CS 4395 Spring 2025

Midterm Review

First

- Closed-book closed-notes
- See the cover page for more information
- You don’t have to write the explanations if confident about the answer.

Cover Page will look like this

- Do not write on the backside of the pages, we’ve left plenty of space for each question

Name:

NetID:

Do not write answers on the backside of the pages, only contents on the front pages will be scored. -- We’ve left enough space for answering each question.

Instructions (**VERY IMPORTANT TO READ****):**

- **Sit with one seat apart.**
- IF YOU HAVE ANY QUESTIONS ABOUT A QUESTION ON THE TEST, PLEASE JUST INDICATE THIS IN YOUR ANSWER. State any assumptions that you need to make to answer the question and then just answer the question as best as you can.
- **Only raise your hand to ask us a question if you believe that there is an error in the test.**
- **Leave your computations as numeric fractions** (rather than computing the decimal equivalent). More specifically, numerical answers should be left in fractional form (e.g. 9/98) or a product (or sum, etc.) of fractions (e.g. 9/98 x 1/54 x 3/7) rather than decimal form.
- Use a **pen or dark-leaded pencil** so that your scanned exam is readable.
- You have **1h and 10 minutes** to complete this exam. The exam is a closed-book / closed-notes exam.

Introduction to NLP

Why is NLP hard?

- Different languages require different methods for interpretation.

- Language is ambiguous:
  - **Syntax** - E.g. I saw a man with a telescope.
  - **Discourse** - E.g. Jack saw Sam arrive at the party. **He** had been drinking too much.
  - **Pragmatics** - E.g.
    - Magd: Do you want some lunch? / Boy, you look frazzled.
    - Claire: I just came from Collegetown Bagels.

He == Jack or Sam? Coreference Resolution

- Language is variable, many different ways to say the same thing.

Difficulties for ML Algorithms

- Language is <u>discrete.</u>
  - Words are symbols whose meaning is external to them.
- Language is <u>compositional</u>.
  - Meaning of a phrase is a function of the words that comprise it.
  - Algorithms can't operate on just word level.
- Commonsense Reasoning.
  - E.g.
    - <u>The large ball</u> crashed right through the table as <u>it</u> was made of steel.
    - The large ball crashed right through <u>the table</u> as <u>it</u> was made of styrofoam.

External Knowledge:
Steel is stronger than styrofoam.

Language Models

Language Models

- Statistical models that assign probabilities to the possible next words or a whole sequence of words. For the sentence: Mayenne ate my ______. [cake]
- Probability of next word:

$$\mathbf{P(w_4) = P(w_4 \mid w_1\ w_2\ w_3) = P(cake \mid Mayenne\ ate\ my)}$$

- Probability of sequence:

$$\mathbf{P(W) = P(w_1\ w_2\ w_3\ w_4) = P(Mayenne\ ate\ my\ cake)}$$

ISSUES:

- Need a large corpus of text.
- Data intensive.
- High Variance - Grammatical sentences can have probability 0 if they don’t occur exactly in the text.

Chain Rule for Probability of a Sequence

$$p(w_1, w_2, \ldots, w_{n-1}, w_n)$$

$$P(w_1^n) = P(w_1)P(w_2|w_1)P(w_3|w_1^2)\ldots P(w_n|w_1^{n-1})$$
$$= \prod_{k=1}^{n} P(w_k|w_1^{k-1})$$

P(Mayenne ate my cake) = P(Mayenne) x P(ate | Mayenne) x P(my | Mayenne ate) x P(cake | Mayenne ate my)

Data sparsity issue persists!

Calculated w.r.t. a corpus

N-gram Approximations

- <u>Markov assumption</u>
  - Probability of next word depends only on a limited history of previous words.

- E.g. P(cake | Mayenne ate my)

Accuracy increases as
the n in n-grams used increases.

- Unigram: $P(w_4 \mid w_1\ w_2\ w_3)$ = **$P(w_4)$ = P(cake)**
- Bigram: $P(w_4 \mid w_1\ w_2\ w_3)$ = **$P(w_4 \mid w_3)$ = P(cake | my)**
- Trigram: $P(w_4 \mid w_1\ w_2\ w_3)$ = **$P(w_4 \mid w_3\ w_2)$ = P(cake | ate my)**

N is the total no. of words in the corpus

$$P(w_4) = \frac{C(w_4)}{N} = \frac{C(\text{cake})}{N} \qquad P(w_4|w_3) = \frac{C(w_3\ w_4)}{C(w_3)} = \frac{C(\text{my cake})}{C(\text{my})}$$

Counting words in corpora

- Depends on preprocessing
  - Handling punctuation, case-sensitivity, stemming, lemmatization, contractions.
- Word types
  - Distinct words (vocabulary V)
- Word tokens
  - Words in the “running text” (instances of the vocabulary items)

E.g.

all for one and one for all.

Word types = 5, **V** = {all, for, one, and, .}

Word tokens = 8

Unknown Word Handling

- Unknown words
  - Test words not present in the training data.
  - N-gram model will assign 0 probability.

- Two ways of handling unknown words:
  - Closed vocabulary: All test words will be in a predetermined vocabulary. No unknown words.
  - **Open vocabulary**: Assume Out Of Vocabulary (OOV) words can occur.
    - Add pseudo-word <UNK> to vocabulary.
    - Gather counts for <UNK> during training.
      - Fix *k* most common word types as the vocabulary, all others are <UNK>
      - Converting some % of training tokens to <UNK>
        - First occurrence of every word
        - First occurrence of 40% of the most common words
    - When a new word is encountered in test time, convert to <UNK>

Typical method

Unseen N-grams

- N-grams whose individual tokens are in the vocabulary, but don’t occur together, (consecutively and in right order).

- E.g. Consider the tiny corpus: I had green tea.
    - V = {I, had, green, tea, .}
    - (I green), (tea green) are some unseen-bigrams.

Not consecutive

Not in order

How to Fix? **SMOOTHING**

Smoothing

cake

- Steal probability from seen n-grams so we have leftover probability for unseen n-grams.
- E.g. P(cake | Mayenne ate my)

- Unigram Smoothing (Add-k)
  - P(cake | Mayenne ate my) = P(cake)

$$P(\text{cake}) =$$

- Bigram Smoothing (Add-k)
  - P(cake | Mayenne ate my) = P(cake | my)

$$P(\text{cake}|\text{my}) =$$

If k = 1, it is **Laplacian Smoothing**.

