CRF with Viterbi decoding for Chunking

Report submitted in fulfillment of the requirements
Of the course

Natural Language Processing - CSE443

by

Shivansh Saini, Anuraj Thakur, Dipesh Kumar
17075054, 17074003, 17075021

Under the guidance of
Dr. A K Singh

Department of Computer Science and Engineering
INDIAN INSTITUTE OF TECHNOLOGY (BHU) VARANASI
Varanasi 221005, India
May 2019

Certificate

*This is to certify that the work contained in this report entitled "**CRF with Viterbi decoding for Chunking**" being submitted by **Shivansh Saini, Anuraj Thakur, Dipesh Kumar** (**Roll No. 17075054, 17074003, 17075021**), carried out in the Department of Computer Science and Engineering, Indian Institute of Technology (BHU) Varanasi, is a bona fide work of our supervision.*

Place: IIT (BHU) Varanasi
Date: May 3, 2019

Dr. A K Singh
Associate Professor
Department of Computer Science and Engineering,
Indian Institute of Technology (BHU) Varanasi,
Varanasi, INDIA 221005.

Abstract

Chunking or shallow parsing is a process of meaningful extraction of short phrases from the sentence. Here we have used a CRF model with Viterbi decoding to perform chunking on a Hindi dataset and performed error analysis at different levels with the help of an efficient pre-compiled library.

Contents

Chapter 1

Introduction

1.1 POS Tagging and Chunking

Part of Speech tagging is the task of assigning grammatical classes to words in a natural language sentence. It's important because subsequent processing stages (such as parsing) become easier if the word class for a word is available. There is a standard set of POS Tags like Noun(N), Verb(V), Adjective(ADJ), etc. Most POS are divided into sub-classes. For example, the sentence *The little yellow dog barked at the cat.* can be POS tagged as follows :

The/**DT** little/**JJ** yellow/**JJ** dog/**NN** barked/**VBN** at/**IN** the/**DT** cat/**NN** ./**.**

Chunking or shallow parsing is a process of meaningful extraction of short phrases from the sentence. Chunks are made up of words and kind of words are defined by the POS tag of the words. Chunking works on top of POS tagging, it uses pos-tags as input and provides chunks as output. Similar to POS tags, there are a standard set of Chunk tags like Noun Phrase(NP), Verb Phrase (VP), etc. For example, the sentence *He reckons the current account deficit will narrow to only # 1.8 billion in September* can be chunked as shown :

[**NP** He] [**VP** reckons] [**NP** the current account deficit] [**VP** will narrow] [**PP** to] [**NP** only # 1.8 billion] [**PP** in [**NP** September]].

Both the steps are an intermediate step to full parsing. These are used to perform tasks like Named Entity Recognition (NER), Sentiment Analysis, etc.

1.2 Conditional Random Fields (CRF)

Machine Learning models have two common categorizations, Generative and Discriminative. Conditional Random Fields are a type of Discriminative classifier, and as such, they model the decision boundary between the different classes. Generative models, on the other hand, model how the data was generated, which after having learnt, can be used to make classifications.

In CRFs, our input data is sequential, and we have to take previous context into account when making predictions on a data point. To model this behavior, we use Feature Functions, which have multiple input values, like - current word, pos tag of current word, position of word, etc. The purpose of the feature function is to express some kind of characteristic of the sequence that the data point represents.

Each feature function is based on the label of the previous word and the current word, and is either a 0 or a 1. To build the conditional field, we next assign each feature function a set of weights (parameter values), which the algorithm is going to learn.

1.3 Viterbi Decoding

Viterbi decoding is a dynamic programming approach that is used for the maximum likely-hood decoding of a sequence of states. In this method, we start with some

initial states and the calculate the best possible state sequence up to some length. After that we use these values to calculate the next best possible state.

Since it's using the information stored up to a certain level thus this approach becomes a dynamic programming based algorithm.

Chapter 2

Related Work

Most previous work used two main machine-learning approaches to sequence labeling. The first approach relies on k-order generative probabilistic models of paired input sequences and label sequences, for instance hidden Markov models (HMMs) (Freitag and McCallum, 2000). The second approach views the sequence labeling problem as a sequence of classification problems, one for each of the labels in the sequence. Previous work on Chunking for Hindi dataset had been done using CRF model (Himanshu, 2016).

