


# **Question Answering, ASR, and TTS**

Natalie Parde

UIC CS 521



# What is question answering?

- The process of **automatically retrieving** compact quantities of correct, relevant **information** in response to a user's **query**

**We use  
question  
answering  
systems  
everyday.**

What is UIC's mascot?

× | 🔊 | C


[All](#) [Images](#) [Shopping](#) [Maps](#) [News](#) [More](#) [Settings](#) [Tools](#)


About 167,000 results (1.07 seconds)


University of Illinois at Chicago / Mascot

## Sparky D. Dragon

People also search for

 **The University of Chicago**  
Phil the Phoenix

 **Loyola University Chicago**  
LU Wolf

 **DePaul University**  
DIBS

[Feedback](#)

[dos.uic.edu](#) › [About](#) › [Student Handbook](#) ▼

### [UIC History, Traditions, Symbols | Office of the Dean of ...](#)

**UIC Symbols:** School Colors, **Mascot**, Song. Our athletic teams are known as the “Flames,” a name chosen by **UIC** students in honor of the Great Chicago Fire.

# People have been interested in question answering systems nearly as long as computers have existed.

How many games did the Yankees play in July?<sup>1</sup>

<sup>1</sup>Bert F. Green Jr., Alice K. Wolf, Carol Chomsku, and Kenneth Laughery. 1961. Baseball: An Automatic Question Answerer. Link: <https://web.stanford.edu/class/linguist289/p219-green.pdf>



20



What is the answer to the Ultimate Question Of Life, The Universe, and Everything?<sup>1</sup>

<sup>1</sup>The Hitchhiker's Guide to the Galaxy



42



Technology > TEDx

How did supercomputer Watson beat Jeopardy! champion Ken Jennings? Experts discuss.

Posted by: [Kate Torgovnick May](#) April 5, 2013 at 1:59 pm EDT



**Question  
answering  
systems have  
even won game  
shows!**

<https://blog.ted.com/how-did-supercomputer-watson-beat-jeopardy-champion-ken-jennings-experts-discuss/>

## Question Answering Systems

- Typically focus on **factoid questions**
  - **Factoid Questions:** Questions that can be answered with simple facts expressed in short texts

When was UIC founded?

How far is UIC from the University of Chicago?

What is the average CS class size?

# Question Answering Systems

- Two major paradigms:
  - **Information retrieval-based** question answering
  - **Knowledge-based** question answering
- Less common paradigms:
  - **Language model-based** question answering
  - **Classic rule- or feature-based** question answering



## **Information Retrieval-based Question Answering**

- Relies on text from the web or from large corpora
- Given a user question:
  1. Find relevant documents and passages of text
  2. Read the retrieved documents or passages
  3. Extract an answer to the question directly from spans of text

## Knowledge-based Question Answering

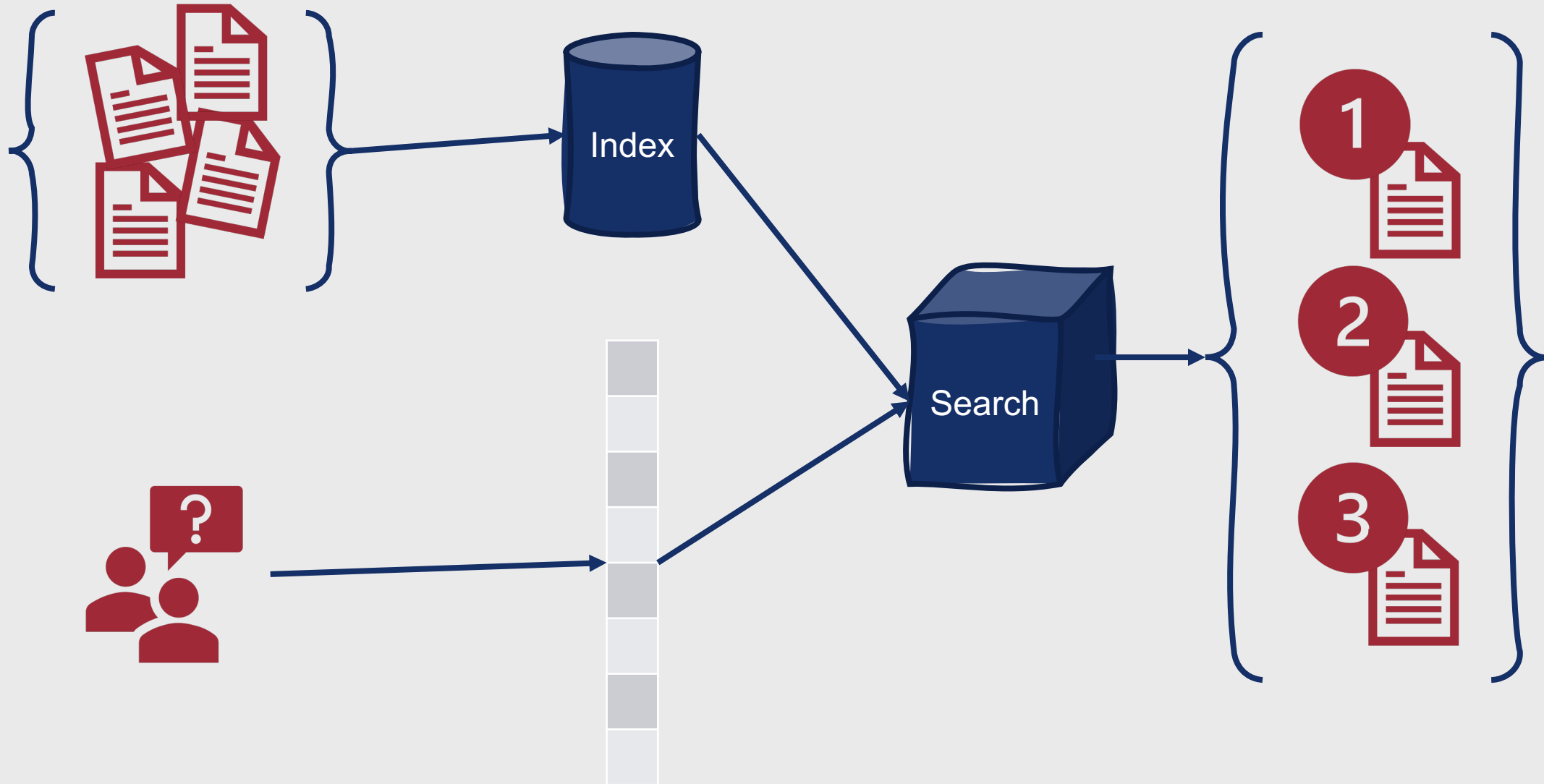
- Builds a semantic representation of the user's query
  - When was UIC founded? → `founded(UIC, x)`
- Uses these representations to query a database of facts



# What is information retrieval?

- **Information retrieval:** The process of selecting the most relevant media (e.g., text documents or images) from a large corpus

# How does information retrieval work?



# How are documents represented?

- Assign term weights to individual terms in documents
- Assign document scores based on which terms are present in a user's query
- Two common term weighting schemes:
  - TF-IDF
    - $tf_{t,d} = \log_{10}(\text{count}(t, d) + 1)$
    - $idf_t = \log_{10} \frac{N}{df_t}$
    - $tf-idf(t, d) = tf_{t,d} * idf_t$
  - BM25

# BM25

$$\text{BM25}(t, d) = \frac{\text{tf}_{t,d}}{k \left( 1 + \frac{b + b}{|d|} \left( \frac{|d|}{|d_{avg}|} \right) \right) + \text{tf}_{t,d}} * \text{idf}_t$$

Balance between term frequency and inverse document frequency

Importance of document length normalization

# Document Scoring

- Weight terms in documents to create document vectors
- Weight terms in queries to create query vectors
- Compute cosine similarity between a given document,  $d$ , and query,  $q$ 
  - $\text{score}(q, d) = \cos(\mathbf{q}, \mathbf{d}) = \frac{\mathbf{q} \cdot \mathbf{d}}{|\mathbf{q}| |\mathbf{d}|}$
- This can be slightly simplified since queries are usually short and query normalization won't have an impact on document ranking
  - $\text{score}(q, d) = \sum_{t \in q} \frac{\text{tf-idf}(t, d)}{|d|}$

# Document Scoring: Case Example

CS is the best topic!

CS 521 covers statistical NLP.

521 is the best class.

CS 521

$$\text{tf-idf}(t, d) = \log_{10}(\text{count}(t, d) + 1) * \log_{10} \frac{N}{\text{df}_t}$$

# Document Scoring: Case Example

CS is the best topic!

CS 521 covers statistical NLP.

521 is the best class.

CS 521

$$\text{tf-idf}(t, d) = \log_{10}(\text{count}(t, d) + 1) * \log_{10} \frac{N}{\text{df}_t}$$

Document 1

word	count	TF
CS	1	0.301
is	1	0.301
the	1	0.301
best	1	0.301
topic	1	0.301
521	0	0
covers	0	0
statistical	0	0
NLP	0	0
class	0	0

# Document Scoring: Case Example

CS is the best topic!

CS 521 covers statistical NLP.

521 is the best class.

CS 521

$$\text{tf-idf}(t, d) = \log_{10}(\text{count}(t, d) + 1) * \log_{10} \frac{N}{\text{df}_t}$$

Document 1

word	count	TF	# docs	IDF
CS	1	0.301	2	0.176
is	1	0.301	2	0.176
the	1	0.301	2	0.176
best	1	0.301	2	0.176
topic	1	0.301	1	0.477
521	0	0	2	0.176
covers	0	0	1	0.477
statistical	0	0	1	0.477
NLP	0	0	1	0.477
class	0	0	1	0.477

# Document Scoring: Case Example

CS is the best topic!

CS 521 covers statistical NLP.

521 is the best class.

CS 521

$$\text{tf-idf}(t, d) = \log_{10}(\text{count}(t, d) + 1) * \log_{10} \frac{N}{\text{df}_t}$$

Document 1

word	count	TF	# docs	IDF	TF-IDF
CS	1	0.301	2	0.176	0.053
is	1	0.301	2	0.176	0.053
the	1	0.301	2	0.176	0.053
best	1	0.301	2	0.176	0.053
topic	1	0.301	1	0.477	0.144
521	0	0	2	0.176	0
covers	0	0	1	0.477	0
statistical	0	0	1	0.477	0
NLP	0	0	1	0.477	0
class	0	0	1	0.477	0

# Document Scoring: Case Example

CS is the best topic!

CS 521 covers statistical NLP.

521 is the best class.

CS 521

$$\text{tf-idf}(t, d) = \log_{10}(\text{count}(t, d) + 1) * \log_{10} \frac{N}{\text{df}_t}$$

word		count	TF	# docs	IDF	TF-IDF
word		count	TF	# docs	IDF	TF-IDF
CS		1	0.301	2	0.176	0.053
is		0	0	2	0.176	0.053
the		0	0	2	0.176	0.053
best		0	0	2	0.176	0.053
topic		0	0	1	0.477	0
521		1	0.301	2	0.176	0.053
covers		1	0.301	1	0.477	0
statistical		1	0.301	1	0.477	0
NLP		1	0.301	1	0.477	0
class		0	0	1	0.477	0.144

# Document Scoring: Case Example

CS is the best topic!

CS 521 covers statistical NLP.

521 is the best class.

CS 521

$$\text{tf-idf}(t, d) = \log_{10}(\text{count}(t, d) + 1) * \log_{10} \frac{N}{\text{df}_t}$$

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

Doc. 1	Doc. 2	Doc. 3
0.053	0.053	0
0.053	0	0.053
0.053	0	0.053
0.053	0	0.053
0.144	0	0
0	0.053	0.053
0	0.144	0
0	0.144	0
0	0.144	0
0	0	0.144

Doc.	d	TF-IDF("CS")	TF-IDF("521")	Score

# Document Scoring: Case Example

CS is the best topic!

CS 521 covers statistical NLP.

521 is the best class.

CS 521

$$\text{tf-idf}(t, d) = \log_{10}(\text{count}(t, d) + 1) * \log_{10} \frac{N}{\text{df}_t}$$

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

Doc. 1	Doc. 2	Doc. 3
0.053	0.053	0
0.053	0	0.053
0.053	0	0.053
0.053	0	0.053
0.144	0	0
0	0.053	0.053
0	0.144	0
0	0.144	0
0	0.144	0
0	0	0.144

Doc	d	TF-IDF("CS")	TF-IDF("521")	Score
1	0.179	0.053	0	0.296

$$\sqrt{0.053^2 + 0.053^2 + 0.053^2 + 0.053^2 + 0.144^2 + 0^2 + 0^2 + 0^2 + 0^2 + 0^2} = 0.179$$

# Document Scoring: Case Example

CS is the best topic!

CS 521 covers statistical NLP.

521 is the best class.

CS 521

$$\text{tf-idf}(t, d) = \log_{10}(\text{count}(t, d) + 1) * \log_{10} \frac{N}{\text{df}_t}$$

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

Doc. 1	Doc. 2	Doc. 3
0.053	0.053	0
0.053	0	0.053
0.053	0	0.053
0.053	0	0.053
0.144	0	0
0	0.053	0.053
0	0.144	0
0	0.144	0
0	0.144	0
0	0	0.144

Doc	d	TF-IDF("CS")	TF-IDF("521")	Score
1	0.179	0.053	0	0.296
2	0.260	0.053	0.053	0.408

$$\sqrt{0.053^2 + 0^2 + 0^2 + 0^2 + 0^2 + 0.053^2 + 0.144^2 + 0.144^2 + 0.144^2 + 0^2} = 0.260$$

# Document Scoring: Case Example

CS is the best topic!

CS 521 covers statistical NLP.

521 is the best class.

CS 521

$$\text{tf-idf}(t, d) = \log_{10}(\text{count}(t, d) + 1) * \log_{10} \frac{N}{\text{df}_t}$$

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

Doc. 1	Doc. 2	Doc. 3
0.053	0.053	0
0.053	0	0.053
0.053	0	0.053
0.053	0	0.053
0.144	0	0
0	0.053	0.053
0	0.144	0
0	0.144	0
0	0.144	0
0	0	0.144

Doc	d	TF-IDF("CS")	TF-IDF("521")	Score
1	0.179	0.053	0	0.296
2	0.260	0.053	0.053	0.408
3	0.179	0	0.053	0.296

$$\sqrt{0^2 + 0.053^2 + 0.053^2 + 0.053^2 + 0^2 + 0.053^2 + 0^2 + 0^2 + 0^2 + 0.144^2} = 0.179$$

# Document Scoring: Case Example

CS is the best topic!

CS 521 covers statistical NLP.

521 is the best class.

CS 521

$$\text{tf-idf}(t, d) = \log_{10}(\text{count}(t, d) + 1) * \log_{10} \frac{N}{\text{df}_t}$$

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

Doc. 1	Doc. 2	Doc. 3
0.053	0.053	0
0.053	0	0.053
0.053	0	0.053
0.053	0	0.053
0.144	0	0
0	0.053	0.053
0	0.144	0
0	0.144	0
0	0.144	0
0	0	0.144

Doc	d	TF-IDF("CS")	TF-IDF("521")	Score
1	0.179	0.053	0	0.296
2	0.260	0.053	0.053	0.408
3	0.179	0	0.053	0.296

# Stopword Removal?



Less common in modern IR than it used to be



Advantage

Document vectors have lower dimensionality



Disadvantage

More difficult to find good matches for queries that contain stopwords for important reasons

# How do we store document vectors?

Inverted Index

CS (DF=2)  $\rightarrow$  {Doc. 1 (TF=1), Doc. 2 (TF=1)}

is (DF=2)  $\rightarrow$  {Doc. 1 (TF=1), Doc. 3 (TF=1)}

...

class (DF=1)  $\rightarrow$  {Doc. 3 (TF=1)}

# Evaluating IR Systems

- Regular NLP metrics
  - Precision
  - Recall
- More sophisticated IR-specific metrics
  - Precision-recall curve
  - Mean average precision

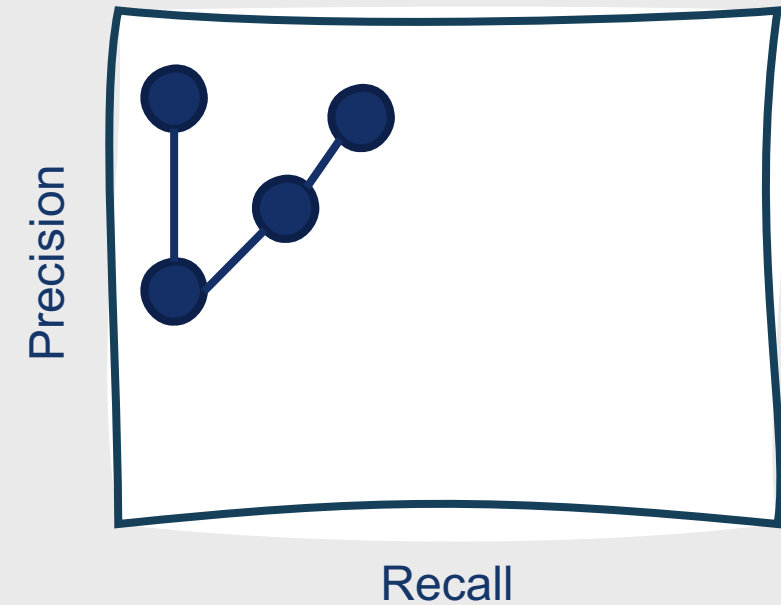
# Precision-Recall Curve

Rank	Precision at Rank	Recall at Rank
1	1.0	0.1
2	0.5	0.1
3	0.67	0.2
4	0.75	0.3

$$\text{IntPrecision}(r) = \max_{i \geq r} \text{Precision}(i)$$

Interpolated Precision	Recall
1.0	0.0
1.0	0.1
0.75	0.2
0.75	0.3

Avg. Interpolated Precision	Recall
1.0	0.0
0.75	0.1
0.5	0.2
0.4	0.3





# Mean Average Precision

- Single score across multiple queries
- Given a set of queries  $Q$  and a set,  $R_r$ , of relevant documents  $d$  at or above rank  $r$ :
  - $AP = \frac{1}{|R_r|} \sum_{d \in R_r} \text{Precision}_r(d)$
  - $MAP = \frac{1}{|Q|} \sum_{q \in Q} AP(q)$

