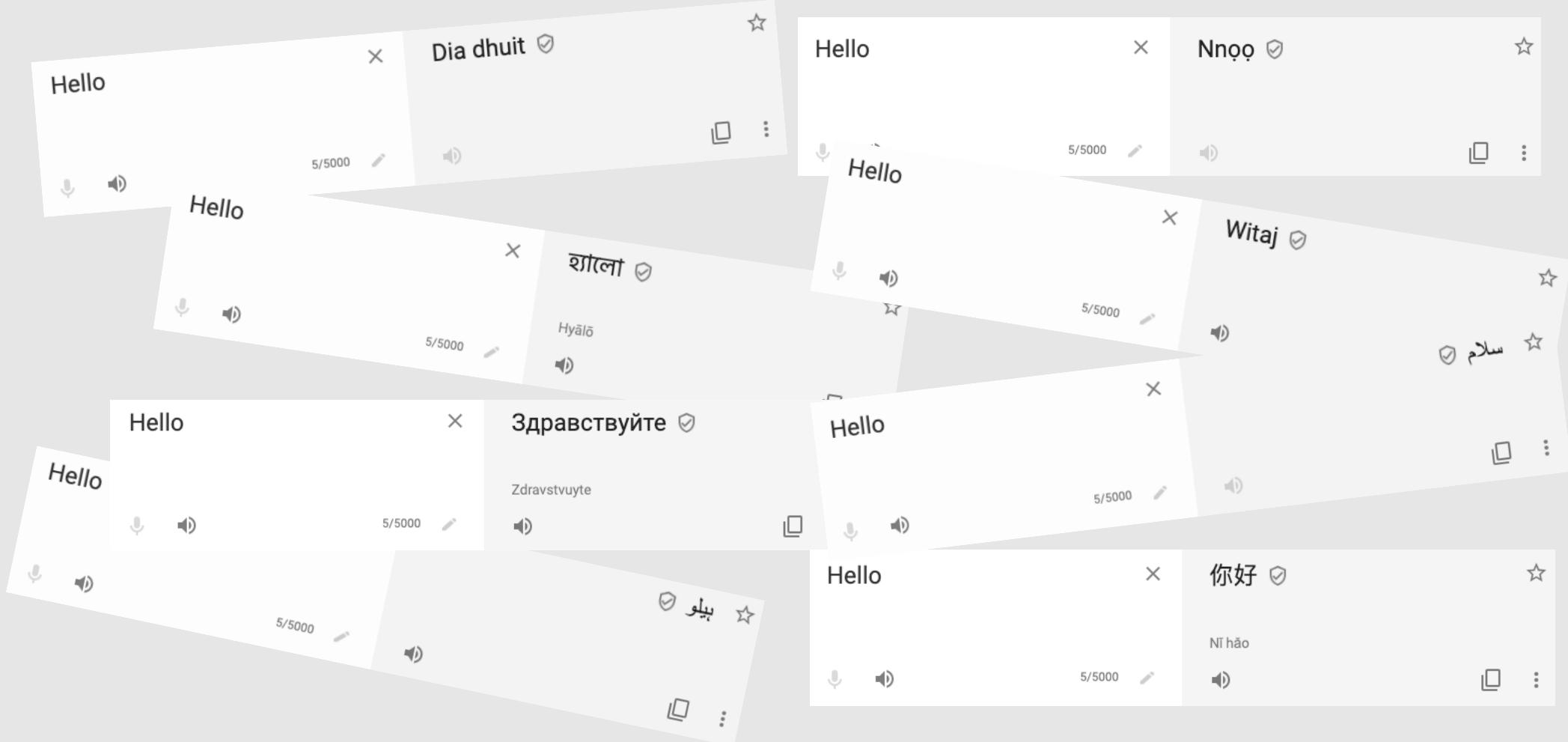


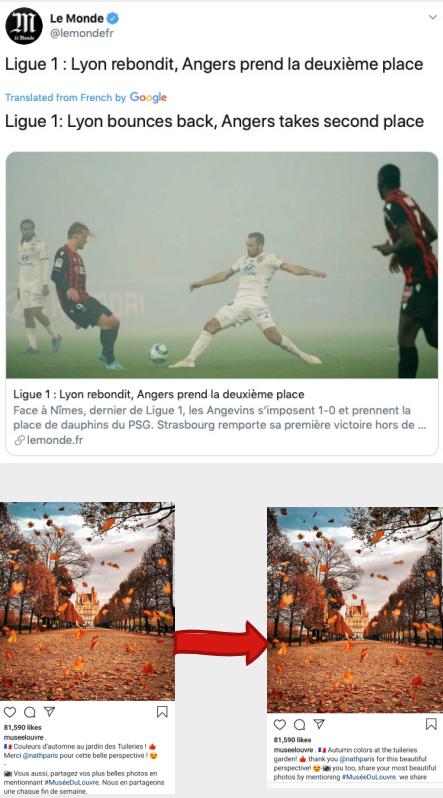
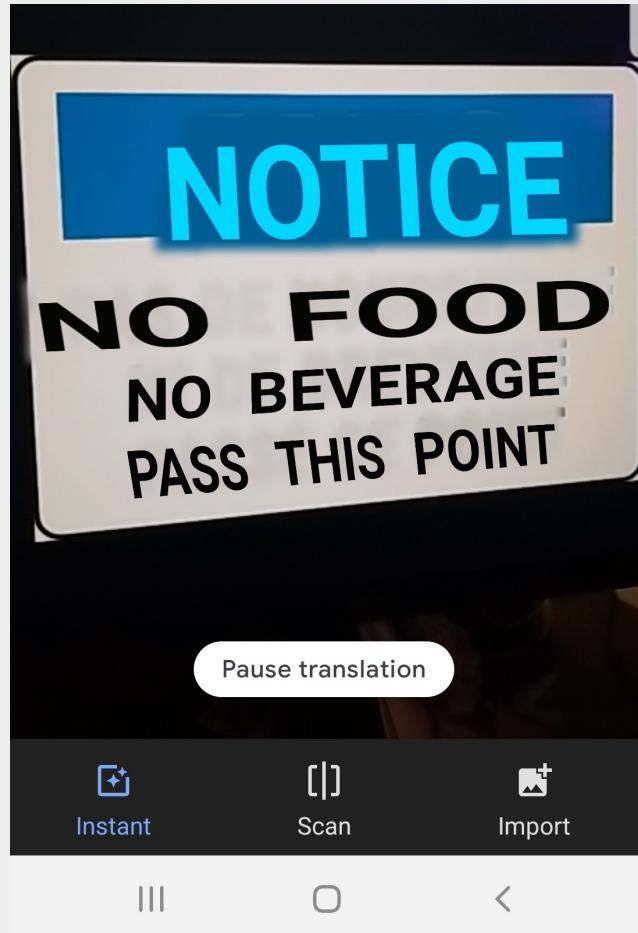
# **Machine Translation, Question Answering and Encoder-Decoder Models**

Natalie Parde

UIC CS 521

**Machine Translation:** The process of automatically converting a text from one language to another.





Machine translation is increasingly ubiquitous, but also challenging for many reasons.

Structural and lexical differences between languages

Differences in word order

Morphological differences

Stylistic and cultural differences

# Cross-Linguistic Similarities and Differences

- **Typological Differences:**
  - Systematic structural differences between languages
- **Morphological Differences:**
  - Number of morphemes per word
    - **Isolating languages:** Each word generally has one morpheme
    - **Polysynthetic languages:** Each word may have many morphemes
  - Degree to which morphemes can be segmented
    - **Agglutinative languages:** Morphemes have well-defined boundaries
    - **Fusion languages:** Morphemes may be conflated with one another



# Cross-Linguistic Similarities and Differences

- **Syntactic Differences:**

- Primary difference between languages: Word order
  - **SVO languages:** Verb tends to come between the subject and object
  - **SOV languages:** Verb tends to come at the end of basic clauses
  - **VSO languages:** Verb tends to come at the beginning of basic clauses
- Languages with similar basic word order also tend to share other similarities
  - SVO languages generally have prepositions
  - SOV languages generally have postpositions

- **Differences in Argument Structure and Linking**



**Verb-framed languages:** Mark the direction of motion on the verb, leaving its satellites (particles, prepositional phrases, and adverbial phrases) to mark the manner of motion



**Satellite-framed languages:** Mark the direction of motion on the satellite, leaving the verb to mark the manner of motion

The bottle floated out.

La botella salió flotando.

The bottle exited floating.

# Cross-Linguistic Similarities and Differences

- **Differences in Permissible Omissions:**
  - **Pro-Drop languages:** Can omit pronouns when talking about certain referents
  - Some pro-drop languages permit more pronoun omission than others
    - **Referentially dense** and **sparse** languages
  - Converting text from pro-drop languages (e.g., Japanese) to non-pro-drop languages (e.g., English) requires that all missing pronoun locations are identified and their appropriate **anaphors** recovered
- **Differences in noun-adjective order**
  - Blue house → Maison bleue
- **Differences in homonymy and polysemy**
- **Differences in grammatical constraints**
  - Some languages require gender for nouns
  - Some languages require gender for pronouns
- **Lexical gaps**
  - No word or phrase in the target language can express the meaning of a word in the source language

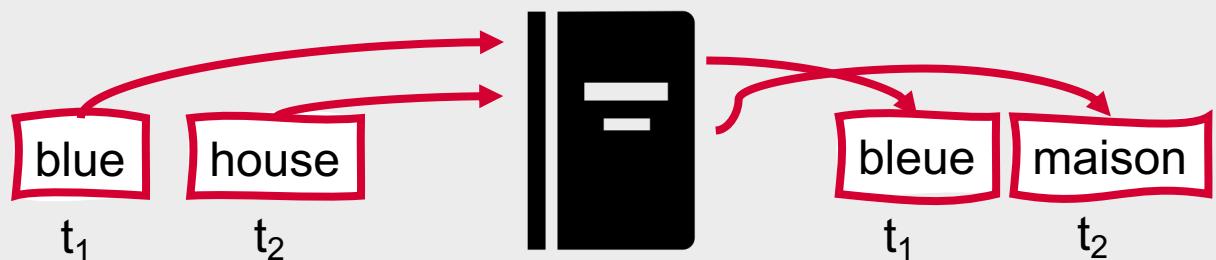
# Machine Translation

- Classical Machine Translation
  - Direct translation
  - Transfer approaches
  - Interlingua approaches
  - Statistical methods
- Modern Machine Translation
  - Encoder-decoder models



# Classical Machine Translation

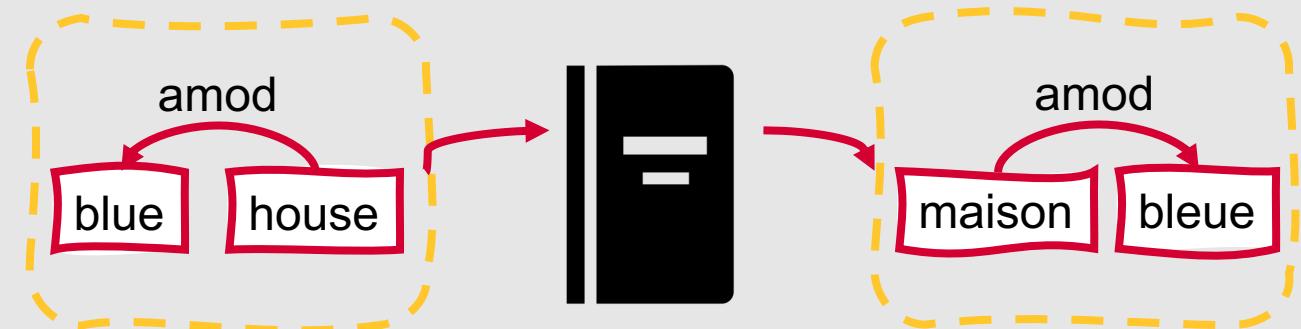
- **Direct translation**
  1. Take a large bilingual dictionary
  2. Proceed through the source text word by word
  3. Translate each word according to the dictionary
- No intermediate structures
- Simple reordering rules may be applied
  - For example, moving adjectives so that they are after nouns when translating from English to French
- Dictionary entries may be relatively complex
  - Rule-based programs for translating a word to the target language



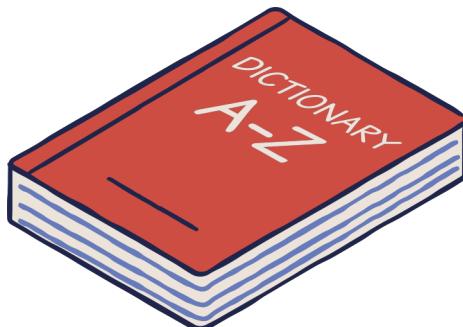
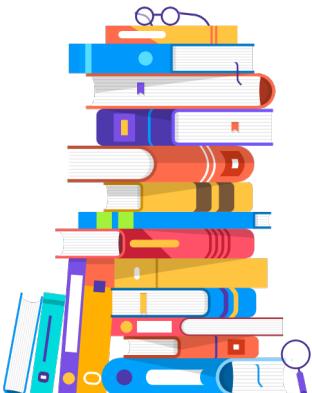
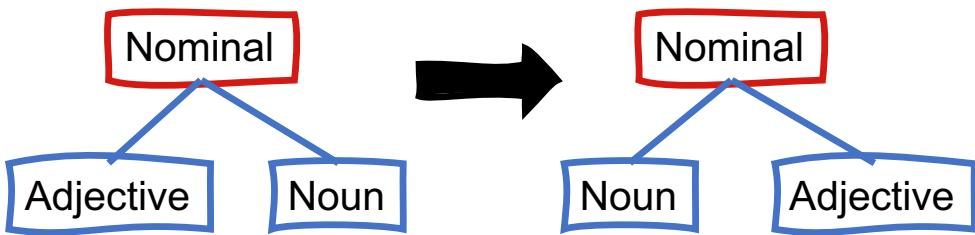
# Classical Machine Translation

---

- **Transfer approaches**
  - Parse the input text
  - Apply rules to transform the source language parse structure into a target language parse structure
- Two subcategories of transformations:
  - **Syntactic** transfer
  - **Lexical** transfer



# Transfer Approaches



- **Syntactic Transfer:** Modifies the source parse tree to resemble the target parse tree
  - For some languages, may also include **thematic structures**
    - Directional or locative prepositional phrases vs. recipient prepositional phrases
- **Lexical Transfer:** Generally based on a bilingual dictionary
  - As with direct translation, dictionary entries can be complex to accommodate many possible translations

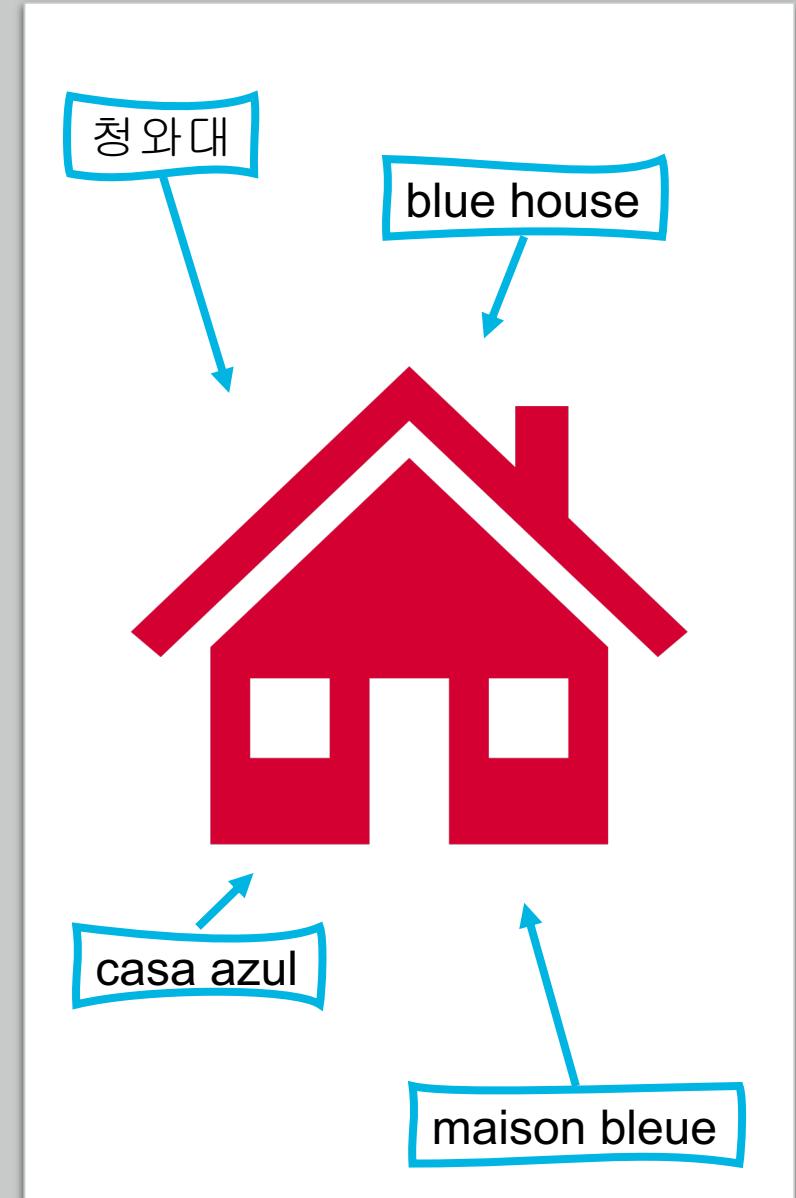
# Classical Machine Translation

- **Interlingua approaches**
  - Convert the source language text into an abstract meaning representation
  - Generate the target language text based on the abstract meaning representation
  - Require more analysis work than transfer approaches
    - Semantic analysis
    - Sentiment analysis
  - No need for syntactic or lexical transformations



# Interlingua Approaches

- Goal: Represent all sentences that mean the same thing in the same way, regardless of language
- What kind of representation scheme should be used?
  - Classical approaches:
    - First-order logic
    - Semantic primitives
    - Event-based representation
  - More recently, neural machine translation models follow a similar intuition



# When to use each classical approach?

## Direct Translation

- Pros:
  - Simple
  - Easy to implement
- Cons:
  - Cannot reliably handle long-distance reorderings
  - Cannot handle reorderings involving phrases or larger structures
  - Too focused on individual words

## Transfer Approaches

- Pros:
  - Can handle more complex language phenomena than direct translation
- Cons:
  - Still not sufficient for many cases!

## Interlingua Approaches

- Pros:
  - Direct mapping between meaning representation and lexical realization
  - No need for transformation rules
- Cons:
  - Extra (often unnecessary) work
  - Classical approaches require an exhaustive analysis and formalization of the semantics of the domain

---

# Statistical Machine Translation

- Models automatically learn to map from the source language to the target language
  - No need for intermediate transformation rules
  - No need for an explicitly defined internal meaning representation
- Goal: Produce an output that maximizes some function representing translation **faithfulness** and **fluency**
- One possible approach: **Bayesian noisy channel model**
  - Assume a possible target language translation  $t_i$  and a source language sentence  $s$
  - Select the translation  $t'$  from the set of all possible translations  $t_i \in T$  that maximizes the probability  $P(t_i|s)$ , using Bayes' rule
    - $t' = \operatorname{argmax}_{t_i \in T} P(s|t_i)P(t_i)$
  - Often a **phrase-based translation model** is used



# The Phrase- Based Translation Model

- Computes the probability that a given translation  $t_i$  generates the original sentence  $s$  based on its **constituent phrases**
- Stages of phrase-based translation:
  1. Group the words from the source sentence into phrases
  2. Translate each source phrase into a target language phrase
  3. (Optionally) reorder the target language phrases

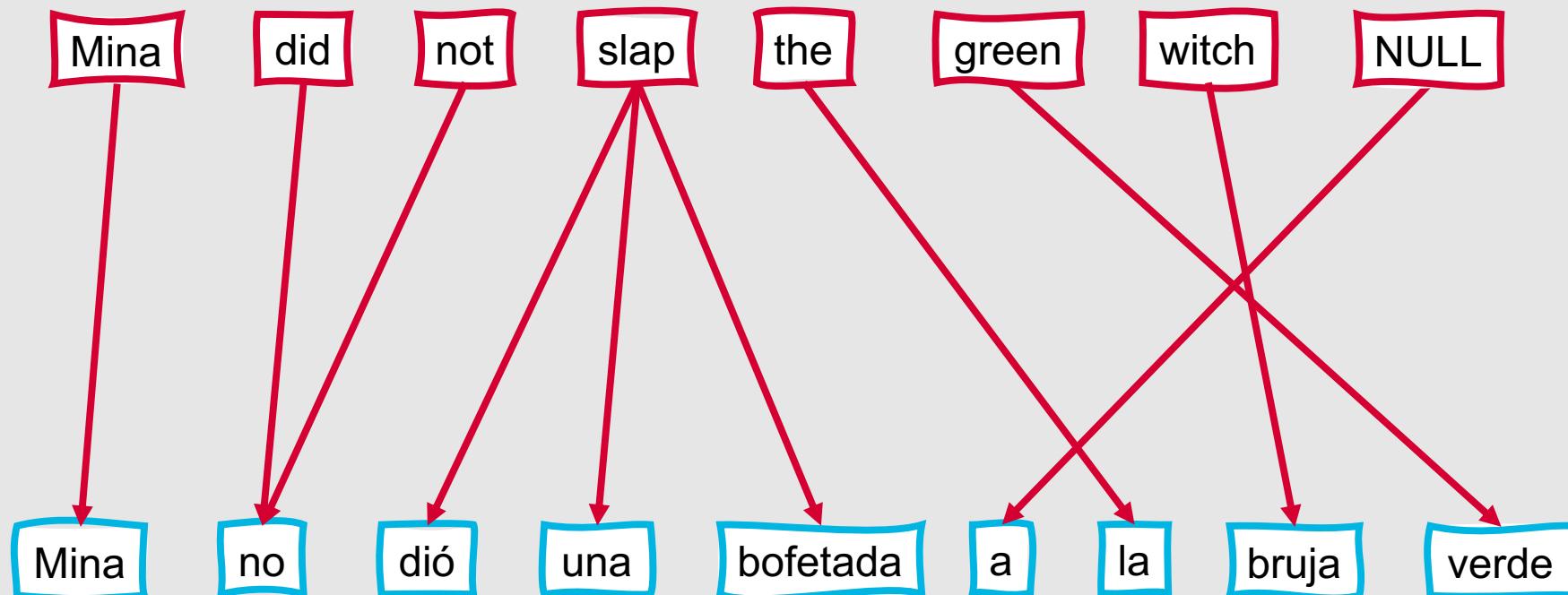
# Probability in Phrase-Based Translation Models

---

- Relies on two probabilities:
  - **Translation probability**
    - Probability of generating a source language phrase from a target language phrase,  $\phi(\bar{t}_i, \bar{s}_i)$
  - **Distortion probability**
    - Probability of two consecutive target language phrases being separated in the source language by a word span of a particular length,  $d(a_i - b_{i-1})$
- To learn these probabilities, we need to train two sets of parameters:
  - $\phi(\bar{t}_i, \bar{s}_i)$
  - $d(a_i - b_{i-1})$
- We learn these using phrase-aligned bilingual training sets

# Decoding for Phrase-Based Machine Translation

- Aligned phrases can be stored in a **phrase-translation** table
- Decoding algorithms** can then search through this table to find the overall translation that maximizes the phrase translation probabilities



# Machine Translation

- Classical Machine Translation
  - Direct translation
  - Transfer approaches
  - Interlingua approaches
  - Statistical methods
- Modern Machine Translation
  - Encoder-decoder models



---

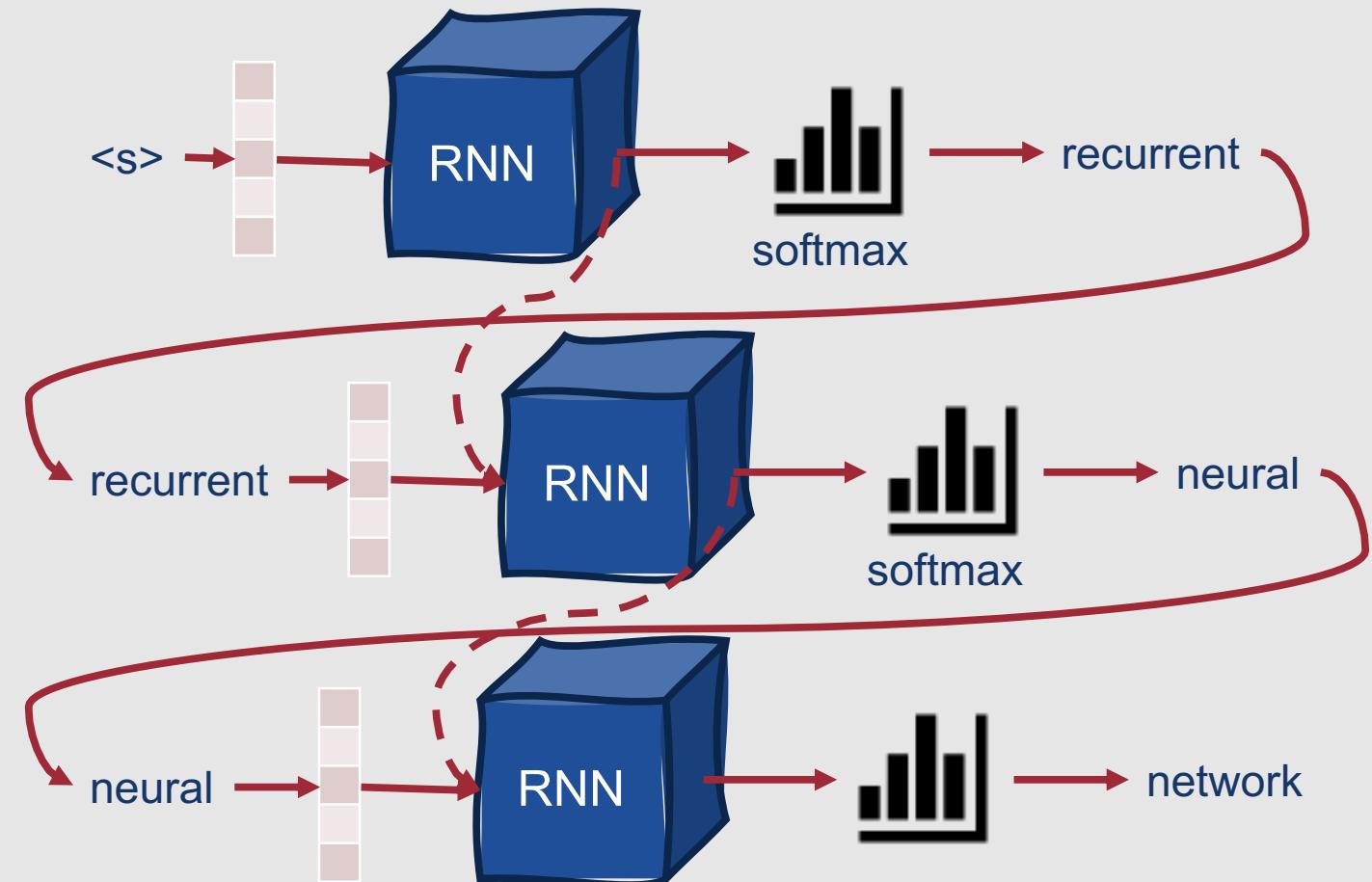
# Encoder-Decoder Models

- Generate **contextually-appropriate, arbitrary-length** output sequences
- Basic premise:
  - Use a neural network to **encode an input to an internal representation**
  - Pass that internal representation as input to a second neural network
  - Use that neural network to **decode the internal representation to a task-specific output sequence**
- This method allows networks to be trained in an **end-to-end** fashion

# Where did this idea come from?

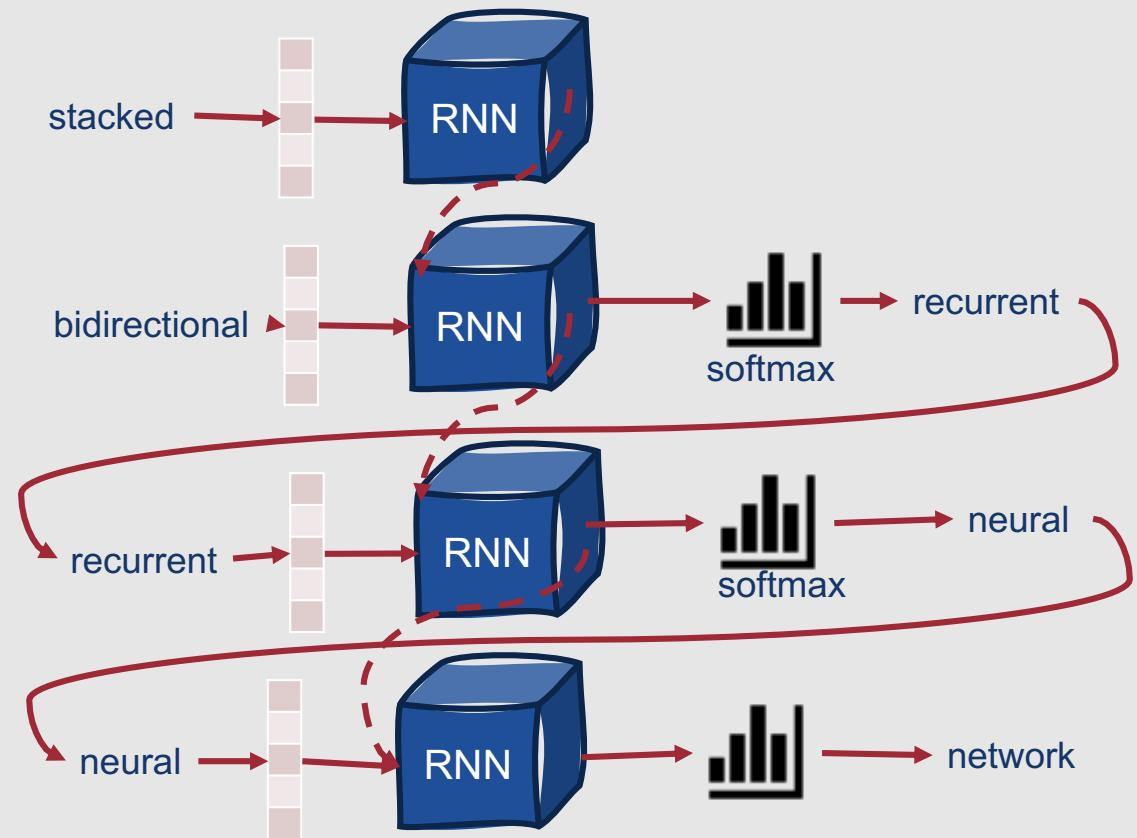
Recall our discussion of autoregressive generation:

- Start with a seed token (e.g., <s>)
- Predict the most likely next word in the sequence
- Use that word as input at the next timestep
- Repeat until an end token (or max length) is reached



# This setup can be extended to generate text given a specific prefix....

- Pass the specified prefix through the language model, in sequence
- End with the hidden state corresponding to the last word of the prefix
- Start the autoregressive process at that point
- **Goal: Output sequence should be a reasonable completion of the prefix**





# We can build upon this idea to transform one type of sequence to another.

- Machine translation example:
  1. Take a **sequence of text from Language #1**
  2. Take the **translation of that text from Language #2**
  3. **Concatenate the two sequences**, separated by a marker
  4. Use these concatenated sequences to **train the autoregressive model**
  5. Test the model by **passing in only the first part of a concatenated sequence** (text from Language #1) and checking to see what the generated completion (text from Language #2) looks like

# Intuition: Machine Translation

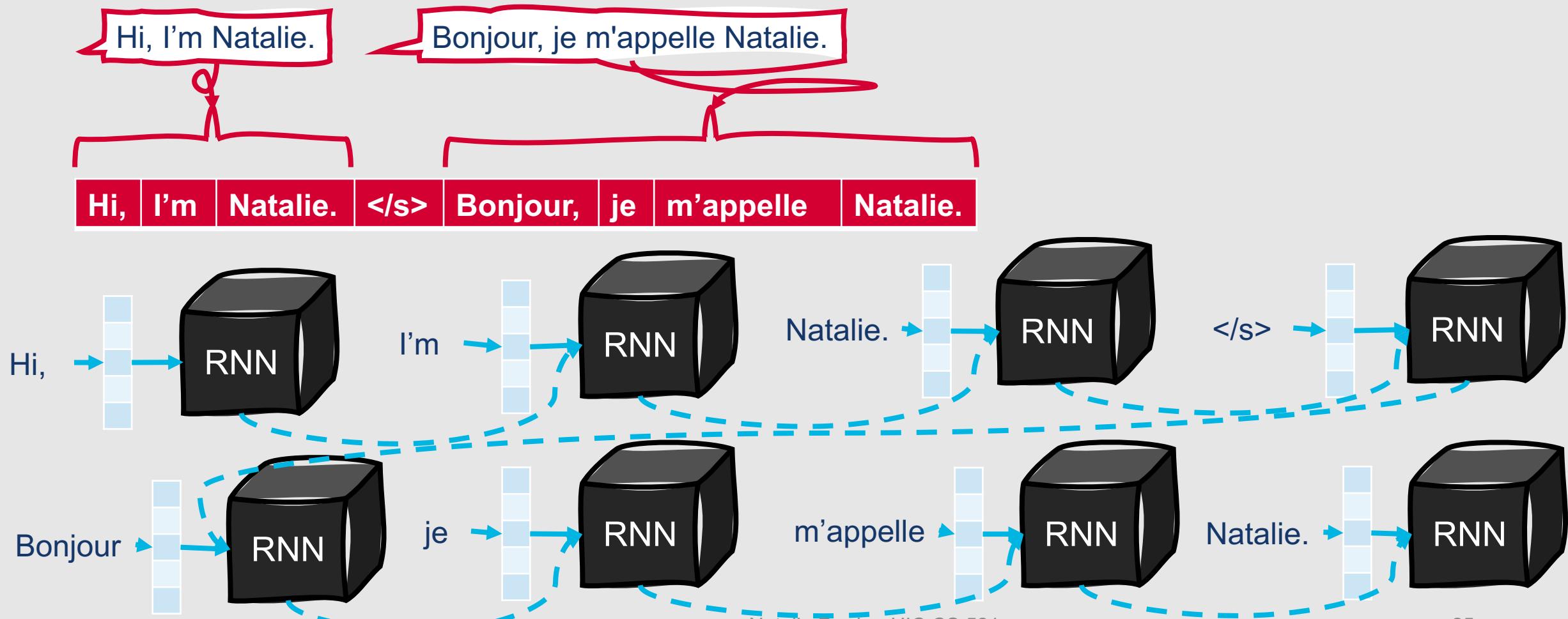
Hi, I'm Natalie.