Chapter 3

Problem Description

Chunking or shallow parsing is a process of meaningful extraction of short phrases from the sentence.Chunks are made up of words and kind of words are defined by the POS tag of the words. In this project, we had to explore the efficient pre-compiled library of CRF with Viterbi decoding to solve the Chunking problem for Hindi and perform error analysis at three levels : Word-level, POS tag level and Sentence level, and find the common pattern among the erroneous prediction.

Deliverable

Input	Hindi Tagged Data (Hindi Tree Bank)
Output	Chunked sentences and comparative results with different feature sets.
Metrics	Precision, Recall, F-Score, Accuracy
Error analysis	The error analysis should be at three levels: Word-level,POS tag level, Sentence level, and the common pattern among erroneous prediction.

3.1 Dataset

For the hindi tagged dataset, we have used the Hindi Treebank Data provided at IL-TDIL. Hindi Tree Bank data is in Shakti Standard Format (SSF). It is a

common representation for data. It allows information in a sentence to be represented in the form of one or more trees together with a set of attribute-value pairs with nodes of the trees. It has been annotated for surface form, POS tags, lemma and morpholgical analysis.

Statistics :

1. Size of corpus : 50.8 MB
2. No of sentences : 21086
3. Dataset Source : Hindi Treebank Data, IIITH

Chapter 4

Methodology

4.1 Preprocessing

The Hindi Treebank Data, contains the data stored in the Shakti Standard Format (SSF). Although a very common data format for the representation of the morphological analyzed data.

```
<Sentence id='26'>
1»  ((» NP» <fs name='NP' drel='k7:VGF'>
1.1» मोरचे»NN» <fs af='मोरचा,n,m,sg,3,o,0,0' name='मोरचे' posn='10'>
1.2» पर»  PSP»<fs af='पर,psp,,,,,,' name='पर' posn='20'>
»   ))
2»  ((» NP» <fs name='NP2' drel='k1:VGF'>
2.1» पानी»NN» <fs af='पानी,n,m,sg,3,d,0,0' name='पानी' posn='30'>
»   ))
3»  ((» NP» <fs name='NP3' drel='k1s:VGF'>
3.1» बड़ी» JJ» <fs af='बडा,adj,f,sg,,d,,' name='बड़ी' posn='40'>
3.2» समस्या»   NN» <fs af='समस्या,n,f,sg,3,d,0,0' name='समस्या' posn='50'>
»   ))
4»  ((» VGF»<fs name='VGF' stype='declarative' voicetype='active'>
4.1» थी»  VM» <fs af='था,v,f,sg,any,,था,WA' name='थी' posn='60'>
»   ))
5»  ((» BLK»<fs name='BLK' drel='rsym:VGF'>
5.1»।»   SYM»<fs af='।,punc,,,,,,' name='।' posn='70'>
»   ))
</Sentence>
```

Figure 4.1 Data in SSF format

However, the data in this form is not that useful for our task. So, we pre-process the data in the following format.

```
[
  {
    'word': 'बौद्ध',
    'word_category': 'NNP',
    'chunk_category': 'B-NP',
    'morph_features': ['बौद्ध', 'n', 'm', 'sg', '3', 'd', '0', "0' name='बौद्ध' posn='30"]
  },
  {
    'word': 'तीर्थ',
    'word_category': 'NN',
    'chunk_category': 'I-NP',
    'morph_features': ['तीर्थ', 'n', 'm', 'sg', '3', 'd', '0', "0' name='तीर्थ' posn='20"]
  },
  {
    'word': 'कुशीनगर',
    'word_category': 'NNP',
    'chunk_category': 'B-NP',
    'morph_features': ['कुशीनगर', 'n', 'm', 'sg', '3', 'd', '0', "0' name='कुशीनगर' posn='30"]
  }
]
```


Figure 4.2 Data after processing

In this format, each sentence is represented by a list of dictionaries. Each dictionary corresponds to each word in the sentence. Each dictionary consists of data : word, word category, chunk category and morph features (includes number, gender, case, root word, category, etc.)

Moreover, we have divided the chunk tags in two parts: B (marks the beginning of the chunk) and I (marks the internal word of a chunk), called the chunk boundary markers.

4.2 Feature selection

For the purpose of analyzing the training data in different feature sets, we have used the approach of bit-masking for selection of various feature for training the model. We pass a variable *MODE* to the feature extraction part of the code. On the value of the parameter *MODE*, the extractor decides which features are to be included.

As depicted in Table 4.1, each of *MODE* bits corresponds to a morphological feature, if turned on.

