.22
5	R	.60	.33
6	R	.66	.44
7	N	.57	.44
8	R	.63	.55
9	N	.55	.55
10	N	.50	.55

Assumption: there are 9 relevant documents

- Or we can use mean average precision (MAP), which is the average of all precision values where a relevant document has been found:
  - For a single query we have Average Precision at a rank r were R_r is the set of relevant documents at or above r: $AP = \frac{1}{|R_r|} \sum_{d \in R_r} Precision_r(d)$
  - For set of queries q in the set Q, we have the following formulation for MAP: $MAP = \frac{1}{|Q|} \sum_{q \in Q} AP(q)$

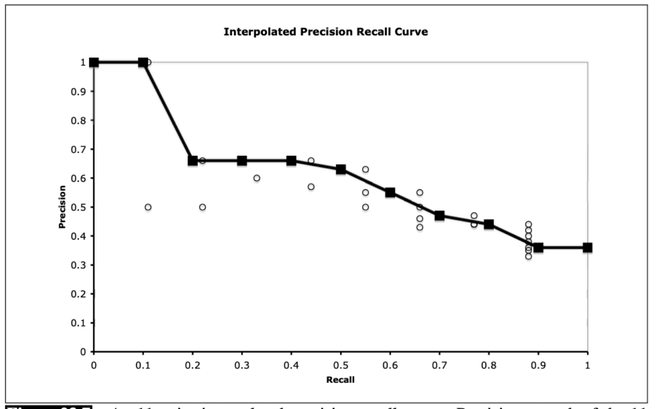


Figure 23.7 An 11 point interpolated precision-recall curve. Precision at each of the 11 standard recall levels is interpolated for each query from the maximum at any higher level of recall. The original measured precision recall points are also shown.

IR WITH DENSE VECTORS

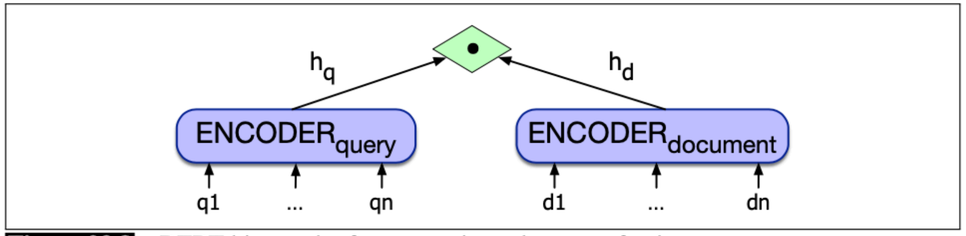


Figure 23.8 BERT bi-encoder for computing relevance of a document to a query.

- Classic IR approaches do exact keyword matching and cannot handle synonymous words (the vocabulary mismatch problem), e.g., "tragic love" and "star-crossed lovers" don't have matching words
- Solution: IR approaches with dense embeddings (e.g., from BERT), where we use a bi-encoder (two separate encoder models) to encode the query and document, and use dot product of the resulting vectors to compute the (relevance) score:

  $h_q = BERT_Q(q)[CLS]$ - we represent the query as the [CLS] token vector for the query encoder

  $h_d = BERT_D(d)[CLS]$ - we represent the document as the [CLS] token vector for the document encoder

  $score(d, q) = h_q \cdot h_d$
- This is a very active research area
  - Some approaches use average pooling across all BERT outputs instead of the CLS token
  - Others introduce weight matrices after the encoding or dot product steps
  - Explorations of how to do end-to-end training for the entire IR task (e.g., + and - examples)
  - How to efficiently find all nearby vectors to a dense vector is computationally challenging (nearest neighbor search usually not feasible in high dimensions; typically would use approximate nearest neighbor vector search)

IR-BASED QA

OVERVIEW

- IR-based QA (aka open domain QA) finds short text segments from the web, or other document corpuses
- Typically, IR-based QA follows a retrieve and read model
  - First we retrieve relevant passages from a text collection using a search engine/IR system
  - Then we apply a neural reading comprehension algorithm to find the span of text with the likely answer (given the query and a text passage)

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

Figure 23.9 Some factoid questions and their answers.