N is the no. of word tokens in the corpus
k is the smoothing parameter
V is the vocabulary size

Smoothing

- Steal probability from seen n-grams so we have leftover probability for unseen n-grams.
- E.g. P(cake | Mayenne ate my)

- Unigram Smoothing (Add-k)
  - P(cake | Mayenne ate my) = P(cake)

$$P(\text{cake}) = \frac{C(\text{cake}) + k}{N + kV}$$

- Bigram Smoothing (Add-k)
  - P(cake | Mayenne ate my) = P(cake | my)

$$P(\text{cake}|\text{my}) = \frac{C(\text{my cake}) + k}{C(\text{my}) + kV}$$

If k = 1, it is **Laplacian Smoothing**.

N is the no. of word tokens in the corpus
k is the smoothing parameter
V is the vocabulary size

Q1. Language Modelling

Assume that the text below (actual lyrics from Diamonds by Rihanna) is provided as the (entire) training corpus for a **bigram language model**.

We're beautiful like diamonds in the sky
Eye to eye so alive
We're beautiful like diamonds in the sky
Shine bright like a diamond
Shine bright like a diamond
Shining bright like a diamond
We're beautiful like diamonds in the sky

For preprocessing:

- assume that all words are converted to **lower case**;
- do **not** add beginning (or end) of sentence markers;
- assume that punctuation, i.e. the ",", is a word type in the vocabulary.
- For contractions, assume they are 1 token (do not split 'can't' into multiple tokens)

Unknown word handling is **not** required.

How many word types and word tokens are there in the corpus?

Assume that the text below (actual lyrics from Diamonds by Rihanna) is provided as the (entire) training corpus for a **bigram language model**.

We're beautiful like diamonds in the sky
Eye to eye so alive
We're beautiful like diamonds in the sky
Shine bright like a diamond
Shine bright like a diamond
Shining bright like a diamond
We're beautiful like diamonds in the sky

We're — 3
beautiful — 3
⋮
Shining — 1

For preprocessing:

- assume that all words are converted to **lower case**;
- do **not** add beginning (or end) of sentence markers;
- assume that punctuation, i.e. the ”,”, is a word type in the vocabulary.
- For contractions, assume they are 1 token (do not split ‘can’t‘ into multiple tokens)

Unknown word handling is **not** required.

How many word types and word tokens are there in the corpus?

The number of word types is the size of the vocabulary of the corpus, the number of word tokens is the length of the corpus.

1. we’re - 3
2. beautiful - 3
3. like - 6
4. diamonds - 3
5. in - 3
6. the - 3
7. sky - 3
8. eye - 2
9. to - 1
10. so - 1
11. alive - 1
12. shine - 2
13. bright - 3
14. a - 3
15. diamond - 3
16. shining - 1

Total number of word types: V = **16**

Total number of word tokens: N = **41**

How many unseen bigrams are there in the corpus?

Assume that the text below (actual lyrics from Diamonds by Rihanna) is provided as the (entire) training corpus for a **bigram language model**.

We're beautiful like diamonds in the sky
Eye to eye so alive
We're beautiful like diamonds in the sky
Shine bright like a diamond
Shine bright like a diamond
Shining bright like a diamond
We're beautiful like diamonds in the sky

For preprocessing:

- assume that all words are converted to **lower case**;
- do **not** add beginning (or end) of sentence markers;
- assume that punctuation, i.e. the ",", is a word type in the vocabulary.
- For contractions, assume they are 1 token (do not split 'can't' into multiple tokens)

Unknown word handling is **not** required.

How many unseen bigrams are there in the corpus?

Seen bigrams*:

1. we're beautiful - 3
2. beautiful like - 3
3. like diamonds - 3
4. diamonds in - 3
5. in the - 3
6. the sky - 3
7. eye to - 1
8. to eye - 1
9. eye so - 1
10. so alive - 1
11. shine bright - 2
12. bright like - 3
13. like a - 3
14. a diamond - 3
15. shining bright - 1

Bigram is (word1, word2)

16 possible words

16 possible words

Size of vocabulary = 16

Total number of possible bigrams =

Total number of seen bigrams =

Total number of unseen bigrams =

**Assuming every line is a new sentence, with no connection to the previous one;*
If we do not make this assumption, you should treat the corpus as a single sequence

How many unseen bigrams are there in the corpus?

Seen bigrams*:

1. we’re beautiful - 3
2. beautiful like - 3
3. like diamonds - 3
4. diamonds in - 3
5. in the - 3
6. the sky - 3
7. eye to - 1
8. to eye - 1
9. eye so - 1
10. so alive - 1
11. shine bright - 2
12. bright like - 3
13. like a - 3
14. a diamond - 3
15. shining bright - 1

Bigram is (word1, word2)

16 possible words

16 possible words

Size of vocabulary = 16

Total number of possible bigrams = 16 x 16 = 16^2 = **256**

Total number of seen bigrams = **15**

Total number of unseen bigrams = 16^2-15 ~~= 256 - 15 =~~ **241**

**Assuming every line is a new sentence, with no connection to the previous one;*
If we do not make this assumption, you should treat the corpus as a single sequence

We're beautiful like diamonds in the sky
Eye to eye so alive
We're beautiful like diamonds in the sky
Shine bright like a diamond
Shine bright like a diamond
Shining bright like a diamond
We're beautiful like diamonds in the sky

For example, in the no assumptions case, we c

sky Eye
alive we're

are also bigrams

*Assuming every line is a new sentence, with no connection to the previous one;
If we do not make this assumption, you should treat the corpus as a single sequence

Using Maximum Likelihood Estimation and a bigram model, calculate P(beautiful like diamonds).

P(beautiful like diamonds) = P(beautiful) x P(like | beautiful) x P(diamonds | like)

$$= \frac{C(\text{beautiful})}{N} \times \frac{C(\text{beautiful like})}{C(\text{beautiful})} \times \frac{C(\text{like diamonds})}{C(\text{like})}$$

$$= \frac{3}{41} \times \frac{3}{3} \times \frac{3}{6}$$

⬅️ leave the result in a multiplication of fractions is acceptable

~~= 9 / 246~~

Using Add-1(Laplacian smoothing) and a bigram model, calculate P(beautiful like diamonds).

Note: When computing the sentence-initial bigram, use the unsmoothed (MLE) unigram probability.

