HomeWork 1

NLP – CS6320

Saivikas Meda
Sxm190011

PROBLEM 1: Regular Expressions (5 points)

1. Language 1: the set of all strings with two consecutive repeated words (e.g., “Humbert Humbert” and “the the” but not “the bug” or “the big bug”);
   a. “(\w+)\s+(\1)\b/”
   b. **Assumption**: for the consecutive words I made an assumption that words can be splited using space and then compare.

2. Language 2: all strings that start at the beginning of the line with an integer and that end at the end of the line with a word; (2 points)
   a. “^[0-9].*\b[a-zA-Z]+\b$”
   b. **Assumption**: The word starting with a number and ending with alphabet not with any punctuations. The match to the string will be as a whole or none if it satisfies the conditions.

3. Language 3: all strings that have both the word “grotto” and the word “raven” in them (but not, e.g., words like grottos that merely contain the word grotto);
   a. “.*(?=.*\b[Gg]rotto\b)(?=.*\b[Rr]aven\b).*”
   b. **Assumption**:Here I am assuming the input as a string which has ravenand grotto together. The input will be a full match or none if either of word is not present.

PROBLEM 2: N-Grams (40 points)

1. All the bigrams counts are stored in Bigram_Counts.json, BigramMatrix.xlsx and BigramMatrixLaplace.xlsx file which will be auto generated.
2. **Calculation of bigrams without padding**

```
(<s>,Sales)=0.001
(Sales,of)=1.0
(of,the)=0.2835820895522388
(the,company)=0.05874125874125874
(company,to)=0.013605442176870748
(to,return)=0.0030816640986132513
(return,to)=0.36363636363636365
(to,normalcy)=0.0015408320493066256
(normalcy,.)=1.0
(.,</s>)=0.9259259259259259

Bigram probablity for:  <s> Sales of the company to return to normalcy . </s>  =>  3.62
3421529813562e-13



(<s>,The)=0.127
(The,new)=0.02631578947368421
(new,products)=0.015151515151515152
(products,and)=0.23076923076923078
(and,services)=0.0029542097488921715
(services,contributed)=0.07692307692307693
(contributed,to)=0.6666666666666666
(to,increase)=0.0030816640986132513
(increase,revenue)=0.14285714285714285
(revenue,.)=0.14285714285714285
(.,</s>)=0.9259259259259259

Bigram probablity for:  <s> The new products and services contributed to increase rev
enue . </s>  =>  1.0309230980059047e-13
```

We observe S1 is more probable than S2 without smoothing.

3. **For Laplacian smoothing**

```
(<s>,Sales)=0.0003026176426085641
(Sales,of)=0.00035650623885918
(of,the)=0.04078203757446158
(the,company)=0.012075578917459867
(company,to)=0.0005211952744961779
(to,return)=0.00047938638542665386
(return,to)=0.0008896797153024911
(to,normalcy)=0.00031959092361776926
(normalcy,.)=0.00035650623885918
(.,</s>)=0.14013317191283292

Bigram probablity after Laplacian normalcy for:  <s> Sales of the company to return to n
ormalcy . </s>  =>  1.885648052552011e-28


(<s>,The)=0.019367529126948103
(The,new)=0.0008679048776254123
(new,products)=0.0003524229074889868
(products,and)=0.000711490572749911
(and,services)=0.0004772510340439071
(services,contributed)=0.0003557452863749555
(contributed,to)=0.0005345687811831789
(to,increase)=0.00047938638542665386
(increase,revenue)=0.00035612535612535614
(revenue,.)=0.00035612535612535614
(.,</s>)=0.14013317191283292

Bigram probablity after Laplacian normalcy for:  <s> The new products and services cont
ributed to increase revenue . </s>  =>  3.2591300673489e-33
```

We can clearly observe S1 is more Probable than S2 after performing Laplacian Smoothing.

PROBLEM 3: Vector Semantics (25 points)

For the left right 5 words checking I am only considering the and removing the punctuations as we need nearest words relation.

