**2019-2020**

**Examiners report can be found at** [C490\_Examination Feedback.pdf (ic.ac.uk)](https://exams.doc.ic.ac.uk/feedback/2019-2020/C490_Examination%20Feedback.pdf), has lots of detail.

1a.

Examiners report: 92% answered Q1, average mark was 14.1/20

Without negative sampling, computing a single forward pass of the model can be prohibitively expensive on large vocabularies, as the SoftMax activation function involves summing across the entire vocabulary.

Skip-Gram with Negative Sampling (SGNS) uses an approximation rather than a full probabilistic model - each training sample only modifies a sma ll percentage of the weights, rather than all of them.

We randomly select a small number of negative words to update the weights for. In the output layer, only the weights for our positive word and our negative words will be updated, rather than all the weights. (e.g. 1 positive word + 5 negative words for a 10k corpus with a dimensionality of 300 (N) = 1800 weights vs 3 million)

In the hidden layer, only the weights for the input word are updated (which is the same for skip-gram without negative sampling).

With negative sampling, the problem becomes a binary classification task.

Examiners report:

* Mention differences in architecture
* Mention intuition behind binary classification
* Mention why binary sampling is more efficient

1b.

i)

Punctuation removal

Normalization (remove capitals)

Wouldn’t say this is preprocessingBack off – Use trigram if you have a lot of evidence (large corpus) otherwise bigram, otherwise unigram.

Interpolation – Mix unigram, bigram, trigram

Lemmatization - ‘plays’ == ‘play’

<- Probably not for n-gram

Examiners report:

* Mention more than 1 technique (and make sure you mention more than 0!!)

ii) Your tables below are likely to vary based on the pre-processing steps you have chosen above.

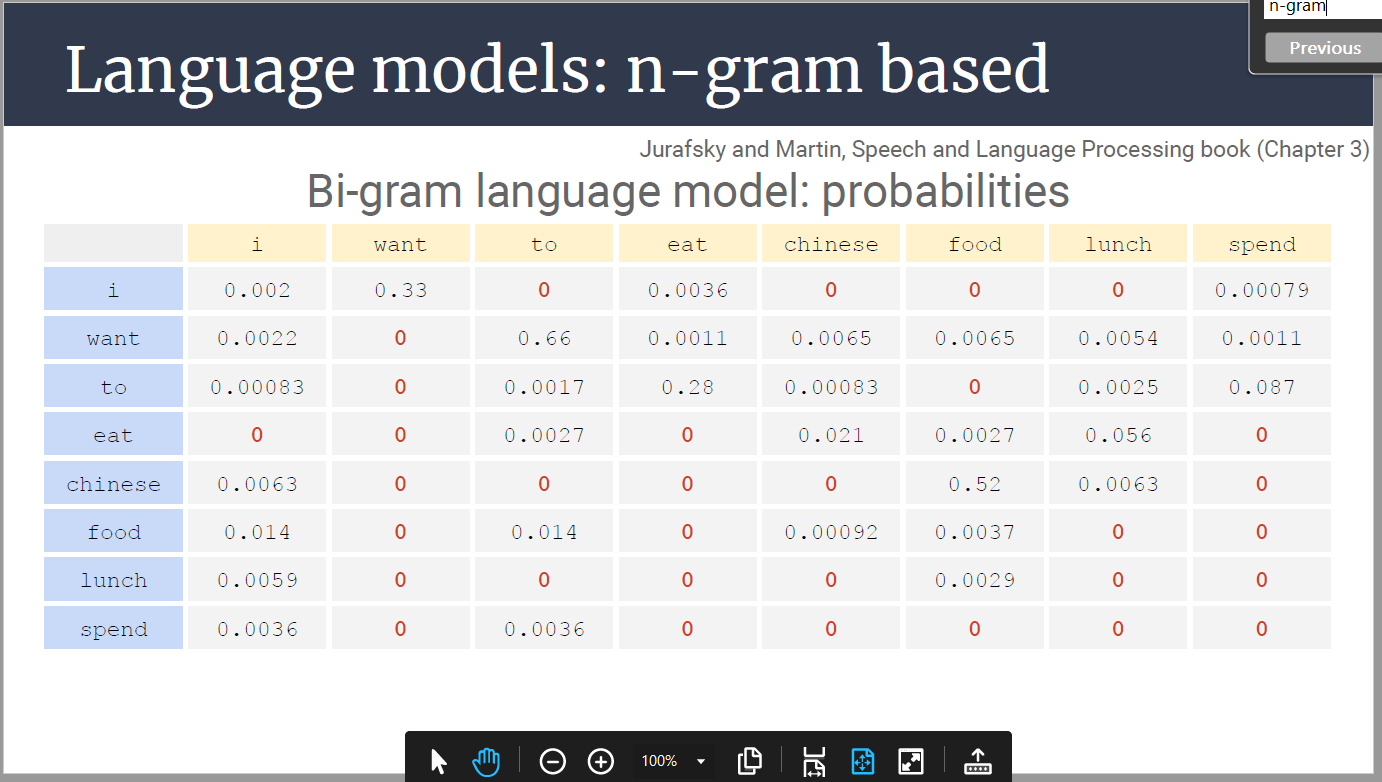
(alternative would be an add one model)