Thus, we can select a set of the above features by keeping the respective bits **ON**. Besides the above features, following features are always extracted:

MODE & 1	Gender
MODE & 2	Number
MODE & 4	Person
MODE & 8	Case
MODE & 16	Root word & (RootWord == Word)

Table 4.1 Relation of MODE bits and Morphologial features

- Word
- POS Tag of the word
- Bigram features:
  - POS Tag of previous word
  - POS Tag of next word

Thus, in order to analyze the data on different feature sets, we have made a list of modes, the data is trained on the modes in the list and result is analyzed and compared against the given modes.

Following modes are used in our approach:

MODE = 0	Word, POS tag & bigram features
MODE = 1	Word, POS tag, bigram features & Gender
MODE = 2	Word, POS tag, bigram features & Number
MODE = 3	Word, POS tag, bigram features, Gender & Number
MODE = 8	Word, POS tag, bigram features & Case
MODE = 11	Word, POS tag, bigram features, Gender, Number & Case
MODE = 16	Word, POS tag, bigram features, Root Word & (RootWord == Word)
MODE = 24	Word, POS tag, bigram features, Case, Root Word & (RootWord == Word)

4.3 Training Model

For the model we have used the **sklearn-crfsuite** library. We use the CRF model of the library for training on our dataset and predicting the results. For training of the model, we used L-BFGS optimization algorithm, a popular algorithm used for parameter estimation in machine learning.

4.3. Training Model

For hyperparameters optimization, the L1 and L2 regularization parameters (supported in L-BFGS) are selected using randomized search and 3-fold cross-validation. The best estimator is obtained after running 50 iterations (a total of 50 * 3 = 150 fits) by checking all of the parameter space.

Hence we used following hyperparameters for our CRF model:

- c1 (coefficient of L1 regularization) = 0.2096570893088954
- c2 (coefficient of L2 regularization) = 0.038587807344039826
- max iterations for optimization = 50
- generates transition features that associate all of possible label pairs

Chapter 5

Results and Analysis

5.1 Result Comparison

F1-score metric is evaluated of our CRF models with different feature sets, as listed in Table 5.1.

MODE	**F1-SCORE**
MODE = 0	0.9743319462389266
MODE = 1	0.974209532811804
MODE = 2	0.9746951056358707
MODE = 3	0.9744745938032985
MODE = 8	0.975352854818989
MODE = 11	0.9754029270950031
MODE = 16	0.9743288423863753
MODE = 24	0.9752093597423765

Table 5.1 F1 score of various modes

https://www.overleaf.com/project/5de65d8e9db5f300010c8687

5.2 Error Analysis

5.2.1 Data-level analysis

This is the common type of error analysis done on any ML model. Model errors are looked on the scale of all of the data. Learning curves, as shown in Fig. 5.1, are a

useful way to understand if our model is over-fitting/under-fitting with respect to the training data.

