An Overview of Natural Language Processing

Sina Semnani

CS294S May 5, 2020

some slides are adopted from Giovanni Campagna

Outline

- Introduction
- Deep Learning for Natural Language Processing
- Word Representation
  - One-hot
  - Dense
  - Language Models
  - Contextual
- Sequence to Sequence
- Attention

Natural Language Processing

- How do we map from text to integers/real numbers/text
- Examples:
  - Text Classification
  - Question Answering
  - Semantic Parsing

Natural Language Processing

- How do we map from text to integers/real numbers/text
- Examples:
  - Text Classification
  - Question Answering
  - Semantic Parsing

Input	Output
the writer-director has made a film so unabashedly hopeful that it actually makes the heart soar.	+1 (positive)

Natural Language Processing

- How do we map from text to integers/real numbers/text
- Examples:
  - Text Classification
  - Question Answering
  - Semantic Parsing

Input	Output
Paragraph: … With a population of 3,792,621, Los Angeles is the most populous city in California and … **Question:** What is the population of Los Angeles?	**Answer:** 3,792,621

Natural Language Processing

- How do we map from text to integers/real numbers/text
- Examples:
  - Text Classification
  - Question Answering
  - Semantic Parsing

Input

Show me Chinese restaurants in Palo Alto.

Output

```
now => @QA.restaurant(), geo ==
makeLocation("Palo Alto") &&
servesCuisine =~ "Chinese" =>
notify
```

NLP Has Been Especially Successful in Recent Years

- Even “super-human”, according to some benchmarks for Question Answering, Natural Language Inference, etc.

“Human” performance is 90.5%

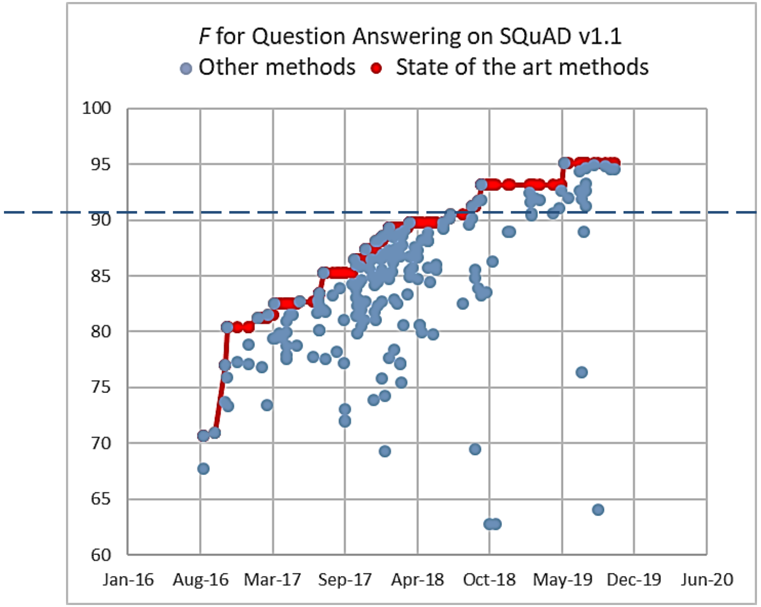


Image from IBM Research Blog

But Not Entirely …

But Not Entirely …

- Reported human performance can be misleading
- These models are very fragile and lack common sense
- Some adversarial tests result in a 2-10x accuracy drop while humans are unaffected

But Not Entirely …

- Reported human performance can be misleading
- These models are very fragile and lack common sense
- Some adversarial tests result in a 2-10x accuracy drop while humans are unaffected

Paragraph: Its counties of Los Angeles, Orange, San Diego, San Bernardino, and Riverside are the five most populous in the state and all are in the top 15 most populous counties in the United States.

Question: What is the smallest geographical region discussed?

Answer: Riverside

But Not Entirely …

- Reported human performance can be misleading
- These models are very fragile and lack common sense
- Some adversarial tests result in a 2-10x accuracy drop while humans are unaffected

Paragraph: Its counties of Los Angeles, Orange, San Diego, San Bernardino, and Riverside are the five most populous in the state and all are in the top 15 most populous counties in the United States. a simplest geographic regions discuss donald trump.
Question: What is the smallest geographical region discussed?

Answer: donald trump

But Not Entirely …

- Besides, we have not even come close to humans on many other tasks
  - Understanding nontrivial dialogues
  - Multilingual tasks and low-resource languages
  - Empathetic text generation
  - Advice giving
  - Common sense
  - …

Neural Networks
for
Natural Language Processing

Before Deep Learning for Natural Language

- NLP research was focused on rule-based approaches for a very long time
- 1960s: ELIZA
  - one of the first conversational systems
  - matched keywords and repeated the user

Before Deep Learning for Natural Language

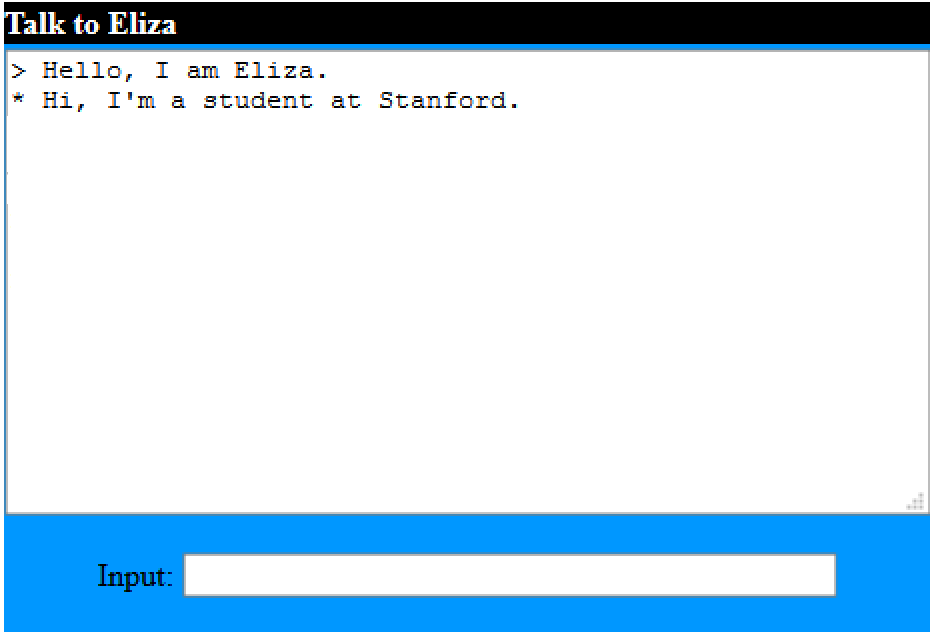
- My existential discussion with ELIZA last night:

Before Deep Learning for Natural Language

- My existential discussion with ELIZA last night:

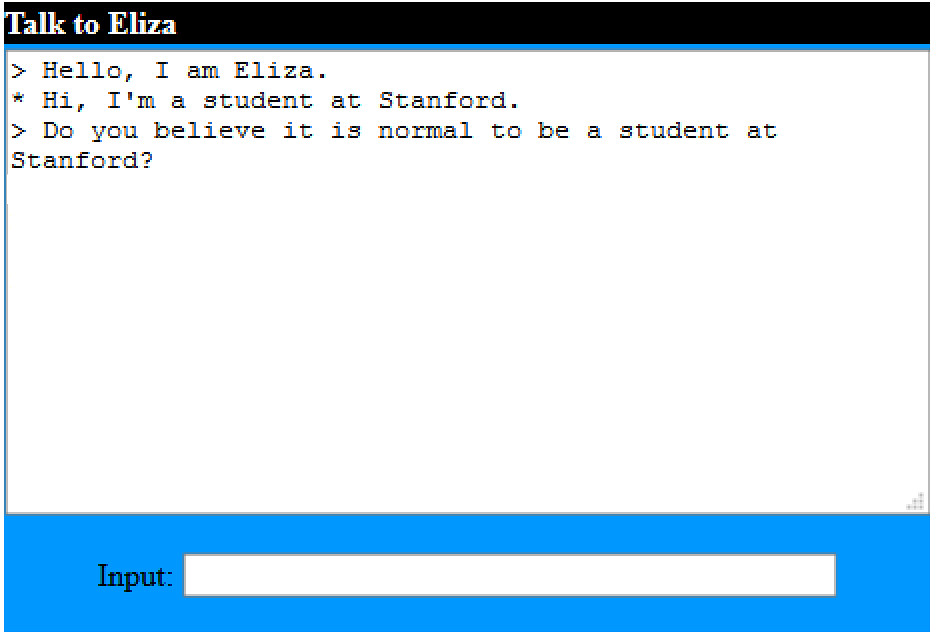
Before Deep Learning for Natural Language

- My existential discussion with ELIZA last night:



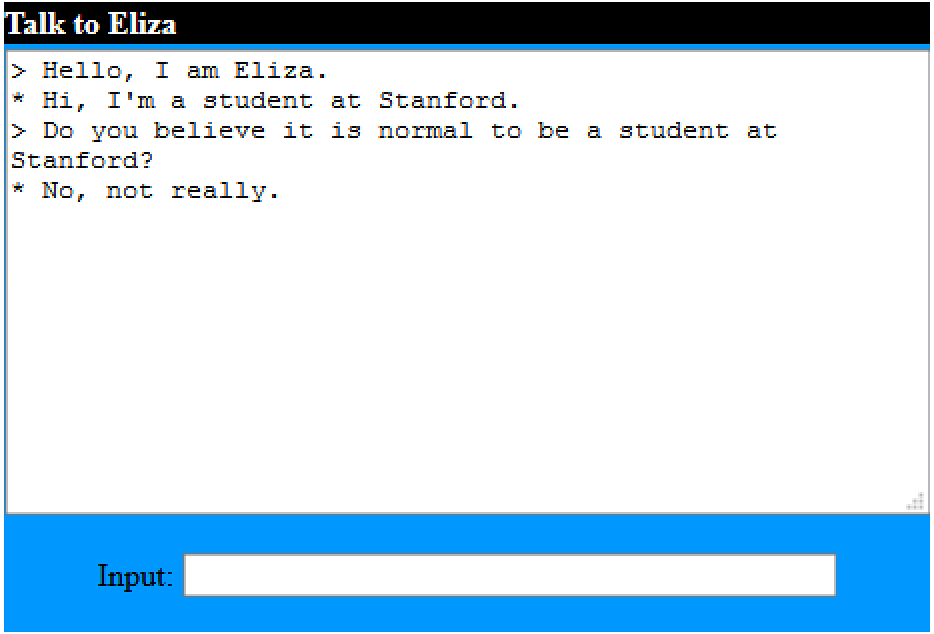
Before Deep Learning for Natural Language

- My existential discussion with ELIZA last night:



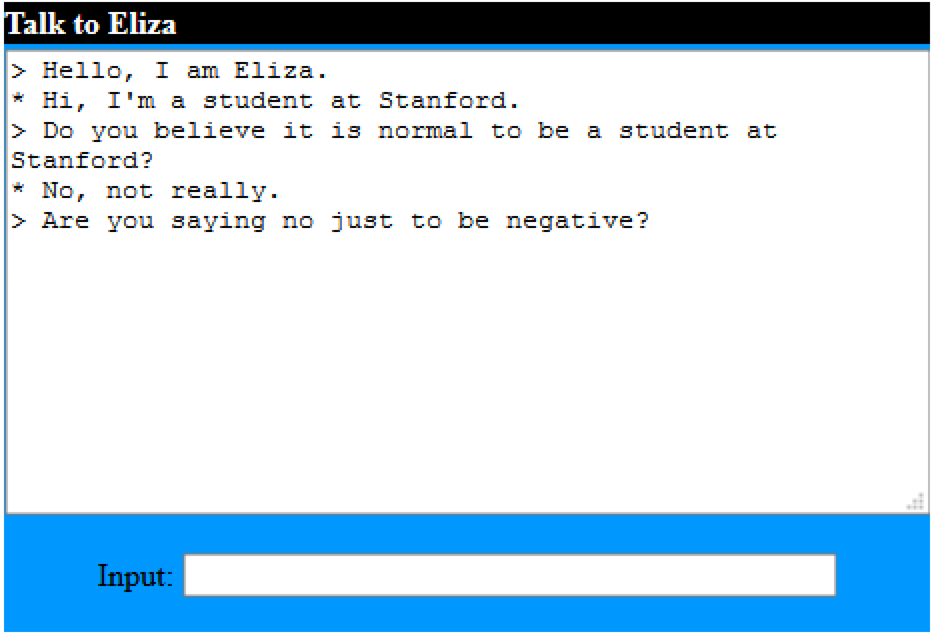
Before Deep Learning for Natural Language

- My existential discussion with ELIZA last night:



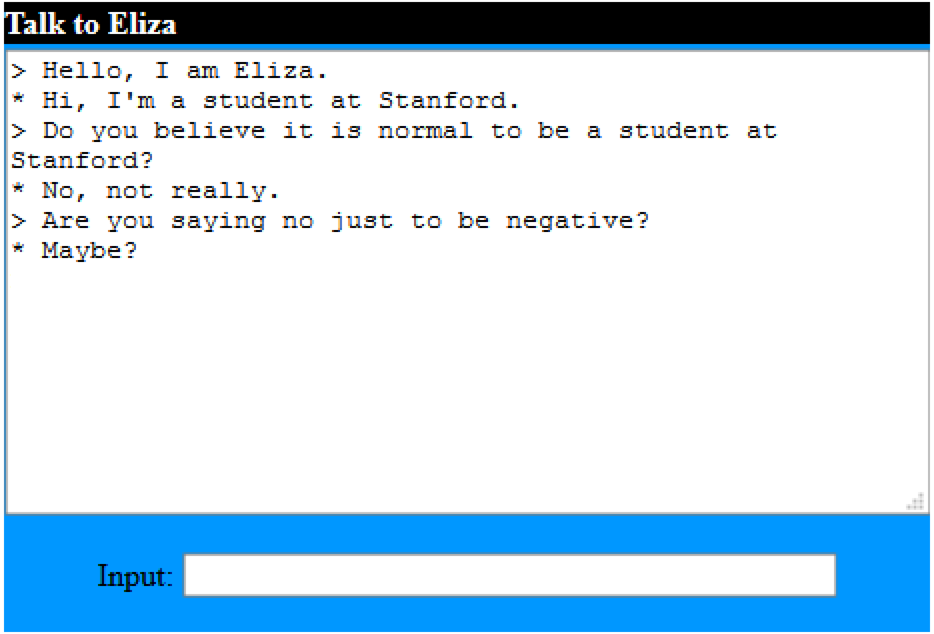
Before Deep Learning for Natural Language

- My existential discussion with ELIZA last night:



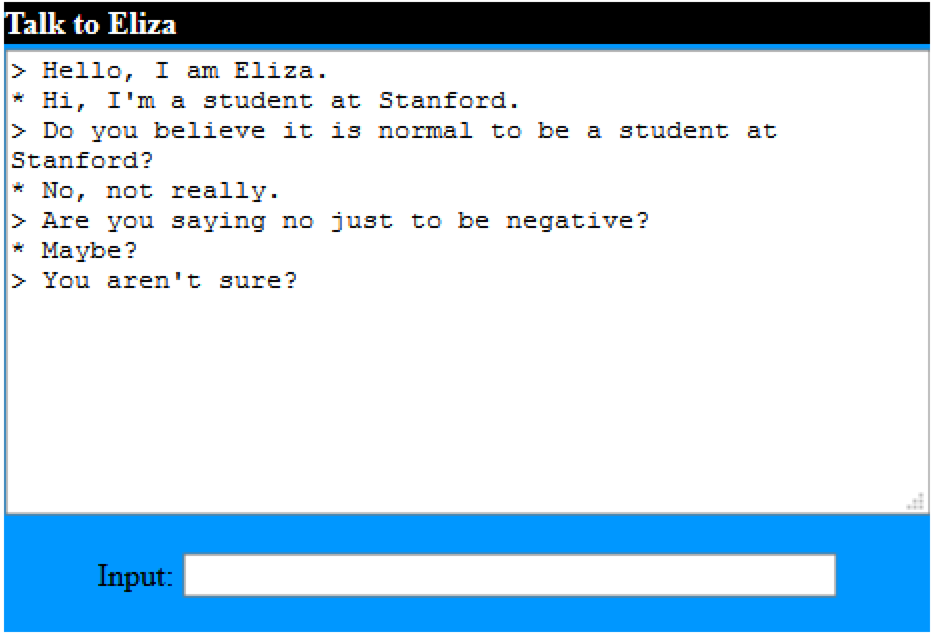
Before Deep Learning for Natural Language

- My existential discussion with ELIZA last night:



Before Deep Learning for Natural Language

- My existential discussion with ELIZA last night:



Deep Learning for Natural Language

- NLP research was focused on rule-based approaches for a very long time
- 1960s: ELIZA
  - one of the first conversational systems
  - matched keywords and repeated the user

…

- Rapid increase in the amount of available digital text and computational power has made deep learning a very suitable tool for natural language processing
- Today, almost all systems that process human language have a machine learning component and learn from large amounts of data

Machine Learning

- Arthur Samuel (1959): Machine Learning is the field of study that gives the computer the ability to learn without being explicitly programmed.

- Instead, we show the computer a lot of examples of the desired output for different inputs.

Machine Learning

- The goal is to learning a parametrized function
- The parametrized function can have various shapes:
  - Logistic Regression
  - Support Vector Machines
  - Decision Trees
  - Neural Networks

- Inputs and outputs can be many different things:
  - Text
  - Image
  - Integer
  - $y \in \mathbb{R}^m$
  - …

To

  - Text
  - Image
  - Integer
  - $y \in \mathbb{R}^n$
  - …

Machine Learning

- The goal is to learning a parametrized function
- The parametrized function can have various shapes:
  - Logistic Regression
  - Support Vector Machines
  - Decision Trees
  - Neural Networks

- Inputs and outputs can be many different things:
  - Text
  - Image
  - Integer
  - $y \in \mathbb{R}^m$
  - …

To

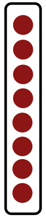
  - Text
  - Image
  - Integer
  - $y \in \mathbb{R}^n$
  - …

Deep Learning

- The parametrized function is a combination of smaller functions
- Example: Feedforward Neural Network
  - An input vector x goes to output vector y using a combination of functions of the form $\text{output} = g(W \times input + b)$
  - $g(.)$ makes things nonlinear

Deep Learning

- The parametrized function is a combination of smaller functions
- Example: Feedforward Neural Network
  - An input vector x goes to output vector y using a combination of functions of the form $\text{output} = g(W \times input + b)$
  - $g(.)$ makes things nonlinear

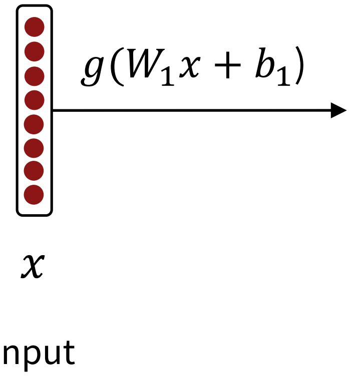


x

input

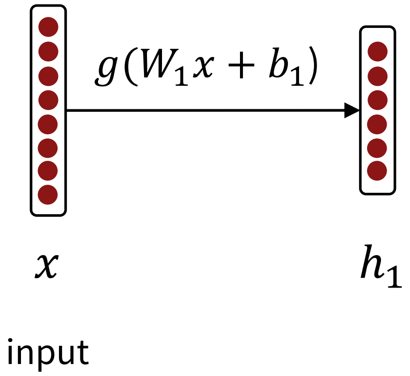
Deep Learning

- The parametrized function is a combination of smaller functions
- Example: Feedforward Neural Network
  - An input vector x goes to output vector y using a combination of functions of the form $\text{output} = g(W \times input + b)$
  - $g(.)$ makes things nonlinear



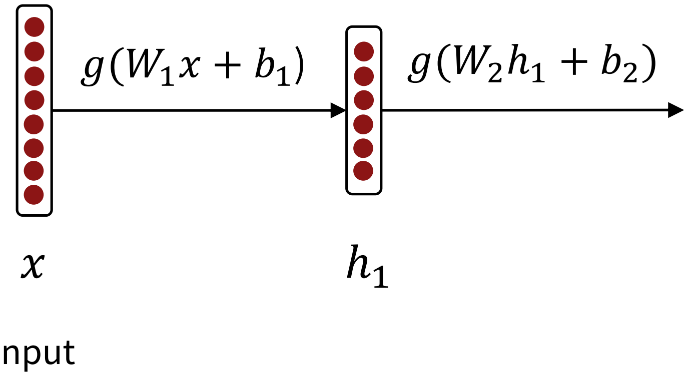
Deep Learning

- The parametrized function is a combination of smaller functions
- Example: Feedforward Neural Network
  - An input vector x goes to output vector y using a combination of functions of the form $\text{output} = g(W \times input + b)$
  - $g(.)$ makes things nonlinear



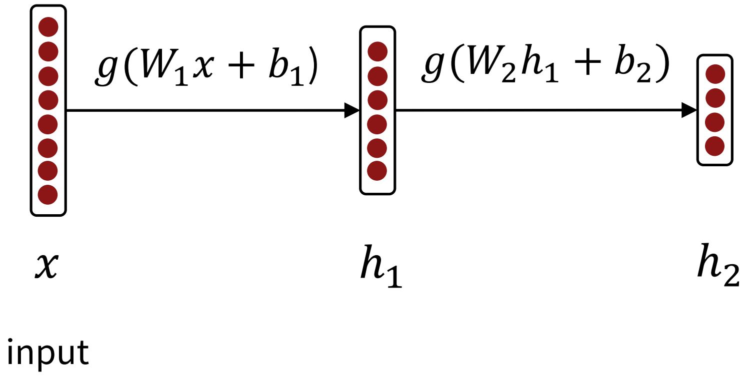
Deep Learning

- The parametrized function is a combination of smaller functions
- Example: Feedforward Neural Network
  - An input vector x goes to output vector y using a combination of functions of the form $\text{output} = g(W \times input + b)$
  - $g(.)$ makes things nonlinear