# IR with Dense Vectors

- Recent work has explored dense vectors as an alternative to TF-IDF or BM25 vectors
  - Advantage:
    - More capable of handling synonymy
  - Disadvantage:
    - Fixed encoder input length
    - Less efficient
- Typically done by:
  - Training a **biencoder** to separately encode document and queries
    - $h_q = \text{Encoder}_Q(q)$
    - $h_d = \text{Encoder}_D(d)$
  - Computing the dot product between a given document and query to find the document score
    - $\text{score}(q, d) = h_q \cdot h_d$

# IR-based Factoid Question Answering

## Goal

Goal: Find relevant answers to questions by searching through documents in a corpus

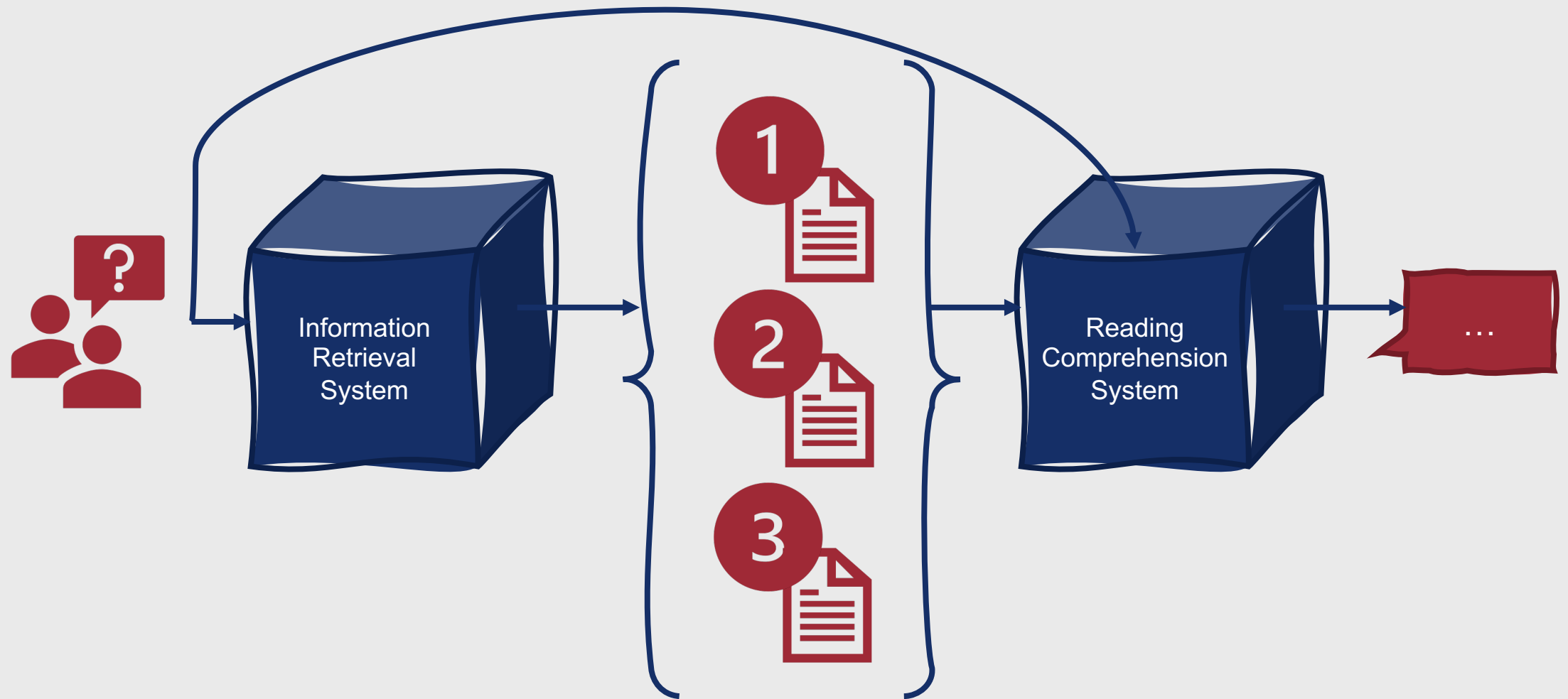


## Retrieve and Read

Dominant Paradigm: **Retrieve and read** model

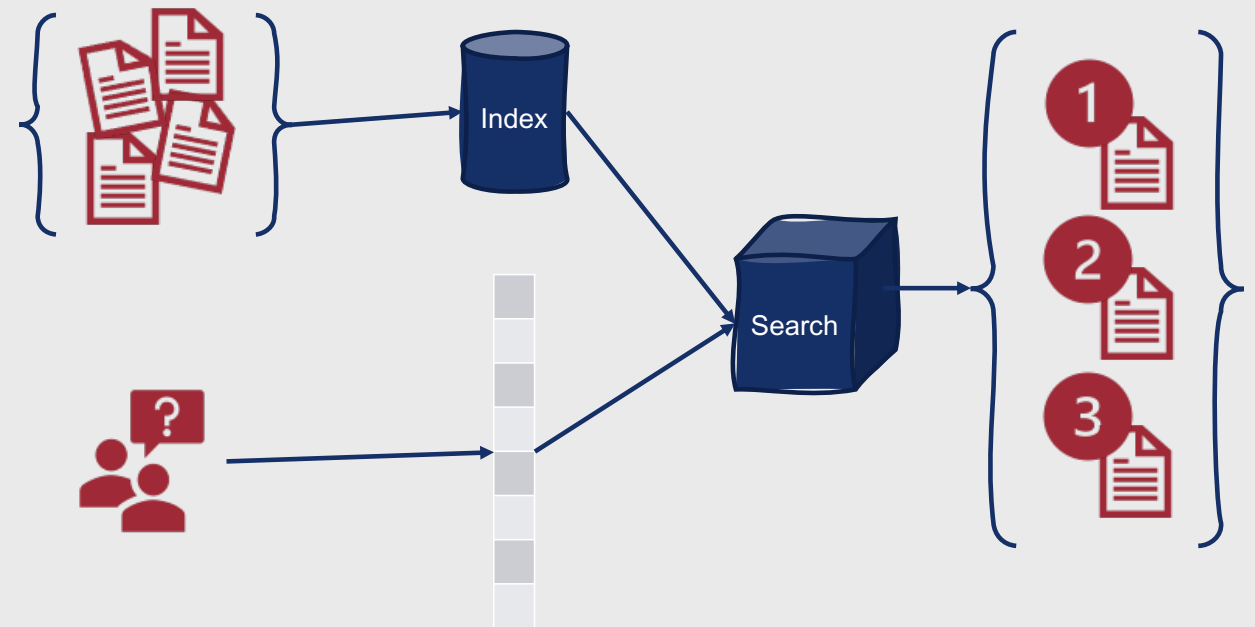
- **Retrieve** relevant documents for the given query
- **Read** those documents to find text segments that answer the query

# Retrieve and Read Model



# Step #1: Retrieve

- Performed using a standard information retrieval architecture



# Step #2: Read

- Performed using a **reading comprehension** model
- **Reading comprehension:** Given a document and a query, select (if available) the span of text from the document that answers the query



# Reading Comprehension

- A task designed to measure natural language understanding performance
- Basic premise: Take children's reading comprehension tests, and use them to evaluate text comprehension algorithms

## Prime\_number

### The Stanford Question Answering Dataset

A prime number (or a prime) is a natural number greater than 1 that has no positive divisors other than 1 and itself. A natural number greater than 1 that is not a prime number is called a composite number. For example, 5 is prime because 1 and 5 are its only positive integer factors, whereas 6 is composite because it has the divisors 2 and 3 in addition to 1 and 6. The fundamental theorem of arithmetic establishes the central role of primes in number theory: any integer greater than 1 can be expressed as a product of primes that is unique up to ordering. The uniqueness in this theorem requires excluding 1 as a prime because one can include arbitrarily many instances of 1 in any factorization, e.g.,  $3$ ,  $1 \cdot 3$ ,  $1 \cdot 1 \cdot 3$ , etc. are all valid factorizations of 3.

**What is the only divisor besides 1 that a prime number can have?**

Ground Truth Answers: itself itself itself itself itself

**What are numbers greater than 1 that can be divided by 3 or more numbers called?**

Ground Truth Answers: composite number composite number composite number primes

**What theorem defines the main role of primes in number theory?**

Ground Truth Answers: The fundamental theorem of arithmetic fundamental theorem of arithmetic arithmetic fundamental theorem of arithmetic arithmetic fundamental theorem of arithmetic

**Any number larger than 1 can be represented as a product of what?**

Ground Truth Answers: a product of primes product of primes that is unique up to ordering primes primes primes that is unique up to ordering

**Why must one be excluded in order to preserve the uniqueness of the**

- Stanford Question Answering Dataset (SQuAD)
  - English
  - Passages from Wikipedia
  - Associated questions
    - Many have answers that are spans from the passage
    - Some are designed to be unanswerable
  - <https://rajpurkar.github.io/SQuAD-explorer/>
- HotpotQA
  - English
  - Question-answer pairs based on multiple context documents
  - <https://hotpotqa.github.io/>
- Natural Questions
  - English
  - Based on real, anonymized queries to Google Search
  - <https://ai.google.com/research/NaturalQuestions>
- TyDi QA
  - Question-answer pairs from typologically diverse languages
  - <https://ai.google.com/research/tydiqa>

# Reading Comprehension Datasets

# Answer Span Extraction

- Goal: Compute, for each token, the probability that it is:
  - The start of the answer span
  - The end of the answer span

How many floors are in the Science and Engineering Offices building?

Although there are 13 floors in SEO, the elevator only goes to the 12<sup>th</sup> floor since the architect didn't like how elevator boxes look on the top of buildings.

$P_{\text{start}}("13")$

$P_{\text{end}}("13")$

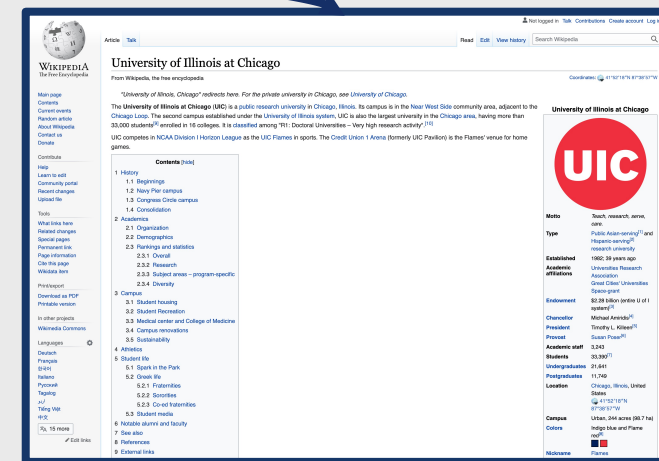
# Answer Span Extraction

- Common approach
  - Concatenate the query and passage, separated by a [SEP] token
  - Pass the concatenated sequence to an encoder
  - Add a linear layer that learns span-start ( $S$ ) and span-end ( $E$ ) embeddings
  - Compute span-start and span-end probabilities for each token  $p_i$  in a passage  $P$ 
    - $P_{\text{start}_i} = \frac{e^{S \cdot p_i}}{\sum_{j=0}^{|P|} e^{S \cdot p_j}}$
    - $P_{\text{end}_i} = \frac{e^{E \cdot p_i}}{\sum_{j=0}^{|P|} e^{E \cdot p_j}}$
  - Compute a score for each passage
    - $\text{score}(i, j) = S \cdot p_i + E \cdot p_j$
  - Select the highest-scoring passage for which  $j \geq i$
- This is **extractive** QA approach

# Entity Linking

- **Entity linking:** Associating mentions in text with the concepts to which they correspond in a structured knowledge base
- Typically done using a two-stage process:
  - **Mention detection:** Detecting that a concept has been mentioned
  - **Mention disambiguation:** Determining which concept has been mentioned

The coolest department at UIC is the Department of Computer Science.



# Classic Baseline: TAGME

- TAGME:  
<https://dl.acm.org/doi/10.1145/1871437.1871689>
- Setup:
  - For each Wikipedia page  $e$  in an index of all Wikipedia entries, compute the number of Wikipedia pages that point to  $e$ ,  $\text{in}(e)$
  - Build an anchor dictionary containing, for each page  $e$ , all hyperlinked spans of text that point to  $e$
  - For each unique span of anchor text  $a$  pointing to  $e$ , count:
    - Its overall frequency in Wikipedia,  $\text{freq}(a)$
    - The number of times it occurs as a link,  $\text{link}(a)$
    - Its link probability,  $\text{linkprob}(a) = \frac{\text{link}(a)}{\text{freq}(a)}$

# Classic Baseline: TAGME

- **Mention Detection**
  - Given an input question, find all token sequences of less than or equal to six words
  - For each token sequence, check to see if it exists in the anchor dictionary
  - Prune mentions with low link probabilities
- **Mention Disambiguation**
  - When an anchor mention matches multiple entities, compute a score based on the prior probability for each anchor-entity pair and the relatedness between each entity and the other entities in the question, using values calculated during the setup process

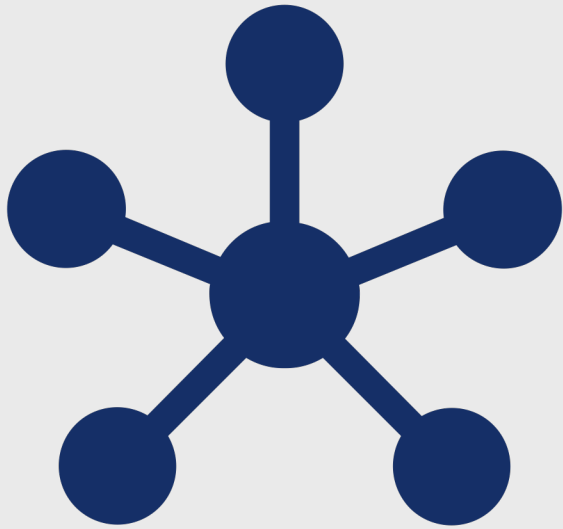
# Neural Graph-based Entity Linking

- Modern approaches often make use of **biencoders**
  - One encoder is trained to encode a candidate mention
  - One encoder is trained to encode an entity (e.g., a Wikipedia page)
  - The dot product between the two encoded representations is computed
- Require annotated data indicating mention boundaries and corresponding entity links
  - **WebQuestionsSP:**  
<https://www.microsoft.com/en-us/download/details.aspx?id=52763>
  - **GraphQuestions:**  
<https://github.com/ysu1989/GraphQuestions>

# Knowledge-based Question Answering

- Answers questions by mapping them to formal queries over structured knowledge sources





# Knowledge-based Question Answering

---

- Two common paradigms:
  - **Graph-based** question answering
  - Question answering by **semantic parsing**
- Both require entity linking

# Graph-based Question Answering

- Facts are stored as (subject, predicate, object) triples
  - Sometimes referred to as RDF triples
- Entity mentions are linked to entities in a knowledge graph
- Queries are mapped to canonical relations
  - “Where is UIC’s computer science department located?” → LOCATIONOF(“UIC CS”, ?x)
  - Can be done using similar methods to neural entity linking
- Triples matching the canonical relations are identified and ranked
  - Can be done based on entity graph structure



# Question Answering by Semantic Parsing

---

- Maps questions directly to logical form using a semantic parser
  - First-order logic
  - SQL
- Logical form is used to query a knowledge base directly

# Language Model-based Question Answering

- Alternative strategy for question answering: Derive answers entirely from the information stored in a high-quality **language model**
  - In pretraining, train an encoder-decoder architecture to fill in masked spans of text
  - In finetuning, train the decoder to output an answer for a given question
- **Advantages:**
  - Simple approach
  - Decent performance
- **Disadvantages:**
  - Lower performance than models trained specifically for question answering
  - Poor interpretability

