Assignment-2: Chunking

By: Aditya Sharma | 170050043
Yash Jain | 170050055
Debabrata Mandal | 170050073

Performance Comparison of the three models

Method	Overall Accuracy
MEMM	93.25%
CRF	95%
Bi-LSTM	96.17%

References:

- **MEMM**
  1. Viterbi Algorithm for decoding used from POS tagger HMM
  2. https://www.nltk.org/_modules/nltk/classify/maxent.html
  3. https://www.deep-teaching.org/notebooks/sequence-learning/exercise-memm
  4. https://github.com/yh1008/MEMM

- **CRF**
  a. Shallow parsing Using Conditional Random Fields (by Fei Sha and Fernando Pereira)

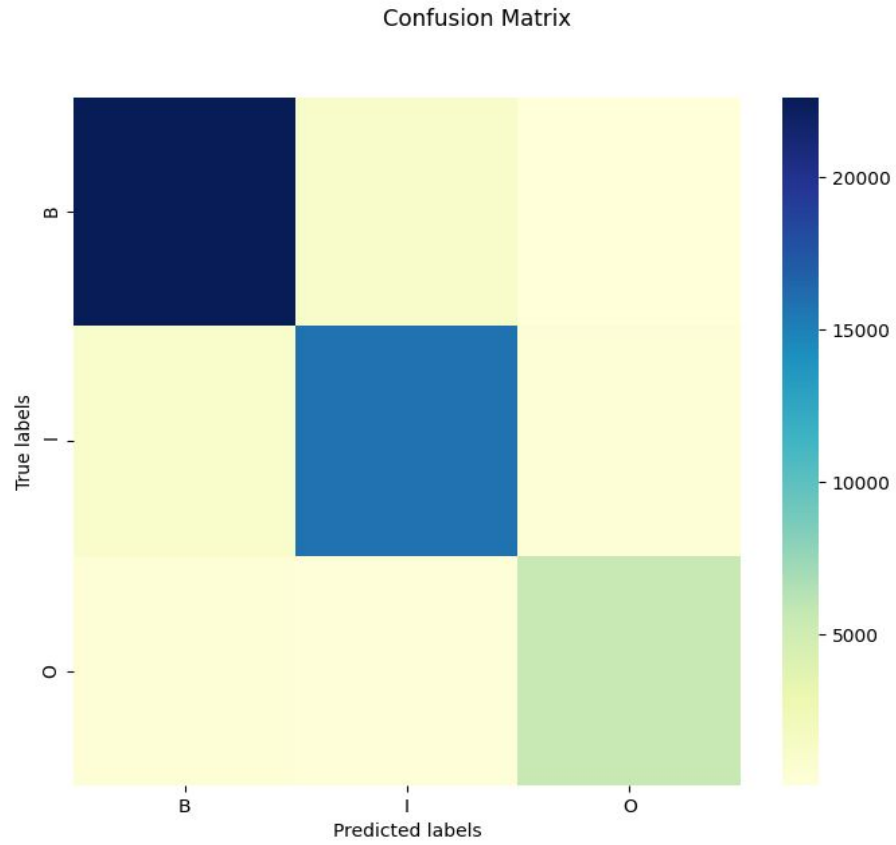
- **Bi-LSTM**
  1. www.pytorch.org

MEMM

Accuracy:

	Precision	Recall	F-score
Overall	93.25%	93.25%	0.9325
B Tag	94.00%	94.86%	0.9443
I Tag	92.27%	91.75%	0.9201
O Tag	93.40%	91.26%	0.9217

Confusion Matrix : Plotted for the Test data



Features Used:

✓ Prev_to_prev_POS_tag ✓ Prev_POS_tag ✓ POS_tag ✓ Prev_to_Prev_Chunk ✓ Prev_Chunk ✓ Cur_Stem ✓ Prev_Stem ✓ Prev_to_Prev_Stem ✓ Suffixes

Strengths of using MEMM for Shallow Parsing:

1. Decoding is very efficient and fast -> Complexity: O(9L). Because there are 3 tags and we are looking at the previous 2 tags, complexity becomes O(9L).
2. Faster training as compared to SVM and Bi-LSTM. We have used the Maxent Classifier of nltk.
3. Features can be customised for the data and depending on tags and properties

Weakness witnessed for MEMM for Shallow Parsing:

1. MEMM assumes Markov condition holds, which may not always be true. Therefore this is limited by first order markov property.
2. The effect of far off terms on is not considered while deciding the tags for the given word.
3. It is sometimes difficult to tag words where the uncommon tag is shadowed by the dominant tag in Emission probabilities.(example given below)

Learnings:

There was a lot to be learnt from MEMM. Firstly, all the theoretical aspects like probability, Entropy Maximization and Markov principle.Then we also learnt the implementation and the aspects of Viterbi algorithm.

Then there were other aspects to learn like on what conditions there can be error and why that occurs(examples mentioned below). Aspects of Machine Learning like accuracy, recall and F-score were also understood during the assignment.

We also got exposed to various new libraries like NLTK which has a very good database of corpuses. We also explored various classifiers like Maxent Classifier and some insight into the working. We also explored the dependence of accuracy when we increase/decrease the training iterations on maxent classifier.

Error Analysis:

- It is unnecessary to use Viterbi and Maxent classifier for O tags with punctuations like [, . “ ‘]. Therefore we add a rule to directly tag these as O.

Accuracy jumps to 93.71%

	Precision	Recall	F-score
O tag	97.94%	90.06%	0.9383

- Misclassification of Conjunctions:

Conjunctions take both O and I tags depending on context:
Many streets and_I sidewalks buckled, and_O subterranean water mains and_I service connections ruptured.

- VBZ Misclassification of B and I tags following VBZ:

Words which have tags VBZ tend to be the beginners of Verb Phrase like for instance:
He has_VBZ_B given_VBN_I the exam.

But in some cases it also behaves like a verb indicating possession:
He has_VBZ_B options_NNS_I to choose from.

Our classifier tends to this as case when, especially when VERBS are used as a NOUN. For example:
H. Anthony Ittleson was elected a director of this company, which primarily has_VBZ_B interests_NNS_B in radio …...

- Verb-Noun Confusion:

Pre-refunded bonds are called at their earliest call date with the escrowed proceeds of another bond issue.
Here “escrowed” is VBN_I and “proceeds” is NNS_I, but parser assigns B tag to “proceeds” thinking it as a VERB