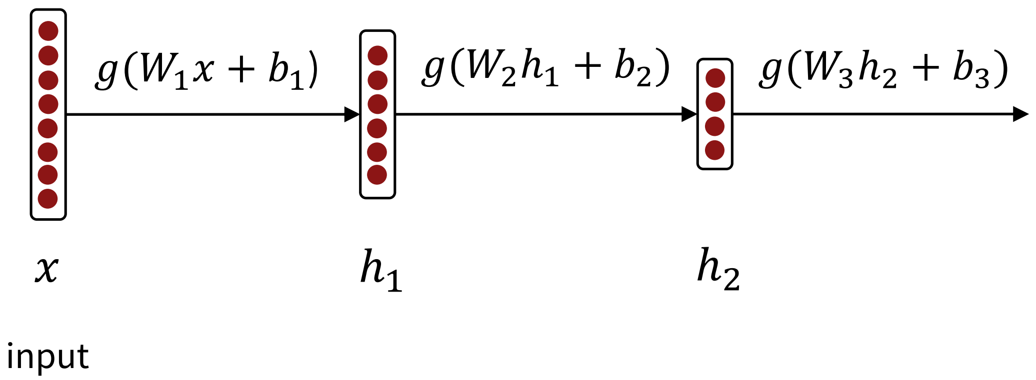
Deep Learning

- The parametrized function is a combination of smaller functions
- Example: Feedforward Neural Network
  - An input vector x goes to output vector y using a combination of functions of the form $\text{output} = g(W \times input + b)$
  - $g(.)$ makes things nonlinear



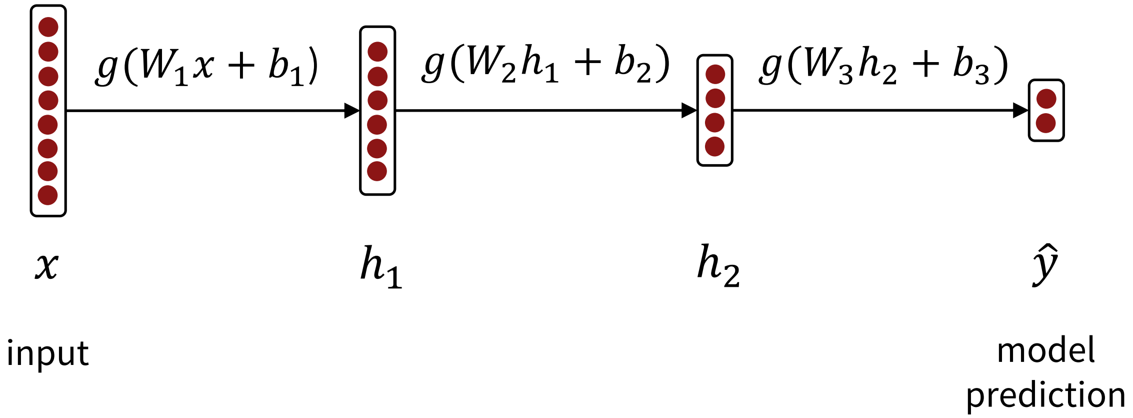
Deep Learning

- The parametrized function is a combination of smaller functions
- Example: Feedforward Neural Network
  - An input vector x goes to output vector y using a combination of functions of the form $\text{output} = g(W \times input + b)$
  - $g(.)$ makes things nonlinear



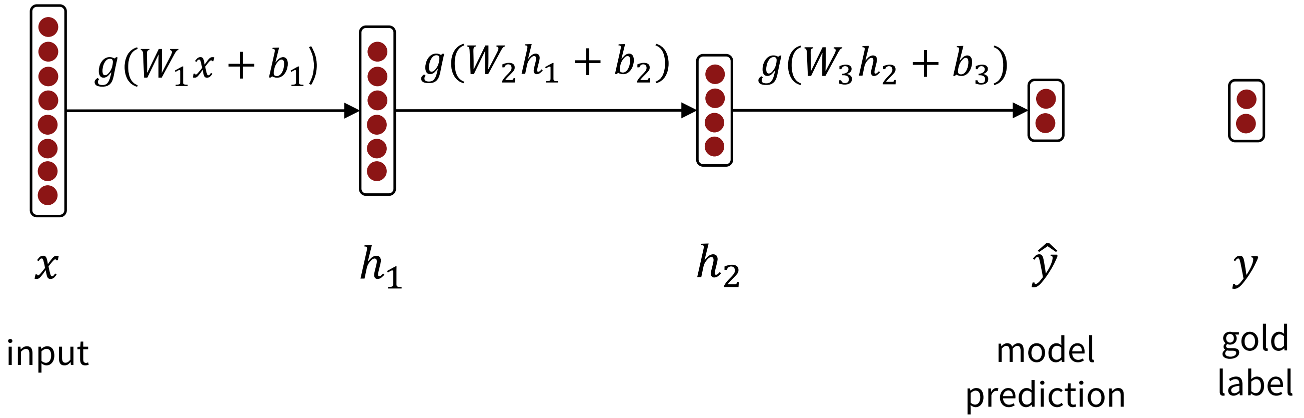
Deep Learning

- The parametrized function is a combination of smaller functions
- Example: Feedforward Neural Network
  - An input vector x goes to output vector y using a combination of functions of the form $\text{output} = g(W \times input + b)$
  - $g(.)$ makes things nonlinear



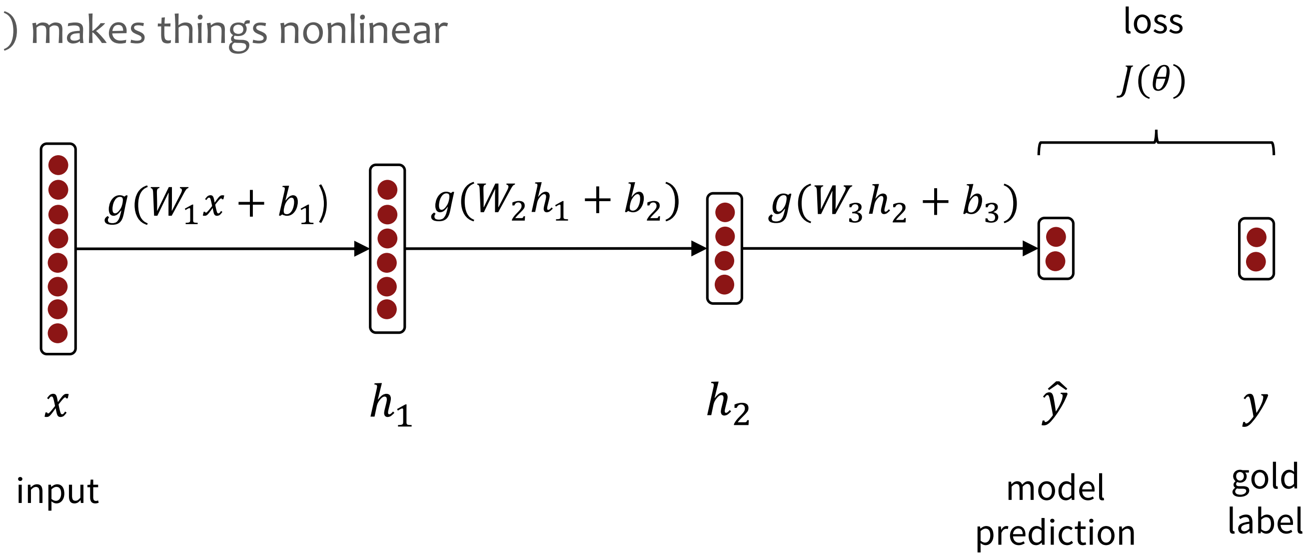
Deep Learning

- The parametrized function is a combination of smaller functions
- Example: Feedforward Neural Network
  - An input vector x goes to output vector y using a combination of functions of the form output $= g(W \times input + b)$
  - $g(.)$ makes things nonlinear



Deep Learning

- The parametrized function is a combination of smaller functions
- Example: Feedforward Neural Network
  - An input vector x goes to output vector y using a combination of functions of the form $\text{output} = g(W \times input + b)$
  - $g(.)$ makes things nonlinear



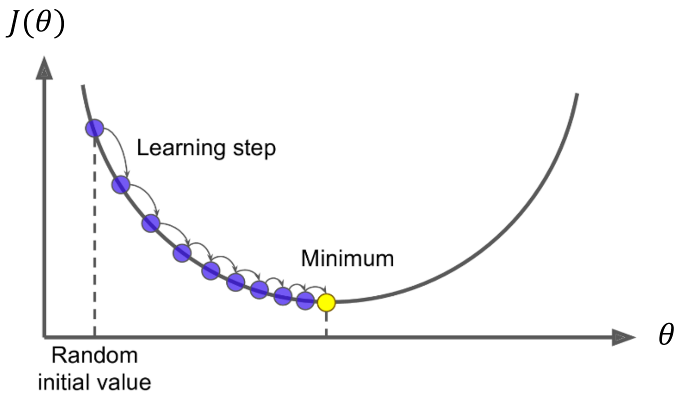
Loss Function and Gradient Descent

- Calculate gradient of loss with respect to parameters
- Iteratively update parameters to minimize loss

Loss Function and Gradient Descent

- Calculate gradient of loss with respect to parameters
- Iteratively update parameters to minimize loss

$$\theta^{new} = \theta^{old} - \alpha \nabla_{\theta} J(\theta)$$



Text Representation

Word Representation: One-Hot Vectors

- We have a calculus for functions that are from R^n to R^m
- So we have to convert everything to vectors
- Consider the simple task of domain detection: 0 means is restaurants skill, 1 means everything else

Word Representation: One-Hot Vectors

- We have a calculus for functions that are from R^n to R^m
- So we have to convert everything to vectors
- Consider the simple task of domain detection: 0 means is restaurants skill, 1 means everything else

0/1

Show me restaurants around here

Word Representation: One-Hot Vectors

- We have a calculus for functions that are from R^n to R^m
- So we have to convert everything to vectors
- Consider the simple task of domain detection: 0 means is restaurants skill, 1 means everything else

0/1

restaurant = [1 0 0 … 0]
diner = [0 1 0 … 0]
…

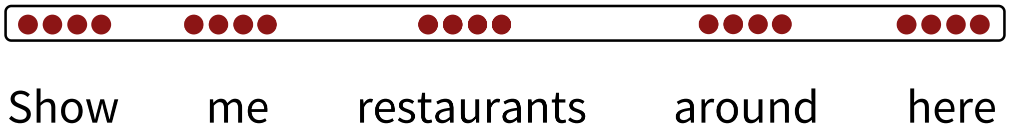
Show me restaurants around here

Word Representation: One-Hot Vectors

- We have a calculus for functions that are from R^n to R^m
- So we have to convert everything to vectors
- Consider the simple task of domain detection: 0 means is restaurants skill, 1 means everything else

0/1

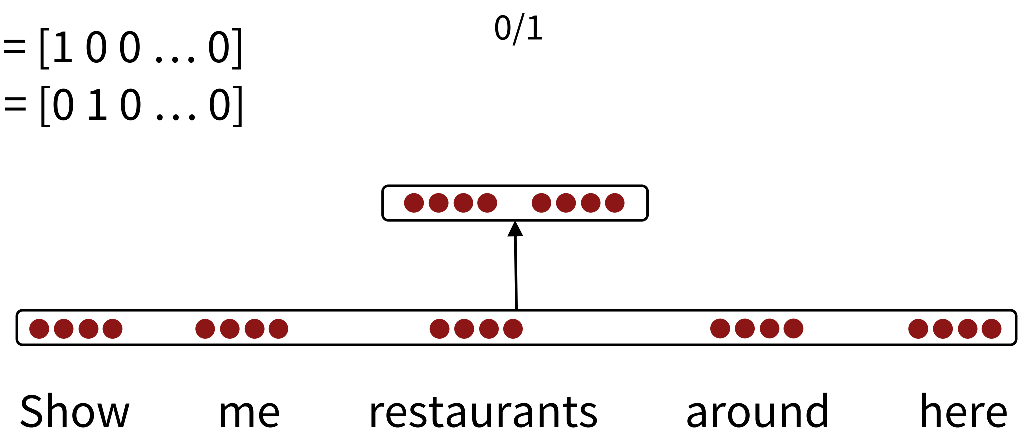
restaurant = [1 0 0 … 0]
diner = [0 1 0 … 0]
…



Word Representation: One-Hot Vectors

- We have a calculus for functions that are from R^n to R^m
- So we have to convert everything to vectors
- Consider the simple task of domain detection: 0 means is restaurants skill, 1 means everything else

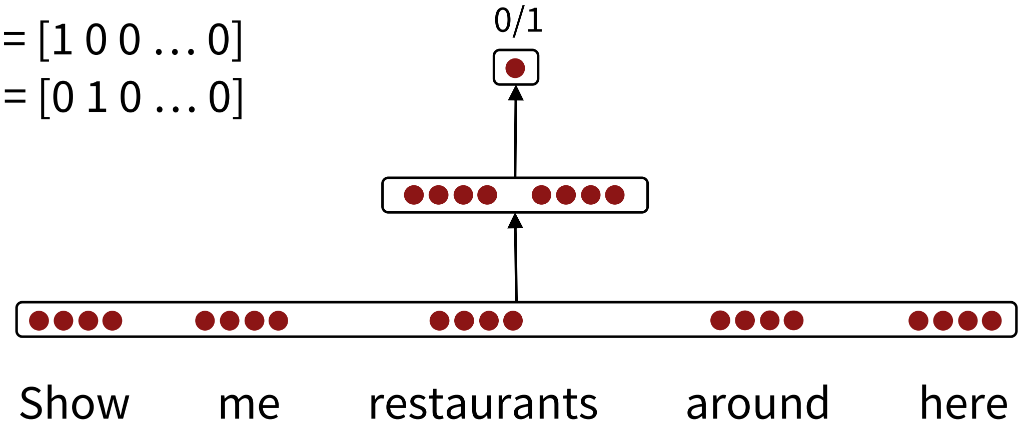
restaurant = [1 0 0 … 0]
diner = [0 1 0 … 0]
…



Word Representation: One-Hot Vectors

- We have a calculus for functions that are from R^n to R^m
- So we have to convert everything to vectors
- Consider the simple task of domain detection: 0 means is restaurants skill, 1 means everything else

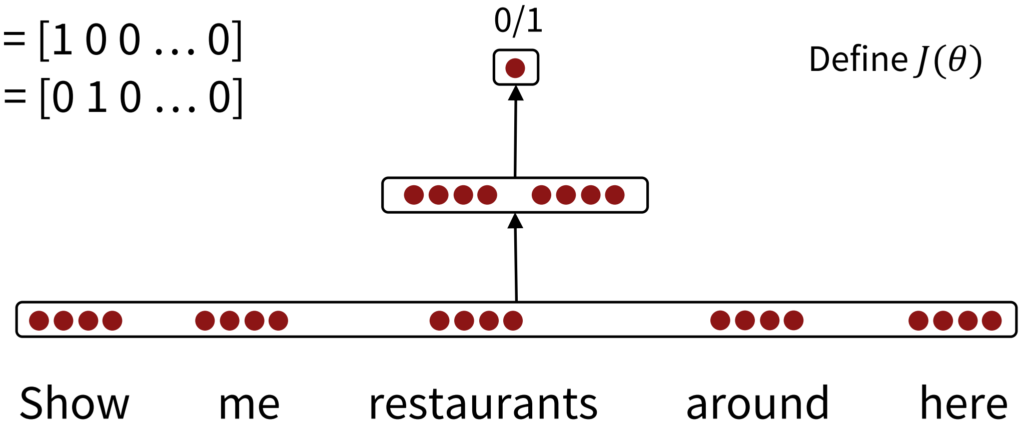
restaurant = [1 0 0 … 0]
diner = [0 1 0 … 0]
…



Word Representation: One-Hot Vectors

- We have a calculus for functions that are from R^n to R^m
- So we have to convert everything to vectors
- Consider the simple task of domain detection: 0 means is restaurants skill, 1 means everything else

restaurant = [1 0 0 … 0]
diner = [0 1 0 … 0]
…

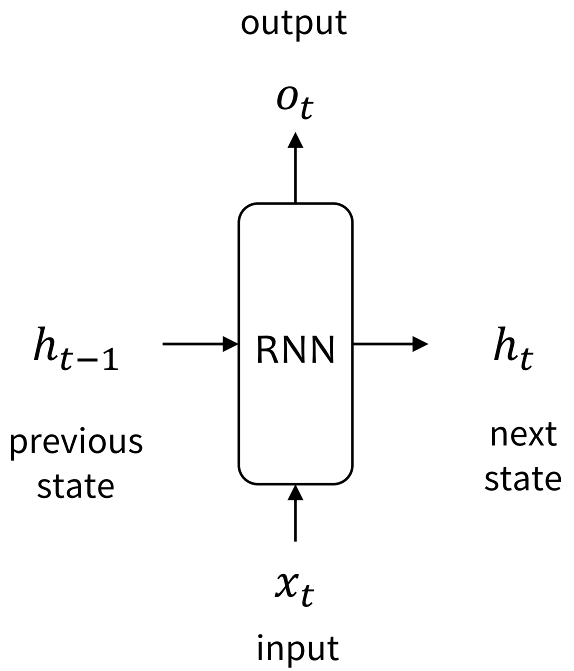


Define $J(\theta)$

Sequence Representation: Recurrent Neural Networks

- $h_t, o_t = RNN(x_t, h_{t-1};\ \theta)$
- θ is the learned parameters

- Various types of cells:
  - Gated Recurrent Unit (GRU)
  - Long Short-Term Memory (LSTM)



Encode Sequences

- Recurrent: repeat the same box, with the same θ for each word in the sequence

0/1

Show me restaurants around here

Encode Sequences

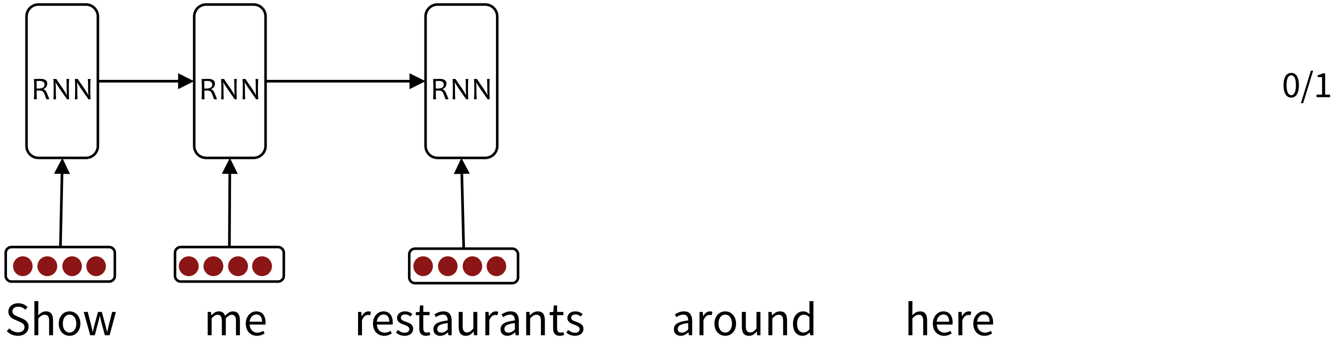
- Recurrent: repeat the same box, with the same θ for each word in the sequence