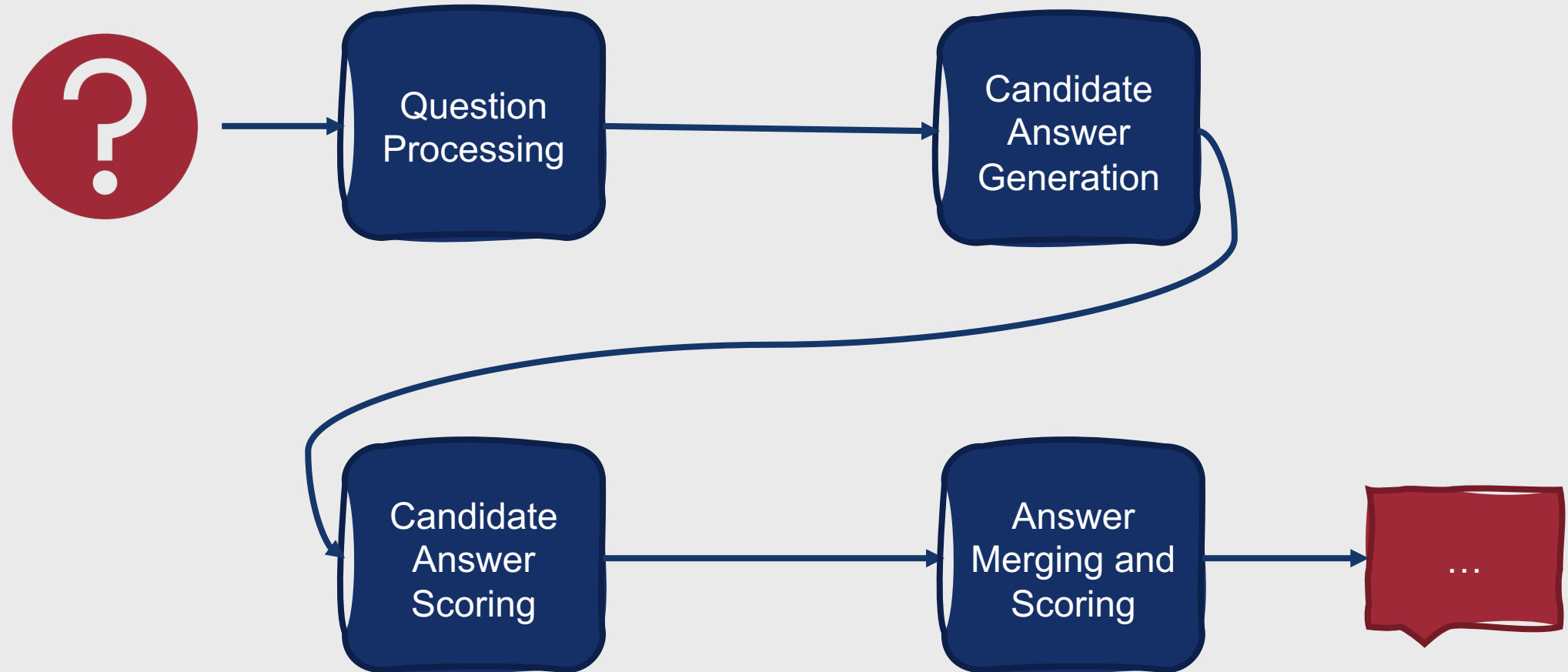
# Hybrid Rule- and Feature- based Question Answering

- Remember ...there's no need to limit a system to using *only* on strategy!
- Until recently, a popular question answering paradigm involved leveraging a combination of rule-based methods and feature-based classification techniques

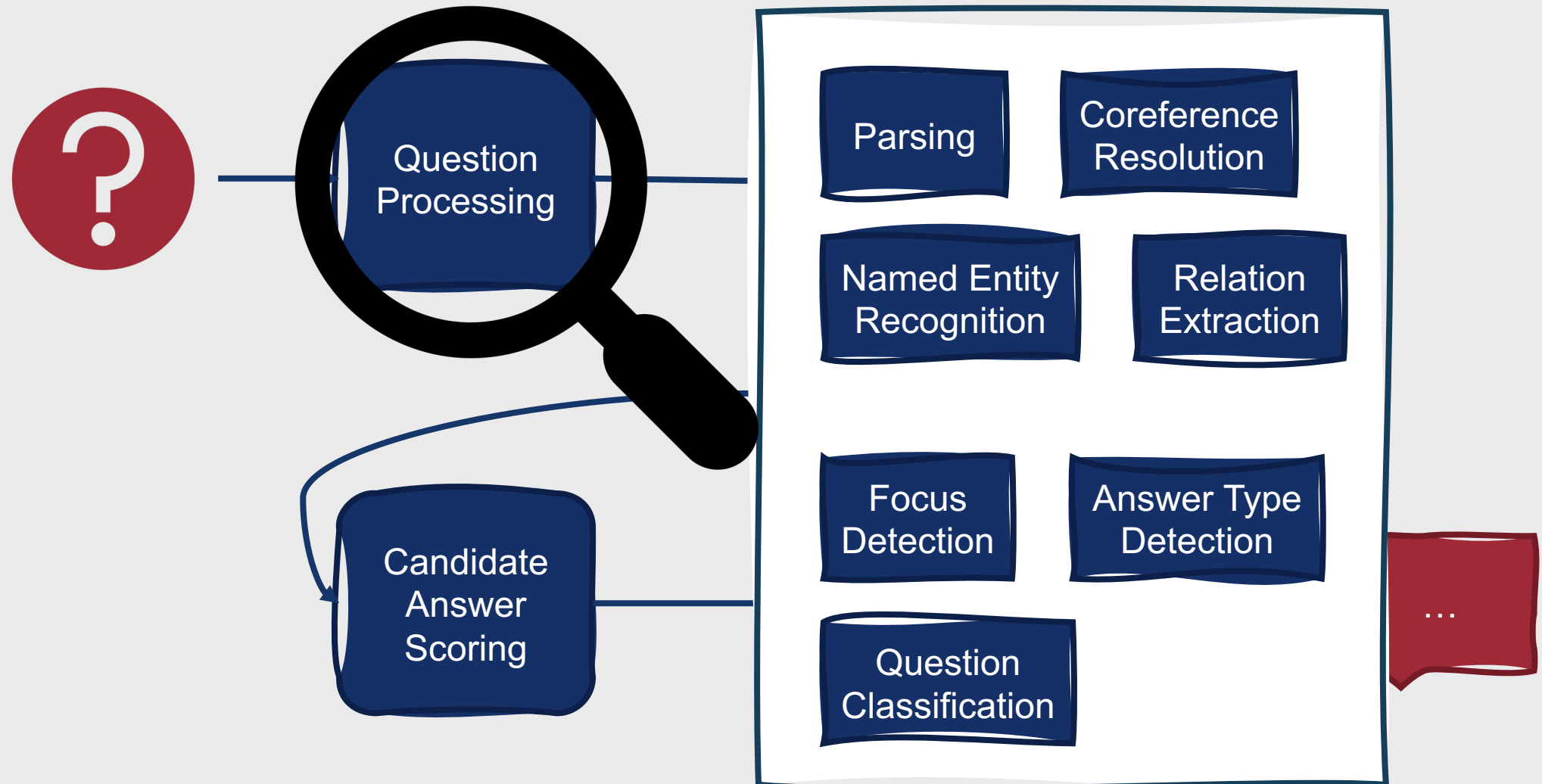
# Case Example: DeepQA

- Question answering component of Watson
- Four stages:
  1. Question processing
  2. Candidate answer generation
  3. Candidate answer scoring
  4. Answer merging and scoring

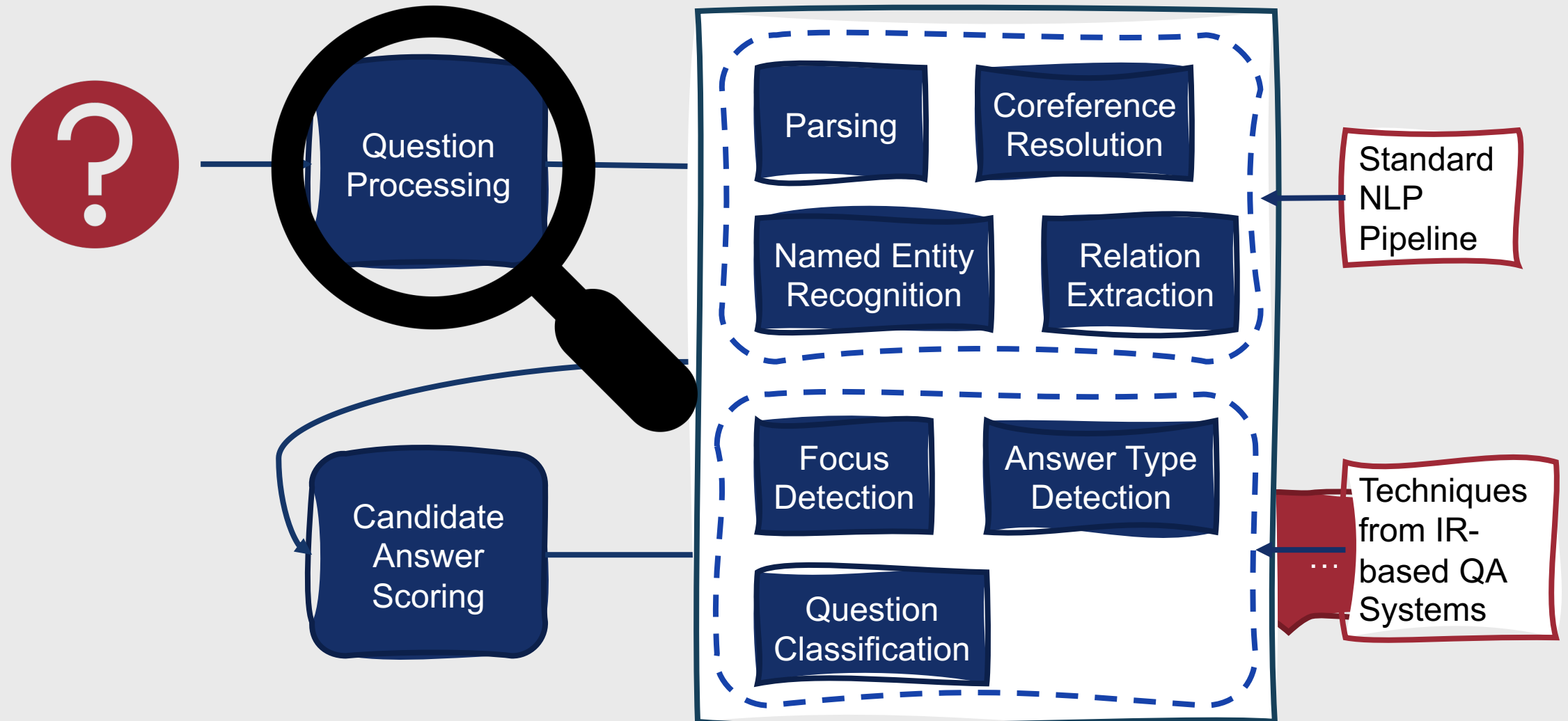
# Case Example: DeepQA



# Stage 1: Question Preprocessing



# Stage 1: Question Preprocessing



# Stage 1: Question Preprocessing

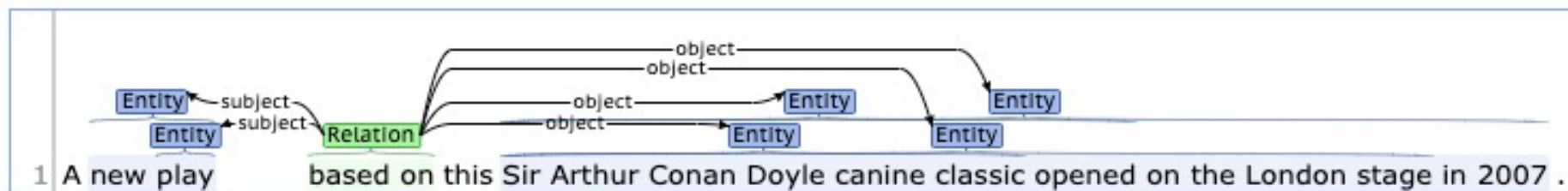
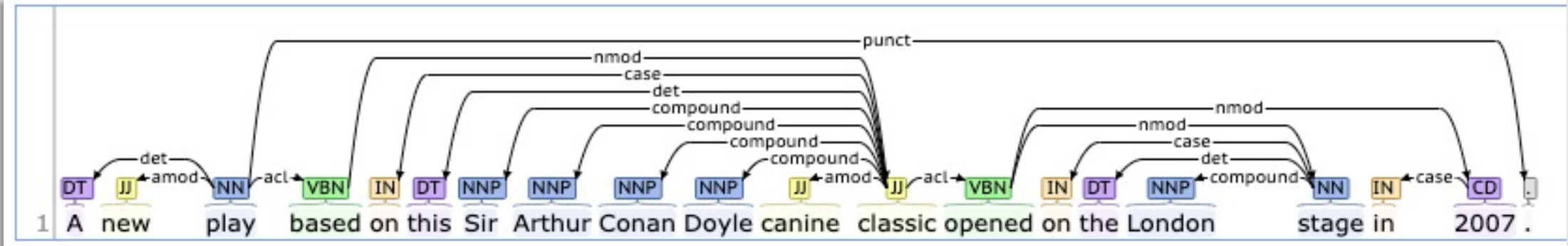
## Jeopardy! Example:

A new play based on this Sir Arthur Conan Doyle canine classic opened on the London stage in 2007.

# Stage 1: Question Preprocessing

## Jeopardy! Example:

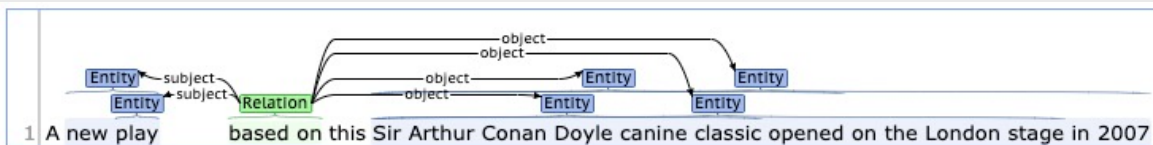
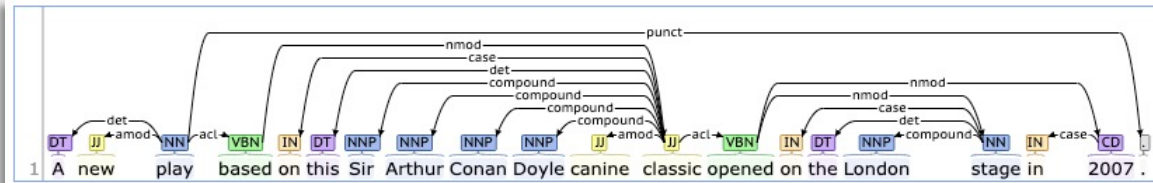
A new play based on this Sir Arthur Conan Doyle canine classic opened on the London stage in 2007.



# Stage 1: Question Preprocessing

## Jeopardy! Example:

A new play based on **this Sir Arthur Conan Doyle canine classic** opened on the London stage in 2007.



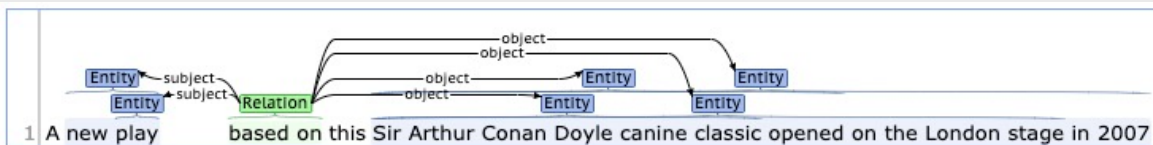
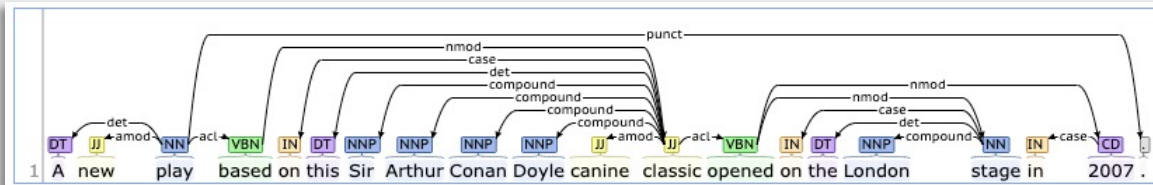
**Focus Detection:** Which part of the question co-refers with the answer?

Extracted using handwritten rules in DeepQA

# Stage 1: Question Preprocessing

## Jeopardy! Example:

A new play based on **this Sir Arthur Conan Doyle canine classic** opened on the London stage in 2007.



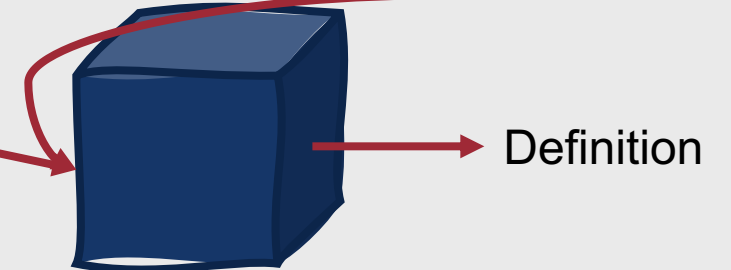
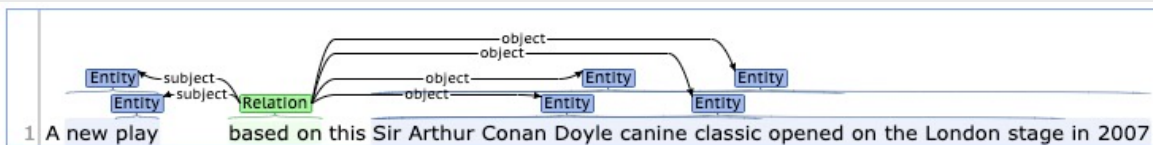
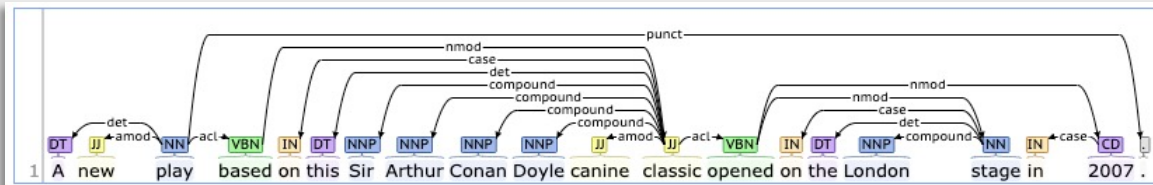
**Answer Type Detection:** Which word tells us about the semantic type of answer to expect?