For words [‘chairman’,’company’]

In context of [‘said’,’of’,’board’]

1. **With nill padding**

   Counts

   ```
    [[110. 232. 13.]

   [21. 29. 26.]]
   ```

```
chairman  with the context of  said :  0.027805556684263225
chairman  with the context of  of :  0.10994384325474159
chairman  with the context of  board :  0.0
company  with the context of  said :  0.0
company  with the context of  of :  0.0
company  with the context of  board :  1.9186540449243565
```

2. **With 2 padding**

   Counts

   ```
   [[ 112. 234. 15]

   [23. 31. 28.]]
   ```

with two padding

```
chairman  with the context of  said :  0.025847186674620458
chairman  with the context of  of :  0.11582403180069015
chairman  with the context of  board :  0.0
company  with the context of  said :  0.0
company  with the context of  of :  0.0
company  with the context of  board :  1.8147010512924409
```

3. **For words [‘chairman’, ‘company’ , ‘ sales’, ‘economy’]**

   In context of [‘said’, ’of’, ‘board’]

   Using 2 paddings

   [[112. 234. 15.]

   [23. 31. 28.]

   [5. 6. 2.]

   [3. 6. 2.]]

Similarity values:

```
chairman company  Similarity:  0.8271729785544567
chairman sales  Similarity:  0.9517791334782534
chairman economy  Similarity:  0.9730674156462568
company sales  Similarity:  0.9285735474162691
company economy  Similarity:  0.931680742420691
sales economy  Similarity:  0.9745586289152095
```

Chairman company similarity: `0.8271729785544567`
Company sales similarity: `0.9285735474162691`
Company economy similarity: `0.931680742420691`

We see that **economy** and **company** are **more similar**, the reason for these two words to be similar can be that the no of occurrence can be similar with the words to which we calculated the context of matrix. Here as the vectors calculated with the count of the occurrence with respect of the context words we found. Both of them as most similar to each other.

4. **Using Glove.**

```
chairman company  Similarity:  0.5737977615857037
chairman sales  Similarity:  0.3132328672556363
chairman economy  Similarity:  0.3397174688938959
company sales  Similarity:  0.7634717732060282
company economy  Similarity:  0.47354638478858574
sales economy  Similarity:  0.6253732528163337
```

Chairman company similarity: `0.5737977615857037`
Company sales similarity: **`0.7634717732060282`**
Company economy similarity: `0.47354638478858574`

Using Glove we observe that **Company is more similar to Sales**. The similarity varies from w hat we observed on the provided corpus as the given corpus is of low volume to that Glove has used to train the data. The Glove vectors contains 50dimensions for a single word so obs erved similarity varies a lot.

PROBLEM 4: Part-of-speech tagging (30 points)

1. Hidden Markov Model
    a. In folder NLP_HW1_P4 contains the design of markov model for S1 and S2.

2. Viterbi table :
    a. In Folder Viterbi_P4.xlsx contains the table values for S1 and S2 statements.

3. The final probability of assigning tags
    a. The probability of assigning the tag sequence Sentence S1 " The chairman of the board is completely bold" is **2.9E-05**
    b. The probabiklity of assigning the tag sequence Sentence S2 " A chair was found i n the middle of the road" is **1.911E-05**

4. Using NLTK pos_tag the tag sequence is:
    a. Pos for: The chairman of the board is completely bold. => [('The', 'DT'), ('chairman', 'NN'), ('of', 'IN'), ('the','DT'), ('board','NN'), ('is','VBZ'), ('completely','RB'), ('bold','JJ'), ('.','.')].
    b. Pos for: A chair was found in the middle of the road. => [('A','DT'), ('chair','NN'), ('was','VBD'), ('found', 'VBN'), ('in','IN'), ('the','DT'), ('middle','NN'), ('of','IN'), ('the','DT'), ('road','NN'), ('.','.')].

5. The code to find the tag is attached in folder with **HW1_P4_StandfordPOS.py** file name.
    a. To execute first run the following lines in python terminal
    b. Just use "**python3 HW1_P4_StandfordPOS.py**"
    c. On comparison with penn treebank POS computed Viterbi and NLTK pos_tag S1 shows same result with in S2 with find one difference was/VBN in Viterbi while was/VBD. The reason for the conflicts can be the provided tag list is lower the information for the tag is lost in it, the second reason for the conflict can be the training data for the both pos taggers. NLTK pos tagger is more accurate as it uses a greater number of tags for comparisons.