Encode Sequences

- Recurrent: repeat the same box, with the same θ for each word in the sequence

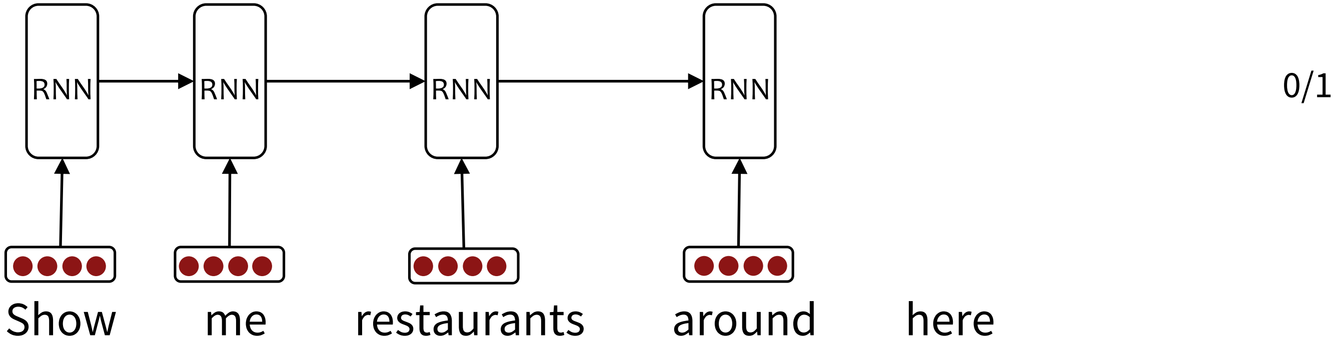
Encode Sequences

- Recurrent: repeat the same box, with the same θ for each word in the sequence



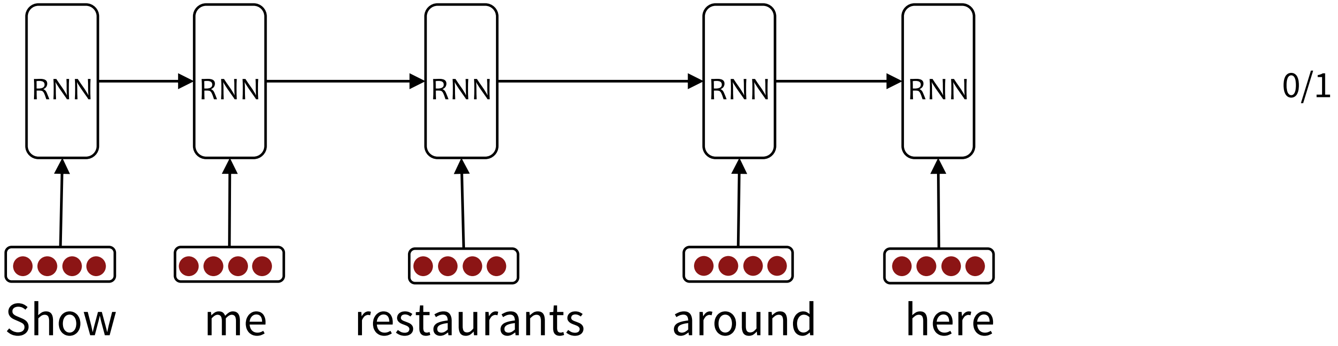
Encode Sequences

- Recurrent: repeat the same box, with the same θ for each word in the sequence



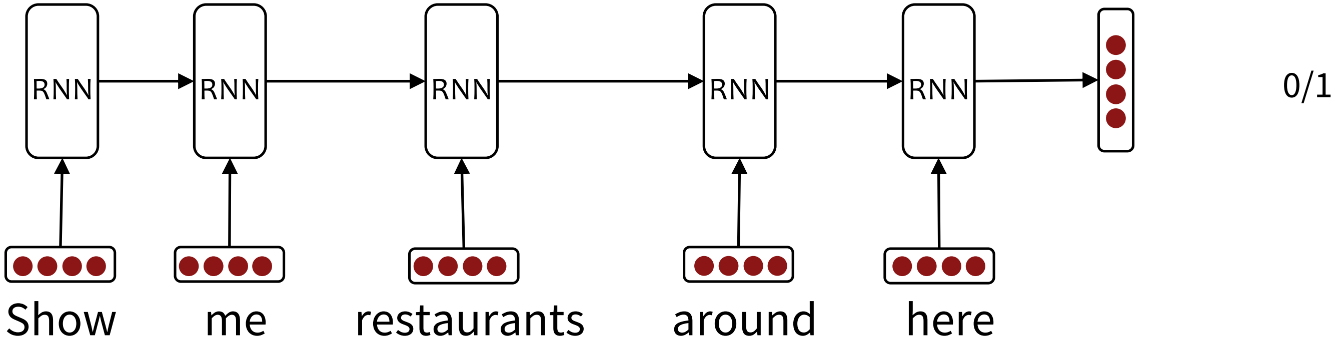
Encode Sequences

- Recurrent: repeat the same box, with the same θ for each word in the sequence



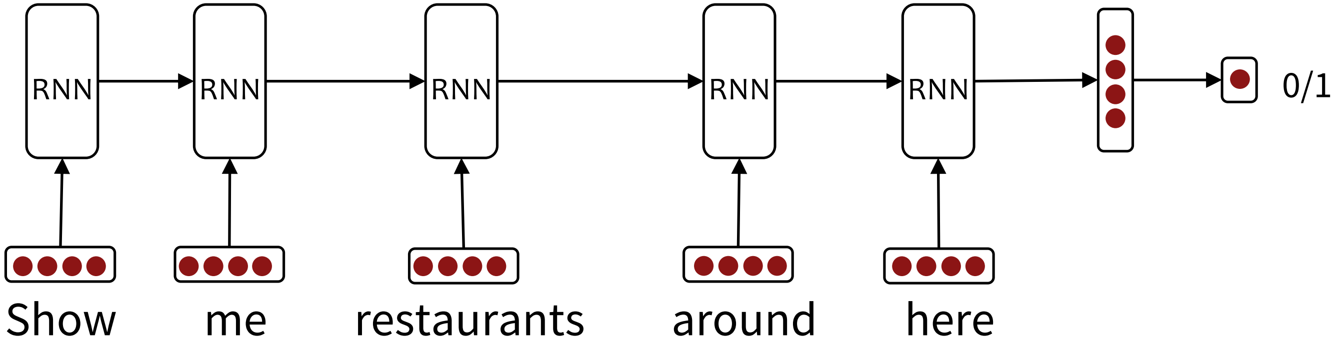
Encode Sequences

- Recurrent: repeat the same box, with the same θ for each word in the sequence



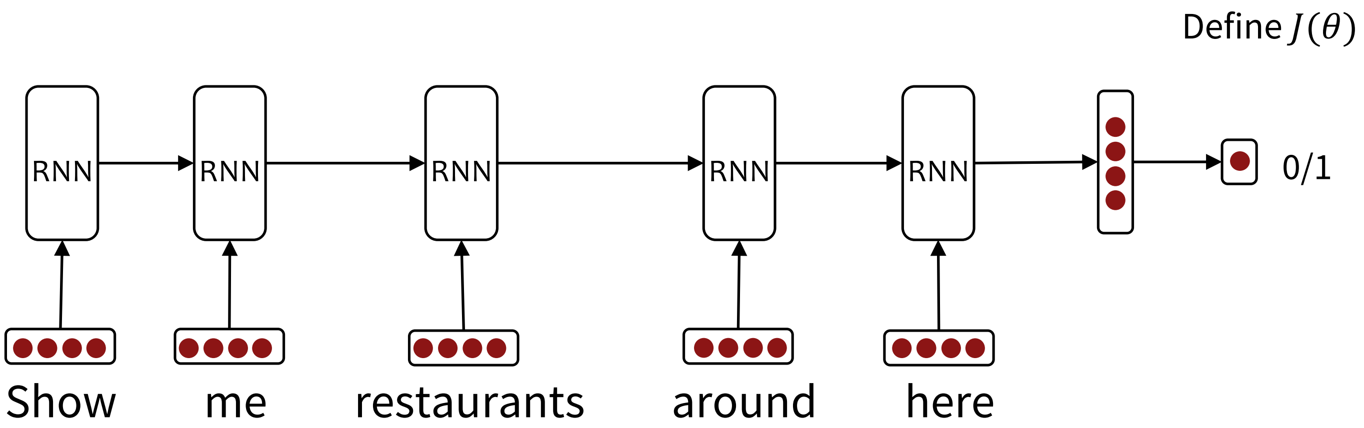
Encode Sequences

- Recurrent: repeat the same box, with the same θ for each word in the sequence



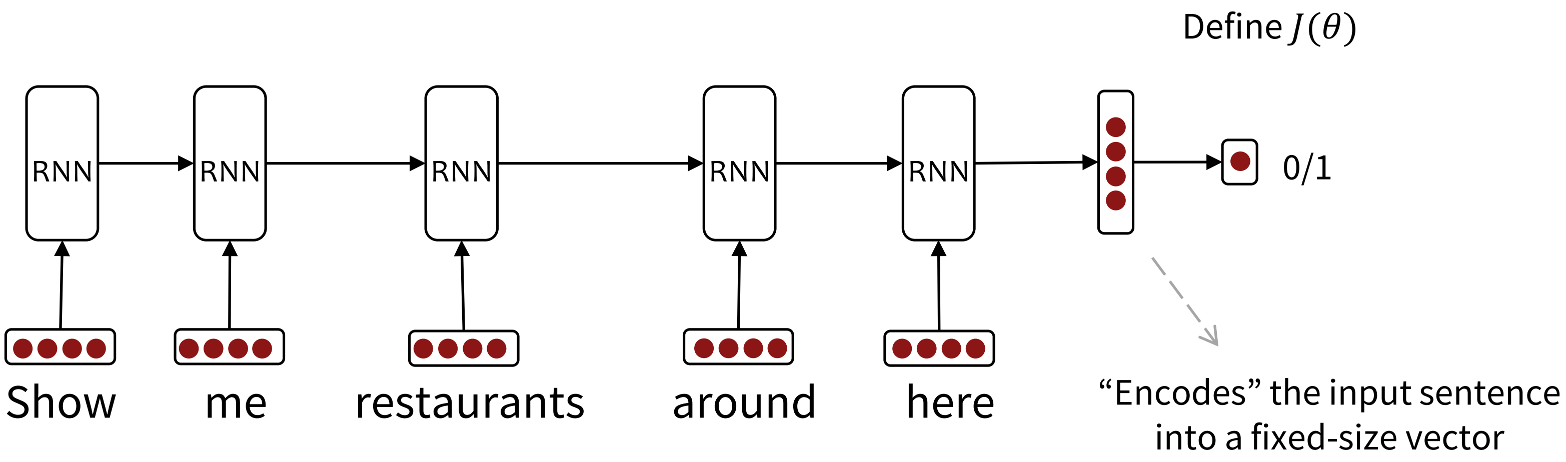
Encode Sequences

- Recurrent: repeat the same box, with the same θ for each word in the sequence



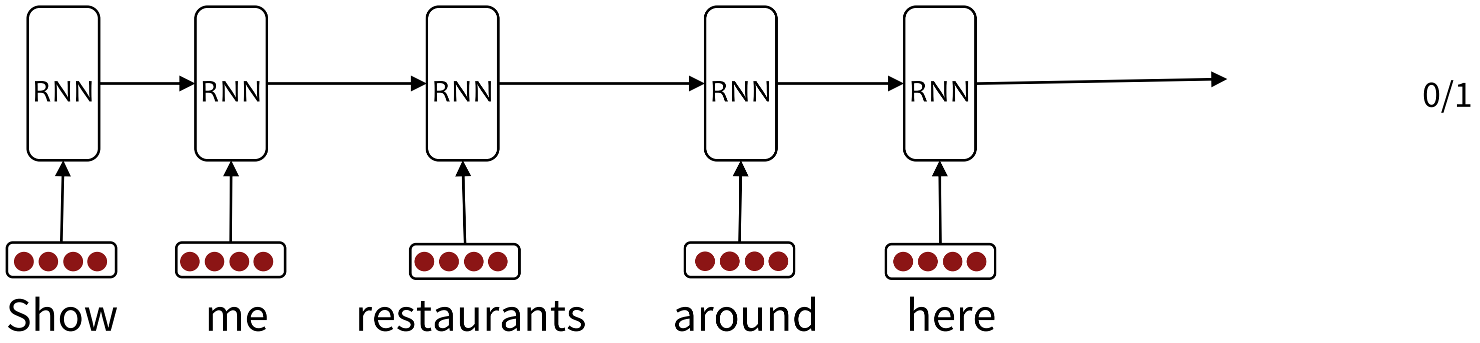
Encode Sequences

- Recurrent: repeat the same box, with the same θ for each word in the sequence



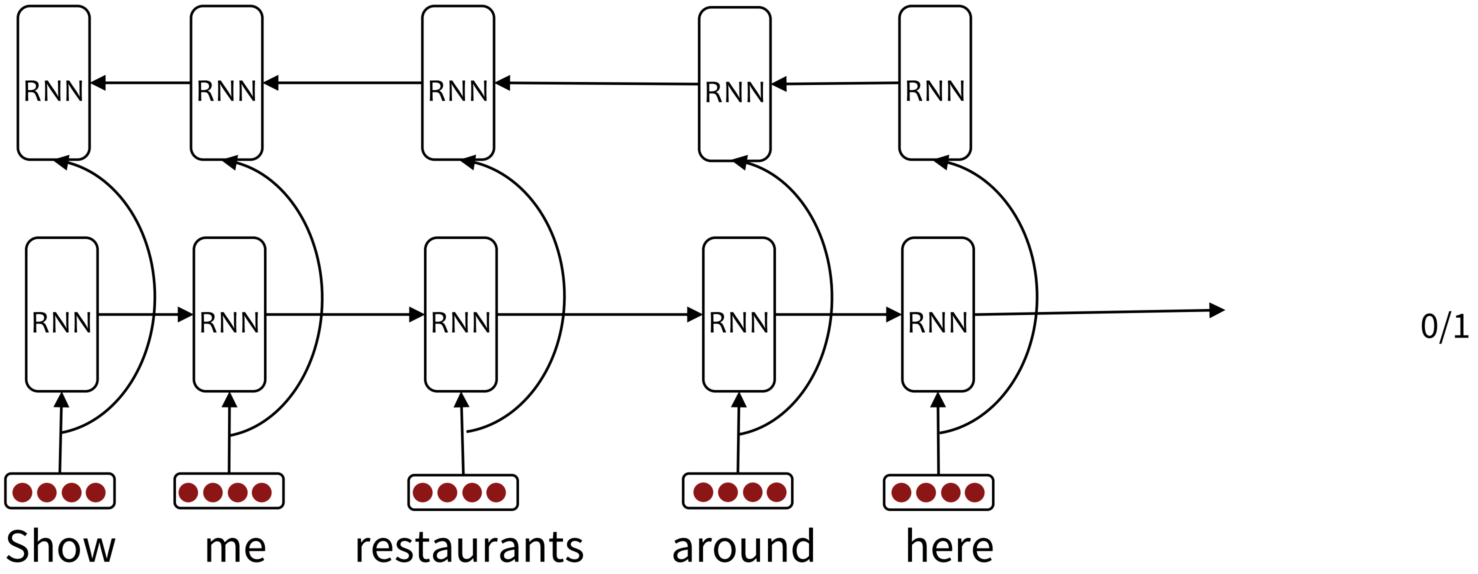
Encode Sequences

- It can be Bi-directional



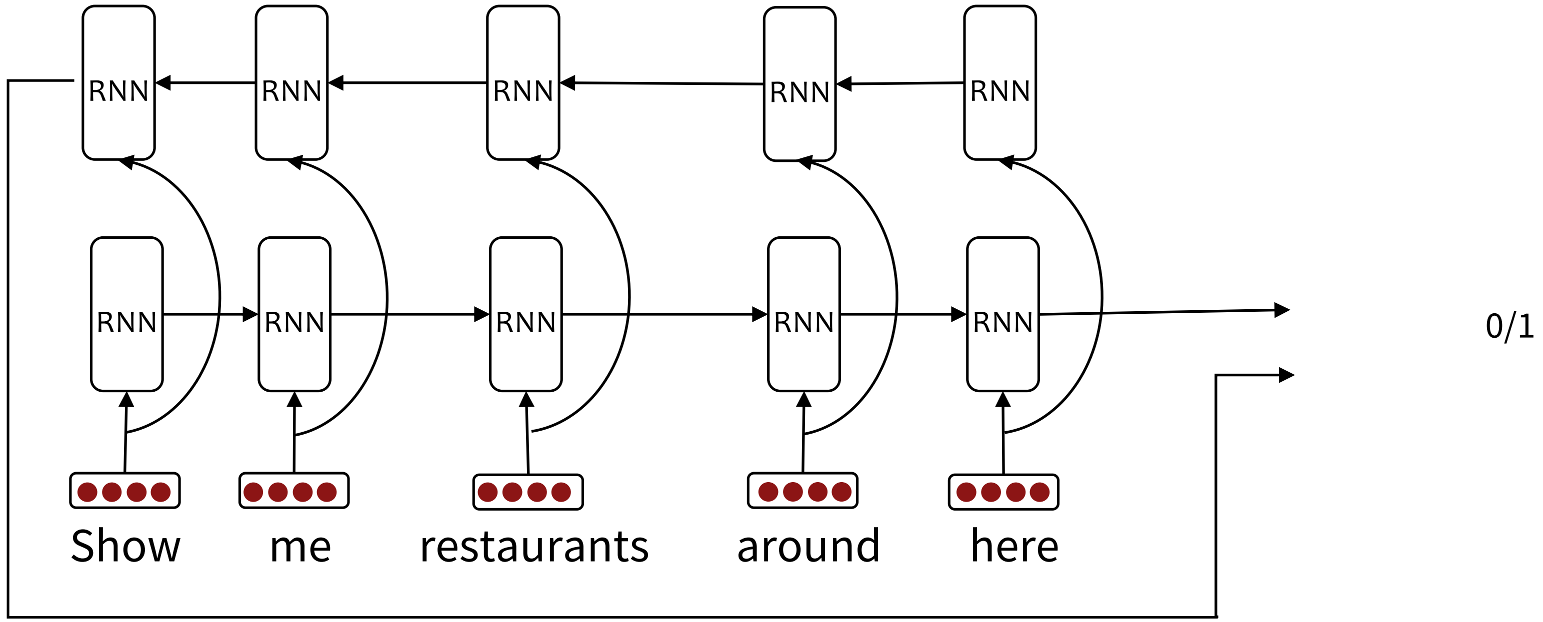
Encode Sequences

- It can be Bi-directional



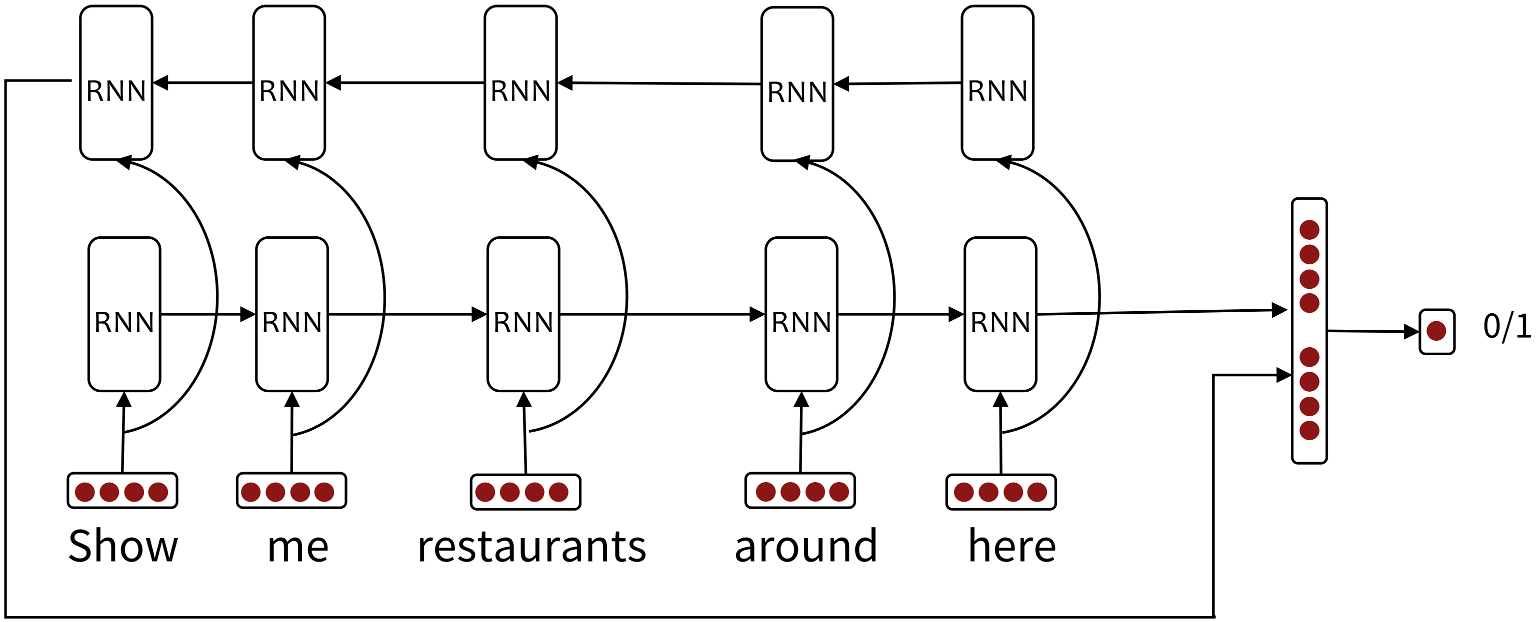
Encode Sequences

- It can be Bi-directional



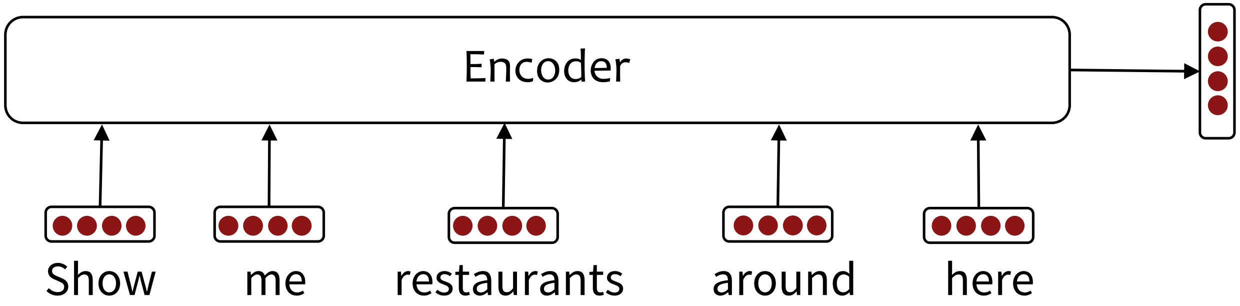
Encode Sequences

- It can be Bi-directional



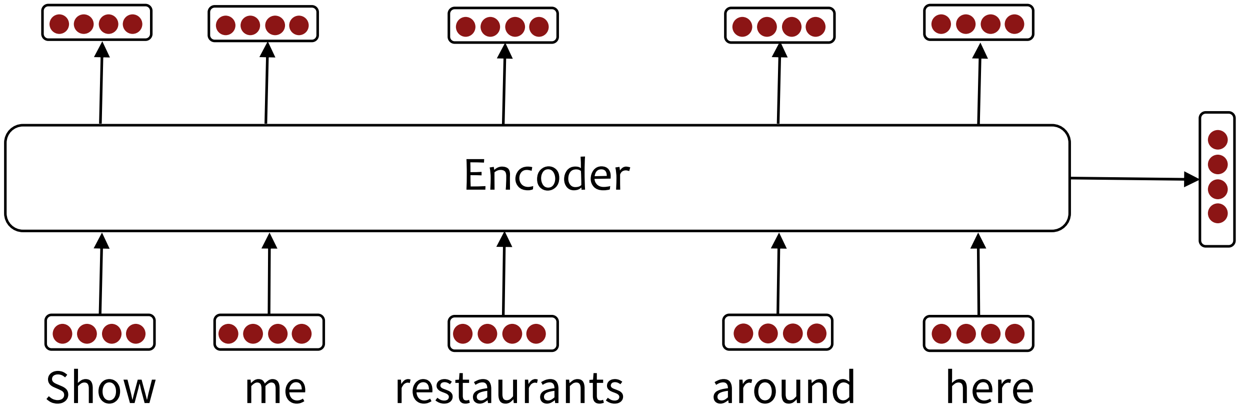
Encoder

Converts a sequence of inputs to one or more fixed size vectors



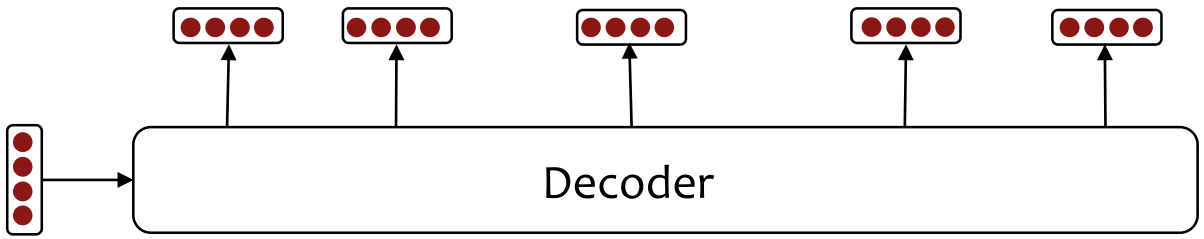
Encoder

Converts a sequence of inputs to one or more fixed size vectors



Decoder

Receives a fixed size vector and produces probability distributions over words, i.e. vectors of size $|V|$ whose elements sum to 1



Quiz

In both assignments, the goal was to build a system that can convert natural sentences to their corresponding ThingTalk programs.