DeepQA extracts roughly 5000 possible answer types (some questions may take multiple answer types), using a rule-based approach

# Stage 1: Question Preprocessing

## Jeopardy! Example:

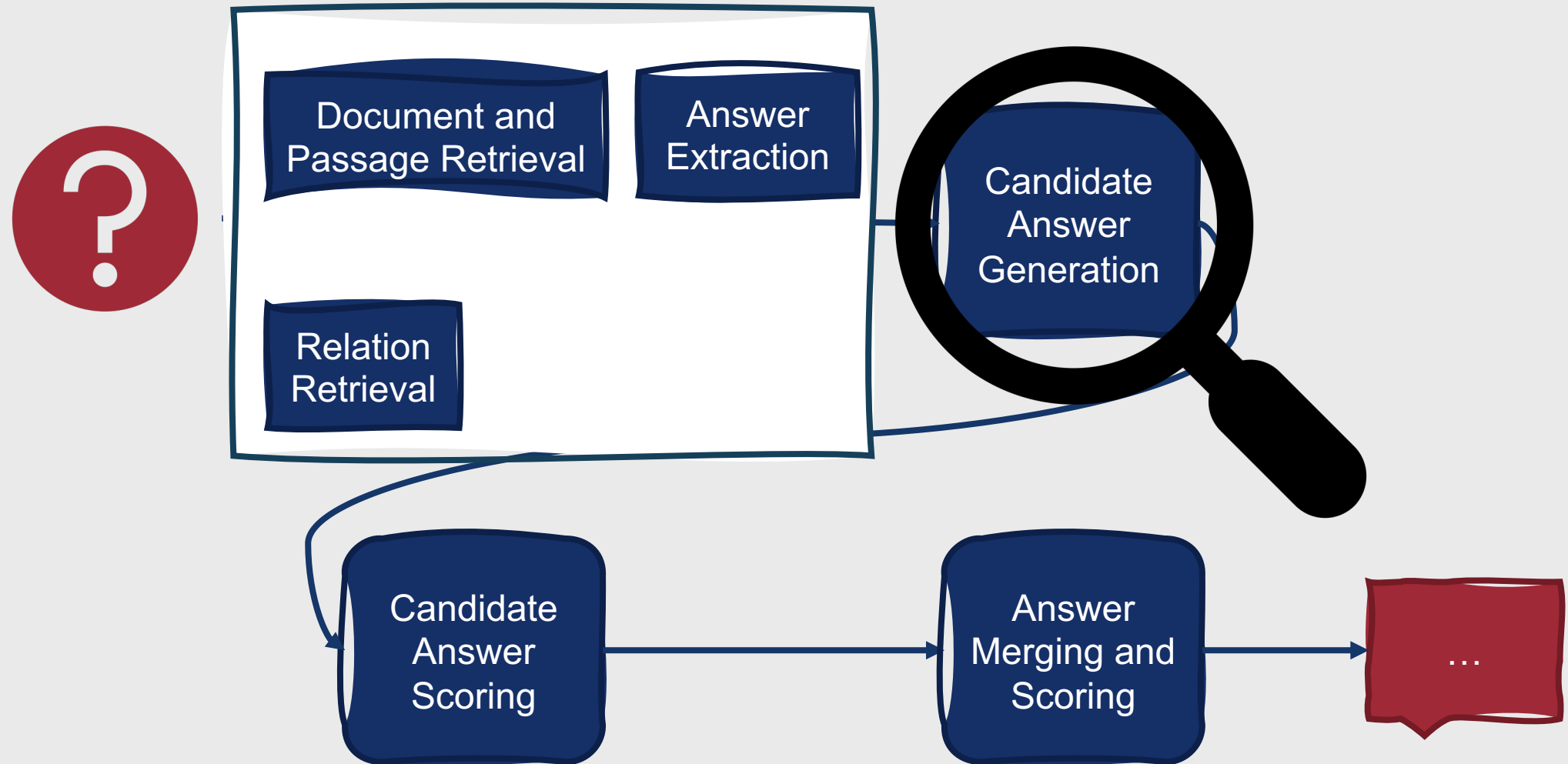
A new play based on **this Sir Arthur Conan Doyle canine classic** opened on the London stage in 2007.



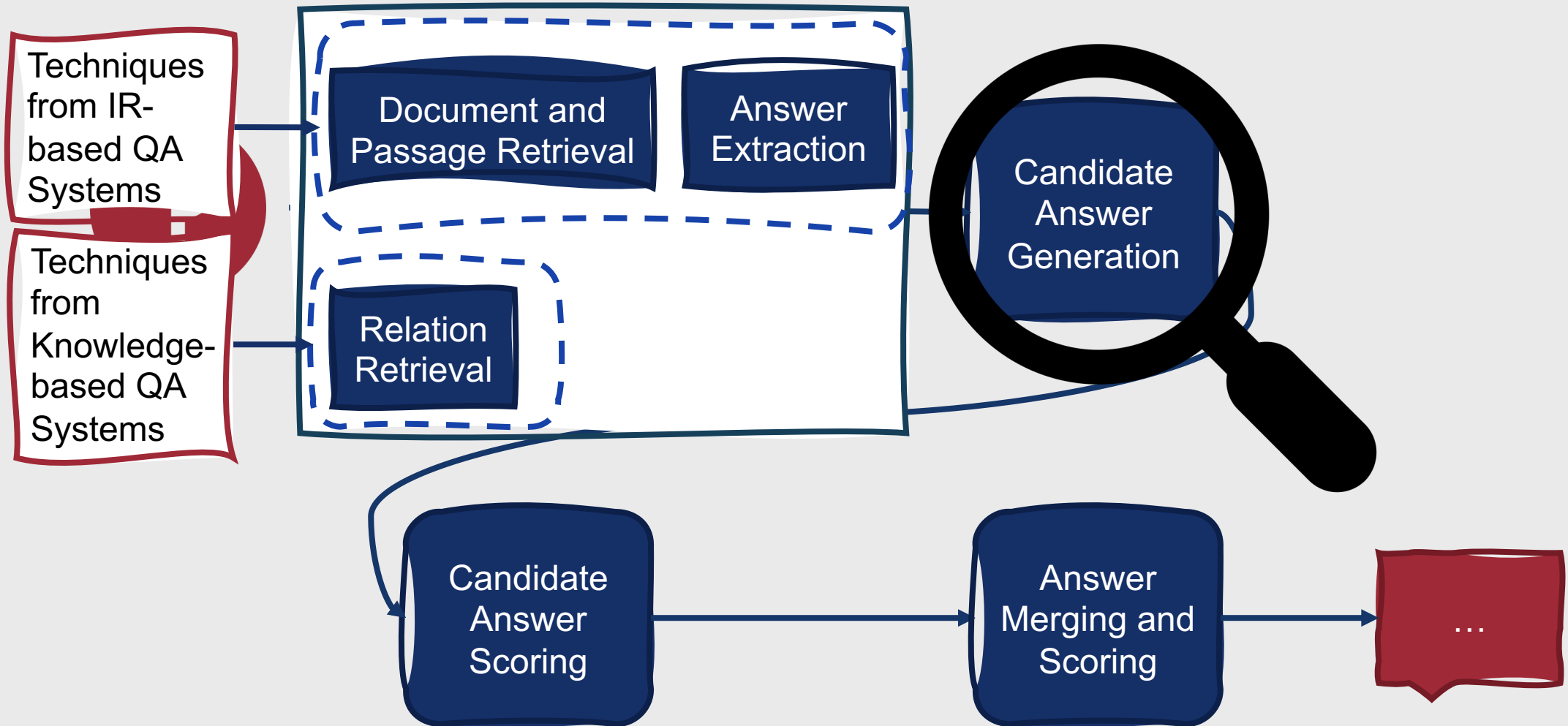
**Question Classification:** What type of question is this (multiple choice, fill-in-the-blank, definition, etc.)?

Generally done using pattern-matching regular expressions over words or parse trees

# Stage 2: Candidate Answer Generation



# Stage 2: Candidate Answer Generation



# Stage 2: Candidate Answer Generation

## Jeopardy! Example:

A new play based on **this Sir Arthur Conan Doyle canine classic** opened on the London stage in 2007.

Document and  
Passage Retrieval

In 2007, Peepolykus Theatre Company premiered a new adaptation of *The Hound of the Baskervilles* at West Yorkshire Playhouse in Leeds.

The play is an adaptation of the Arthur Conan Doyle's novel: *The Hound of the Baskervilles* (1901).

# Stage 2: Candidate Answer Generation

## Jeopardy! Example:

A new play based on **this Sir Arthur Conan Doyle canine classic** opened on the London stage in 2007.

Document and  
Passage Retrieval

In 2007, Peepolykus Theatre Company premiered a new adaptation of *The Hound of the Baskervilles* at West Yorkshire Playhouse in Leeds.

The play is an adaptation of the Arthur Conan Doyle's novel: *The Hound of the Baskervilles* (1901).

Answer  
Extraction

The Hound of the Baskervilles

The Hound of the Baskervilles (1901)

# Stage 2: Candidate Answer Generation

**Jeopardy! Example:**

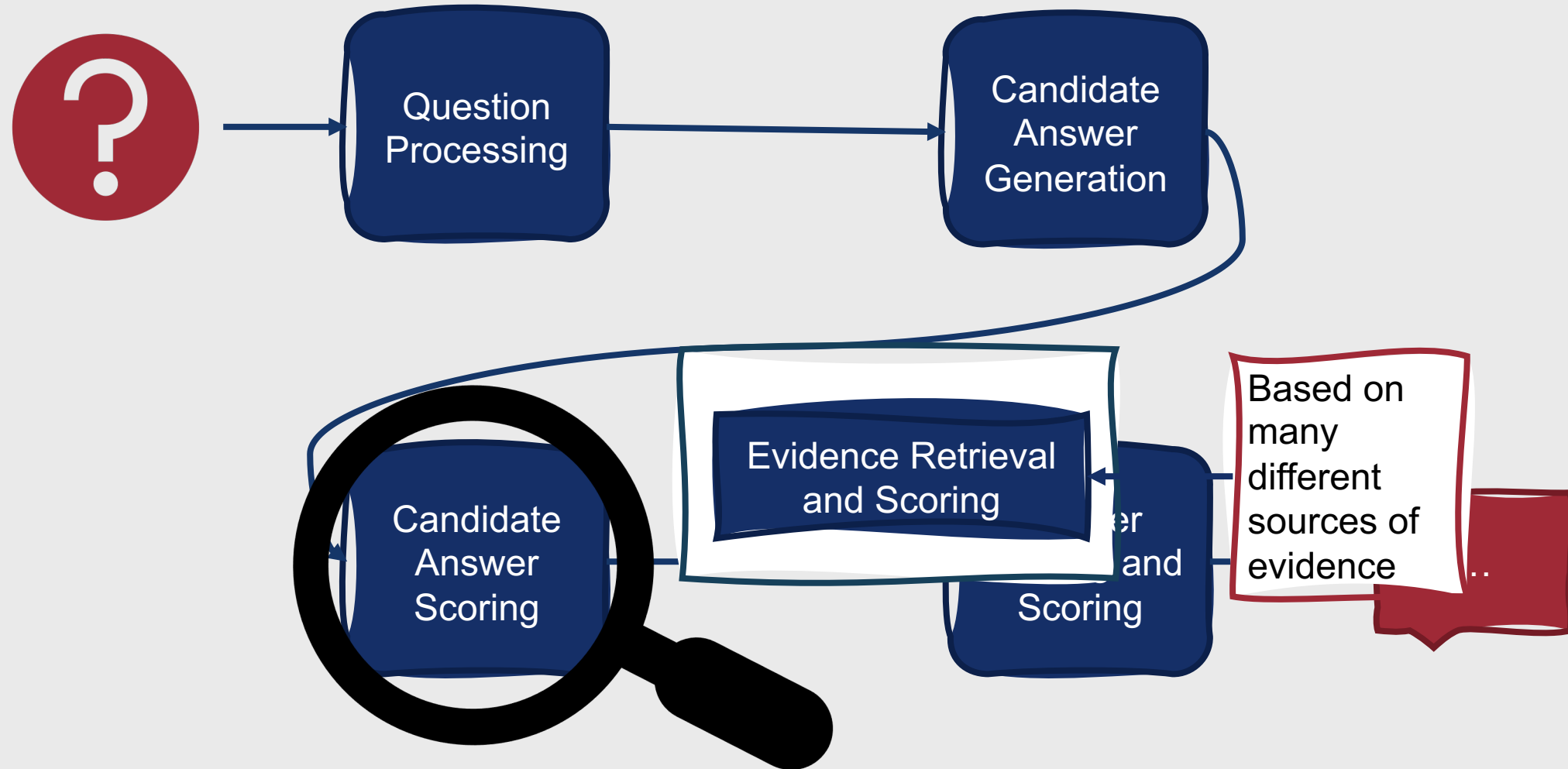
`basedOn(x, "Sir Arthur Conan Doyle canine classic")`

Relation Retrieval



The Hound of the Baskervilles

# Stage 3: Candidate Answer Scoring



# Stage 3: Candidate Answer Scoring

The Hound of the Baskervilles

The Hound of the Baskervilles

The Hound of the Baskervilles (1901)

Information extracted from structured knowledge bases

Retrieved passages with terms matching the question

# Stage 3: Candidate Answer Scoring

The Hound of the Baskervilles

The Hound of the Baskervilles

The Hound of the Baskervilles (1901)

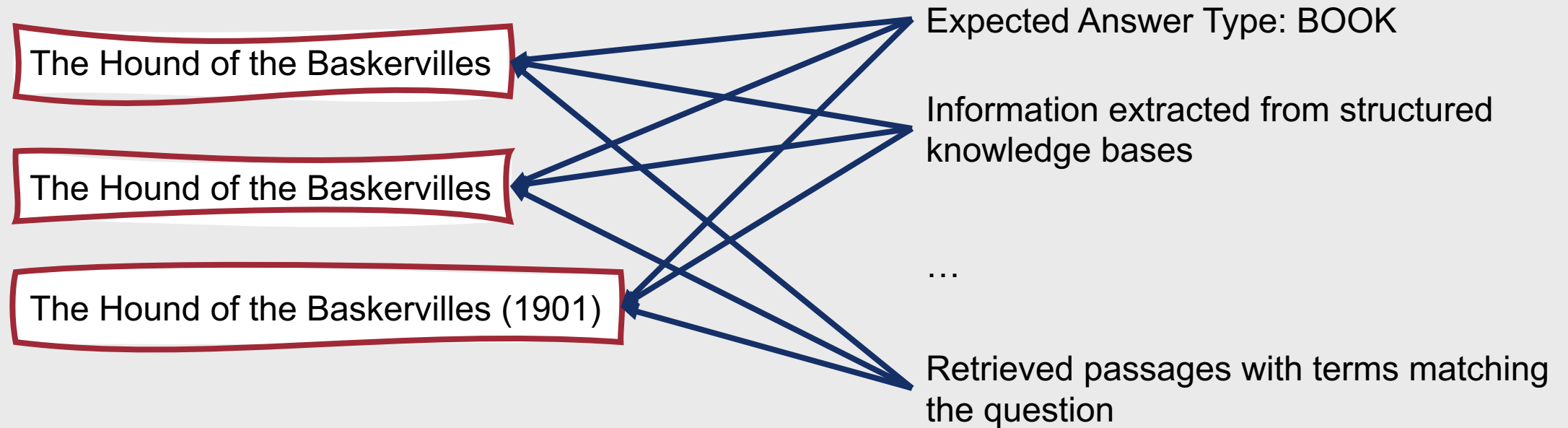
Expected Answer Type: BOOK

Information extracted from structured knowledge bases

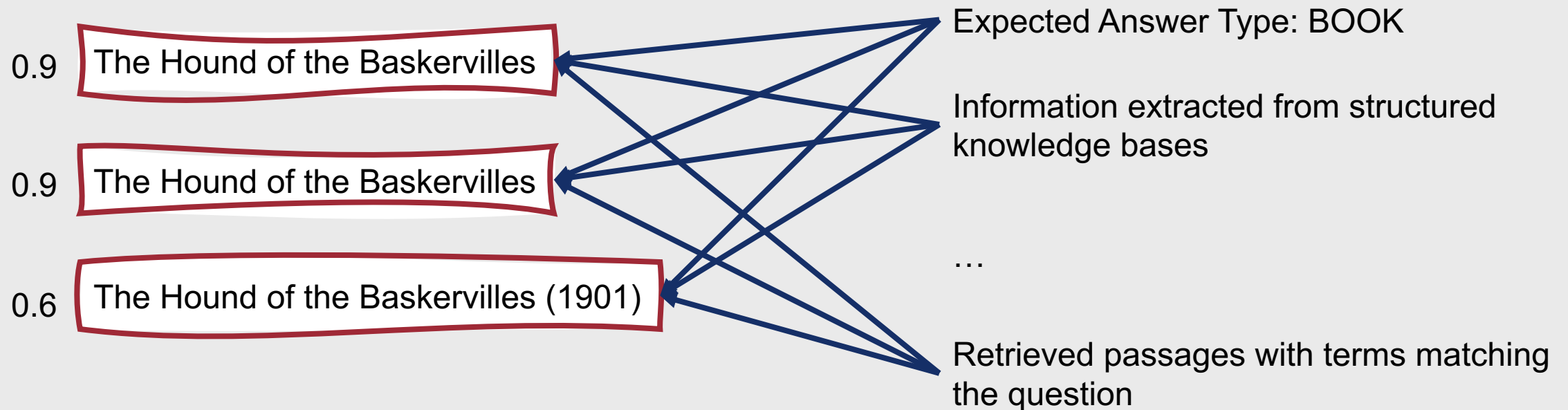
...

