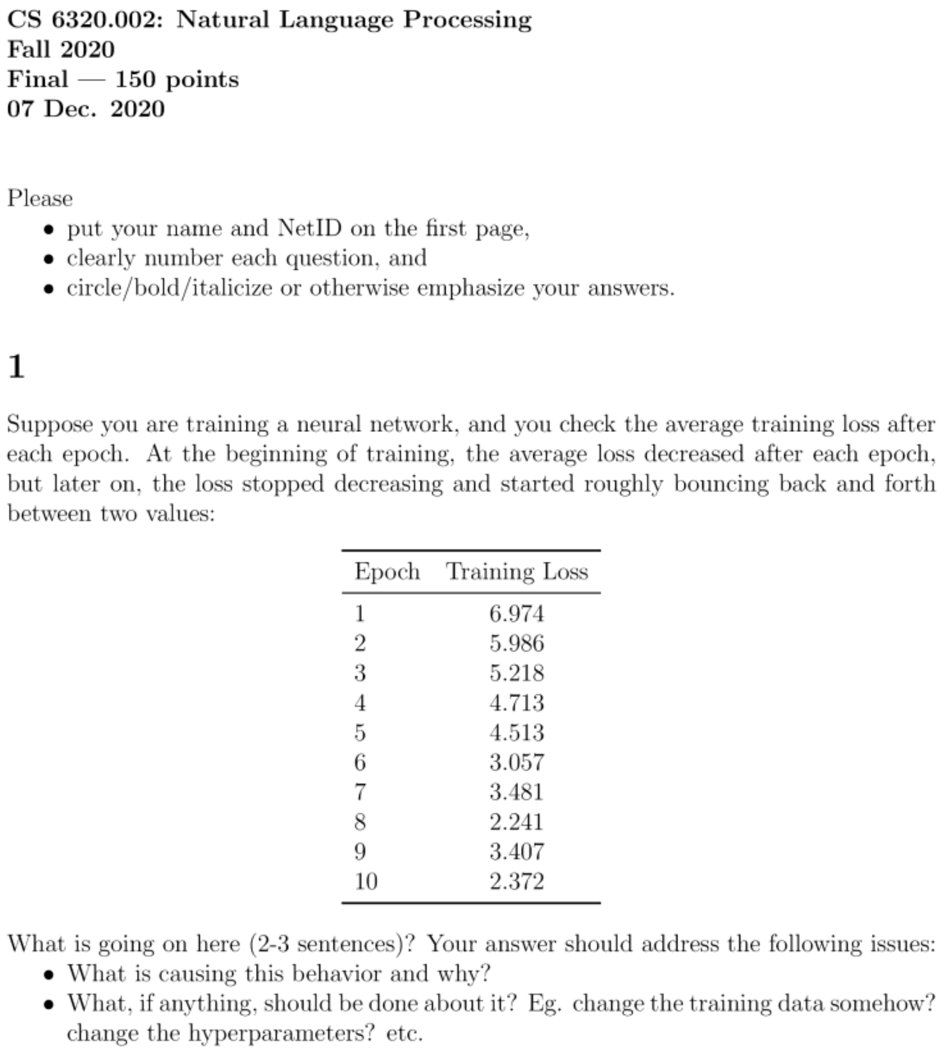
rxp190007

07 December 2020 09:31

rxp190007



CS 6320.002: Natural Language Processing
Fall 2020
Final — 150 points
07 Dec. 2020

Please

- put your name and NetID on the first page,
- clearly number each question, and
- circle/bold/italicize or otherwise emphasize your answers.

1

Suppose you are training a neural network, and you check the average training loss after each epoch. At the beginning of training, the average loss decreased after each epoch, but later on, the loss stopped decreasing and started roughly bouncing back and forth between two values:

Epoch	Training Loss
1	6.974
2	5.986
3	5.218
4	4.713
5	4.513
6	3.057
7	3.481
8	2.241
9	3.407
10	2.372

What is going on here (2-3 sentences)? Your answer should address the following issues:

- What is causing this behavior and why?
- What, if anything, should be done about it? Eg. change the training data somehow? change the hyperparameters? etc.

Possible causes :

1. Learning rate too high, after each epoch SGD overshoots and then again tries to correct itself.

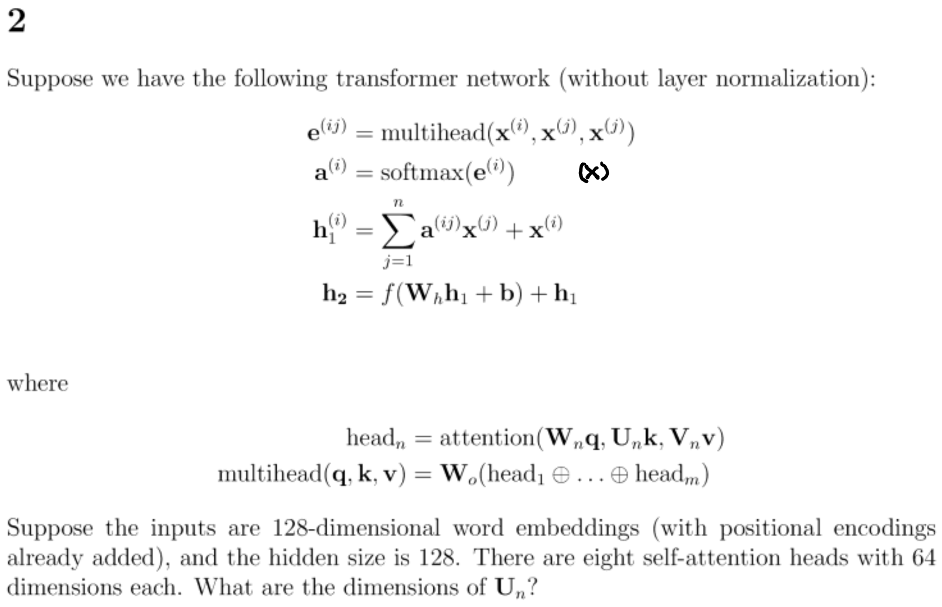
Solution try reducing learning rate.

2. Noisy data

We can address noisy data by performing preprocessing on it.

