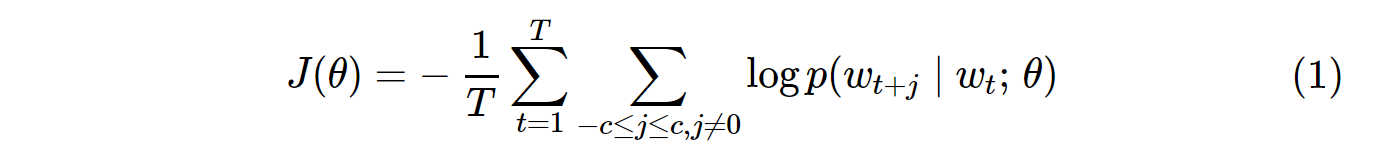
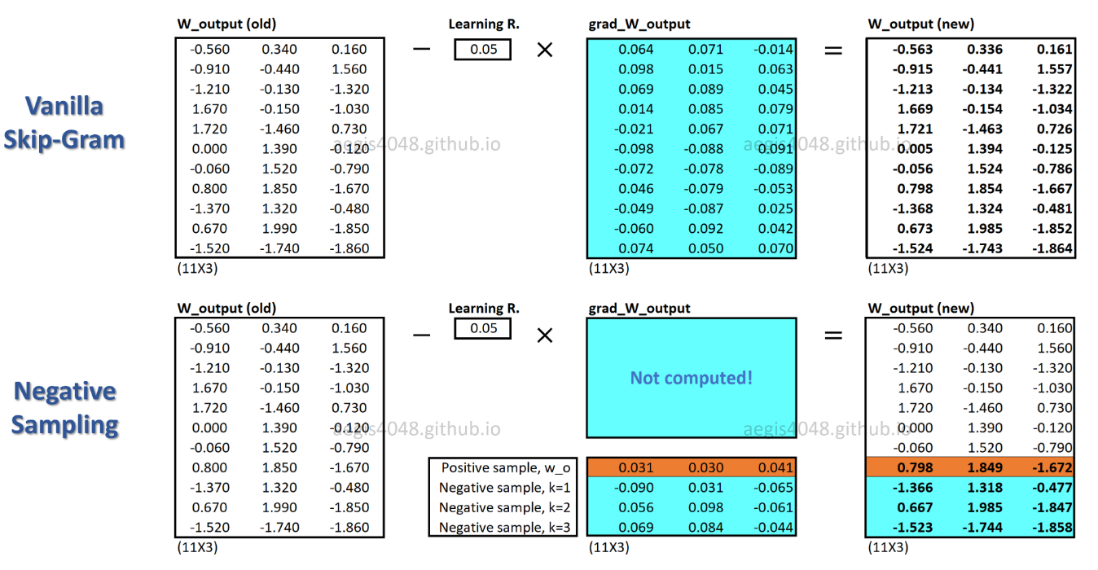
1a.

i) Negative sampling:

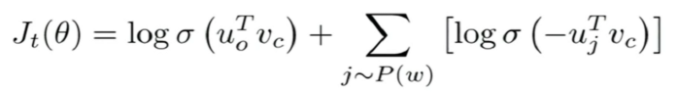
An alternative computation for similarity compared to the softmax function where we approximate softmax to reduce complexity. Rather than using a one-hot vector the size of the entire vocabulary, we randomly select a small number of negative words to update the weights for. In the output layer, only the weights for the positive word and our negative words will be updated, rather than all the weights.





ii) advantage : reduced computational efficiency as you don’t have to use all words in calculations

iii)



The first term tries to maximize the probability of occurrence for actual words that lie in the context window, i.e., they co-occur. The second term tries to iterate over some random words j that don’t lie in the window and minimize their probability of co-occurrence.

1b.

30000

1c.

i)

Word normalisation (lower casing, lemmatisation / stemming)

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |

My go at a corrected count table:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | mary | play | the | piano | john | get | ticket | for | we | see | game | system | </s> |
| <s> | 1 |  |  |  | 2 |  |  |  | 1 |  |  |  |  |
| mary |  | 2 |  |  |  |  |  |  |  |  |  |  |  |
| play |  |  | 1 |  |  |  |  |  |  |  | 1 |  | 1 |
| the |  | 1 |  | 1 |  |  |  |  |  |  |  | 1 |  |
| piano |  |  |  |  |  |  |  |  |  |  |  |  | 1 |
| john |  |  |  |  |  | 1 |  |  |  |  | 1 |  |  |
| get |  |  |  |  |  |  | 1 |  |  |  |  |  |  |
| ticket |  |  |  |  |  |  |  | 1 |  |  |  |  |  |
| for |  |  | 1 |  |  |  |  |  |  |  |  |  |  |
| we |  |  |  |  |  |  |  |  |  | 1 |  |  |  |
| see | 1 |  |  |  |  |  |  |  |  |  |  |  |  |
| game |  |  | 1 |  |  |  |  |  |  |  |  |  | 1 |
| system |  |  |  |  |  |  |  |  |  |  |  |  | 1 |