Bi-gram language model: **counts** (all empty cells = 0)

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Mary | plays | the | piano | John | got | ticket | for | play | game | </s> |
| Mary |  | 1 |  |  |  |  |  |  |  |  |  |
| plays |  |  | 1 |  |  |  |  |  |  |  |  |
| the |  |  |  | 1 |  |  |  |  | 1 | 1 |  |
| piano |  |  |  |  |  |  |  |  |  |  | 1 |
| John |  |  |  |  |  | 1 |  |  |  |  |  |
| got |  |  |  |  |  |  | 1 |  |  |  |  |
| ticket |  |  |  |  |  |  |  | 1 |  |  |  |
| for |  |  | 1 |  |  |  |  |  |  |  |  |
| play |  |  | 1 |  |  |  |  |  |  |  | 1 |
| game |  |  |  |  |  |  |  |  |  |  | 1 |
| <s> | 1 |  |  |  | 1 |  |  |  | 1 |  |  |

Bi-gram language model: **probabilities** (all empty cells = 0)

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Mary | plays | the | piano | John | got | ticket | for | play | game | </s> |
| Mary |  | 1 |  |  |  |  |  |  |  |  |  |
| plays |  |  | 1 |  |  |  |  |  |  |  |  |
| the |  |  |  | 0.33 |  |  |  |  | 0.33 | 0.33 |  |
| piano |  |  |  |  |  |  |  |  |  |  | 1 |
| John |  |  |  |  |  | 1 |  |  |  |  |  |
| got |  |  |  |  |  |  | 1 |  |  |  |  |
| ticket |  |  |  |  |  |  |  | 1 |  |  |  |
| for |  |  | 1 |  |  |  |  |  |  |  |  |
| play |  |  | 0.5 |  |  |  |  |  |  |  | 0.5 |
| game |  |  |  |  |  |  |  |  |  |  | 1 |
| <s> | 0.33 |  |  |  | 0.33 |  |  |  | 0.33 |  |  |



Count table but with lemmatisation

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Mary | play | the | piano | John | get | ticket | for | game | </s> |
| Mary |  | 1 |  |  |  |  |  |  |  |  |
| plays |  |  | 1 |  |  |  |  |  |  |  |
| the |  | 1 |  | 1 |  |  |  |  | 1 |  |
| piano |  |  |  |  |  |  |  |  |  | 1 |
| John |  |  |  |  |  | 1 |  |  |  |  |
| got |  |  |  |  |  |  | 1 |  |  |  |
| ticket |  |  |  |  |  |  |  | 1 |  |  |
| for |  |  | 1 |  |  |  |  |  |  |  |
| play |  |  | 1 |  |  |  |  |  |  | 1 |
| game |  |  |  |  |  |  |  |  |  | 1 |
| <s> | 1 | 1 |  |  | 1 |  |  |  |  |  |