Extra Credit :

Problem2

Model with one hidden layer;

LINK: https://colab.research.google.com/drive/15aCeCzVh5dSeJL7iyClJBnyitV4J_r-K

Build the Model:

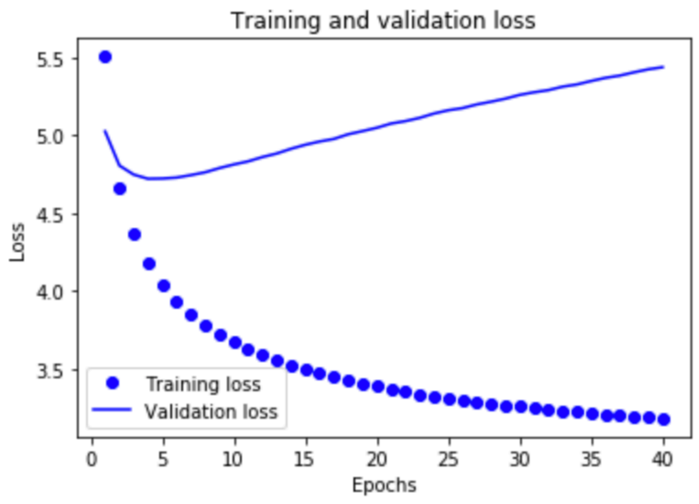
Layer(Type)	**Output Shape**	**Param #**
Embedding (Embedding)	(None, None, 50)	500000
Reshape(Reshape)	(None, 250)	0
Dense(Dense)	(None, 10000)	2510000

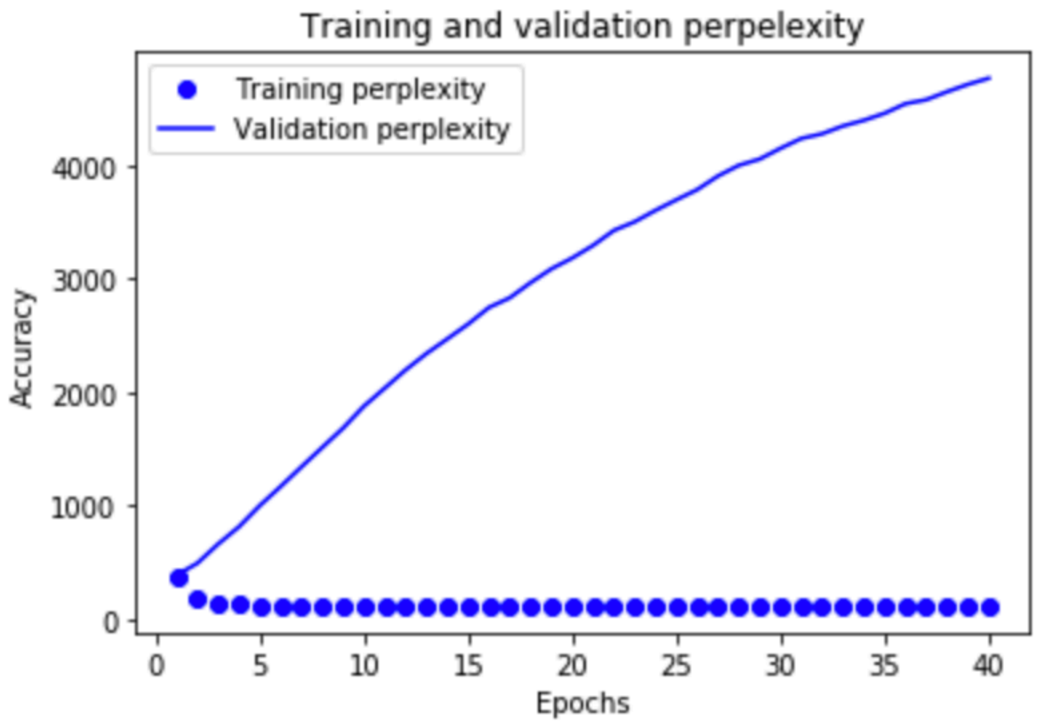
Total params: 3,010,000 ; Trainable params: 3,010,000; Non- trainable params: 0