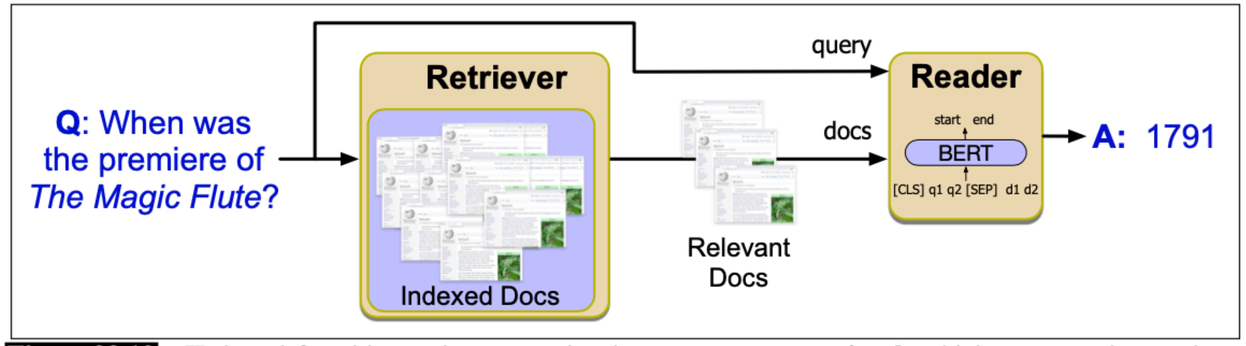


Figure 23.10 IR-based factoid question answering has two stages: **retrieval**, which returns relevant documents from the collection, and **reading**, in which a neural reading comprehension system extracts answer spans.

DATASETS FOR OPEN DOMAIN QA

- Reading comprehension datasets contain tuples of (passage, question, answer) labeled by humans
- The reading comprehension task involves
  - Applying the reading comprehension on a tuple of (question, passage) and
  - Returning an answer (a span of text)

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

Figure 23.11 A (Wikipedia) passage from the SQuAD 2.0 dataset (Rajpurkar et al., 2018) with 3 sample questions and the labeled answer spans.

- Stanford Question Answering Dataset (SQuAD) (2016) consists on passages from Wikipedia and associated questions and answers (spans of text)
  - Humans were asked to read a Wikipedia passage, come up with questions and then choose the answer (span of text) from within the passage
  - SQuAD 2.0 augments the original SQuAD with questions that are unanswerable (150K questions)

DATASETS FOR OPEN DOMAIN QA (CONT.)

- HotpotQA Dataset (2018) tries to make the questions more complex
  - Generated by showing crowd workers multiple context documents when coming up with questions
  - Answering a question intended to require reasoning over multiple documents
  - Multi-hop QA problems
- TriviaQA (2017) tries to make sure annotators are not influenced by the words in the passage they read before creating a question
  - Trivia enthusiasts created questions first (in the absence of any text passages) - 94k questions
  - Wikipedia and other docs are then used to annotate answers to the questions

DATASETS FOR OPEN DOMAIN QA (CONT.)

- Natural Questions dataset (2019)
  - Real Google queries (anonymized)
  - Annotators are given a query along with the top 5 Wikipedia pages (resulting from search)
  - And annotate both long (paragraph long) and short answers, or mark the Wikipedia page with Null if the answer is not contained in it
- TyDi QA dataset (2020)
  - 204K question-answer pairs from 11 linguistically diverse languages, including Arabic, Bengali, Kiswahili, Russian and Thai

OPEN DOMAIN QA READER (FOR ANSWER SPAN EXTRACTION)

- Given a passage, find the span of text where the answer to a question lies
  - Extractive QA - the answer is a span of text from the passage, e.g.,

    Question: *How tall is Mt Everest?*
    Passage: *Reaching 29,029 feet at its summit*
    Answer: *29,029*
  - Abstractive QA - where the system writes an answer to a question, not drawn exactly from the passage

ANSWER EXTRACTION OVERVIEW

- Answer extraction is modeled by span labeling:

  Given a question q of n tokens $q_1, q_2, \ldots, q_n$, and a passage p of m tokens $p_1, p_2, \ldots, p_m$, compute the the probability that each possible span a is the answer, i.e., $P(a \mid q, p)$

  Assuming a starts at index a_s and ends at index a_e, we can simplify the above as follows:

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

  For each token in the passage p_i, we would be computing

  - The probability that it is the start of the span, $p_{start}(i)$
  - The probability that it is the end of the span, $p_{end}(i)$