P(beautiful like diamonds) = P(beautiful) x P(like | beautiful) x P(diamonds | like)

$$= \frac{C(\text{beautiful})}{N} \times \frac{C(\text{beautiful like}) + k}{C(\text{beautiful}) + kV} \times \frac{C(\text{like diamonds}) + k}{C(\text{like}) + kV}$$

$$= \frac{3}{41} \times \frac{3+1}{3+(1\times 16)} \times \frac{3+1}{6+(1\times 16)}$$

⬅️leave the result in a multiplication of fractions is acceptable

~~= **48 / 17138**~~

Evaluation Metrics

- Extrinsic
  - Embed Language Model (LM) in downstream task - e.g. text classification
  - Use metrics like accuracy to test performance between LMs on the final task
  - Time consuming - needs labelled data to compare true vs predicted results
- Intrinsic
  - Measure model performance independent of task
  - Perplexity (PP)
    - Intuition - Better model has a tighter fit to the test data
    - **Higher** the (estimated) **probability** of a sequence, **lower** the **perplexity**
    - When comparing models, **lower perplexity** means a **better model**

For test set $W = w_1 w_2 \ldots w_N$

$$PP(W) = P(w_1 w_2 \ldots w_N)^{-1/N}$$

N is the number of word tokens in the test set.

$$= \sqrt[N]{\frac{1}{P(w_1 w_2 \ldots w_N)}}$$

Limitations of Markov Assumption

- Hard to keep track of long range context with lower order n-grams.
- Other features of previous history like part of speech, gender, etc. could be useful for predictions, as well as co-occurrence features.

Word Embeddings

Skip-gram algorithm

Skip-gram algorithm

- Treat target and neighboring context as positive examples
- Randomly sample other words to get **negative samples**
- Use **logistic regression** to train a classifier to distinguish neighbor/not neighbor
- Use the **regression weights** as the embeddings

Skip-gram algorithm (negative exampling)

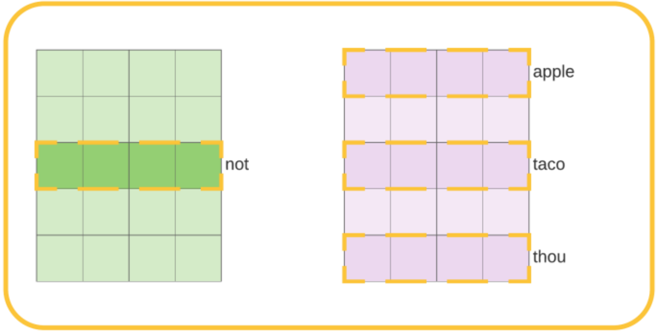
- How do we pick these negative examples?
  - Pick randomly from vocabulary

input word	output word	target
not	thou	1
not	?	0
not	?	0
not	shalt	1
not	?	0
not	?	0
not	make	1

Negative examples

Learning skip-gram embeddings

input word	output word	target	input • output	sigmoid()	Error
not	thou	1	0.2	0.55	0.45
not	apple	0	-1.11	0.25	-0.25
not	taco	0	0.74	0.68	-0.68



Update model parameters

Learning skip-gram embeddings

Goal: maximize the corpus probability

$$\arg\max_{\theta} \prod_{(w,c)\in D} p(c|w;\theta)$$

where:

$$\boxed{p(c|w;\theta) = \frac{e^{v_c \cdot v_w}}{\sum_{c' \in C} e^{v_{c'} \cdot v_w}}}$$

if d is the dimensionality of the vectors, we have parameters

$$d \times |V| + d \times |C|$$

Contextualized Word Embeddings (Representations)

ELMo (Contextualized Representations) -- motivation

- **Problem:** Word embeddings are applied in a context free manner

open a bank account   on the river bank

[0.3, 0.2, -0.8, …]

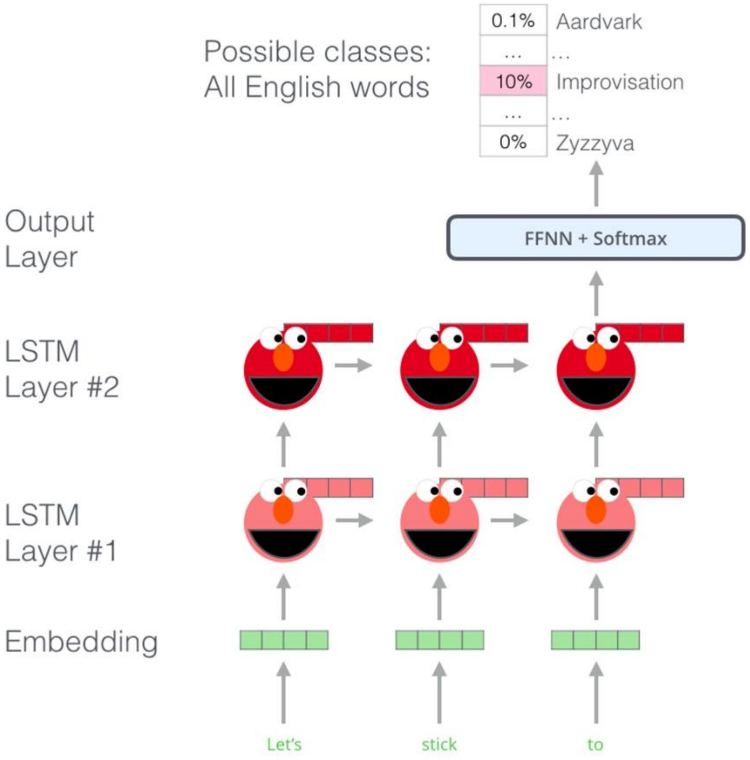
- **Solution:** Train *contextual* representations on text corpus

[0.9, -0.2, 1.6, …]   [-1.9, -0.4, 0.1, …]

open a bank account   on the river bank

ELMo (Contextualized Representations)

- Key Idea: Train a deep bidirectional LSTM as a language model then use context vectors for each word as pre-trained word vectors



ELMo vs. GloVe

	Source	Nearest Neighbors
GloVe	play	playing, game, games, played, players, plays, player, Play, football, multiplayer
biLM	Chico Ruiz made a spectacular play on Alusik 's grounder {...}	Kieffer , the only junior in the group , was commended for his ability to hit in the clutch , as well as his all-round excellent play .
	Olivia De Havilland signed to do a Broadway play for Garson {...}	{...} they were actors who had been handed fat roles in a successful play , and had talent enough to fill the roles competently , with nice understatement .