# Add the probability table

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | mary | play | the | piano | john | get | ticket | for | we | see | game | system | </s> |
| <s> | 0.25 |  |  |  | 0.5 |  |  |  | 0.25 |  |  |  |  |
| mary |  | 1 |  |  |  |  |  |  |  |  |  |  |  |
| play |  |  | 0.33 |  |  |  |  |  |  |  | 0.33 |  | 0.33 |
| the |  | 0.33 |  | 0.33 |  |  |  |  |  |  |  | 0.33 |  |
| piano |  |  |  |  |  |  |  |  |  |  |  |  | 1 |
| john |  |  |  |  |  | 0.5 |  |  |  |  | 0.5 |  |  |
| get |  |  |  |  |  |  | 1 |  |  |  |  |  |  |
| ticket |  |  |  |  |  |  |  | 1 |  |  |  |  |  |
| for |  |  | 1 |  |  |  |  |  |  |  |  |  |  |
| we |  |  |  |  |  |  |  |  |  | 1 |  |  |  |
| see | 1 |  |  |  |  |  |  |  |  |  |  |  |  |
| game |  |  | 0.5 |  |  |  |  |  |  |  |  |  | 0.5 |
| system |  |  |  |  |  |  |  |  |  |  |  |  | 1 |

ii)

denominator inside root = p(john | <s>) p(plays | john) p(games | plays) p(</s> | games)

= 0.5 \* 0 \* 0.33 \* 0.5 = 0

Then perplexity = 5th root of 1/0 which is undefined or NaN

denominator inside root = p(we | <s>) p(saw | we) p(mary | saw) p(playing | mary) p(games | playing) p(</s> | games)

0.25 \* 1 \* 1 \* 1 \* 0.33\*0.5 = 1/24 (correction, added 0.5)

Then perplexity = 6th root of 24 = 1.698

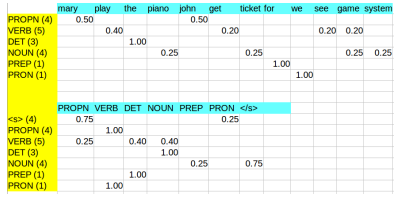
iii)

Add-one smoothing adds one for all pairs, taking away probability from existing pairs. This allows each pair to have some probability, mitigating infinite perplexity issues.

However, this reduces the probability for frequent pairs.

1d.

i)



\*this table is wrong Verb -> PROPN is 0.2\*

Corrected attempt:

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | mary | play | the | piano | john | get | ticket | for | we | see | game | system |
| PROPN | 0.5 |  |  |  | 0.5 |  |  |  |  |  |  |  |
| VERB |  | 0.4 |  |  |  | 0.2 |  |  |  | 0.2 | 0.2 |  |
| DET |  |  | 1 |  |  |  |  |  |  |  |  |  |
| NOUN |  | 0.2 |  | 0.2 |  |  | 0.2 |  |  |  | 0.2 | 0.2 |
| PREP |  |  |  |  |  |  |  | 1 |  |  |  |  |
| PRON |  |  |  |  |  |  |  |  | 1 |  |  |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | PROPN | VERB | DET | NOUN | PREP | PRON | </s> |
| <s> | 0.75 |  |  |  |  | 0.25 |  |
| PROPN |  | 1 |  |  |  |  |  |
| VERB | 0.2 |  | 0.4 | 0.4 |  |  |  |
| DET |  |  |  | 1 |  |  |  |
| NOUN |  |  |  |  | 0.2 |  | 0.8 |
| PREP |  |  | 1 |  |  |  |  |
| PRON |  | 1 |  |  |  |  |  |

ii)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | <s> | John | saw | Mary | gaming | the | piano | </s> |
| PROPN |  | .5\*.75 = 0.375 | 0 | .5\*.075\*.2 = 0.0075 | 0 |  |  |  |
| VERB | 0 | .2\*.375\*1= .075 | 0 | .2\*.0075\*1.0= 0.0015 |  |  |  |
| DET | 0 | 0 | 0 | 0 | 1.0\*.0015\*0.4= 0.0006 |  |  |
| NOUN | 0 | 0 | 0 | 0 |  | .2\*.0006\*1.0= 0.00012 | 0.00012 \* 0.8 = 0.000096 |
| PREP | 0 | 0 | 0 | 0 |  |  |  |
| PRON | 0 | 0 | 0 | 0 |  |  |  |

John saw Mary gaming the piano

PROPN VERB PROPN VERB DET NOUN

2a.

i.

z

Filter size = 3 x 50

Input layer = 6 x 50

Outputs of conv layer = 4 x 50

Outputs of pooling layer = 50

Output layer = 1

ii.