Examiners report:

* Just don’t miscount

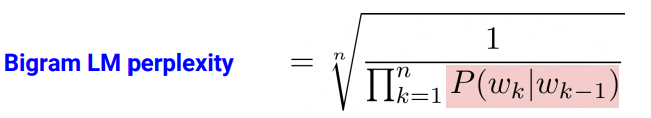
iii)

Sqrt(1/(πP(w k |w k-1)))

Sqrt(1/(0x0))= undefined/NaN (Undefined is probably more correct than infinite)

Never seen John -> plays or plays -> game

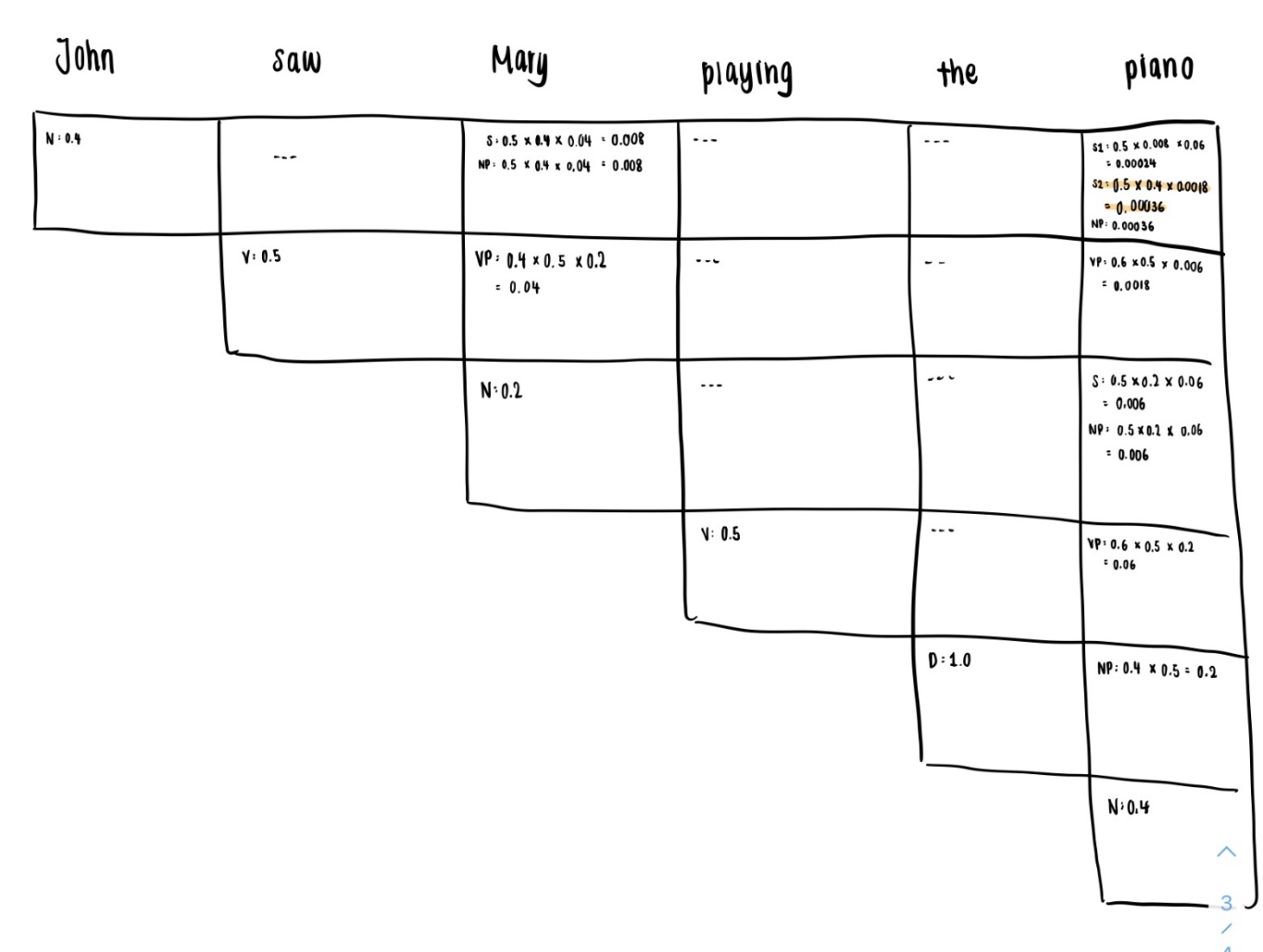
^ If this looks confusing:



Examiners report:

* Cannot be calculated/undefined is correct

1c.

1. The above image in table form (if its hard to read):

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| John | saw | Mary | playing | the | piano |
| N – 0.4 |  | S – 0.5 \* 0.4 \* 0.04 = 0.008  NP – 0.5 \* 0.4 \* 0.04 = 0.008 |  |  | S1 – 0.5 \* 0.008 \* 0.06 = 0.00024  S2 – 0.5 \* 0.4 \* 0.0018 = 0.00036  NP – 0.5 \* 0.4 \* 0.0018 = 0.00036 |
|  | V – 0.5 | VP – 0.4 \* 0.5 \* 0.2 = 0.04 |  |  | VP – 0.6 \* 0.5 \* 0.006 = 0.0018 |
|  |  | N – 0.2 |  |  | S – 0.5 \* 0.2 \* 0.06 = 0.006  NP – 0.5 \* 0.2 \* 0.05 = 0.006 |
|  |  |  | V – 0.5 |  | VP – 0.6 \* 0.5 \* 0.2 = 0.06 |
|  |  |  |  | D – 1 | NP – 0.5 \* 0.4 = 0.2 |
|  |  |  |  |  | N – 0.4 |

Examiners report:

* Show calculations working

ii)

Lexicon

N -> games

CONJ -> and

N -> Jane

Add another auxiliary “conjunction phase” rule:

* CP -> CONJ N (could potentially be NP instead of N, to allow for “X and X and X” but
* probably not needed as we just need to cover the given example)
* NP -> N CP

Examiners report:

* Remember to add grammar too

2a. Examiners report: 98% answered Q2, average mark was 12.5/20

i)

Using Tanh can cause certain issues ([Bounded output regression with neural networks - Dans World](https://dans.world/Bounded-output-networks/))

Instead we don’t modify the output at training time and instead clip the output to [-1, 1] during testing

Examiners report:

* Use linear output
* Need to mention removing sigmoid and decision rule (threshold)

ii)

Mean squared error? Can’t use Cross entropy since it is not a classification issue.

BCELoss/BCELogitsLoss (if we assume ii) and i) are not related and the outputs are still 0, 1)

Examiners report:

* This does assume we’re talking about regression now (but both answers were accepted)
* Explain intuition behind loss function

iii)

RMSE: Measures the average prediction error, with higher weighting for outliers

MAE: Measures the average prediction error but weighting outliers less

(Sklearn literally only gives MSE, MAE, RMSE, R2 and Absolute Variance as regression metrics)

Standard classification metrics like accuracy/precision/recall/etc (assuming ii and iii are not related to I and we’re still outputting 0/1)

Examiners report:

* Same as previous question
* Provide an explanation

2b

i)

Lemmatisation/Stemming

I think only stemming makes more sense

Happy, happier -> happ

Bored, boring -> bor

Stop word removal -> get rid of “us” and “it”

------

I think only lemmatisation makes more sense because I think we get more overlapping words

Stemming result: The movi make me feel happi I wa bore A nice US comedi Thi movi made us happier It wa veri bore

Lemmatization result: The movie make me feel happy, I be bore, A nice US comedy, This movie make us happy, It be very bore

Examiners report:

* Include basic tokenisation pre-processing!!!

ii)

True casing. The only word that would be affected by lowercasing is US. However, the 2 words have different meanings (us – You and I, US – United States of America)

Examiners report:

* N/A

iii)

Yes. The dimensions of the input feature matrix are nxd where d is the embedding size and n is the number of words. Since the 2nd window size is 5, and you lose 1 from each dimension in the first layer both ‘I was bored’ and ‘A nice US comedy’ need padding.