Table 4: Nearest neighbors to "play" using GloVe and the context embeddings from a biLM.

ELMo is...

- Contextual: The representation for each word depends on the entire context in which is is used
- Deep: The word representations combine all layers of a deep pre-trained neural network
- ...

Neural Networks

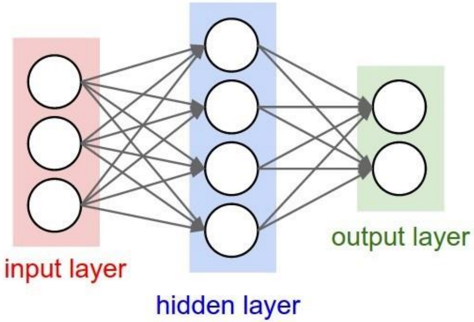
Probabilistic Output from Neural Nets

- What if we want the output to be a probability distribution over possible outputs?
- Normalize the output activations using softmax:

$$y = \mathrm{softmax}(\mathbf{o})$$

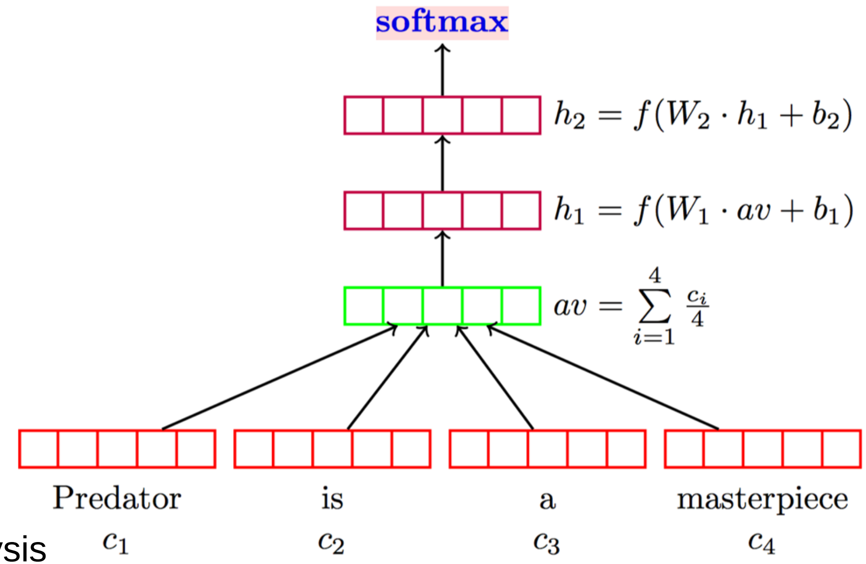
$$\mathrm{softmax}(\mathbf{o}_i) = \frac{\exp(\mathbf{o}_i)}{\sum_{j=1}^{k} \exp(\mathbf{o}_j)}$$

  - Where $\boldsymbol{o}$ is the output layer
  - Usually: no non-linearity before softmax



Neural Bag-of-words (for sentiment classification)

Deep Averaging Networks



IMDB sentiment analysis

* It not very common to put a non-linearity before a softmax.

[Iyyer et al. 2015; Wang and Manning 2012]

FFNN (for Word Pair Classification)

- Goal: build a classifier that given a pair of words, classify if they are the full name of a person or not
- The classifier is a multi-layer-perceptron with three layers
- Make a drawing!
- Write the matrix notation, including dimensionality of matrices (choose as you wish, and as needed)
- What are the parameters to be learned

Inputs: x_l, x_r

Input vocabulary: $\mathcal{V}$

Embedding function: $\phi : \mathcal{V} \rightarrow \mathbb{R}^{256}$

Weight matrices: $\mathbf{W}^1, \mathbf{W}^2, \mathbf{W}^3$

Bias vectors: $\mathbf{b}^1, \mathbf{b}^2, \mathbf{b}^3$

Operations: $2 \times \sigma : \mathbb{R}^* \rightarrow \mathbb{R}^*, 1 \times \text{softmax}$

256

$$h_1 = \sigma(\underbrace{W_1}_{256\cdot 512} \underbrace{[\phi(x_l);\phi(x_r)]}_{512} + \underbrace{b^1}_{256})$$

128

$$h_2 = \sigma(\underbrace{W^2}_{128\cdot 256} \cdot \underbrace{h_1}_{256} + \underbrace{b^2}_{128})$$

2

$$h_3 = W^3 \cdot h_2 + b^3$$

Normlize

$p(y) = \text{Softmax}(h_3)$

Binary-class

Recurrent Neural Networks

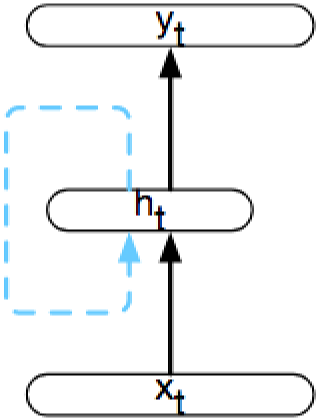
RNNs

- Any network that contains a **cycle**
- In the general case, networks with cycles are hard to train
- Not the case for simple RNNs
- Hidden layer activation depends on the

  (1) input layer;

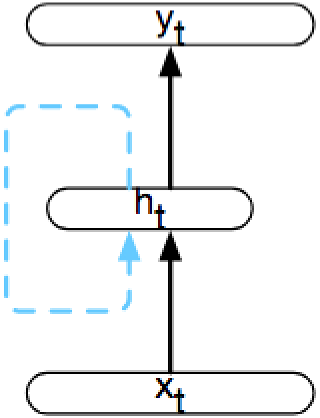
  (2) the **activation of the hidden layer**

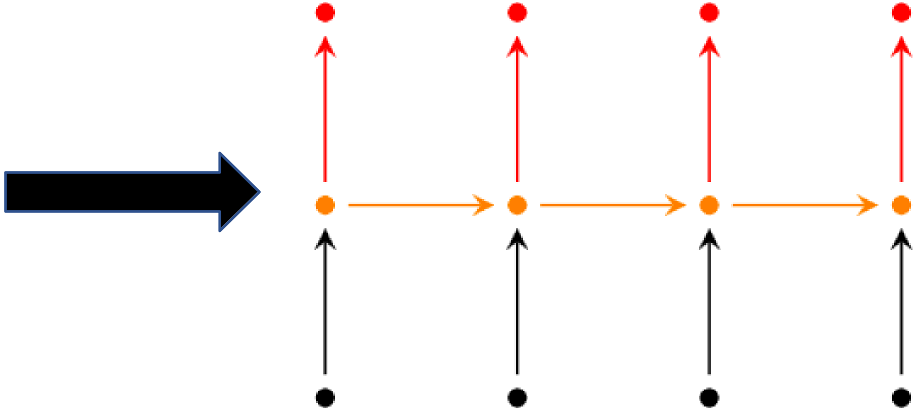
  from the previous timestep



RNN unfolded in time (forward computation)

