Assignment 2

CMSC 673 — Natural Language Processing

Faisal Rasheed Khan

VB02734

vb02734@umbc.edu

**Question 1----------**

1.

a. This is a binary classification problem as we need to determine whether it is grammatically acceptable or not.

b.
feature1= [ 1 if the sentence is in noun-verb agreement, else 0]
feature2= [ 1 if the sentence noun-verb both are either both plural or both singular, else 0]
feature3= [ 1 if the sentence lemmas(i.e root word of noun/verb) noun-verb are either both singular or both plural, else 0]
x1="she loves NLP" and x2 ="he love NLP."
f(x1)=[1,1,1]
f(x2)=[1,0,1]
it's a 3 feature vector

c.
θ weights = number of features x number of labels
θ weights = 3x2 (here for x1 and x2)

d.
θ weights =

$$\begin{bmatrix} -1 & 1 \\ -1 & 0 \\ -1 & 0 \end{bmatrix}$$

Features are 3x2
$\Theta^T$ = 2x3
And the resultant will be 2x2
Logits = $\Theta^T$ * f(x)

$$\begin{bmatrix} -1 & -1 & -1 \\ 1 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 1 & 1 \\ 0 & 1 \\ 1 & 1 \end{bmatrix}$$

Logits = $\begin{bmatrix} -3 & -2 \\ 1 & 0 \end{bmatrix}$

e.
z(x1) = exp(u(1,0)) +exp(u(1,1))
z(x1) = exp(-3) + exp(-2)
z(x1) = 0.185

z(x2) = exp(u(2,0)) +exp(u(2,1))
z(x2) = exp(1) + exp(0)
z(x2) = 3.71

f.
p( y | x1) = exp(u(1,y)) / z(x1)

p(0 | x1)= exp(u(1,0)) / 0.185
p(0 | x1)= exp(-3) / 0.185
p(0 | x1)= 0.269

p(1 | x1)= exp(u(1,1)) / 0.185
p(1 | x1)= exp(-2) / 0.185
p(1 | x1)= 0.729
p( y | x1) =[0.269 , 0.729]

p( y | x2) = exp(u(1,y)) / z(x2)

p(0 | x2)= exp(u(1,0)) / 3.71
p(0 | x2)= exp(1) / 3.71
p(0 | x2)= 0.73

p(1 | x2)= exp(u(1,1)) / 3.71
p(1 | x2)= exp(0) / 3.71
p(1 | x2)= 0.26

p( y | x1) =[0.73, 0.26]

g.
argmax(p( y| x1))= argmax(p( 0| x1), p( 1| x1))
= argmax( [0.269 , 0.729])
x1="she loves NLP" is acceptable

argmax(p( y| x2))= argmax(p( 0| x2), p( 1| x2))
= argmax( [0.73, 0.26])
x2 ="he love NLP."

h.
if the weights were uniformly distributed then the probabilities will be equal for every label.
z(x1) = exp(u(1,0)) +exp(u(1,1))
z(x1) = exp(0) + exp(0)
z(x1) = 2
p( y | x1) = exp(u(1,y)) / z(x1)

p(0 | x1)= exp(u(1,0)) / 2
p(0 | x1)= exp(0) / 2
p(0 | x1)= 0.5

p(1 | x1)= exp(u(1,1)) / 2
p(1 | x1)= exp(0) / 2
p(1 | x1)= 0.5
p( y | x1) =[0.5 , 0.5]

z(x2) = exp(u(2,0)) +exp(u(2,1))
z(x2) = exp(0) + exp(0)
z(x2) = 2
p( y | x2) = exp(u(2,y)) / z(x2)

p(0 | x2)= exp(u(2,0)) / 2
p(0 | x2)= exp(0) / 2
p(0 | x2)= 0.5

p(1 | x2)= exp(u(2,1)) / 2
p(1 | x2)= exp(0) / 2
p(1 | x2)= 0.5
p( y | x2) =[0.5 , 0.5]

**Question 2----------**

2.

a.

correct labels = [T, T, F, T, F, T, F, T]
predicted labels = [T, F, T, T, F, F, F, T]

| | Actual : T | F |
|---|---|---|
| Predicted T | TP 3 | FP 1 |
| F | FN 3 | TN 3 |

Precision = $\frac{TP}{TP+FP}$ =$\frac{3}{3+1}$=0.75

Recall = $\frac{TP}{TP+FN}$ =$\frac{3}{3+2}$=0.6

Accuracy = $\frac{TP+TN}{TP+FP+FN+TN}$ =$\frac{3+2}{3+1+2+2}$=$\frac{5}{8}$ =0.625