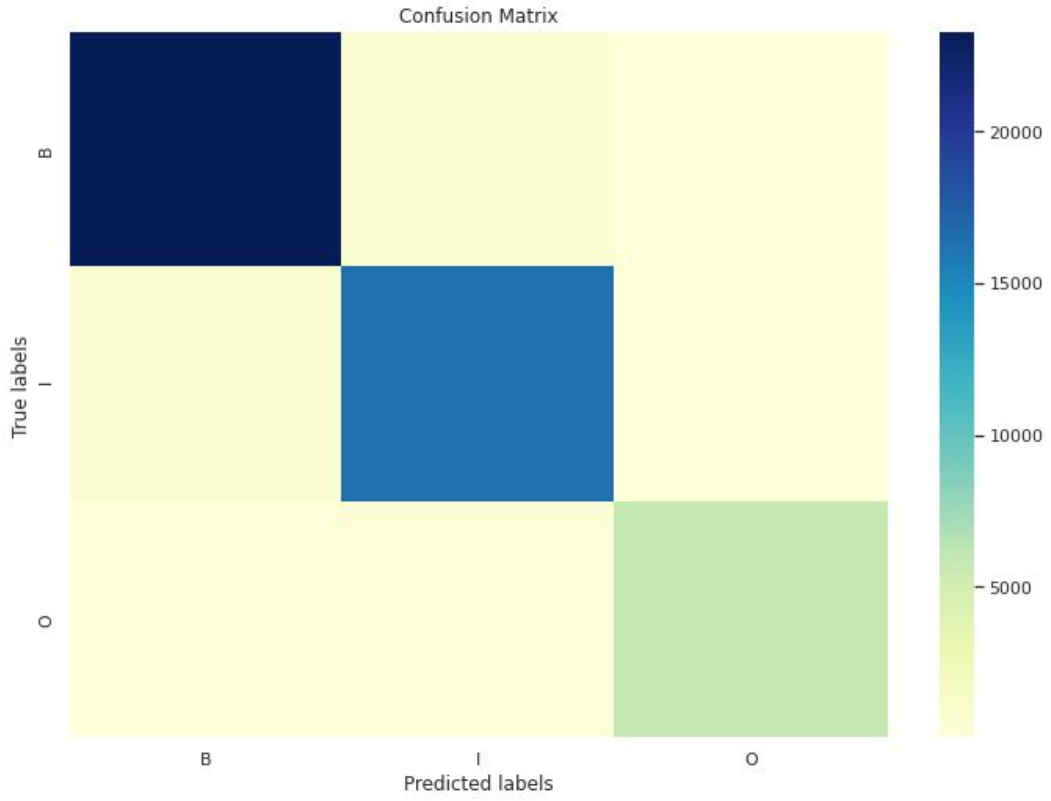
Bi-LSTM

Performance:

The train dataset was split into train and validation sets in 4:1 ratio. The complete test dataset was used in assessing the performance of the Bi-LSTM model. The model was trained for 9 epochs on Adam optimizer with a learning rate of 1e-5.

	Precision	Recall	F-score
Overall	96%	96%	0.96
B Tag	96%	98%	0.97
I Tag	96%	95%	0.95
O Tag	97%	95%	0.96

Confusion Matrix : Plotted for the Test data



Model Architecture:

```
BiLSTMNet(
  (embeddings1): Embedding(8119, 1024)
  (embeddings2): Embedding(44, 1024)
  (bilstm): LSTM(2048, 1024, batch_first=True, bidirectional=True)
  (fc): Linear(in_features=2048, out_features=3, bias=True)
  (dropout): Dropout(p=0.3, inplace=False)
  (softmax): Softmax(dim=2)
)
```

Features Used:

As evinced by the model architecture, we didn't employ features in our Bi-LSTM model to let it understand the context in an unsupervised fashion. Note that embedding 1 denotes the "word token" and embedding 2 denote its corresponding "pos" tag.

Strengths of Bi-LSTM model in chunking:

- No feature engineering is required
- Model learns appropriate embeddings according to the task
- Method is able to utilise the context of far-off words from both sides of the sentence giving better chunking results

Weakness witnessed in chunking:

- Slowest in training compared to other methods
- Does not have a good explainability of the reason behind performance compared to other methods
- Hidden dimensions of Bi-LSTM: Tuning this hyperparameter showed us the fragility of our system with respect to data available. For eg. when kept to 512: the average accuracy was not more than 80%

Error Analysis and learnings:

- To+infinitive: Like the POS system, chunking also suffers from this ambiguity problem and almost never predicts 'to' and '<verb>' in the same chunk.

  Sent: 'People tend to be mos' oov to brands that have oov oov , such as oov and oov .'

  POS: 'NNS VBP TO VB RBS JJ TO NNS WDT VBP JJ NNS , JJ IN NNS CC NN .'

Prediction: 'B B B I B I B B B B B I O B I B I B O’

Target: 'B B I I B I B B B B B I O B I B I I O’

- Input embeddings: We tried to give single input embeddings to the BiLSTM model by having “word pos” as a single token input. This resulted in a total token vocabulary size to be around 8700 approximately. However, even after tuning the hyperparameters we could not surpass 85% accuracy from the model.

  Instead, we separated the “word” and “pos” as individual tokens giving a total vocabulary size of approximately 8100 and 50 respectively. Despite such marginal decrease in vocabulary size the performance of this representation shot up to 96% easily.

  We believe that merging “pos” with the “word” masked its effect which was better highlighted when both of them learnt individual representations.