Bonjour, je m'appelle Natalie.

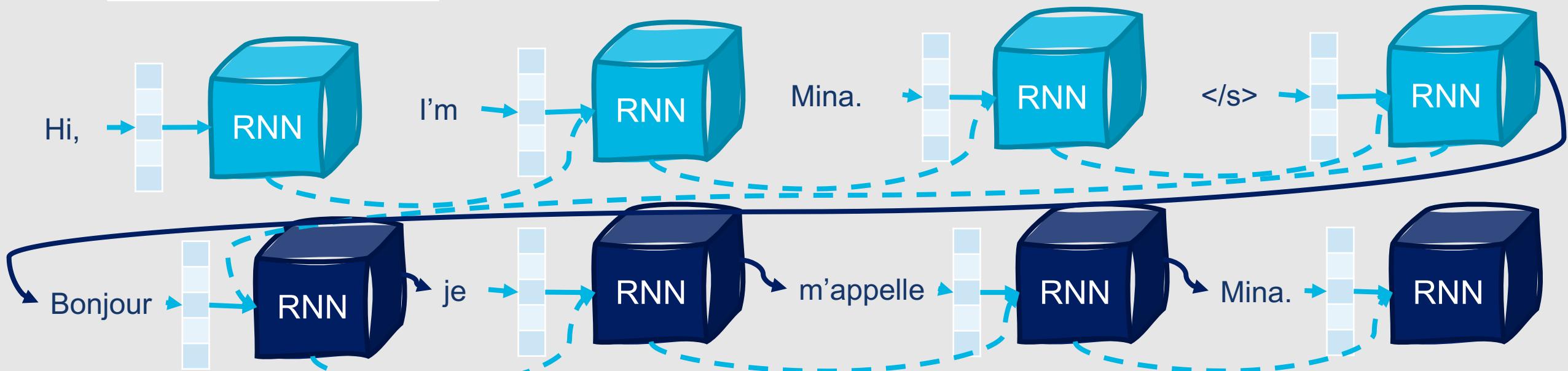
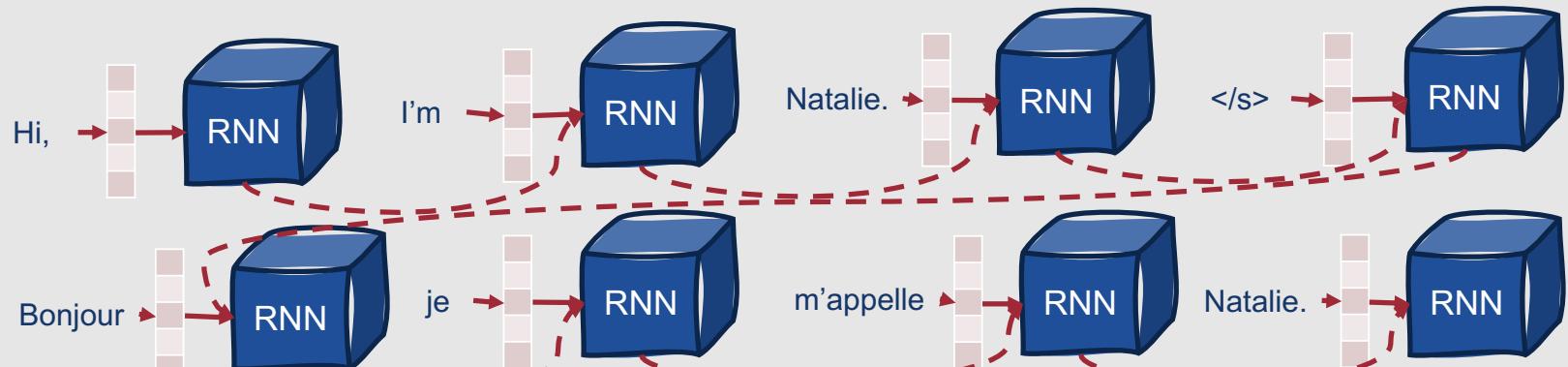
# Intuition: Machine Translation



# Intuition: Machine Translation



# Intuition: Machine Translation





# Key Elements of an Encoder- Decoder Network

- **Encoder**
  - Accepts an input sequence,  $x_1^n$
  - Generates a sequence of contextualized representations,  $h_1^n$
- **Context vector**
  - A function,  $c$ , of  $h_1^n$  that conveys the basic meaning of  $x_1^n$  to the decoder
  - (Might just be equivalent to  $h_1^n$ )
- **Decoder**
  - Accepts  $c$  as input
  - Generates an arbitrary-length sequence of hidden states,  $h_1^m$ , from which a corresponding sequence of output states  $y_1^m$  can be obtained

# Encoders

---

- Can be any type of neural network
  - Feedforward network
  - CNN
  - RNN
  - LSTM/BiLSTM
  - GRU/BiGRU
  - Transformer
- These networks can be stacked on top of one another



# Decoders

- Need to perform autoregressive generation to produce the output sequence
- Can be any type of sequence processing network
  - RNN
  - LSTM
  - GRU
  - Transformer

- Formally....

- 
- The diagram illustrates the flow of information in a sequence-to-sequence model. It starts with the **Final hidden state of the encoder**, which is labeled  $c = h_n^e$ . A dashed arrow points from this state to the **First hidden state of the decoder**, labeled  $h_0^d = c$ . From this initial state, a dashed arrow points to the **Hidden state at time t**, labeled  $h_t^d = g(\widehat{y_{t-1}}, h_{t-1}^d)$ . This state is then used to produce the **Output at time t**, labeled  $z_t = f(h_t^d)$ , and finally the **Probability distribution at time t**, labeled  $y_t = \text{softmax}(z_t)$ .
- $c = h_n^e$
  - $h_0^d = c$
  - $h_t^d = g(\widehat{y_{t-1}}, h_{t-1}^d)$
  - $z_t = f(h_t^d)$
  - $y_t = \text{softmax}(z_t)$

# Decoders

- Need to perform autoregressive generation to produce the output sequence
- Can be any type of sequence processing network
  - RNN
  - LSTM
  - GRU
  - Transformer
- Formally....

$$\begin{aligned} \bullet & c = h_n^e \\ \bullet & h_0^d = c \end{aligned}$$

$$\begin{aligned} \bullet & h_t^d = g(\widehat{y_{t-1}}, h_{t-1}^d) \\ \bullet & z_t = f(h_t^d) \\ \bullet & y_t = \text{softmax}(z_t) \end{aligned}$$

Embedding for the output sampled from the previous step

Some type of sequence processing model

# Decoders

- Need to perform autoregressive generation to produce the output sequence
- Can be any type of sequence processing network
  - RNN
  - LSTM
  - GRU
  - Transformer
- Formally....
  - $c = h_n^e$
  - $h_0^d = c$

Regular ending steps (activation function applied to hidden state outputs, and softmax applied to activation outputs)

- $h_t^d = g(\widehat{y_{t-1}}, h_{t-1}^d)$
- $z_t = f(h_t^d)$
- $y_t = \text{softmax}(z_t)$

# A couple useful extensions....

- Formally....

- $c = h_n^e$
- $h_0^d = c$

- $h_t^d = g(\widehat{y_{t-1}}, h_{t-1}^d) \rightarrow h_t^d = g(\widehat{y_{t-1}}, h_{t-1}^d, c)$
- $z_t = f(h_t^d)$
- $y_t = \text{softmax}(z_t)$

Make the context vector available at each timestep when decoding, so that its influence doesn't diminish over time

# A couple useful extensions....

- Formally....
    - $c = h_n^e$
    - $h_0^d = c$
    - $h_t^d = g(\widehat{y_{t-1}}, h_{t-1}^d) \rightarrow h_t^d = g(\widehat{y_{t-1}}, h_{t-1}^d, c)$
    - $z_t = f(h_t^d)$
    - $y_t = \text{softmax}(z_t) \rightarrow y_t = \text{softmax}(\widehat{y_{t-1}}, z_t, c)$
- Condition output on not only the hidden state, but the previous output and encoder context (easier to keep track of what's been generated already)

# What other ways can we improve the decoder's output quality?

- Beam search
- Improved context vector
  - Final hidden state tends to be more focused on the end of the input sequence
  - Can be addressed by using bidirectional RNNs, summing the encoder hidden states, or averaging the encoder hidden states



# Beam Search

---

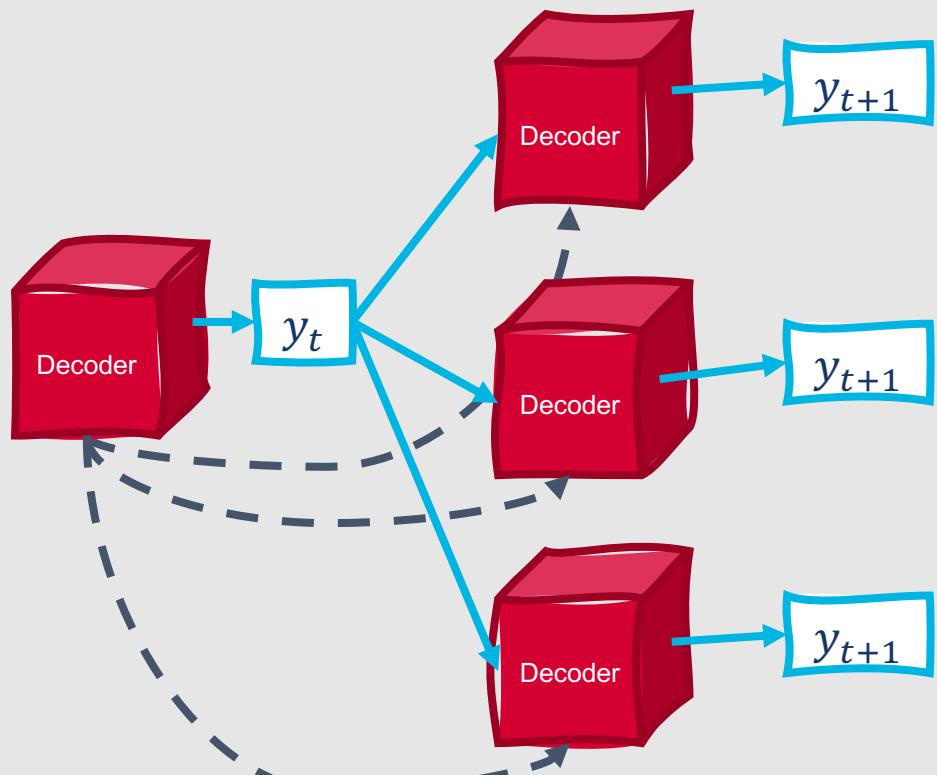
- Selects from multiple possible outputs by framing the task as a **state space search**
- Combines **breadth-first search** with a **heuristic filter**
  - Continually prunes search space to stay a fixed size (**beam width**)
- Results in a set of  $b$  **hypotheses**, where  $b$  is the beam width

# How does beam search work?



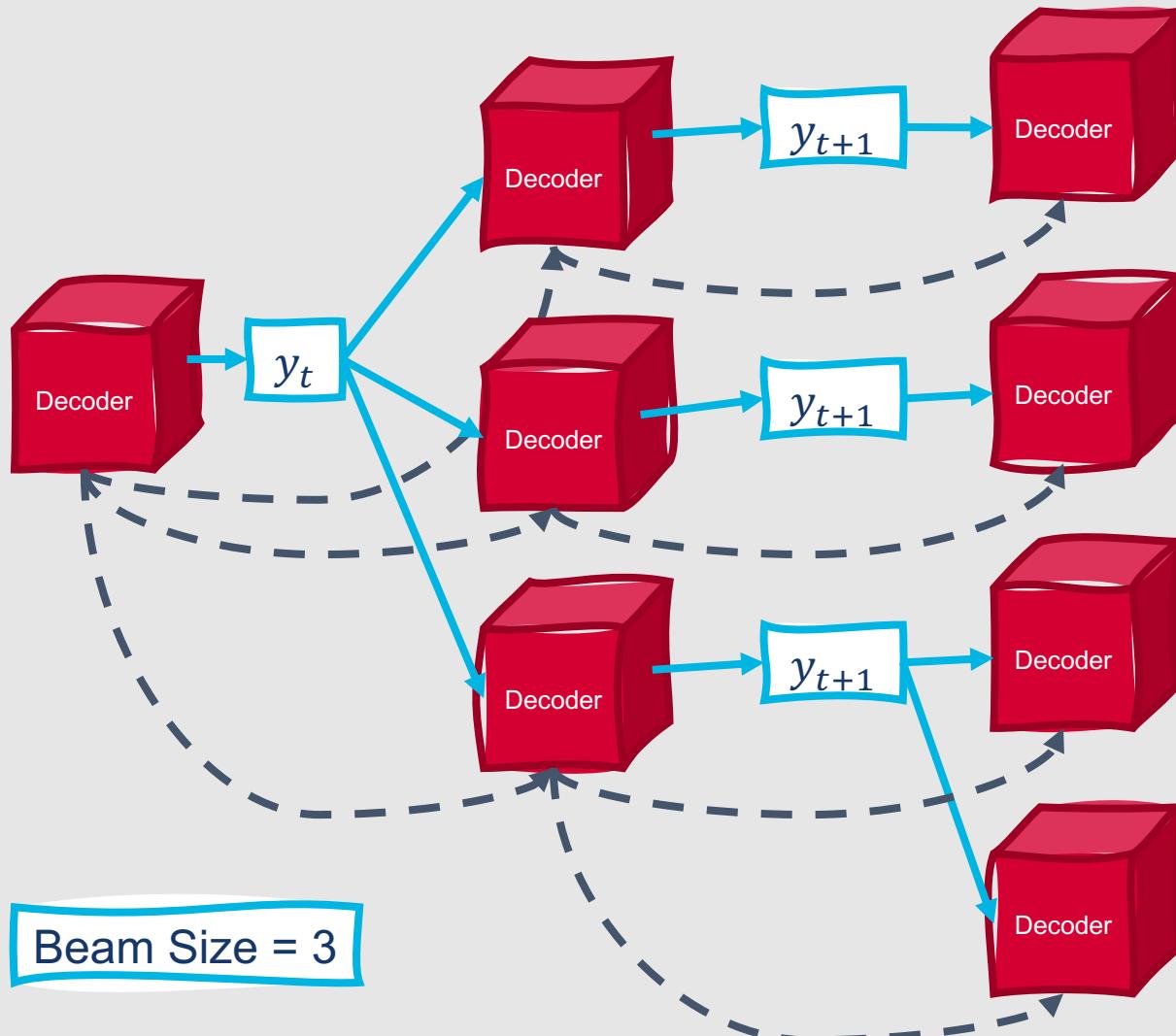
Beam Size = 3

# How does beam search work?

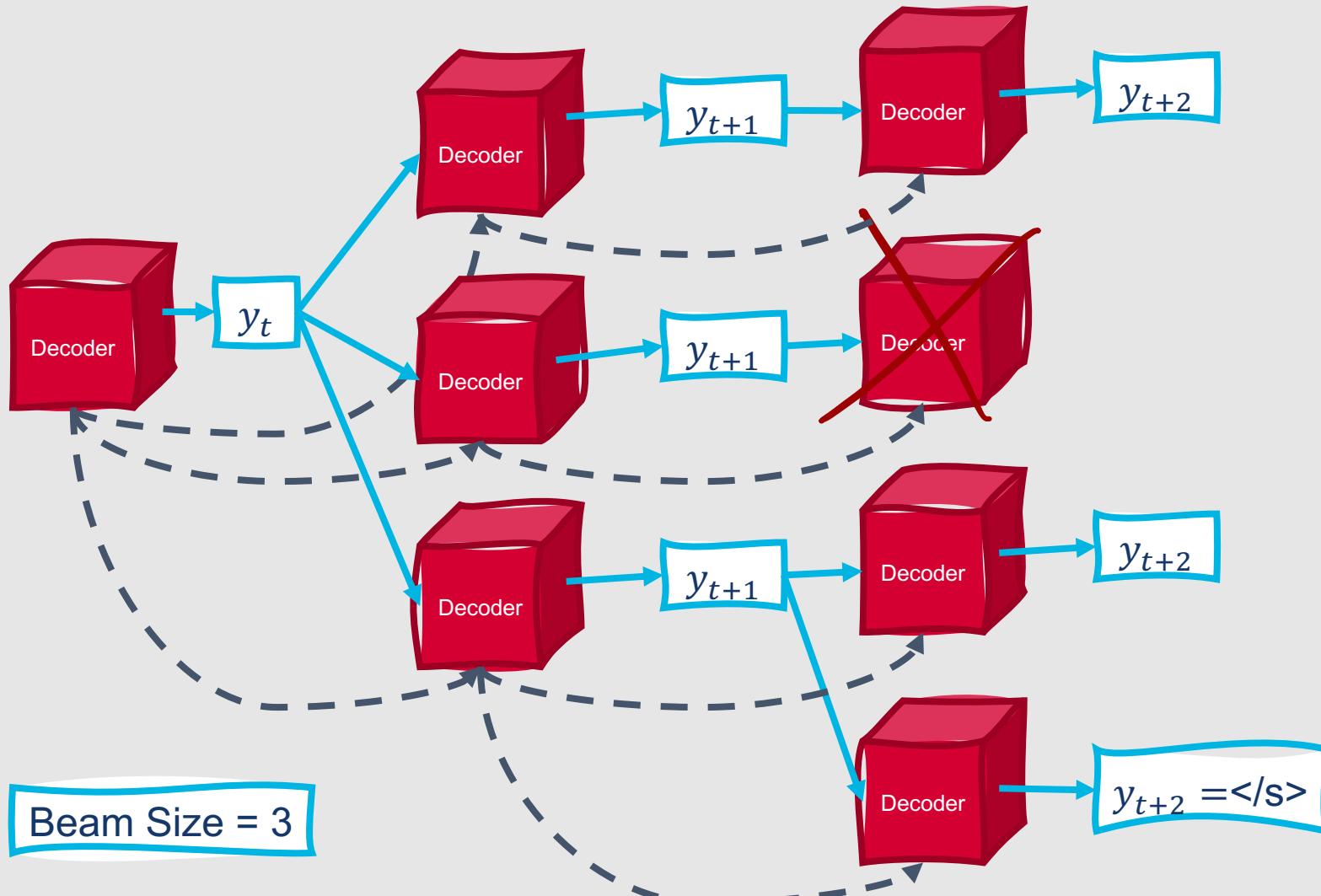


Beam Size = 3

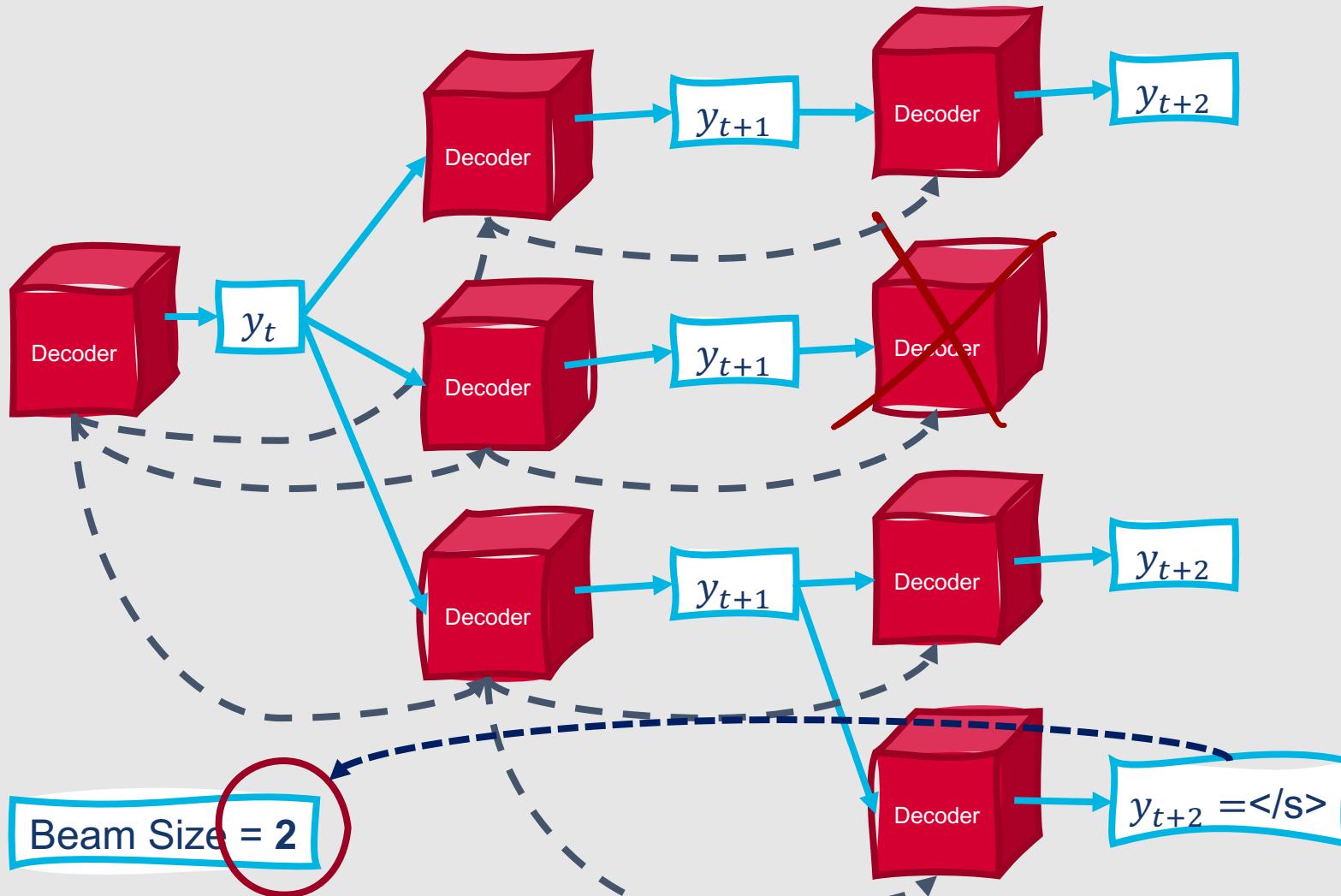
# How does beam search work?



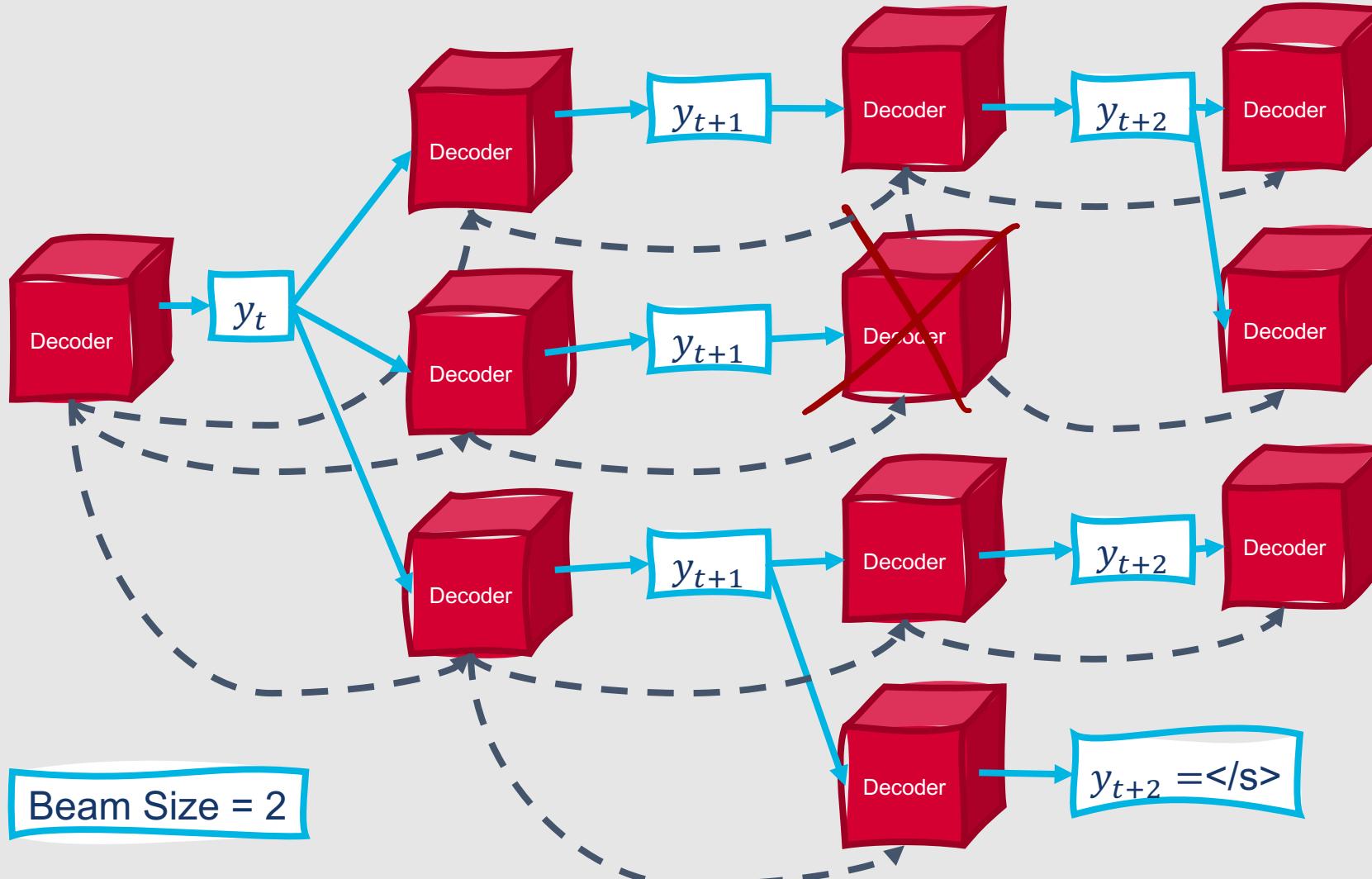
# How does beam search work?