Retrieved passages with terms matching the question

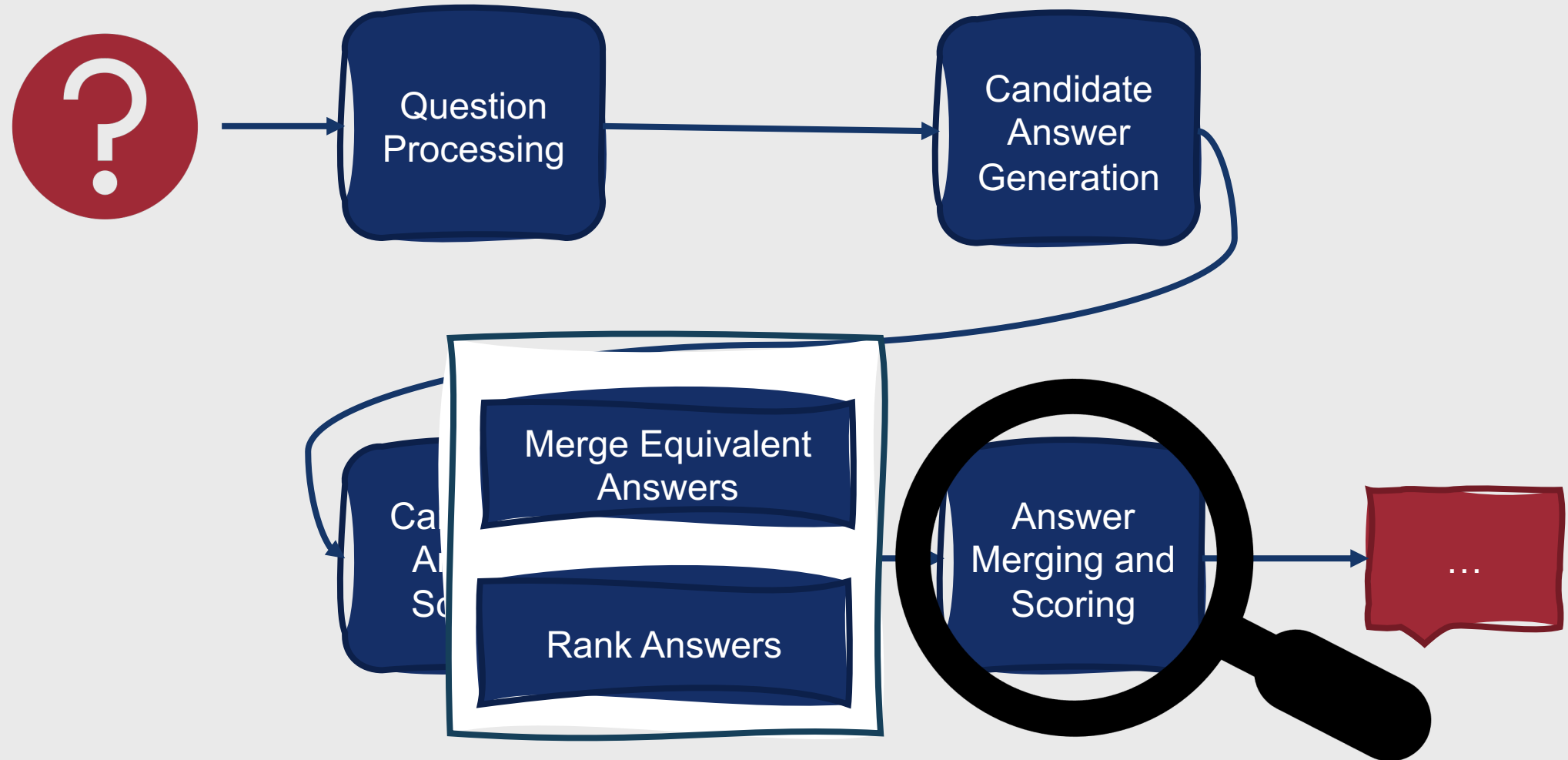
# Stage 3: Candidate Answer Scoring



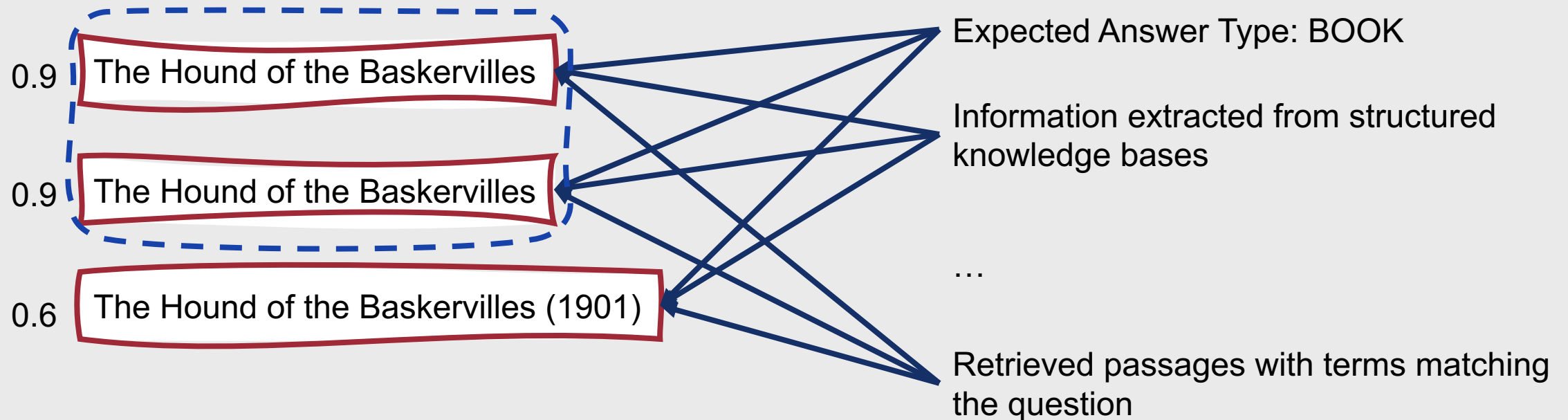
# Stage 3: Candidate Answer Scoring



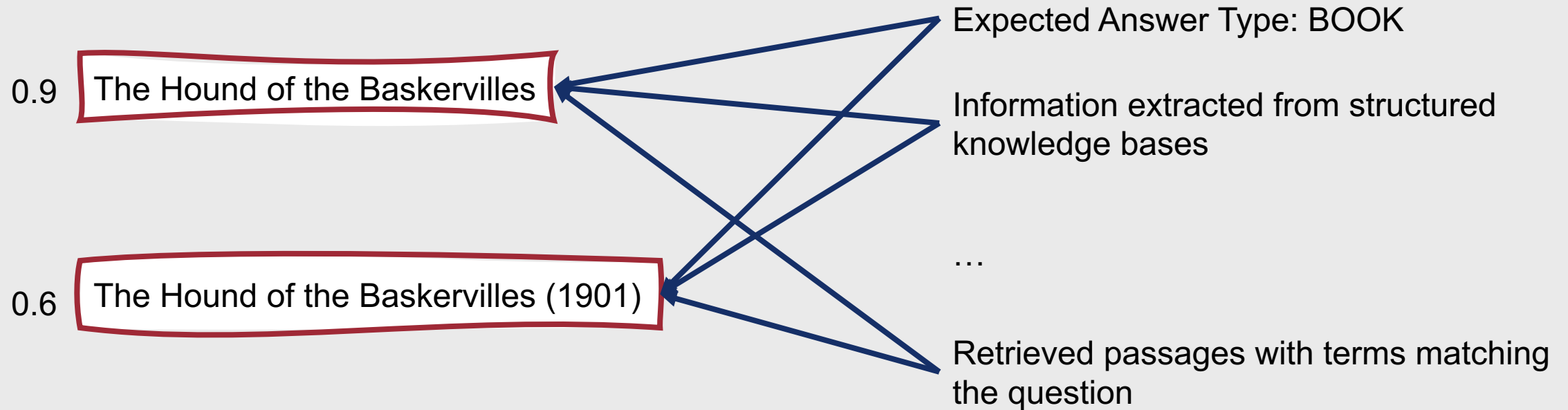
# Stage 4: Answer Merging and Scoring



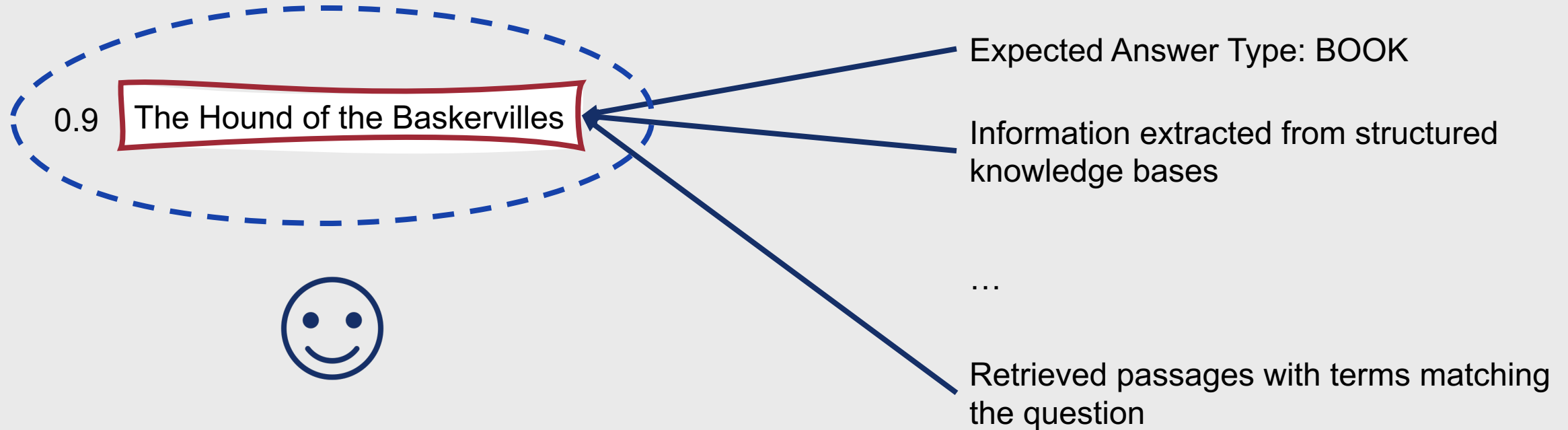
# Stage 4: Answer Merging and Scoring



# Stage 4: Answer Merging and Scoring

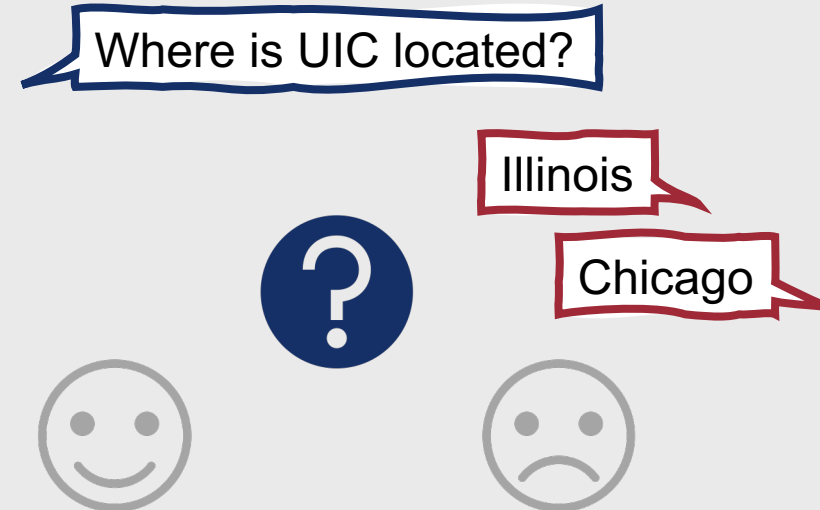


# Stage 4: Answer Merging and Scoring



# How are question answering systems evaluated?

- Common metric for factoid question answering: **Mean Reciprocal Rank**
  - Assumes that gold standard answers are available for test questions
  - Assumes that systems return a short ranked list of answers



# Mean Reciprocal Rank

- Scores each question according to the reciprocal of the rank of the first correct answer
  - Highest ranked correct answer is ranked fourth  $\rightarrow$  reciprocal rank =  $\frac{1}{4}$
- Assigns a score of 0 to questions with no correct answers returned
- System's overall score is the average of all individual question scores
  - $$\text{MRR} = \frac{1}{N} \sum_{i=1}^N \frac{1}{r_i}$$

# Mean Reciprocal Rank

Where is UIC located? ← Question

Gold Standard → Chicago

# Mean Reciprocal Rank

Where is UIC located? ← Question

Gold Standard → Chicago

Prediction	Rank
Illinois	1
West Loop	2
Chicago	3
Little Italy	4

# Mean Reciprocal Rank

Where is UIC located? ← Question

Gold Standard → Chicago

Prediction	Rank
Illinois	1
West Loop	2
Chicago	3
Little Italy	4

# Mean Reciprocal Rank

Where is UIC located? ← Question

Gold Standard → Chicago

Prediction	Rank
Illinois	1
West Loop	2
Chicago	3
Little Italy	4

Reciprocal  
Rank =  $1/3$

# Mean Reciprocal Rank

Where is UIC located?

← Question

Gold Standard →

Chicago

Prediction	Rank
Illinois	1
West Loop	2
Chicago	3
Little Italy	4

Reciprocal  
Rank =  $1/3$

Who is the head of  
UIC's Department of  
Computer Science?

← Question

Gold Standard →

Robert Sloan

Prediction	Rank
Peter Nelson	1
Robert Sloan	2
Natalie Parde	3
Grace Hopper	4

# Mean Reciprocal Rank

Where is UIC located?

← Question

Gold Standard

→ Chicago

Prediction	Rank
Illinois	1
West Loop	2
Chicago	3
Little Italy	4

Reciprocal  
Rank =  $1/3$

Who is the head of  
UIC's Department of  
Computer Science?

← Question

Gold Standard

→ Robert Sloan

Prediction	Rank
Peter Nelson	1
Robert Sloan	2
Natalie Parde	3
Grace Hopper	4

Reciprocal  
Rank =  $1/2$

# Mean Reciprocal Rank

Where is UIC located?

← Question

Gold Standard →

Chicago

Prediction	Rank
Illinois	1
West Loop	2
Chicago	3
Little Italy	4

Reciprocal  
Rank =  $1/3$

Who is the head of  
UIC's Department of  
Computer Science?

← Question

Gold Standard →

Robert Sloan

Prediction	Rank
Peter Nelson	1
Robert Sloan	2
Natalie Parde	3
Grace Hopper	4

Reciprocal  
Rank =  $1/2$

$$\text{MRR} = \frac{\frac{1}{3} + \frac{1}{2}}{2} = 0.417$$

## Other Evaluation Metrics for Question Answering Systems

- **Exact Match**
  - Remove punctuation and articles
  - Compute the percentage of predicted answers that match the gold standard answer exactly

### Leaderboard

SQuAD2.0 tests the ability of a system to not only answer reading comprehension questions, but also abstain when presented with a question that cannot be answered based on the provided paragraph.

Rank	Model	EM	F1
	Human Performance <i>Stanford University</i> (Rajpurkar & Jia et al. '18)	86.831	89.452
1 Jun 04, 2021	IE-Net (ensemble) RICOH_SRCB_DML	90.939	93.214
2 Feb 21, 2021	FPNet (ensemble) Ant Service Intelligence Team	90.871	93.183
3 May 16, 2021	IE-NetV2 (ensemble) RICOH_SRCB_DML	90.860	93.100
4 Apr 06, 2020	SA-Net on Albert (ensemble) QIANXIN	90.724	93.011

## Other Evaluation Metrics for Question Answering Systems

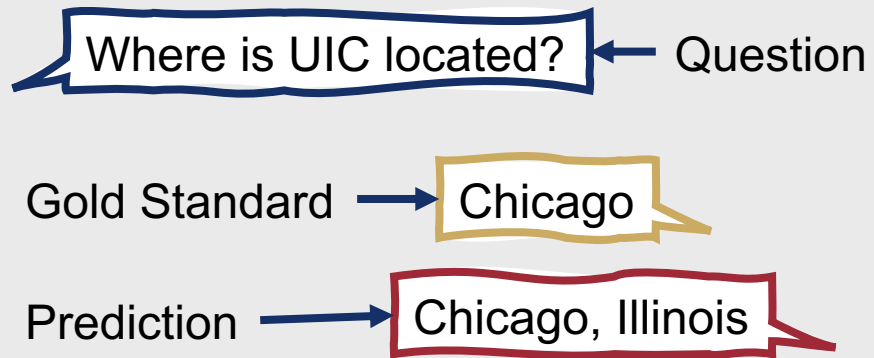
- **F<sub>1</sub> Score**
  - Remove punctuation and articles
  - Treat the predicted and gold standard answers as bags of tokens
  - True positives: Tokens that exist in both the gold standard and predicted answers
  - Average F<sub>1</sub> over all questions

### Leaderboard

SQuAD2.0 tests the ability of a system to not only answer reading comprehension questions, but also abstain when presented with a question that cannot be answered based on the provided paragraph.