Page 2

07 December 2020 09:32



2

Suppose we have the following transformer network (without layer normalization):

$$\mathbf{e}^{(ij)} = \text{multihead}(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}, \mathbf{x}^{(j)})$$
$$\mathbf{a}^{(i)} = \text{softmax}(\mathbf{e}^{(i)}) \qquad (\mathrm{x})$$
$$\mathbf{h}_1^{(i)} = \sum_{j=1}^{n} \mathbf{a}^{(ij)}\mathbf{x}^{(j)} + \mathbf{x}^{(i)}$$
$$\mathbf{h_2} = f(\mathbf{W}_h\mathbf{h}_1 + \mathbf{b}) + \mathbf{h}_1$$

where

$$\text{head}_n = \text{attention}(\mathbf{W}_n\mathbf{q}, \mathbf{U}_n\mathbf{k}, \mathbf{V}_n\mathbf{v})$$
$$\text{multihead}(\mathbf{q},\mathbf{k},\mathbf{v}) = \mathbf{W}_o(\text{head}_1 \oplus \ldots \oplus \text{head}_m)$$

Suppose the inputs are 128-dimensional word embeddings (with positional encodings already added), and the hidden size is 128. There are eight self-attention heads with 64 dimensions each. What are the dimensions of $\mathbf{U}_n$?

Given 8 self attention heads. with 64 dim

Hidden size : 128 dim   X : 128 dim

as $k \rightarrow [128 \text{ dim}]$

In paper they use $\left(d_k = \frac{d_{model}}{h}\right)$

From paper we see : $U \rightarrow$ depends on d_{model}, d_k
(128) (64)

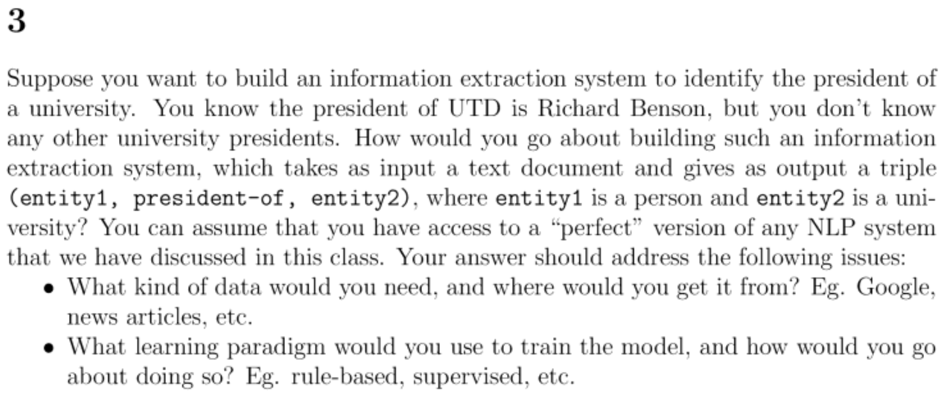
So, we can have :

$k : (128,1)$

$U : (64,128)$

Page 3

07 December 2020 09:32



3

Suppose you want to build an information extraction system to identify the president of a university. You know the president of UTD is Richard Benson, but you don't know any other university presidents. How would you go about building such an information extraction system, which takes as input a text document and gives as output a triple `(entity1, president-of, entity2)`, where `entity1` is a person and `entity2` is a university? You can assume that you have access to a "perfect" version of any NLP system that we have discussed in this class. Your answer should address the following issues:

- What kind of data would you need, and where would you get it from? Eg. Google, news articles, etc.
- What learning paradigm would you use to train the model, and how would you go about doing so? Eg. rule-based, supervised, etc.

It is a relation extraction task where we are trying to extract
(entity1, president-of, entity2)

PER - ORG

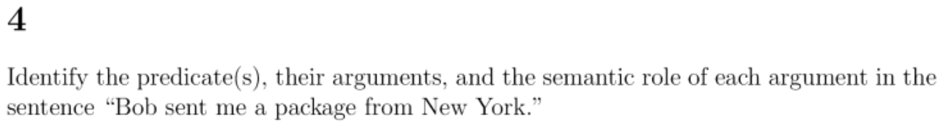
Data : Specifically universities related text corpus probably from wikipedia info boxes

Learning paradigm :

1. As wikipedia info boxes give us a structured knowledge base. We can build a rule/pattern based approach with frames being filled by the values captured from the wikipedia infobox data.

Page 4

07 December 2020 09:32



4

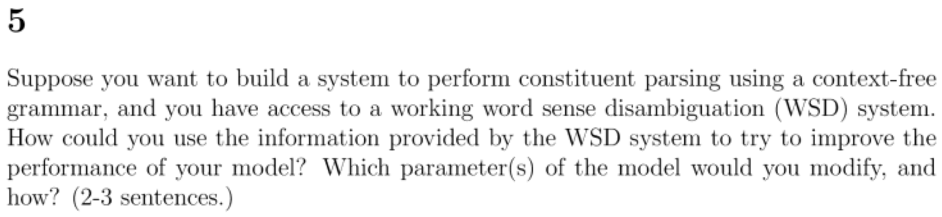
Identify the predicate(s), their arguments, and the semantic role of each argument in the sentence “Bob sent me a package from New York.”

Predicate - send

Bob - Agent
Me - goal
Package - theme
From New York - source

Page 5

07 December 2020 09:32



5

Suppose you want to build a system to perform constituent parsing using a context-free grammar, and you have access to a working word sense disambiguation (WSD) system. How could you use the information provided by the WSD system to try to improve the performance of your model? Which parameter(s) of the model would you modify, and how? (2-3 sentences.)

WSD for constituency parsing :

"The fact that there are many grammatically correct but semantically unreasonable parses for naturally occurring sentences is an irksome problem that affects all parsers.

