50.007 Machine Learning - Summer 2023

1D Project Report

Group Members:

Anutham Mukunthan (1006202),

Jash Jignesh Veragiwala (1006185),

Atul Parida (1006184),

Ansh Oswal (1006265)

|  |
| --- |

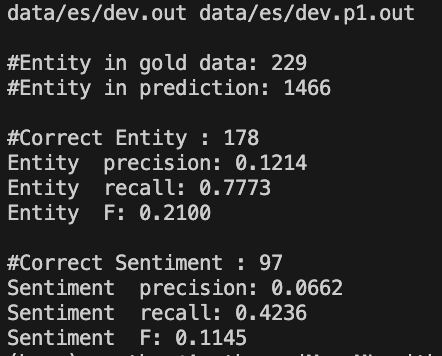
## **Task 1**

Approach:

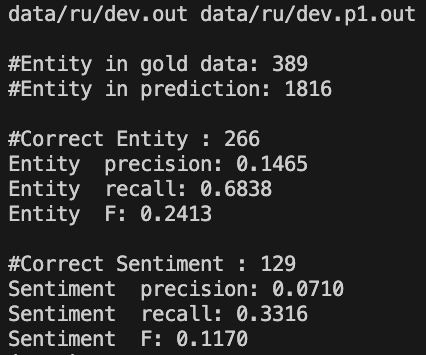
1. **Data Preparation:** The training data provided includes labeled sentences in the format of one token per line with a token and sentiment tag separated by a tab. Sentences are separated by a single empty line. The training data is loaded for both the ES (Spanish) and RU (Russian) datasets.
2. **Emission Parameter Estimation:** The emission parameters are estimated using Maximum Likelihood Estimation (MLE). The frequency of words associated with each sentiment tag is counted, and then the emission probability is calculated as the ratio of the count of that word with the sentiment tag to the total count of that sentiment tag.
3. **Handling unseen words:** To handle words not seen in the training set during testing, a special token "#UNK#" is introduced. The emission parameter calculation is modified to include a smoothing factor. If a word appears in the training set, its emission parameter is calculated as usual. If the word is "#UNK#", a fixed value (k) is added to the denominator and numerator of the emission parameter calculation.
4. **Sentiment analysis:** A simple sentiment analysis system is implemented using the calculated emission parameters. For each word in a sentence, the system predicts the sentiment tag that maximizes the emission probability for that word. If the word is not present in the training data, it is replaced by the "#UNK#" token, and the emission parameter is calculated accordingly.
5. **Prediction & output:** The implemented system predicts sentiment tags for words in the development data and writes them to an output file ("dev.p1.out"). This file contains the predicted sentiment tags for each word in the input sentences.
6. **Evaluation:** The precision, recall, and F-score of the implemented baseline sentiment analysis system are calculated using an evaluation script provided. The system's output ("dev.p1.out") is compared to the gold-standard sentiment tags provided in the development output file ("dev.out").

Result:

1. ES Dataset:



1. RU dataset:



Conclusion: Using just emission probabilities alone does not give good predictions

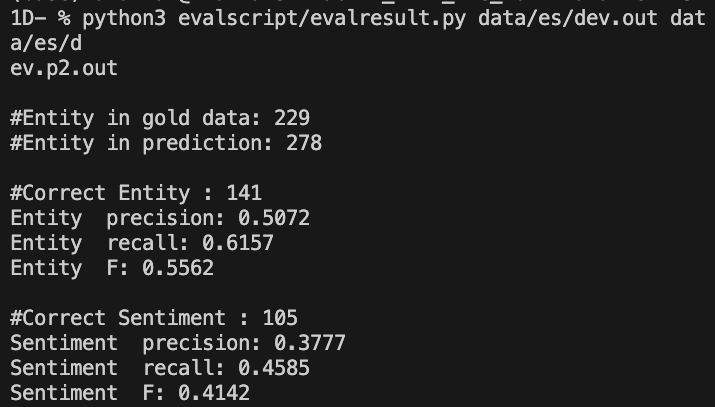
## **Task 2**

Approach:

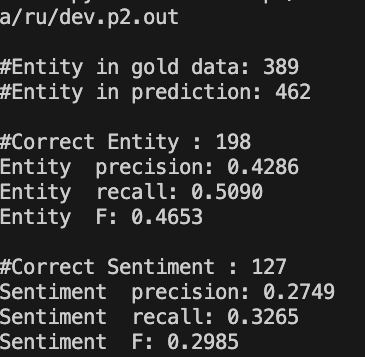
1. **Data Preparation:** The training data provided contains labeled sentences, and the sentiment tags are separated by a tab. Sentences are separated by an empty line. Training data is loaded for the ES (Spanish) dataset. Each sentence is split into words and associated sentiment tags, including special tags "START" and "STOP" to account for transitions.
2. **Transition Parameter Estimation:** Transition parameters are estimated using Maximum Likelihood Estimation (MLE). The count of transitions between states and the count of occurrences of each state are calculated. Transition probabilities are computed based on the counts, considering special cases "q(STOP|yn)" and "q(y1|START)".
3. **Emission Parameter Estimation:** Similar to the previous task, emission parameters are estimated using MLE. Emission probabilities are calculated for words given sentiment tags, considering the "#UNK#" token for handling unseen words.
4. **Viterbi algorithm implementation:** The Viterbi algorithm is implemented to find the most likely sequence of sentiment tags for a given sequence of words. The algorithm uses transition and emission probabilities, as well as special cases for "START" and "STOP" tags. The algorithm calculates probabilities and back pointers using dynamic programming.
5. **Prediction & output:** The implemented Viterbi algorithm predicts sentiment tags for each word in the development data and writes them to an output file ("dev.p2.out"). This file contains the predicted sentiment tags for each word in the input sentences, considering the "START" and "STOP" tags.
6. **Evaluation:** The precision, recall, and F-score of the Viterbi-based sentiment analysis system are evaluated using an evaluation script. The system's output ("dev.p2.out") is compared to the gold-standard sentiment tags provided in the development output file ("dev.out").

Result:

1. ES Dataset:



1. RU dataset:



Conclusion: Using the Viterbi algorithm provides a much better result compared to just emission probabilities. However this can still be improved.

## **Task 3**

Approach:

To find the k-th best output sequences, an extended Viterbi algorithm is implemented. This modification involves maintaining k best paths at each step instead of just one. The dynamic programming approach is adapted to keep track of multiple paths and probabilities.

Initialization: Initialize a list of size k to store sequences and their corresponding probabilities at each step. At the first step, the probabilities are calculated based on the emission parameters.

Recursion: For each subsequent step, the k best paths are calculated based on the probabilities of transitioning to different tags from the previous step. The probabilities are combined with the emission probabilities to find the k best paths for the current step.

Termination: Once the end of the sequence is reached, the k best paths are selected based on their probabilities.

Backtracking: To retrieve the k-th best output sequence, backtrack from the end of the sequence to the beginning, considering the paths and probabilities stored during the recursion.

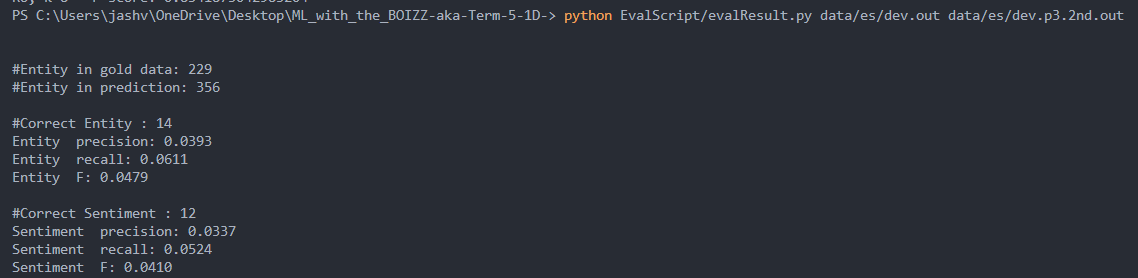
Prediction & Output: The modified algorithm is applied to the development data for both the ES and RU datasets. For each dataset, the algorithm generates outputs for the 2-nd and 8-th best sequences. The outputs are written to "dev.p3.2nd.out" and "dev.p3.8th.out" files.

Evaluation: The precision, recall, and F-score are calculated for the generated outputs using the evaluation script provided. The outputs ("dev.p3.2nd.out" and "dev.p3.8th.out") are compared against the gold-standard sentiment tags from the development output files ("dev.out").