F1 = $\frac{2*Precision*Recall}{Precision+Recall}$ = $\frac{2*TP}{2*TP+FP+FN}$ = $\frac{2*0.75*0.6}{1.35}$ = $\frac{0.9}{1.35}$ = 0.666

b.

correct labels = [T, F, F, F, F, F, F, T]
predicted labels = [F, T, F, F, F, F, F, F]

| | | |
|---|---|---|
| | Actual : T | F |
| Predicted : T | TP 0 | FP 1 |
| F | FN 2 | TN 5 |

Precision = $\frac{TP}{TP+FP}$ =$\frac{0}{0+1}$=0

Recall = $\frac{TP}{TP+FN}$ =$\frac{0}{0+2}$=0

Accuracy = $\frac{TP+TN}{TP+FP+FN+TN}$ =$\frac{0+5}{0+1+2+5}$=$\frac{5}{8}$ =0.625

F1 = $\frac{2*Precision*Recall}{Precision+Recall}$ = $\frac{2*TP}{2*TP+FP+FN}$ = $\frac{2*0}{2*0+1+2}$ = 0

c.

correct labels are [T, F, U, F, F, F, U, T]
predicted labels are [F, T, U, F, F, F, F, T]

| | Actual: T | U | F |
|---|---|---|---|
| Predicted: T | 1 | FP for T 0 | FP for T 1 |
| U | FN for T 0 | 1 | 0 |
| F | FN for T 1 | 1 | 3 |

F- Default Class, and 2 positive classes = T,U

Accuracy = $\frac{\sum_c TP_c + TN_c}{\sum_c TP_c + TN_c + FN_c + FP_c}$ = $\frac{(1+1)+3}{(1+1)+(1+0)+3+(1+1)}$=$\frac{5}{8}$ = 0.625

Macro Precision = $\frac{1}{c}\sum_c \frac{TP_c}{TP_c + FP_c}$ = $\frac{1}{2}(\frac{1}{1+1} + \frac{1}{1+0})$ = $\frac{3}{4}$ = 0.75

Macro Recall = $\frac{1}{c}\sum_c \frac{TP_c}{TP_c + FN_c}$ = $\frac{1}{2}(\frac{1}{1+1} + \frac{1}{1+1})$ = $\frac{1}{2}$ = 0.5

Micro Precision = $\frac{\sum_c TP_c}{\sum_c TP_c + \sum_c FP_c}$ = $\frac{(1+1)}{(1+1+0)+(1+0+0))}$=$\frac{2}{3}$ = 0.666

Micro Recall = $\frac{\sum_c TP_c}{\sum_c TP_c + \sum_c FN_c}$ = $\frac{(1+1)}{(1+1+0)+(1+1+0))}$=$\frac{2}{4}$ = 0.5

Macro F1 = $\frac{2 * Macro\ Precision * Macro\ Recall}{Macro\ Precision + Macro\ Recall}$ = $\frac{2*\frac{3}{4}*\frac{1}{2}}{\frac{3}{4}+\frac{1}{2}}$ = $\frac{3}{5}$ =0.6

Micro F1 = $\frac{2 * Micro\ Precision * Micro\ Recall}{Micro\ Precision + Micro\ Recall}$ = $\frac{2*\frac{2}{3}*\frac{2}{4}}{\frac{2}{3}+\frac{2}{4}}$ = $\frac{8}{14}$ =0.571

$Recall_T$ = $\frac{TP_T}{TP_T + FN_T}$= $\frac{1}{1+1}$ =0.5

$Recall_U$ = $\frac{TP_U}{TP_U + FN_U}$= $\frac{1}{1+1}$ =0.5

AverageRecall = $\frac{1}{2}$ ($Recall_T$ + $Recall_U$ ) = $\frac{1}{2}$ ( $\frac{1}{2}$ +$\frac{1}{2}$ ) =0.5

$Precision_T$ = $\frac{TP_T}{TP_T + FP_T}$= $\frac{1}{1+1}$ =0.5

Precision$_U$ = $\frac{TP_U}{TP_U + FP_U}$= $\frac{1}{1+0}$ =1

F1$_T$ = $\frac{2 * P_T * R_T}{P_T + R_T}$= $\frac{2*\frac{1}{2}*\frac{1}{2}}{\frac{1}{2}+\frac{1}{2}}$ =0.5

F1$_U$ = $\frac{2 * P_U * R_U}{P_U + R_U}$= $\frac{2*1*\frac{1}{2}}{1+\frac{1}{2}}$ = 0.666

Not included F Class in Precision, Recall and F1 in the calculation because it is negative class and remaining 2 classes T,U are positive class and it is a multi class classification with negative label F. In order to see how the positive labels performed we consider only T,U classes and do not take F class in considertation.

d.