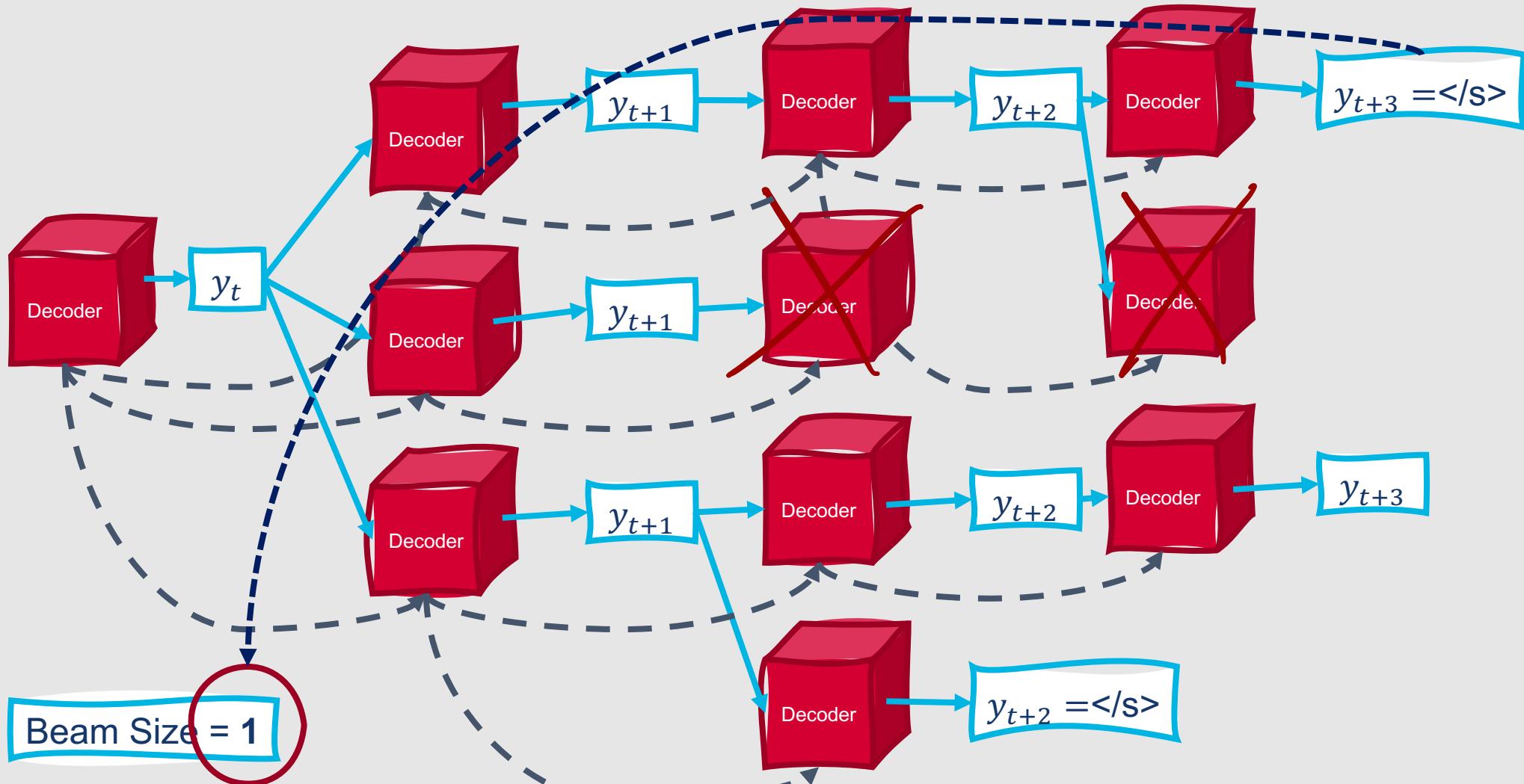
# How does beam search work?



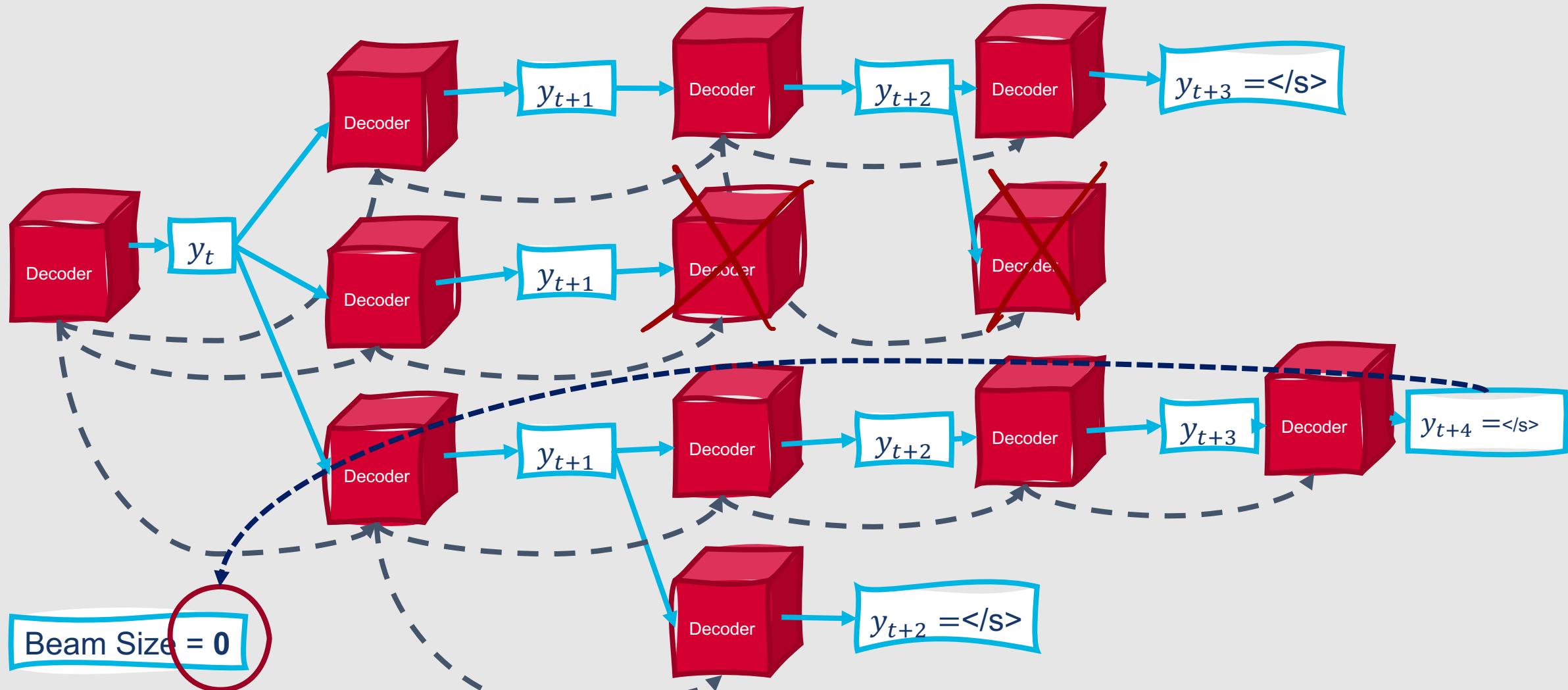
# How does beam search work?



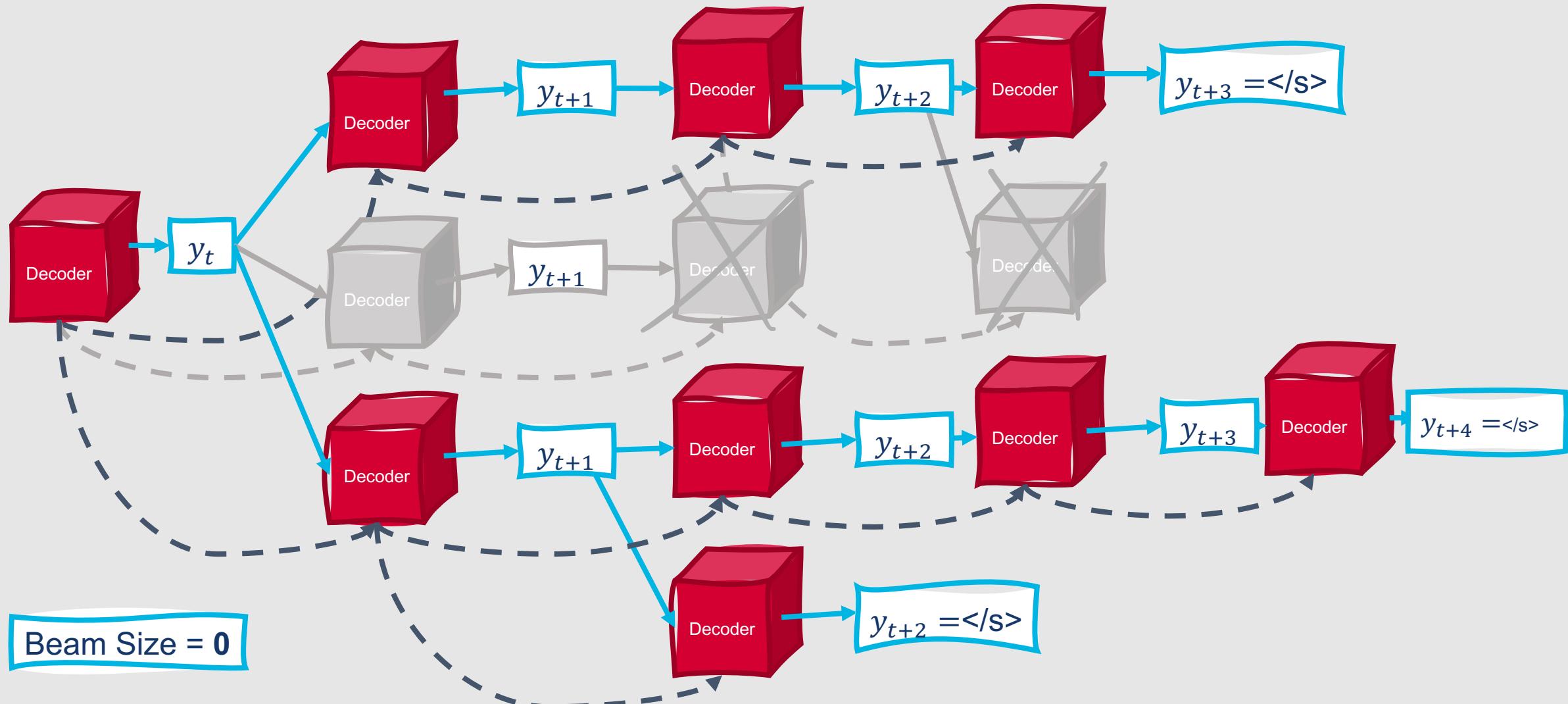
# How does beam search work?



# How does beam search work?



# How does beam search work?



# How do we choose a best hypothesis?

- Probabilistic scoring scheme
- Pass all or a subset of hypotheses to a downstream application

**So far, the encoder context vectors we've seen have been simple and static.**

- Can we do better?
  - Yes!



# Attention Mechanism

- Takes entire encoder context into account
- Can be embodied in a fixed-size vector

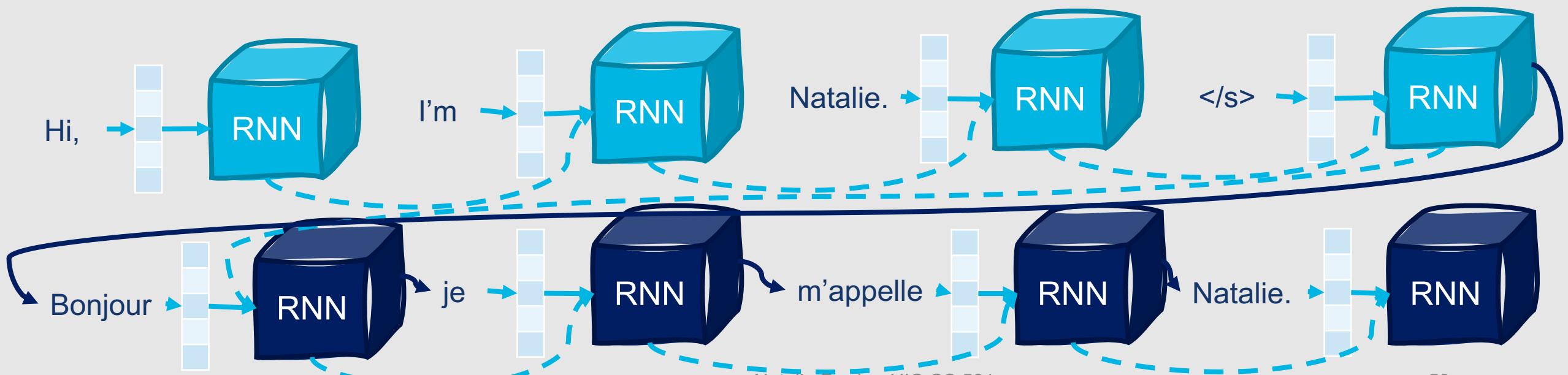
# Recall....

- We've already made our context vector available at each timestep when decoding
  - $h_t^d = g(\widehat{y_{t-1}}, h_{t-1}^d, c)$
- The first step in creating our attention mechanism is to update our hidden state such that it is conditioned on an updated context vector with each decoding step
  - $h_t^d = g(\widehat{y_{t-1}}, h_{t-1}^d, c_t)$

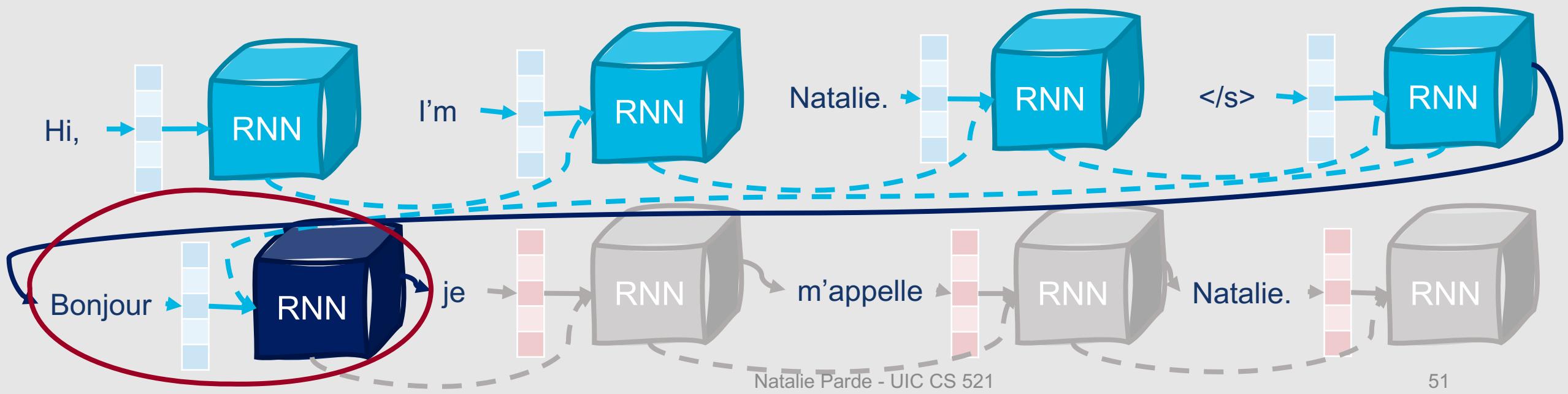
# How do we dynamically create a new context vector at each step?

- Compute a vector of scores that capture the relevance of each encoder hidden state to the decoder hidden state,  $h_{i-1}^d$ 
  - $score(h_{i-1}^d, h_j^e) = h_{i-1}^d \cdot h_j^e$

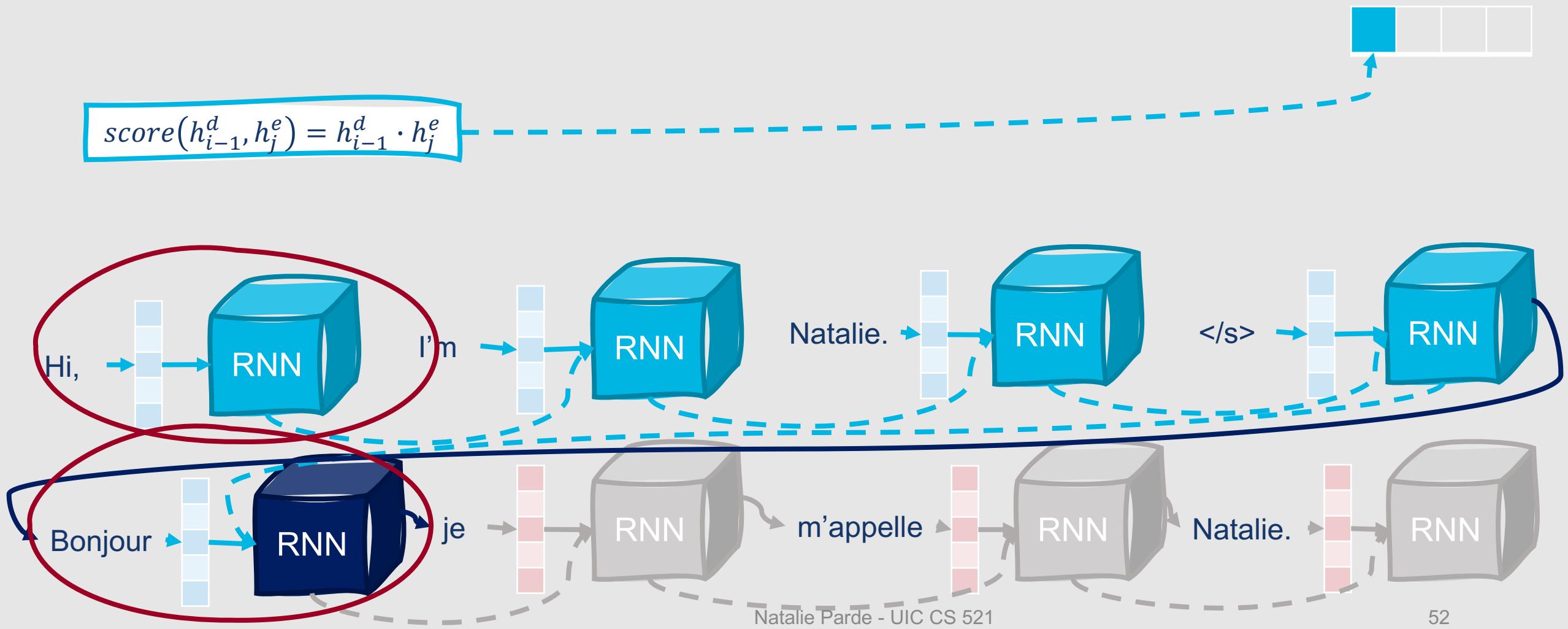
# Vector of Context Scores



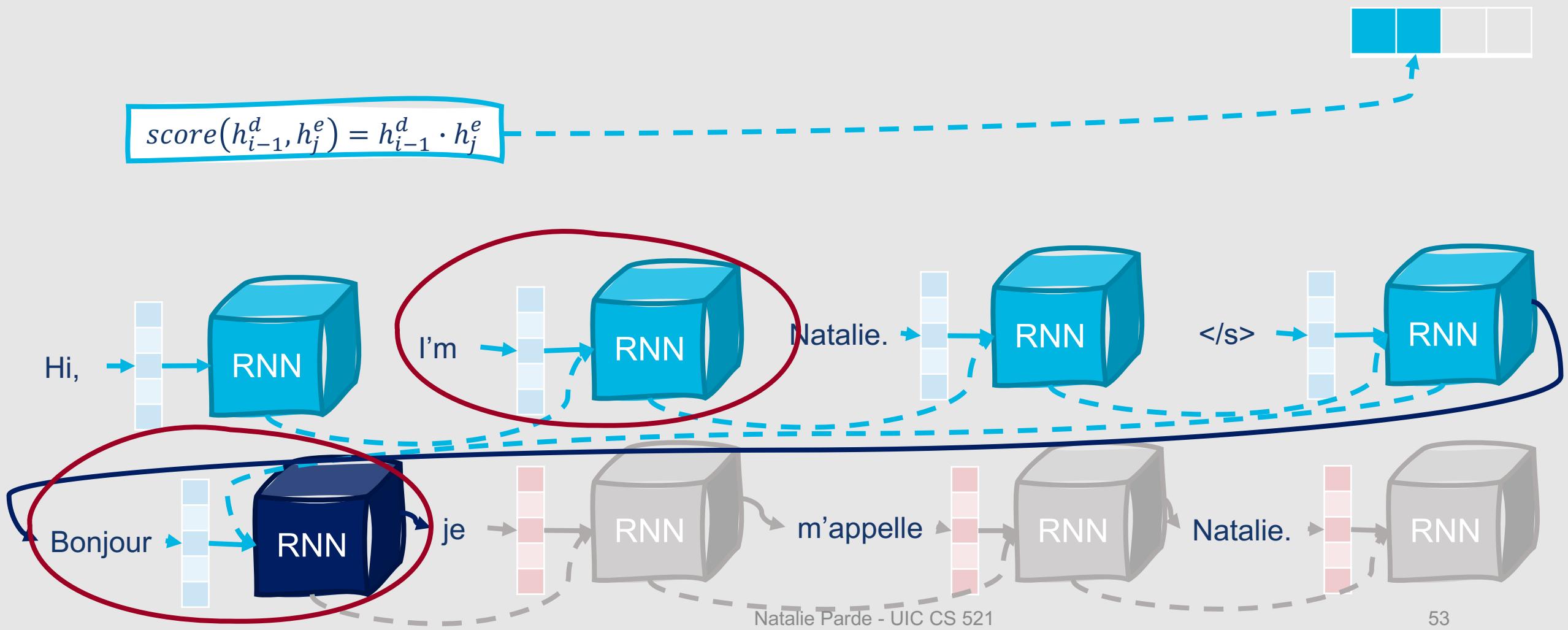
# Vector of Context Scores



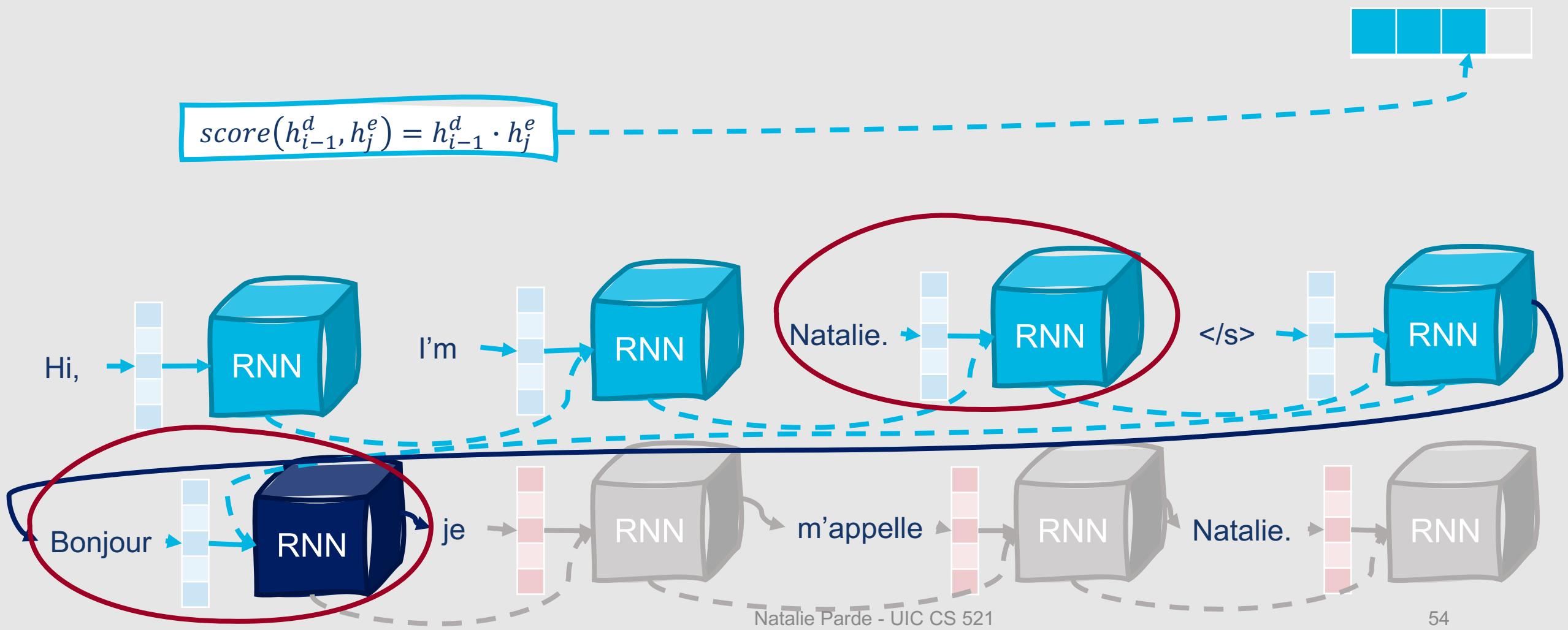
# Vector of Context Scores



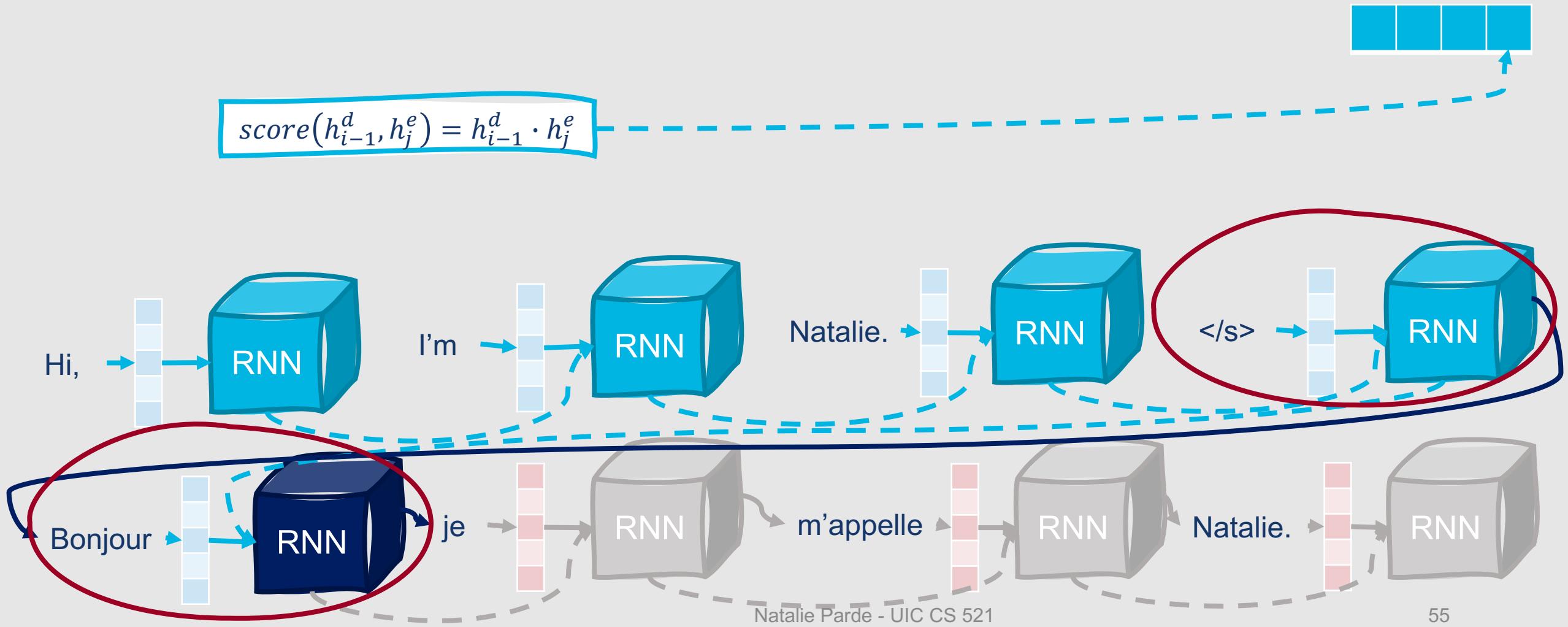
# Vector of Context Scores



# Vector of Context Scores



# Vector of Context Scores



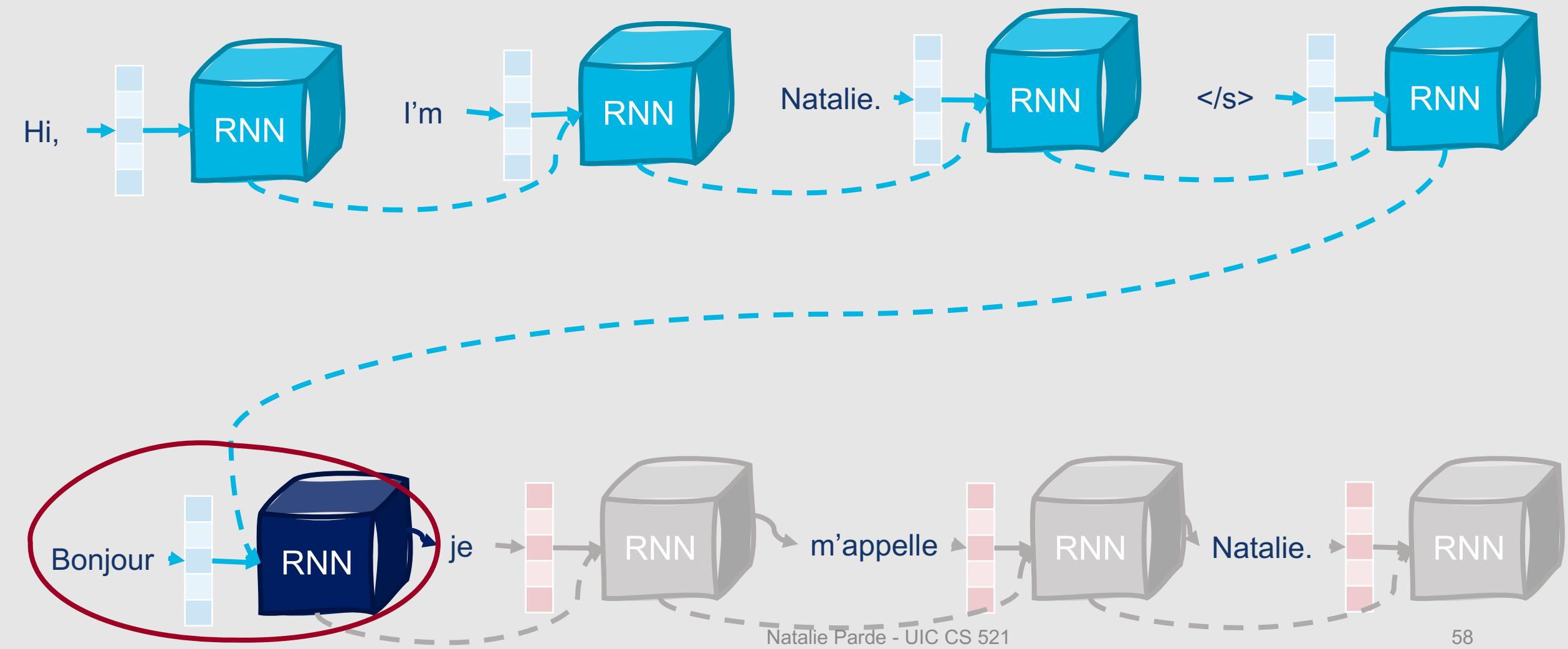
# How can we make use of context scores?

