NLP Assignment 1 Report, Group No. 32

Members –
Shreeyash Khalate, MT22069
Amey Pawar, MT22010
Swaib Ilias Mazumder, MT22078
Rahul IIITD, 2021554

TASK 1: Byte Pair Encoding and Tokenization

Assumptions:

1. corpus.txt does not contain any special symbols
2. Tokenization is carried solely based on merge rules generated
3. tokens.txt contains all the found tokens in the input sample
4. vocab.txt contains all the found tokens in the whole corpus

Methodology:

There are 2 methods in Tokenizer() class:

1. learn_vocabulary(corpus, num_of_merges):
    1. Each word is initially divided into a stream of characters and a character separator '-' is used to separate characters in each word.
    2. At the end of every word a special symbol '$' is added as the word separator
    3. For every merge iteration, a pair frequency is calculated for each character pair and the pair frequency with maximum frequency is selected as the merge rule and appended to merge_rules list
    4. Every pair in the corpus corresponding to this merge rule, is merged as a single token for next iteration

2. tokenize(sample):
    1. Initially a '$' is appended to the end of each word in sample
    2. Each word is divided into individual characters
    3. Each rule in merge_rules list is scanned sequentially and the corresponding pairs of characters are combined in the word
    4. We also keep track of all the intermediate tokens generated and write them to tokens.txt
    5. After scanning all the merge rules, we return the created tokens of the given sample (and also write them to tokenized_samples.txt)

Evaluation: (all files attached)

a. tokens.txt -> all the tokens found while tokenizing a sample
b. vocab.txt -> all the possible tokens in the vocabulary i.e. the whole corpus
c. tokenized_samples.txt -> all actual tokens generated for the given sample
d. merge_rules.txt -> all merge rules, given the number of merges (we gave 10)

TASK 2: Bigram Language Modelling, Smoothing and Emotion Oriented Sentence Generation

Assumptions:

1. End of sentence marker is “$”, which also acts as the start token for generation of sentences if no start token is provided

Methodology:

<Each of the functionalities are implemented in separate/helper methods>

1. The learn_model(corpus) method creates the unigram and bigram dictionaries with their counts. It further builds Laplace and KneserNey probability dictionaries.
2. It also further adds emotion component for each of the bigram probability in the bigram dictionary and stores this computed dictionary in the pickle file for future reference.
3. The function laplace_smoothing_probability(bigram) computes and returns the Laplace probability of the bigram
4. The function kneserney_smoothing_probability (bigram) computes and returns the KneserNey probability of the bigram
5. The functions:
   generate_next_token(prev_token)
   generate_next_token_using_laplace(prev_token)
   generate_next_token_using_kneserney(prev_token)
   will generate next token based on prev token either by standard bigram probability, laplace probability or kneserney probability
6. Similarly, the functions:
   generate_sentences_standard_bigram(num_samples, start_token)
   generate_sentences_laplace(num_samples, start_token)
   generate_sentences_kneserney(num_samples, start_token)
   will generate sentences based on standard bigram probability, laplace probability or kneserney probability.
   If start token is not provided then the “$” acts as a start token by default
7. The compare_probabilities(prev_token) compares the standard bigram, laplace and kneserney probabilities using a dataframe
8. **Argument:**
   As we can see, their is a huge difference in Laplace probabilities and original Bigram probabilities. This is because Laplace smoothing steals a large amount of probabilities from non-zero counts of tokens to distribute it into tokens with zero occurrences. Thus in Laplace smoothing, the reconstructed counts of non-zero tokens could change largely sometimes by a factor of 10 from their original counts. For example, token "i" has a Laplace probability of 0.269 compared to its original probability of 0.87 i.e. a 1/3rd of difference from original probability while KneserNey has somewhat managed to maintain the probability in proportion with original probability. Thus Laplace haven't worked in our Bigram model well and doesn't work well in general for n-grams.
   On the other hand, Kneser-Ney has somewhat maintained the probabilities with original Bigram probabilities compared to Laplace.
   Thus KneserNey smoothing is better than Laplace in this case

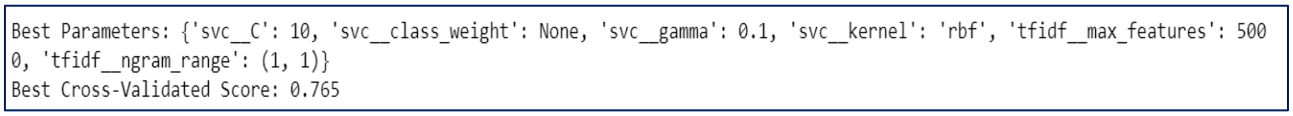
9. The **beta in the emotion component is incorporated at bigram as well as sample level** in such a way :

   emotional_bigram_probability = (1-β)*standard_bigram_prob + (β*emotion_factor)
   where emotion_factor is the emotion score of the input bigram for a specific emotion