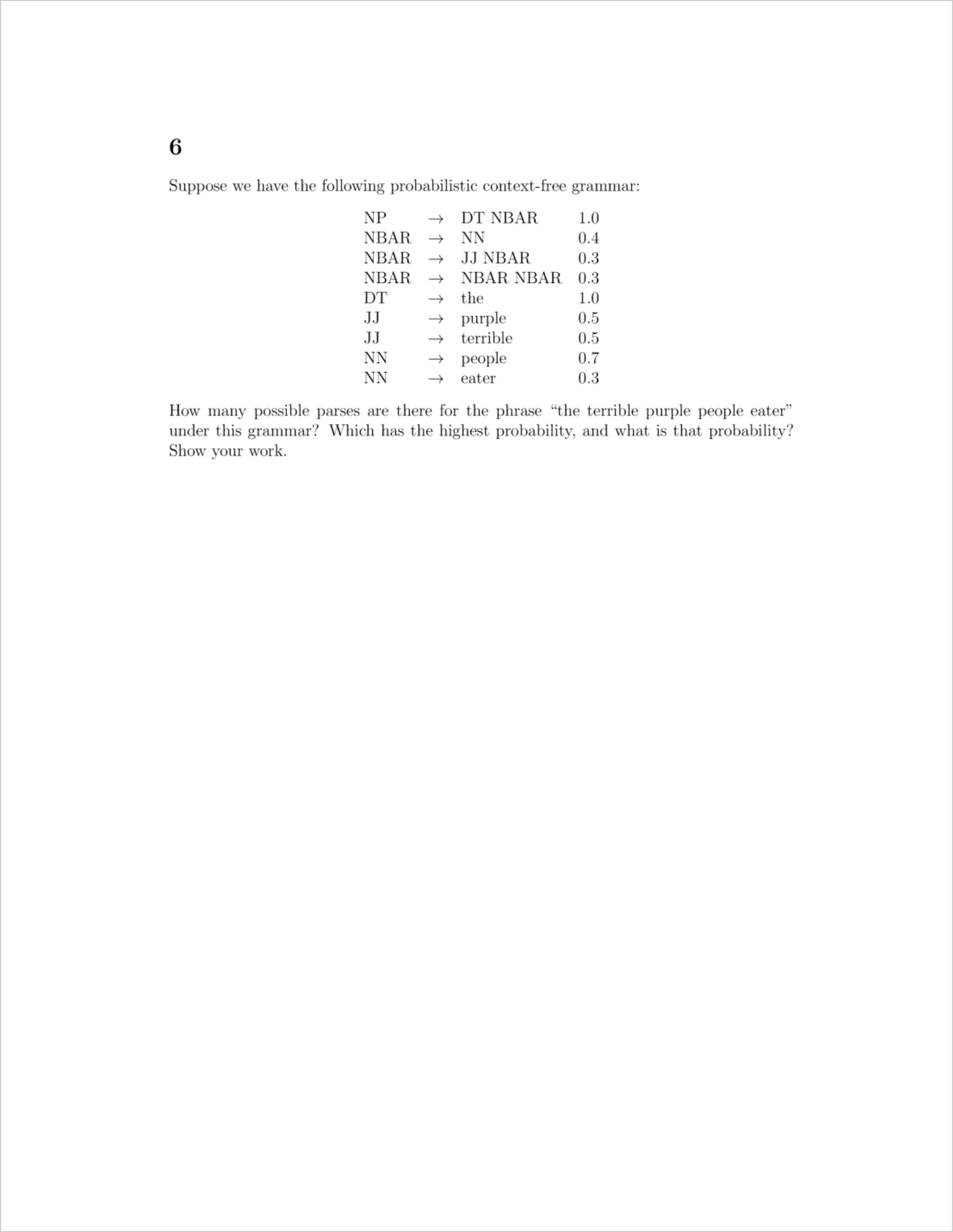
Effective disambiguation algorithms require statistical, semantic, and contextual knowledge sources that vary in how well they can be integrated into parsing algorithms." - From text book.

Here WSD system can help us generating effective parse.

As context of a word plays a major role in determining the phrase structure and there by parse.
We can **replace word by the sense of the word** in aiding to generate a better semantic parse.

Page 6

07 December 2020 09:32



6

Suppose we have the following probabilistic context-free grammar:

NP	→	DT NBAR	1.0
NBAR	→	NN	0.4
NBAR	→	JJ NBAR	0.3
NBAR	→	NBAR NBAR	0.3
DT	→	the	1.0
JJ	→	purple	0.5
JJ	→	terrible	0.5
NN	→	people	0.7
NN	→	eater	0.3

How many possible parses are there for the phrase "the terrible purple people eater" under this grammar? Which has the highest probability, and what is that probability? Show your work.

Given CFG is not in CNF.

Converting the rules into CNF by following the procedure gives us:

NP --> DT NBAR 1.0
NBAR --> NN 0.4
NBAR --> JJ NBAR 0.3
NBAR --> NBAR NBAR 0.3
DT --> the 1.0
JJ --> purple 0.5
JJ --> terrible 0.5
NN --> people 0.7
NN --> eater 0.3

1. Removing nullable variables
2. Removing unit variables

NP --> DT NBAR 1.0
NBAR --> NN 0.4
NBAR --> JJ NBAR 0.3
NBAR --> NBAR NBAR 0.3
DT --> the 1.0
JJ --> purple 0.5
JJ --> terrible 0.5
NN --> people 0.7
NN --> eater 0.3

NBAR --> people 0.4*0.7 = 0.28
NBAR --> eater 0.3*0.4 = 0.12

3. Removing offending variables

NP --> DT NBAR 1.0
NBAR --> JJ NBAR 0.3
NBAR --> NBAR NBAR 0.3
DT --> the 1.0
JJ --> purple 0.5
JJ --> terrible 0.5
NN --> people 0.7
NN --> eater 0.3
NBAR --> people 0.28
NBAR --> eater 0.12

	the	terrible	purple	people	eater
the	DT 1.0	-			
terrible		JJ 0.5	-		
purple			JJ 0.5	NBAR 0.3*0.5*0.28 = 0.042	
people				NN 0.7, NBAR 0.28	NBAR 0.3*0.28*0.12=0.0101
eater					NN 0.3, NBAR 0.12

	the	terrible	purple	people	eater
the	DT 1.0	-	-		
terrible		JJ 0.5	-	NBAR 0.3*0.5*0.042 =0.0063	
purple			JJ 0.5	NBAR 0.042	NBAR 0.3*0.5*0.0101=0.0015, NBAR 0.3* 0.042*0.12 = 0.0015
people				NN 0.7, NBAR 0.28	NBAR 0.0101
eater					NN 0.3, NBAR 0.12

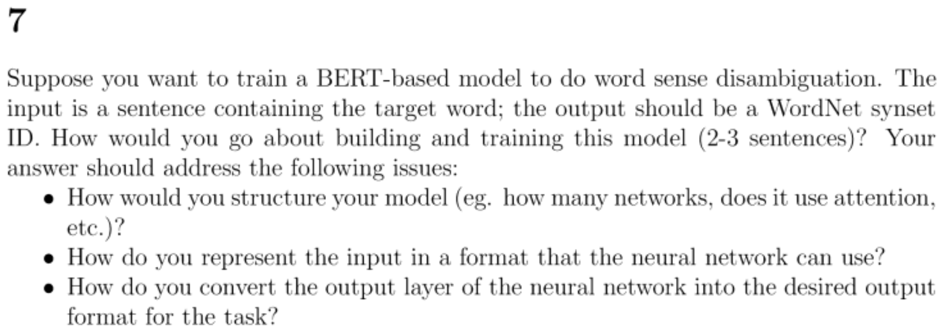
	the	terrible	purple	people	eater
the	DT 1.0	-	-	NP 0.0063	
terrible		JJ 0.5	-	NBAR 0.0063	NBAR 0.3*0.5*0.0015 = 0.0002, NBAR 0.3*0.5*0.0015=0.0002, NBAR 0.3*0.0063*0.12 = 0.0002
purple			JJ 0.5	NBAR 0.042	NBAR 0.0015, NBAR 0.0015
people				NN 0.7, NBAR 0.28	NBAR 0.0101
eater					NN 0.3, NBAR 0.12

	the	terrible	purple	people	eater
the	DT 1.0	-	-	NP 0.0063	NP, NP, NP 0.0002
terrible		JJ 0.5	-	NBAR 0.0063	NBAR 0.0002,NBAR 0.0002, NBAR 0.0002
purple			JJ 0.5	NBAR 0.042	NBAR 0.0015, NBAR 0.0015
people				NN 0.7, NBAR 0.28	NBAR 0.0101
eater					NN 0.3, NBAR 0.12

There are 3 possible parses for this sentence

Page 7

07 December 2020 09:32



7

Suppose you want to train a BERT-based model to do word sense disambiguation. The input is a sentence containing the target word; the output should be a WordNet synset ID. How would you go about building and training this model (2-3 sentences)? Your answer should address the following issues:

- How would you structure your model (eg. how many networks, does it use attention, etc.)?
- How do you represent the input in a format that the neural network can use?
- How do you convert the output layer of the neural network into the desired output format for the task?