- Parameterize these scores with weights
- This allows the model to learn which aspects of similarity between the encoder and decoder states are important

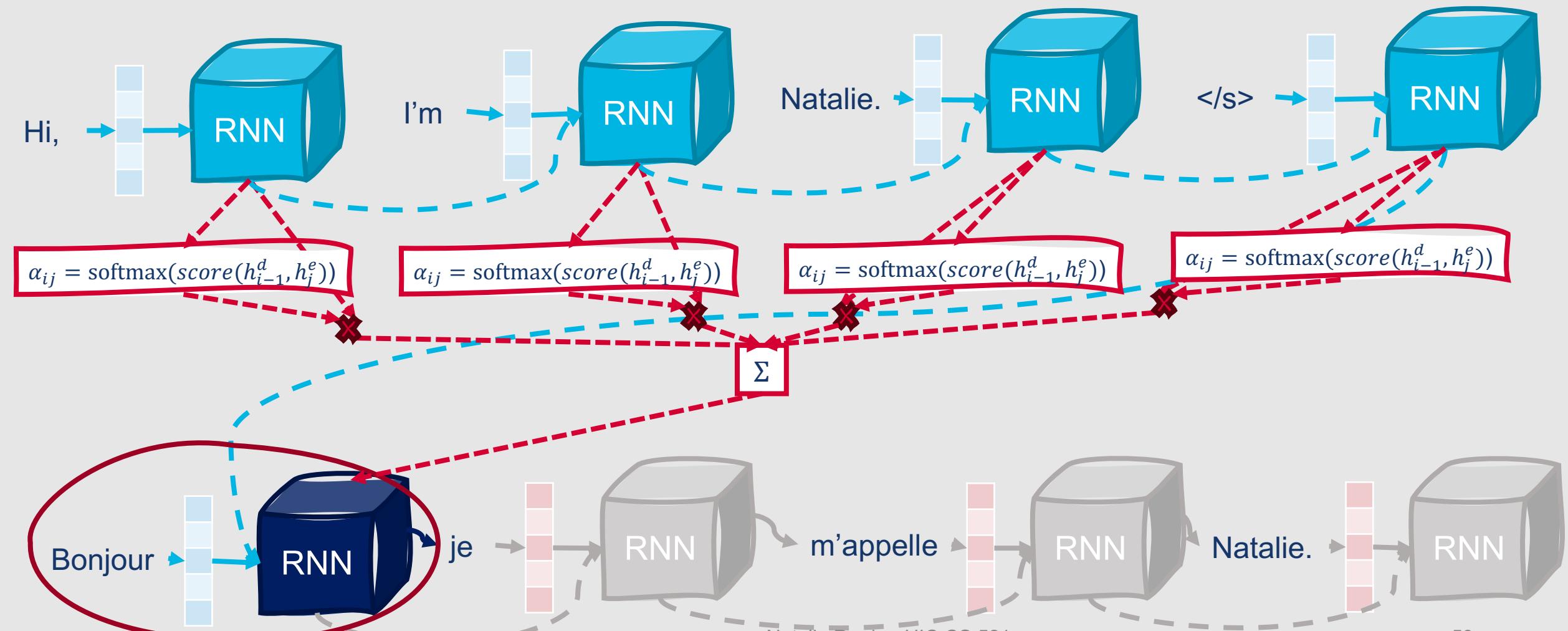
# Attention Weights

- **Normalize context scores** to create a vector of weights,  $\alpha_{ij}$ 
  - $\alpha_{ij} = \text{softmax(score}(h_{i-1}^d, h_j^e)\forall j \in e)$
  - Provides the proportional relevance of each encoder hidden state  $j$  to the current decoder state  $i$
- Finally, **take a weighted average over all the encoder hidden states** to create a fixed-length context vector for the current decoder state
  - $c_i = \sum_j \alpha_{ij} h_j^e$

# Thus, we finally have an encoder-decoder model with attention!



# Thus, we finally have an encoder-decoder model with attention!

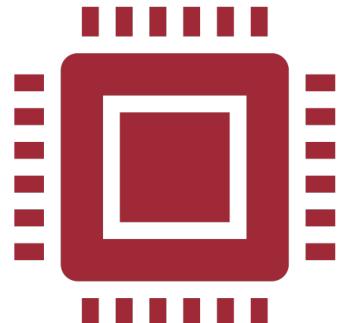


# Other Attention Weights

- More sophisticated scoring functions can be used as well
- Common: Parameterize the attention score with its own set of trainable weights
  - $\text{score}(\mathbf{h}_{i-1}^d, \mathbf{h}_j^e) = \mathbf{h}_{i-1}^d \mathbf{W}_s \mathbf{h}_j^e$
  - Advantage: Allows the encoder and decoder to use vectors with different dimensionality (dot-product attention requires the encoder and decoder hidden states to have the same dimensionality)

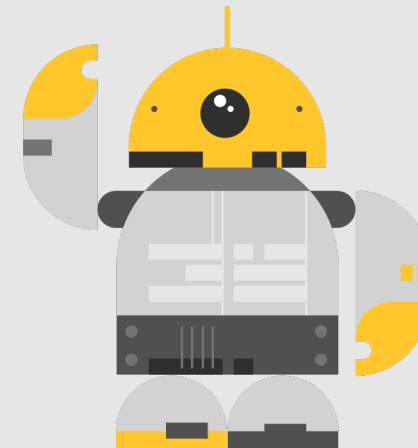
## Advanced RNNs are a powerful tool, but they are not without their limitations.

- Remaining challenges:
  - Even with sophisticated architectures, processing long-distance dependencies through **many recurrences** can eventually lead to loss of valuable information
  - Recurrent models cannot productively leverage **parallel resources**

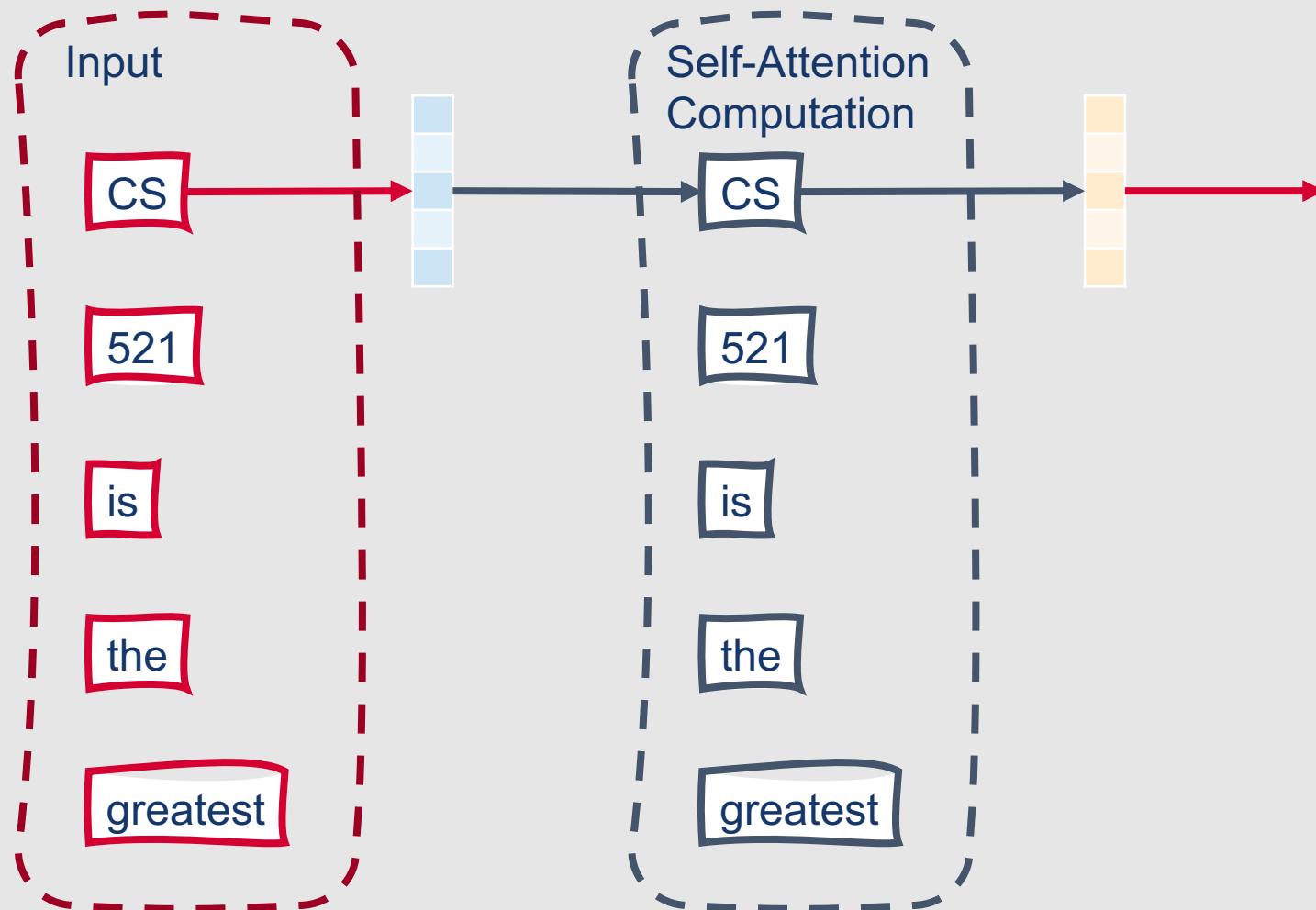


# Transformers

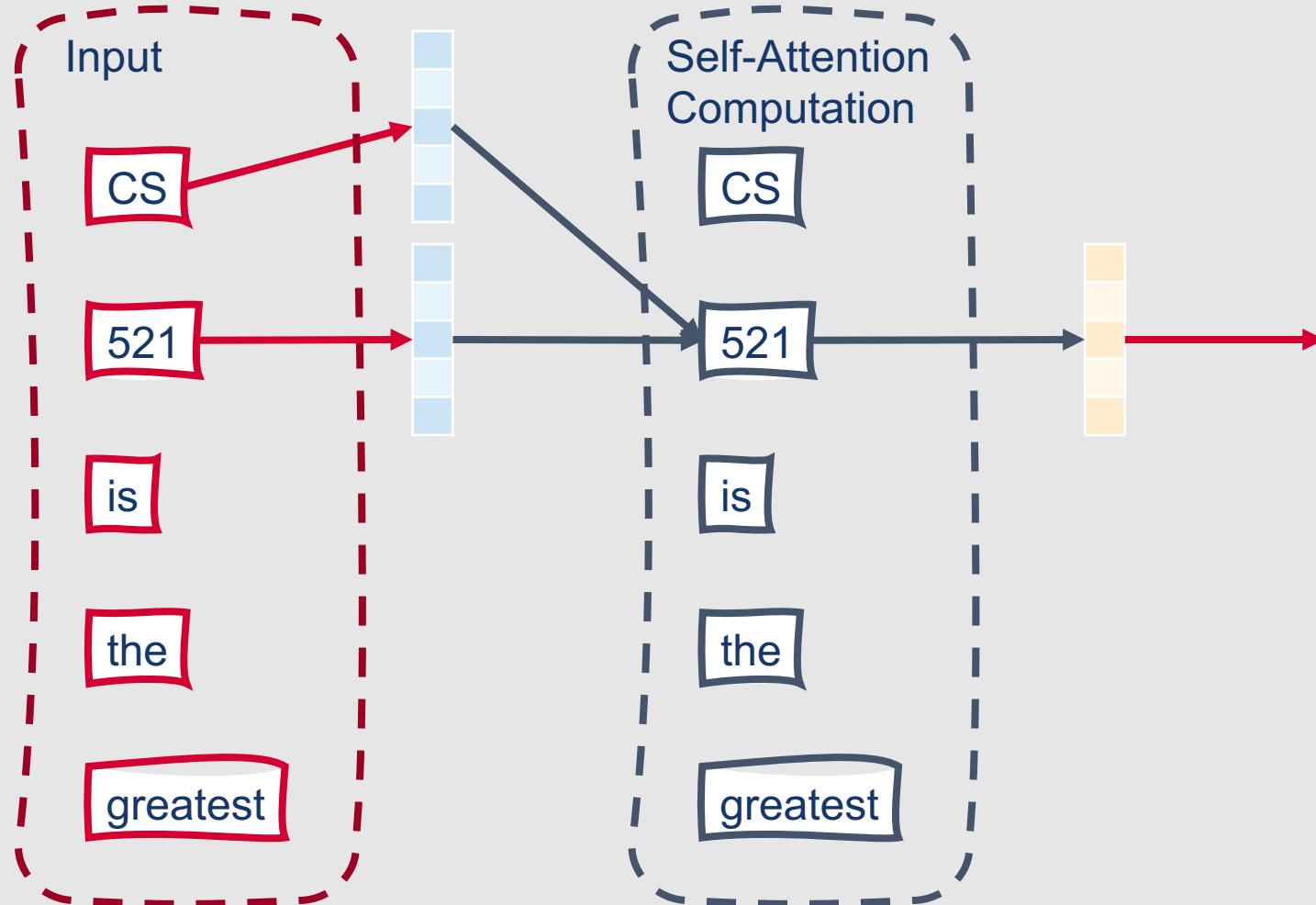
- Entirely do away with recurrences
- Stacks of:
  - Linear layers
  - Feedforward layers
  - **Self-attention** layers
    - For a given element in a sequence, determines which other element(s) up to that point are most relevant to it
      - Each computation is independent of other computations → easy parallelization
      - Each computation only considers elements up to that point in the sequence → easy language modeling
- Goal: Map sequences of input  $(x_1, \dots, x_n)$  to sequences of output  $(y_1, \dots, y_n)$  of the same length



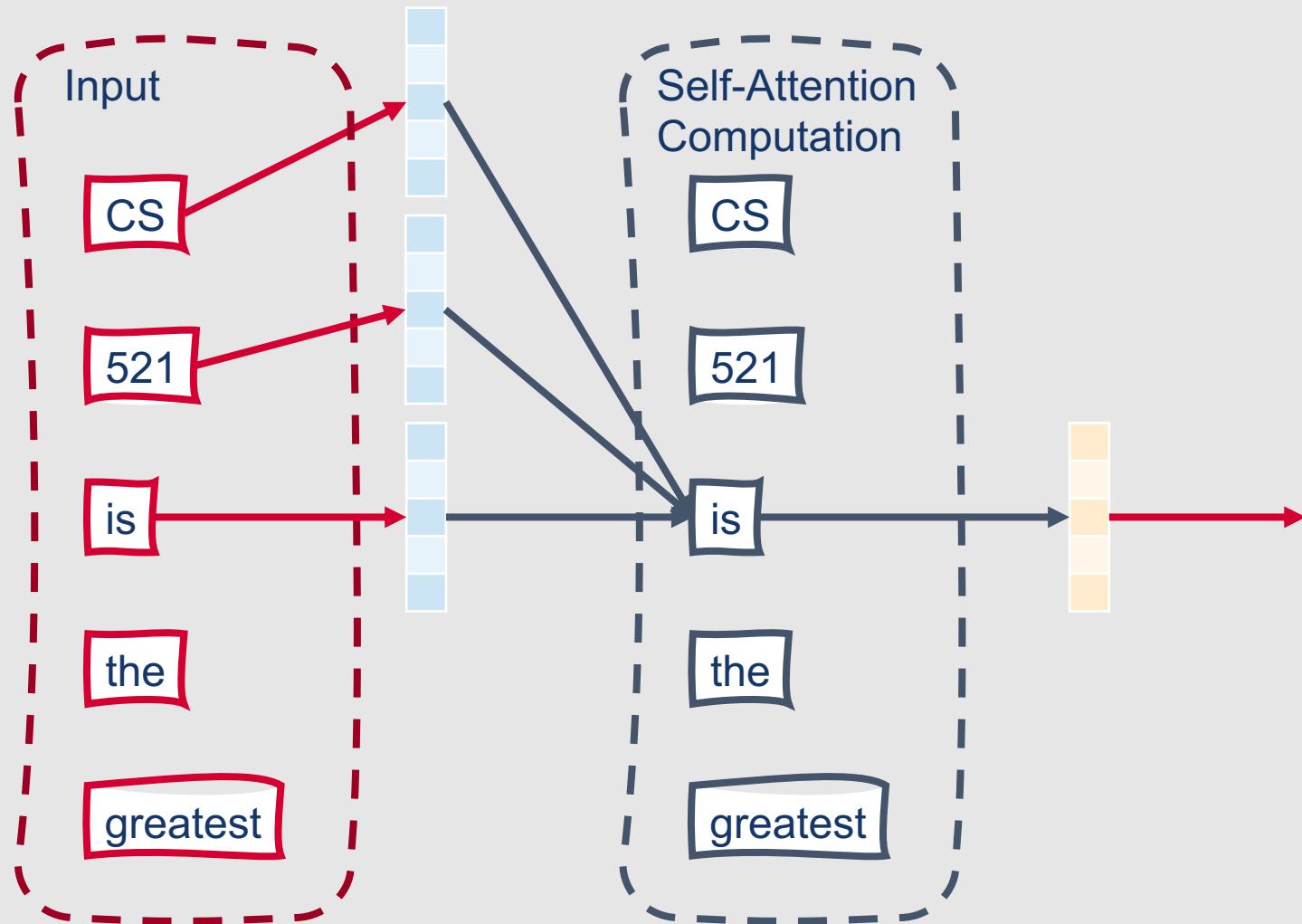
# Self-Attention



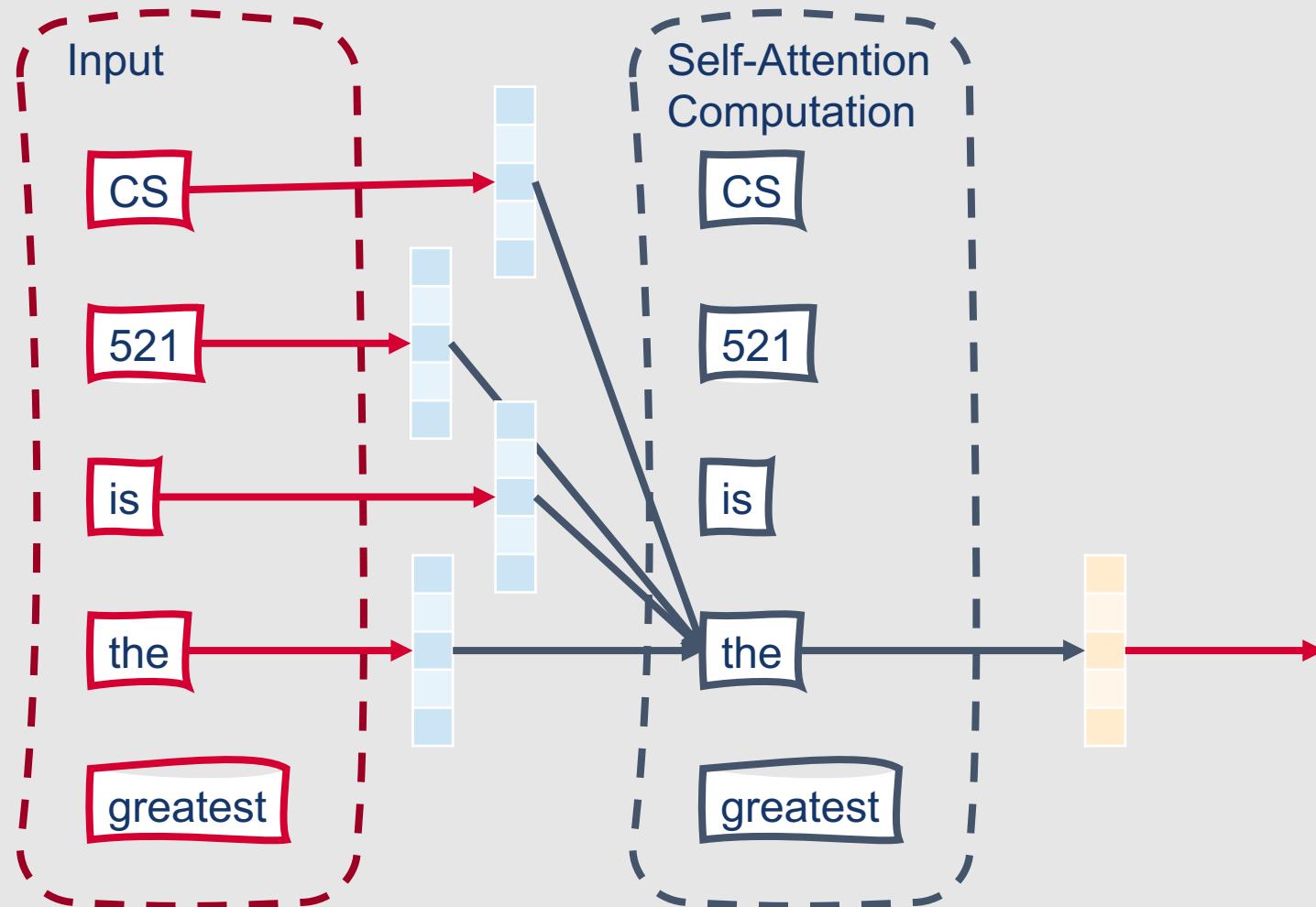
# Self-Attention



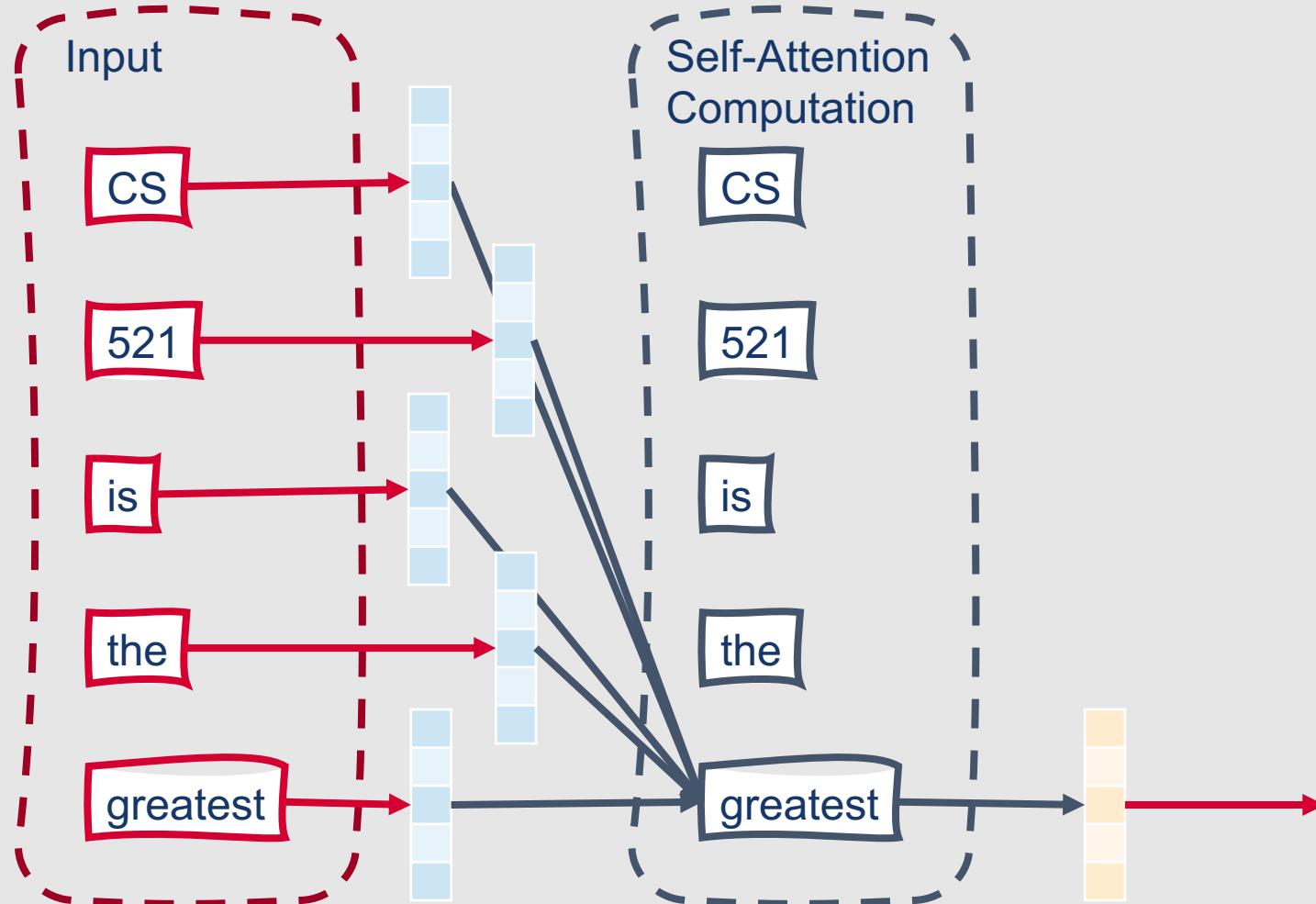
# Self-Attention



# Self-Attention



# Self-Attention



# Computing Self-Attention

- Simplest method:
  - Take the dot product between a given input element  $x_i$  and each input element  $(x_1, \dots, x_i)$  up until that point
    - $\text{score}(x_i, x_j) = x_i \cdot x_j$
  - Apply softmax normalization to create a vector of weights,  $\alpha_i$ , indicating proportional relevance of each sequence element to the current focus of attention,  $x_i$ 
    - $\alpha_{ij} = \text{softmax}(\text{score}(x_i, x_j)) \forall j \leq i = \frac{e^{\text{score}(x_i, x_j)}}{\sum_{k=1}^i e^{\text{score}(x_i, x_k)}} \forall j \leq i$
  - Take the sum of inputs thus far weighted by  $\alpha_i$  to produce an output  $y_i$ 
    - $y_i = \sum_{j \leq i} \alpha_{ij} x_j$

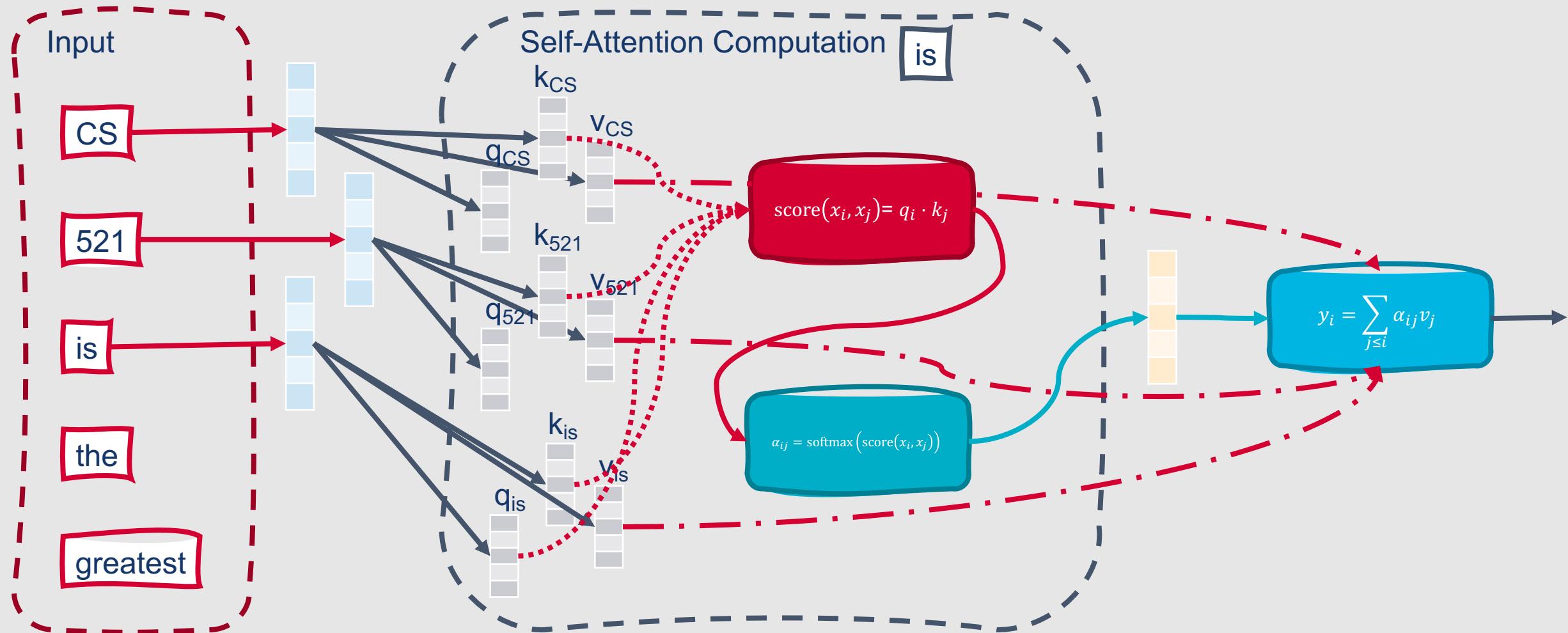
# How do Transformers learn?