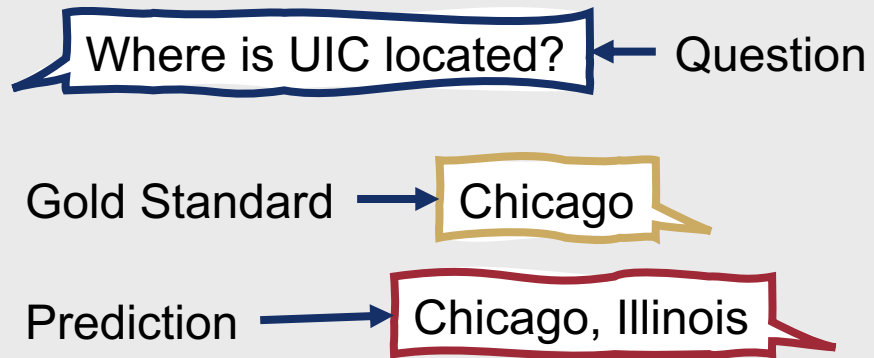
Rank	Model	EM	F1
	Human Performance <i>Stanford University</i> (Rajpurkar & Jia et al. '18)	86.831	89.452
1 Jun 04, 2021	IE-Net (ensemble) RICOH_SRCB_DML	90.939	93.214
2 Feb 21, 2021	FPNet (ensemble) Ant Service Intelligence Team	90.871	93.183
3 May 16, 2021	IE-NetV2 (ensemble) RICOH_SRCB_DML	90.860	93.100
4 Apr 06, 2020	SA-Net on Albert (ensemble) QIANXIN	90.724	93.011

# Computing $F_1$ for Question Answering Systems



	Actual True	Actual False
Predicted True		
Predicted False		

# Computing $F_1$ for Question Answering Systems



	Actual True	Actual False
Predicted True	1	1
Predicted False	0	

# Computing $F_1$ for Question Answering Systems

Where is UIC located? ← Question

Gold Standard → Chicago

Prediction → Chicago, Illinois

	Actual True	Actual False
Predicted True	1	1
Predicted False	0	

$$\text{Precision} = \frac{TP}{TP+FP} = \frac{1}{1+1} = 0.5$$

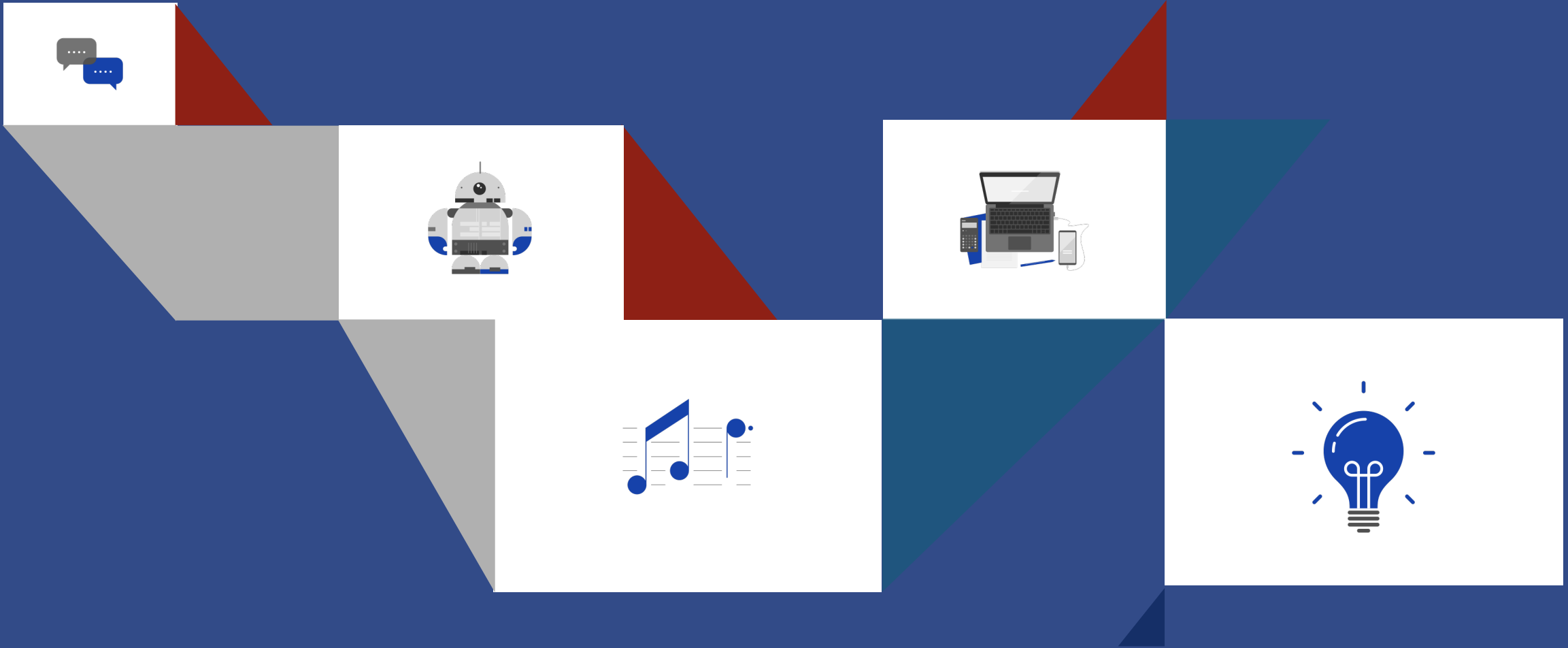
$$\text{Recall} = \frac{TP}{TP+FN} = \frac{1}{1+0} = 1$$

$$F_1 = \frac{2*P*R}{P+R} = \frac{2*0.5*1}{0.5+1} = 0.67$$



# Summary: Question Answering

- **Question answering** is the process of retrieving relevant information and fluently presenting it to users in response to their queries
- QA systems often use **knowledge-based** or **information retrieval** methods to formulate answers to questions
- Some systems also use **language modeling** or **rule-/feature-based approaches**
- QA systems are often evaluated using **mean reciprocal rank**, **exact match**, or  **$F_1$  metrics**



# Moving on to ASR and TTS!

# Automated Speech Recognition



"RADIO REX" by Leo Reynolds is licensed with CC BY-NC-SA 2.0. To view a copy of this license, visit <https://creativecommons.org/licenses/by-nc-sa/2.0/>



"Amazon Echo" by adambowie is licensed with CC BY-NC-SA 2.0.

# Automated Speech Recognition



I love natural language processing!

# ASR Task: Behind the Scenes

- System design often depends on:
  - **Vocabulary size**
  - **Target listener**
  - **Background noise**
  - **Speaker-class characteristics**

# Publicly Available ASR Corpora

- **LibriSpeech:** 1000 hours of English audiobooks
  - <https://www.openslr.org/12>
- **Switchboard:** 240 hours of English telephone conversations between strangers
  - <https://catalog.ldc.upenn.edu/LDC97S62>
- **CALLHOME:** 60 hours of English telephone conversations between friends
  - <https://ca.talkbank.org/access/CallHome/eng.html>
- **Santa Barbara Corpus of Spoken American English:** Conversations between English speakers in a wide range of settings
  - <https://www.linguistics.ucsb.edu/research/santa-barbara-corpus>
- **CORAAL:** Sociolinguistic interviews with speakers of African American Language
  - <http://lingtools.uoregon.edu/coraal/>
- **CHiME:** English conversations in difficult or noisy settings
  - <https://chimechallenge.github.io/chime6/>
- **HKUST:** 200 hours of Mandarin telephone conversations between friends or strangers
  - <https://catalog.ldc.upenn.edu/LDC2005S15>
- **AISHELL:** 170 hours of Mandarin read speech from various domains
  - <https://www.openslr.org/33/>

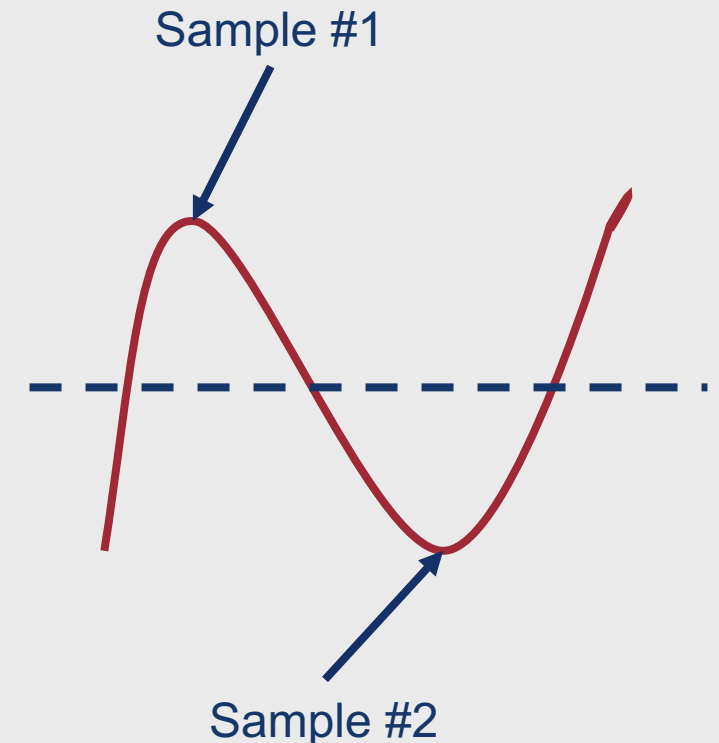
# How do we train an ASR system?

- Acoustic waveforms are commonly converted into sequences of acoustic feature vectors through a process of **sampling** and **quantization**
- Each vector represents a small slice of time in the audio sequence
- Common feature type: **Log mel spectrum vectors**



# Sampling

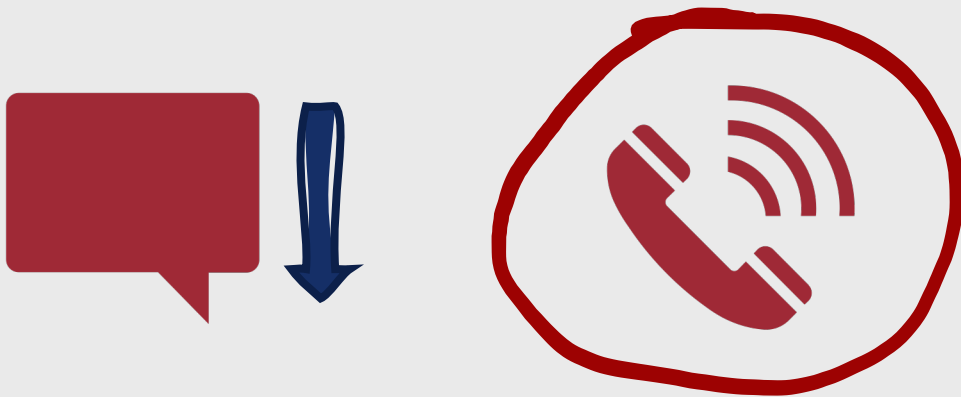
- Measure the signal's amplitude at a specific time point
- **Sampling rate:** Number of samples taken per second
- At least two samples per cycle are needed to measure an audio wave (more samples → higher accuracy)
- Maximum frequency wave for a given sampling rate = half the sampling rate
  - Known as the **Nyquist frequency**
    - For example, a sampling rate of 20,000 Hz would be needed for speech at frequencies of  $\leq 10,000$  Hz
  - 1 Hz = once per second



# Sampling

---

- Can't combine sampling rates when training the same system!
- Downsample sources with higher sampling rates to match the source with the lowest sampling rate



## Common Sampling Rates

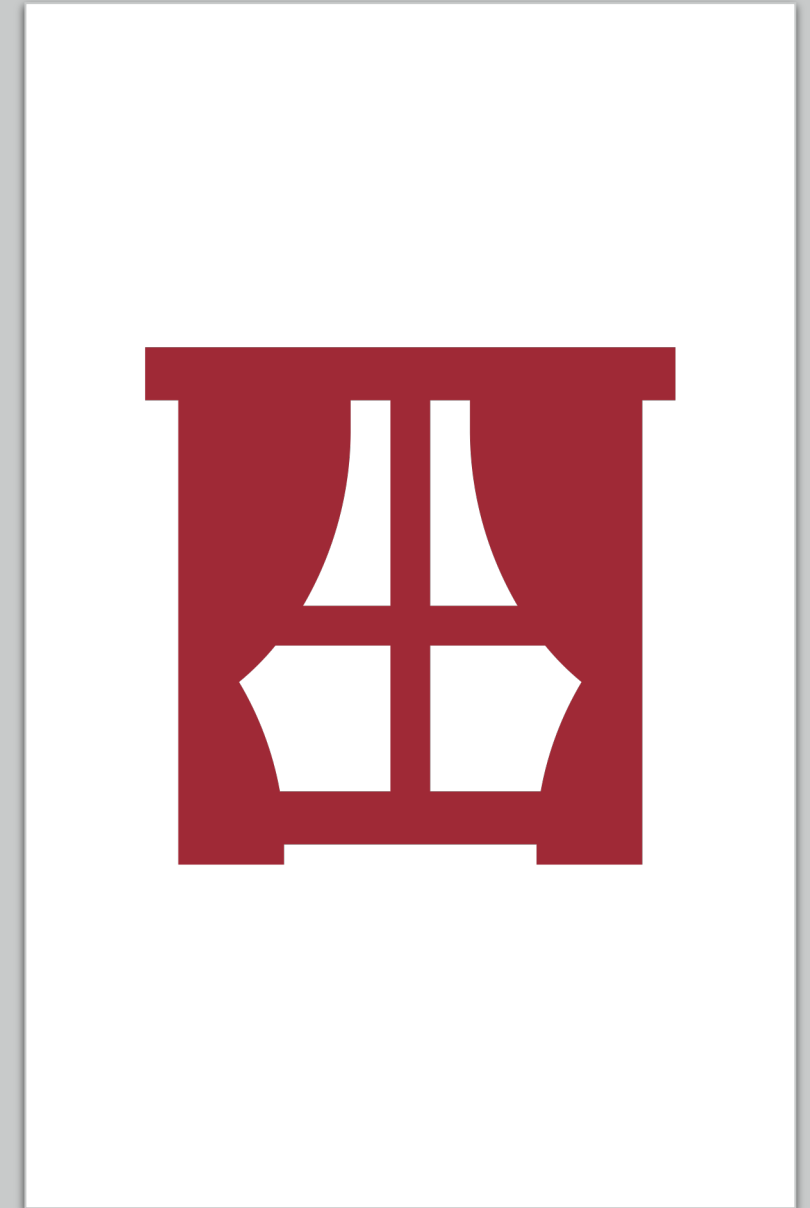
- Human Speech
  - $\leq 10,000$  Hz
- Telephone Speech
  - $\leq 4,000$  Hz

# Quantization

- Real-valued amplitudes are represented as integers
- **Quantum size:**
  - 8 bit (-128 through 127)
  - 16 bit (-32,768 through 32,767)
- Values closer together than the specified granularity are represented identically
- Sample amplitude of waveform  $x$  at time index  $n$  is  $x[n]$

# Windowing

- Extracting features from a roughly stationary window of the quantized waveform (a **speech frame**)
- Three windowing parameters:
  - **Frame size:** Width of the window in milliseconds
  - **Frame Stride:** Offset between successive windows
  - **Shape:** Behavior of the window near its boundaries
- Windowed waveform at time  $n$ ,  $y[n]$ , is obtained by multiplying the signal by the window value at that time
  - $y[n] = w[n]s[n]$



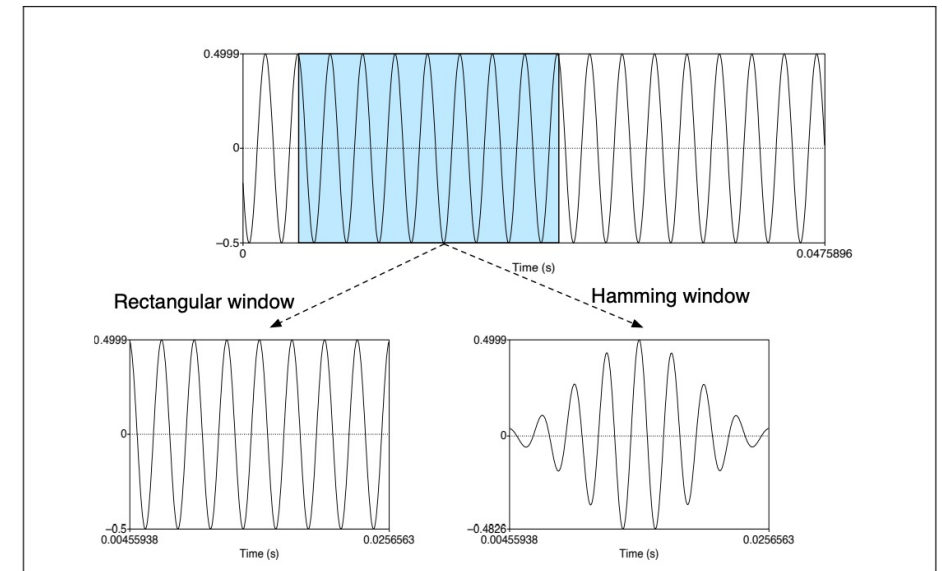
# Window Shape

- **Rectangular**

- Segment of the original signal without any further changes
- $w[n] = \begin{cases} 1, & 0 \leq n \leq L - 1 \\ 0, & \text{otherwise} \end{cases}$
- Abrupt cutoffs at start and end points can pose challenges for Fourier analysis (necessary for feature extraction)