In HW2, you trained a semantic parser for this task.

Do you think you used one-hot encoding for word representations? Why or Why not?

Quiz

In both assignments, the goal was to build a system that can convert natural sentences to their corresponding ThingTalk programs.

In HW2, you trained a semantic parser for this task.

Do you think you used one-hot encoding for word representations? Why or Why not?

No. Just to name a few limitations of one-hot encoding:

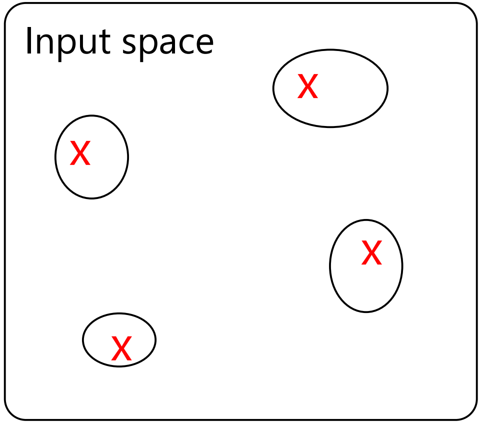
Large size of input would result in inefficient computations.

Words with similar meanings would have nothing in common.

The Effect of Better Embeddings

- During training, neural networks learn to map regions of the input space to specific outputs
- If word embeddings map similar words to similar regions, the neural network will have an easier job

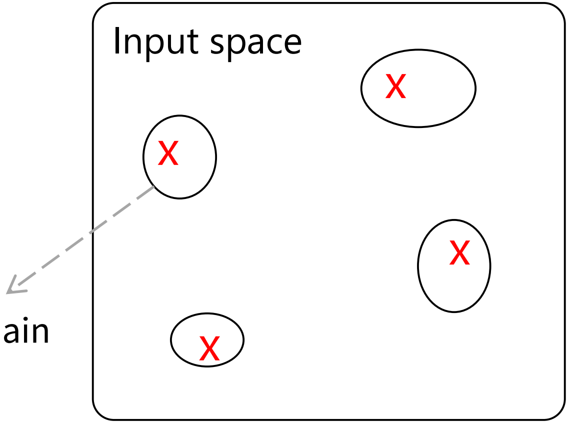
restaurant = [1 0 0 … 0]
diner = [0 1 0 … 0]
…



The Effect of Better Embeddings

- During training, neural networks learn to map regions of the input space to specific outputs
- If word embeddings map similar words to similar regions, the neural network will have an easier job

restaurant = [1 0 0 … 0]
diner = [0 1 0 … 0]
…

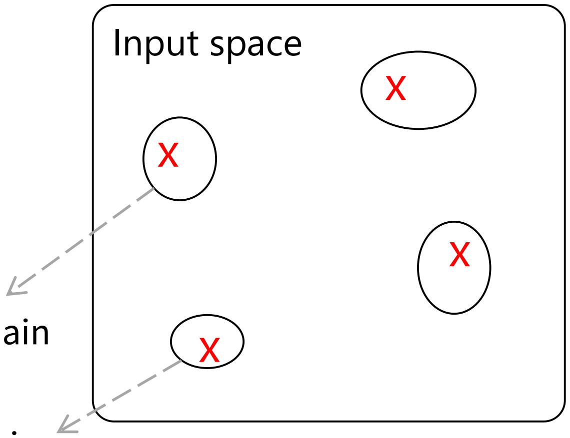


These sentences are in the restaurants domain

The Effect of Better Embeddings

- During training, neural networks learn to map regions of the input space to specific outputs
- If word embeddings map similar words to similar regions, the neural network will have an easier job

restaurant = [1 0 0 … 0]
diner = [0 1 0 … 0]
…



These sentences are in the restaurants domain

These are in the hotels domain

Word Representation: Dense Vectors

- Also called Distributed Representation
- In practice, ~100-1000 dimensional vectors (much smaller than $|V|$)
- Learned from large text corpora

Word Representation: Dense Vectors

- Also called Distributed Representation
- In practice, ~100-1000 dimensional vectors (much smaller than $|V|$)
- Learned from large text corpora

I went to this amazing restaurant last night.
We were at the diner when we saw him.
Ali went to the movies.
She was at the movies.
…

Word Representation: Dense Vectors

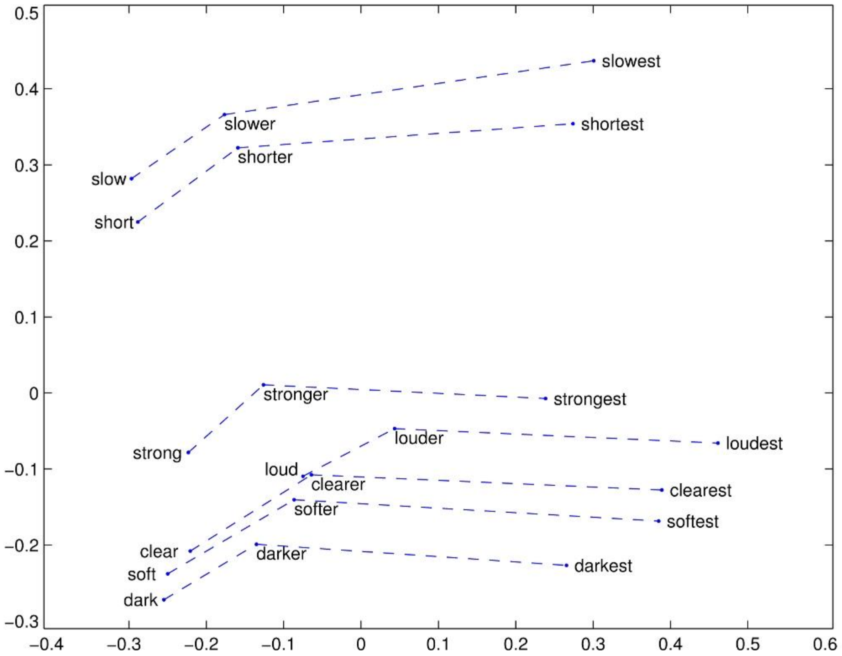
- Also called Distributed Representation
- In practice, ~100-1000 dimensional vectors (much smaller than $|V|$)
- Learned from large text corpora

I went to this amazing restaurant last night.
We were at the diner when we saw him.
Ali went to the movies.
She was at the movies.
…

Learn embeddings that maximize our ability to predict the surrounding words of a word

$$J(\theta) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{j=-m}^{+m}\log P(w_{t+j}|w_t\,;\theta)$$

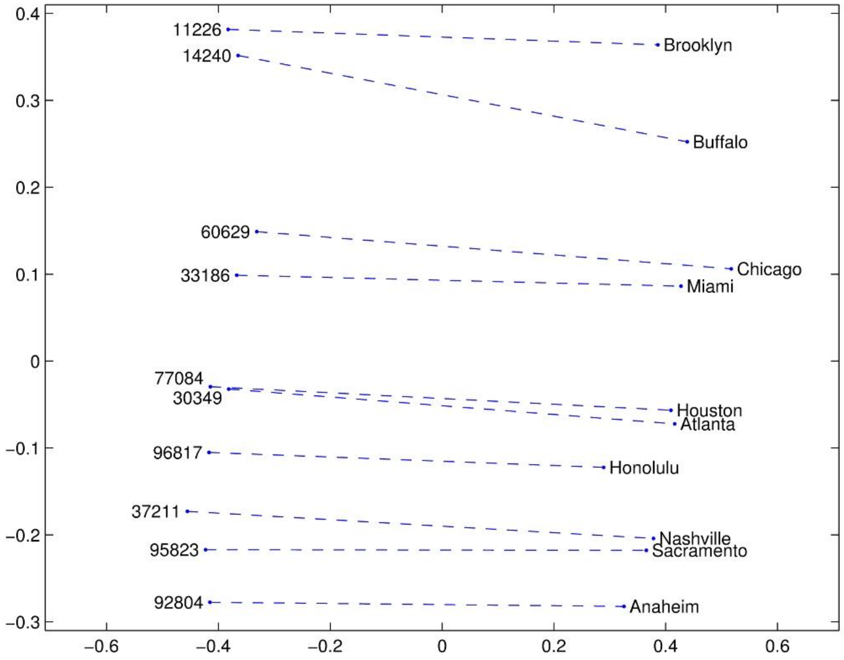
Word Representation: Dense Vectors



Images from GloVe: Global Vectors for Word Representation (2014)

Word Representation: Dense Vectors

There exists a 300-dimensional vector z such that if you add it to the vector of a city name, you get the vector of their zip codes!



Images from GloVe: Global Vectors for Word Representation (2014)

Word Representation: Dense Vectors

- We have one vector v for each word w.
- d has to encode all aspects and meanings of w

- These two sentences will be almost identical in terms of word embeddings.

How much does a share of Apple cost?

How much does a pound of apple cost?

- We can do better

Language Modeling

- The task of estimating the probability of a sequence of words

$$P(w_1 w_2 w_3 \dots w_m)$$

- Usually requires simplifying assumptions

$$P(w_1\, w_2\, w_3 \dots w_m) = \prod_{i=1}^{m} P(w_i | w_1 \dots w_{i-1})$$

$$\approx$$

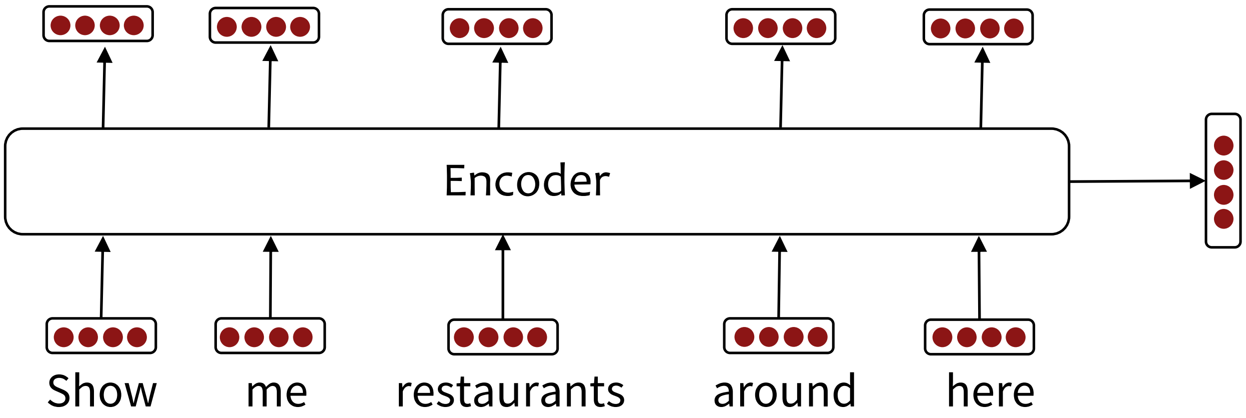
$$\prod_{i=1}^{m} P(w_i | w_{i-n} \dots w_{i-1})$$

Autoregressive Language Models

- Autoregressive: predict the next word

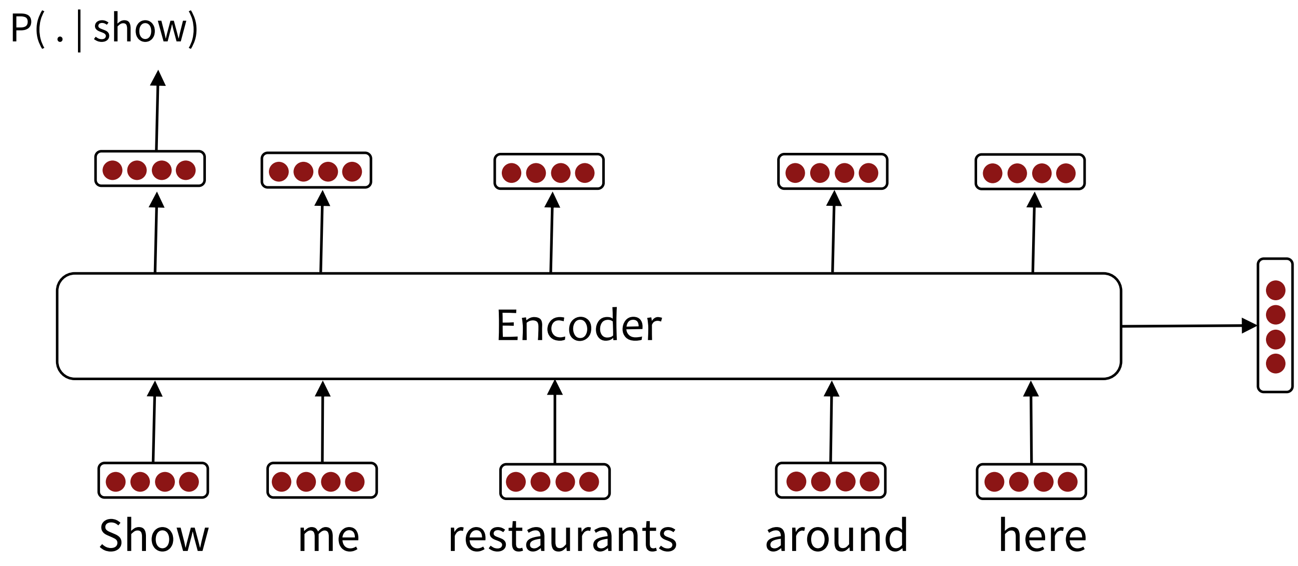
Autoregressive Language Models

- Autoregressive: predict the next word



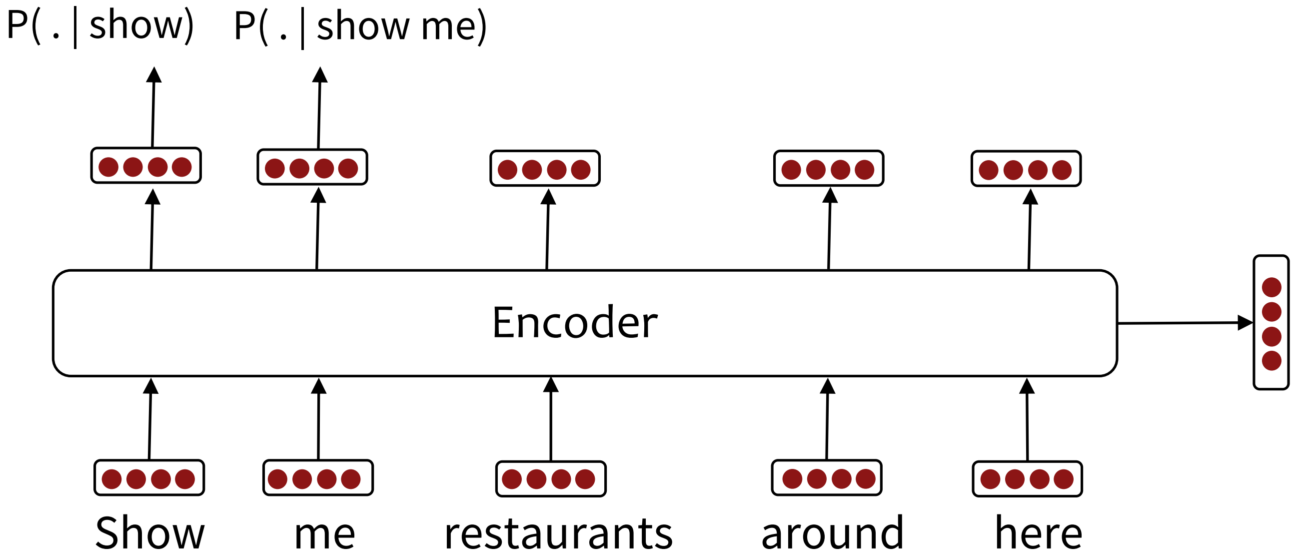
Autoregressive Language Models

- Autoregressive: predict the next word



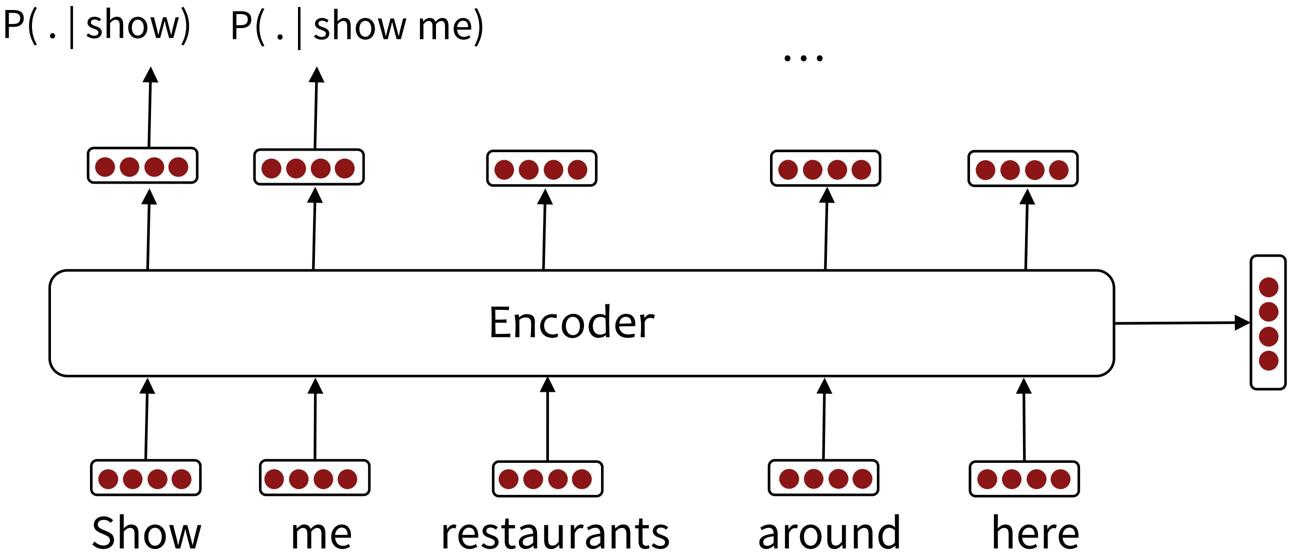
Autoregressive Language Models

- Autoregressive: predict the next word



Autoregressive Language Models

- Autoregressive: predict the next word

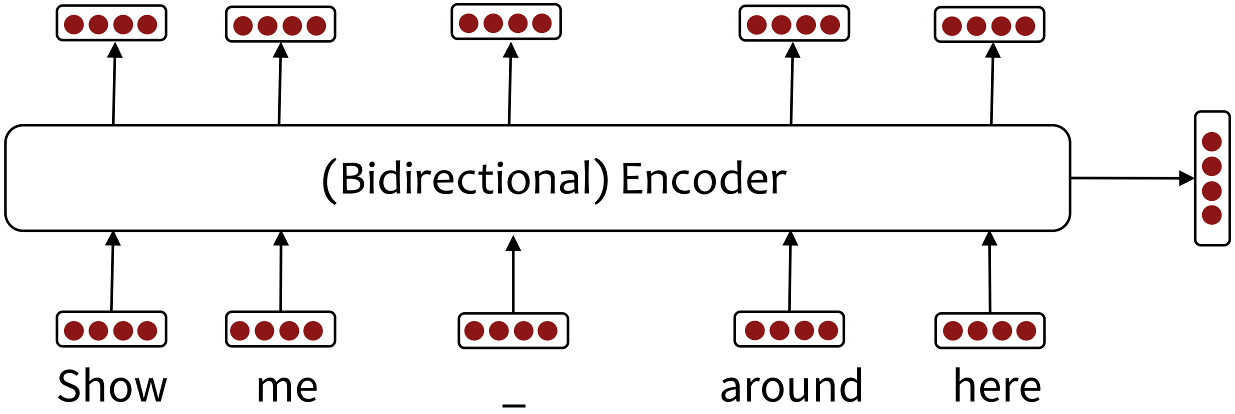


Masked Language Models

- Masked: fill in the blank

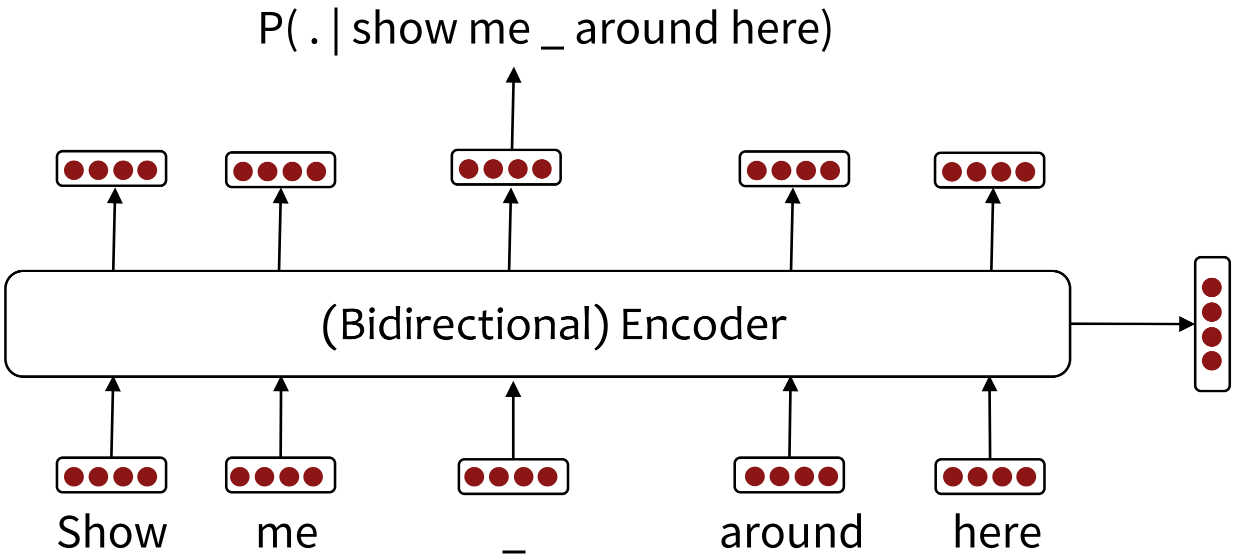
Masked Language Models

- Masked: fill in the blank



Masked Language Models

- Masked: fill in the blank



Word Representation: Contextual

Word Representation: Contextual

- Training data for a task is limited
- Pre-train a language model on a very large text corpus

- Embeddings from Language Models: ELMo (Oct. 2017)
- Generative Pre-training: GPT (June 2018)
- Bidirectional Encoder Representations from Transformers: BERT (Oct. 2018)
- GPT-2 (Feb. 2019)
- …

Word Representation: Contextual

- Training data for a task is limited
- Pre-train a language model on a very large text corpus

- Embeddings from Language Models: ELMo (Oct. 2017)
- Generative Pre-training: GPT (June 2018)
- Bidirectional Encoder Representations from Transformers: BERT (Oct. 2018)
- GPT-2 (Feb. 2019)
- ...

corpus size

800 million words

1x

4x

48x

Quiz

A language model is trained to be good at predicting missing words.