BERT gives us pretrained embeddings for words.
As it is based on strategy of fine tuning where we add task specific model on top of it and train it end-end.
BERT to an extent helps us in reducing the effort put in to generate good features.

As we have already seen a problem of WSD using RNN in sample final.
We can use that approach here.

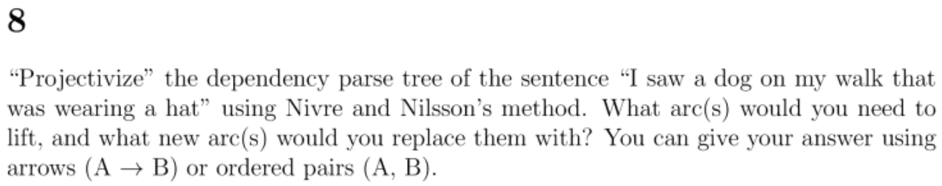
We follow the same design as described in sample final for constructing RNN. But we use embeddings generated by BERT as starting point.

So,

BERT Embeddings + RNN (described sample final)

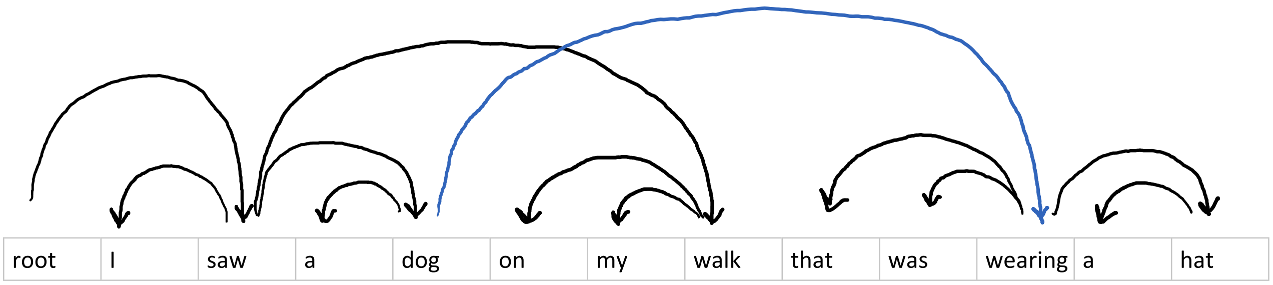
Page 8

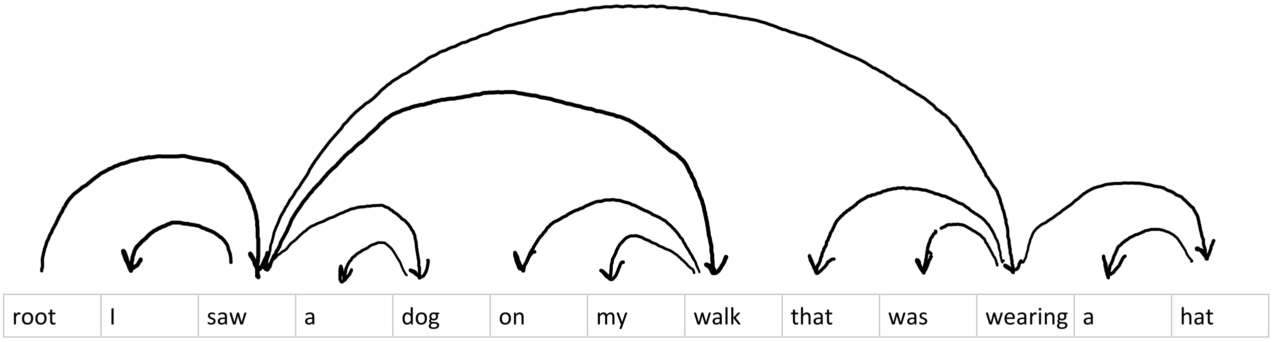
07 December 2020 09:32



8

“Projectivize” the dependency parse tree of the sentence “I saw a dog on my walk that was wearing a hat” using Nivre and Nilsson’s method. What arc(s) would you need to lift, and what new arc(s) would you replace them with? You can give your answer using arrows (A → B) or ordered pairs (A, B).





Page 9

07 December 2020 09:32

9

Suppose we have the following single-layer, unidirectional recurrent network:

$$\overrightarrow{\mathbf{h}}_1^{(t)} = f_1(\mathbf{W}_1\mathbf{x}^{(t)} + \mathbf{U}_1\overrightarrow{\mathbf{h}}_1^{(t-1)})$$
$$\hat{\mathbf{y}}^{(t)} = f_y(\mathbf{V}\overrightarrow{\mathbf{h}}_1^{(t)})$$

We want to improve this model to be more like the Google Translate encoder network. We want to make it deep by adding an additional layer and also make the first layer bidirectional. We also want to add residual connections for every pair of adjacent layers. What are the equations for this new network? Keep your variable names and notation as close as possible to those in the original network.

We have single layer :

$$\vec{h}_1^{(t)} = f_1(W_1 x^{(t)} + U_1 \vec{h}_1^{(t-1)})$$
$$\hat{y}^{(t)} = f_y(V \vec{h}_1^{(t)})$$

Requirements :

1. Deep by adding additional layer
2. First layer bidirectional
3. Residual connections.

Layer-1 : Bidirectional :

$$\vec{h}_t^{(1)} = f_1(Wx_t + U\vec{h}_{t-1} + b)$$
$$\overleftarrow{h}_t^{(1)} = f_1(Wx_t + U\vec{h}_{t+1} + b)$$
$$y_t = f_y(V \cdot (\vec{h}_t^{(1)} \oplus \overleftarrow{h}_t^{(1)}))$$

Considering concatenated hidden state

$$\vec{h}_t^{(1)} = \vec{h}_t^{(1)} \oplus \overleftarrow{h}_t^{(1)}$$

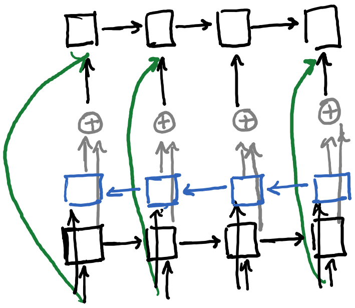
$h_t^{(1)}$ ↰ LAYER NO.