ANSWER EXTRACTION BASELINE APPROACH

A BERT encoder is the baseline approach for span labeling

- The Question and Passage tokens are concatenated with a SEP token in between
- 2 new (trainable) vectors are introduced: S (span-start embedding) and E (end-span embedding)
- For each passage token p_i, BERT computes an embedding p'_i,

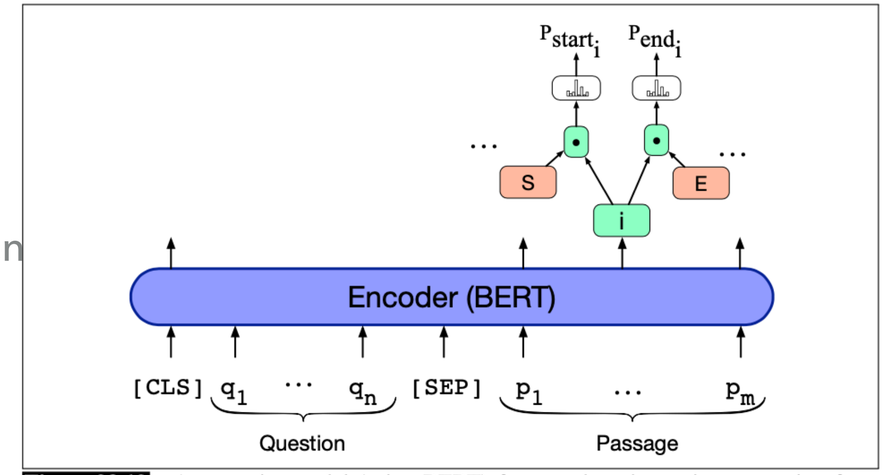


Figure 23.12 An encoder model (using BERT) for span-based question answering from reading-comprehension-based question answering tasks.

- To compute $p_{start}(i)$ and $p_{end}(i)$, we take the dot product of p'_i with S and E vector respectively, and pass it through a softmax function in order to normalize the computation over all tokens in the passage

$$P_{start_i} = \frac{exp(S \cdot p'_i)}{\sum_j exp(S \cdot p'_j)} \text{ and } P_{end_i} = \frac{exp(E \cdot p'_i)}{\sum_j exp(E \cdot p'_j)}$$

- The highest probability span is chosen as the answer ($P(a|q,p) = P_{start}(p_i|q,p)P_{end}(p_j|q,p)$)
- The training loss is computed based on the sum of the negative log-likelihoods of the correct start and end positions for each training example: $L = -logP_{start_i} - logP_{end_i}$
- For answers not contained in the passage, the start and end index will point at the CLS token

ANSWER EXTRACTION IN LONG PASSAGES

- The max input into BERT including the [CLS], and [SEP] tokens, and the n question tokens, is 512 tokens
- When we have passages that don't fit into the max input, i.e., $m > 512 - 2 - n$, we do the following:
  - Split the passage into pseudo-passages with length $m = 512 - 2 - n$, and do a prediction for each one (concatenated with the question, separated with [SEP] as usual)
  - For each passage, label with [CLS] the start and end for pseudo-passages that are not part of the answer, or with the correct indices if the pseudo passage contains part or all of the answer
  - Choose the span with the highest probability

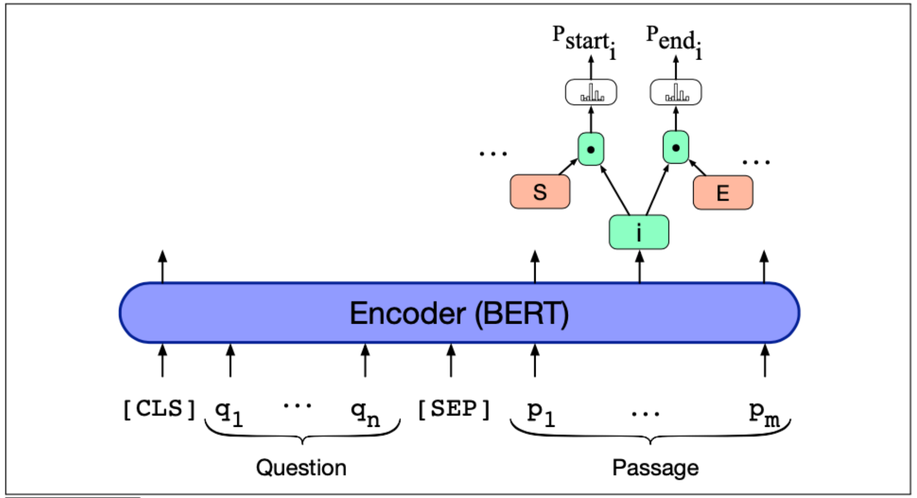


Figure 23.12 An encoder model (using BERT) for span-based question answering from reading-comprehension-based question answering tasks.