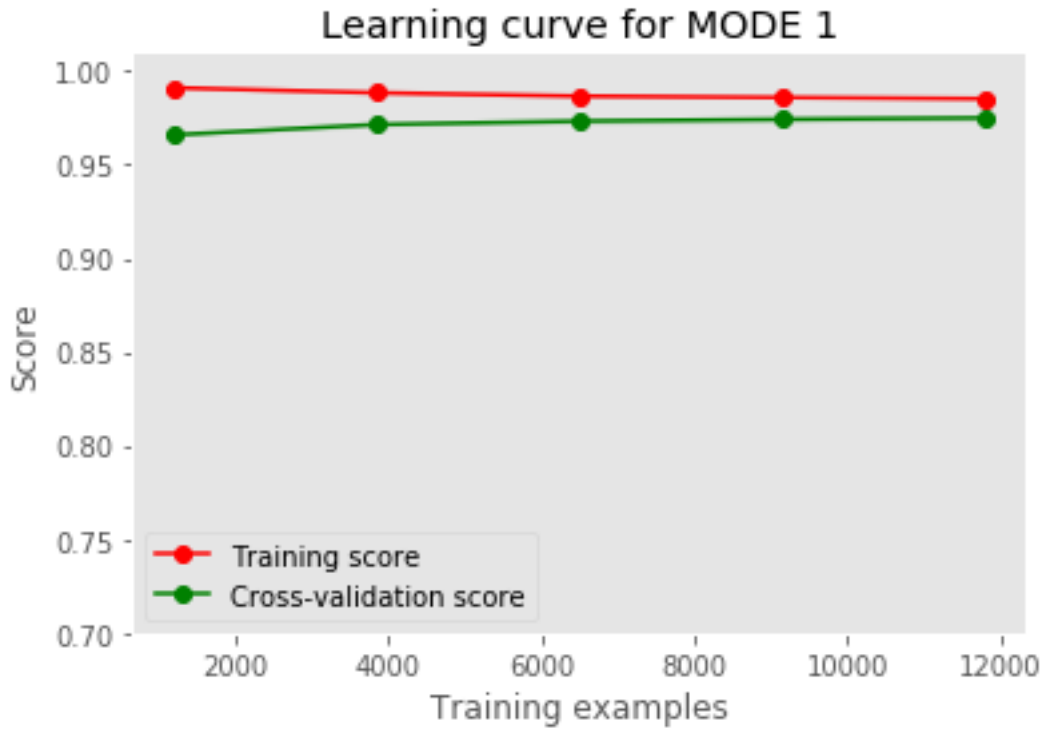

Figure 5.1 Learning curve (similar in all of possible feature sets)

5.2.2 Tag-level analysis

At this level of error analysis, metrics (precision, recall and F1-score) are evaluated and compared against the various feature sets, with respect to classification of chunk classes. The problem is reduced down to binary classification of a particular label for the input word, like a one-vs-all classification. The classes are the chunk tags (combination of chunk boundary markers (B/I) and chunk label markers) for this analysis. Comparison against the given 8 modes is done for F1-score (Fig. 5.2), Precision (Fig. 5.3) and Recall (Fig. 5.4).

Now here we will explain certain tredns in the above graph / plots above. In the above graphs it can be seen that, the performance of the tags FRAGP and NEGP