- **Hamming**

- Shrinks the signal values near the start and end points
- $w[n] = \begin{cases} 0.54 - 0.46 \cos(\frac{2\pi n}{L}), & 0 \leq n \leq L - 1 \\ 0, & \text{otherwise} \end{cases}$



**Figure 26.3** Windowing a sine wave with the rectangular or Hamming windows.

Source: Dan Jurafsky and James H. Martin, *Speech and Language Processing*, Chapter 26:  
<https://web.stanford.edu/~jurafsky/slp3/26.pdf>

# Discrete Fourier Transform

---

- Extracts **spectral information** (how much energy the signal contains at different frequency bands) from windowed signals
  - $X[k] = \sum_{n=0}^{N-1} x[n]e^{-j\frac{2\pi}{N}kn}$ , where  $k$  is a given frequency band and  $x[n]$  is a windowed signal at time  $n$
- Can be computed using the **fast Fourier transform**
  - Divide-and-conquer approach for computing the discrete Fourier transform


# Mel Frequencies

- Humans do not perceive differences in pitch equally across all frequency bands
  - Low frequencies → easy to detect differences in pitch
  - High frequencies → harder to detect differences in pitch
- This can be modeled using the **mel scale**
- **Mel:** Unit of pitch
  - Pairs of sounds that are *perceived* as being equidistant in pitch are positioned equidistant to one another on the mel scale

# Mel Frequencies

---

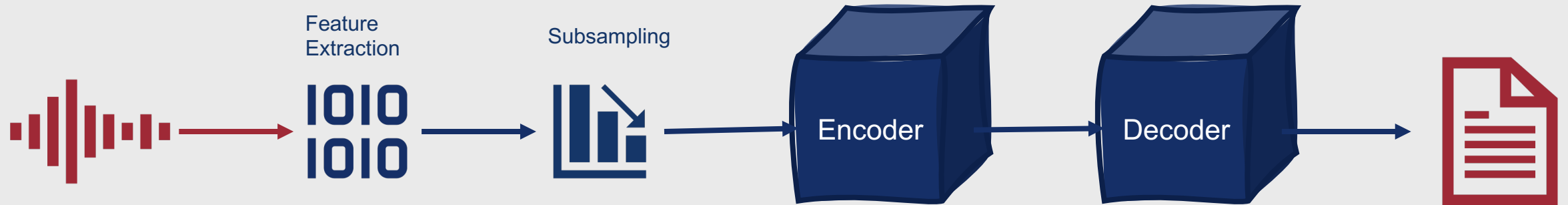
- To convert from raw acoustic frequency to **mel frequency**:
  - $\text{mel}(f) = 1127 \ln \left( 1 + \frac{f}{700} \right)$
- Then, compute energy at each mel frequency band
  - Fine-grained resolution at low frequencies
  - Coarser-grained resolution at high frequencies
- Take the log of mel spectrum values
  - Humans are also less sensitive to changes in higher amplitudes
  - Makes feature estimates less sensitive to input variations (e.g., power variations resulting from variations in microphone distance)



# How do we build speech recognizers?

- Goal: Map from log mel spectral features to letters or morphemes
- Basic architecture: **Attention-based encoder-decoder (AED)** model
  - Well-suited for tasks in which input and output sequences are expected to be of different lengths (e.g., long sequences of acoustic features being mapped to relatively small sequences of letters)
  - Implemented using RNNs or Transformers
  - Very similar to machine translation!

# Encoder-Decoder Speech Recognizer



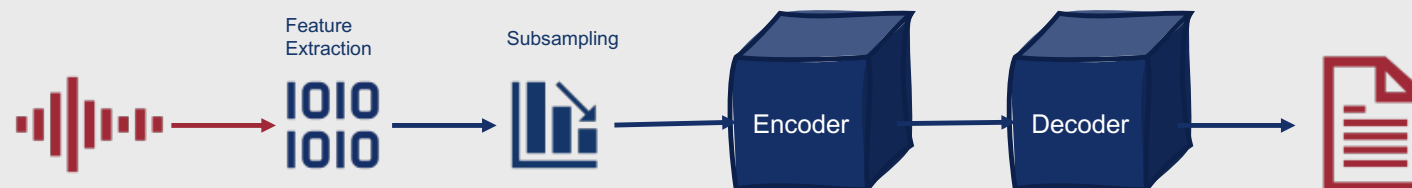
# Subsampling

- Difference between input and output sequence lengths is *very* large for speech recognition!
- Subsampling compresses the input sequence before it is passed into the encoder
- Simplest method: **Low frame rate**<sup>1</sup>
  - At time  $i$ , concatenate the acoustic feature vector  $f_i$  with the previous two acoustic feature vectors,  $f_{i-1}$  and  $f_{i-2}$
  - Delete  $f_{i-1}$  and  $f_{i-2}$
  - Repeat at  $f_{i+3}$  and so forth to convert the original sequence of  $t$  vectors, each with a dimensionality of  $d$ , to a new sequence of  $\frac{t}{3}$  vectors, each with a dimensionality of  $3d$

<sup>1</sup>Pundak, Golan, and Tara N. Sainath. "Lower Frame Rate Neural Network Acoustic Models." *Interspeech 2016* (2016): 22-26.

# How can we improve speech recognition performance?

- Encoder-decoder models for speech recognition implicitly learn character-level language models, but not necessarily good ones
- It's much easier to find text-only training data than paired text-speech training data!
- Solution: We can use large text-based language models to improve speech recognition performance




# Adding Language Models to Speech Recognizers

- Simplest approach:
  - Run beam search to get an **n-best list** of recognized sentences
  - Use a large, text-based language model to rescore each hypothesis,  $Y$ , for the input speech,  $X$
  - Interpolate the original speech recognizer score,  $P(Y|X)$ , with the text-based language model score,  $P_{LM}(Y)$ , using a finetuned weight parameter,  $\lambda$ , and normalize by the number of characters in the hypothesis,  $|Y|_c$ 
    - $\text{score}(Y|X) = \frac{1}{|Y|_c} \log P(Y|X) + \lambda \log P_{LM}(Y)$

# Training Speech Recognizers

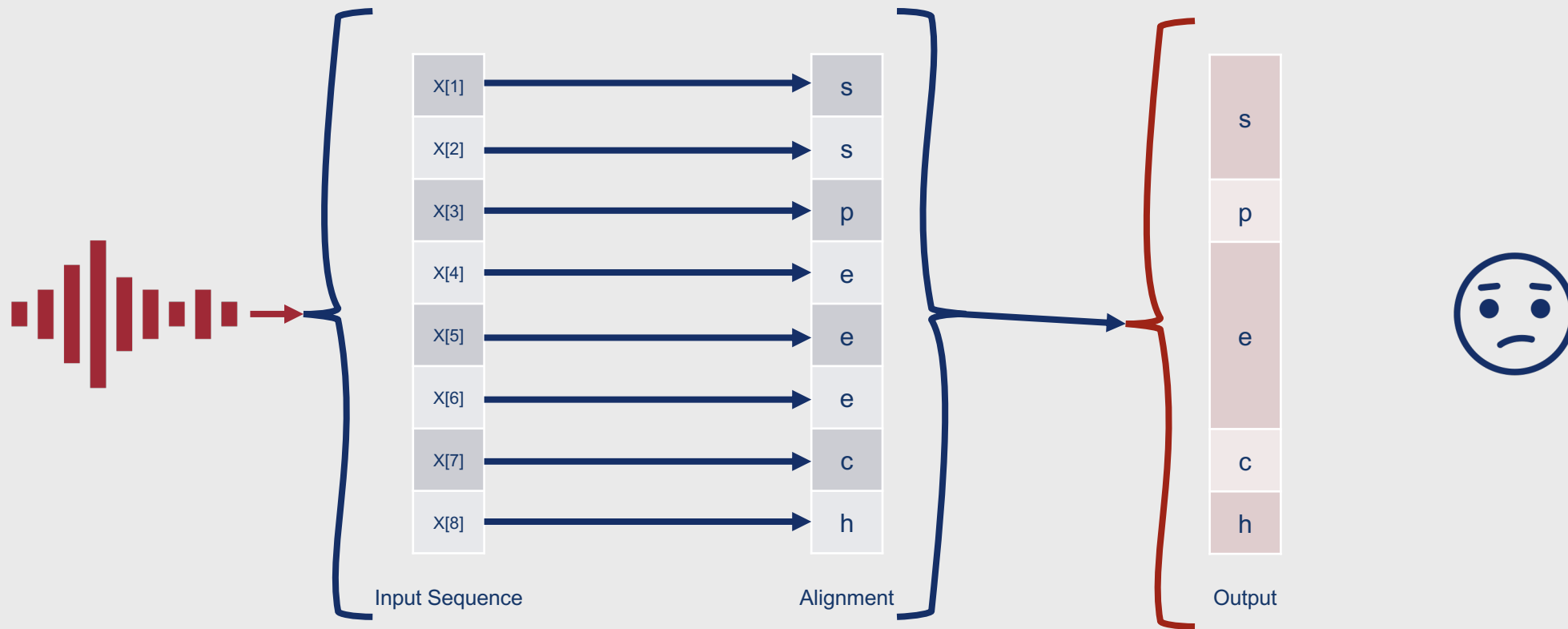
- Normal cross-entropy loss
- **Teacher forcing** is generally employed to improve training efficiency
  - Pass the gold standard previous token into the decoder at a given time step rather than the predicted token



# Connectionist Temporal Classification

- Alignment-based alternative to encoder-decoder models
- Works by predicting a single character for each element of the input sequence, and then merging redundant characters

# Naïve Alignment with Collapsing Function



# Limitations of Naïve Alignment

---

- Can't handle repeated letters
- Can't handle silence

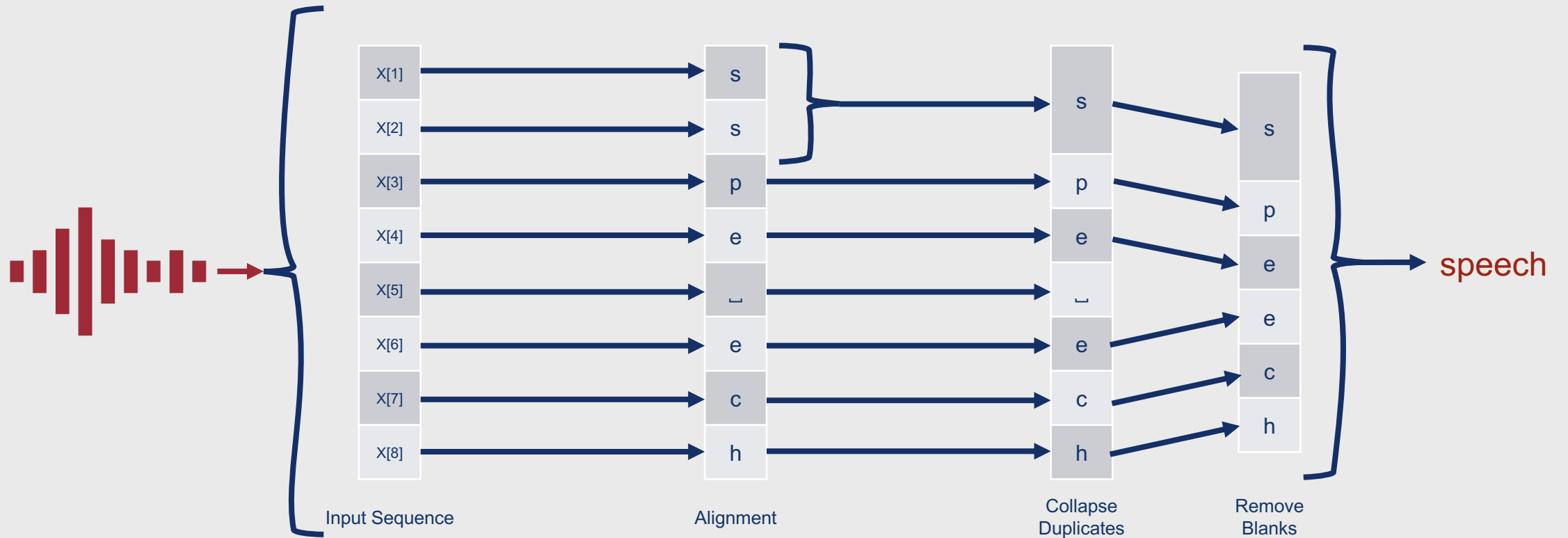


# How does CTC address these issues?



- Adds a blank (“\_”) character to the transcription alphabet
  - Can be inserted between repeated characters to prevent them from being collapsed together
  - Can be inserted when no letter needs to be transcribed (e.g., silence)
- Collapsing function then:
  - Collapses repeated letters
  - *Then* removes blanks

# Connectionist Temporal Classification Algorithm



# Alignment Selection

- Many input alignments can exist!
- How can CTC select from among many possible alignments?
  - Assume conditional independence, letting  $A = \{a_1, \dots, a_T\}$  represent an alignment for  $X$ 
    - $P_{\text{CTC}}(A|X) = \prod_{t=1}^T P(a_t|X)$
  - Greedily select the best alignment,  $\hat{A} = \{\hat{a}_1, \dots, \hat{a}_T\}$ , by choosing the character,  $c$ , that maximizes the probability at each time  $t$ 
    - $\hat{a}_t = \underset{c \in \mathcal{C}}{\operatorname{argmax}} P_t(c|X)$



However,  
the most  
likely  
alignment  
might not  
correspond  
to the most  
likely  
output! 🤯

- The most probable output sequence  $\hat{Y}$  is the output sequence  $Y$  that has the highest sum over the probability of all of its possible alignments,  $A \in B^{-1}(Y)$ 
  - $P_{\text{CTC}}(Y|X) = \sum_{A \in B^{-1}(Y)} P(A|X)$
  - $\hat{Y} = \underset{Y}{\operatorname{argmax}} P_{\text{CTC}}(Y|X)$
- We can approximate this using a Viterbi extension<sup>1</sup> of beam search to avoid summing over all alignments
- Just like with encoder-decoder models, we can interpolate a text-based language model score,  $P_{\text{LM}}(Y)$ , with the CTC probability and a length factor,  $L(Y)$ 
  - $\text{score}_{\text{CTC}}(Y|X) = \log P_{\text{CTC}}(Y|X) + \lambda_1 \log P_{\text{LM}}(Y) \lambda_2 L(Y)$

<sup>1</sup>Hannun, A. Y., Maas, A. L., Jurafsky, D., & Ng, A. Y. (2014). First-pass large vocabulary continuous speech recognition using bi-directional recurrent DNNs. *arXiv preprint arXiv:1408.2873*.

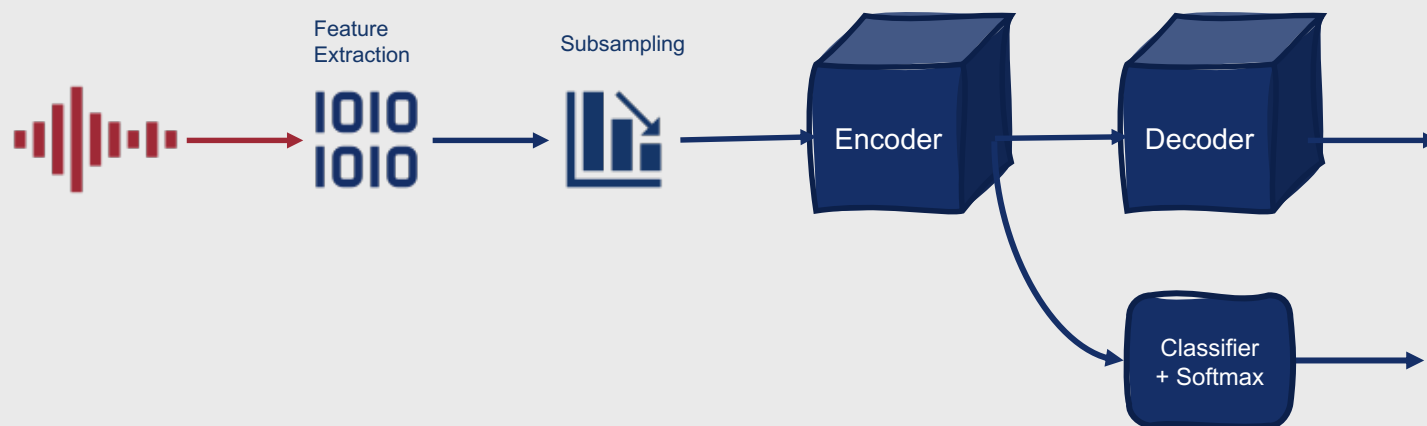
# Training CTC- based ASR Systems

- CTC loss function for a dataset  $D$ :
  - $L_{\text{CTC}} = \sum_{(X,Y) \in D} -\log P_{\text{CTC}}(Y|X)$
- To avoid summing over all possible alignments, we can use a variation of the forward-backward algorithm<sup>1</sup>

<sup>1</sup>Hannun, A. (2017). Sequence modeling with ctc. *Distill*, 2(11), e8.

If we want, we  
can even  
combine CTC  
with an  
encoder-  
decoder  
architecture.

