NLP Assignment 2

Neemesh Yadav (2020529) | Mohammad Aflah Khan (2020082)

Files Present in Submission:

- A2 Neg.ipynb: Used to generate negative sentences in parallel with positive using same model
- A2 Pos.ipynb: Used to generate positive sentences in parallel with negative using same model
- A2_test_dataset_preprocessed.csv: Preprocessed Test Set to Save time during Testing
- LanguageModels.py: All the language model utils
- Bigrams.py: File to generate bigrams and store them
- Customregexes.py: Preprocessing Regex Utils
- preprocess_text.py: Main Preprocessing File
- evaluation.py: Has utils for testing
- get_top_k_words.py: Retrieves top-250 words for prompting
- intrinsicEval.ipynb: Notebook which shows Intrinsic Eval over all data generated
- smoothers.py: Implementation of Smoothing Function
- testMain.py: Runs all models and reports all accuracies
- Unigram_sentiment_dumps.py: Generates Dump of Sentiment for each token used during generation

Individual Contributions:

Both members ideated together before implementing their respective sections

- Neemesh: Implemented LM, Smoothing Techniques, Evaluation Processes and Preprocessing
- Aflah: Implemented LM, Smoothing Techniques and Bigram Generation

Methodology Used

- For Text Preprocessing we reuse our pipeline from the last assignment with certain modifications. Our steps are as follows:
    - Lowercase Text (This significantly reduces the number of tokens)
    - Remove URL and HTML Tags
    - Remove Usernames
    - Remove Punctuations (Since the tweets are really small, punctuations are not very useful in capturing semantics and in our experiments caused poor results as very often users used things like … which would increase the bigram counts for . followed by a .)
    - Remove extra whitespaces
    - Tokenization using NLTK's tweet Tokenizer

    - Spelling Correction using Autocorrect/Textblob library to reduce number of tokens optionally
    - Remove tokens which are numbers majorly as they are rare are not very useful
- We go over the vocabulary and find the top 250 most positive and most negative words according to Vader and Hugging Face Sentiment Model
- We also tested 3 different smoothing techniques namely Laplace, Add K and Add K with Unigram Priors. We noticed Laplace is good enough for our use case and the others don't bring any new gains so we stick with it
- We then use these words as starting tokens in our model with different configurations:
    - We multiply our numerator optionally with the sentiment score of the predicted tokens to choose more sentiment oriented tokens
    - We divide our denominator optionally for the same reason
    - We externally add a factor to steer our model towards sentiment oriented tokens by adding the sentiment score of the token
    - We penalize our model for repetitions
    - We optionally penalize our model for increasing perplexity by dividing the probability scores by the perplexity
    - We add randomness for diverse generation by picking any of the top 5 tokens at each time step weighted by their probabilities

Top 4 Bigrams after smoothing by Laplace Smoothing are:

- ('in', 'the'), 0.015780400477390266
- ('going', 'to'), 0.015692868151884545
- ('i', 'have'), 0.012978294920563885
- ('for', 'the'), 0.01256447559846581

Results

Accuracy on A1 dataset: 87.27

Type	Accuracy	Perplexity
Using the prompts generated using vader polarity scores, and adding beta externally	89.75	Negative: 1149.5631564240427 Positive: 1033.066400924827 Average: 1091.31477867
Using the prompts generated using vader polarity scores, adding beta in the numerator and externally	89.91	Negative: 6955.083128301696 Positive: 6892.620049010152 Average: 6923.85158866
Using the prompts generated using vader polarity scores, and multiplying beta with the numerator	89.29	Negative: 1303.5915012298801 Positive: 1026.3095571643635 Average:1164.9505292
Using the prompts generated using vader polarity scores, and dividing beta from the denominator	89.29	Negative: 1161.5361432277948 Positive: 973.9499512734698 Average: 1067.74304725
Using the prompts generated using vader polarity scores, and normalizing using perplexity	88.82	Negative: 3681.8924298403463 Positive: 2527.2396708423166 Average: 3104.56605034
Using the prompts generated using Vader Polarity Scores, using beta externally, and textblob autocorrect	88.2	Negative: 4836.6544482263425 Positive: 4252.288359524329 Average: 4544.47140388
Using the prompts generated using HuggingFace polarity scores, and adding beta externally	88.98	Negative: 3655.251042664615 Positive: 2622.3527374782907 Average: 3138.80189007

