Homework 1

Problem 1

A) Write regular expressions.

1. Language 1:

/(\b[A-Za-z]+\b)\s\1/

(From the given examples, I use \s here to represent a break between two words. If we should consider other punctuations, we should change \s into other expression.)

2. Language 2:

/(?=.\*hedge)(?=.\*fund)/

(From the given example of ‘funds’, I do not specify \b for hedge and fund.)

B) Produce a QA system.

Below is the running result for reading queries from a file.

Text, letter

Description automatically generated

It can also respond to console input.

Problem 2

S1: Sales of the company to return to normalcy.

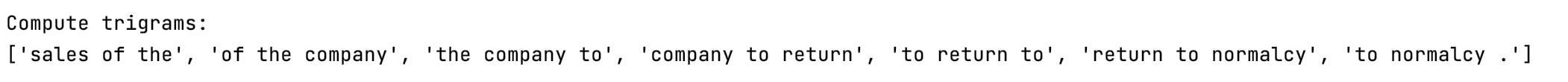
S2: The new products and services contributed to increase revenue.

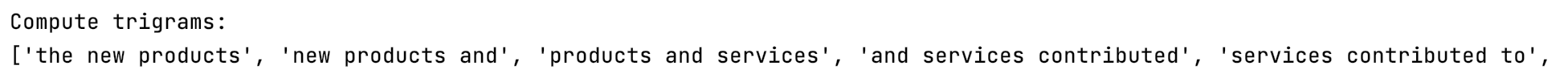
A) Compute trigram probabilities.

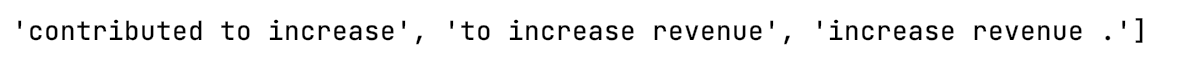
1. Compute the tragrams:

S1: ['sales of the', 'of the company', 'the company to', 'company to return', 'to return to', 'return to normalcy', 'to normalcy .']

S2: ['the new products', 'new products and', 'products and services', 'and services contributed', 'services contributed to', 'contributed to increase', 'to increase revenue', 'increase revenue .']







2. Compute trigram counts and probabilities without smoothing.

See related tables in /tables.

3. Compute trigram counts, probabilities and re-constituted counts with Laplace (add-one) smoothing.

See related tables in /tables.

4. Compute trigram probabilities with Katz back-off smoothing.

See related tables in /tables.

For S1:

Running results:

Text

Description automatically generated with low confidence

For S2:

Running results:

Text

Description automatically generated with low confidence

From my observation, all zero count in trigram level can all find non-zero count in bigram level. Hence, for the sentence we have, zero unigram probabilities are computed.

5. Compute three kind of total trigram probabilities.

For S1:

Running results:

Text

Description automatically generated

For S2:

Running results:

Text

Description automatically generated with medium confidence

B) Learn a neural language model on Google cloud.

Actually, a validation set with size of 10000 has been given in the source code.

Graphical user interface, text, application

Description automatically generated

Hence, I use loss, perplexity and accuracy graphs to evaluate different models.

I use the same first two layers for all models:

model = keras.Sequential()

model.add(keras.layers.Embedding(vocab\_size, 50, weights=[embedding\_matrix]))

model.add(keras.layers.Reshape([50\*N]))

After this, each model have different layer settings:

1 Intermediate Layer

model.add(keras.layers.Dense(vocab\_size, activation=tf.nn.softmax))

2 Intermediate Layers

#model.add(keras.layers.Dense(16, activation=tf.nn.relu))

model.add(keras.layers.Dense(vocab\_size, activation=tf.nn.softmax))

3 Intermediate Layers (a)

model.add(keras.layers.Dense(16, activation=tf.nn.relu))

model.add(keras.layers.Dense(1024, activation=tf.nn.relu))

model.add(keras.layers.Dense(vocab\_size, activation=tf.nn.softmax))

3 Intermediate Layers (b)

model.add(keras.layers.Dense(16, activation=tf.nn.relu))

model.add(keras.layers.Dense(128, activation=tf.nn.relu))

model.add(keras.layers.Dense(vocab\_size, activation=tf.nn.softmax))

|  |  |
| --- | --- |
| 1 intermediate layer | 2 intermediate layers |
| A picture containing chart  Description automatically generated | A picture containing graphical user interface  Description automatically generated |
| Text  Description automatically generated with low confidence | Text  Description automatically generated |
| A picture containing diagram  Description automatically generated | A picture containing graphical user interface  Description automatically generated |
| 3 intermediate layers (a) | 3 intermediate layers (b) |
| A picture containing chart  Description automatically generated | A picture containing chart  Description automatically generated |
| A picture containing graphical user interface  Description automatically generated | A picture containing graphical user interface  Description automatically generated |
| A picture containing diagram  Description automatically generated | A picture containing diagram  Description automatically generated |

To be honest, the given 2 intermediate layers model has pretty good performance in loss and perplexity. I tried many parameters and some of them can have higher accuracy like 3 intermediate layers model (a).

Problem 3

1. Compute the PPMI.

Table

Description automatically generated

2. Compare the similarity.

Table

Description automatically generated

For the [chairman, company] has the biggest cosine value, they are the most similar among these pairs.

Problem 4

For sentences:

S1: The chairman of the board is completely bold.

S2: A chair was found in the middle of the road.

1. Present only the transition and observation likelihoods in the states reached after three steps.

Shape

Description automatically generated with medium confidenceShape

Description automatically generated

2. Create the Viterbi table and populate it entirely.

Shape

Description automatically generated

Shape

Description automatically generated

3. Calculate the probability of assigning the tag sequence for each of the sentences.

For S1:

The transition probabilities from RB to NN and from JJ to </s> are all 0.

P(S1) = 0

This is strange. Because a sentence end with “bold” is quite normal. If the transition probabilities from JJ to </s> is not 0, we could tag S1 as:

S1: The/DT chairman/NN of/IN the/DT board/NN is/VBZ completely/RB bold/JJ.

For S2:

We can tag S2 as:

S2: A/DT chair/NN was/VBN found/VBN in/IN the/DT middle/NN of/IN the/DT road/NN.

P(S2) = 1 \* 0.38 \* 1 \* 0.58 \* 0.69 \* 0.32 \* 1 \* 0.2 \* 0.99 \* 0.11 \* 1 \* 0.57 \* 1 \* 0.58 \* 0.66 \* 0.25 \* 1 \* 0.57 \* 1 \* 0.58 \* 1 \* 0.11 = 2.1E-6

4. Execute the Stanford POS-tagger.

Results:

S1: The\_DT chairman\_NN of\_IN the\_DT board\_NN is\_VBZ completely\_RB bold\_JJ .\_.

S2: A\_DT chair\_NN was\_VBD found\_VBN in\_IN the\_DT middle\_NN of\_IN the\_DT road\_NN .\_.

The result of S2 is more accurately, because it has a new tag named VBD.

Actually, the result of S1 is just the same as we get in part 3 if the transition probabilities from JJ to </s> is not 0. Like I said in part 3, it is strange and I think the data that Stanford POS-tagger use has some difference from ours.