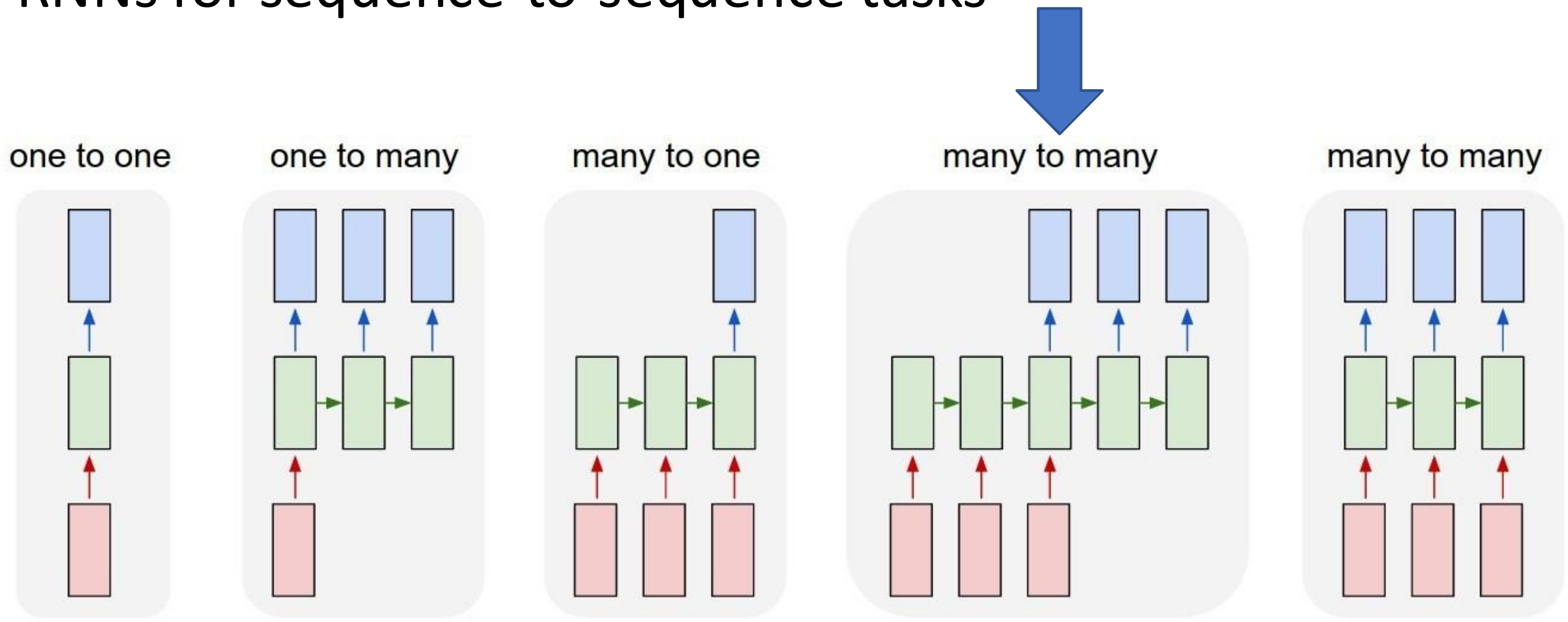
In this visualization of RNNs, **each node represents an entire layer; each edge represents a weight matrix**



