SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

50.007 Machine Learning

1D Project Report

Name	Student ID
Parth Kumar	1006257
Ethan Gyan Singh	1006171
Ryner Tan	1005982
Shwetha Iyer	1006308

This report will detail the approaches used to solve the four parts of the Design Project, as well as discuss the results obtained.

Part I

The code in this part involves a very rudimentary sentiment analysis system that produces the tag based on the conditional probability of a specific observation being associated with that tag. Additionally, we implemented a special word token “#UNK#” with the occurrence k=1 in the event of an unknown word in the test data set. For the sentiment analysis, we produce the tag y* that has the highest likelihood of occurrence of all tags given that observation. This sentiment analysis system resulted in the following scores:

For ES:

Entity Precision: 0.1214
Entity Recall: 0.7773
Entity F Score: 0.2100
Sentiment Precision: 0.0622
Sentiment Recall: 0.4236
Sentiment F Score: 0.1145

For RU:

Entity Precision: 0.1465
Entity Recall: 0.6838
Entity F Score: 0.2413
Sentiment Precision: 0.0710
Sentiment Recall: 0.3316
Sentiment F Score: 0.1170

This method of sentiment analysis has very low F scores, as we only consider the likelihood of the observation being associated with the tag, but do not consider the likelihood of transitions between states. By ignoring the state transitions, we are overlooking the context which may contribute to the sentiment of each observation. When developing a model to analyze sentiment of social media texts, it is important to include contextual information as the context would greatly vary in social media and a model trained without considering context may perform well in a specific context, but will not generalize well.

Part II

Firstly, the first part of the code estimates transition parameters using the given formula, while including special cases such as y1 to START, as well as STOP to yn. We then discovered that there is bias towards transitions which are more likely to be seen due to higher probabilities, with the most probable being ‘START’ to ‘O’ with a probability of 0.928.

In the second part, we implemented the Viterbi algorithm to compute the most probable sequence of labels for a given input sentence using estimated transition and emission parameters. We first learned the model parameters from the training data. Then, we applied the Viterbi algorithm on the development set (dev.in) using the learned models and generated the corresponding output (dev.p2.out) for two different datasets.

To address potential numerical underflow issues in the Viterbi algorithm, we employed techniques such as working with logarithmic probabilities and using dynamic programming to efficiently calculate the most probable sequence. By considering both transition and emission probabilities, we could accurately determine the sequence of labels that maximizes the joint probability of the input sentence and the label sequence.

After running the Viterbi algorithm, we evaluated the performance of the system by calculating precision, recall, and F1 scores for each dataset. These metrics provided insights into the effectiveness of the model and its ability to correctly predict labels.

For ES:

Entity Precision: 0.2459
Entity Recall: 0.5852
Entity F Score: 0.3463
Sentiment Precision: 0.1780
Sentiment Recall: 0.4236
Sentiment F Score: 0.2506

For RU:

Entity Precision: 0.3884
Entity Recall: 0.4833
Entity F Score: 0.4307
Sentiment Precision: 0.2665
Sentiment Recall: 0.3316
Sentiment F Score: 0.2955

Part III

The code for this part presents an approach using the Viterbi algorithm, a dynamic programming technique, to find the most likely sequence of labels for a given input sequence of words.

The code follows a comprehensive approach to the problem by utilizing emission and transition probabilities. Emission probabilities capture the likelihood of observing a word given a specific label, while transition probabilities represent the likelihood of transitioning from one label to another. The algorithm computes these probabilities based on the training data and employs them to determine the most probable sequence of labels for each input sentence.

One of the significant features of this approach is the utilization of *k*-best Viterbi decoding. Instead of settling for the single best label sequence, the algorithm explores multiple potential label sequences and retains the top *k* candidates. This flexibility can help in capturing different nuances in the data and provides a range of possible labeling solutions.

The results achieved from the implementation of the Viterbi algorithm on the development data are as follows:

Spanish (ES) Results:

<u>ES/dev.p3.2nd.out:</u>
Precision: 0.1647
Recall: 0.3631
F Score: 0.2266

<u>ES/dev.p3.8th.out:</u>
Precision: 0.1511
Recall: 0.3153
F Score: 0.2043

In the second output file for Spanish (ES/dev.p3.2nd.out), the algorithm achieved a precision of approximately 0.165, indicating that among the predicted entities, around 16.5% were accurate. The recall score of 0.363 suggests that the algorithm captured 36.3% of the actual entities present in the data. The F-score of 0.227, which balances precision and recall, emphasizes that this configuration's performance lies between precision and recall, capturing a moderate number of relevant instances while maintaining a relatively acceptable precision-recall trade-off.

However, in the eighth output file for Spanish (ES/dev.p3.8th.out), the precision slightly decreased to 0.151, while the recall also reduced to 0.315. This resulted in a lower F-score of 0.204, indicating a relatively weaker performance in accurately identifying entities compared to the second output. The challenge in this configuration seems to be maintaining a good balance between precision and recall.