Generated Sentences and Some Intuition:

<u>Do note that our generated sequence length was set for 10</u>

1) Using the prompts generated using vader polarity scores, and adding beta externally: Here we first use the vader sentiment analyzed over the vocab to choose the 250 most positive and negative words which we then use as prompts. Subsequently we also add beta as an external factor where it depends on the token's sentiment score when it's being considered for selection multiplied by a scaling factor of 1 (We also tried other factors 1 worked best in our testing)

   Some Generated Sentences:

   a) Positive
      i) best thing in the same time i have to get
      ii) xd i love to see you and it was in

iii) joy love you are you get to the best i
iv) pleasure paradise love you are so much i am not
v) celebrate with me to go for my baby boy born

b) Negative
i) tragedy utter raping hell my phone with me feel better
ii) torture me too bad i love it was just got
iii) wtf slavery and its been a good morning at the
iv) retarded slavery and i have a good to get ready
v) ban kill me to get to the same thing is

2) Using the prompts generated using vader polarity scores, adding beta in the numerator and externally: Here we first use the vader sentiment analyzed over the vocab to choose the 250 most positive and negative words which we then use as prompts. Subsequently we also add beta as an external factor and also add the same term in the numerator scaled down where it depends on the token's sentiment score and if it's negative we negate it to be positive. This adds low value for less sentiment rich tokens and more for others. However we notice heavy repetition in these cases.

Some Generated Sentences:

a) Positive
i) inspire perfectly freedom glorious paradise love best greatest love greatest
ii) celebrate glorious love freedom perfectly greatest love glorious perfectly freedom
iii) kind greatest glorious freedom glorious perfectly paradise greatest freedom best
iv) vip freedom greatest perfectly paradise love glorious perfectly best best
v) heaven love freedom perfectly love glorious greatest paradise best freedom

b) Negative
i) raping slavery kill hell kill kill hell killed slavery slavery
ii) horrific hell raping killed killed slavery slavery hell kill killed
iii) shittiest killed slavery raping raping raping raping kill killed hell
iv) dead kill hell hell hell kill raping hell hell raping
v) vile slavery raping raping slavery kill hell kill kill raping

3) Using the prompts generated using vader polarity scores, multiplying beta to the numerator and adding it externally: Here we first use the vader sentiment analyzed over the vocab to choose the 250 most positive and negative words which we then use as prompts. Subsequently we also add beta as an external factor and also multiply the same term in the numerator scaled down where it depends on the token's sentiment score and if it's negative we negate it to be positive. Doing this retains the nature of the bigram's probability by scaling it between 0 and 1. We can already see that the results are better than the previous ones with respect to repetition.

Some Generated Sentences:

a) Positive
i) bliss glorious sunny hot relaxing monday played girls night and
ii) loves that i am in your time i dont want
iii) won im not sure you and i have a little
iv) divine greatest love you and i cant wait to see
v) honorable mention a little better than i dont think i

b) Negative
i) raping slavery and a great time to the best i
ii) kill me too early today i was just woke up

iii) drowned raping nick paid weekly webcomic hell yeah i love
iv) cruel bitch haha yeah i have the best thing in
v) hate those injured killed im in the best friend were

4) Using the prompts generated using vader polarity scores, dividing beta from the denominator and adding it externally: Here we first use the vader sentiment analyzed over the vocab to choose the 250 most positive and negative words which we then use as prompts. Subsequently we also add beta as an external factor and also divide the same term from the denominator scaled down where it depends on the token's sentiment score and if it's negative we negate it to be positive. Doing this retains the nature of the bigram's probability by scaling it between 0 and 1. We can already see that the results are better than the previous ones.