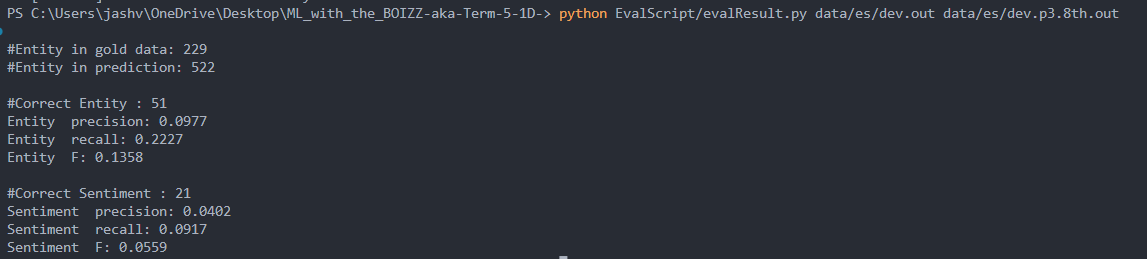
Results:

ES Dataset:

2-nd Best Output:

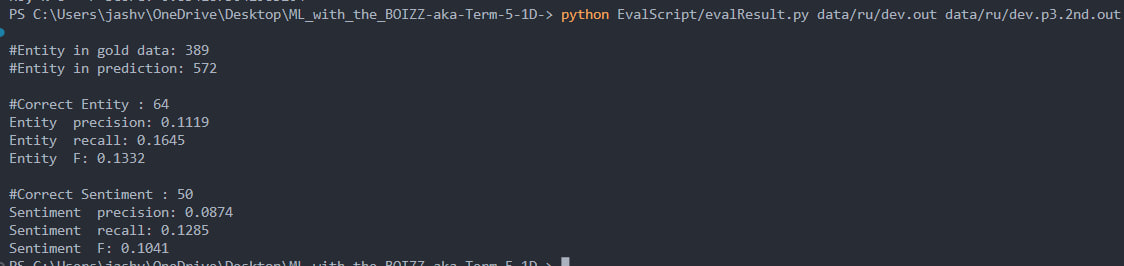


8-th Best Output:

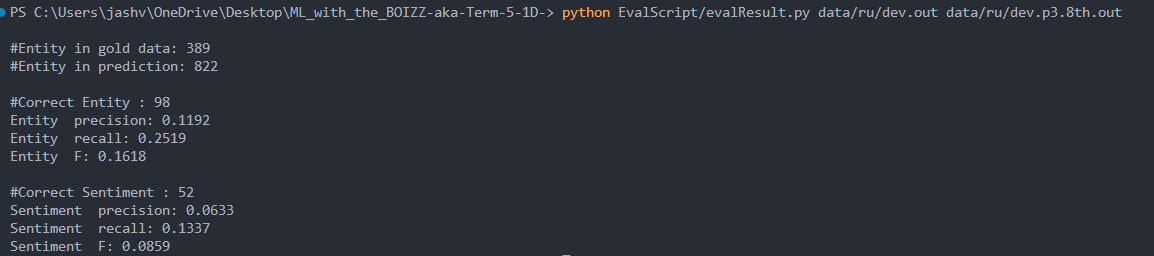


RU Dataset:

2-nd Best Output:



8-th Best Output:



## 

## 

## **Task 4**

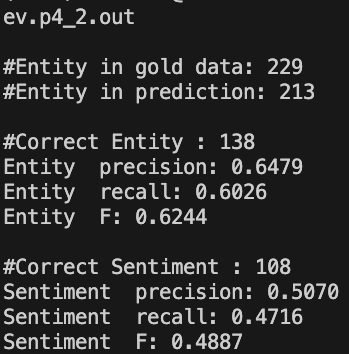
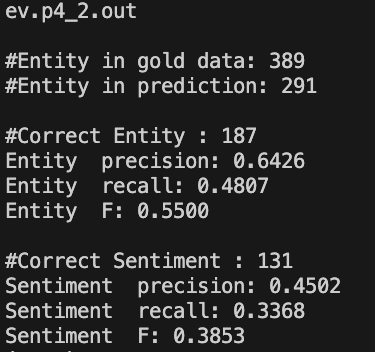
Approach:

In order to attack this problem, we worked together to brainstorm possible ideas to implement. We came up with interesting ideas such as using logistic regression, neighboring n-grams, Brill’s tagger, perceptron, and deep learning concepts such as LSTM. Iterating through these ideas we found that it was difficult to implement them without the help of external libraries or pre-trained models. One idea was to import a dictionary of synonyms and attempt to replace an unknown word with a synonym that has appeared in the training set before. However this would be computationally excessive as we would need to store and iterate through an entire dictionary. One possible edit we made was to replace the values for the unknown token with 1e-10 in the task 2 code. The smaller probability of emission when compared to the calculated unknown token values provided marginal improvements in f-scores and accuracies.

We also decided to go through the perceptron method as was suggested in the project briefing pdf. The following are our results for the same.

Edited task 2 code for #UNK# token (task4\_2\_xx.ipynb)

(ES) (RU)

Conclusion:

setting up the emission probabilities as a very small constant for unknown or not seen words helps improve model performance