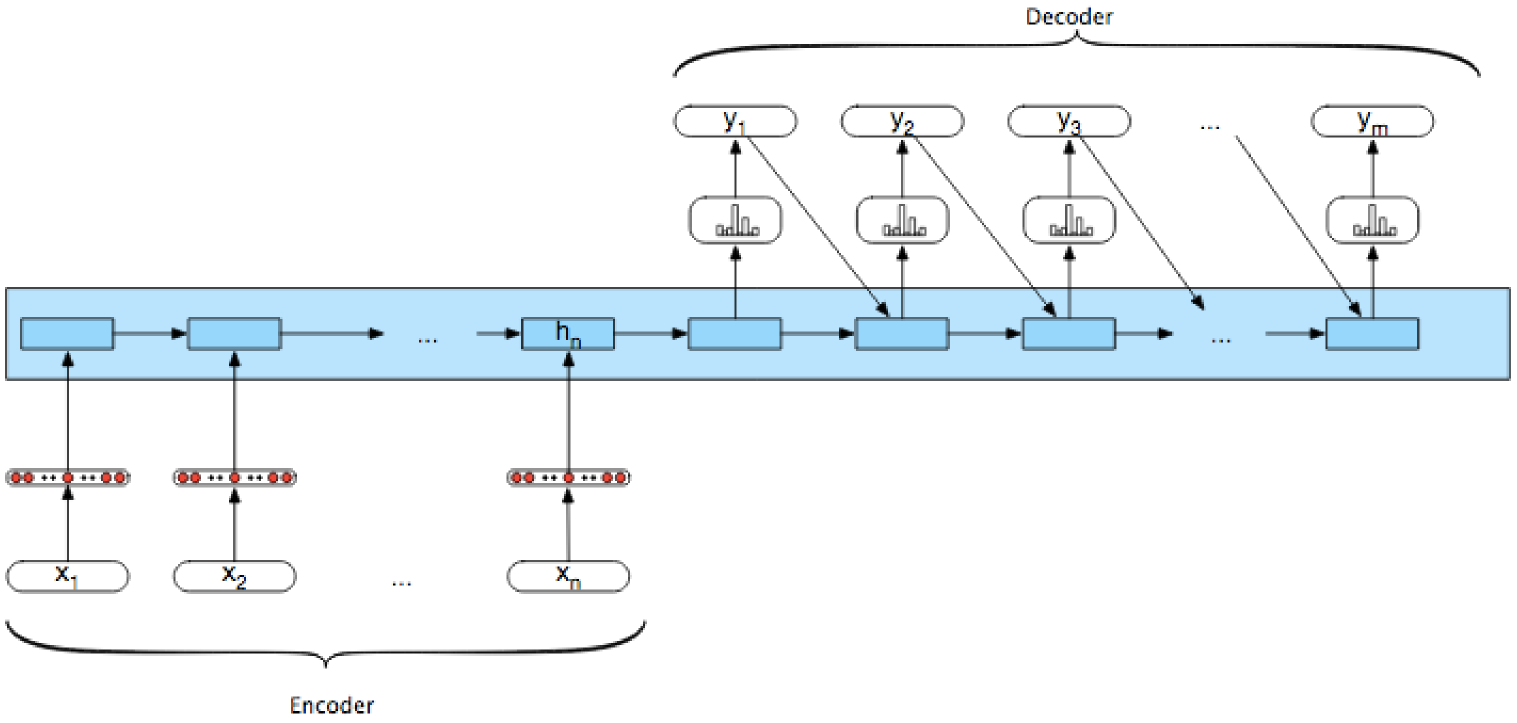


Encoder-decoder models

- RNNs for sequence-to-sequence tasks



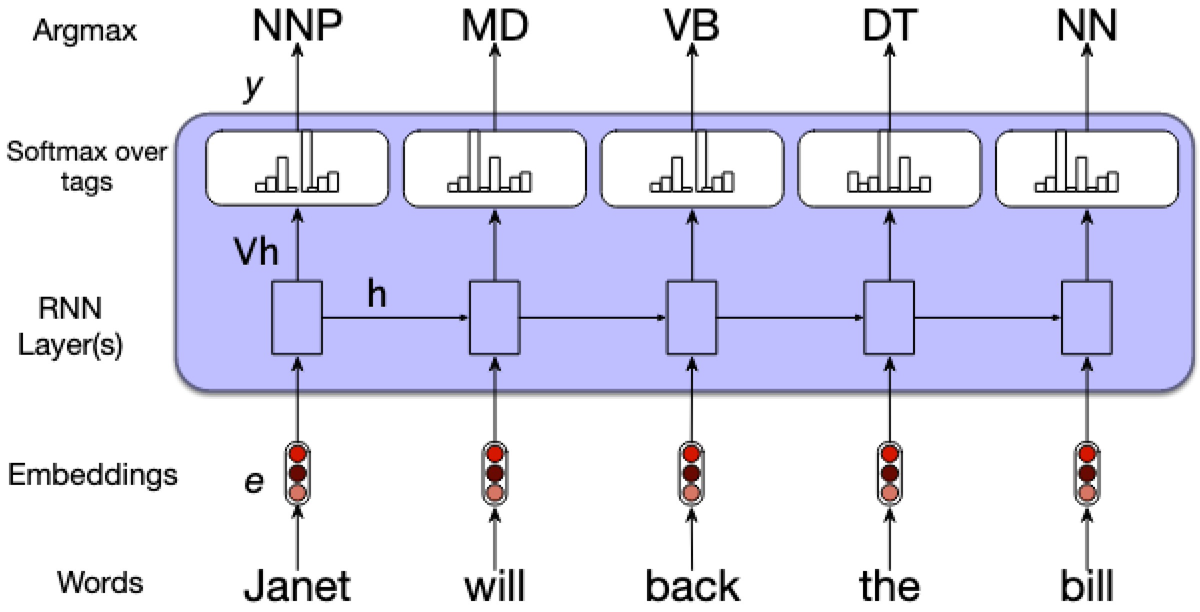
General RNN-based encoder-decoder architecture



RNN Applications

RNN Applications (Sequence labeling task)

- POS tagging

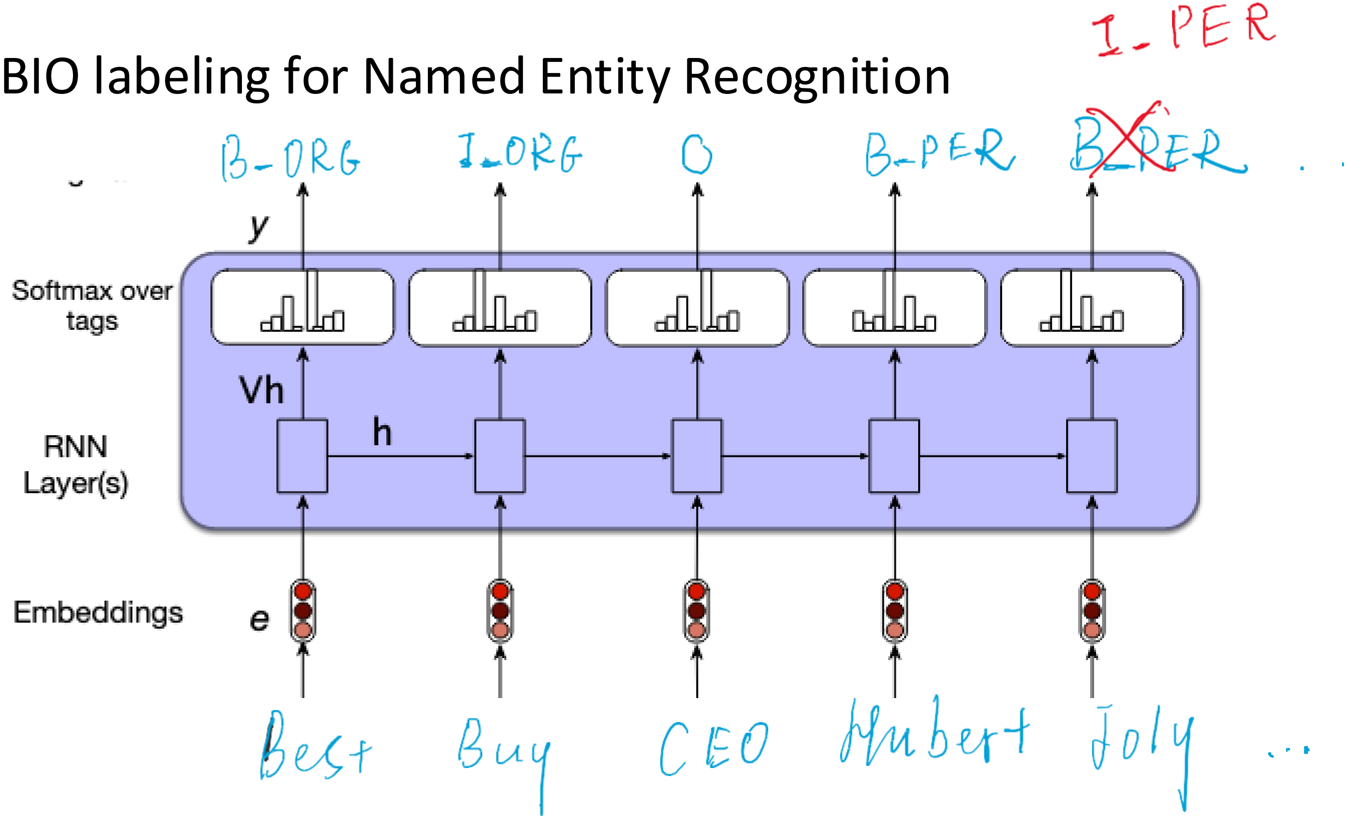


RNN Applications (Sequence labeling task)