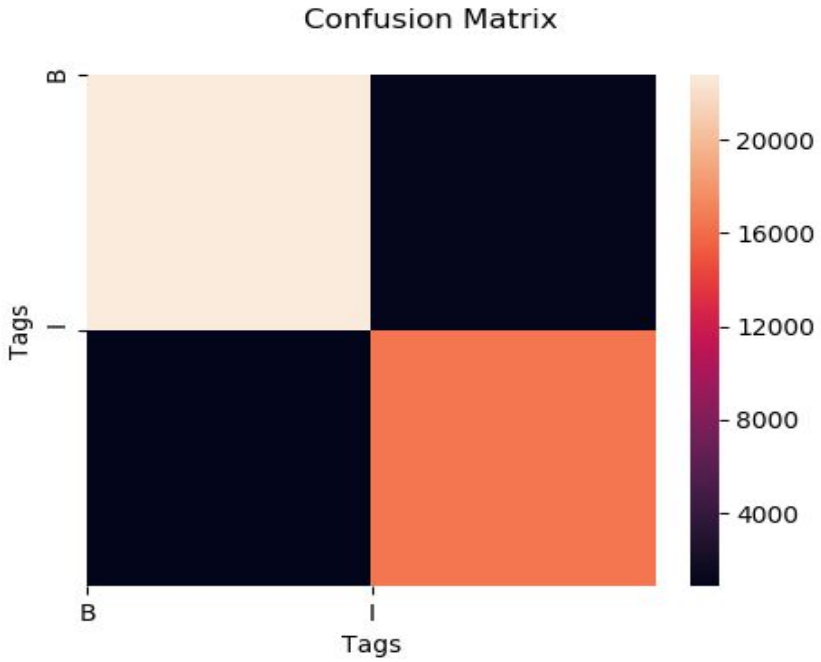
CRF

Performance:

Trained on the whole dataset. Approximate time required to train ~ 1 min. (O tags removed before training).

Tag-score	Precision	Recall	F1-score
Overall	95.27	95.26	95.27
B	96.3	95.5	95.9
I	93.9	94.9	94.4

Confusion Matrix : Calculated on Test Data.



Training Algorithm:

We used the training algorithm used in the paper “Shallow parsing with conditional random fields” by Fei Sha and Fernando Pereira. It involves using an optimization algorithm named Limited memory BFGS which is already available pre-implemented in the sklearn library for ready use. Hence once the feature vectors were ready we invoked the training algorithm from the CRFsuite

Features Used:

W_i , W_{i-1} , W_{i-2}
Word features like is_upper, is_title, is_digit
Word suffixes i.e. last 3 and last 2 letters
POS_i , POS_{i-1} , POS_{i-2}
Is_first_word, is_second_word

Error Analysis

A few significant observations are :

1. Wrong classification regarding POS tags NN and VB - Many instances where chunk tags are not correctly classified relate to the case when there is a NNx tag preceding a VBx tag. This could result from the fact that there is a bias in the dataset chosen to chunk verbs with previously occurring noun words.
   (Below, the first tag corresponds to the previously occurring word and the second to the second word the first chunk tag is the correct chunk tag of current word and second the one it was misclassified to. The number represents the number of such instances.)
   Example : NNP_VBN {'B_I': 13, 'I_B': 0}
   NN_VBG {'B_I': 12, 'I_B': 2}
   NNP_VBG {'B_I': 10, 'I_B': 0}
   NNS_VBN {'B_I': 3, 'I_B': 0}
   NN_VBN {'B_I': 23, 'I_B': 1}
   NNP_VB {'B_I': 5, 'I_B': 1}
   NN_VB {'B_I': 24, 'I_B': 0}
   Hence clearly this is although not a determining factor for the accuracy to go down, but a possible cause.

2. Wrong classification regarding POS tags TO and NN - Similar to the above observation, tags seem to be mis-classified heavily when the previous POS tag is TO and the next is NNx. The significant one is the chunk tag B getting misclassified to I.
   Example : TO_NNP {'B_I': 18, 'I_B': 0}
   TO_NN {'B_I': 20, 'I_B': 0}
   TO_NNS {'B_I': 20, 'I_B': 0}