correct labels = [C, C, A, C, C, C, C, C, B, A, C, C, C]
predicted labels = [C, C, C, C, C, C, C, C, B, A, A, C, C]

| | Actual: A | B | C |
|---|---|---|---|
| Predicted: A | 1 | FP for A 0 | FP for A 1 |
| B | FN for A 0 | 1 | 0 |
| C | FN for A 1 | 1 | 3 |

Here all classes are considered

Accuracy = $\frac{\sum_c TP_c + TN_c}{\sum_c TP_c + TN_c + FN_c + FP_c}$ = $\frac{(1+1)+9}{(1+1+9)+(1+0+1)}$=$\frac{11}{13}$ = 0.846

Macro Precision = $\frac{1}{c}\sum_c \frac{TP_c}{TP_c + FP_c}$ = $\frac{1}{3}$ ($\frac{1}{1+1} + \frac{1}{1+0} + \frac{9}{9+1}$) = $\frac{2.4}{3}$ = 0.8

Macro Recall = $\frac{1}{c}\sum_c \frac{TP_c}{TP_c + FN_c}$ = $\frac{1}{3}$ ($\frac{1}{1+1} + \frac{1}{1+0} + \frac{9}{9+1}$) ) = 0.8

Micro Precision = $\frac{\sum_c TP_c}{\sum_c TP_c + \sum_c FP_c}$ = $\frac{(1+1+9)}{(1+1+9)+(1+0+1))}$=$\frac{11}{13}$ = 0.846

Micro Recall = $\frac{\sum_c TP_c}{\sum_c TP_c + \sum_c FN_c}$ = $\frac{(1+1+9)}{(1+1+9)+(1+0+1))}$=$\frac{11}{13}$ = 0.846

Macro F1 = $\frac{2 * Macro\ Precision * Macro\ Recall}{Macro\ Precision + Macro\ Recall}$ = $\frac{2*\frac{8}{10}*\frac{8}{10}}{\frac{8}{10}+\frac{8}{10}}$ = $\frac{8}{10}$ =0.8

Micro F1 = $\frac{2 * Micro\ Precision * Micro\ Recall}{Micro\ Precision + Micro\ Recall}$ = $\frac{2*\frac{11}{13}*\frac{11}{13}}{\frac{11}{13}+\frac{11}{13}}$ = $\frac{11}{13}$ =0.846

Recall$_A$ = $\frac{TP_A}{TP_A + FN_A}$= $\frac{1}{1+1}$ = 0.5

Recall$_B$ = $\frac{TP_B}{TP_B + FN_B}$= $\frac{1}{1+0}$ = 1

Recall$_C$ = $\frac{TP_C}{TP_C + FN_C}$= $\frac{9}{9+1}$ = 0.9

AverageRecall = $\frac{1}{3}$ (Recall$_A$ + Recall$_B$ + Recall$_C$) = $\frac{1}{3}$ ( $0.5 + 1 + 0.9$ ) =0.8

Precision$_A$ = $\frac{TP_A}{TP_A + FP_A}$= $\frac{1}{1+1}$ =0.5

Precision$_B$ = $\frac{TP_B}{TP_B + FP_B}$= $\frac{1}{1+0}$ =1

Precision$_C$ = $\frac{TP_C}{TP_C + FP_C}$= $\frac{9}{9+1}$ =0.9

$F1_A$ = $\frac{2 * P_A * R_A}{P_A + R_A}$= $\frac{2*\frac{1}{2}*\frac{1}{2}}{\frac{1}{2}+\frac{1}{2}}$ =0.5

$F1_B$ = $\frac{2 * P_B * R_B}{P_B + R_B}$= $\frac{2*1*1}{1+1}$ = 1

$F1_C$ = $\frac{2 * P_C * R_C}{P_C + R_C}$= $\frac{2*\frac{9}{10}*\frac{9}{10}}{\frac{9}{10}+\frac{9}{10}}$ = $\frac{9}{10}$ =0.9

Here every class is important, here we need to find for what class the particular instance is for. Each class has importance and based on the highest probability of that class, the instance is assigned to that class, this is also a multi class. Each class will have its own importance and our criteria should be to map the instance to their respective classes

**Question 3----------**

3.
November, Novembrer, and Novembr
November is not broken into sub words where as the other 2 words are broken by specifying ##, the prefix word is before the ## word. (Nov is the prefix word)

```
['[CLS]', 'November', '[SEP]']
['[CLS]', 'Nov', '##em', '##bre', '##r', '[SEP]']
['[CLS]', 'Nov', '##em', '##b', '##r', '[SEP]']
```

Bert uses this kind of breaking the words such that it can handle the tokens which have the same prefix without wasting the memory, and the Out of Vocabulary words could also be handled if they have the same prefix.