- BIO labeling based applications
  - Chunking
  - Opinion extraction
  - NE recognition
  - …

RNN Applications (Sequence labeling task)

- E.g., BIO labeling for Named Entity Recognition

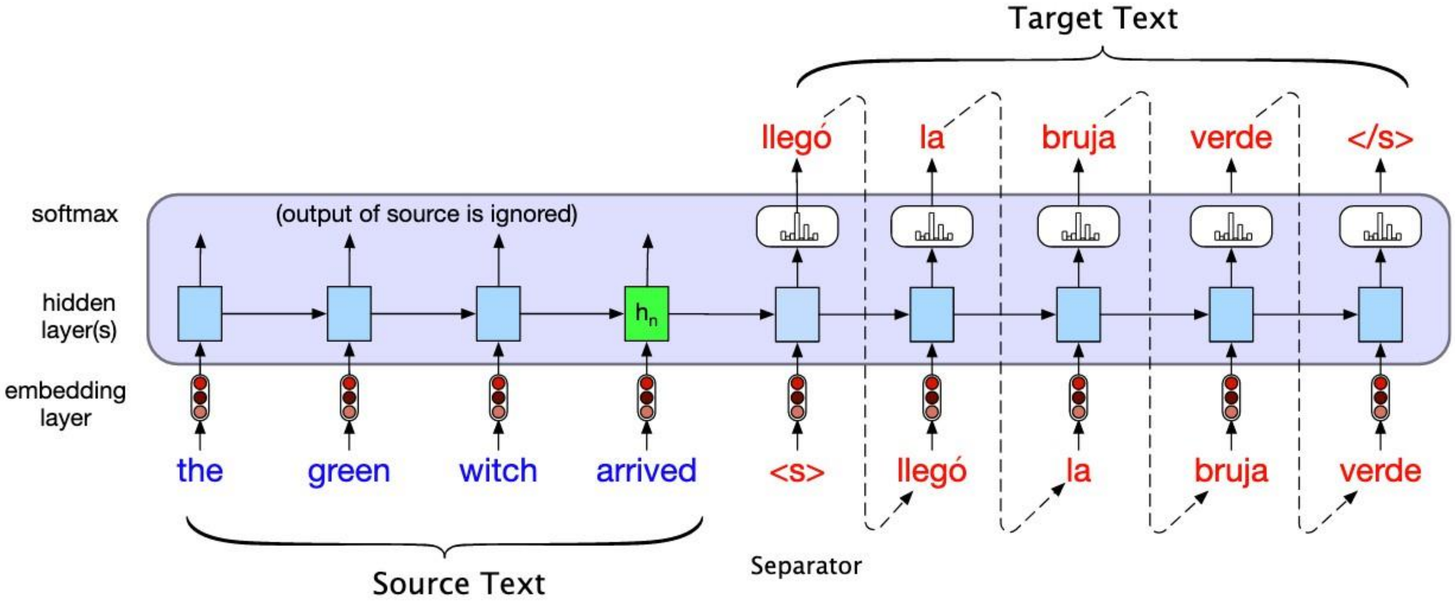


RNN Applications (Encoder-decoder models)

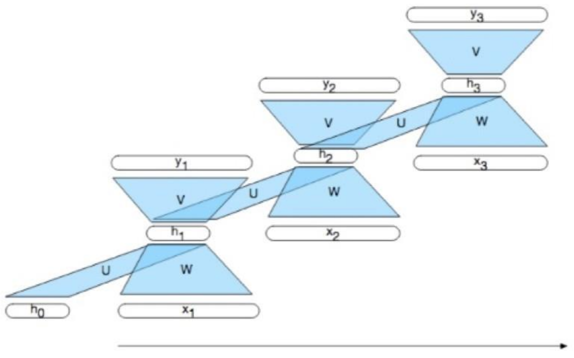
- Mainly generation-based tasks
  - Machine translation
  - Abstractive summarization
  - Image captioning
  - Chatbots
  - Response generation
  - …

RNN Applications (Encoder-decoder models)

- E.g., machine translation (at inference time)



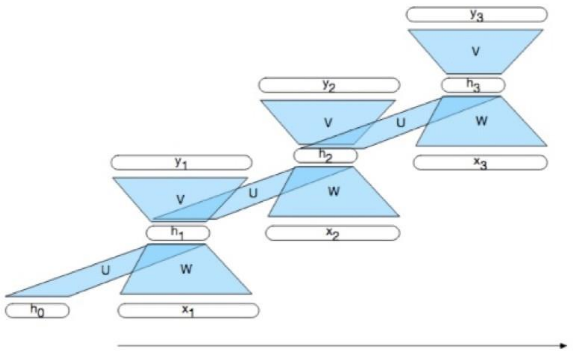
Example Question



1. Consider the task of using a simple Recurrent Neural Network (RNN) (shown above) for the task of Language Modeling (i.e., to predict the next word in a sequence of words).

(3pt) what does each output vector y_i represent this task? what is its dimension

Example Question

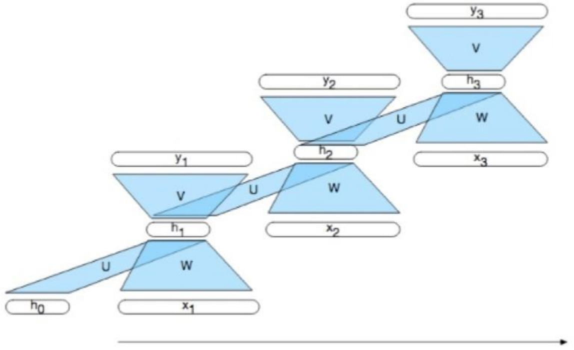


1. Consider the task of using a simple Recurrent Neural Network (RNN) (shown above) for the task of Language Modeling (i.e., to predict the next word in a sequence of words).