- Weight the two losses using a finetuned weight parameter,  $\lambda$ 
  - $L = -\lambda \log P_{\text{encoder-decoder}}(Y|X) - (1 - \lambda) \log P_{\text{CTC}}(Y|X)$
- Also weight the contributions of the encoder-decoder, CTC, and text-based language model using finetuned weights when making predictions
  - $\hat{Y} = \underset{Y}{\operatorname{argmax}} [\lambda \log P_{\text{encoder-decoder}}(Y|X) - (1 - \lambda) \log P_{\text{CTC}}(Y|X) + \gamma \log P_{\text{LM}}(Y)]$



# How do CTC speech recognizers compare with encoder-decoder speech recognizers?

- **CTC speech recognizers**
  - Generally lower accuracy
  - Can be used for streaming audio signals
- **Encoder-decoder speech recognizers**
  - Generally higher accuracy
  - Cannot be used for streaming since they need to compute attention over the entire input before passing information to the decoder



# Word Error Rate

- Measures the difference between the predicted and gold standard transcriptions
- Based on minimum edit distance, setting penalties for insertions, deletions, and substitutions to 1

# Word Error Rate

- Normalize the total number of word **substitutions**, **insertions**, and **deletions** that are necessary to map the predicted transcript to the gold standard transcript by the number of words in the gold standard transcript
  - Word Error Rate =  $100 * \frac{\text{Insertions} + \text{Substitutions} + \text{Deletions}}{\text{\# Words in Gold Standard Transcript}}$

# Word Error Rate: Case Example

I really love to see his fight on a run

I really love CS 521

run						
a						
on						
fight						
his						
see						
to						
love						
really						
I						
#						
	#	I	really	love	CS	521

# Word Error Rate: Case Example

I really love to see his fight on a run

I really love CS 521

<b>run</b>	10	9	8	7	7	7
<b>a</b>	9	8	7	6	6	6
<b>on</b>	8	7	6	5	5	5
<b>fight</b>	7	6	5	4	4	4
<b>his</b>	6	5	4	3	3	3
<b>see</b>	5	4	3	2	2	2
<b>to</b>	4	3	2	1	1	2
<b>love</b>	3	2	1	0	1	2
<b>really</b>	2	1	0	1	2	3
<b>I</b>	1	0	1	2	3	4
<b>#</b>	0	1	2	3	4	5
	<b>#</b>	<b>I</b>	<b>really</b>	<b>love</b>	<b>CS</b>	<b>521</b>

# Word Error Rate: Case Example

I really love to see his fight on a run

I really love CS 521

run	10	9	8	7	7	7
a	9	8	7	6	6	6
on	8	7	6	5	5	5
fight	7	6	5	4	4	4
his	6	5	4	3	3	3
see	5	4	3	2	2	2
to	4	3	2	1	1	2
love	3	2	1	0	1	2
really	2	1	0	1	2	3
I	1	0	1	2	3	4
#	0	1	2	3	4	5
	#	I	really	love	CS	521

# Word Error Rate: Case Example

I really love to see his fight on a run

I really love CS 521

$$\text{WER} = 100 * \frac{I + S + D}{\text{\# Words in Gold Standard}}$$

run	10	9	8	7	7	7	D=5
a	9	8	7	6	6	6	
on	8	7	6	5	5	5	
fight	7	6	5	4	4	4	
his	6	5	4	3	3	3	
see	5	4	3	2	2	2	S=2
to	4	3	2	1	1	2	
love	3	2	1	0	1	2	#=5
really	2	1	0	1	2	3	
I	1	0	1	2	3	4	
#	0	1	2	3	4	5	
	#	I	really	love	CS	521	

# Word Error Rate: Case Example

I really love to see his fight on a run

I really love CS 521

$$\text{WER} = 100 * \frac{I + S + D}{\text{\# Words in Gold Standard}}$$

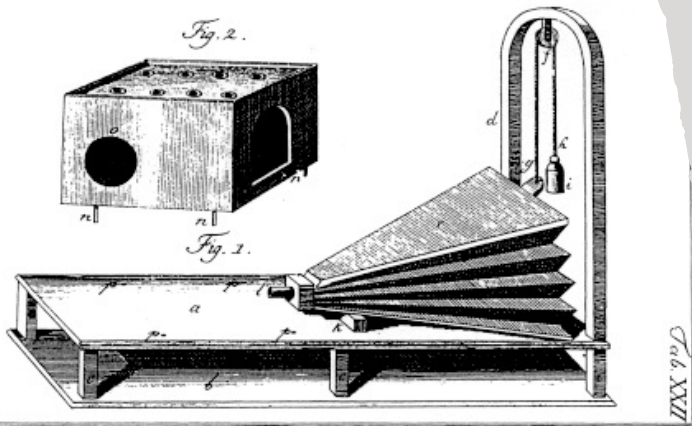
$$\text{WER} = 100 * \frac{7}{5} = 140\%$$



run	10	9	8	7	7	7	D=5
a	9	8	7	6	6	6	
on	8	7	6	5	5	5	
fight	7	6	5	4	4	4	
his	6	5	4	3	3	3	
see	5	4	3	2	2	2	S=2
to	4	3	2	1	1	2	
love	3	2	1	0	1	2	#=5
really	2	1	0	1	2	3	
I	1	0	1	2	3	4	
#	0	1	2	3	4	5	
	#	I	really	love	CS	521	

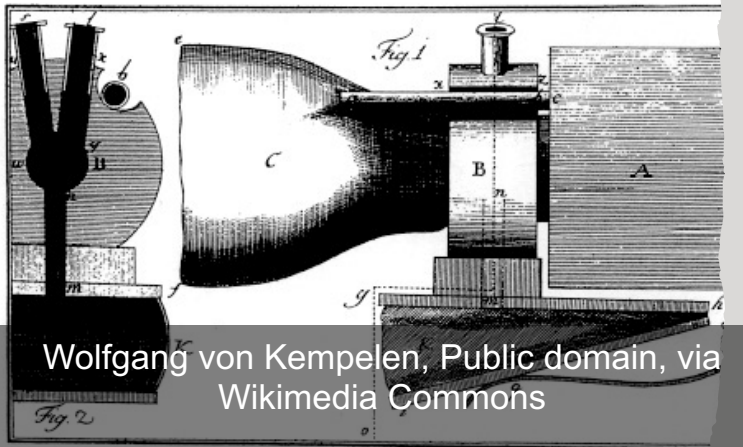
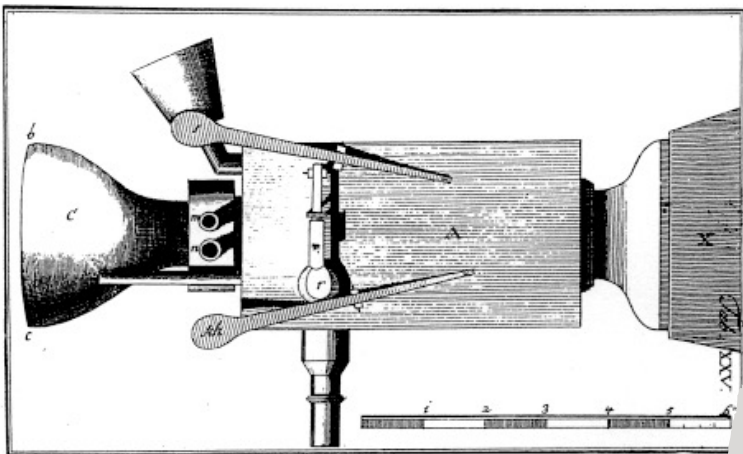
# Matched-Pair Sentence Segment Word Error (MAPSSWE)

- Test for determining whether the difference in word error rate between two ASR systems is statistically significant
- Averages across word error rates for different segments
- Letting  $N_A^i$  be the number of errors made on segment  $i$  by system  $A$ ,  $N_B^i$  be the number of errors made on segment  $i$  by system  $B$ , and  $Z_i = N_A^i - N_B^i$ , we would expect the average of all  $Z$  values when comparing identical systems to be 0
- We can estimate the true average from our sample as:
  - $\widehat{\mu_z} = \frac{\sum_{i=1}^n Z_i}{n}$
- We can estimate the variance as:
  - $\sigma_z^2 = \frac{1}{n-1} \sum_{i=1}^n (Z_i - \mu_z)^2$
- Finally, we can compute:
  - $W = \frac{\widehat{\mu_z}}{\sigma_z / \sqrt{n}}$
- Assuming a large enough  $n$ , we can reject the null hypothesis ( $\mu_z = 0$ ) if  $P(Z \geq |w|) \leq 0.05$



# Text-to-Speech

- Goal: Map from sequences of letters to acoustic signals
- Many practical applications in chatbots and assistive devices
- Earliest speech synthesizer was built in the late 1700s!



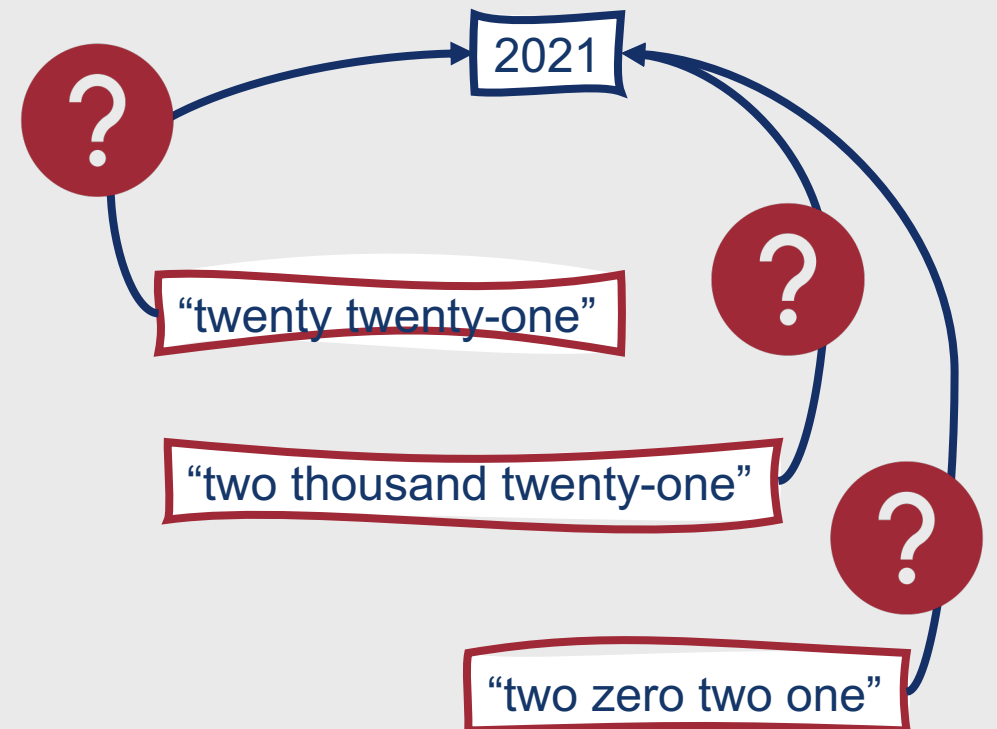
Wolfgang von Kempelen, Public domain, via Wikimedia Commons

# How can we create TTS systems?

- 
- **Encoder-decoder** architecture
  - Usually trained to have one consistent voice
    - Requires less training data than ASR systems, which must be able to generalize across many voices
  - Two subtasks:
    - **Spectrogram prediction:** Mapping from sequences of letters to sequences of mel spectral values
    - **Vocoding:** Mapping from sequences of mel spectral values to waveforms

# TTS Preprocessing

- TTS systems must preprocess text sequences before predicting spectrograms to handle words with pronunciations that vary depending on context
- Pronunciations often depend on the word's **semiotic class**
  - Abbreviation
  - Date
  - Year
  - Money
  - Percentage
- This process is generally referred to as **text normalization**





# Text Normalization

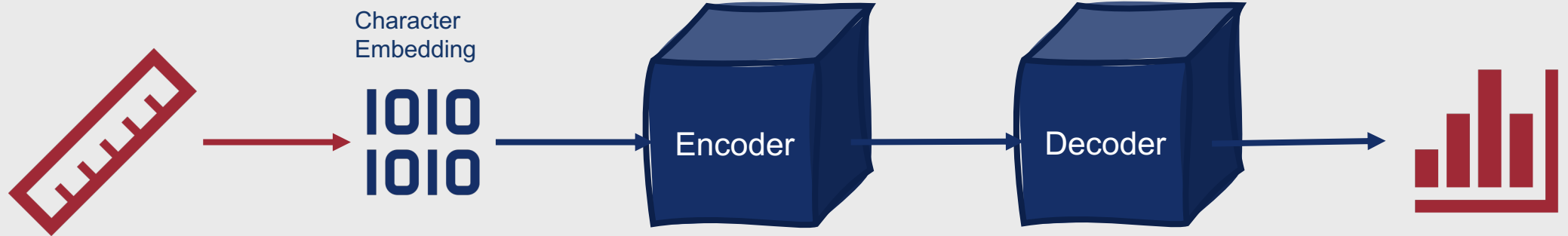
- Can be done using rule-based techniques or an encoder-decoder model
- **Rule-based normalization:**
  - Write regular expressions to detect non-standard words
  - Write rules specifying how different semiotic classes should be verbalized
- **Encoder-decoder normalization:**
  - Train the system to map from an initial realization to a target verbalization

# Rule-based vs. encoder- decoder normalization?

- **Rule-based normalization**
  - Doesn't require any training data
  - Requires potentially time-consuming rule creation by experts
- **Encoder-decoder normalization**
  - Requires a labeled training set
  - Generally achieve higher performance

# Spectrogram Prediction

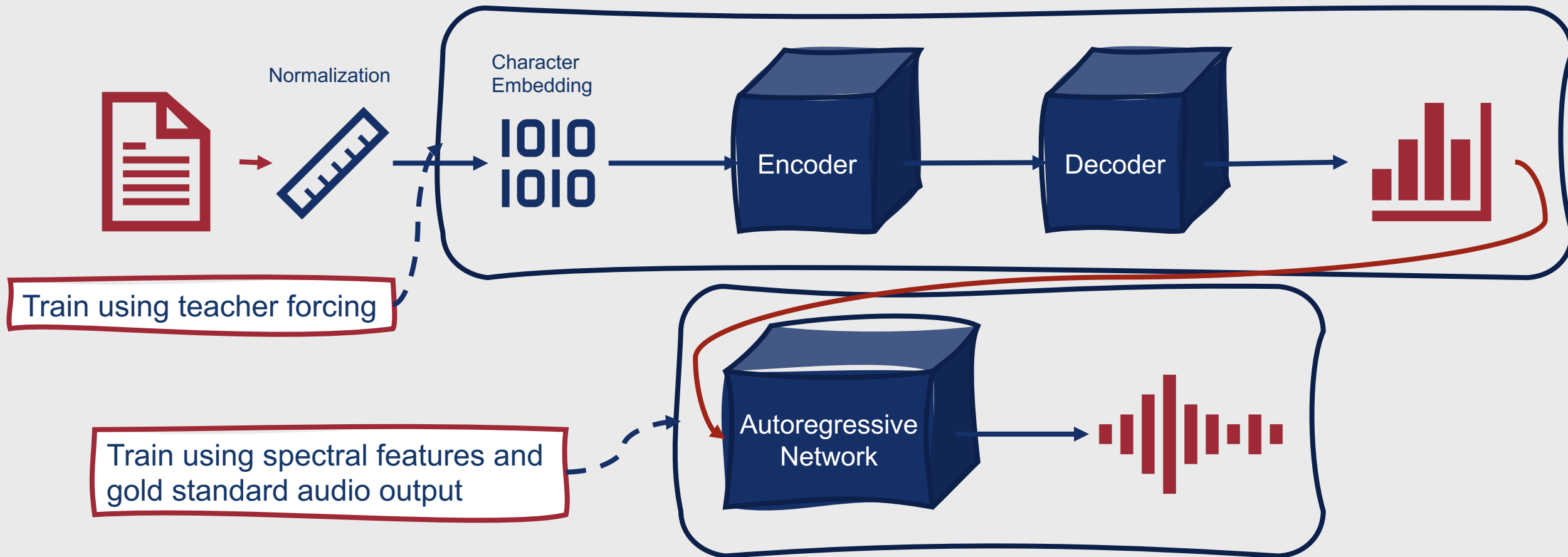
- Can be performed using an encoder-decoder with attention
- Very similar to ASR systems!



# Vocoding

- Goal: Convert a log mel spectrogram to an acoustic waveform
- Can be done using architectures like those used for autoregressive language modeling
- Generally predict **mu-law audio samples** at each timestep
  - Compressed 8-bit acoustic samples

# Putting it all together....



# TTS Evaluation

- No good automated metrics!
- TTS systems are generally evaluated by human listeners
  - Play audio clip
  - Request a **mean opinion score (MOS)**
  - Compare to other systems based on their MOS scores on the same sentences
- **AB testing** can also be used
  - Play a synthesized sentence from system A
  - Play the same synthesized sentence from system B
  - Ask listener to rate which one is better
  - Compare the two systems to see which one has more sentences rated as being better

# Speech processing isn't limited to ASR and TTS!

## Application-specific tasks:

- Wake word detection
- Language identification

## Multiparty interaction tasks:

- Speaker diarization
- Speaker recognition

# Summary: ASR and TTS

- **Automatic speech recognition** is the task of mapping an input acoustic signal to a string of text
- This is often achieved by converting the acoustic signal to an acoustic feature vector through a process of **sampling** and **quantization**, and then passing that feature vector into an encoder-decoder model
- An alternative to using encoder-decoder models is **connectionist temporal classification**, which outputs a character for each input frame and then collapses these together to produce an alignment between input signal and output text
- ASR performance is often evaluated using **word error rate**
- **Text-to-speech synthesizers** map strings of text to acoustic signals
- This is often achieved by predicting a **spectrogram** for the text string, and then **vocoding** from sequences of spectral values to acoustic waveforms
- TTS systems are often evaluated by **human raters**