   For sample level beta incorporation, we generated multiple samples for given emotion, and returned the top 50 samples ranked by their sentence level emotion scores for that emotion

10. Extrinsic Evaluation:
    We generated 50 samples for each emotion and stored in <emotion>.txt files. And we trained the SVC model using TFIDF vectorizer and used grid search for selecting the best parameters

    Best cross-validated score was: 0.765
    Best parameters obtained through Grid Search were:
    Best Parameters: {'svc__C': 10, 'svc__class_weight': None, 'svc__gamma': 0.1, 'svc__kernel': 'rbf', 'tfidf__max_features': 5000, 'tfidf__ngram_range': (1, 1)}



```
Best Parameters: {'svc__C': 10, 'svc__class_weight': None, 'svc__gamma': 0.1, 'svc__kernel': 'rbf', 'tfidf__max_features': 500
0, 'tfidf__ngram_range': (1, 1)}
Best Cross-Validated Score: 0.765
```

11. The accuracy obtained was: 0.91 (91%)

Evaluation:

1. Top 5 bigrams before and after applying smoothing

   Top 5 Bigrams before Smoothing:

('href', 'http')	1.0
('mooshilu', '$')	1.0
('tychelle', 'to')	1.0
('hang', 'out')	1.0
('nonexistent', 'social')	1.0

   Top 5 Bigrams after Laplace Smoothing:

('$', 'i')	0.2693486590038314
('i', 'feel')	0.11042412409155006
('feel', 'like')	0.035092684307343996
('i', 'am')	0.03189066059225513
('$', 'im')	0.027203065134099615

   Top 5 Bigrams after Kneser-Ney Smoothing:

('don', 't')	0.970358782691682
('href', 'http')	0.9700023363576185
('didn', 't')	0.9583657827447011
('sort', 'of')	0.9565093900961343
('supposed', 'to')	0.9183832405280168

2. Reasoning for method used for including emotion component
   The emotion component is incorporated in the following way:
   emotional_bigram_probability = (1-β)*standard_bigram_prob + (β*emotion_factor)

   As for a given β, (1- β) weight is given to the standard bigram probability and β weight is given for the emotional factor of that bigram for a specific emotion. This focuses more on the emotion of the bigram rather than it's own probability of occurrence (given the β is high) and a separate normalization is not required as the weight is distributed such that the final probabilities will always sum upto 1. The context of the generated text won't make any difference in case of bigrams as it only looks for limited context (1 word) in the past. Thus the emotional factor can be given more weightage than the bigram probability. We took β=0.7, i.e. 70% weightage to emotional factor of the bigram and 30% to its probability of occurrence

3. Generated samples for each emotion
   Samples for sadness
   $ this journey of x amount of dumb feeling disheartened
   $ is alone thinking i stand feeling disheartened $ ill

   Samples for joy
   $ during our quirky little too feel welcomed here alive
   $ when ever feeling kinda popular distribution and blessings pronounced

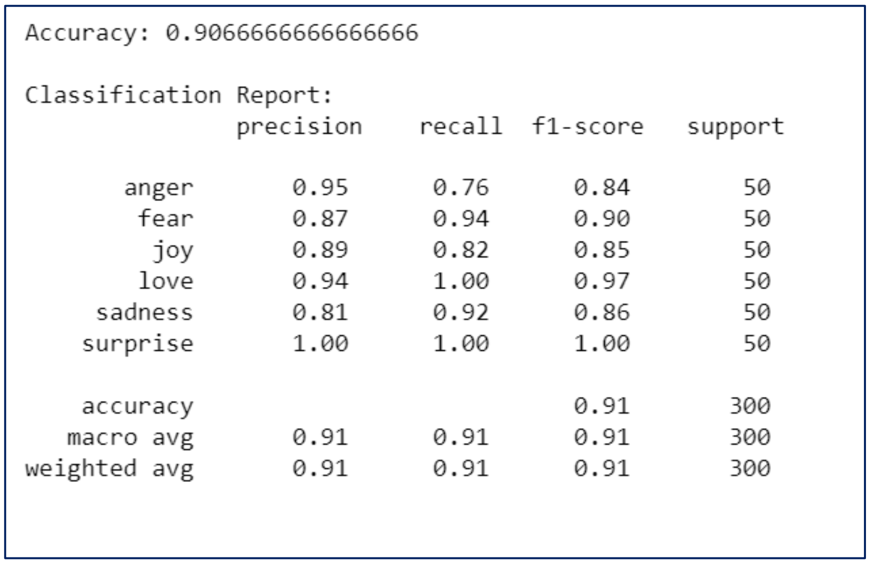
   Samples for love
   $ i liked the nostalgic atmosphere it warms my voice
   $ i enjoyed the naughty for supportive examples $ no