Layer-2 : To make it deep :

$$h_t^{(2)} = f_2(W_2 \hat{h}_t^{(1)} + U_2 h_{t-1}^{(2)} + b_2)$$

Assuming residual connections as shown in green

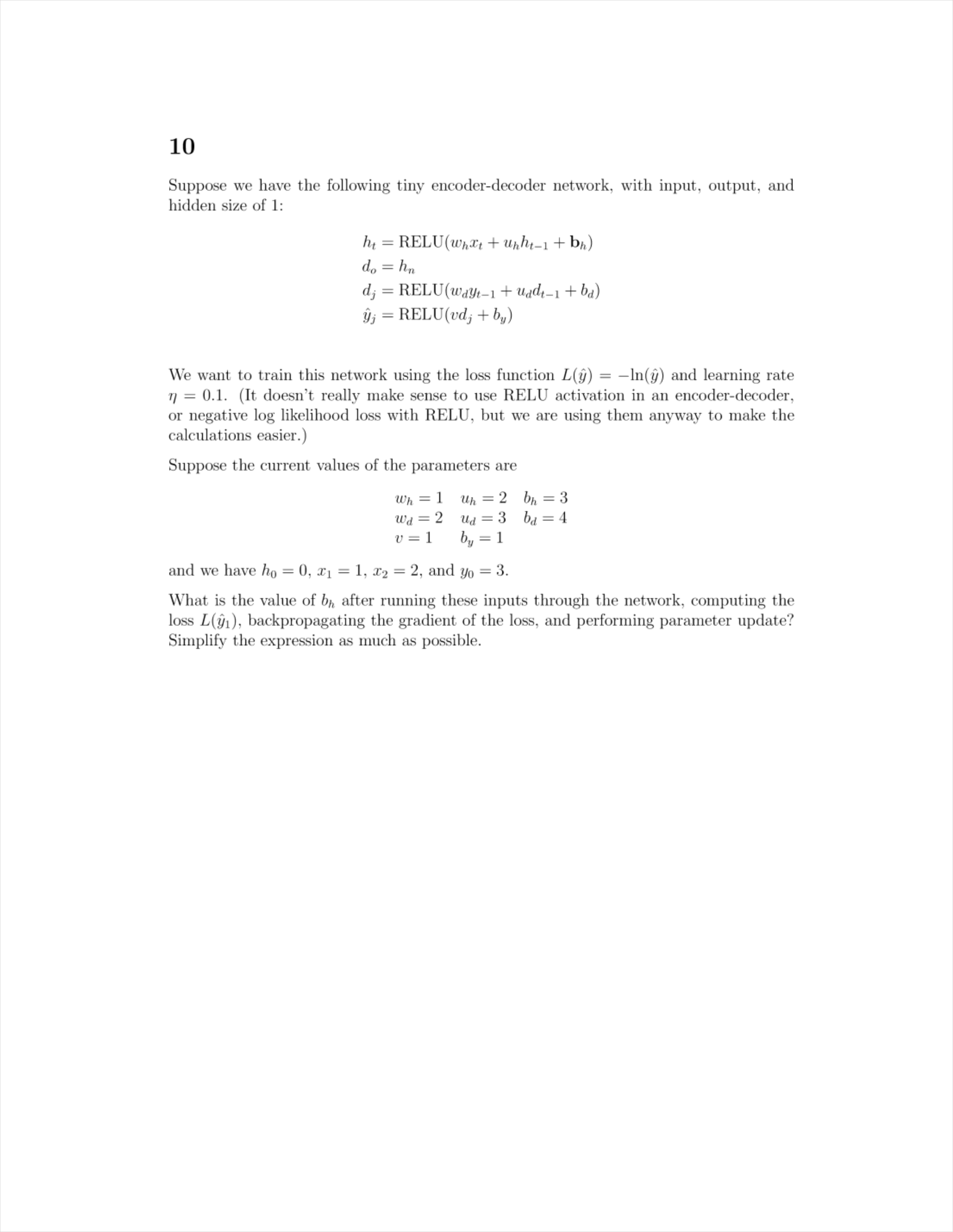
ELEMENT WISE **ADDITION** :



$$\hat{h}_t^{(1)} = h_t^{(1)} + x_t$$

Page 10

07 December 2020 09:32



10

Suppose we have the following tiny encoder-decoder network, with input, output, and hidden size of 1:

$$\begin{aligned} h_t &= \mathrm{RELU}(w_h x_t + u_h h_{t-1} + \mathbf{b}_h) \\ d_o &= h_n \\ d_j &= \mathrm{RELU}(w_d y_{t-1} + u_d d_{t-1} + b_d) \\ \hat{y}_j &= \mathrm{RELU}(v d_j + b_y) \end{aligned}$$