- Continually updating weight matrices applied to inputs
- Weight matrices are learned for each of three roles when computing self-attention:
  - **Query:** The focus of attention when it is being compared to inputs up until that point,  $W^Q$
  - **Key:** An input that is being compared to the focus of attention,  $W^K$
  - **Value:** A value being used to compute the output for the current focus of attention,  $W^V$

# Training Transformers

- Weight matrices are applied to inputs in the context of their respective roles
  - $q_i = W^Q x_i$
  - $k_i = W^K x_i$
  - $v_i = W^V x_i$
- Then, we can update our equations for computing self-attention so that these roles are reflected in them:
  - $\text{score}(x_i, x_j) = q_i \cdot k_j$
  - $\alpha_{ij} = \text{softmax}(\text{score}(x_i, x_j)) \forall j \leq i$
  - $y_i = \sum_{j \leq i} \alpha_{ij} v_j$

# Self-Attention

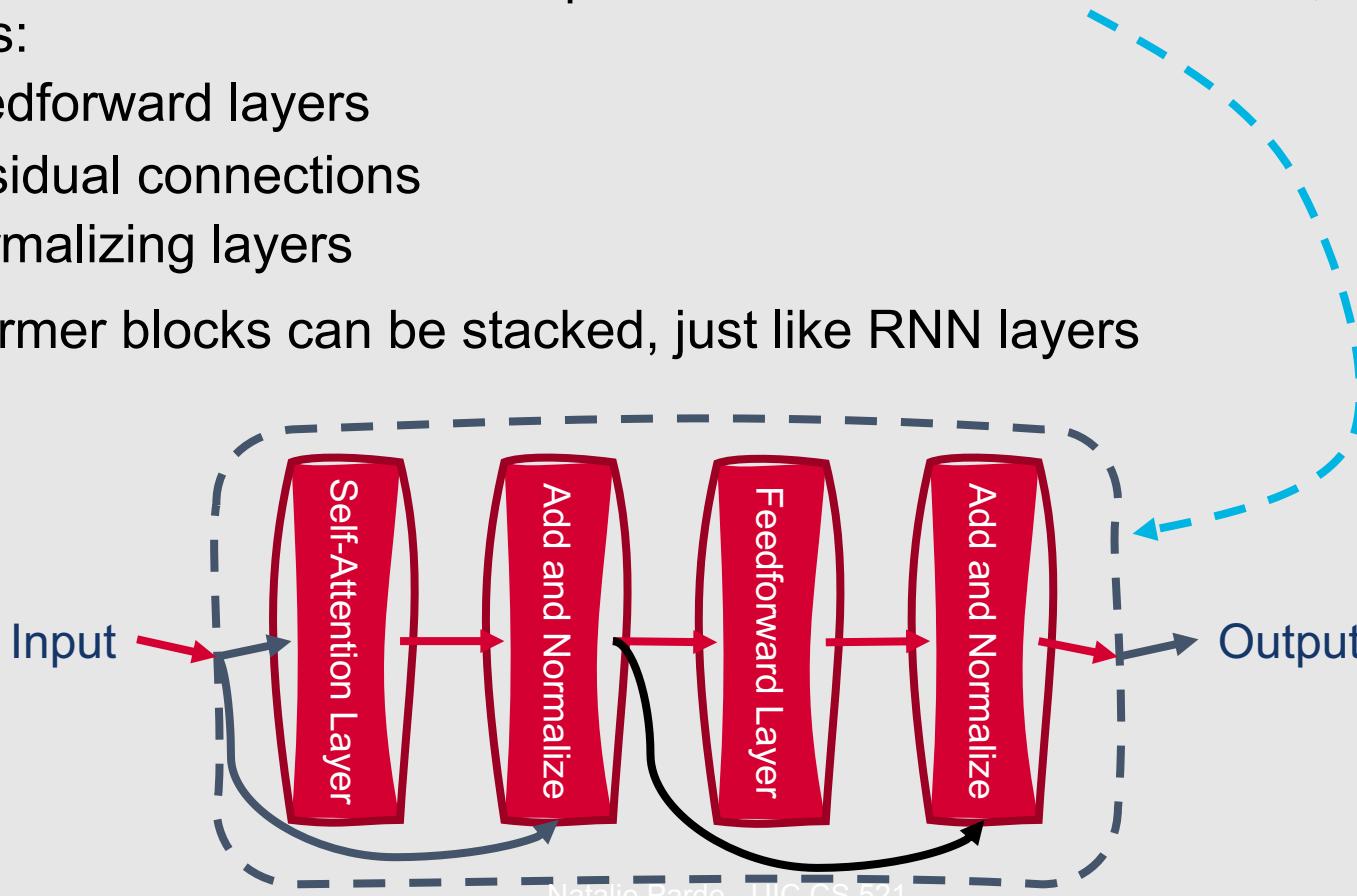


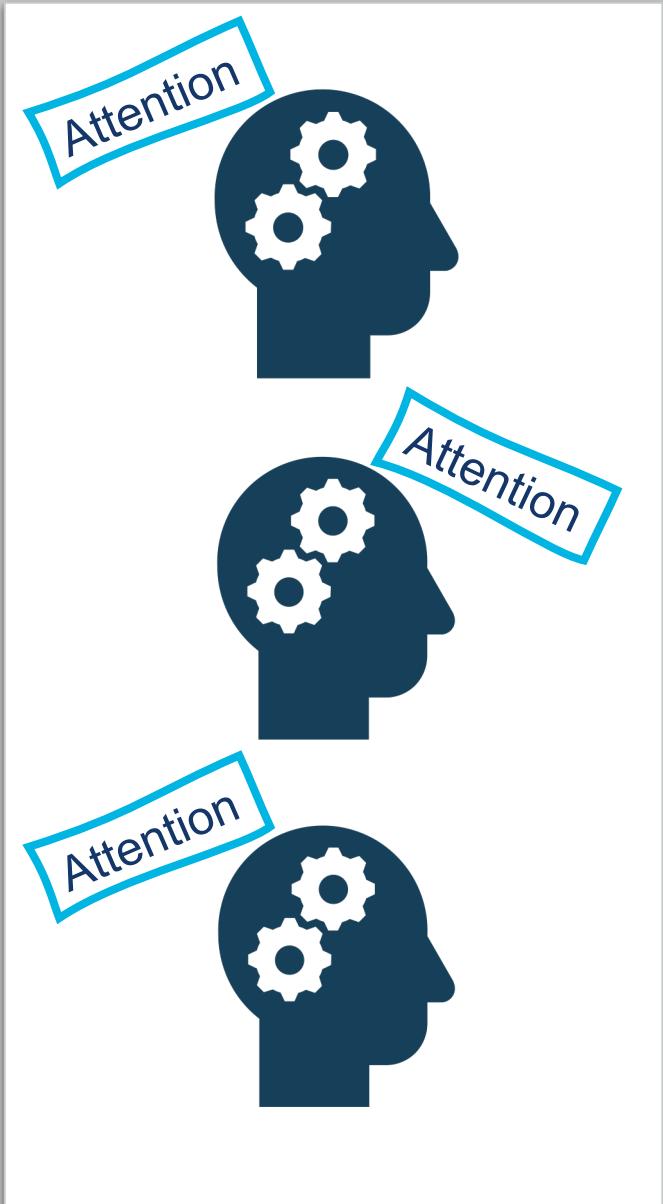
# Practical Considerations

- Combining a dot product with an exponential (as in softmax) may lead to arbitrarily large values
- It is common to scale the scoring function based on the dimensionality of the key (and query) vectors,  $d_k$ 
  - $\text{score}(x_i, x_j) = \frac{q_i \cdot k_j}{\sqrt{d_k}}$
- Each  $y_i$  is computed independently, so we can parallelize computations using efficient matrix multiplication routines where  $X$  is a matrix containing all input embeddings
  - $Q = W^Q X$
  - $K = W^K X$
  - $V = W^V X$
  - $\text{SelfAttention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$ 
    - Make sure to avoid including knowledge of future words in language modeling settings!

# Transformer Blocks

- Self-attention is the central component of a **Transformer block**, which also includes:
  - Feedforward layers
  - Residual connections
  - Normalizing layers
- Transformer blocks can be stacked, just like RNN layers





# Multihead Attention

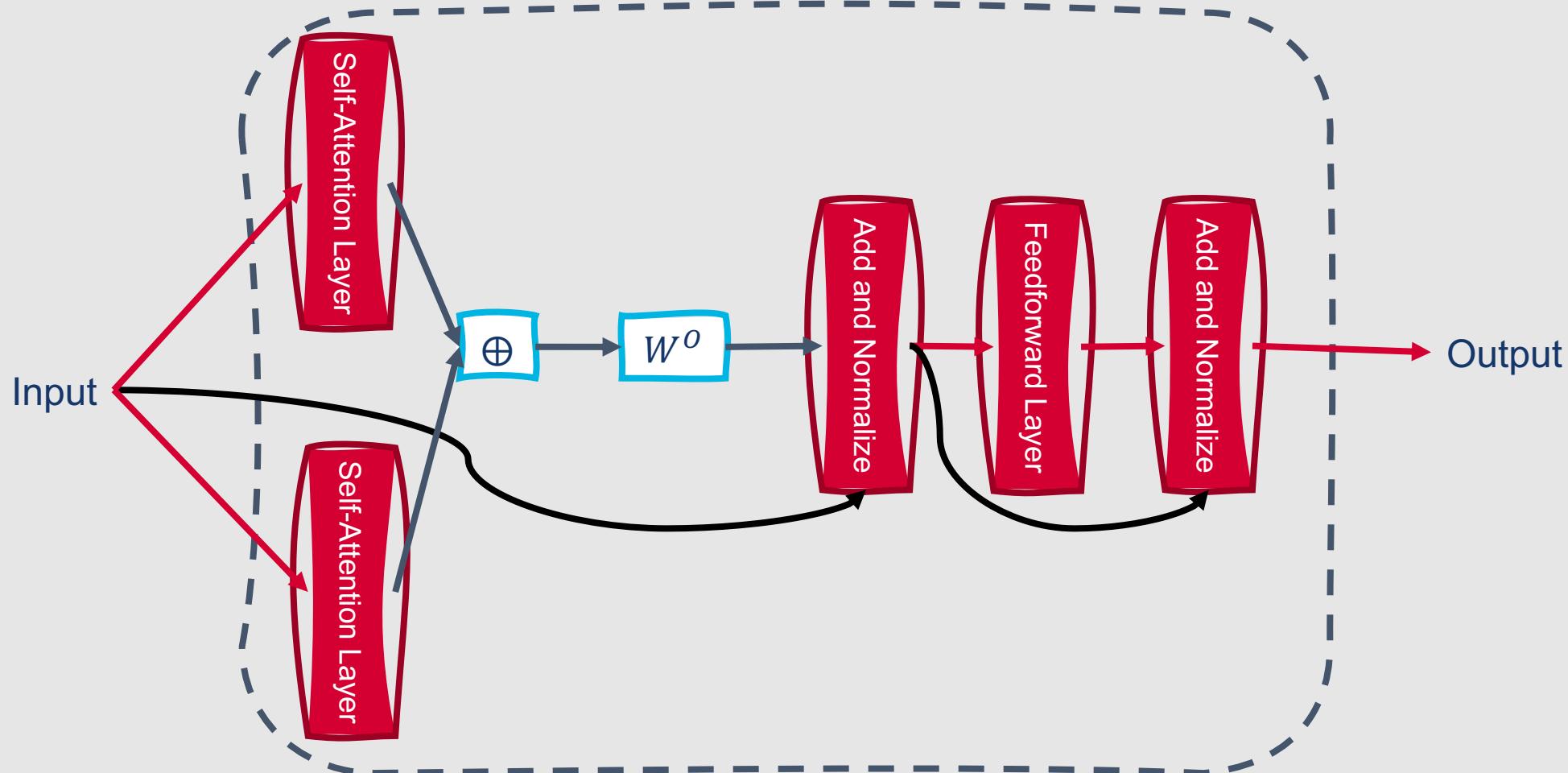
- Each self-attention layer represents a single **attention head**
- **Multihead attention** places multiple attention heads in parallel in the Transformer model
  - Since each attention head has its own set of weights, each one can learn different aspects of the relations between input elements at the same level of abstraction

# Computing Multihead Attention

---

- Each head in the self-attention layer is parameterized with its own weights
  - $Q = W_i^Q X$
  - $K = W_i^K X$
  - $V = W_i^V X$
- The output of a multihead attention layer with  $n$  heads comprises  $n$  vectors of equal length
- These heads are concatenated and then reduced to the original input/output dimensionality
  - $\text{head}_i = \text{SelfAttention}(W_i^Q X, W_i^K X, W_i^V X)$
  - $\text{MultiheadAttention}(Q, K, V) = W^O(\text{head}_1 \oplus \text{head}_2 \oplus \dots \oplus \text{head}_n)$

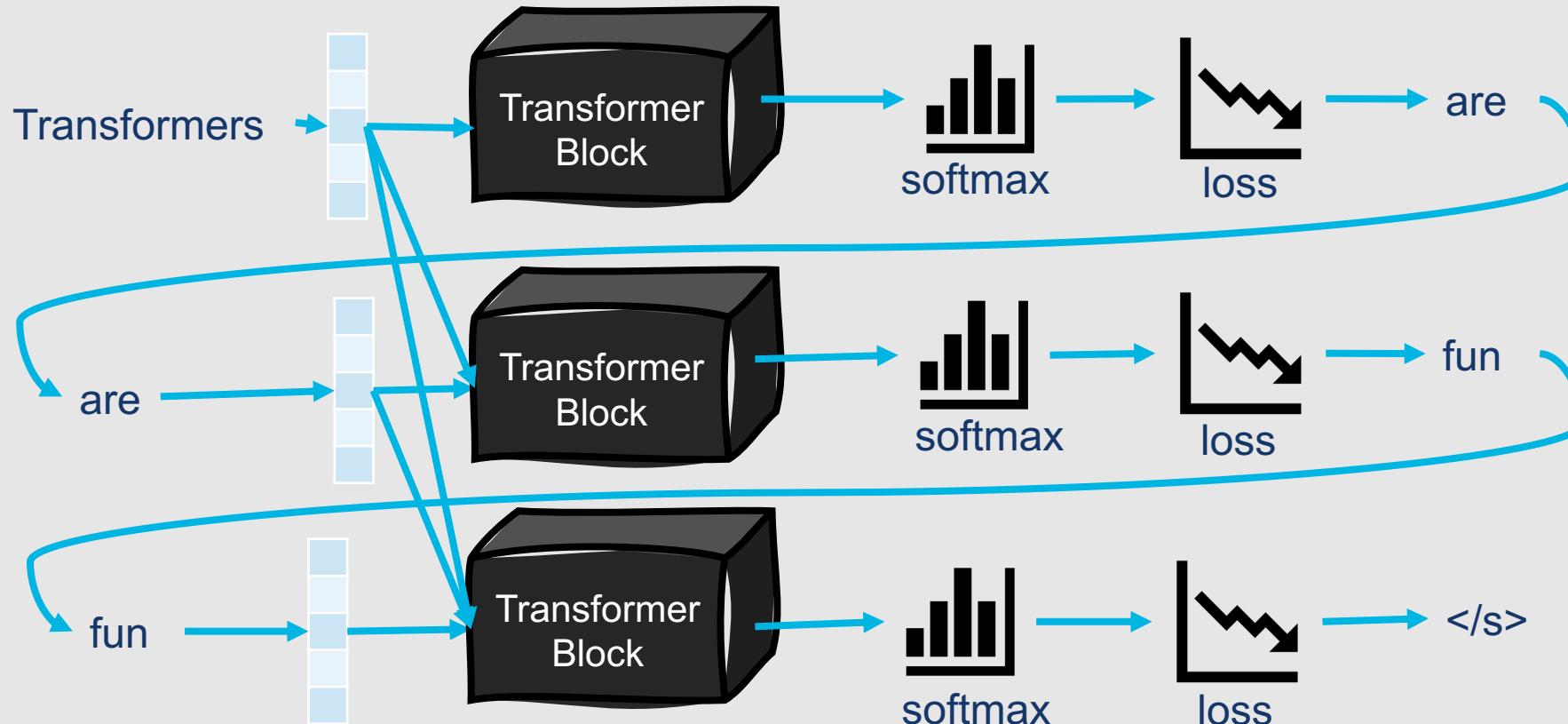
# Multihead Attention



# Positional Embeddings

- Since Transformers don't make use of recurrent connections, they instead employ separate **positional embeddings** to encode positionality
  - Randomly initialize an embedding for each input position
  - Update weights during the training process
  - Input embedding with positional information = word embedding + positional embedding
- Static functions mapping positions to vectors can be used as an alternative

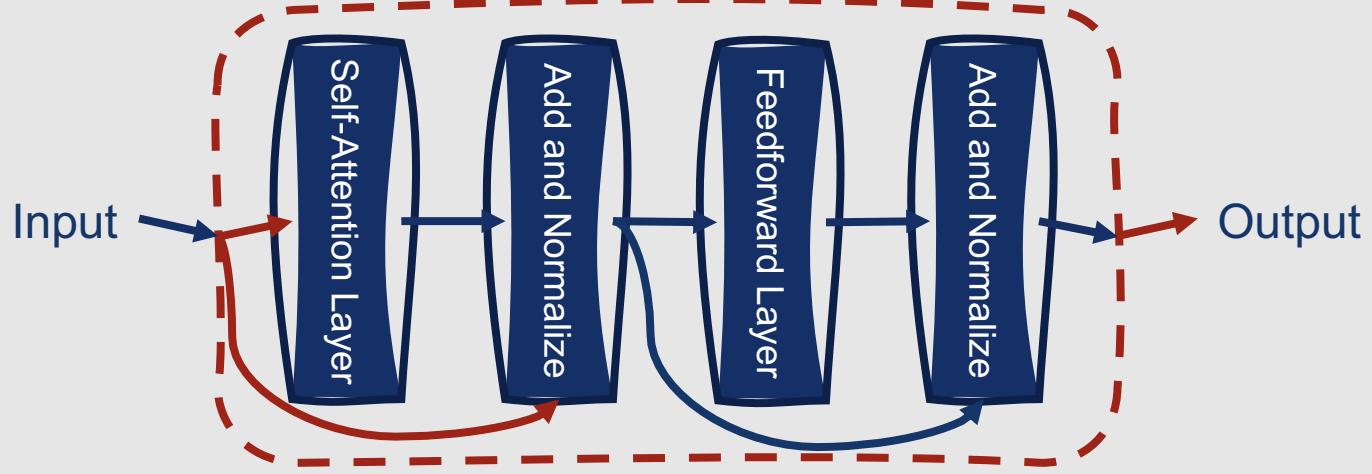
# Transformers as Autoregressive Language Models



# Encoder- Decoder Models with Transformers

- Similar to other encoder-decoder models
  - Encoder (Transformer model) maps sequential input to an output representation
  - Decoder (Transformer model) attends to the encoder representation and generates sequential output autoregressively
- However....
  - Transformer blocks in the decoder include an extra **cross-attention** layer

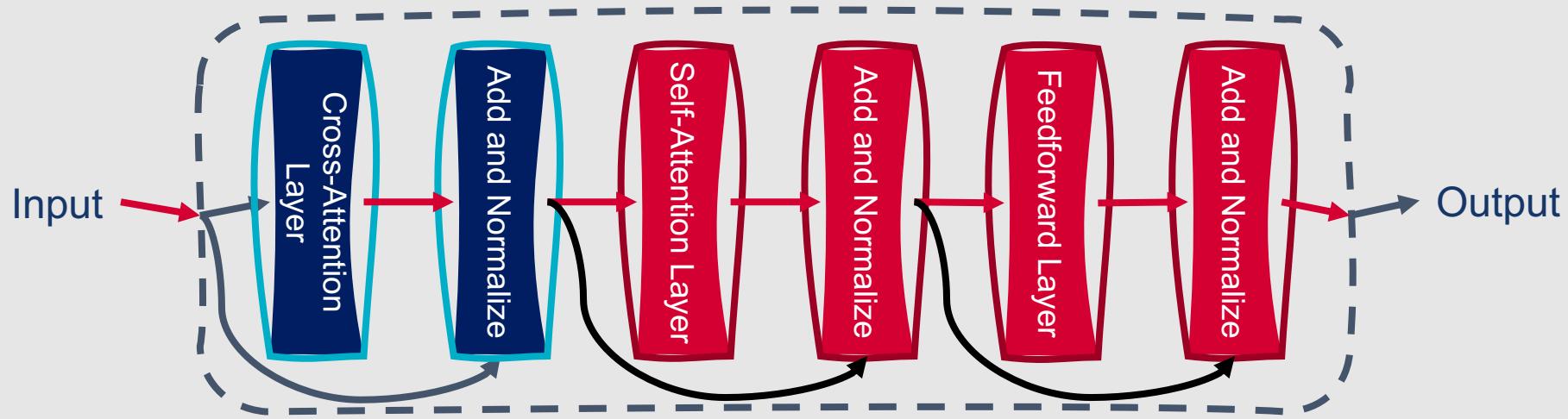
Reminder: Normal Transformer block



- Same form as **multiheaded self-attention** in a normal Transformer block, with one difference: queries come from the previous layer of the decoder as usual, but **keys and values come from the output of the encoder**
  - $\mathbf{Q} = \mathbf{W}^{\mathbf{Q}} \mathbf{H}^{dec[i-1]}$
  - $\mathbf{K} = \mathbf{W}^{\mathbf{Q}} \mathbf{H}^{enc}$
  - $\mathbf{V} = \mathbf{W}^{\mathbf{V}} \mathbf{H}^{enc}$
- $CrossAttention(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V}$

# Cross-Attention

# Updated Decoder Transformer Block



# Encoder-Decoder Models with Transformers

- Why is cross-attention useful?
  - Allows the decoder to attend to the entire encoder sequence
- Training Transformer-based encoder-decoders is similar to training RNN-based encoder-decoders
  - Use teacher forcing
  - Train autoregressively

# Practical Details for Building MT Systems

## Vocabulary

- MT systems typically use a fixed vocabulary generated using byte pair encoding or other wordpiece algorithms
- Vocabulary is usually shared across the source and target languages

## Corpora

- Parallel corpora with the same content communicated in multiple languages
- Common sources:
  - Government documents for nations with multiple official languages
  - Subtitles for movies and TV shows
- Often, text from the source and target language(s) is aligned at the sentence level

# What if we don't have much training data?

- Parallel corpora are difficult to find, especially for lower-resource language pairs
- **Backtranslation:**
  1. Train an intermediate target-to-source MT system on a small parallel corpus
  2. Translate additional monolingual data from the target language to the source language using this intermediate system
  3. Consider this new, synthetic parallel data as additional training data
  4. Train a source-to-target MT system on the expanded training dataset

# Summary: Machine Translation Methods and Encoder- Decoder Models

- Machine translation is challenging due to many **typological**, **morphological**, and other differences between languages
- Classical machine translation used **dictionary-based**, **direct transfer**, and **interlingua** approaches
- A popular statistical MT model is the **Bayesian noisy channel approach**, which relies on **phrase-based translation**
- **Encoder-decoder models** draw upon similar techniques for autoregressive language modeling to convert input to an intermediate vector representation and then convert that intermediate representation to output
- One newer architecture that can be used in encoder-decoder settings is the **Transformer** model

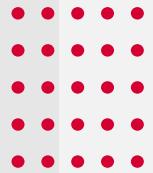
# How do we evaluate machine translation models?

- Translation quality tends to be very subjective!
- Two common approaches:
  - **Human ratings**
  - **Automated metrics**



# Evaluating Machine Translation Using Human Ratings

- Typically evaluated along multiple dimensions
- Tend to check for both **fluency** and **adequacy**
- **Fluency:**
  - Clarity
  - Naturalness
  - Style
- **Adequacy:**
  - Fidelity
  - Informativeness



# Evaluating Machine Translation Using Human Ratings

- How to get quantitative measures of fluency?
  - Ask humans to rate different aspects of fluency along a scale
  - Measure how long it takes humans to read a segment of text
  - Ask humans to guess the identity of the missing word
    - “After such a late night working on my project, it was hard to wake up this \_\_\_\_\_!”



# Evaluating Machine Translation Using Human Ratings

- How to get quantitative measures of adequacy?
  - Ask bilingual raters to rate how much information was preserved in the translation
  - Ask monolingual raters to do the same, given access to a gold standard reference translation
  - Ask raters to answer multiple-choice questions about content present in a translation

# Another set of human evaluation metrics considers post- editing.

- Ask a human to **post-edit** or “fix” a translation
- Compute the number of edits required to correct the output to an acceptable level
  - Can be measured via number of word changes, number of keystrokes, amount of time taken, etc.

# Automated Metrics

- Less accurate than human evaluation, but:
  - Useful for iteratively testing system improvements
  - Can be used as an automatic loss function
- Two main families:
  - Character- or word-overlap
  - Embedding similarity

# Popular Lexical Overlap Metrics

- **BLEU**
  - Measure of word overlap
- **METEOR**
  - Measure of word overlap, considering stemming and synonymy
- **Character F-Score (chrF)**
  - Measure of character n-gram overlap



# BLEU

- Weighted average of the number of n-gram overlaps with human translations
- **Precision-based metric**
  - What percentage of words in the candidate translation also occur in the gold standard translation(s)?
- To compute BLEU:
  - Count how many times each n-gram is used in the candidate translation,  $c(ngram)$
  - Clip that amount so that the highest it can be is  $c_{max}(ngram)$ , defined as the maximum number of times it is used in a reference translation
  - Compute precision for each word in the candidate translation:
    - $$\text{prec}_n = \frac{\sum_{c \in \{\text{Candidates}\}} \sum_{ngram \in c} \min(c(ngram), c_{max}(ngram))}{\sum_{c \in \{\text{Candidates}\}} \sum_{ngram \in c} c(ngram)}$$
  - Take the geometric mean of the modified n-gram precisions for unigrams, bigrams, trigrams, and 4-grams

# Then, add a penalty for translation brevity....

- Otherwise, extremely short translations (e.g., “the”) could receive perfect scores!
- The penalty is based on:
  - The sum of the lengths of the reference sentences,  $r$
  - The sum of the lengths of the candidate translations,  $c$
- Formally, the penalty is set to:
  - $$BP = \begin{cases} 1 & \text{if } c > r \\ e^{(1-\frac{r}{c})} & \text{if } c \leq r \end{cases}$$
- The full BLEU score for a set of translations is then:
  - $$BLEU = BP * (\prod_{n=1}^4 \text{prec}_n)^{\frac{1}{4}}$$

# Example: Computing BLEU

Mina no dió una bofetada a la bruja verde.

Source Sentence

Mina didn't slap the green witch.

Reference Translation

Mina did not give a slap to the green witch.

Candidate Translation

# Example: Computing BLEU

Mina no dió una bofetada a la bruja verde.

Source Sentence

Mina didn't slap the green witch.

Reference Translation

Mina did not give a slap to the green witch.

Candidate Translation

$$\text{prec}_n = \frac{\sum_{c \in \{\text{Candidates}\}} \sum_{\text{ngram} \in c} \min(c(\text{ngram}), c_{\max}(\text{ngram}))}{\sum_{c \in \{\text{Candidates}\}} \sum_{\text{ngram} \in c} c(\text{ngram})}$$

$$BP = \begin{cases} 1 & \text{if } c > r \\ e^{(1-\frac{r}{c})} & \text{if } c \leq r \end{cases}$$

$$BLEU = BP * \left( \prod_{n=1}^4 \text{prec}_n \right)^{\frac{1}{4}}$$

# Example: Computing BLEU

Mina didn't slap the green witch.

Mina did not give a slap to the green witch.

$$\text{prec}_n = \frac{\sum_{c \in \{\text{Candidates}\}} \sum_{\text{ngram} \in c} \min(c(\text{ngram}), c_{\max}(\text{ngram}))}{\sum_{c \in \{\text{Candidates}\}} \sum_{\text{ngram} \in c} c(\text{ngram})}$$

Unigram	Unigram Frequency (Candidate)	Unigram Frequency (Reference)
Mina	1	1
did	1	0
not	1	0
give	1	0
a	1	0
slap	1	1
to	1	0
the	1	1
green	1	1
witch	1	1
.	1	1

$$BP = \begin{cases} 1 & \text{if } c > r \\ e^{(1-\frac{r}{c})} & \text{if } c \leq r \end{cases}$$

$$BLEU = BP * \left( \prod_{n=1}^4 \text{prec}_n \right)^{\frac{1}{4}}$$

# Example: Computing BLEU

Mina didn't slap the green witch.

Mina did not give a slap to the green witch.

$$\text{prec}_n = \frac{\sum_{c \in \{\text{Candidates}\}} \sum_{\text{ngram} \in c} \min(c(\text{ngram}), c_{\max}(\text{ngram}))}{\sum_{c \in \{\text{Candidates}\}} \sum_{\text{ngram} \in c} c(\text{ngram})}$$

Unigram	Unigram Frequency (Candidate)	Unigram Frequency (Reference)
Mina	1	1
did	1	0
not	1	0
give	1	0
a	1	0
slap	1	1
to	1	0
the	1	1
green	1	1
witch	1	1
.	1	1

$$BP = \begin{cases} 1 & \text{if } c > r \\ e^{(1-\frac{r}{c})} & \text{if } c \leq r \end{cases}$$

$$BLEU = BP * \left( \prod_{n=1}^4 \text{prec}_n \right)^{\frac{1}{4}}$$

$$p_1 = \frac{1 + 0 + 0 + 0 + 0 + 1 + 0 + 1 + 1 + 1 + 1}{1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1} = \frac{6}{11}$$

# Example: Computing BLEU

Mina didn't slap the green witch.

Mina did not give a slap to the green witch.

$$\text{prec}_n = \frac{\sum_{c \in \{\text{Candidates}\}} \sum_{\text{ngram} \in c} \min(c(\text{ngram}), c_{\max}(\text{ngram}))}{\sum_{c \in \{\text{Candidates}\}} \sum_{\text{ngram} \in c} c(\text{ngram})}$$

$$BP = \begin{cases} 1 & \text{if } c > r \\ e^{(1-\frac{r}{c})} & \text{if } c \leq r \end{cases}$$

Bigram	Bigram Frequency (Candidate)	Bigram Frequency (Reference)
Mina did	1	0
did not	1	0
not give	1	0
give a	1	0
a slap	1	0
slap to	1	0
to the	1	0
the green	1	1
green witch	1	1
witch .	1	1

$$BLEU = BP * \left( \prod_{n=1}^4 \text{prec}_n \right)^{\frac{1}{4}}$$

$$p_1 = \frac{1 + 0 + 0 + 0 + 0 + 1 + 0 + 1 + 1 + 1 + 1}{1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1} = \frac{6}{11}$$

$$p_2 = \frac{0 + 0 + 0 + 0 + 0 + 0 + 0 + 1 + 1 + 1}{1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1} = \frac{3}{10}$$

# Example: Computing BLEU

Mina didn't slap the green witch.

Mina did not give a slap to the green witch.

$$\text{prec}_n = \frac{\sum_{c \in \{\text{Candidates}\}} \sum_{\text{ngram} \in c} \min(c(\text{ngram}), c_{\max}(\text{ngram}))}{\sum_{c \in \{\text{Candidates}\}} \sum_{\text{ngram} \in c} c(\text{ngram})}$$

$$BP = \begin{cases} 1 & \text{if } c > r \\ e^{(1-\frac{r}{c})} & \text{if } c \leq r \end{cases}$$

Trigram	Trigram Frequency (Candidate)	Trigram Frequency (Reference)
Mina did not	1	0
did not give	1	0
not give a	1	0
give a slap	1	0
a slap to	1	0
slap to the	1	0
to the green	1	0
the green witch	1	1
green witch .	1	1

$$BLEU = BP * \left( \prod_{n=1}^4 \text{prec}_n \right)^{\frac{1}{4}}$$

$$p_1 = \frac{6}{11} \quad p_2 = \frac{3}{10}$$

$$p_3 = \frac{0 + 0 + 0 + 0 + 0 + 0 + 0 + 0 + 1 + 1}{1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1} = \frac{2}{9}$$

# Example: Computing BLEU

Mina didn't slap the green witch.