   Samples for anger
   $ no energy every believer should feel bothered being way
   $ is here long so dissatisfied just because unlike batman

   Samples for fear
   $ id be baffled as neurotic feeling afraid to step
   $ id feel afraid if not writing to midori that

   Samples for surprise
   $ i mostly feel shocked that shocked me shocked at
   $ i was shocked looking there feeling shocked over and

4. Accuracy and macro F1 Scores:
   Accuracy obtained was: **0.9066666666666666**



```
Accuracy: 0.9066666666666666

Classification Report:
              precision    recall  f1-score   support

       anger       0.95      0.76      0.84        50
        fear       0.87      0.94      0.90        50
         joy       0.89      0.82      0.85        50
        love       0.94      1.00      0.97        50
     sadness       0.81      0.92      0.86        50
    surprise       1.00      1.00      1.00        50

    accuracy                           0.91       300
   macro avg       0.91      0.91      0.91       300
weighted avg       0.91      0.91      0.91       300
```

5. Reasoning for generation of samples

   1. Sadness = $ this journey of x amount of dumb feeling disheartened

      Reason: As the start token is not given, hence starting with ‘$’ as previous token, the next token with highest probability of occurrence for given emotion sadness would’ve been ‘this’ combined with (emotion score of ‘$ this’ * β) and so on for selecting next tokens in the sentence. Also the bigrams “dumb feeling” and “feeling disheartened” might have more emotion scores in context of sadness and hence be generated

   2. joy = $ during our quirky little too feel welcomed here alive

      Reason: As the start token is not given, hence starting with ‘$’ as previous token, the next token with highest probability of occurrence for given emotion joy would’ve been ‘during’ combined with (emotion score of ‘during’ * β) and so on for selecting next tokens in the sentence. Also the bigrams “feel welcomed” and “welcomed here” might have more emotion scores in context of joy and hence be generated

   3. love = $ i liked the nostalgic atmosphere it warms my voice

      Reason: As the start token is not given, hence starting with ‘$’ as previous token, the next token with highest probability of occurrence for given emotion love would’ve been ‘i’ combined with (emotion score of ‘i’ * β) and so on for selecting next tokens in the sentence. Also the bigrams “i liked” and “liked the” and “it warms” might have more emotion scores in context of love and hence be generated

   4. anger = $ no energy every believer should feel bothered being way

      Reason: As the start token is not given, hence starting with ‘$’ as previous token, the next token with highest probability of occurrence for given emotion anger would’ve been ‘no’ combined with (emotion score of ‘no’ * β) and so on for selecting next tokens in the sentence. Also the bigrams “no energy” and “feel bothered” might have more emotion scores in context of anger and hence be generated

5. fear = $ id be baffled as neurotic feeling afraid to step

   Reason: As the start token is not given, hence starting with ‘$’ as previous token, the next token with highest probability of occurrence for given emotion fear would’ve been ‘id’ combined with (emotion score of ‘id’ * β) and so on for selecting next tokens in the sentence. Also the bigrams “be baffled” and “feeling afraid” and “afraid to” might have more emotion scores in context of anger and hence be generated

6. surprise = $ i mostly feel shocked that shocked me shocked at

   Reason: As the start token is not given, hence starting with ‘$’ as previous token, the next token with highest probability of occurrence for given emotion surprise would’ve been ‘i’ combined with (emotion score of ‘i’ * β) and so on for selecting next tokens in the sentence. Also the bigrams “feel shocked”, “shocked that” and “shocked me”, etc. might have more emotion scores in context of fear and hence be generated

Credit Statement:
MT22010, Amey Pawar
Completed Task 1 – learn_vocabulary() method
Completed Task 2 – SubTask4 – Extrinsic Evaluation

MT22069, Shreeyash Khalate
Completed Task2-SubTask1 & SubTask2 & SubTask3 - Implementation of Bigram Language Model with supporting methods for next token selection and sentence generation using all 3 techniques i.e. standard bigram, kneserney

MT22078, Swaib Ilias Mazumder
Completed Task1 – tokenizer() method, added emotion component to bigram probabilities and Report

2021554, Rahul IIITD
Completed Task2 SubTask2 – Laplace Implementation and Comparison of Probabilities