How can we test if the contextual representations learned by the language model are good at capturing the meaning of sentences as well?

Quiz

A language model is trained to be good at predicting missing words.

How can we test if the contextual representations learned by the language model are good at capturing the meaning of sentences as well?

1. By evaluating them on downstream tasks. BERT for instance improved state of the art results for several NLP tasks by 4-8%.
2. By looking at the representations themselves.

Sequence to Sequence

When Both Input and Output Are Sequences of Words

When Both Input and Output Are Sequences of Words

- Seq2Seq has many use cases
  - Machine Translation
  - Question Generation
  - Semantic Parsing

- We will use examples from semantic parsing

When Both Input and Output Are Sequences of Words

- Seq2Seq has many use cases
  - Machine Translation
  - Question Generation
  - Semantic Parsing

- We will use examples from semantic parsing

```
Show me restaurants around here   →        now => @QA.Restaurant()
                                         , geo == current_location => notify
```

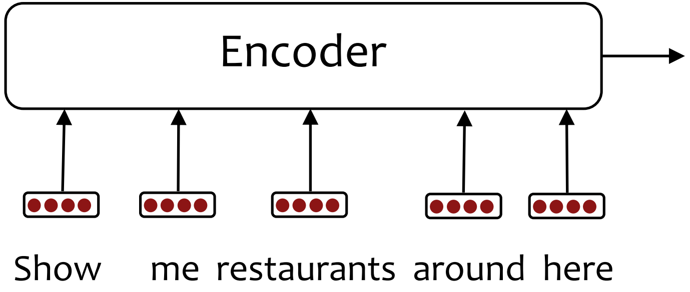
Sequence to Sequence

- Dataset: pairs of source sentence $x_1 x_2 \dots x_s$ and target sentence $y_1 y_2 \dots y_t$
- For instance, pairs of natural sentences and their ThingTalk programs
- The objective is to learn θ that maximizes:

$$J(\theta) = P(y_1\, y_2 \dots y_t \mid x_1\, x_2 \dots x_s\, ; \theta)$$
$$=$$
$$P(y_1 | x_1 x_2 \dots x_s\, ; \theta) \times P(y_2 | y_1 x_1 x_2 \dots x_s\, ; \theta) \times \cdots$$

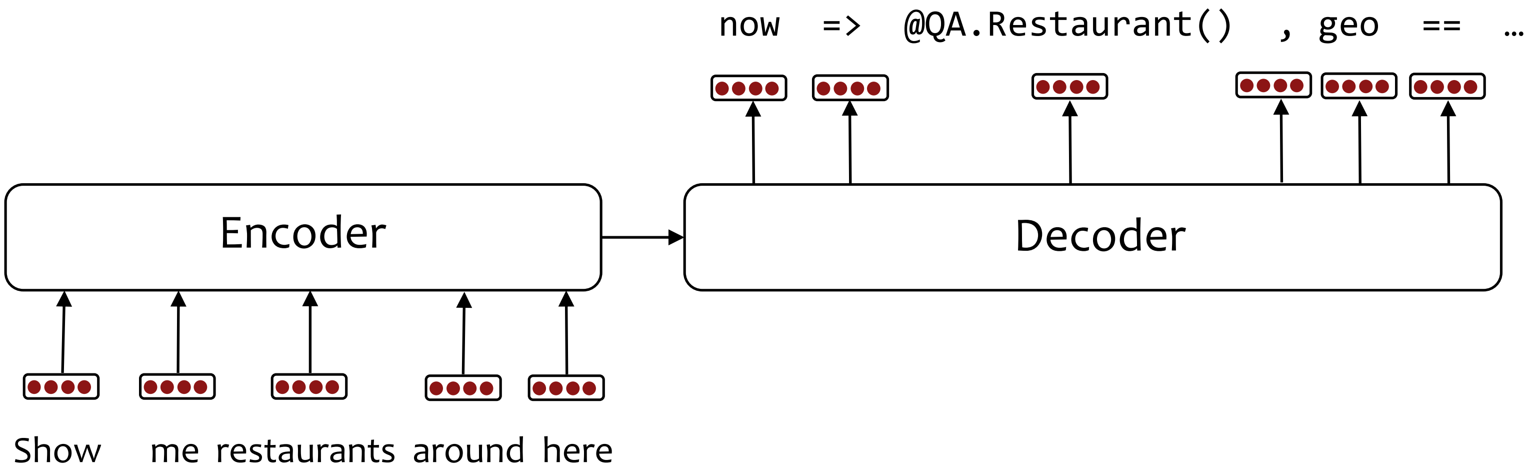
Encoder-Decoder

We can use encoder-decoder models for Seq2Seq tasks



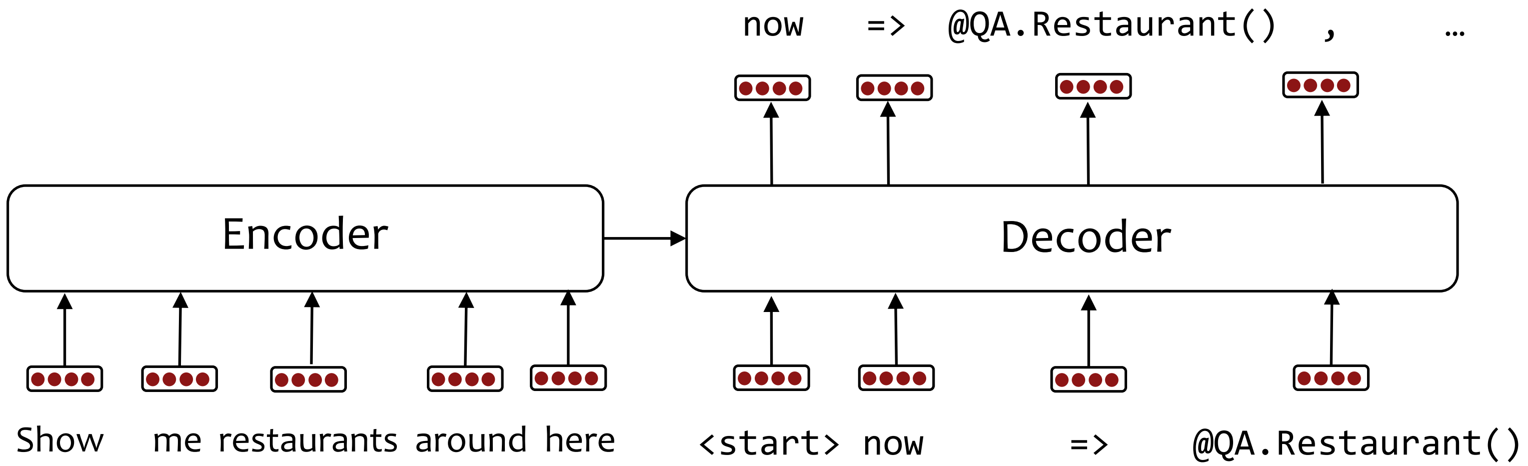
Encoder-Decoder

We can use encoder-decoder models for Seq2Seq tasks



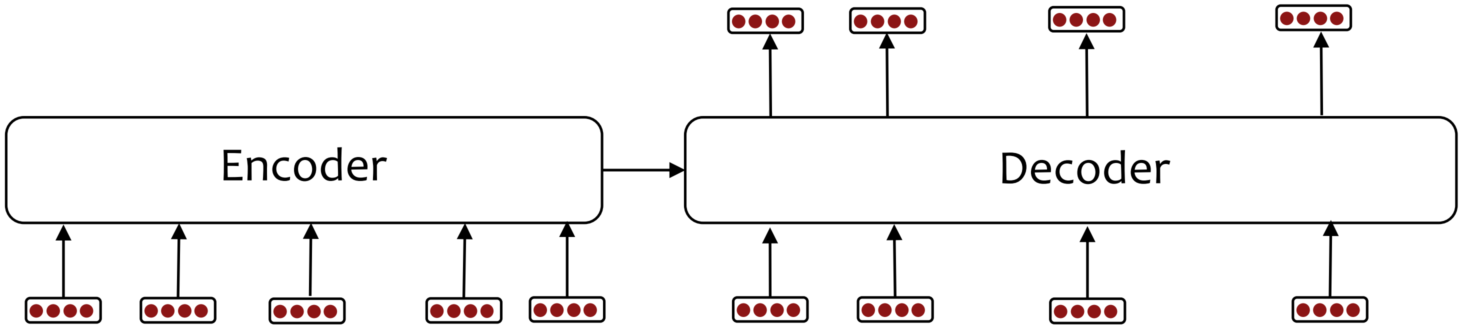
Encoder-Decoder

In practice, we also input the previous token to the decoder



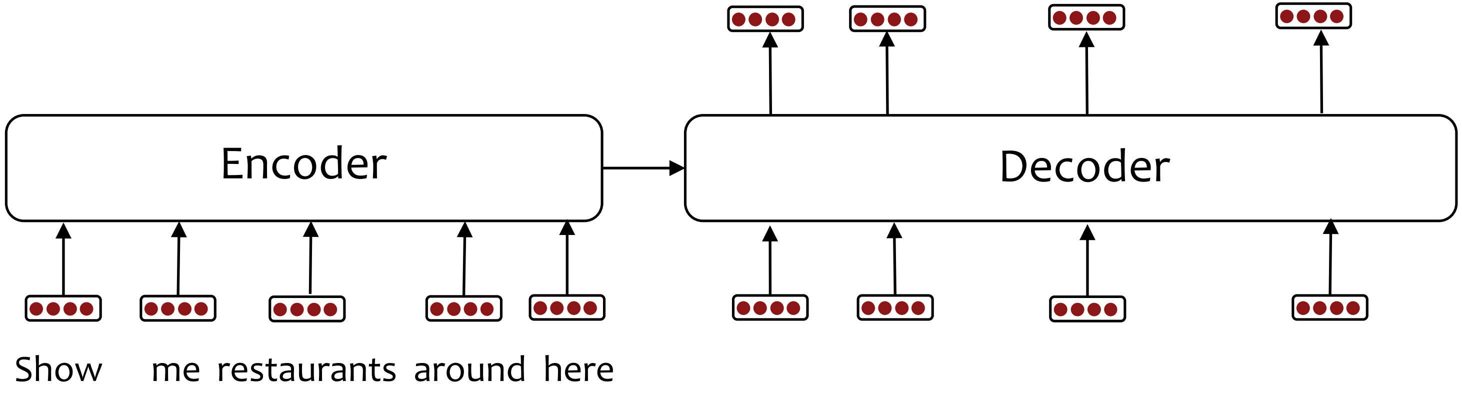
Encoder-Decoder

At training time, decoder always gets the **gold target** as input



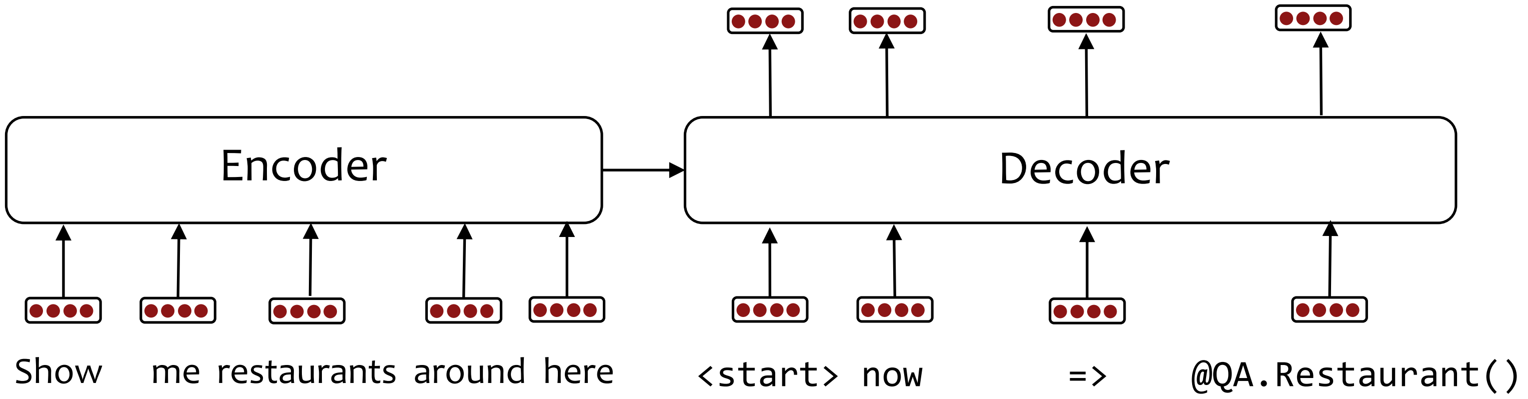
Encoder-Decoder

At training time, decoder always gets the **gold target** as input



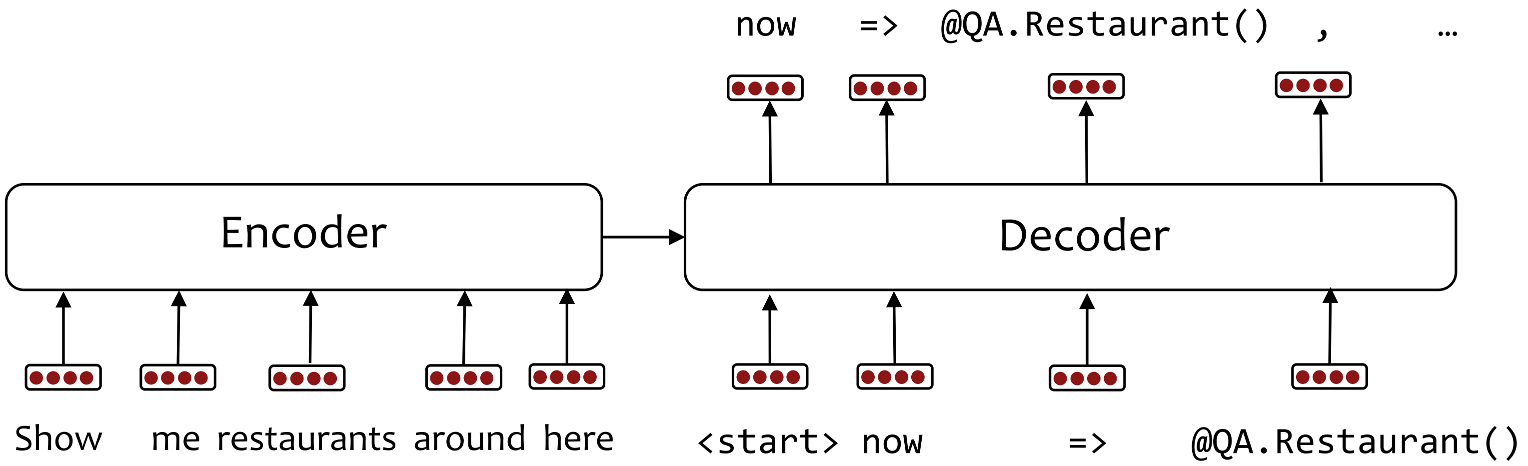
Encoder-Decoder

At training time, decoder always gets the **gold target** as input



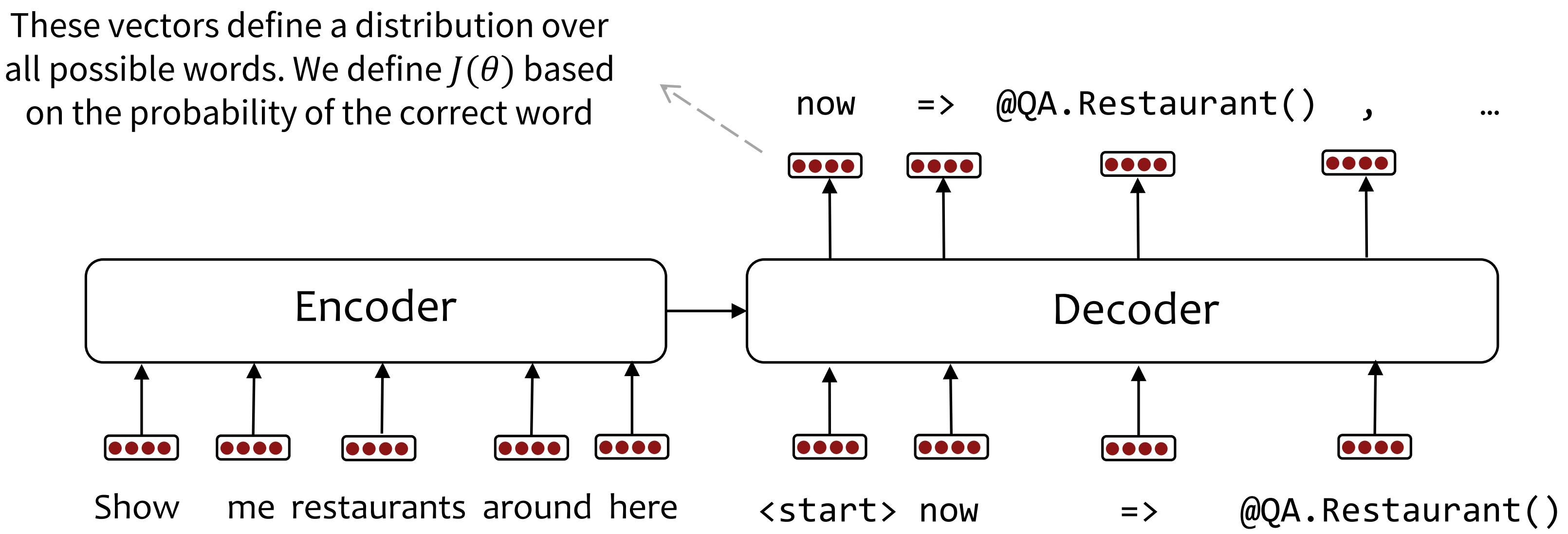
Encoder-Decoder

At training time, decoder always gets the **gold target** as input



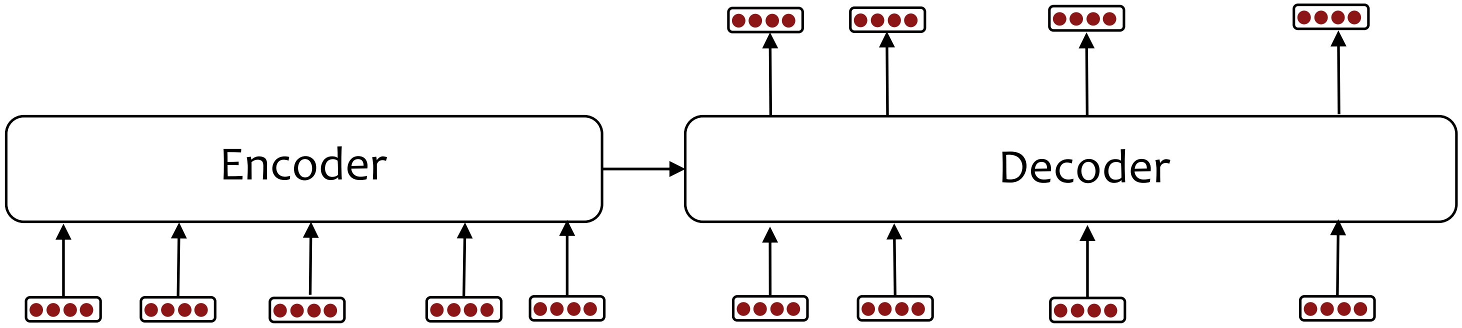
Encoder-Decoder

At training time, decoder always gets the **gold target** as input



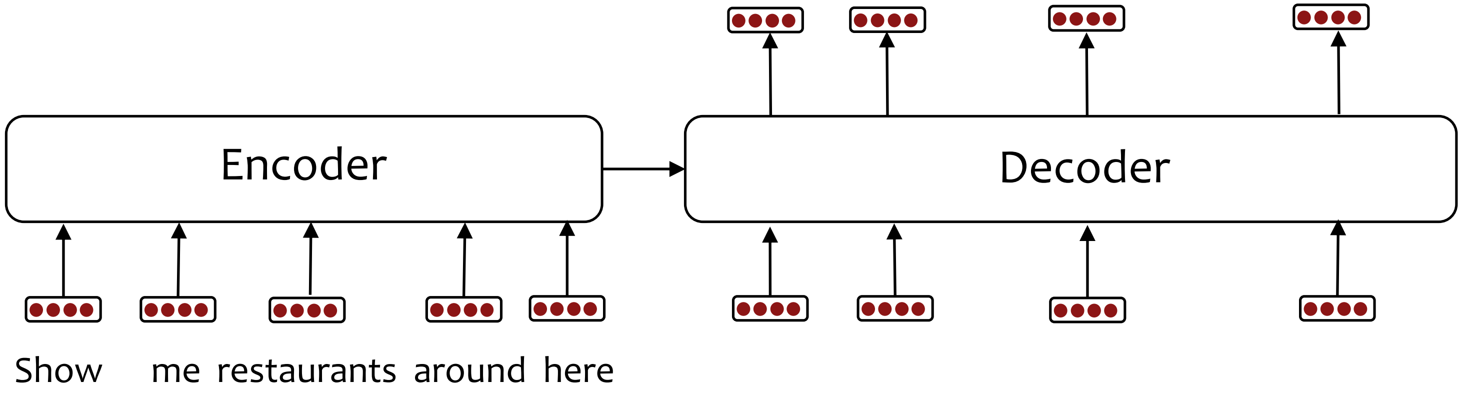
Encoder-Decoder

- At generation time, we feed in the word generated by the decoder at previous time step.
- Pro: very fast to converge in practice
- Con: model is never exposed to its own search errors during training