ENTITY LINKING

ENTITY LINKING OVERVIEW

- Entity linking associates a mention of a real-world entity in text with the representation of the entity in an ontology
- Wikipedia is the most common ontology for factoid questions
  - Each Wikipedia page serves as a unique ID for a particular entity
  - Wikification - the task of figuring out the page that matches an individual that is referenced in a piece of text
- Entity linking happens in two stages: mention detection & mention disambiguation
  - Traditional approach: anchor dictionaries and web graph
  - Modern approach: Neural graph-based linking

LINKING BASED ON ANCHOR DICTIONARIES AND WEB GRAPH: TAGME LINKER

- Each wikipedia page is considered to be a unique entity e
- The $TAGME$ linker (used for Wikipedia)
  - Creates a catalog of entities and imports them in a standard IR engine e.g., Lucene
  - For each page e, it computes an in-link count $in(e)$: the total number of in-links from other Wikipedia pages that point to e (can be extracted from Wikipedia dumps)
  - Has an anchor dictionary, which has a list of anchor texts for each Wikipedia page
    - Anchor texts are hyperlinked spans of text in pages that are in-links for this page
      - E.g., the following anchor text *San Jose State University* has an anchor link San Jose State University

LINKING BASED ON ANCHOR DICTIONARIES AND WEB GRAPH: TAGME LINKER (CONT.)

- The anchor dictionary for each page e contains:
  - e's title and all anchor texts from all the pages that point to e
  - The link probability of each anchor text a is $linkprob(a) = \frac{link(a)}{freq(a)}$, where $link(a)$ is the number of times a occurs as a link, and $freq(a)$ is the total frequency of a in Wikipedia (including non-anchor uses)
  - Anchors with low link probability are typically pruned from the dictionary for efficiency
- Mention Detection: $TAGME$ queries the anchor dictionary for each token sequence up to 6 words (pruning the rest)
  - Given the question *When was Ada Lovelace born?*, will result in anchors **Ada Lovelace** and **Ada**

    -Lovelace has a low link probability so it will be pruned

    -born will likely have been excluded from the anchor dictionary (due to very low link probability)
- Mention Disambiguation
  - A mention span may point to more than one entity, and thus require disambiguation

TAGME MENTION DISAMBIGUATION

- To deal with cases when a span may point to more than one entity, TAGME uses two factors: prior probability and relatedness/coherence
- Prior probability $p(e \mid a)$ is the probability that for each page $e \in \mathscr{E}(a)$, a points to e is computed as follows:

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

(ratio of the number of links into e with anchor text a, to the total number of occurrences of a as a link)

- Just looking at the prior is not enough, e.g., the highest prior for the question "What Chinese Dynasty came before the Yuan?" is for the currency, not the dynasty
  - This is where the relatedness factor comes in - we need to capture how well the candidate entity is related to the other entities in the input question

TAGME MENTION DISAMBIGUATION: RELATEDNESS FACTOR

- Given a question q, for each anchor span a detected in q, we assign a relatedness score, $rel(A, B)$, to each possible entity $e \in \mathscr{E}(a)$ of a
  - The relatedness score captures the extent to which Wikipedia pages share many in-links
- Overall relatedness of anchor a to entity X is computed as the sum of the votes of all the anchors detected in q (how many of the other anchor spans generated from the question, point to the entity):

$$relatedness(a \rightarrow X) = \sum_{b \in \mathscr{X}_q, except\ a} vote(b, X)$$

We use the entity with the highest relatedness score, and among those within ϵ of each other, we use the prior $p(e \mid a)$.