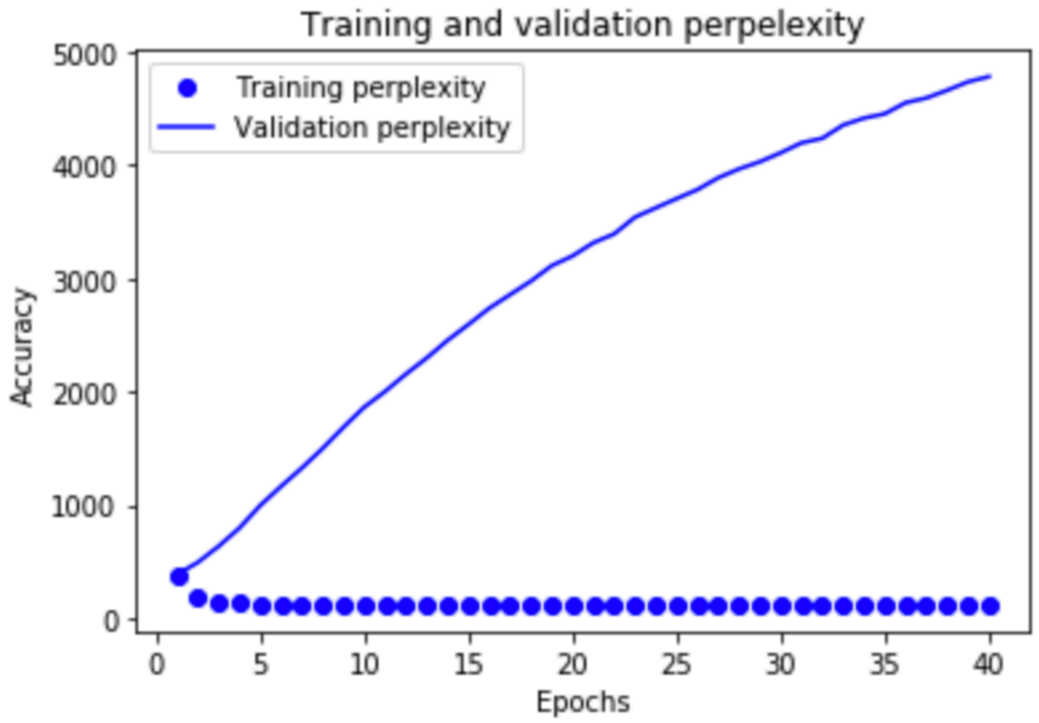
Epoch 40/40 1246754/124654 [======================] - 15s 12us/sample - loss: 3.1796 - perplexity: 124.7307 – sparse_top_k_categorical_accuracy: 0.5901 - val_perplexity: 4780.6191 – val_sparse_top_k_category_accuracy: 0.4354

Evaluate Model:

Loss: 5.3866; perplexity: 4781.1763; acc@5: 0.4401.




Training and validation perpelexity
Training perplexity
Validation perplexity
Accuracy
Epochs
0
1000
2000
3000
4000
0
5
10
15
20
25
30
35
40


Training and validation perpelexity
Training perplexity
Validation perplexity
Accuracy
Epochs
0
1000
2000
3000
4000
5000
0
5
10
15
20
25
30
35
40

Model with 3 hidden layers

LINK: https://colab.research.google.com/drive/1n64uuC6FOn3Vo22BQI8PkLor_C6R77HZ

Build the Model:

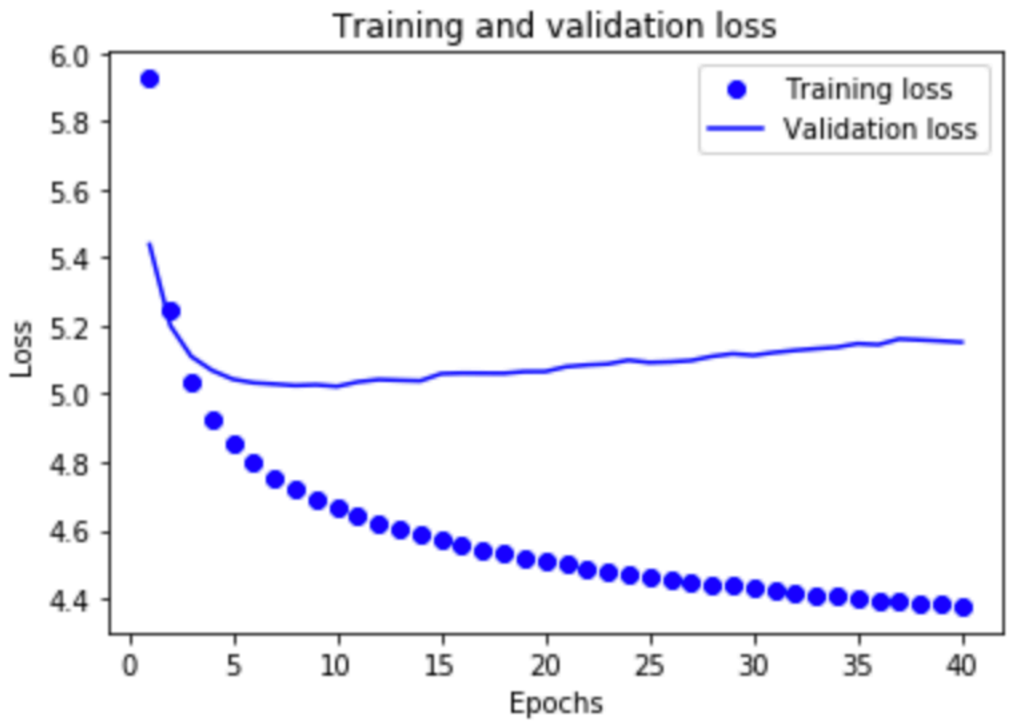
Layer(type)	**Output Shape**	**Param#**
Embedding_2(Embedding)	(None, None, 50)	500000
Reshape_2(Reshape)	(None, 250)	0
Dense_4 (Dense)	(None, 20)	5020
Dense_5(Dense)	(None,20)	420
Dense_6(Dense)	(None, 10000)	210000

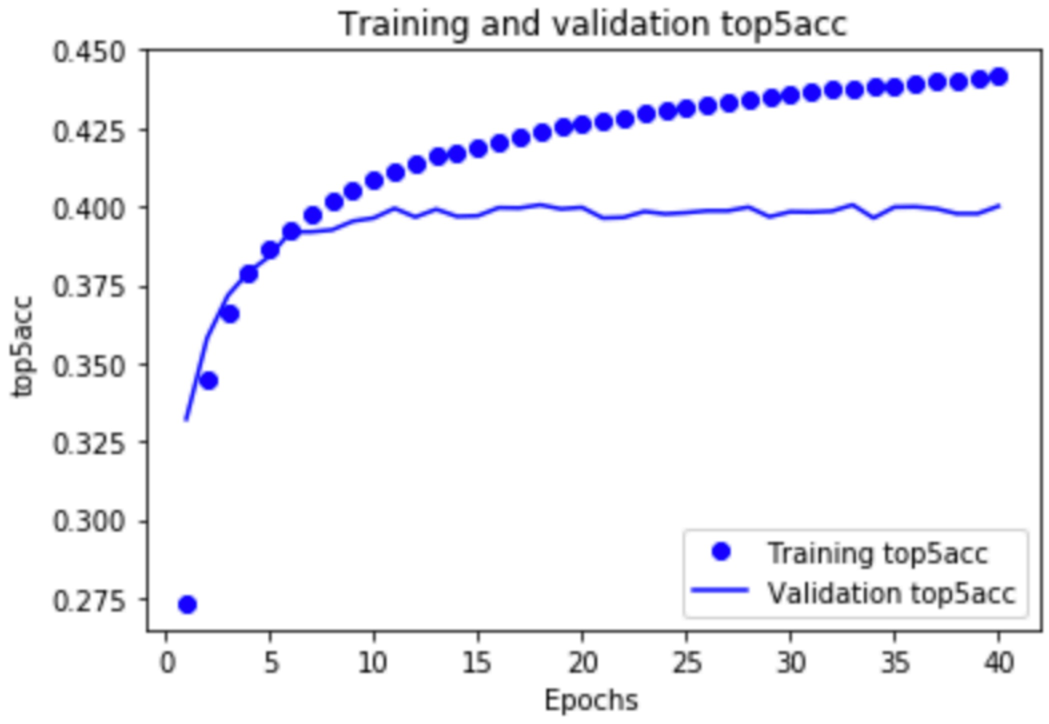
Total params: 715.440; Trainable params: 715,440 ; Non-trainable params:0