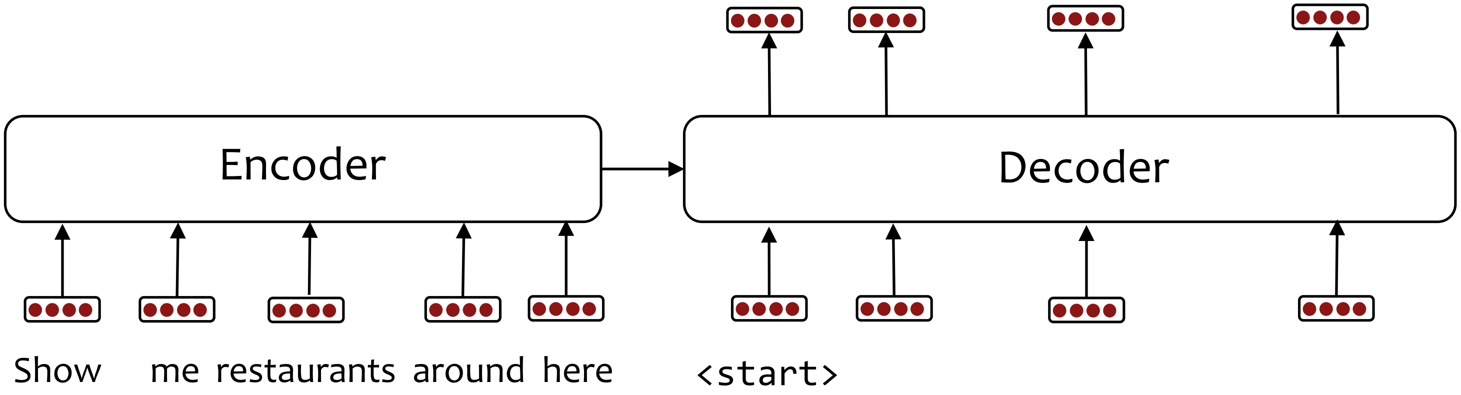
Encoder-Decoder

- At generation time, we feed in the word generated by the decoder at previous time step.
- Pro: very fast to converge in practice
- Con: model is never exposed to its own search errors during training



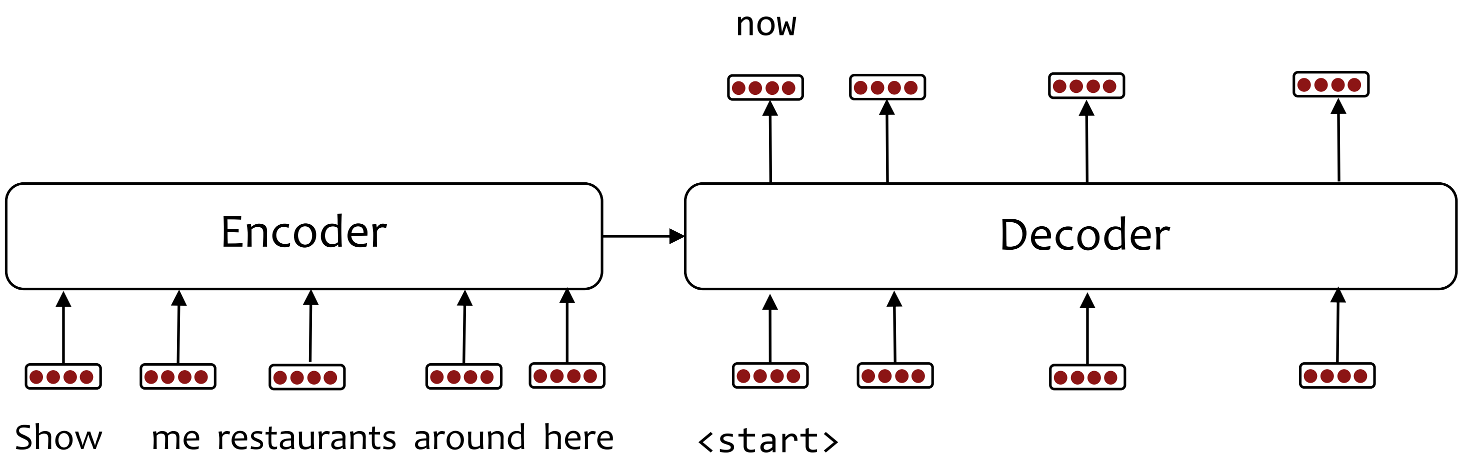
Encoder-Decoder

- At generation time, we feed in the word generated by the decoder at previous time step.
- Pro: very fast to converge in practice
- Con: model is never exposed to its own search errors during training



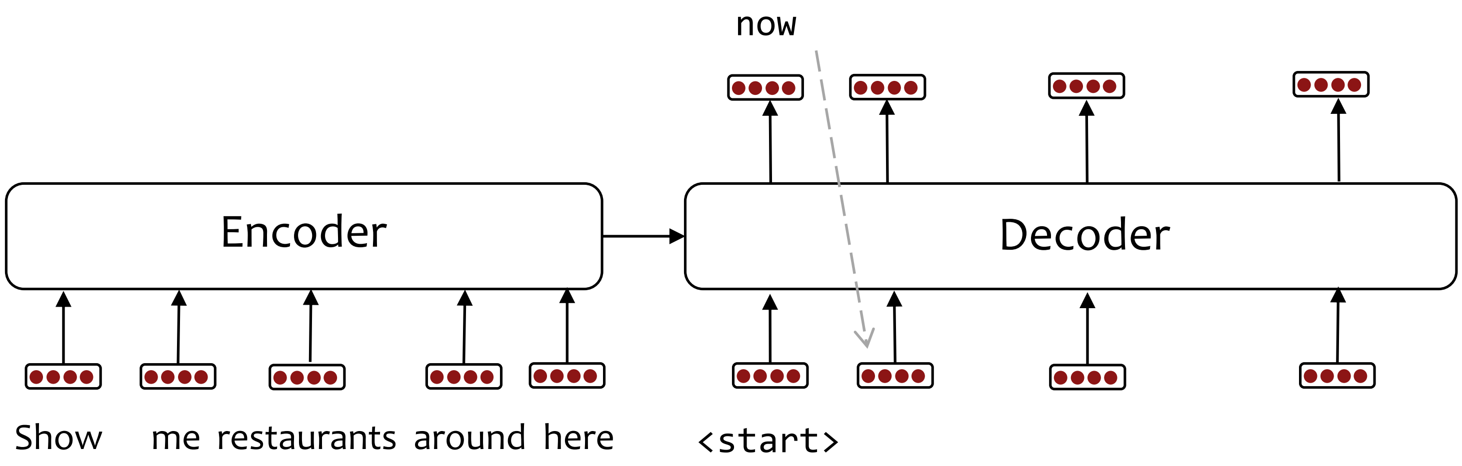
Encoder-Decoder

- At generation time, we feed in the word generated by the decoder at previous time step.
- Pro: very fast to converge in practice
- Con: model is never exposed to its own search errors during training



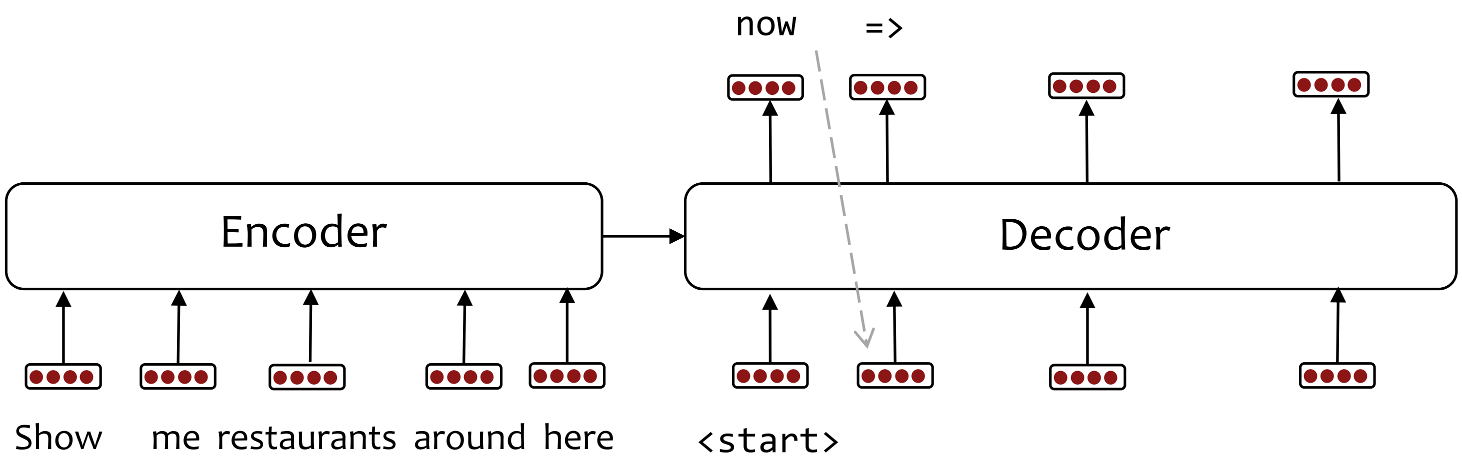
Encoder-Decoder

- At generation time, we feed in the word generated by the decoder at previous time step.
- Pro: very fast to converge in practice
- Con: model is never exposed to its own search errors during training



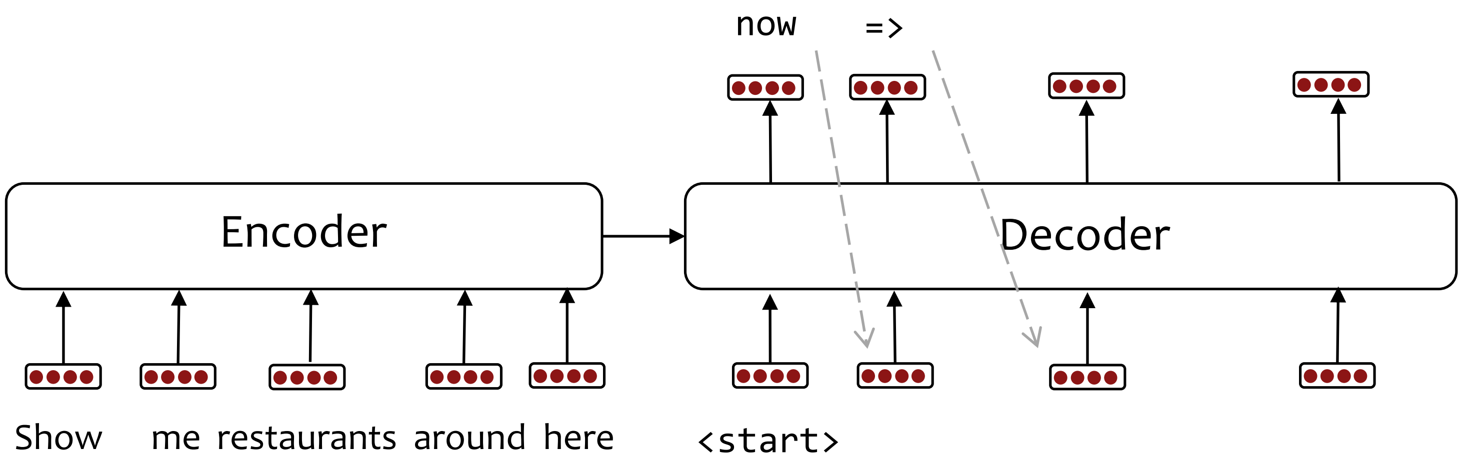
Encoder-Decoder

- At generation time, we feed in the word generated by the decoder at previous time step.
- Pro: very fast to converge in practice
- Con: model is never exposed to its own search errors during training



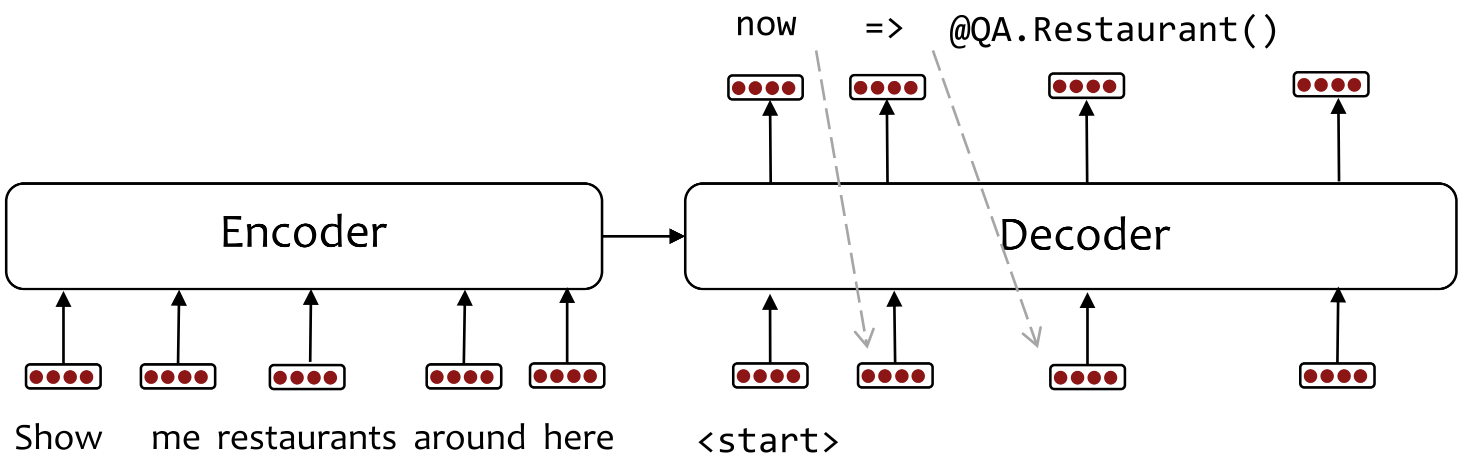
Encoder-Decoder

- At generation time, we feed in the word generated by the decoder at previous time step.
- Pro: very fast to converge in practice
- Con: model is never exposed to its own search errors during training



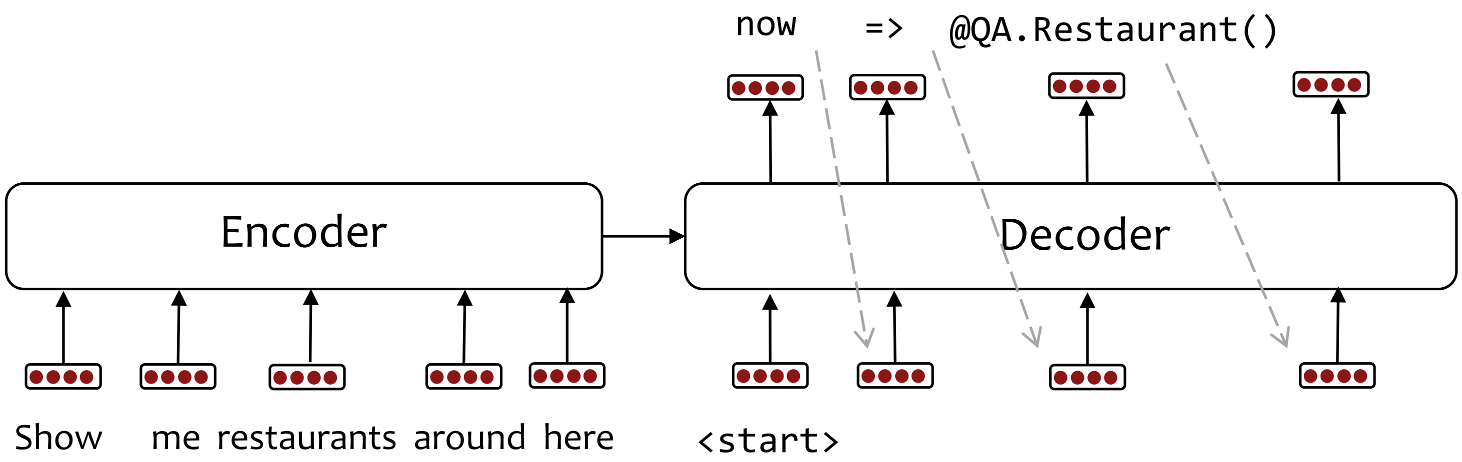
Encoder-Decoder

- At generation time, we feed in the word generated by the decoder at previous time step.
- Pro: very fast to converge in practice
- Con: model is never exposed to its own search errors during training



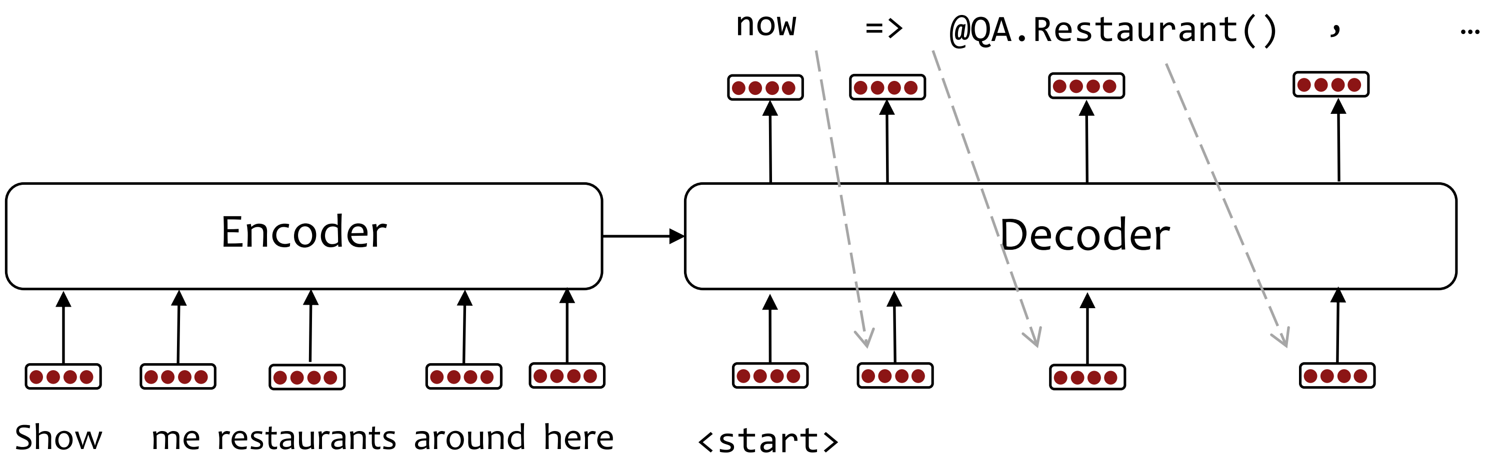
Encoder-Decoder

- At generation time, we feed in the word generated by the decoder at previous time step.
- Pro: very fast to converge in practice
- Con: model is never exposed to its own search errors during training



Encoder-Decoder

- At generation time, we feed in the word generated by the decoder at previous time step.
- Pro: very fast to converge in practice
- Con: model is never exposed to its own search errors during training



From Word Probabilities to Output Sequence

- Greedy decoding: at each step, pick the most probable word
    - Greedy decoding can make search errors: if we choose a wrong word at a step, we might never recover
- Beam Search: at each step, keep the K most probable observed outputs
- Sampling: pick a word at random according to the distribution
- …

Downside of Word-Level Loss

Source: `Show me restaurants around here.`

Gold target: `now => @QA.Restaurant() , geo == current_location => notify`

Model output: `now => @QA.Hospital() , geo == current_location => notify`

Most of the sentence is the same as the gold, so low cost, but you will –literally- end up in a hospital!

A small difference in words is not the same as a small difference in meaning.

Downside of Word-Level Loss

Source: Show me nearby restaurants.

Gold target: mostrami ristoranti nelle vicinanze

Model output: sto cercando un ristorante qui attorno
(I'm looking for a restaurant around here)

Most of the sentence is different from the gold, so high cost, but the answer is correct.

Difference in words is not the same as difference in meaning.

Quiz

Is this a problem in semantic parsing as well?

Quiz

Is this a problem in semantic parsing as well?

Not for ThingTalk. ThingTalk is normalized, that is, each meaning has exactly one ThingTalk code.

Attention

Capturing Long Term Dependencies is Important in NL

- When generating a word, the model has to look at multiple words that are potentially far from each other.

Alice is young, lively and beautiful → Alice è giovane, vivace e → bello / bella

Capturing Long Term Dependencies is Important in NL

- When generating a word, the model has to look at multiple words that are potentially far from each other.

Alice is young, lively and beautiful → Alice è giovane, vivace e → bello / bella

- Some words are more important than others

Capturing Long Term Dependencies is Important in NL

- When generating a word, the model has to look at multiple words that are potentially far from each other.

Alice is young, lively and beautiful → Alice è giovane, vivace e → bello / bella

- Some words are more important than others

How far away is the closest Italian restaurant to me? → `now => [ distance ] of ( compute distance …`

Capturing Long Term Dependencies is Important in NL

- When generating a word, the model has to look at multiple words that are potentially far from each other.

Alice is young, lively and beautiful → Alice è giovane, vivace e → bello / bella

- Some words are more important than others

How far away is the closest Italian restaurant to me? → `now => [ distance ] of ( compute distance …`

Capturing Long Term Dependencies is Important in NL

- When generating a word, the model has to look at multiple words that are potentially far from each other.