The(Assignment1) tokens of the train data of the UD EWT Corpus is 207053 and the vocabulary is 20132.
Bert tokens of the train data of the UD EWT Corpus is 266170 and the vocabulary is 14704.
The tokens of Bert compared to the UD EWT Corpus tokens are huge, because the Bert breaks down the words to sub words that's why the tokens are large and the many tokens are repeated that's why Berts vocabulary is less compared to the UD EWT Corpus vocabulary.

The tokens `[CLS], [SEP]` and the words which are broken into subwords with ## are seen in the bert tokenizer. The [CLS] is there at the start index of the token, while the [SEP] is at the last index.

The benefits of defining Bert Tokenizer:
It can handle Out of Vocabulary words as it divides the words into sub words
Could have more vocabulary by consuming less space compared to the existing predefined vocabulary.
Could handle the words which have same prefix as Bert breaks the prefix of the word.

The benefits of defining Predefined Vocabulary:
Handles the text of a particular topic specific vocabulary.
The tokens are easy to create.

**Question 4----------**

4.

a.

instances are α, α, β, γ, α

most frequent baseline = α

b.

The most frequent label is 0

```
Accuracy: 0.5270758122743683
Macro Precision: 0.26353790613718414
Macro Recall: 0.5
Micro Precision: 0.5270758122743683
Micro Recall: 0.5270758122743683
```

The model is class biased to the one label (which is 0) and the model is predicting everything as 0. Although accuracy is 52% but the model hasn't learned anything.

```
tp 0
fp 0
fn 131
tn 146
```

Its predicting just 0 class.

**Question 5----------**

5.

a.

As the running time is large reduce the epoch to 3 and batchsize is 100 for 1000 training instances and predicted for 100 validation insatnces, loss function is crossentropy loss and activation function is softmax

```
Accuracy: 0.48
Macro Precision: 0.48684210526315785
Macro Recall: 0.4903846153846154
Micro Precision: 0.48
Micro Recall: 0.48
tp 36
fp 40
fn 12
tn 12
```

Changed the epoch to 2 and batchsize is 100 for 1000 training instances and predicted for 100 validation insatnces, loss function is crossentropy loss and activation function is softmax

```
Accuracy: 0.55
Macro Precision: 0.601364522417154
Macro Recall: 0.5625
Micro Precision: 0.55
Micro Recall: 0.55
tp 42
fp 39
fn 6
tn 13
```

b.
batchsize is 100 and epochs are 330, loss function is crossentropy loss and activation function is softmax(trained and validated for all instances of train and validation)

```
Accuracy: 0.5054151624548736
Macro Precision: 0.5049280350438048
Macro Recall: 0.5049409181219283
Micro Precision: 0.5054151624548736
Micro Recall: 0.5054151624548736


The best accuracy if I modified the epochs was this:
Accuracy: 0.5415162454873647
Macro Precision: 0.543108857053226
Macro Recall: 0.543108857053226
Micro Precision: 0.5415162454873647
Micro Recall: 0.5415162454873647

tp 75
fp 71
fn 56
tn 75
```

c.
The most-common baseline learns nothing, it just predicts the most frequent variable
The features used for the part a is overlapping the embeddings and getting the resultant features via one hot encoding

The embeddings here are defined by the Bert Model and sentence1 and sentence2 embeddings are merged based on the cosine similarity.

The accuracy of part a is 48% for 3 epochs and for bert embedding features is 50%
For 2 epochs the accuracy i got is 55% for part a

The best observed accuracy for Dense Embeddings from bert is 54%

The learnings are not that good for both of them part a and part b(bert) but they are better compared to the most frequent baseline classifier, as most frequent baseline classifier just predicts the most frequent variable and its model learns nothing.

Tried combining 2 features of finding the hypothesis words in premise if yes then 1 else 0 and embeddings from bert model with cosine similarity but there were problems to run the model, as its takes a long run.

The lexically defined features tokenizer length was large compared to the bert embeddings length(768). Unable to compute all lexically defined instances and a decent number of epochs, but if that model would have run on large clusters then it might have performed better than the dense embeddings features.
In the bert dense embeddings the each length is 768 so it ran in the effective time compared to lexically defined features. The accuracy is 54% for bert using cosine similarity, could have been better

if used another kind of features, may be multiple features. It runs faster compared to lexically defined features.

References:


Welcome to PyTorch Tutorials — PyTorch Tutorials 2.0.1+cu117 documentation

https://pytorch.org/tutorials/beginner/basics/autogradqs_tutorial.html#disabling-gradient-tracking

02-what-is-nlp.pdf (umbc.edu)

03-nlp-task-overview.pdf (umbc.edu)

04-ml-eval.pdf (umbc.edu)

05-classification-linear.pdf (umbc.edu)

torch.nn.functional.cosine_similarity — PyTorch 2.1 documentation