Some Generated Sentences:

a) Positive
   i) xd i was awesome haha yeah but i dont think
   ii) win xp glorious weather is a little sad that i
   iii) precious greatest girl like a great time is going to
   iv) delicious love it will not a good luck then im
   v) hero love it is a great weekend with the same
b) Negative
   i) raping nick to the last time to get my phone
   ii) whore ever she is going to see you are you
   iii) war raping nick he was awesome now im in the
   iv) wtf iloseweight slavery killed the same thing in the last
   v) bullshit id say hi message raping hell get it was

5) Using the prompts generated using vader polarity scores, adding the beta factor externally and normalizing the score using perplexity: Here we first use the vader sentiment analyzed over the vocab to choose the 250 most positive and negative words which we then use as prompts. Subsequently we also add beta as an external factor and also divide the normalized probability by its current perplexity. Doing this basically gives sentences with a lower perplexity (which is better for our generation purposes) a higher probability.

Some Generated Sentences:

a) Positive
   i) amazing and awesome lol i won the greatest paradise love
   ii) heroine precious has love you wow paradise best thing i
   iii) inspire love it amazing i hope you darling perfectly greatest
   iv) bliss n dc june is amazing love you paradise freedom
   v) rich freedom greatest love you are funny haha i loved
b) Negative
   i) raping slavery killed the shit alone evil and crisis i
   ii) infuriated kill chris raping hell my death bed killed the
   iii) cruel raping nick he is slavery and hell kill myself
   iv) terrifying getoutgetoutgetougetooooouuuuuuuuuuuuuttttttttt kill raping slavery and evil and hell tragedy
   v) crappy raping kill chris brown slavery tragedy arms legs so

6) Using the prompts generated using Vader Polarity Scores, using beta externally, and textblob autocorrect: Here we first use the vader sentiment analyzed over the vocab to choose the 250 most positive and negative words which we then use as prompts. Subsequently we also add beta as an external factor and also one thing to note is that the bigrams we now create are also passed through a preprocessing layer of spelling correction using the TextBlob library. We

made a choice between TextBlob and autocorrect_full by taking into account how long it takes to preprocess and also the difference in results, in our case there was a huge difference in the time taken and the final accuracy after Extrinsic Evaluation. We performed this experiment so that we can get a more general idea of how our results can be varied.

Some Generated Sentences

a) Positive
   i) improvement greatest show love perfectly freedom paradise great gorgeous day
   ii) interest paradise love best perfectly then greatest show good glorious
   iii) dedicated greatest games like love best friends paradise perfectly great
   iv) glad perfectly then love paradise freedom greatest girl glorious great
   v) safe best thing greatest paradise perfectly then i freedom glorious

b) Negative
   i) slavery killed tea so raping nick kill shittiest tragedy killing
   ii) havoc raping nick paid me slavery tragedy killed im not
   iii) worst dreams of the kill chris wants to raping hell
   iv) scam raping killed tea so kill me horrific ended up
   v) sufferers kill chris rocker raping slavery horrific killing it killed

7) Using the prompts generated using HuggingFace polarity scores, and adding beta externally: Here we first use the sentiment scores generated using the HuggingFace pipeline for sentiment analysis from analyzing over the vocab to choose the 250 most positive and negative words which we then use as prompts. Subsequently, we also add beta as an external factor which would take into account the sentiment of our currently generating sentence.

   Some Generated Sentences:

   a) Positive
      i) greatness of the best love you wow paradise greatest girl
      ii) shining and awesome love the best dreams and i won
      iii) lovely sunday sunday night loves u freedom mother n paradise
      iv) wont be a nice day perfectly love paradise look like
      v) nicely freedom mother white horse love you darling glorious sunny

   b) Negative
      i) pathetic hell everybody see you raping kill myself in killed
      ii) appalling kill me sad killed a bad hell yeah raping
      iii) useless bloatware kill chris raping slavery hell killed im tragedy
      iv) beaten loads of hell slavery and i killed the dead
      v) exhausting hell is a bitch in raping slavery and crisis

An interesting observation we made while performing our experiments is that the model was able to generate pragmatic/sarcastic sentences, out of randomness.

We have uploaded our bigrams on a drive folder linked here mainly because the size was too large and couldn't be uploaded directly on Google Classroom along with our other dependencies which also take up a lot of space.