Alice is young, lively and beautiful → Alice è giovane, vivace e → bello / bella

- Some words are more important than others

How far away is the closest Italian restaurant to me? → `now => [ distance ] of ( compute distance …`

Capturing Long Term Dependencies is Important in NL

- When generating a word, the model has to look at multiple words that are potentially far from each other.

Alice is young, lively and beautiful → Alice è giovane, vivace e → bello / bella

- Some words are more important than others

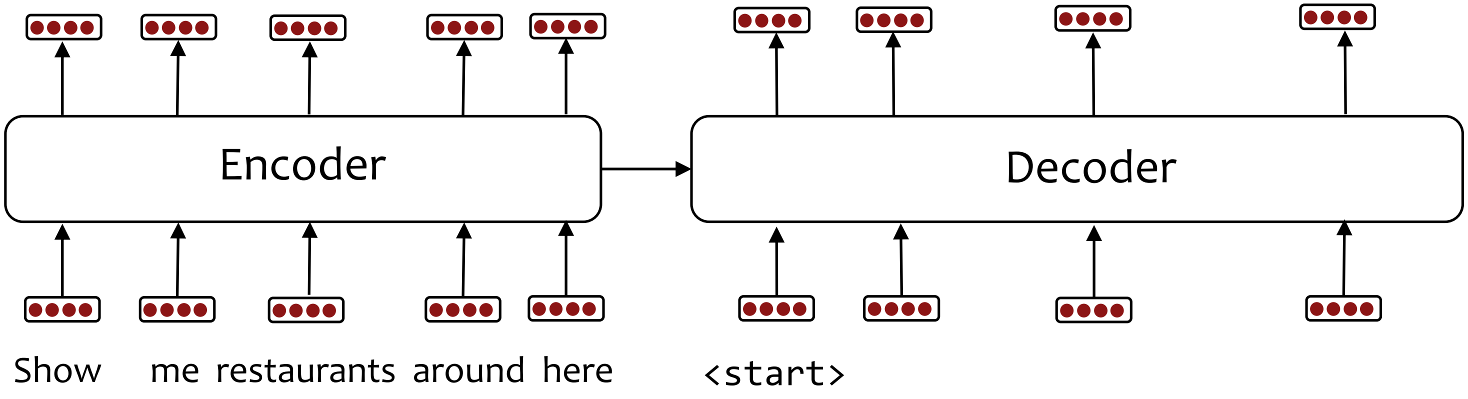
How far away is the closest Italian restaurant to me? → `now => [ distance ] of ( compute distance …`

Attention

- Designed to alleviate this exact problem
- At each decoding step, compute *attention scores* by combining encoder and decoder states
- Normalize scores with *softmax*
- Mix them into a *context vector*
- Mix decoder state and context vector

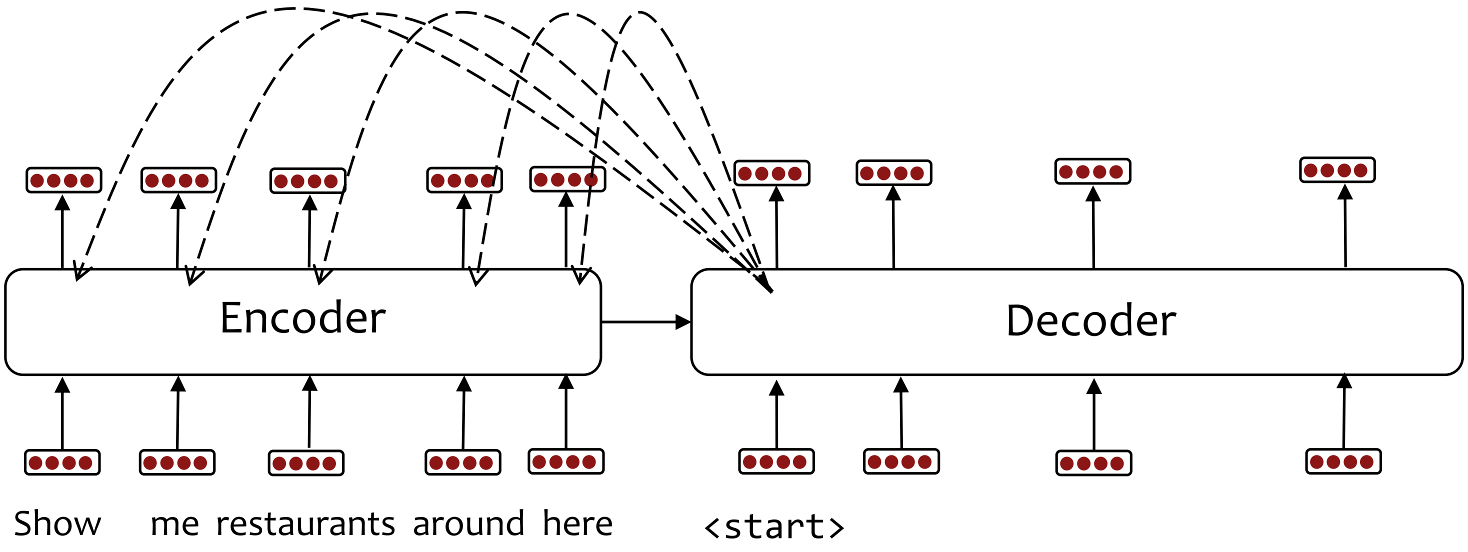
Encoder-Decoder with Attention

When generating a word for the output, directly look at all the words in the input



Encoder-Decoder with Attention

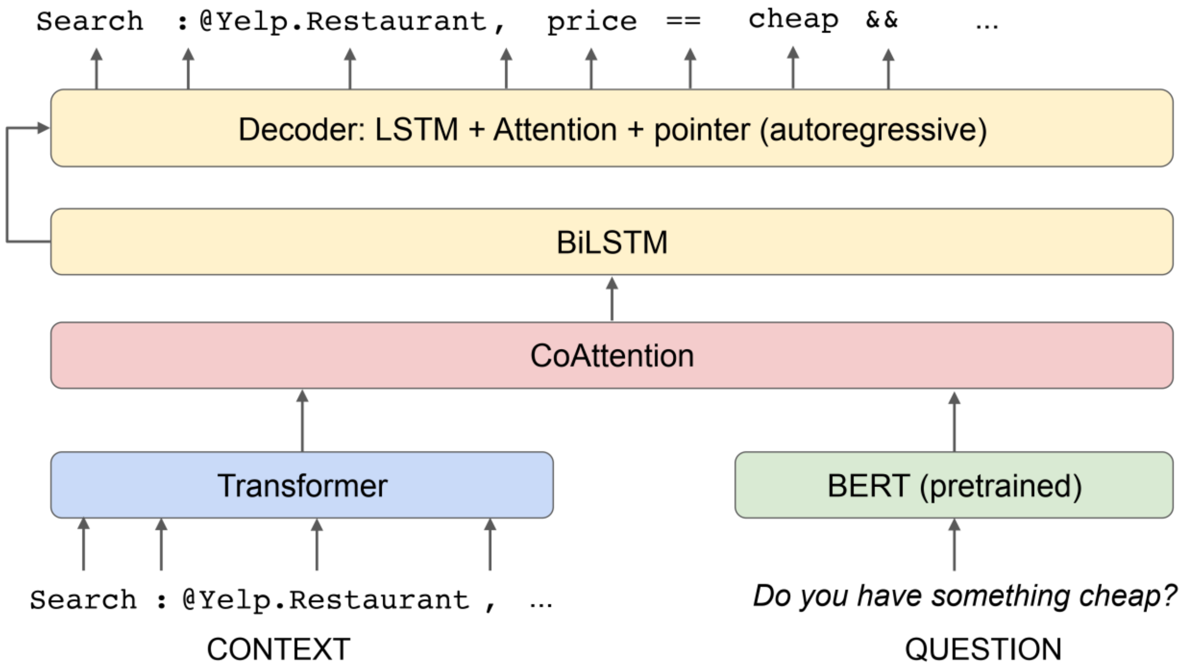
When generating a word for the output, directly look at all the words in the input



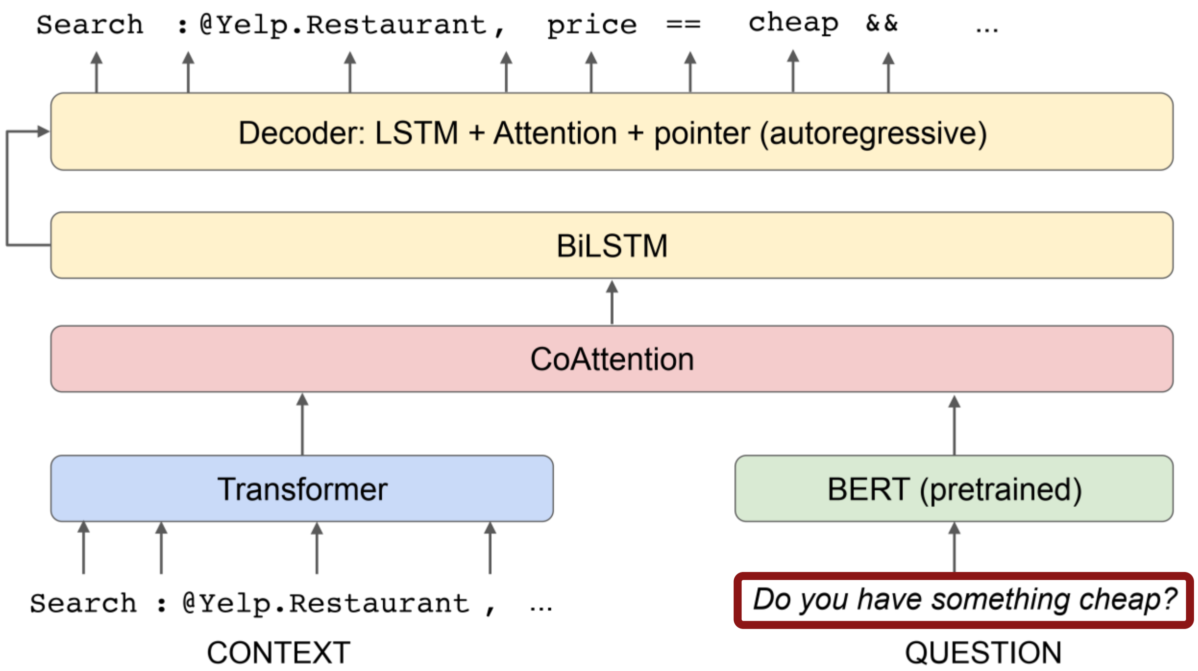
Transformer

- Is a relatively new class of parametrized functions
- Instead of RNNs, is entirely made up of attentions
- Attentions are easy to compute in parallel, which is especially beneficial when using GPUs
- Empirically, Transformer outperforms RNN in a wide range of tasks and datasets.
- Has encoder, decoder and Seq2Seq variants.

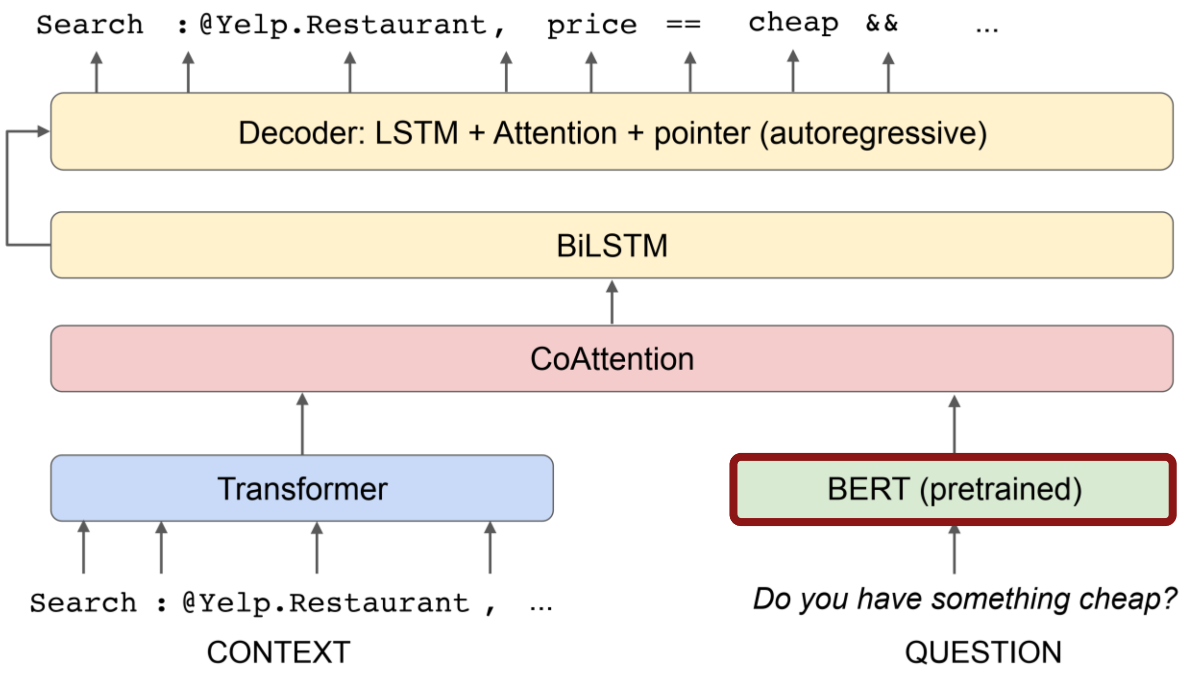
Remember This Image from Lecture 1?



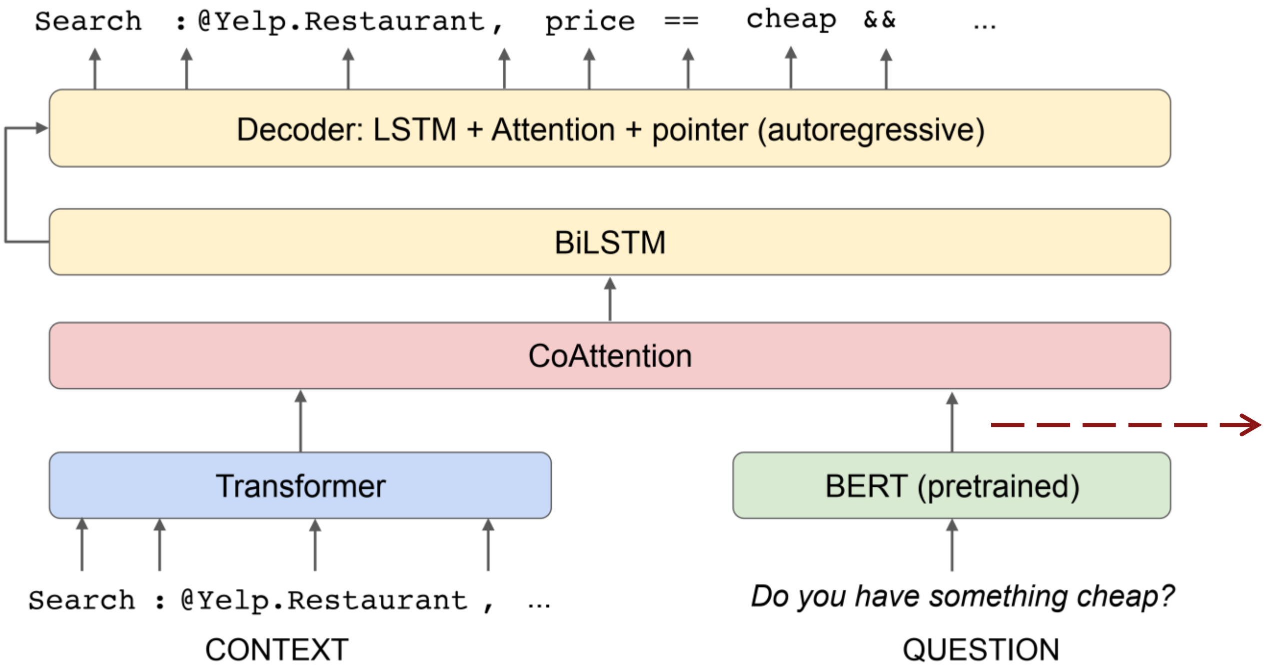
Remember This Image from Lecture 1?



Remember This Image from Lecture 1?

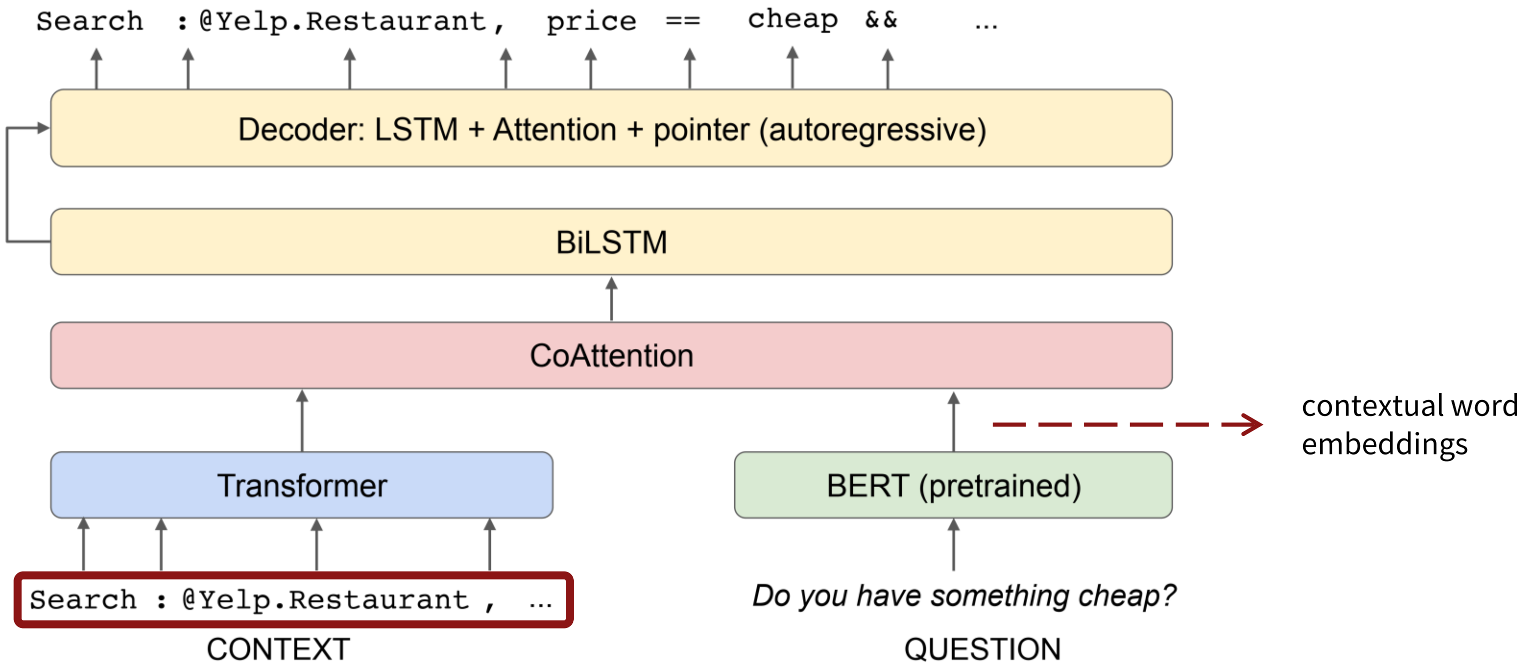


Remember This Image from Lecture 1?