Mina did not give a slap to the green witch.

$$\text{prec}_n = \frac{\sum_{c \in \{\text{Candidates}\}} \sum_{\text{ngram} \in c} \min(c(\text{ngram}), c_{\max}(\text{ngram}))}{\sum_{c \in \{\text{Candidates}\}} \sum_{\text{ngram} \in c} c(\text{ngram})}$$

$$BP = \begin{cases} 1 & \text{if } c > r \\ e^{(1-\frac{r}{c})} & \text{if } c \leq r \end{cases}$$

4-gram	4-gram Frequency (Candidate)	4-gram Frequency (Reference)
Mina did not give	1	0
did not give a	1	0
not give a slap	1	0
give a slap to	1	0
a slap to the	1	0
slap to the green	1	0
to the green witch	1	0
the green witch .	1	1

$$BLEU = BP * \left( \prod_{n=1}^4 \text{prec}_n \right)^{\frac{1}{4}}$$

$$p_1 = \frac{6}{11} \quad p_2 = \frac{3}{10} \quad p_3 = \frac{2}{9}$$

$$p_4 = \frac{0 + 0 + 0 + 0 + 0 + 0 + 0 + 1}{1 + 1 + 1 + 1 + 1 + 1 + 1 + 1} = \frac{1}{8}$$

# Example: Computing BLEU

Mina didn't slap the green witch.

Mina did not give a slap to the green witch.

$r = 7$

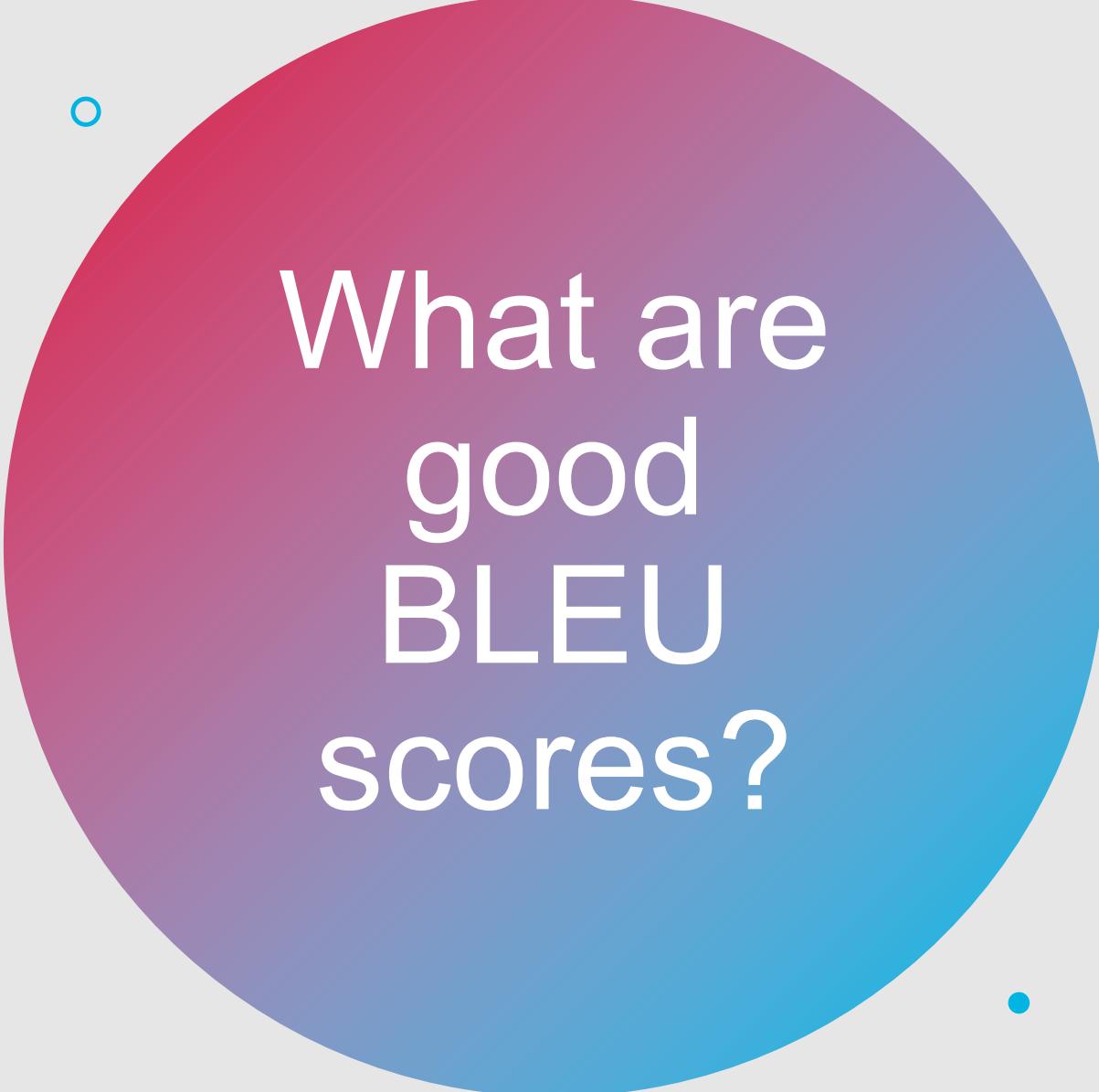
$$\text{prec}_n = \frac{\sum_{c \in \{\text{Candidates}\}} \sum_{\text{ngram} \in c} \min(c(\text{ngram}), c_{\max}(\text{ngram}))}{\sum_{c \in \{\text{Candidates}\}} \sum_{\text{ngram} \in c} c(\text{ngram})}$$

$$BP = \begin{cases} 1 & \text{if } c > r \\ e^{(1-\frac{r}{c})} & \text{if } c \leq r \end{cases}$$

$$p_1 = \frac{6}{11} \quad p_2 = \frac{3}{10} \quad p_3 = \frac{2}{9} \quad p_4 = \frac{1}{8}$$

$$BP = 1$$

$$\text{BLEU} = 1 * (\prod_{n=1}^4 \text{prec}_n)^{\frac{1}{4}} = 1 * (\frac{6}{11} * \frac{3}{10} * \frac{2}{9} * \frac{1}{8})^{\frac{1}{4}} = 1 * 0.0045454545454^{\frac{1}{4}} = 1 * 0.25965358893 = 0.26$$



# What are good BLEU scores?

- No formal score ranges, but in general:
  - < 10: Very poor
  - 10-19: Difficult to understand
  - 20-29: Understandable but with many grammatical errors
  - 30-39: Understandable and reasonable quality
  - 40-49: High quality
  - 50-59: Very high quality
  - >60: May exceed human translators

# Limitations of BLEU

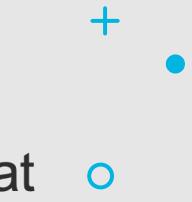
- Word or phrase order is of minimal importance
  - When computing unigram precision, a word can exist anywhere in the translation!
- Does not consider word similarity
- Relatively low correlation with human ratings
- Nonetheless, BLEU is reasonable to use in cases when a quick, automated metric is needed to assess translation performance

# Character F-Score (chrF)

- Same intuition as BLEU: Good machine translations tend to contain the same words and characters as human translations
- Ranks a candidate translation by a function of the number of character n-gram overlaps with a human reference translation
- Less sensitive to differences in wordform than BLEU (e.g., “gives” versus “is giving”)

# How is chrF computed?

- Similarly to “regular” F-score
  - **chrP**: averaged % of character unigrams, bigrams, ..., k-grams in the hypothesis that also occur in the reference
  - **chrR**: averaged % of character unigrams, bigrams, ..., k-grams in the reference that also occur in the hypothesis
  - **$\beta$** : weighting parameter (similarly to F-score) that determines the relative impacts of chrP and chrR on the overall F-score
  - $\text{chrF}\beta = (1 + \beta^2) \frac{\text{chrP} \times \text{chrR}}{\beta^2 \times \text{chrP} + \text{chrR}}$ , or for example,  $\text{chrF2} = \frac{5 \times \text{chrP} \times \text{chrR}}{4 \times \text{chrP} + \text{chrR}}$

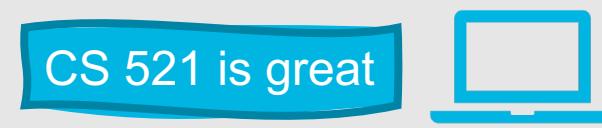


# Example: Computing chrF

CS 521 is the best



# Example: Computing chrF



# Example: Computing chrF



C	S	5	2	1	i	s	t	h	e	b	e	s	t
---	---	---	---	---	---	---	---	---	---	---	---	---	---

C	S	5	2	1	i	s	g	r	e	a	t
---	---	---	---	---	---	---	---	---	---	---	---

# Example: Computing chrF



C	S	5	2	1	i	s	t	h	e	b	e	s	t
C	S	5	2	1	i	s	g	r	e	a	t		

14 unigrams, 13 bigrams  
12 unigrams, 11 bigrams

# Example: Computing chrF

CS 521 is the best



CS 521 is great



C	S	5	2	1	i	s	t	h	e	b	e	s	t
C	S	5	2	1	i	s	g	r	e	a	t		

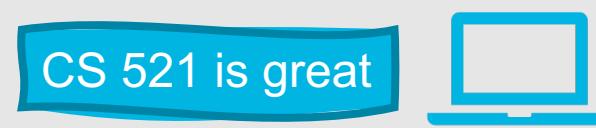
14 unigrams, 13 bigrams  
12 unigrams, 11 bigrams

**chrP:** averaged % of character unigrams, bigrams, ..., k-grams  
in the hypothesis that are also in the reference

k=3

**chrR:** averaged % of character unigrams, bigrams, ..., k-grams  
in the reference that are also in the hypothesis

# Example: Computing chrF



**chrP:** averaged % of character unigrams, bigrams, ..., k-grams in the hypothesis that are also in the reference

**chrR:** averaged % of character unigrams, bigrams, ..., k-grams in the reference that are also in the hypothesis

k=3

9 unigrams are in both the hypothesis and the reference

# Example: Computing chrF



CS	S5	52	21	1i	is	st	th	he	eb	be	es	st
----	----	----	----	----	----	----	----	----	----	----	----	----

14 unigrams, 13 bigrams

CS	S5	52	21	1i	is	sg	gr	re	ea	at
----	----	----	----	----	----	----	----	----	----	----

12 unigrams, 11 bigrams

**chrP:** averaged % of character unigrams, bigrams, ..., k-grams in the hypothesis are also in the reference

**chrR:** averaged % of character unigrams, bigrams, ..., k-grams in the reference that are also in the hypothesis

k=3

9 unigrams are in both the hypothesis and the reference

6 bigrams are in both the hypothesis and the reference

# Example: Computing chrF

CS 521 is the best



CS 521 is great



CS5	S52	521	21i	1is	ist	sth	the	heb	ebe	bes	est	← 14 unigrams, 13 bigrams
CS5	S52	521	21i	1is	isg	sgr	gre	rea	eat	← 12 unigrams, 11 bigrams		

**chrP:** averaged % of character unigrams, bigrams, ..., k-grams in the hypothesis that are also in the reference

**chrR:** averaged % of character unigrams, bigrams, ..., k-grams in the reference that are also in the hypothesis

k=3

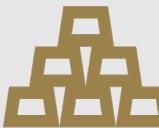
9 unigrams are in both the hypothesis and the reference

6 bigrams are in both the hypothesis and the reference

5 trigrams are in both the hypothesis and the reference

# Example: Computing chrF

CS 521 is the best



CS 521 is great



CS5 | S52 | 521 | 21i | 1is | ist | sth | the | heb | ebe | bes | est ← 14 unigrams, 13 bigrams

CS5 | S52 | 521 | 21i | 1is | isg | sgr | gre | rea | eat ← 12 unigrams, 11 bigrams

**chrP:** averaged % of character unigrams, bigrams, ..., k-grams in the hypothesis that are also in the reference

**chrR:** averaged % of character unigrams, bigrams, ..., k-grams in the reference that are also in the hypothesis

k=3

9 unigrams are in both the hypothesis and the reference

Unigram chrP:  $9/12 = 0.75$

Unigram chrR:  $9/14 = 0.64$

6 bigrams are in both the hypothesis and the reference

5 trigrams are in both the hypothesis and the reference

# Example: Computing chrF

CS 521 is the best



CS 521 is great



CS5 | S52 | 521 | 21i | 1is | ist | sth | the | heb | ebe | bes | est ← 14 unigrams, 13 bigrams

CS5 | S52 | 521 | 21i | 1is | isg | sgr | gre | rea | eat ← 12 unigrams, 11 bigrams

**chrP:** averaged % of character unigrams, bigrams, ..., k-grams in the hypothesis that are also in the reference

**chrR:** averaged % of character unigrams, bigrams, ..., k-grams in the reference that are also in the hypothesis

k=3

9 unigrams are in both the hypothesis and the reference

Unigram chrP:  $9/_{12} = 0.75$

Unigram chrR:  $9/_{14} = 0.64$

6 bigrams are in both the hypothesis and the reference

Bigram chrP:  $6/_{11} = 0.55$

Bigram chrR:  $6/_{13} = 0.46$

5 trigrams are in both the hypothesis and the reference

# Example: Computing chrF

CS 521 is the best



CS 521 is great



CS5 | S52 | 521 | 21i | 1is | ist | sth | the | heb | ebe | bes | est ← 14 unigrams, 13 bigrams

CS5 | S52 | 521 | 21i | 1is | isg | sgr | gre | rea | eat ← 12 unigrams, 11 bigrams

**chrP:** averaged % of character unigrams, bigrams, ..., k-grams in the hypothesis that are also in the reference

**chrR:** averaged % of character unigrams, bigrams, ..., k-grams in the reference that are also in the hypothesis

k=3

9 unigrams are in both the hypothesis and the reference

Unigram chrP:  $9/12 = 0.75$

Unigram chrR:  $9/14 = 0.64$

6 bigrams are in both the hypothesis and the reference

Bigram chrP:  $6/11 = 0.55$

Bigram chrR:  $6/13 = 0.46$

5 trigrams are in both the hypothesis and the reference

Trigram chrP:  $5/10 = 0.5$

Trigram chrR:  $5/12 = 0.42$

# Example: Computing chrF

CS 521 is the best



CS 521 is great



CS5 | S52 | 521 | 21i | 1is | ist | sth | the | heb | ebe | bes | est ← 14 unigrams, 13 bigrams

CS5 | S52 | 521 | 21i | 1is | isg | sgr | gre | rea | eat ← 12 unigrams, 11 bigrams

**chrP:** averaged % of character unigrams, bigrams, ..., k-grams in the hypothesis that are also in the reference

**chrR:** averaged % of character unigrams, bigrams, ..., k-grams in the reference that are also in the hypothesis

k=3

Unigram chrP:  $9/_{12} = 0.75$

Unigram chrR:  $9/_{14} = 0.64$

Bigram chrP:  $6/_{11} = 0.55$

Bigram chrR:  $6/_{13} = 0.46$

Trigram chrP:  $5/_{10} = 0.5$

Trigram chrR:  $5/_{12} = 0.42$

chrP:  $\frac{0.75+0.55+0.5}{3} = 0.6$

# Example: Computing chrF

CS 521 is the best



CS 521 is great



CS5 | S52 | 521 | 21i | 1is | ist | sth | the | heb | ebe | bes | est ← 14 unigrams, 13 bigrams

CS5 | S52 | 521 | 21i | 1is | isg | sgr | gre | rea | eat ← 12 unigrams, 11 bigrams

**chrP:** averaged % of character unigrams, bigrams, ..., k-grams in the hypothesis that are also in the reference

**chrR:** averaged % of character unigrams, bigrams, ..., k-grams in the reference that are also in the hypothesis

k=3

Unigram chrP:  $9/12 = 0.75$

Unigram chrR:  $9/14 = 0.64$

Bigram chrP:  $6/11 = 0.55$

Bigram chrR:  $6/13 = 0.46$

Trigram chrP:  $5/10 = 0.5$

Trigram chrR:  $5/12 = 0.42$

chrP:  $\frac{0.75+0.55+0.5}{3} = 0.6$

chrR:  $\frac{0.64+0.46+0.42}{3} = 0.51$

# Example: Computing chrF

CS 521 is the best



CS 521 is great



CS5 | S52 | 521 | 21i | 1is | ist | sth | the | heb | ebe | bes | est ← 14 unigrams, 13 bigrams

CS5 | S52 | 521 | 21i | 1is | isg | sgr | gre | rea | eat ← 12 unigrams, 11 bigrams

**chrP:** averaged % of character unigrams, bigrams, ..., k-grams in the hypothesis that are also in the reference

**chrR:** averaged % of character unigrams, bigrams, ..., k-grams in the reference that are also in the hypothesis

k=3

$$\text{chrP: } \frac{0.75+0.55+0.5}{3} = 0.6$$

$$\text{chrR: } \frac{0.64+0.46+0.42}{3} = 0.51$$

$$\text{chrF2} = \frac{5 * \text{chrP} * \text{chrR}}{4 * \text{chrP} + \text{chrR}} = \frac{5 * 0.6 * 0.51}{4 * 0.6 + 0.51} = 0.53$$

# Limitations of chrF

- Focuses on differences at a very local scale (i.e., character n-grams)
- Doesn't measure discourse coherence
- Best at measuring performance for different versions of the same system, rather than comparing different systems

# Embedding -Based Evaluation Methods

- Measuring exact word- or character-level overlap might be overly strict
  - Good translations may use words that are synonymous to those in the reference!
- Embedding-based methods measure the *semantic* overlap between reference and hypothesis translations

# Popular Embedding-Based Methods for Evaluating MT Systems

## COMET

- <https://github.com/Unbabel/COMET>

## BLEURT

- <https://github.com/google-research/bleurt>

## BERTScore

- [https://github.com/Tiiiger/bert\\_score](https://github.com/Tiiiger/bert_score)

# What is question answering?

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

A screenshot of a search engine results page. The query "What is UIC's mascot?" is entered in the search bar. Below the search bar, there are navigation links: All, Images, Shopping, Maps, News, More, Settings, and To. It shows approximately 167,000 results found in 1.07 seconds. The top result is from the University of Illinois at Chicago / Mascot, featuring the text "Sparky D. Dragon". A "People also search for" section lists other university mascots: The University of Chicago... (Phil the Phoenix), Loyola University Chicago (LU Wolf), and DePaul University DIBS. At the bottom, there is a link to dos.uic.edu > About > Student Handbook and a note about UIC Symbols.

A screenshot of a question answering interface. A blue button labeled "PA" is on the left. To its right, the question "What is UIC's mascot?" is displayed. Below the question, a green icon representing a question mark or search is shown next to the text: "The University of Illinois at Chicago's mascot is Sparky D. Dragon." This represents a compact and direct retrieval of the correct information in response to the user's query.

# 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 Chomsky, and Kenneth Laughery. 1961. Baseball: An Automatic Question Answerer. Link: <https://web.stanford.edu/class/linquist289/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



# 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

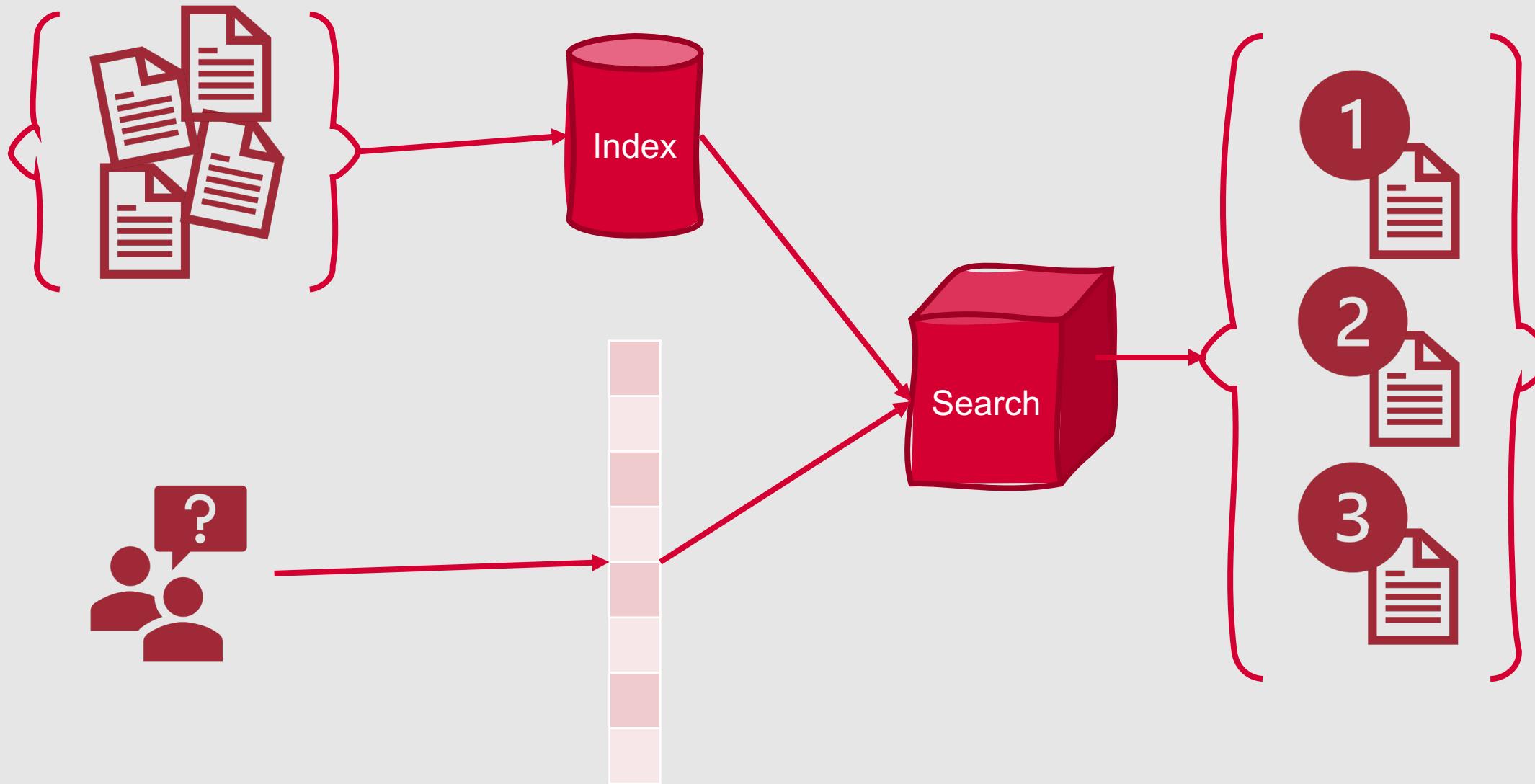
- Up until recently, QA systems operated under two paradigms:
  - **Information retrieval-based** question answering
  - **Knowledge-based** question answering
- More recently, we've seen many systems using:
  - **Language model-based** question answering
- Further back in time, we also saw:
  - **Classic 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

# How does information retrieval work?



# How are documents represented?

- Two common term weighting schemes:

- TF-IDF

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

- BM25

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

Balance between term frequency and inverse document frequency

Importance of document length normalization

# Document Scoring

- Create document and query vectors using term weights
- Compute cosine similarity between a document,  $d$ , and query,  $q$ 
  - $\text{score}(q, d) = \cos(q, d) = \frac{\mathbf{q} \cdot \mathbf{d}}{|\mathbf{q}| |\mathbf{d}|}$
- Simplify since the query won't vary between documents
  - $\text{score}(q, d) = \frac{\mathbf{q} \cdot \mathbf{d}}{|\mathbf{q}| |\mathbf{d}|} = \sum_{t \in q} \frac{\text{tfidf}(t, d)}{|d|}$

# Document Scoring: Case Example

CS is the best topic!

CS 521 covers statistical NLP.

521 is the best class.

CS 521

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

# Document Scoring: Case Example

CS is the best topic!

CS 521 covers statistical NLP.

521 is the best class.

CS 521

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

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

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

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

# Document Scoring: Case Example

CS is the best topic!

CS 521 covers statistical NLP.

521 is the best class.

CS 521

word	Doc. 1	Doc. 2	Doc. 3
CS	0.053	0.053	0
is	0.053	0	0.053
the	0.053	0	0.053
best	0.053	0	0.053
topic	0.144	0	0
521	0	0.053	0.053
covers	0	0.144	0
statistical	0	0.144	0
NLP	0	0.144	0
class	0	0	0.144

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

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

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

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

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

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

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

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

# Document Scoring: Case Example

CS is the best topic!

CS 521 covers statistical NLP.

521 is the best class.

CS 521

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

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

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

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

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

# Document Scoring: Case Example

CS is the best topic!

CS 521 covers statistical NLP.

521 is the best class.

CS 521

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

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

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

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

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

# Document Scoring: Case Example

CS is the best topic!

CS 521 covers statistical NLP.

521 is the best class.

CS 521

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

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

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

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

# 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:
    - Less efficient
  - Typically done by:
    - Separately encoding the 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

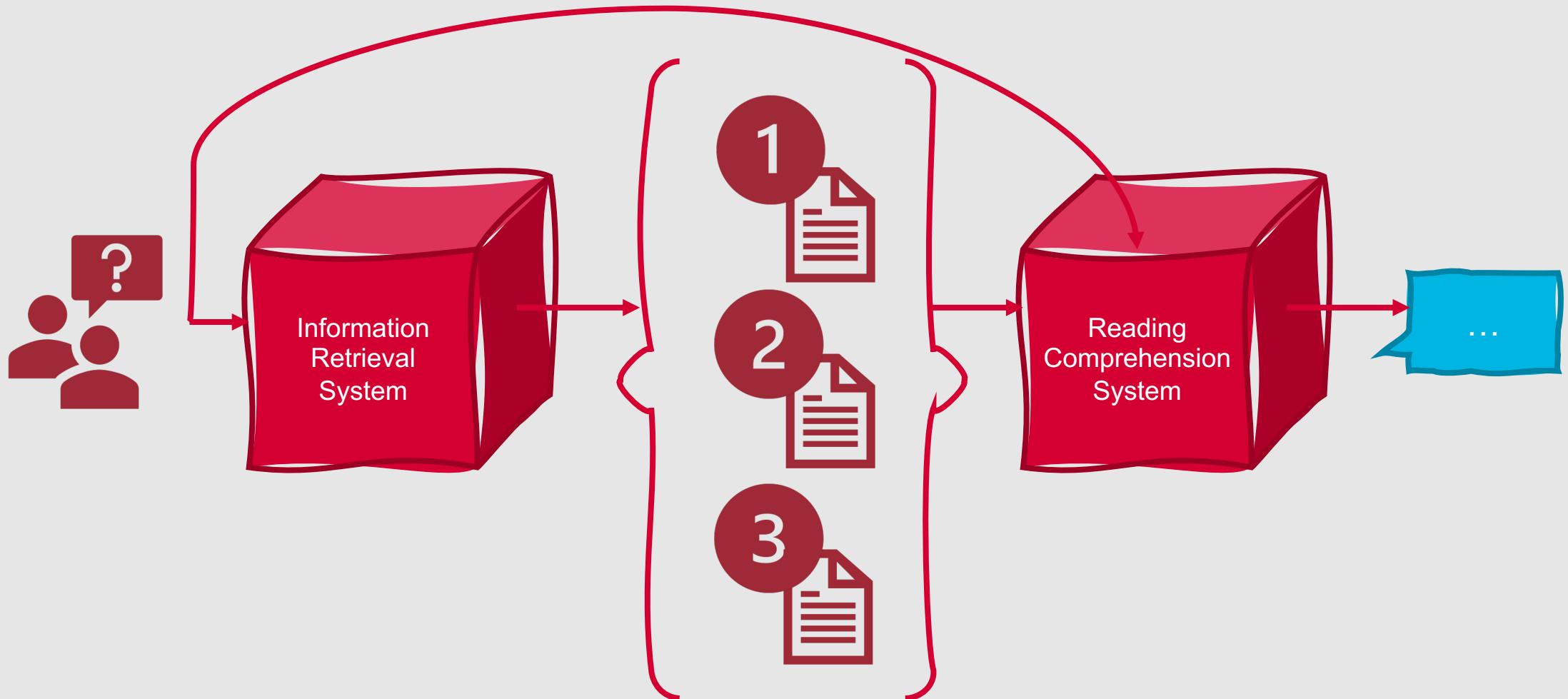


Retrieve  
and Read

Dominant Paradigm: **Retrieve and read** model

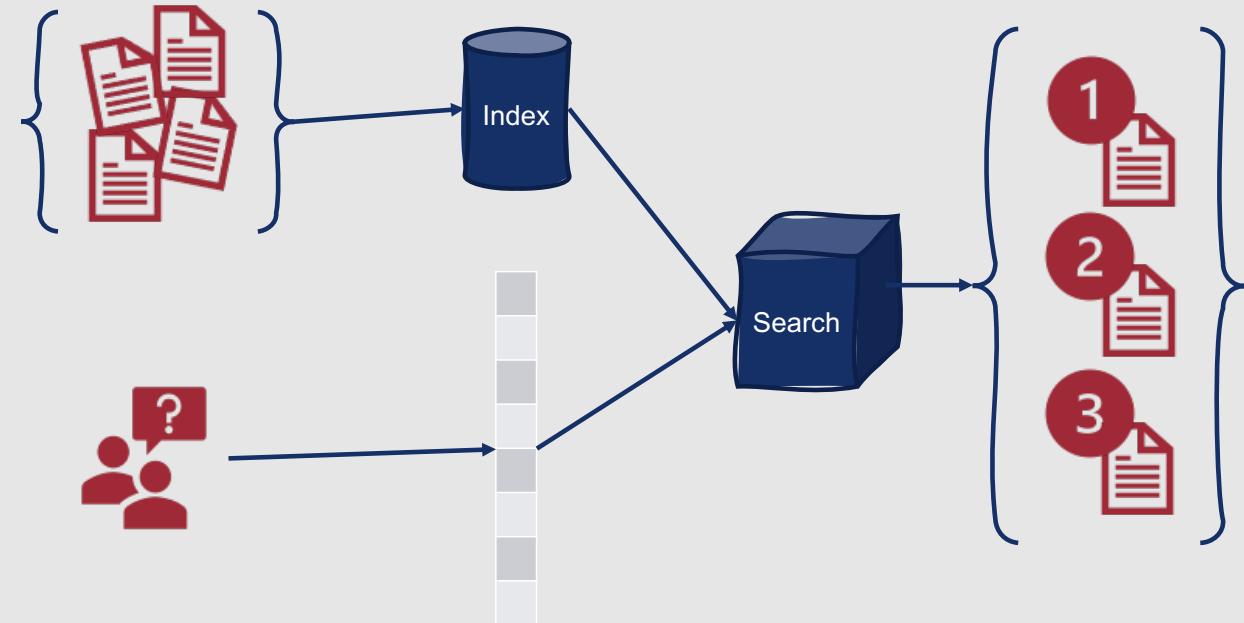
- **Retrieve** relevant documents for the given query
- **Read** those documents to find text segments that answer the query