NEURAL GRAPH-BASED LINKING

ELQ LINKING ALGORITHM

The ELQ linking algorithm is an example of a biencoder approach

- Given a question q and a set of candidate Wikipedia entities and their associated text, it outputs entity id, mention start and mention end: (e, m_s, m_e)
- The question q and each candidate entity e are encoded with a separate encoder
- We perform entity mention detection similarly to span detection
- Then we use the dot product of the span encoding (average of all token embeddings in the span) and the entity encoding
- Then we pass it through a softmax to output the probability of each entity (among the set of entities) for the span under consideration
- ELQ requires a training set with entity boundaries marked (in the mentions/text) and linked to the corresponding entity ID

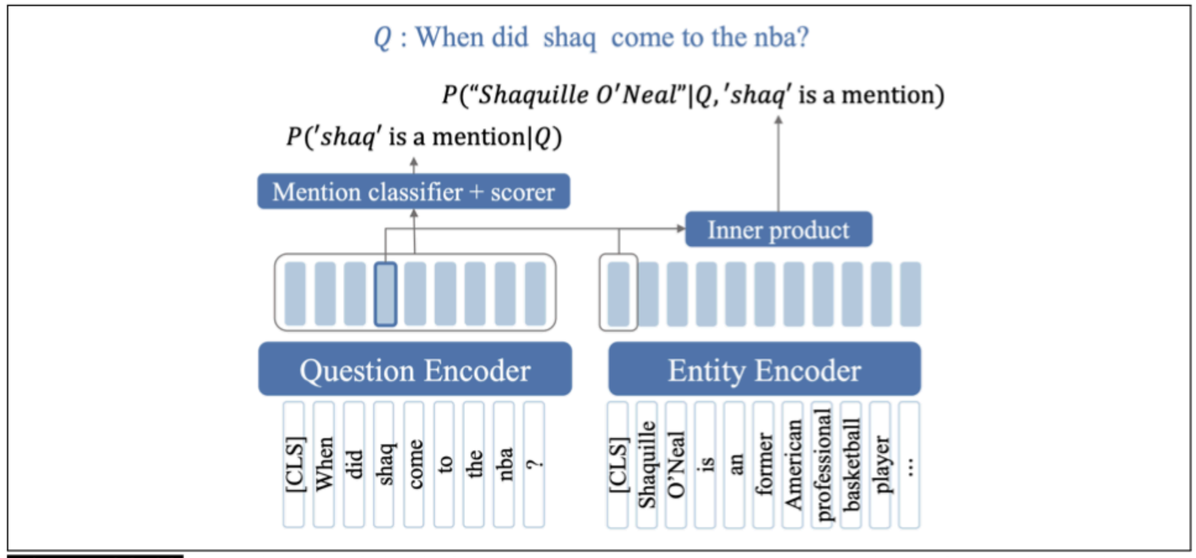


Figure 23.13 A sketch of the inference process in the ELQ algorithm for entity linking in questions (Li et al., 2020). Each candidate question mention span and candidate entity are separately encoded, and then scored by the entity/span dot product.

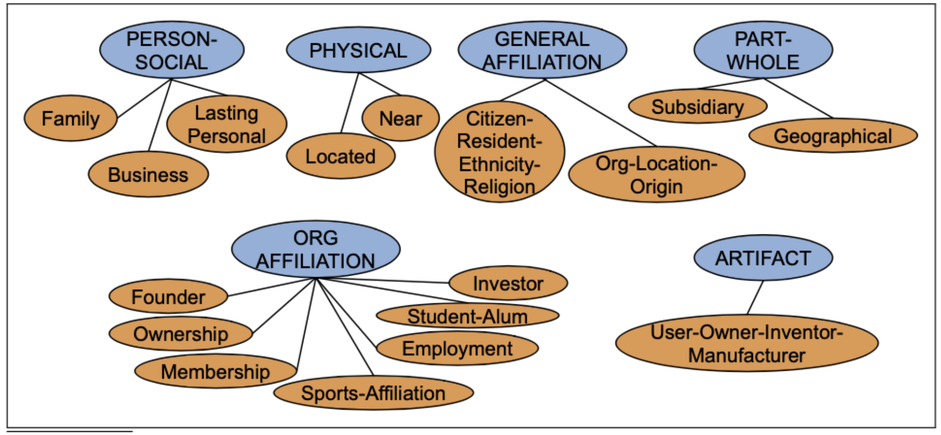
KNOWLEDGE-BASED QA

KNOWLEDGE-BASED QA OVERVIEW

- Knowledge-based QA maps questions to (database) queries
  - Can then be used to extract results from a structured database

Major paradigms of knowledge-based QA:

- Graph-based QA - the knowledge base is a graph
  - Entities are nodes
  - Edges are relations between nodes
- QA by semantic parsing
  - A “semantic parser” converts the query to a logical language that is then used to query a database