Examiners report:

* N/A

iv)

NB: Ignore the fact that the conv is 2D and assume 1D, since it wasn’t taught this year

1st you add 2 padding to the embedding to give an input size of 6\*50.

The filter size of the 1st layer is 2\*50, The output size is 10\*5.

The filter size of the 2nd layer is

5\*50, The output size is 10\*2.

After relu: Same still

After maxpooling: 10\*1 and 10\*1

After concatenation: 10\*2

After output: 1 (sigmoid sentiment for binary)

OR

Filter sizes are 2x50 and 5x1

first layer we have (1,1,6,50) => (1, 10, 5, 1)

Second layer: (1, 10, 5, 1) => (1, 10, 1, 1)

Max pooling: (1, 10, 1, 1) => (1, 10, 1, 1)

Concatenation: (1, 10, 1, 1) => (1, 10)

Output layer: (1, 10,) => (1, 1)

OR

Model Input: 5 x 50 matrix

SizesSizes of Filters: 10 matrixes with dimension (4, 50) and 10 matrixes with dimension (22, 50)

Output of Convolution Layer: 10 Feature Maps with dimension (4, 1) and 10 feature map maps with dimension (1, 1)

Output of Pooling Layer: vector with dimension (20, 1)

Final Output: single value

See discussion: <https://piazza.com/class/kf7uhclsgjbyt?cid=182>

Examiners report:

* Both sequential/parallel calculations were accepted
* Provide explanations!!

v)

It sets the value of any nodes that are less than 0 to 0.

ReLU is useful since it stops us from activating all the neurons at the same time. If neurons are not important in the computation (give a value less than 0) we essentially turn them off.  
This is useful during backpropagation since we can train only the neurons that are “turned on”.

This makes the network far more computationally efficient.

The advantage of using this activation is that it is much less vulnerable to the vanishing gradient problem than sigmoid or tanh are.

Examiners report:

* N/A

3a. Examiners report: 42% of people answered Q3, average mark was 16.5/20

i)

Regular LM - A language model stores the precomputed counts/probabilities from n-grams of length from 1 to n. Values are then computed column-wise into probabilities. This tells us the number of times we see wk-1  and from the same column we can take the row that matches the word wk.

Feed-Forward LM

A FF LM uses a predefined number of word embeddings & passes them in as the network input. The 1st word embedding is the current word, and the others are the previous n-1 words.

Recurrent LM

Passes word embeddings into the network one at a time, The network passes wk-1 first and then wk.

The Markov assumption states the future states depend only upon the present state, not the sequence of events that preceded it.

The FF LM does abide by this assumption since it processes only the current n-gram for predicting the next word, assuming that the current word only depends on those.

A recurrent LM does not since all words before it are processed by the network and taken into account.

Examiners report:

* Explicitly talk about both, not an overview

ii)

Back propagation through time is the application of the backpropagation algorithm to a recurrent neural network applied to sequence data like a time series.

Conceptually, BPTT works by unrolling all input timesteps. Each timestep has one input timestep, one copy of the network, and one output. Errors are then calculated and accumulated for each timestep. The network is rolled back up and the weights are updated.

Classical backprop traverses each layer only once.

The main difference is BPTT is designed for time series inputs and Backpropagation is designed for singular inputs.

Examiners report:

* N/A

3b.

i)

Probably right (from Gio) : LNMT = -ΣtT ytlog(P()t)

Clearer:



We sum the negative log probabilities of our one hot cross entropy losses. P()\_t returns a one hot probability (I think?) and y is a one hot target vector.