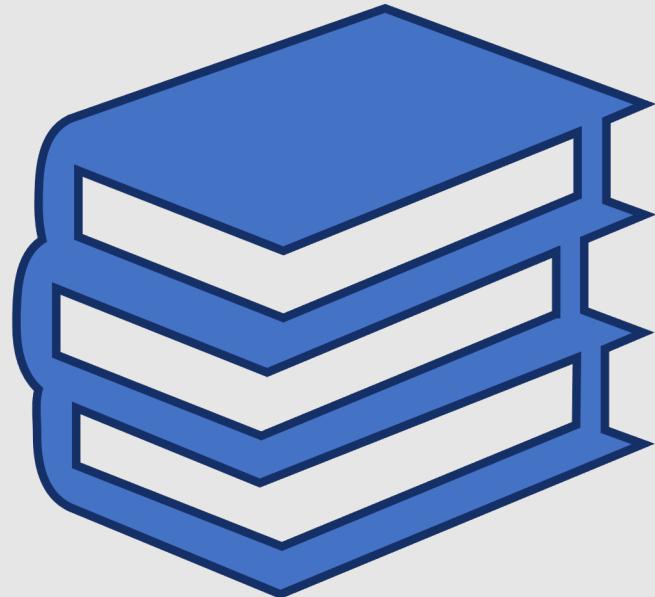
# Retrieve and Read Model



# 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
- Designed to measure natural language understanding performance

## 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

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/tydiga>

# 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.

$P_{start}("13")$

$P_{end}("13")$

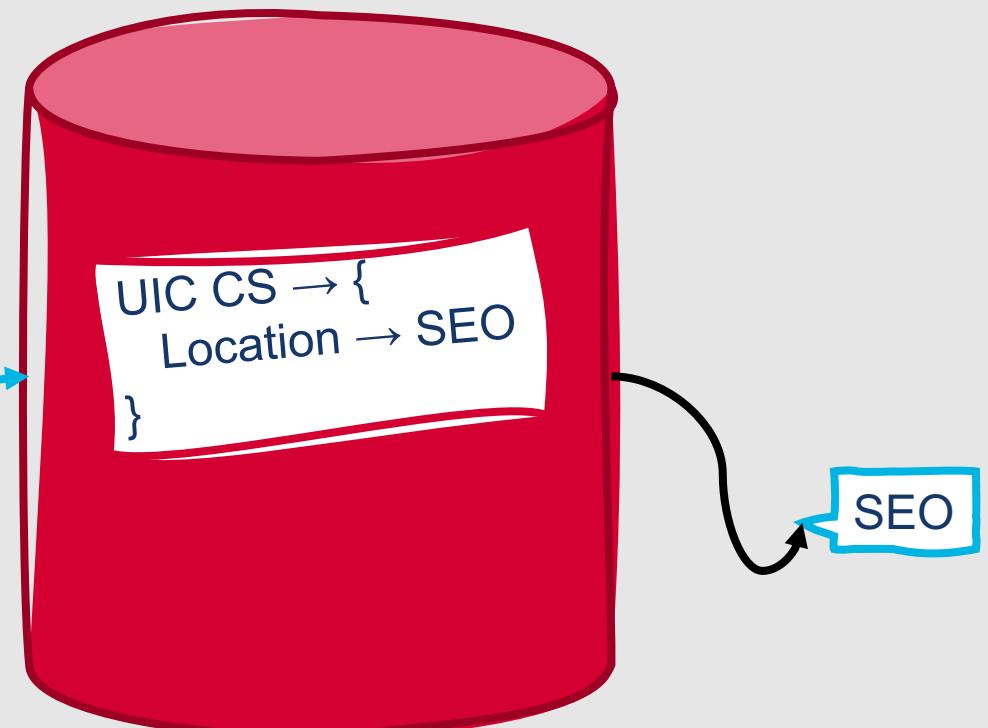
# Answer Span Extraction

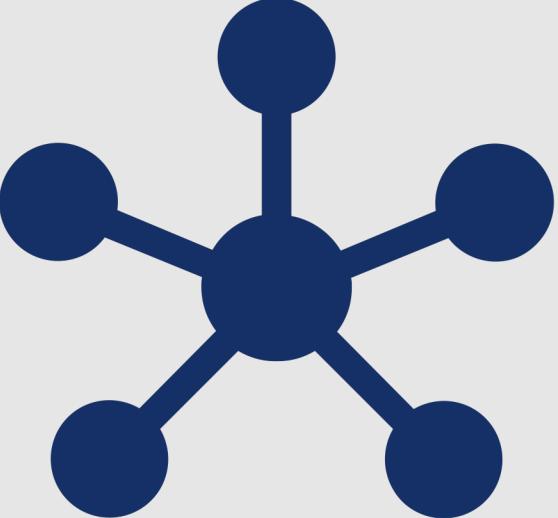
- Common **extractive** QA approach
  - Concatenate the query and passage, separated by a [SEP] token
  - Encode the concatenated sequence
  - Add a linear layer
  - Compute span-start and span-end probabilities for each token  $p_i$  in a passage  $P$ , making use of special span-start (S) and span-end (E) vectors learned during fine-tuning
    - $P_{\text{start}_i} = \frac{e^{S \cdot p_i}}{\sum_{j=0}^{|P|} e^{S \cdot p_j}}$
    - $P_{\text{end}_i} = \frac{e^{E \cdot p_i}}{\sum_{j=0}^{|P|} e^{E \cdot p_j}}$
  - Compute a score for each passage from position  $i$  to  $j$ 
    - $\text{score}(i, j) = S \cdot p_i + E \cdot p_j$
  - Select the highest-scoring passage for which  $j \geq i$

# 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

Where is UIC's computer science department located?





# Knowledge-based Question Answering

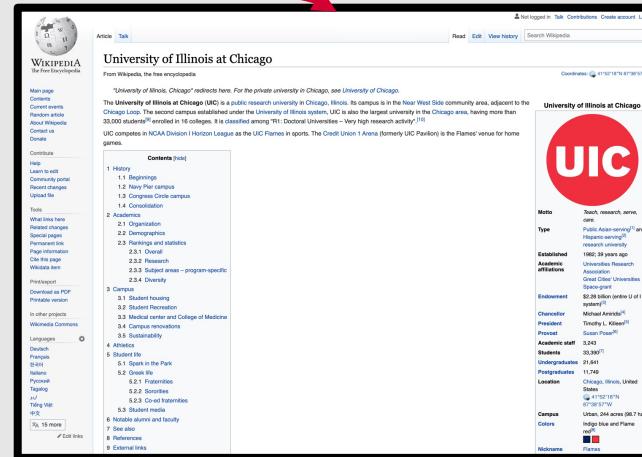
---

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

# Entity Linking

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

- **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



# Neural Graph-based Entity Linking

- Modern approaches often make use of **bidirectional Transformer encoders**
  - 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>

# Graph-based Question Answering

- Facts are stored as (subject, predicate, object) triples
  - Sometimes referred to as RDF (resource description framework) 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)
- Triples matching the canonical relations are identified and ranked 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

# How did classical QA work?

Rule-based  
question  
answering

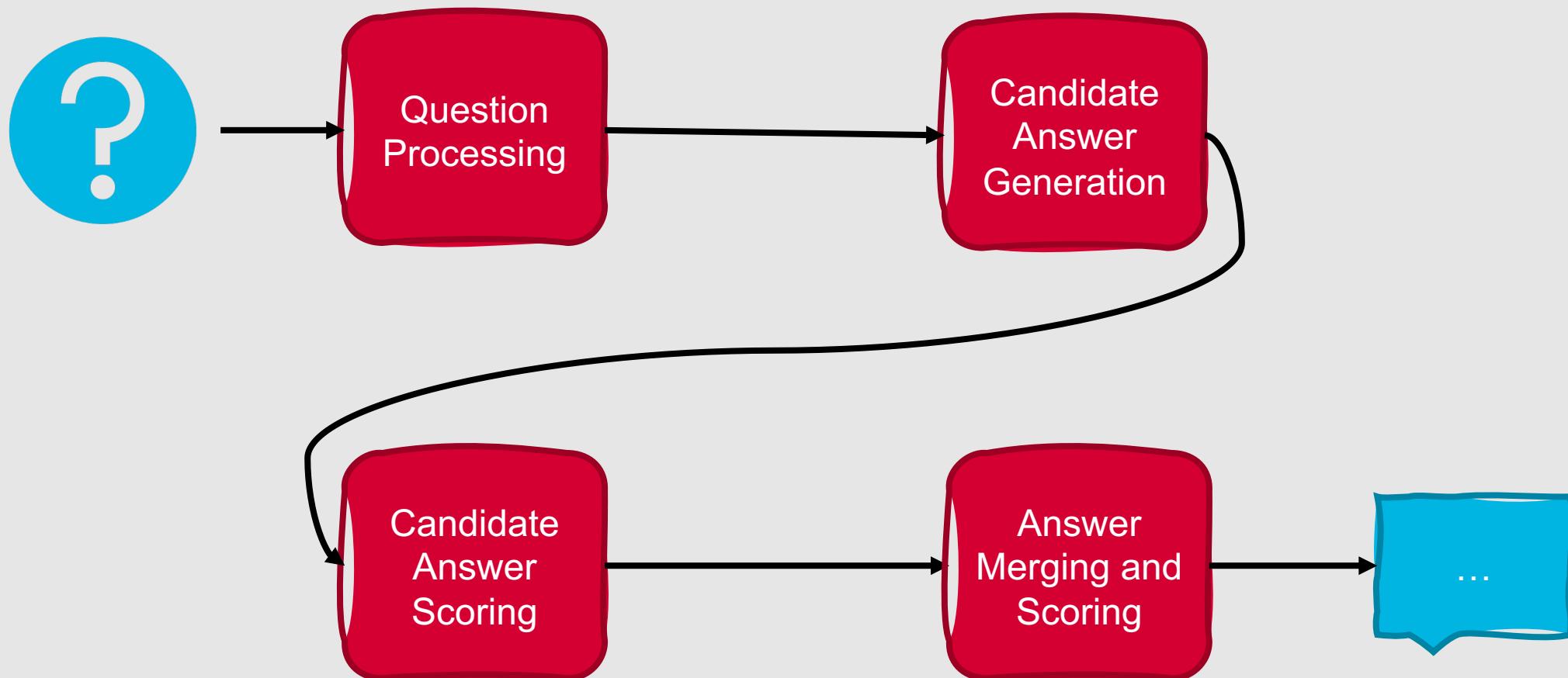
Feature-based  
question  
answering

Hybrid  
techniques that  
incorporated  
both approaches

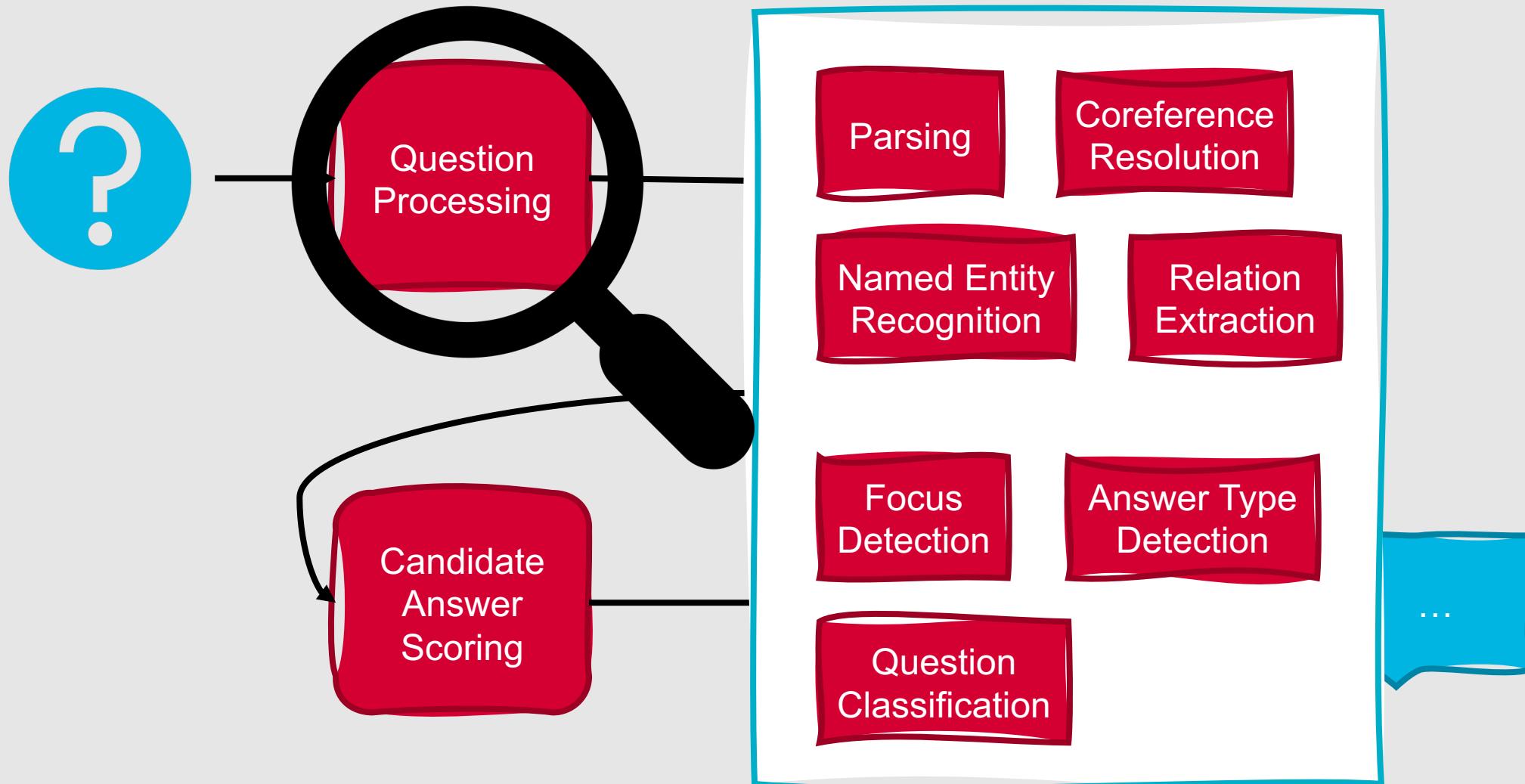
# Hybrid Rule- and Feature- based Question Answering

- Until recently, a popular question answering paradigm involved leveraging a combination of rule-based methods and feature-based classification techniques
  - Question answering component of Watson (DeepQA)
  - Four stages:
    1. **Question processing**
    2. **Candidate answer generation**
    3. **Candidate answer scoring**
    4. **Answer merging and scoring**

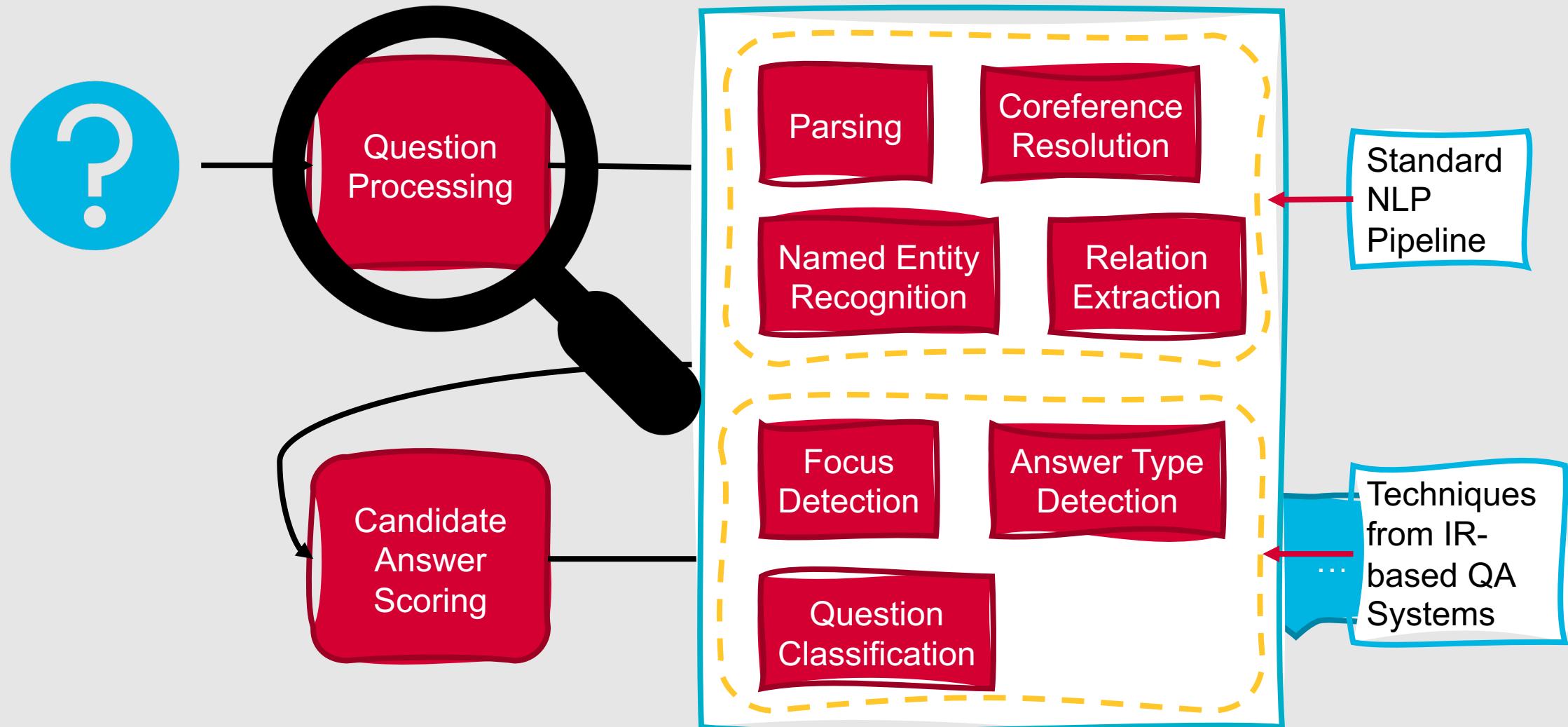
# Case Example: DeepQA



# Stage 1: Question Preprocessing



# Stage 1: Question Preprocessing



# Stage 1: Question Preprocessing

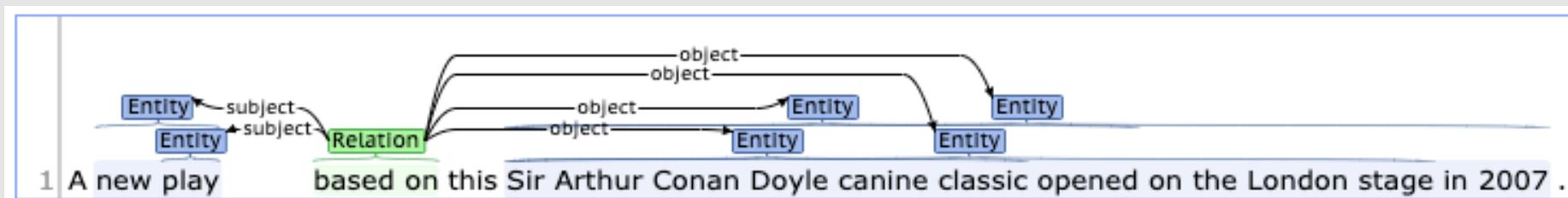
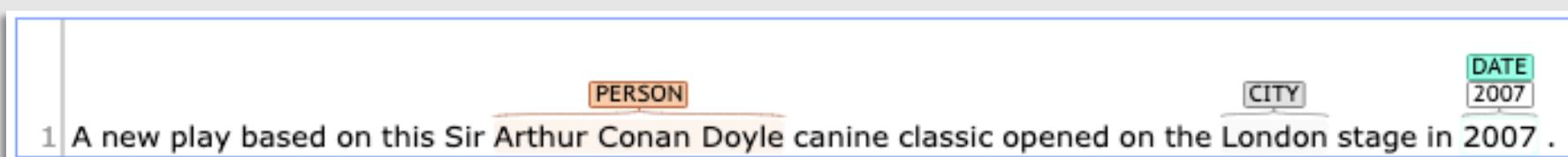
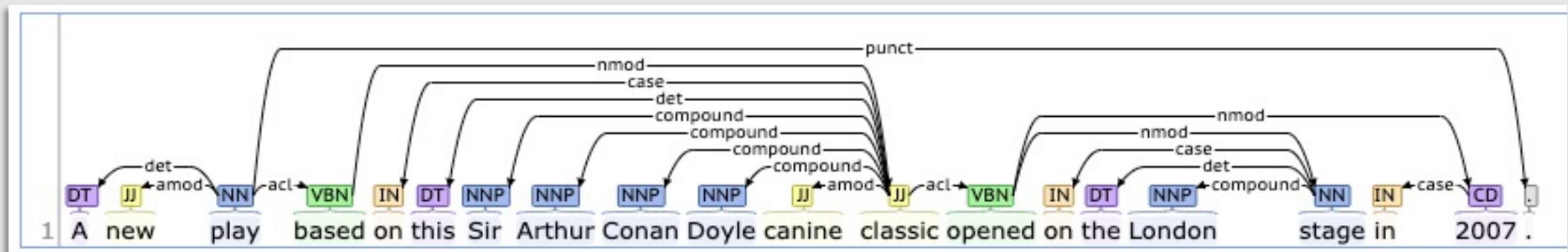
## Jeopardy! Example:

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:

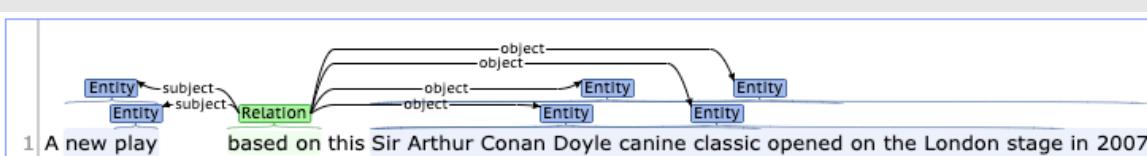
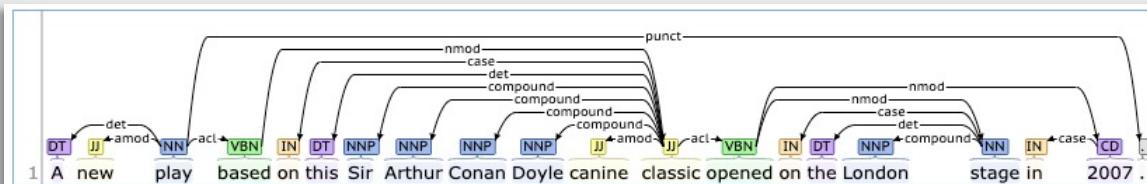
A new play based on this Sir Arthur Conan Doyle canine classic opened on the London stage in 2007.



# 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.



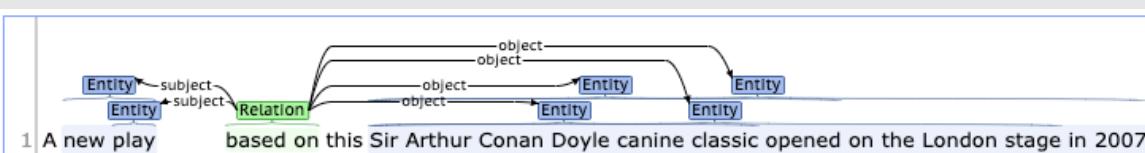
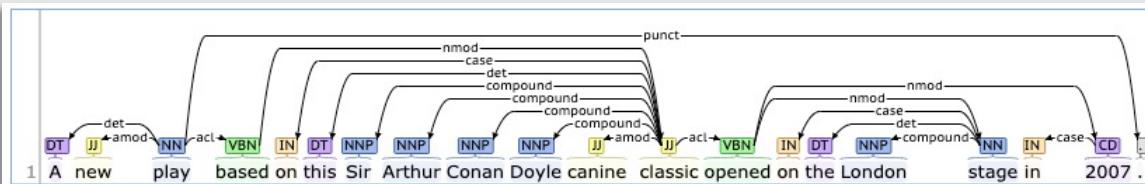
**Focus Detection:** Which part of the question co-refers with the answer?

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.



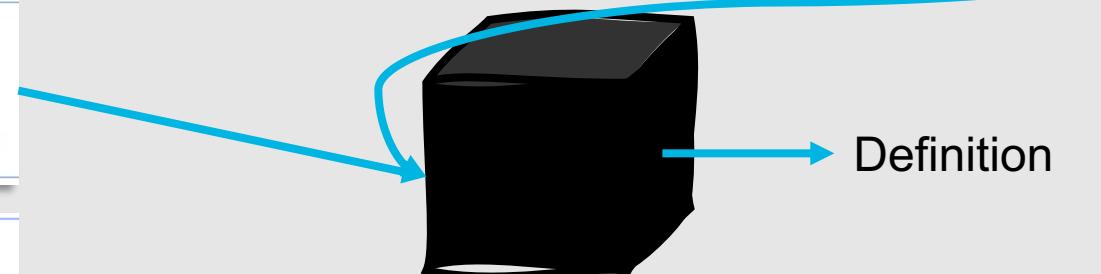
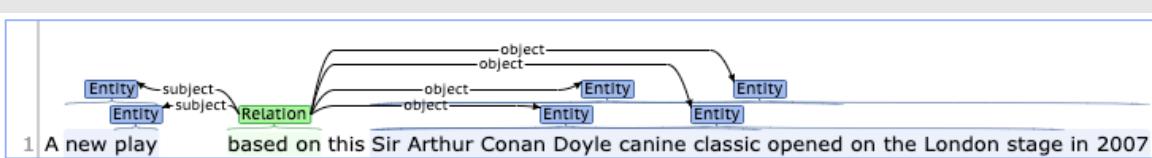
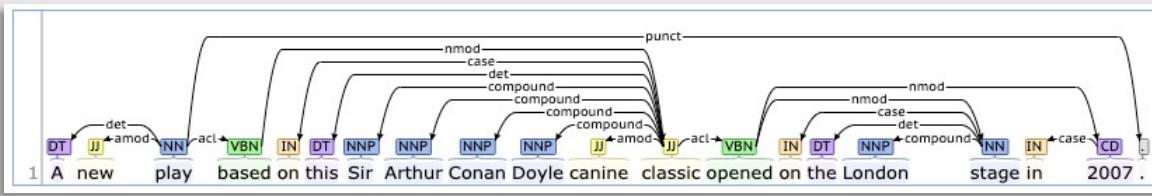
**Answer Type Detection:** Which word tells us about the semantic type of answer to expect?

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:

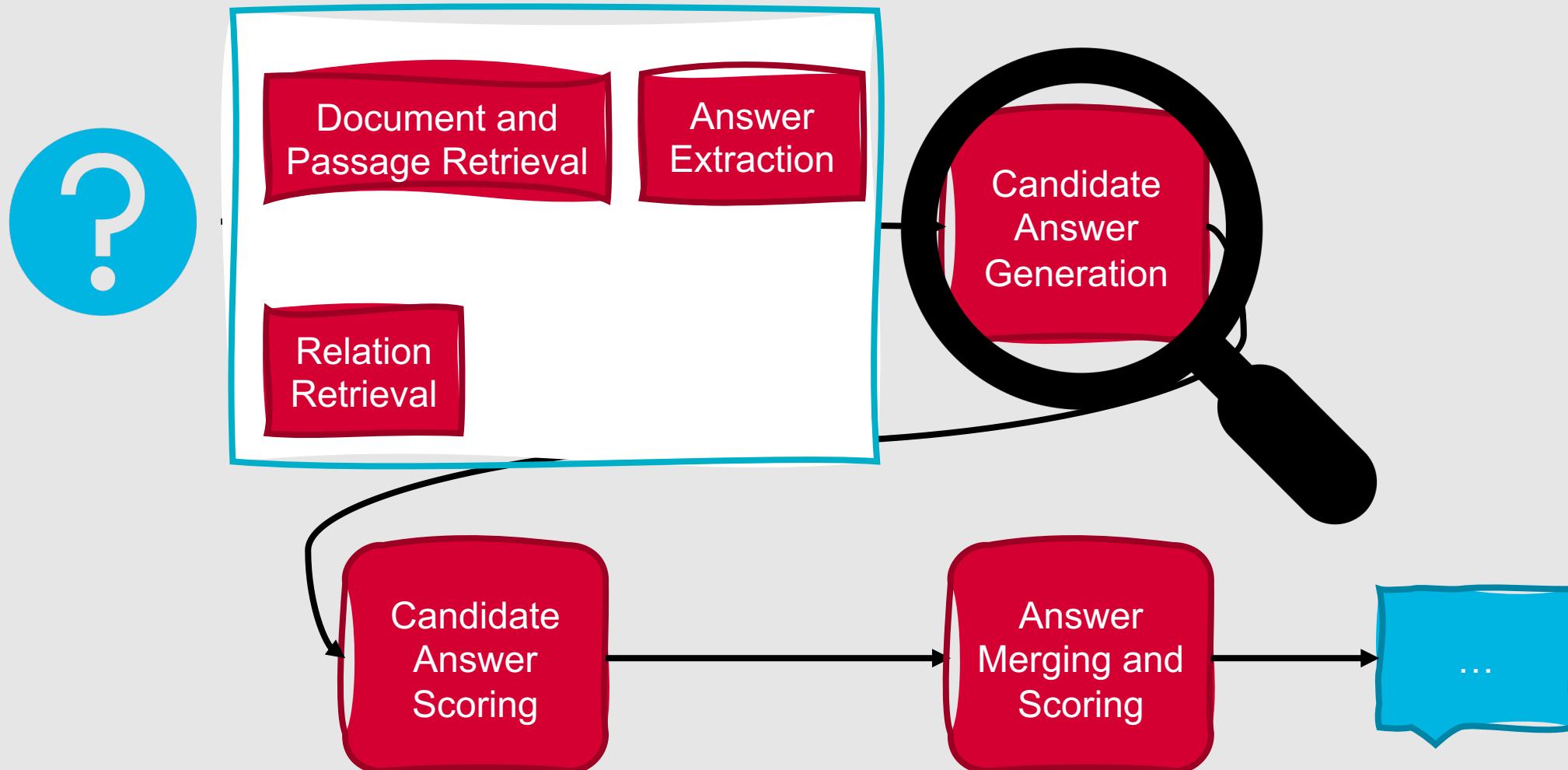
A new play based on this Sir Arthur Conan Doyle canine **classic** opened on the London stage in 2007.