(3pt) what does each output vector y_i represent this task? what is its dimension

Answer: each y_i is the probability distribution over the vocabulary. Dimension: 1 x length of the vocabulary

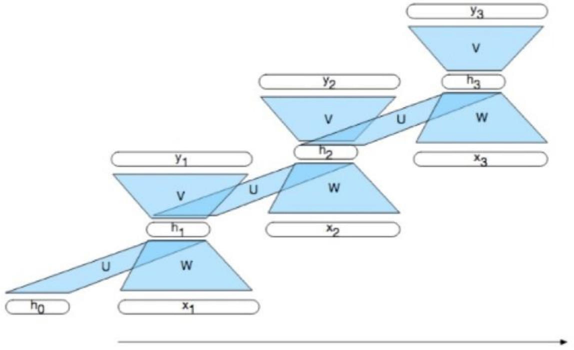
Example Question



1. Consider the task of using a simple Recurrent Neural Network (RNN) (shown above) for the task of Language Modeling (i.e., to predict the next word in a sequence of words).

(3 pts) What is the purpose of the h_i vectors?

Example Question



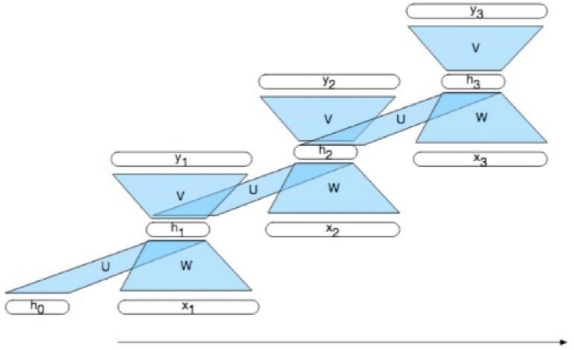
1. Consider the task of using a simple Recurrent Neural Network (RNN) (shown above) for the task of Language Modeling (i.e., to predict the next word in a sequence of words).

(3 pts) What is the purpose of the h_i vectors?

h is the hidden vector.

Trained to learn useful features/representation of the input history.

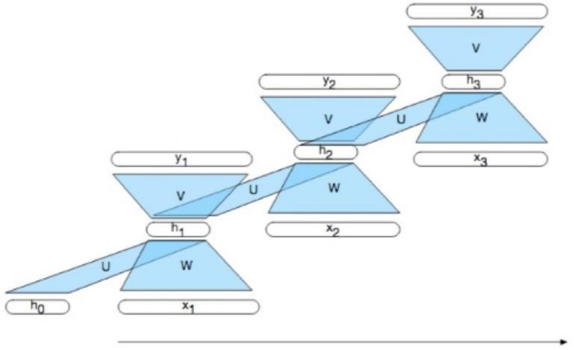
Example Question



1. Consider the task of using a simple Recurrent Neural Network (RNN) (shown above) for the task of Language Modeling (i.e., to predict the next word in a sequence of words).

(3 pts) Is a *softmax* operator needed for this task? Explain your answer.

Example Question



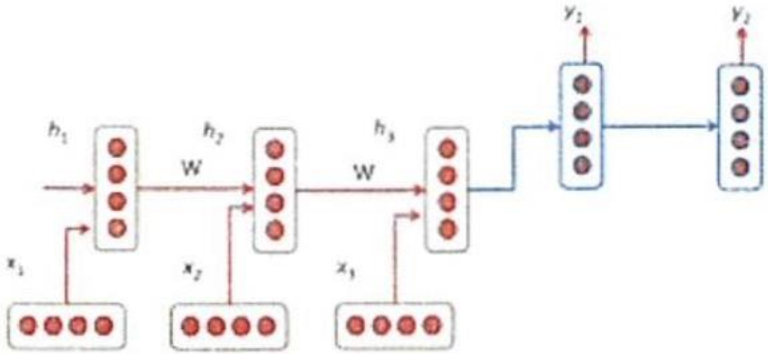
1. Consider the task of using a simple Recurrent Neural Network (RNN) (shown above) for the task of Language Modeling (i.e., to predict the next word in a sequence of words).

(3 pts) Is a *softmax* operator needed for this task? Explain your answer.

Yes, softmax is needed since we want
a probability distribution over the vocabulary.

Example Question

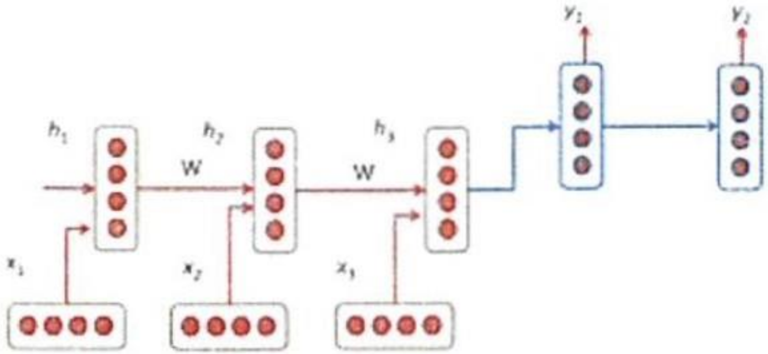
(10pts) Consider the neural network architecture on the right. Assuming the appropriate amount of data is available.



Is model based on this architecture better suited for abstractive (need not contain phrases from the original text) or extractive summarization (summary is comprised of sentences taken from the original text), and briefly explain?

Example Question

(10pts) Consider the neural network architecture on the right. Assuming the appropriate amount of data is available.



Is model based on this architecture better suited for abstractive (need not contain phrases from the original text) or extractive summarization (summary is comprised of sentences taken from the original text), and briefly explain?

Abstractive summarization, encoder to represent the doc, decoder to generate the summary.

Example Question

(3 pts) (True/False. Explain your answer.) Contextual word embeddings (like Elmo or BERT) are helpful for resolving lexical semantic ambiguities (i.e., word sense ambiguity).

Example Question

(3 pts) (True/False. Explain your answer.) Contextual word embeddings (like Elmo or BERT) are helpful for resolving lexical semantic ambiguities (i.e., word sense ambiguity).

True. the embedding captures the context to distinguish among possible senses.