The network outputs a single value. If this value is passed through a sigmoid function it will range from 0-1 Where 1 strongly agrees within the evaluated sentiment and 0 strongly disagrees.

Or can use softmax 1\*2 output where higher value is taken

iii.

The 1st sentence needs 2 padding. The window size is 3 but there is only 1

\*

\*

\*\*

word.

(If punctuation counts then need 1 pad)iv.

I think they want to see how the intuition ties to the corpus here:

Word-segmentation/tokenization -> “well-done” becomes “well” and “done”,

stemming -> “your” becomes “you”

Potentially BPE to recognise sequences ‘you’ and ‘done’

2b. Unsure about these 2

i.

logistic regression - because it learns important features

I acknowledged the above but would personally have gone for an FFNN since it’s powerful enough to learn relationships between the counts of annotated terms and is more extensible if other features are added later.

ii.

Perform sub-word unit conversion to break down vocabularies into their sub words.

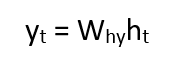
Perform logistic regression, this allows us to use new words we have not seen before.

Char level NN is suited because English and German have a similar alphabet? Might generalize well as words share characteristics e.g. Chemo and chemotherapy. Pre-processing could lowercase and replace unknown German letters with approximate counterparts e.g. ä => ae.

3a.

i.

Recurrent Neural Network is a generalization of feedforward neural network that has an internal memory. RNN is recurrent in nature as it performs the same function for every input of data while the output of the current input depends on the past one computation. After producing the output, it is copied and sent back into the recurrent network. At each timestep it considers the current input and the output that it has learned from the previous input.



ii.

Assume the RNN has an embedding bedding layer. The weights of that layer would be set to those of the pre-trained embeddings, and the layer would be set to not update (or can update as well).

This allows us to start with existing hidden presentations that hopefully allow the network to learn more quickly, (optionally can be trained to further customize for the particular task)

iii.

In transformer networks, self-Attention is used to compute similarity scores between words in a sentence.

In a regular RNN, this is not possible as the input is passed sequentially. Instead, a way would be to apply self-attention between the hidden state at each time step and some k previous timesteps' hidden states. I think that's what Ozan mentioned in the revision lecture too. (from Pavlos)

3b

i.

RNN architectures are limited when working with long sequences, because their ability to retain information from the first elements is lost when new elements were incorporated into the sequence.

The Attention mechanism offers an elegant solution to overcome the bottleneck. We weigh the information stored in each vector in H at every timestep and update our hidden state to include the information we believe is relevant. This

1. helps retain more information

2. Allows all information to be evenly evaluated

The role of attention is to expose relational bias in the context of an input and allow features relating to individual input words to be focused on and weighted accordingly.

(The question asks for NMT specifically so I think something related needs to be said)

Different languages can have different sentence structures, and attention can be used to compute which words influence the translation of the next word most. Compared to a standard RNN, which can only use the information sequentially.

ii.

r = q\*w1

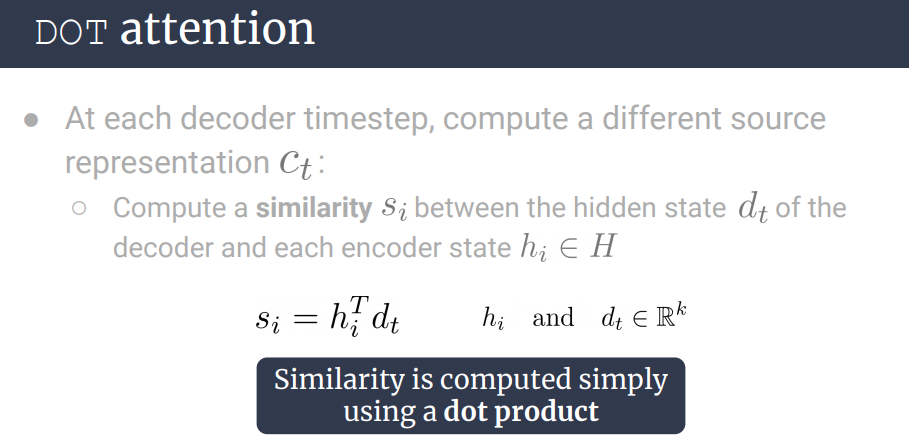
ri = hi\*w2

V = r \* ri for all I

Softmax V

Sum the weighted values of h

Dot attention:



3c.

i.

Cross entropy for classification or translation or prediction (normal bigram language model)

MSE for regression

What about the sequence loss? <- isnt this usually like a sum or product of individual losses?

ANSWER -> It’s the sum of the log probabilities of all the output words.

ii.

Perplexity (assuming a standard language model) describes the ability of an RNN language model to cope with unseen data. The better it performs, the lower the perplexity is and the more data it has seen.

BLEU Score?