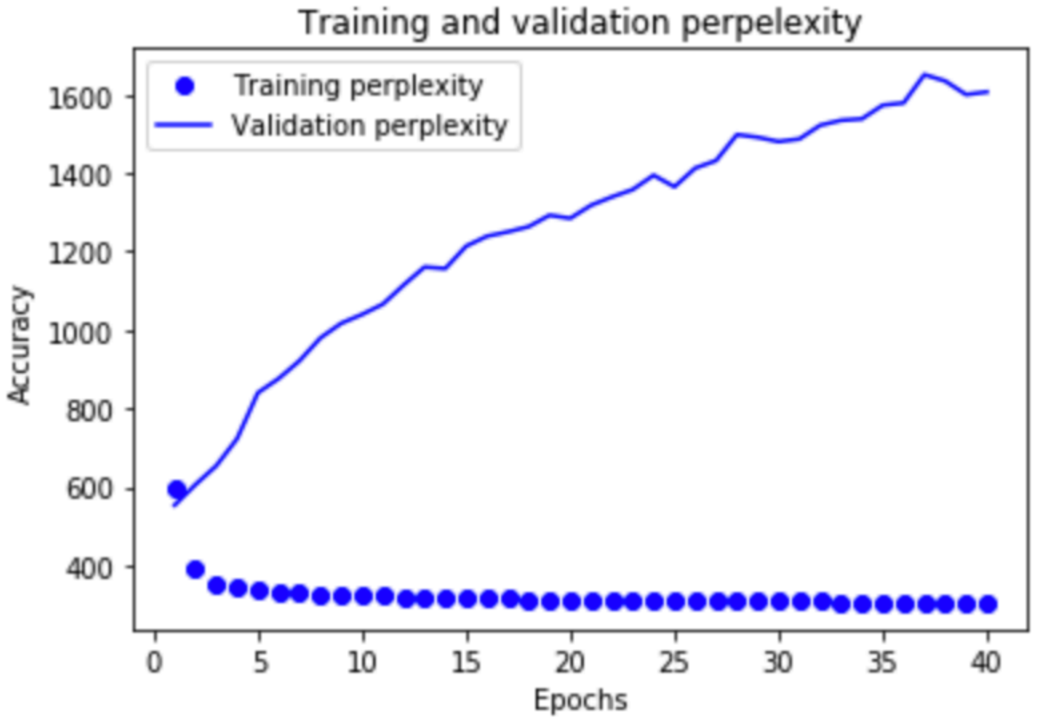
Evaluate Model:

Loss: 5.1130; perplexity: 1637.7172; acc@5: 0.4030

Epoch 40/40 1246754/1246754 [==============================] - 50s 40us/sample - loss: 4.4715 - perplexity: 442.3907 - sparse_top_k_categorical_accuracy: 0.4279 - val_loss: 5.0200 - val_perplexity: 1864.2638 - val_sparse_top_k_categorical_accuracy: 0.3958


Training and validation loss
Training loss
Validation loss
Loss
Epochs
6.0
5.8
5.6
5.4
5.2
5.0
4.8
4.6
4.4
0
5
10
15
20
25
30
35
40


Training and validation top5acc
top5acc
0.450
0.425
0.400
0.375
0.350
0.325
0.300
0.275
0
5
10
15
20
25
30
35
40
Epochs
● Training top5acc
Validation top5acc


Training and validation perpelexity
Accuracy
1600
1400
1200
1000
800
600
400
0
5
10
15
20
25
30
35
40
Epochs
● Training perplexity
Validation perplexity