Examiner's report:

* This should be cross-entropy loss

ii)

H would be a vector of size (S, 400)

Each word is turned into a hidden encoding. We don’t use 500 because that’s the in between layer

iii)

2 separate LSTMs, 1 encodes L to R, 1 encodes R to L. We then concatenate their representations. H is still a set of vectors (matrix) of length s but its dimensions is now 800.

It wouldn’t be (S, 2, 400) because dimension 1 is usually batch size (but here we are only looking at 1 sentence)

3c.

i)

Attention mechanism offers an elegant solution to overcome the data retention bottleneck that sequential networks have. We weigh 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

ii)

* Use all states H instead of a constant vector V
* At each decoder timestep compute a source representation Ct
* Compute a similarity Si between the current hidden state dt of the decoder & each encoder state hi ∈ H (perform at each decoder step) **using the dot product si= hidt**
* Normalise similarity scores {a1,..,a s} = softmax({s1,..,s s})
* Weigh the encoder states c t=Σαh
* Add c t to the hidden values before going through the top layers that evaluate which word to choose.

4a. Examiners report: 68% of people answered Q4, with an average mark of 13.3/20

i)

Word2Vec

Predicts context words from the target word. This only takes 1 word as the context.

N-Gram Language Model

Predicts a word based on the last n-1 words. This is good since you can modify the number of context words you want, however for a large n you need a large corpus size. Unlike Word2Vec, future words are not considered when learning the representation of a target word.

Recurrent NN Language Model

Can take a sequence of previous words (of any length up to around 50) as context

Transformer based embeddings: ATTENTION IS ALL YOU NEED.

An inductive bias (attention) allows a learning algorithm to prioritize one solution (or interpretation) over another independent of the observed data. This lets us use a large context and prioritize the information we deem as important. Can look at the entire/any part of sequence for context, rather than just a few words back and forward.

Examiners report:

* Need to talk about context instead of the model generally

ii)

The Transformer based embeddings. This is because we can use a large context and prioritize the information we deem as important at every timestep. All the others either have a small context or must use the entire thing all the time.

Examiners report:

* Must include WHY
* Encode entire input/use of attention

4b.

Convolution Layers

Learn word combinations that are important (n-gram)

Pooling Layers

We want these n-grams to have the same representation anywhere in the sentence. Convert different feature maps into fixed size.

Dropout mechanism

Prevents the neural network from overfitting. The network tries to balance its calculations across all its nodes. It prevents the network from over-relying on the presence of specific words or features, thus improving performance on unseen sentences.

Examiners report:

* Pooling layer was most often forgotten
* Explanations should have some detail

4c.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | <s> | John | saw | Mary | gaming | the | piano | </s> |
| PROPN |  | .5\*.75=.375 | 0 | .075\*.25\*.5=.009375 | 0 |  |  |  |
| VERB | 0 | .375\*.2\*1 =.075 | 0 | .009375\*0.2\*1.001875 |  |  |
| DET | 0 | 0 | 0 | 0 | .009375\*0.4\*1.00075 |  |
| NOUN | 0 | 0 | 0 | 0 |  | .00075\*.25\*1.0001875 |
| PREP | 0 | 0 | 0 | 0 |  |  |
| PRON | 0 | 0 | 0 | 0 |  |  |

John saw Mary gaming the piano

PROPN VERB PROPN VERB DET NOUN

Examiners report:

* Get your maths right
* Its HMM with viterbi, not CYK
* Everyone got the right tags anyways “it seems like students knew the final answer already”

4d.

* **Non sequential**: sentences are processed as a whole rather than word by word.
* **Self Attention**: this is the newly introduced 'unit' used to compute similarity scores between words in a sentence.
* **Positional embeddings**: another innovation introduced to replace recurrence. The idea is to use fixed or learned weights which encode information related to a specific position of a token in a sentence.

Examiners report:

* Need to elaborate more
* Point out the confusion between language/translation/BERT models