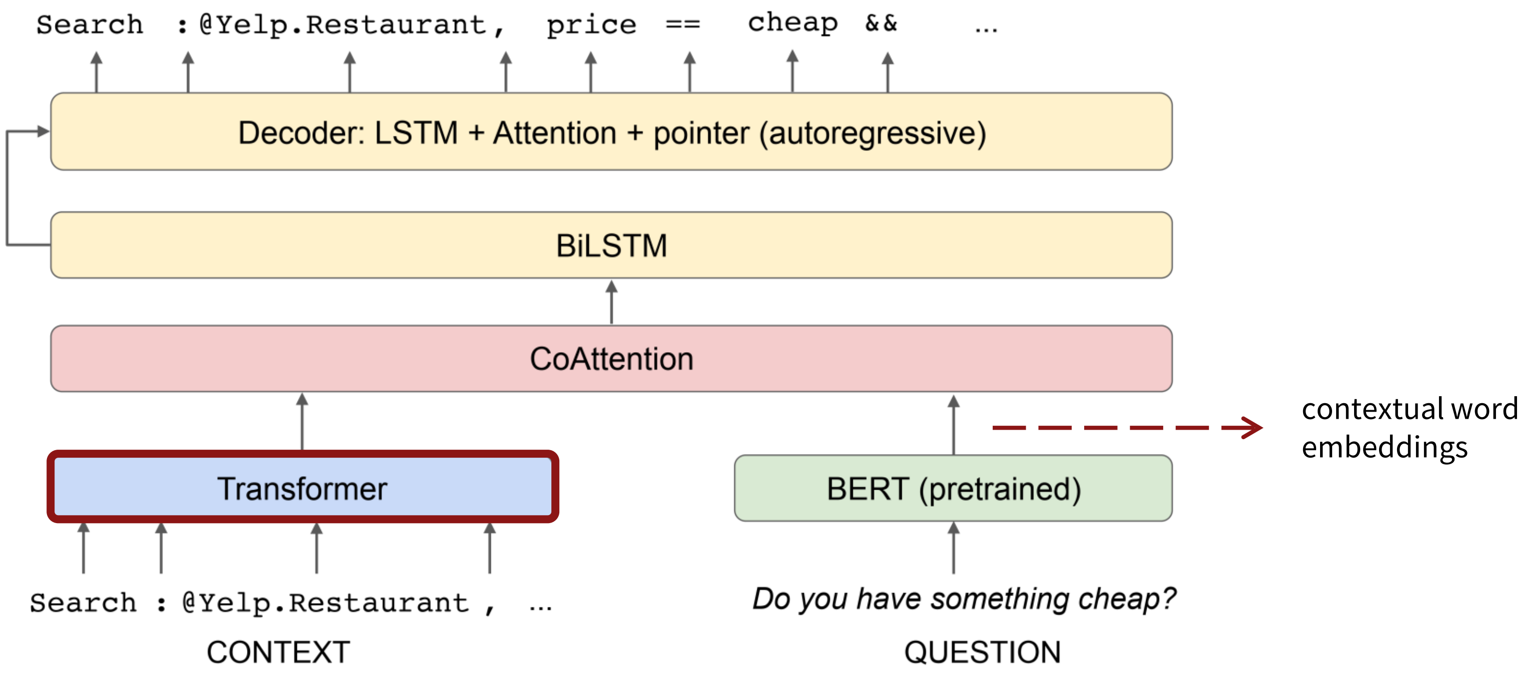


contextual word embeddings

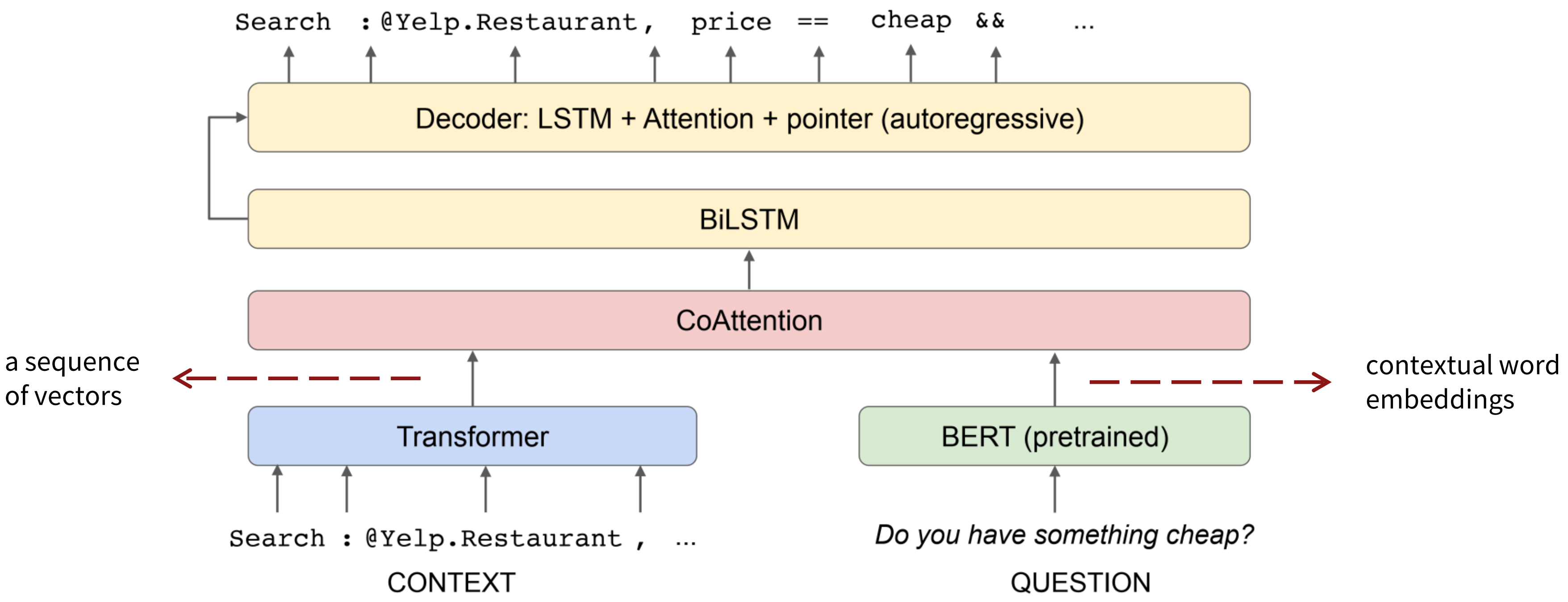
Remember This Image from Lecture 1?



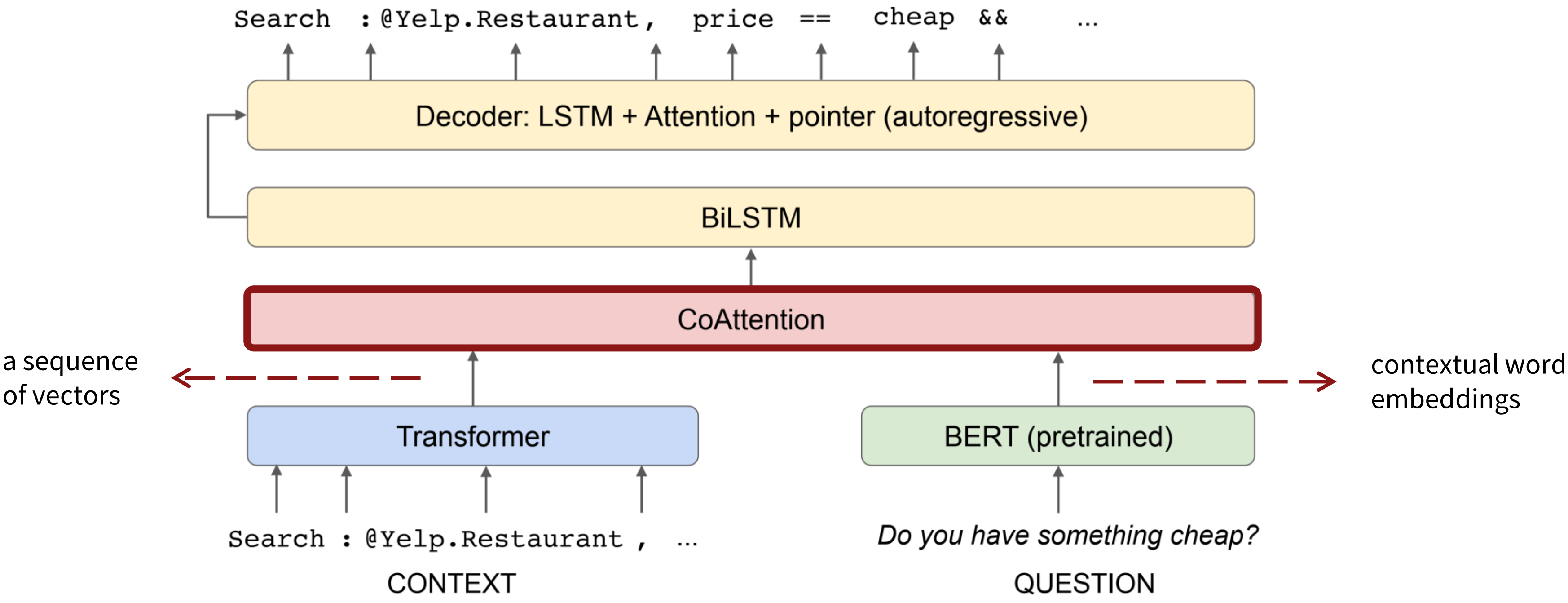
Remember This Image from Lecture 1?



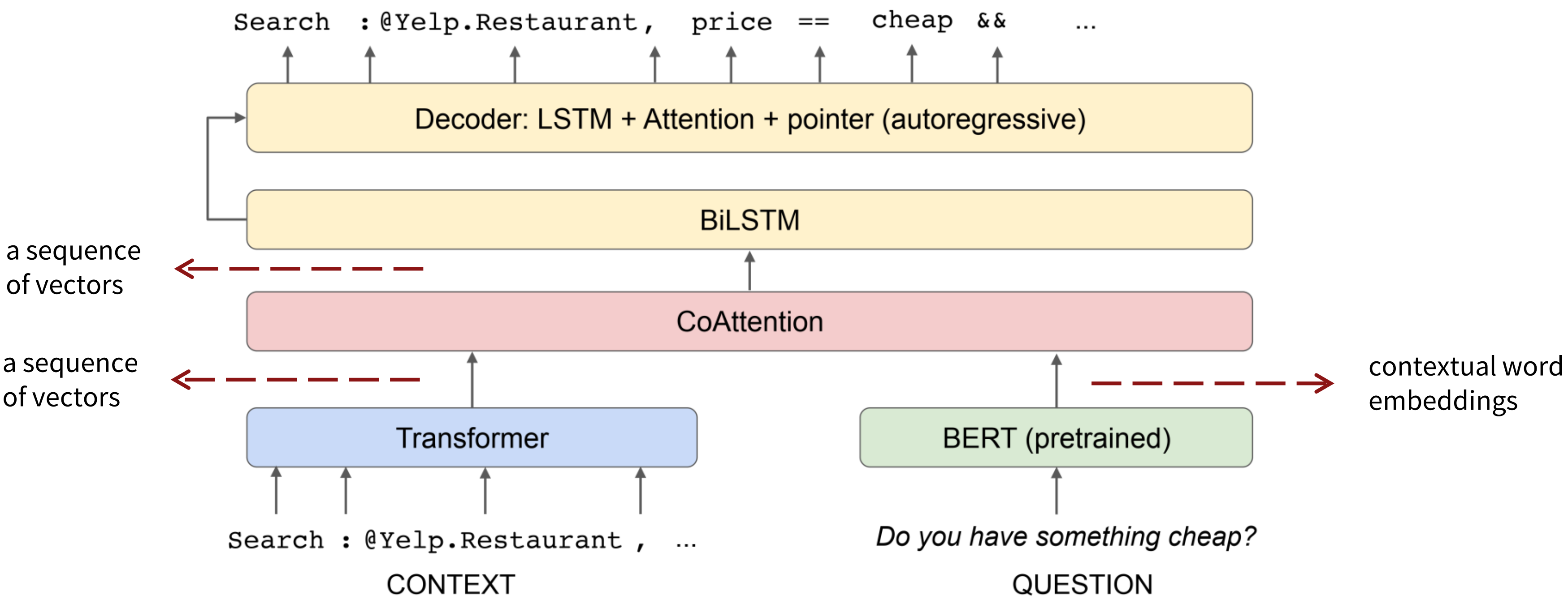
Remember This Image from Lecture 1?



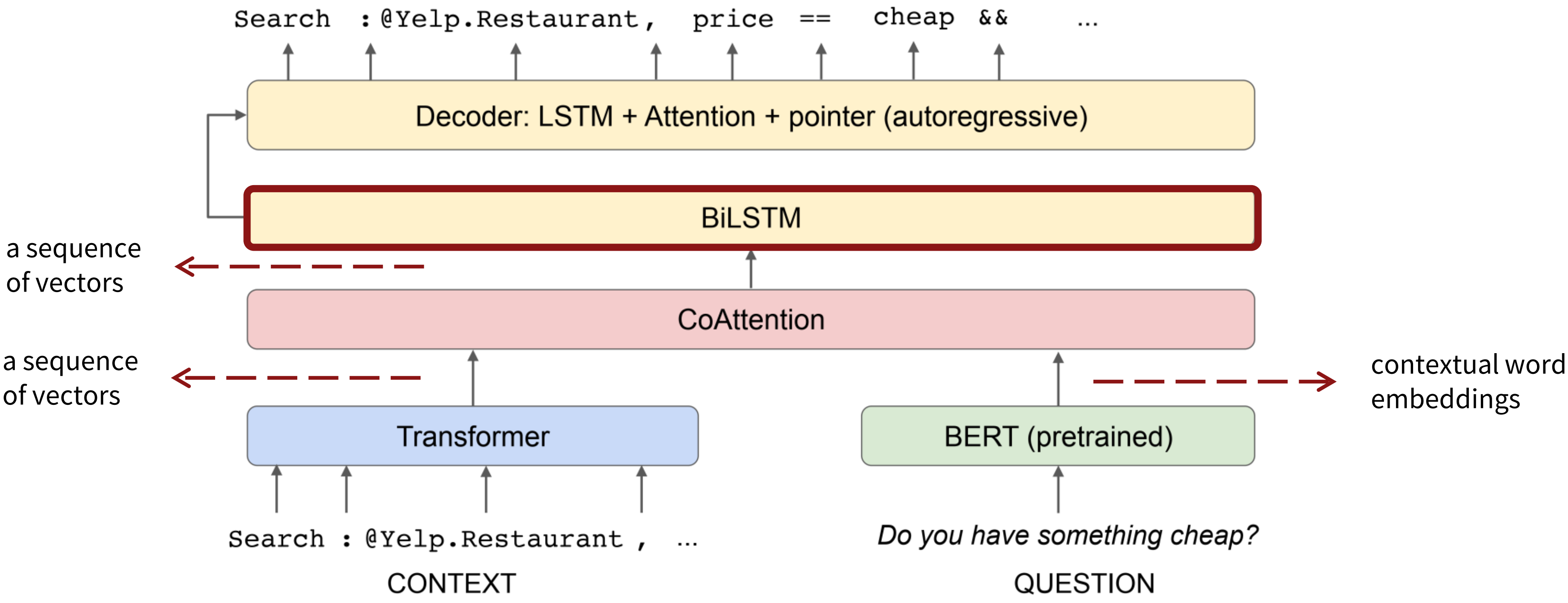
Remember This Image from Lecture 1?



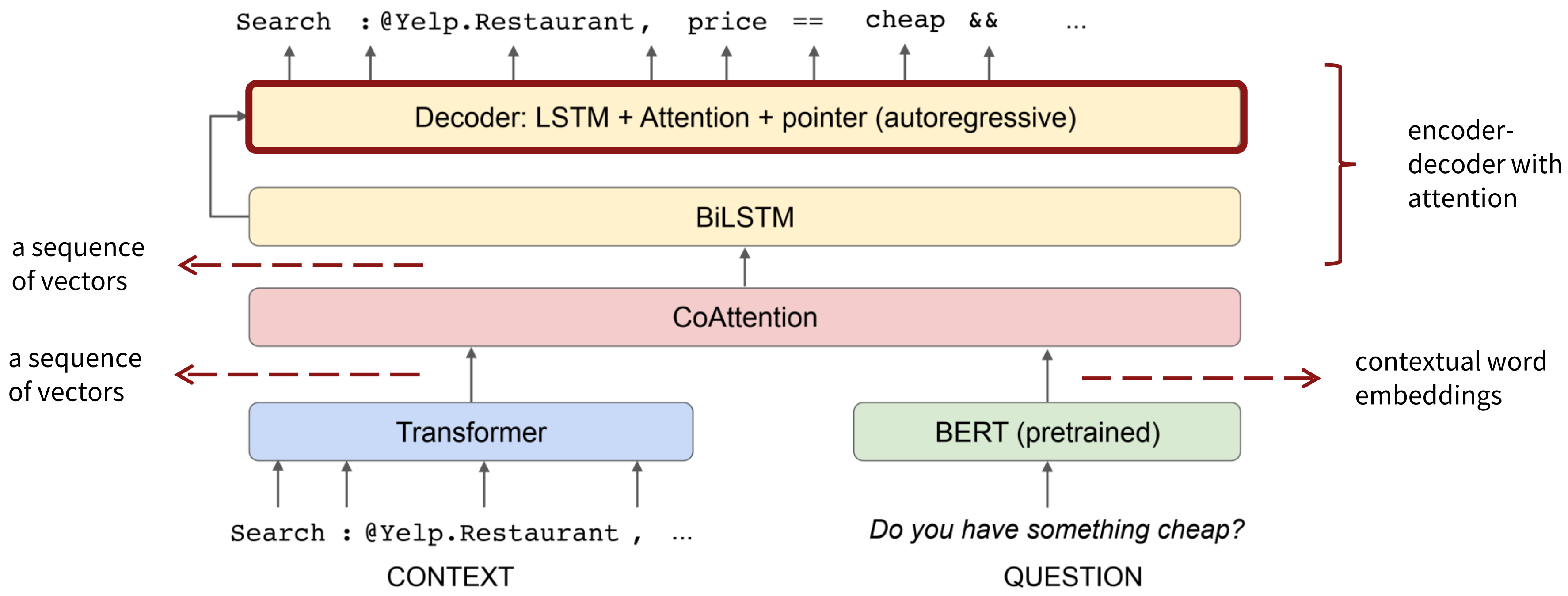
Remember This Image from Lecture 1?



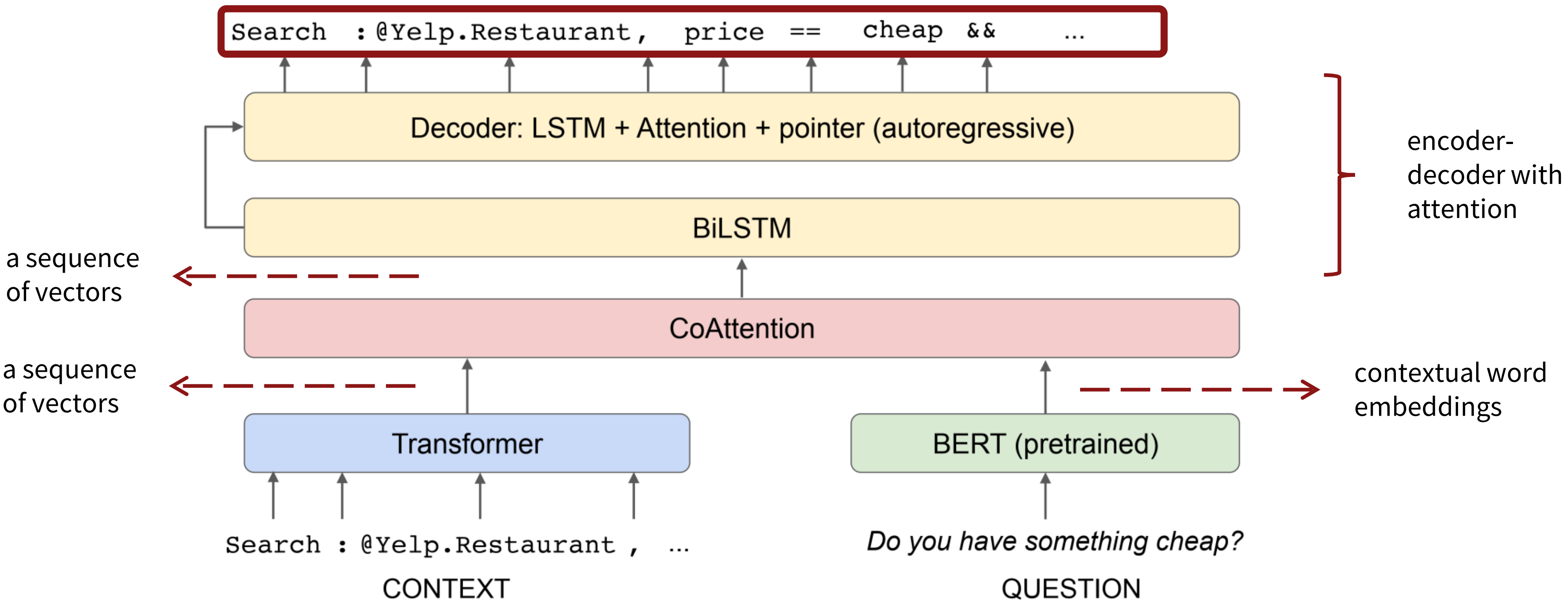
Remember This Image from Lecture 1?



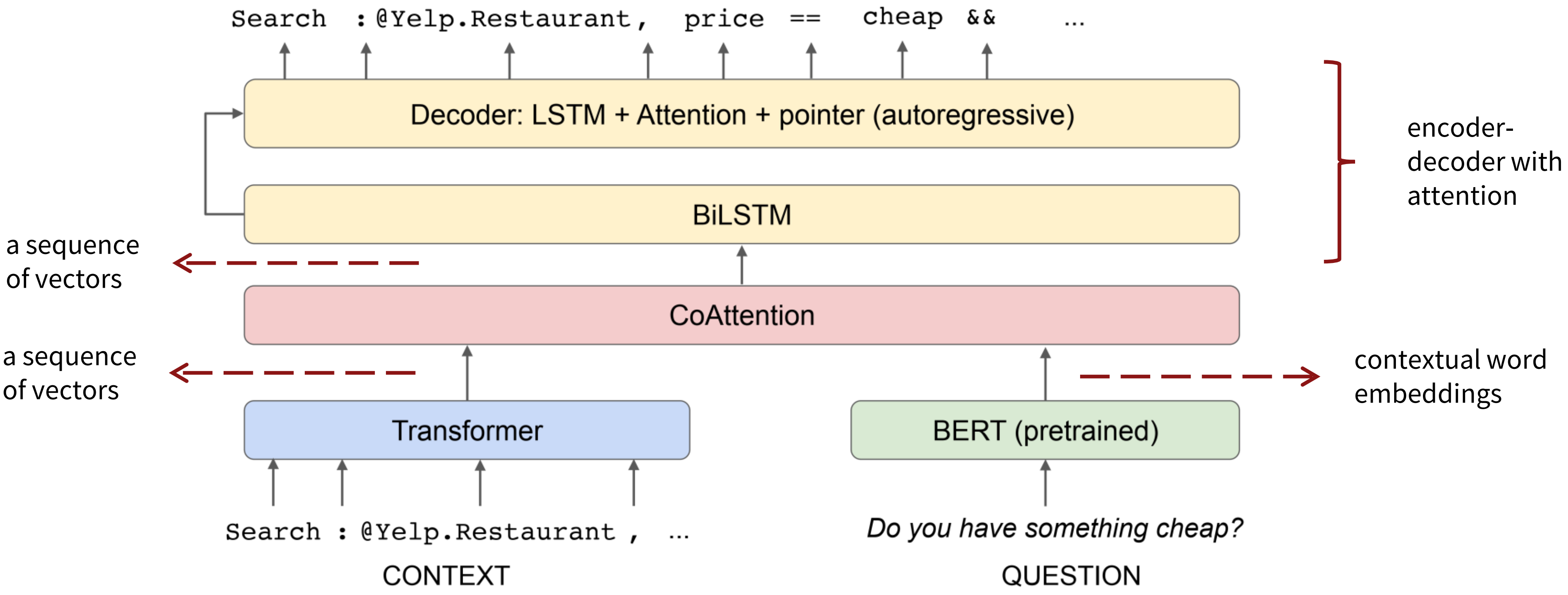
Remember This Image from Lecture 1?



Remember This Image from Lecture 1?



Remember This Image from Lecture 1?



Practical Notes

- Python
- PyTorch
- Genie NLP
- HuggingFace’s transformers package includes state-of-the-art pre-trained language models like BERT