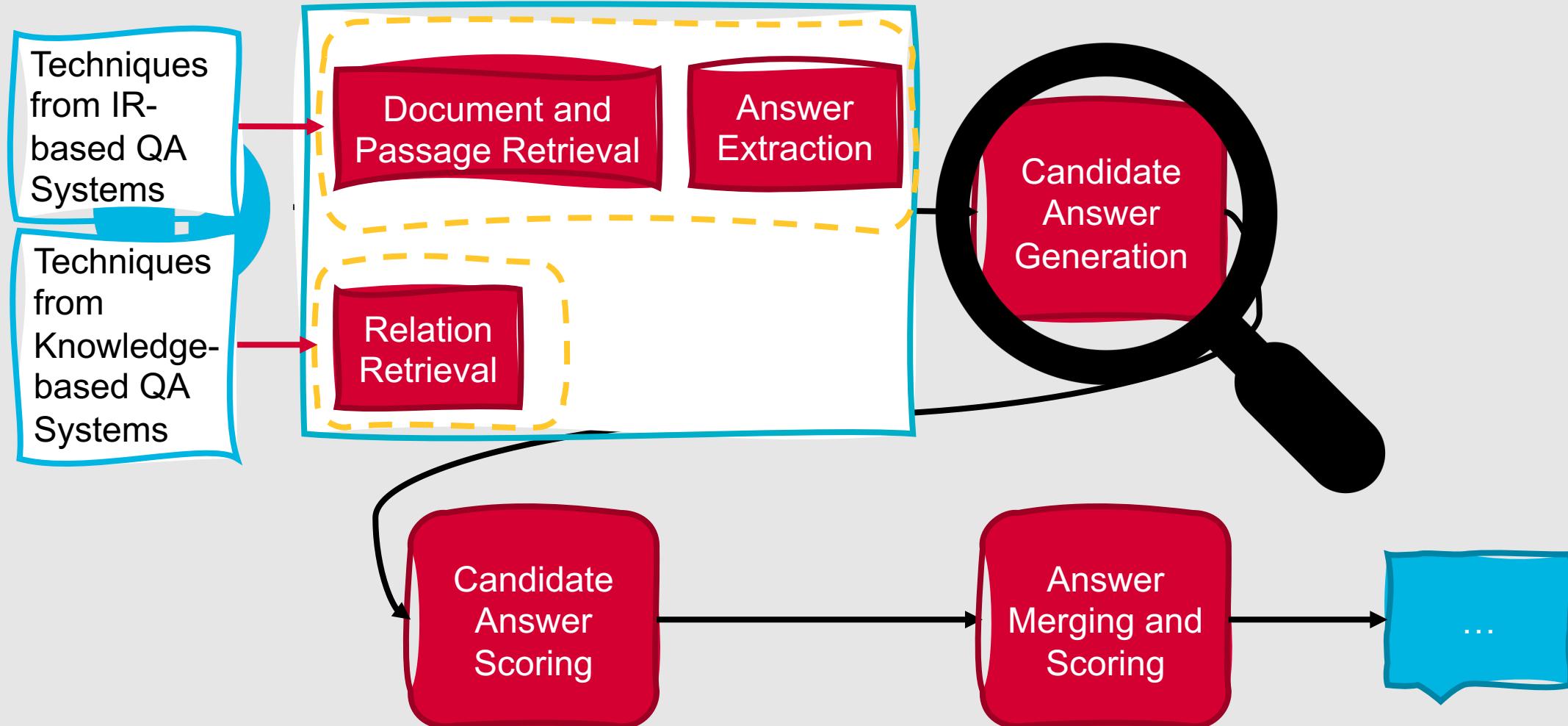
**Question Classification:** What type of question is this (multiple choice, fill-in-the-blank, definition, etc.)?

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:**

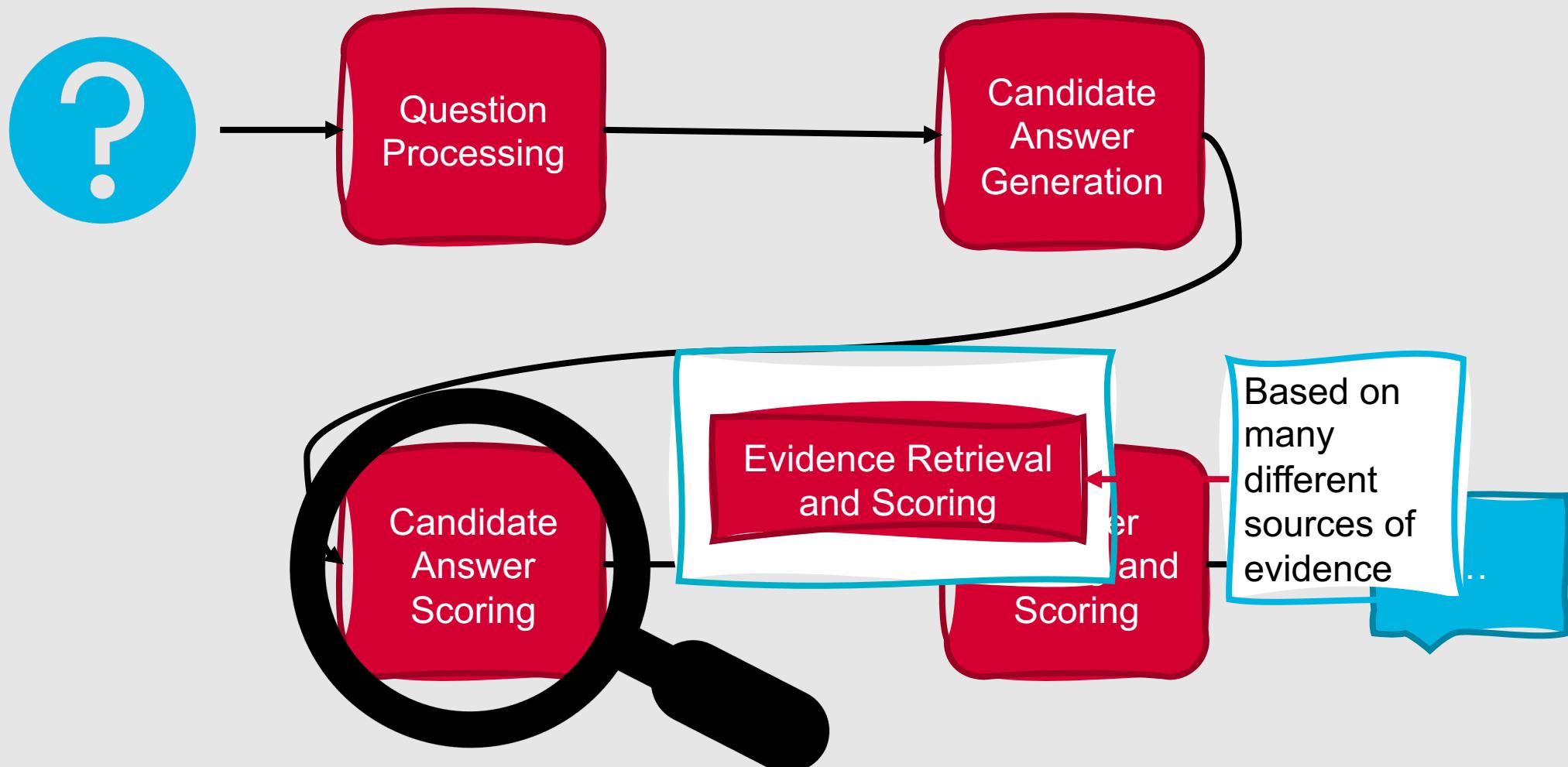
basedOn(x, "Sir Arthur Conan Doyle canine classic")

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)

# Stage 3: Candidate Answer Scoring

The Hound of the Baskervilles

Expected Answer Type: BOOK

The Hound of the Baskervilles

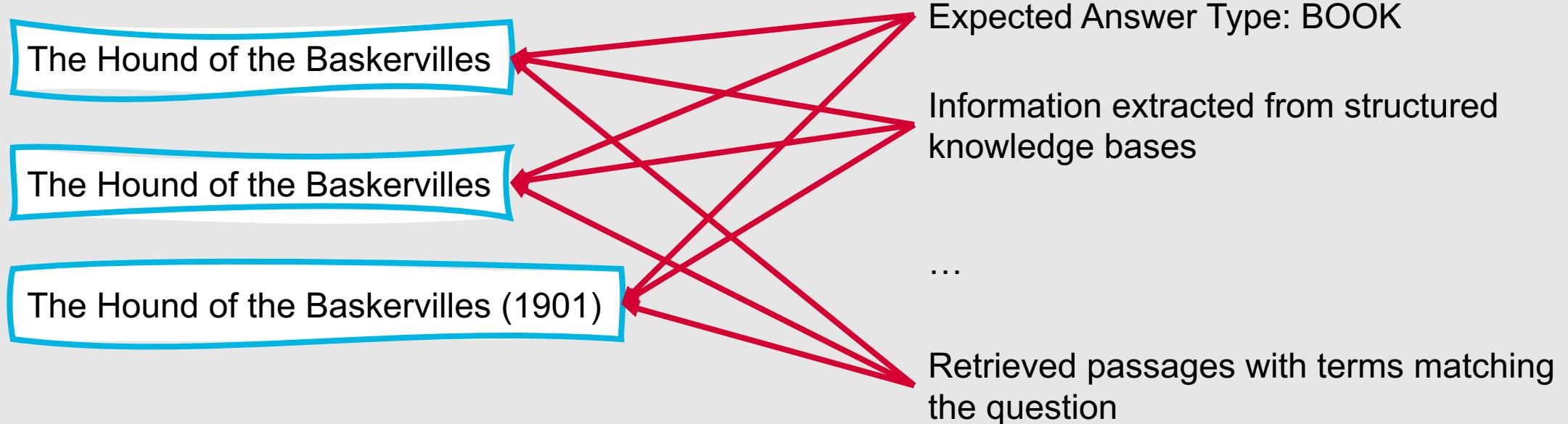
Information extracted from structured knowledge bases

The Hound of the Baskervilles (1901)

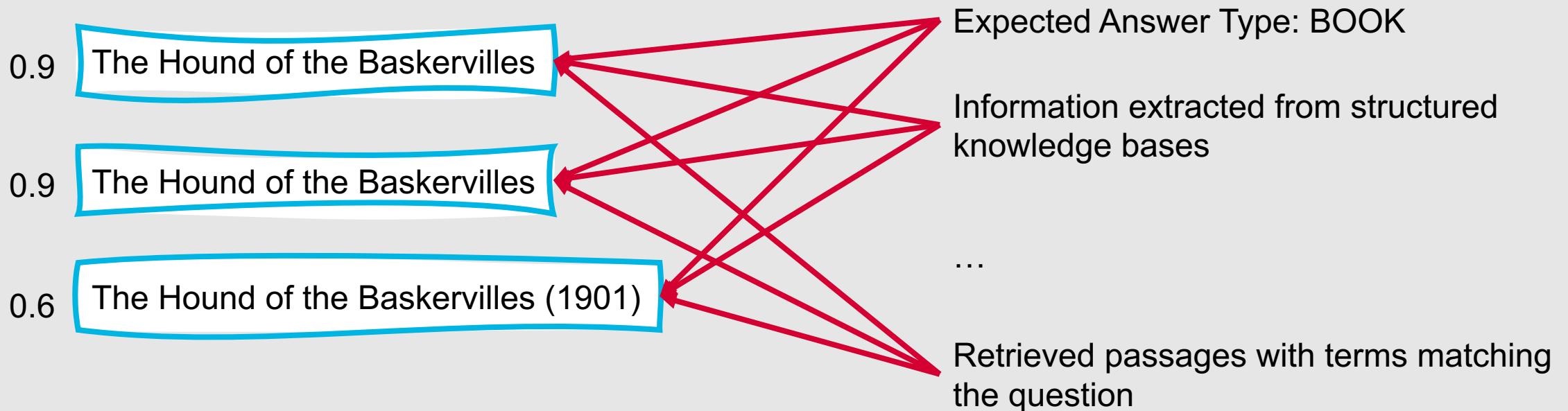
...

Retrieved passages with terms matching the question

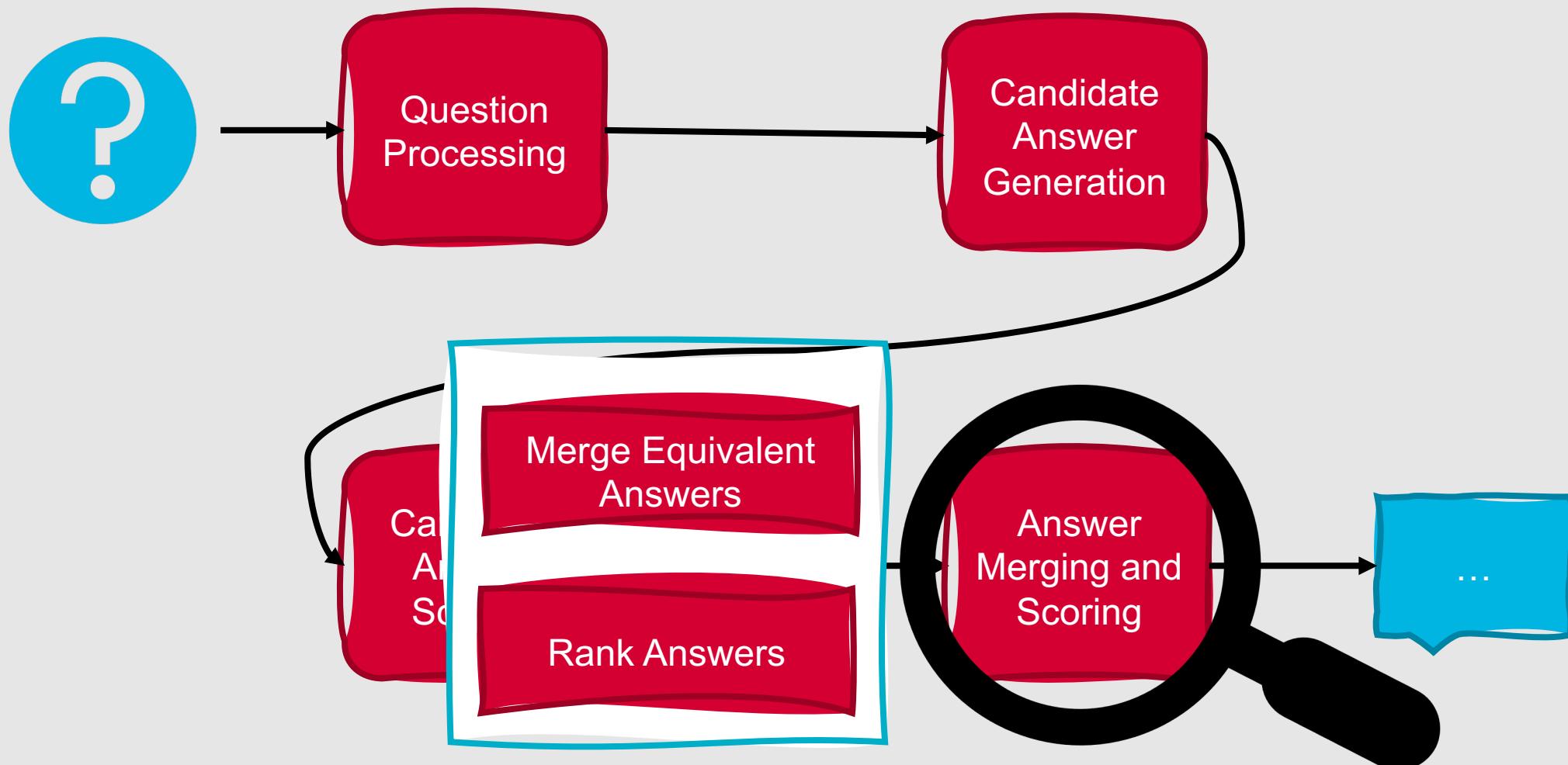
# Stage 3: Candidate Answer Scoring



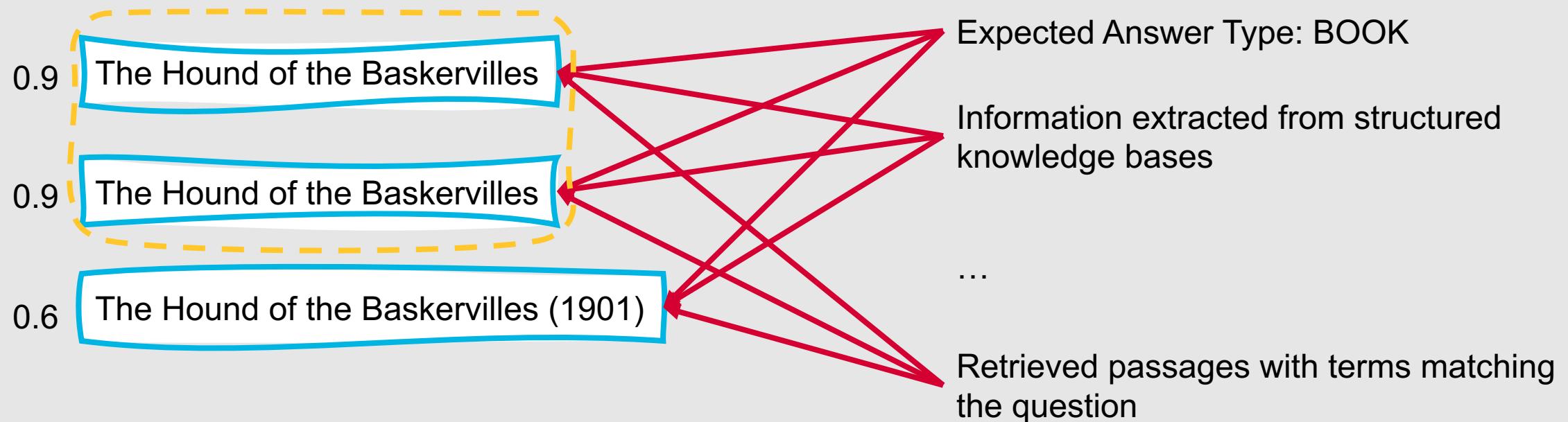
# Stage 3: Candidate Answer Scoring



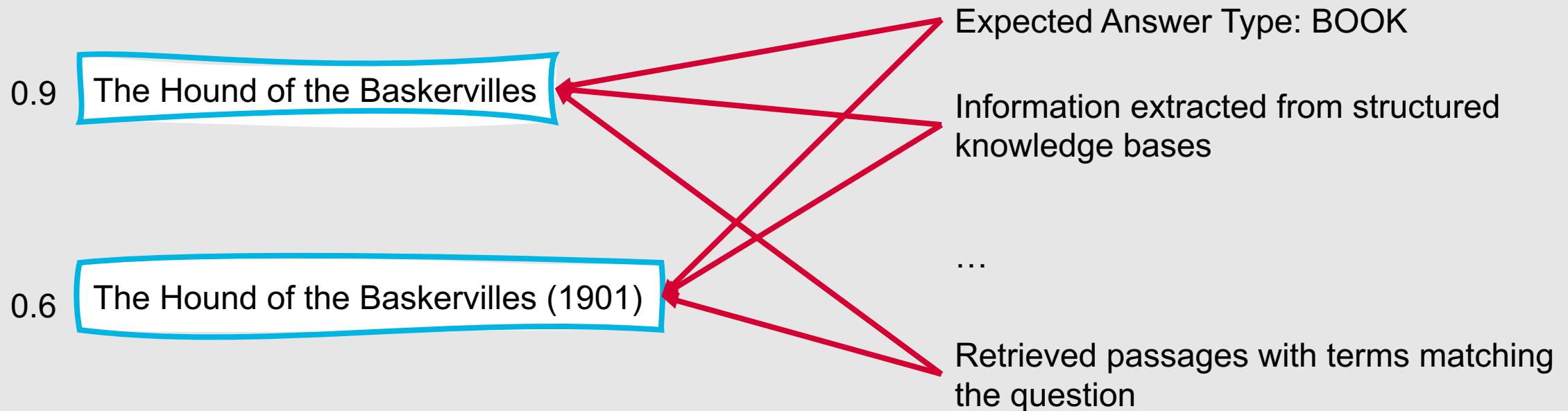
# Stage 4: Answer Merging and Scoring



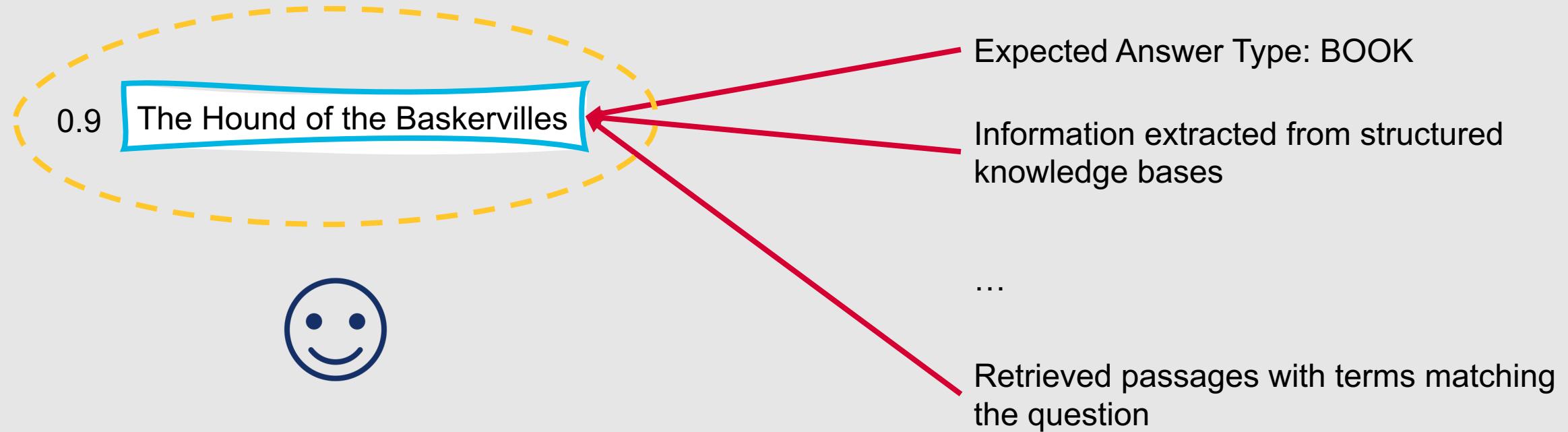
# Stage 4: Answer Merging and Scoring



# Stage 4: Answer Merging and Scoring



# Stage 4: Answer Merging and Scoring



# Where are we today?

- Moving towards **language model-based question answering**
  - 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

PA

How do question answering systems work today?



Today's question answering systems typically use a combination of natural language processing (NLP) and machine learning (ML) techniques to understand the user's question and generate an answer. The process usually involves the following steps:

1. Question understanding: The system converts the text of the user's question into a machine-readable format and analyzes it to identify the intent and extract relevant information.
2. Information retrieval: The system searches a pre-defined knowledge source (such as a database, a web page, or a large corpus of text) to find information that may answer the user's question.
3. Answer generation: The system selects the most relevant information from the retrieved data and generates an answer. In some systems, this may involve additional NLP and ML techniques, such as summarization or text generation, to produce a concise and coherent answer.
4. Answer ranking: The system uses various methods, such as word overlaps, word embeddings, or neural networks, to rank the answers and select the most likely candidate.

Overall, the accuracy and effectiveness of today's question answering systems depend on the quality and scope of their training data, the sophistication of their NLP and ML models, and the ability to handle complex and ambiguous questions.



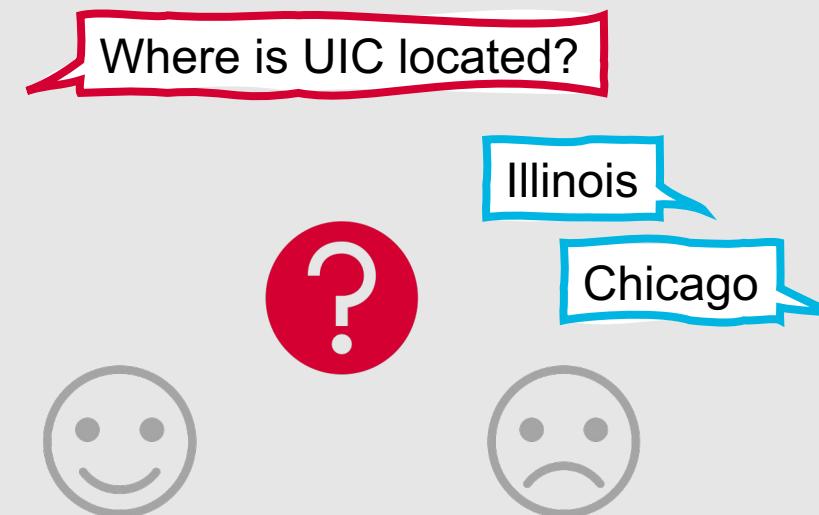


## Language Model-based Question Answering

- **Advantages:**
  - Simple approach
  - Decent performance
- **Disadvantages:**
  - Often lower accuracy in answers
  - Poor interpretability

# 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 → 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

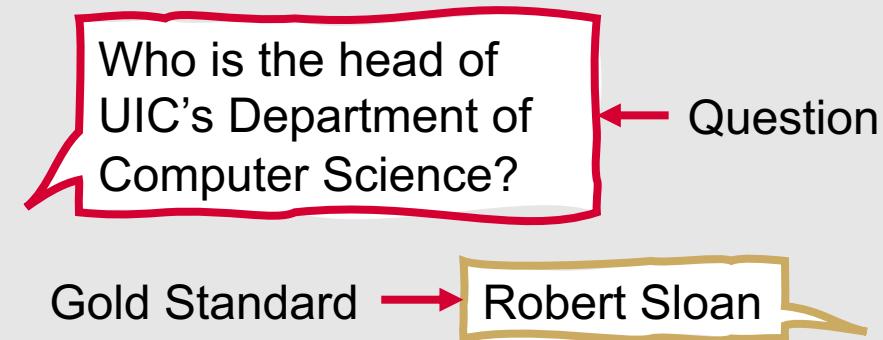
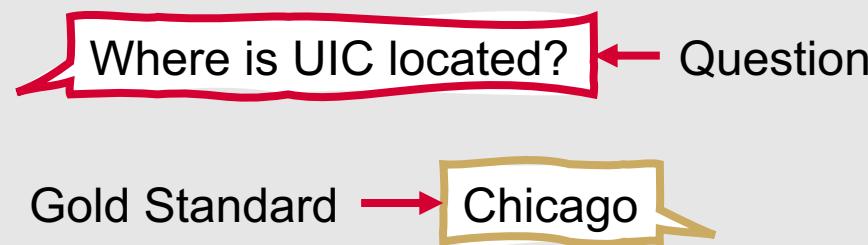
Where is UIC located? ← Question

Gold Standard → Chicago

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

$$\text{Reciprocal Rank} = 1/3$$

# Mean Reciprocal Rank

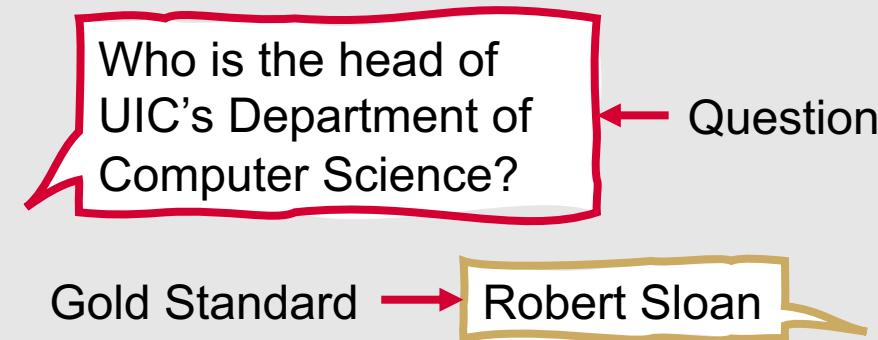
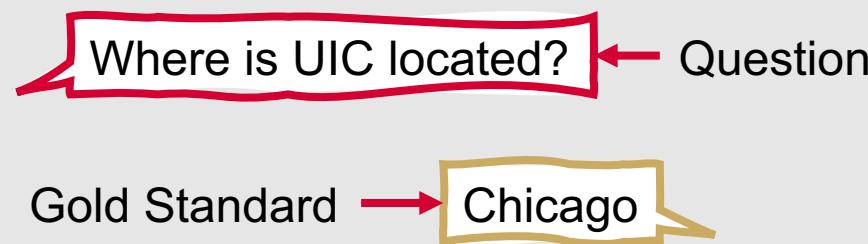


Prediction	Rank
Illinois	1
West Loop	2
Chicago	3
Little Italy	4

$$\text{Reciprocal Rank} = 1/3$$

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

# Mean Reciprocal Rank



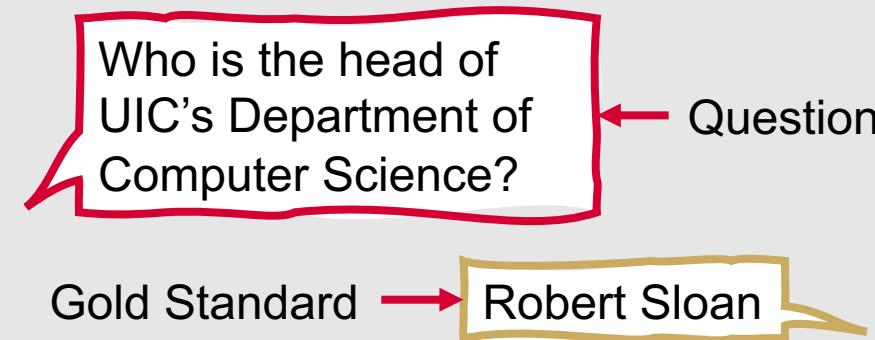
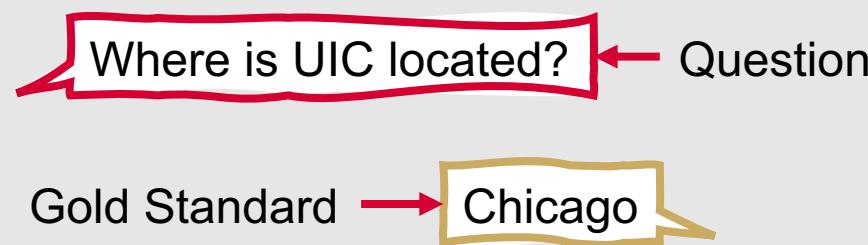
Prediction	Rank
Illinois	1
West Loop	2
Chicago	3
Little Italy	4

Reciprocal  
Rank = 1/3

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

Reciprocal  
Rank = 1/2

# Mean Reciprocal Rank



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

$$\text{Reciprocal Rank} = 1/3$$

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

$$\text{Reciprocal Rank} = 1/2$$

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

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

# Computing $F_1$ for Question Answering Systems

Where is UIC located? ← Question

Gold Standard → Chicago

Prediction → Chicago, Illinois

	Actual True	Actual False
Predicted True		
Predicted False		

# 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	

# 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 and Evaluating MT Systems

- MT systems are commonly evaluated using both **human ratings** and **automated metrics**
- Popular automated metrics include **BLEU**, **chrF**, and **embedding-based measures**
- **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**