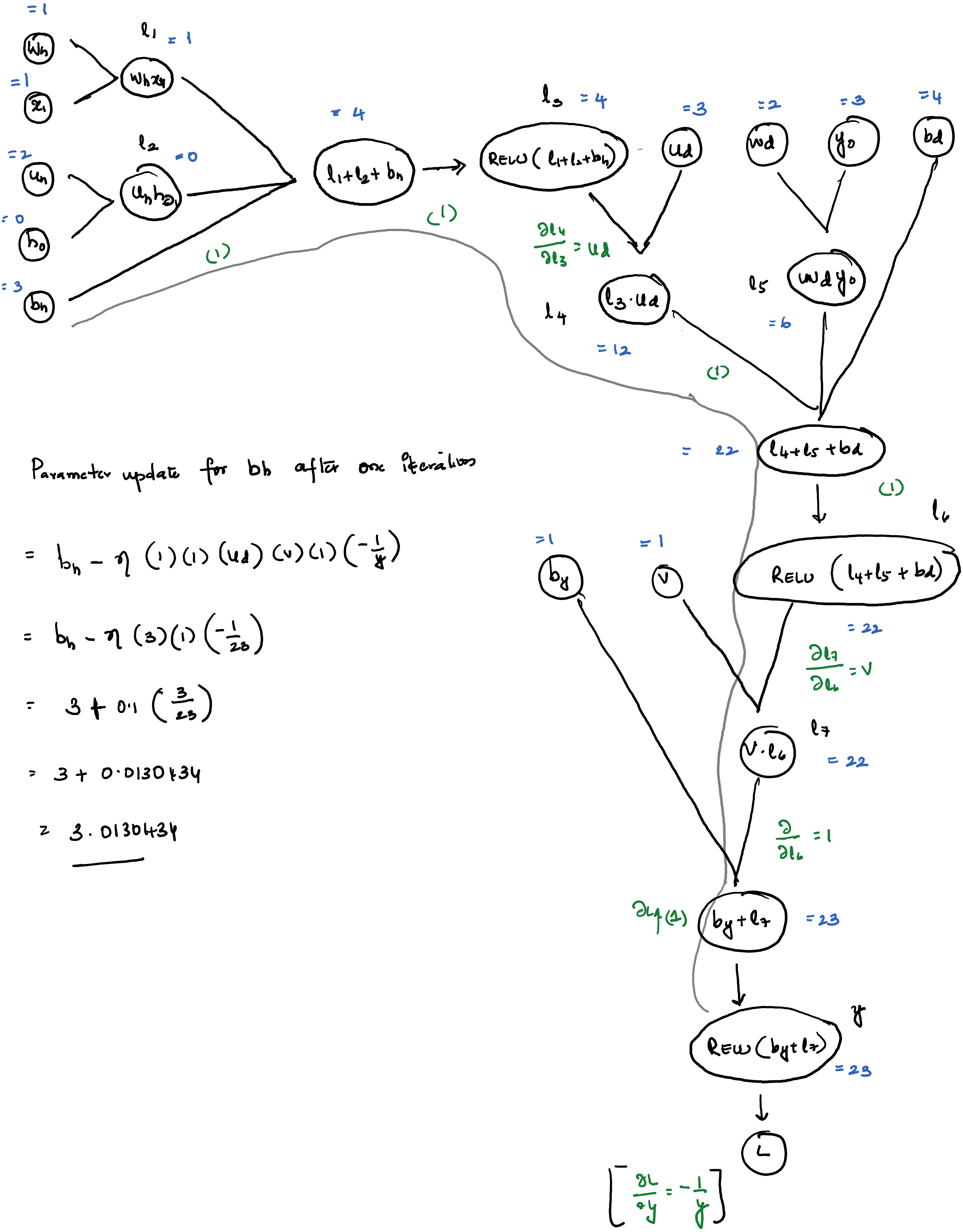
We want to train this network using the loss function $L(\hat{y}) = -\ln(\hat{y})$ and learning rate $\eta = 0.1$. (It doesn't really make sense to use RELU activation in an encoder-decoder, or negative log likelihood loss with RELU, but we are using them anyway to make the calculations easier.)

Suppose the current values of the parameters are

$$\begin{array}{lll} w_h = 1 & u_h = 2 & b_h = 3 \\ w_d = 2 & u_d = 3 & b_d = 4 \\ v = 1 & b_y = 1 & \end{array}$$

and we have $h_0 = 0$, $x_1 = 1$, $x_2 = 2$, and $y_0 = 3$.

What is the value of b_h after running these inputs through the network, computing the loss $L(\hat{y}_1)$, backpropagating the gradient of the loss, and performing parameter update? Simplify the expression as much as possible.



Parameter update for bh after one iteration

$$= b_h - \eta\,(1)(1)(U_d)(V)(1)\left(-\frac{1}{y}\right)$$

$$= b_h - \eta\,(3)(1)\left(-\frac{1}{23}\right)$$

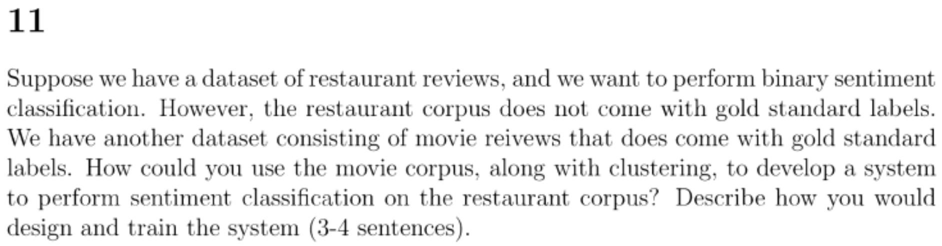
$$= 3 + 0.1\left(\frac{3}{23}\right)$$

$$= 3 + 0.0130434$$

$$= \underline{3.0130434}$$

Page 11

07 December 2020 09:32



11

Suppose we have a dataset of restaurant reviews, and we want to perform binary sentiment classification. However, the restaurant corpus does not come with gold standard labels. We have another dataset consisting of movie reivews that does come with gold standard labels. How could you use the movie corpus, along with clustering, to develop a system to perform sentiment classification on the restaurant corpus? Describe how you would design and train the system (3-4 sentences).

With clustering on movie corpus we can get features responsible for positive/negative labels.

We can use them as features for predicting initial set of labels for restaurant corpus and apply a bootstrapping kind of approach for generating the consequent labels.

We can also use a BERT based approach:

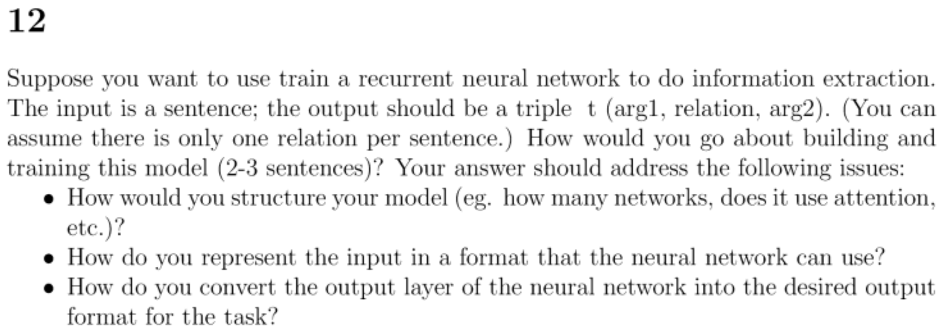
Use BERT to determine sentence embeddings for movie corpus :
With the generated embeddings we can apply clustering over it and generate cluster centers.

Now with the restaurant corpus we can generate BERT embeddings for sentence
Look for the cluster in the movie corpus.