Russian (RU) Results:

RU/dev.p3.2nd.out:
Precision: 0.1780
Recall: 0.3557
F Score: 0.2373

RU/dev.p3.8th.out:
Precision: 0.1545
Recall: 0.2995
F Score: 0.2038

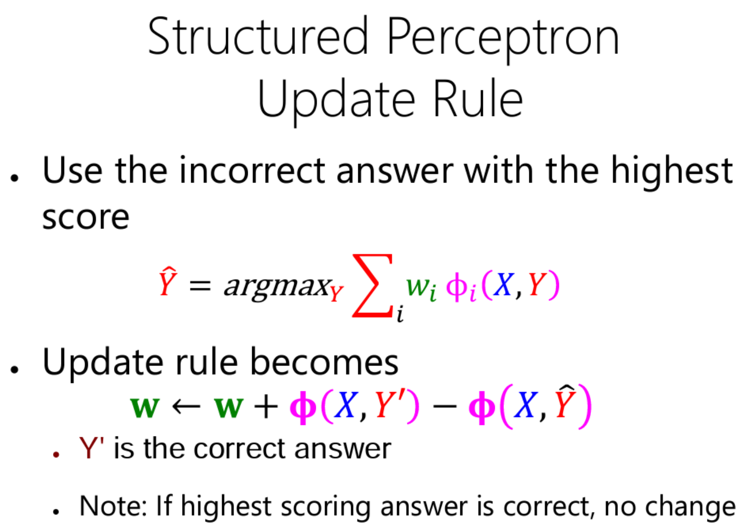
In the second output file for Russian (RU/dev.p3.2nd.out), the algorithm achieved a precision of around 0.178, indicating a higher precision compared to the Spanish results. The recall score of 0.356 suggests that the algorithm captured 35.6% of the actual entities in the data. This results in an F-score of 0.237, which is higher than the Spanish equivalent, indicating a better balance between precision and recall.

However, in the eighth output file for Russian (RU/dev.p3.8th.out), the precision slightly decreased to 0.154, while the recall also reduced to 0.299. This configuration's F-score dropped to 0.204, reflecting a similar trend as observed in the Spanish case. Despite the slightly lower recall, the algorithm maintains a relatively competitive F-score, showing its ability to maintain an acceptable trade-off between precision and recall.

The reported results indicate a trade-off between precision and recall, as evident from the F scores. The variation in F scores across different approaches emphasizes the importance of fine-tuning the parameters and model selection. Furthermore, the *k*-best Viterbi decoding approach contributes to a diverse set of possible labeling solutions, which can be crucial in handling ambiguous cases. The results obtained provide insights into the model's performance and highlight the intricate balance between precision and recall in this task.

Part IV

To refine the handling of emission and transmission probabilities, we have chosen to adopt structured perceptrons. By executing multiple epochs during the training process, we can effectively enhance the modeling of transition and emission probabilities. Each iteration serves as a corrective step, penalizing incorrect predictions and rewarding accurate ones. Since log-likelihood was used, the learning rate was added to the right predictions, and penalized the wrong ones, for both transmissions and emissions.



Structured Perceptron Update Rule

- Use the incorrect answer with the highest score

$$\hat{Y} = argmax_Y \sum_i w_i \, \phi_i(X, Y)$$

- Update rule becomes

$$\mathbf{w} \leftarrow \mathbf{w} + \boldsymbol{\phi}(X, Y') - \boldsymbol{\phi}(X, \hat{Y})$$

  - Y' is the correct answer
  - Note: If highest scoring answer is correct, no change

Source:http://www.cs.umd.edu/class/fall2016/cmsc723/slides/slides_11.pdf

We experimented with a variety of features, including length of sentences, number of capital letters and even 2nd order HMMs, however, we noticed that the data was heavily overfitted due to the addition of these features since our F-scores decreased. After hyperparameter tuning of learning rate and epochs, we were able to optimize values to get the highest F-scores.

Although our F-scores are considerably higher than prior methods, we strongly believe that if we were to pick out the features wisely, our F-score would have been much higher. Given more data, the addition of more features would certainly have helped. Perhaps we should have also modeled the probabilities of transitions of states. Overall, I strongly believe we picked the right algorithm to model the data. Our part 4 scores are:

For ES:

Entity Precision: 0.6009
Entity Recall: 0.5852
Entity F Score: 0.5929
Sentiment Precision: 0.4888
Sentiment Recall: 0.4760
Sentiment F Score: 0.4823

These scores were obtained using a learning rate of 0.1, and training over 20 epochs

For RU:

Entity Precision: 0.4525
Entity Recall: 0.4165
Entity F Score: 0.4337
Sentiment Precision: 0.3296
Sentiment Recall: 0.3033
Sentiment F Score: 0.3159

These scores were obtained using a learning rate of 0.1, and training over 15 epochs