GRAPH-BASED QA OVERVIEW

A Resource Description Framework (RDF) triplet is a 3-tuple with a predicate (or relation) and 2 arguments, e.g.:

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

Basic graph-based QA involves

- Factoids in RDF format + answering questions about one of the missing arguments of the triplet
  - Some possible questions for above example: "When was Ada Lovelace born?", "Who was born in 1815?"
- Entity linking (mentions like Ada Lovelace are linked to the respective entity ID in the knowledge base)
- Mapping from the string in the question to canonical relations in the knowledge base, e.g., from "When was…born?" to relations like *birth-year*
  - We can perform a dot product between the encoding of the question text and the encoding of the possible relations (e.g., using a neutral model like BERT) to identify close matches
  - Then we need to rank the results (as there are usually multiple matches) - many approaches exist for this step

Task examples:

"When was Ada Lovelace born?" $\rightarrow birth-year(Ada\ Lovelace, ?x)$

"What is the capital of England?" $\rightarrow capital-city(?x, England)$

QA BY SEMANTIC PARSING

- A "semantic parser" converts the query to a logical language that is then used to query a database
  - Example logical language is predicate calculus (first order logic)
  - BERT-based approaches often used

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))))

Figure 23.14 Sample logical forms produced by a semantic parser for question answering, including two questions from the GeoQuery database of questions on U.S. Geography (Zelle and Mooney, 1996) with predicate calculus representations, one ATIS question with SQL (Iyer et al., 2017), a program over Freebase relations, and a program in QDMR, the Question Decomposition Meaning Representation (Wolfson et al., 2020).

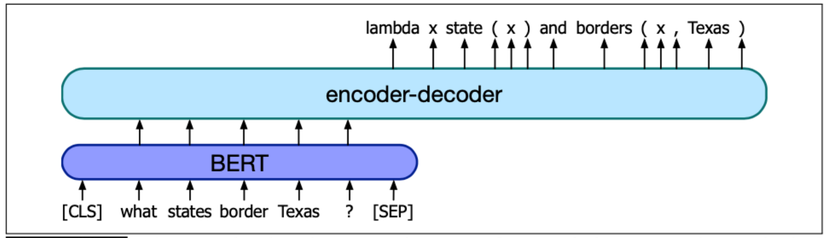


Figure 23.15 An encoder-decoder semantic parser for translating a question to logical form, with a BERT pre-encoder followed by an encoder-decoder (biLSTM or Transformer).

APPLICATION OF LANGUAGE MODELS TO THE QA TASK

- Using a pre-trained language model to answer questions by generating text for masked spans
- Have poor interpretability – do not provide context for their answers as they cannot tell which passage an answer came from

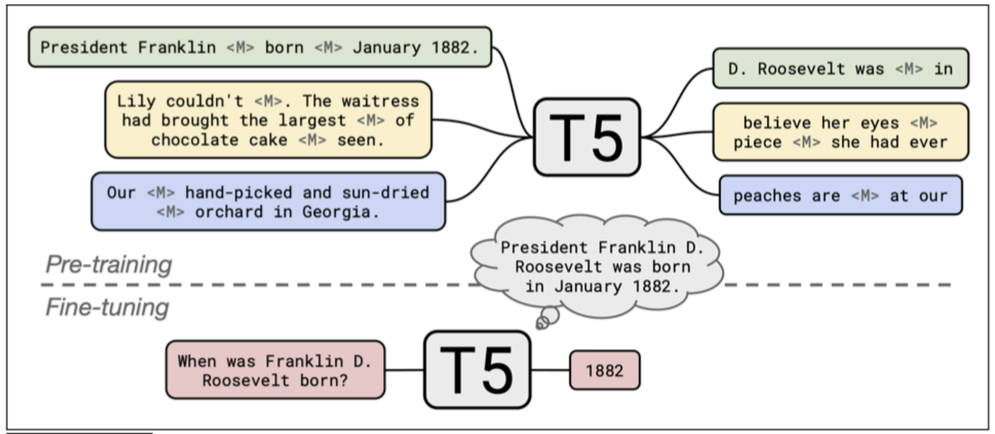


Figure 23.16 The T5 system is an encoder-decoder architecture. In pretraining, it learns to fill in masked spans of task (marked by <M>) by generating the missing spans (separated by <M>) in the decoder. It is then fine-tuned on QA datasets, given the question, without adding any additional context or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