Consider the new embeddings for the restaurant corpus apply clustering to recompute cluster centers

Page 12

07 December 2020 09:32



12

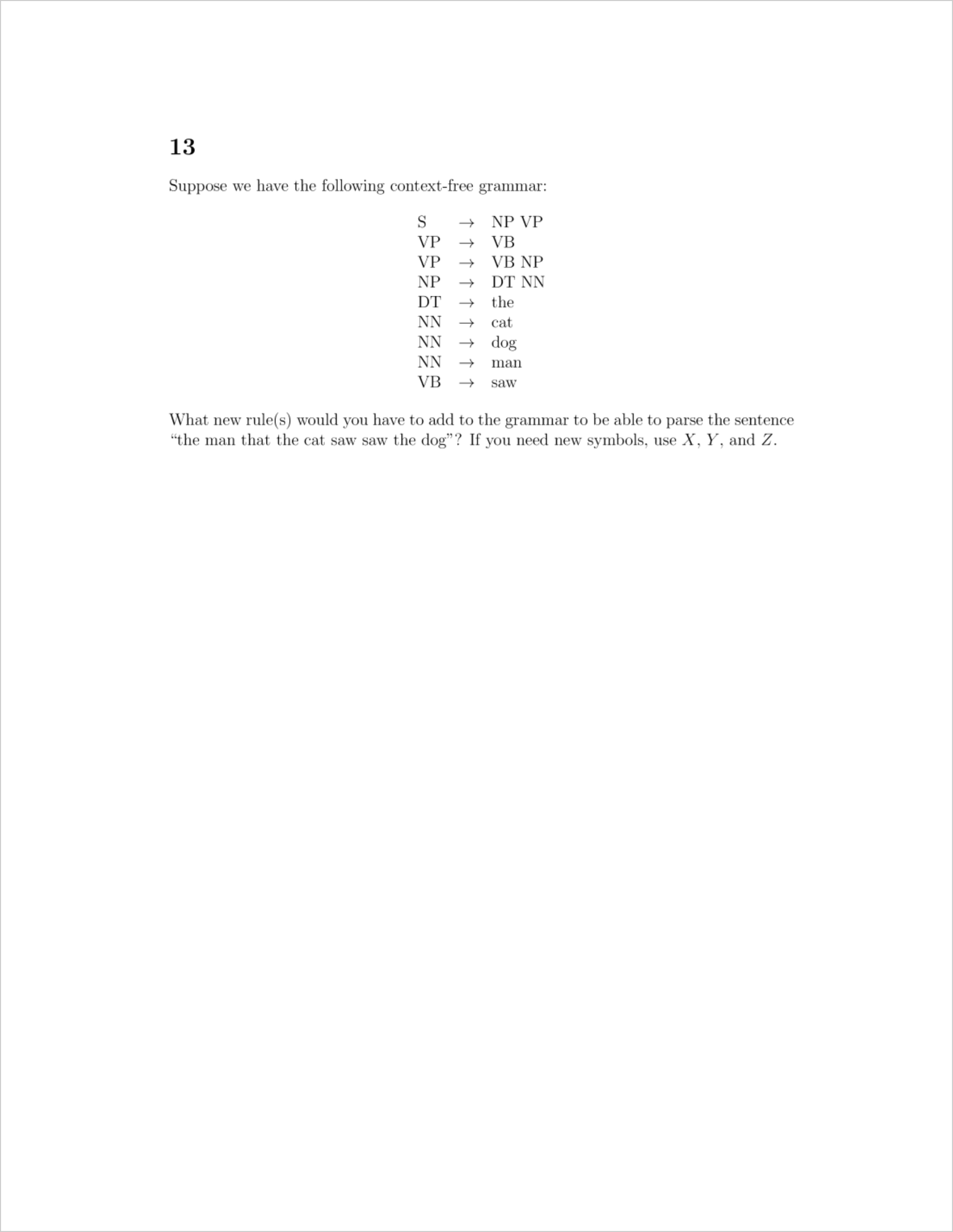
Suppose you want to use train a recurrent neural network to do information extraction. The input is a sentence; the output should be a triple t (arg1, relation, arg2). (You can assume there is only one relation per sentence.) How would you go about building and training this model (2-3 sentences)? Your answer should address the following issues:

- How would you structure your model (eg. how many networks, does it use attention, etc.)?
- How do you represent the input in a format that the neural network can use?
- How do you convert the output layer of the neural network into the desired output format for the task?

Recurrent NN for Information Extraction

Page 13

07 December 2020 09:32



13

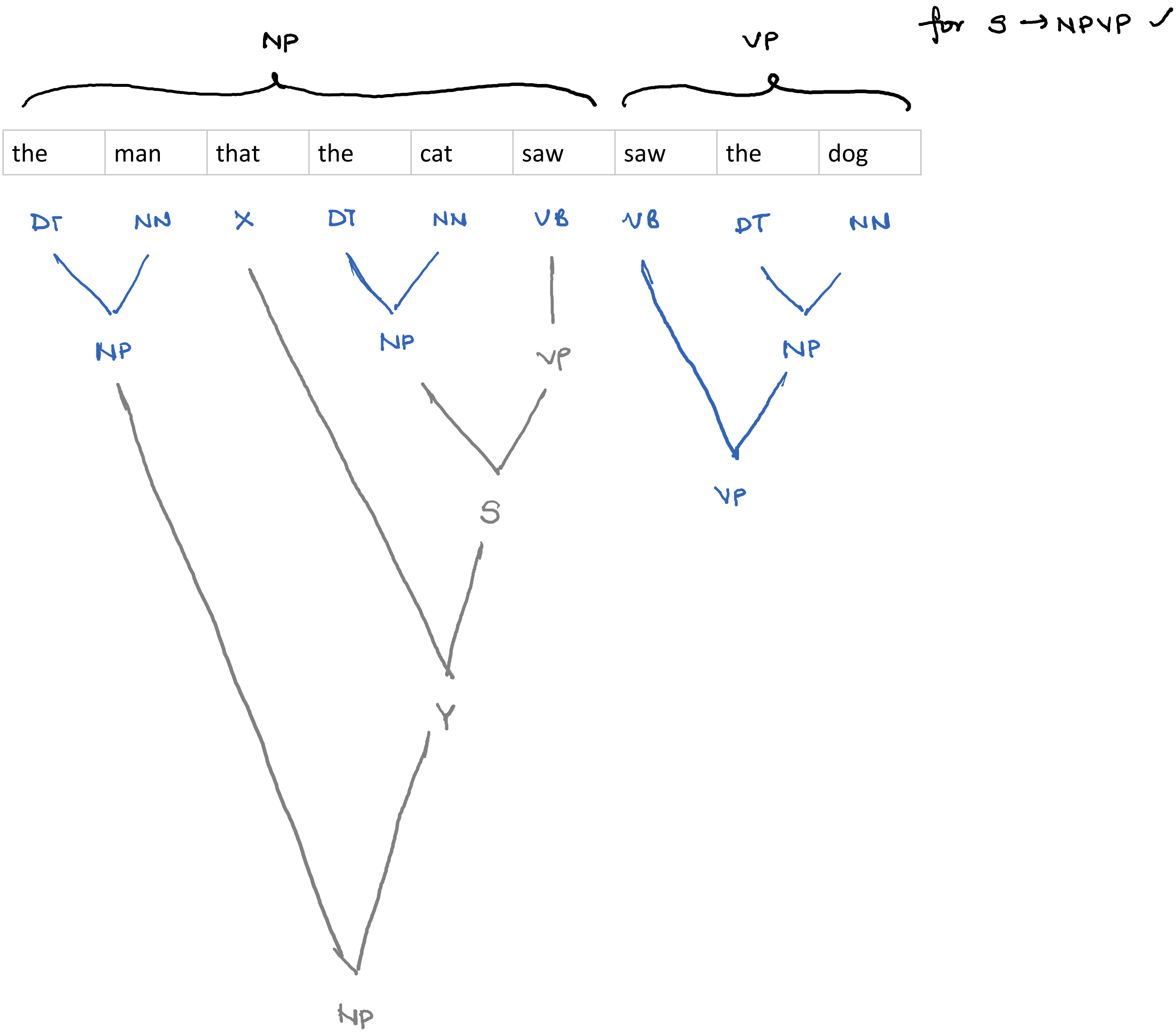
Suppose we have the following context-free grammar:

$$\begin{array}{lcl}
\text{S} & \rightarrow & \text{NP VP} \\
\text{VP} & \rightarrow & \text{VB} \\
\text{VP} & \rightarrow & \text{VB NP} \\
\text{NP} & \rightarrow & \text{DT NN} \\
\text{DT} & \rightarrow & \text{the} \\
\text{NN} & \rightarrow & \text{cat} \\
\text{NN} & \rightarrow & \text{dog} \\
\text{NN} & \rightarrow & \text{man} \\
\text{VB} & \rightarrow & \text{saw}
\end{array}$$

What new rule(s) would you have to add to the grammar to be able to parse the sentence "the man that the cat saw saw the dog"? If you need new symbols, use X, Y, and Z.

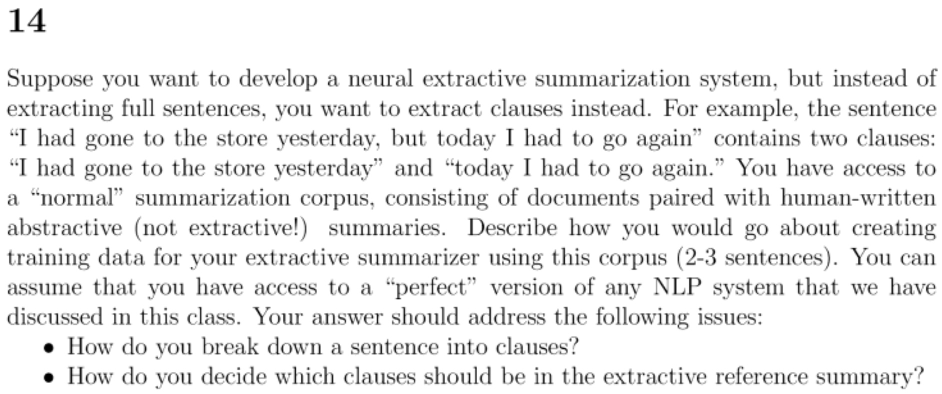
13.New CFG rules to be added

X --> that
Y --> X S
NP --> NP Y



Page 14

07 December 2020 09:32



14

Suppose you want to develop a neural extractive summarization system, but instead of extracting full sentences, you want to extract clauses instead. For example, the sentence "I had gone to the store yesterday, but today I had to go again" contains two clauses: "I had gone to the store yesterday" and "today I had to go again." You have access to a "normal" summarization corpus, consisting of documents paired with human-written abstractive (not extractive!) summaries. Describe how you would go about creating training data for your extractive summarizer using this corpus (2-3 sentences). You can assume that you have access to a "perfect" version of any NLP system that we have discussed in this class. Your answer should address the following issues:

- How do you break down a sentence into clauses?
- How do you decide which clauses should be in the extractive reference summary?

Naïve approach:

1. Take the ngrams for the the sentence
2. Generate embeddings for the above ngrams
3. Based on the similarity score of the above ngram embedding and the corresponding sentence embedding of the abstractive summarization we can get a signal for the quality of ngram.

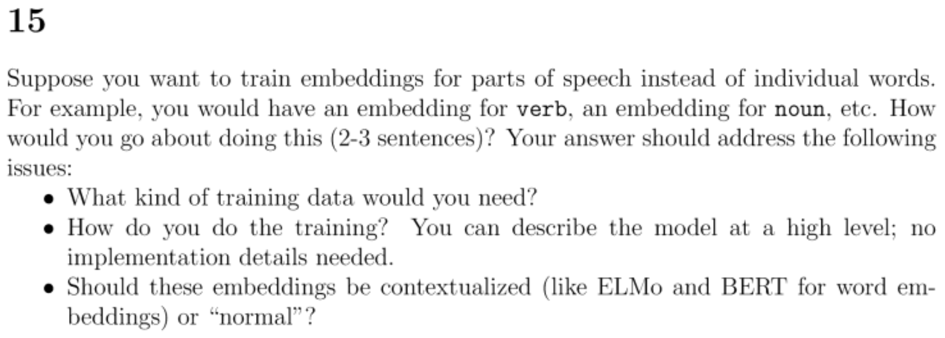
Now instead of ngram if we consider the clauses present in a sentence and follow the same procedure by taking considering the clauses that are of great quality [based on similarity metric as discussed above]

Generating clauses :

We can consider constituency parsing on the sentence.
From the generated parse. We can extract them by considering clausal tags of SBAR, S etc from Penn Treebank

Page 15

07 December 2020 09:32



15

Suppose you want to train embeddings for parts of speech instead of individual words. For example, you would have an embedding for `verb`, an embedding for `noun`, etc. How would you go about doing this (2-3 sentences)? Your answer should address the following issues:

- What kind of training data would you need?
- How do you do the training? You can describe the model at a high level; no implementation details needed.
- Should these embeddings be contextualized (like ELMo and BERT for word embeddings) or "normal"?

We can look up to use to word2vec approach for the generating tag embeddings.

As word2vec uses the following approach of learning a classifier on a binary prediction task of
"Is tag u likely to show up near tag w" [From text book]

We can also use the same approach in constructing a classifier to generate tag embeddings by considering the learned classifier weights as tag embeddings.

DATA: Tag annotated data from treebank is used.

Approach : [As described in text book for word embeddings, changing it to tag embeddings]

1. Treat the target tag and neighbouring context tags as positive examples
2. Randomly sample other tags from tagset to get negative examples
3. Use logistic regression to train a classifier to distinguish those two cases
4. Use regression weights as embeddings

Yes they might help.
We could concatenated contextualized tag embeddings and word embeddings as inputs to downstream tasks.