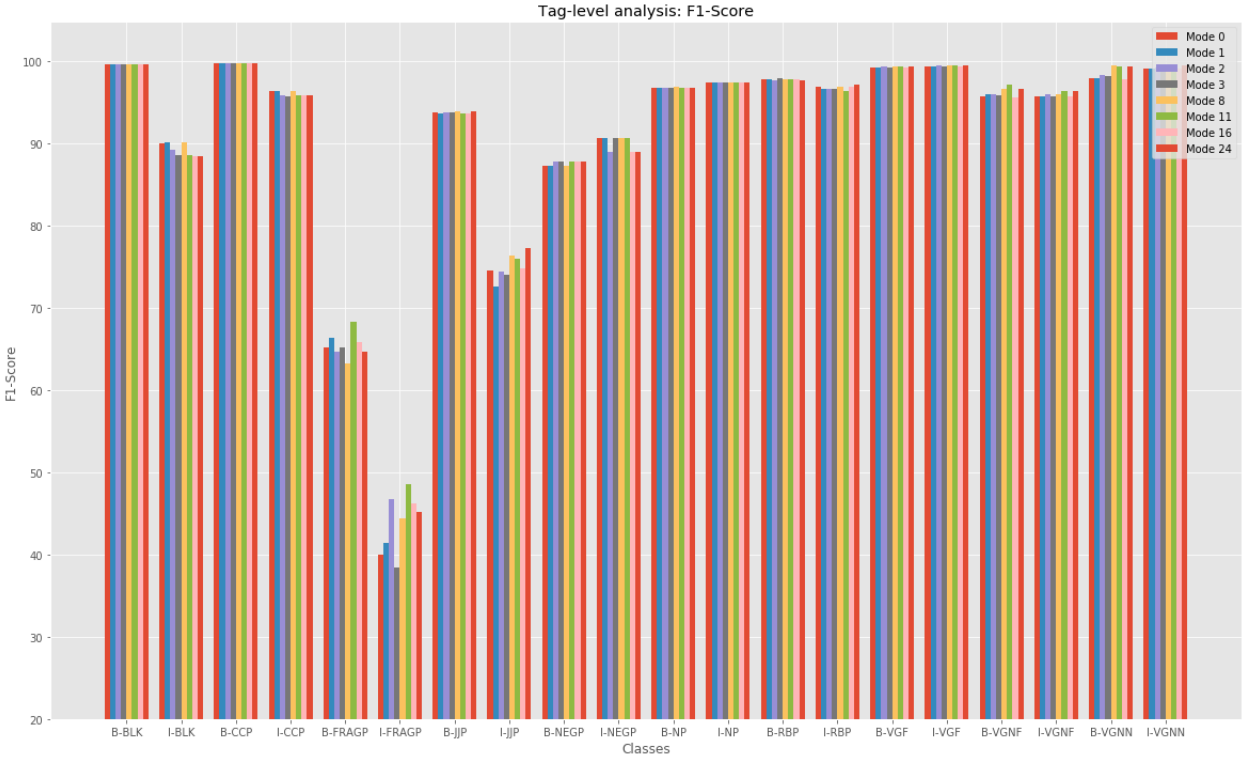

Figure 5.2 F1-Score comparison for 8 modes

Figure 5.3 Precision comparison for 8 modes

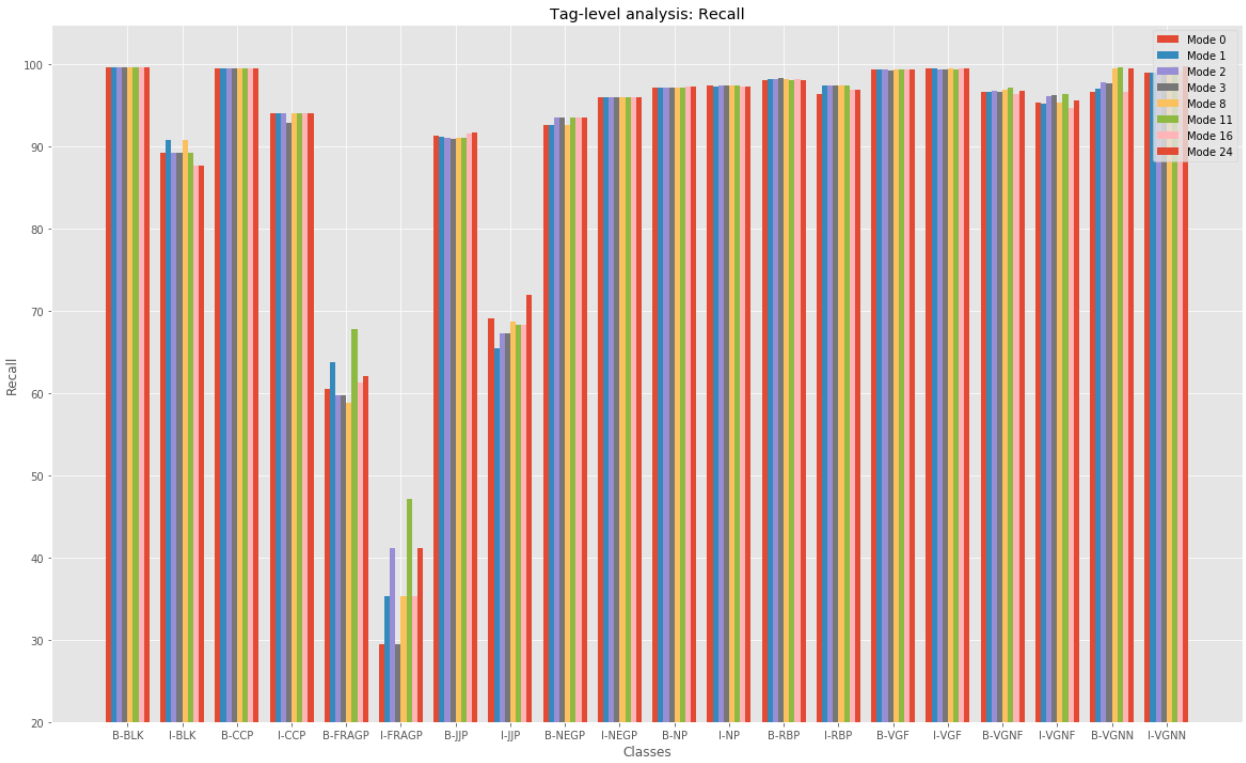

Figure 5.4 Recall comparison for 8 modes

are very staggering.

The main reason for this error is that the probability of predicting a sequence to be NP or NP + FRAGP becomes quite similar in many cases and thus, it becomes hard to get the correct chunk tag.

Similarly is the case with NEGP, which in general combines with verb groups to form VGNF however due to presence of NP in between it is not possible and is often grouped with others.

```
('NP', 12), ('NP', 13), ('FRAGP', 16), ('NP', 17), ('CCP', 18),
('NP', 12), ('NP', 13), ('NP', 17), ('CCP', 18), ('NP', 19), ('
```

Figure 5.5 Example for NP , NP + FRAGP error

```
('RBP', 9), ('NEGP', 10), ('NP', 11), ('VGF', 12), ('BLK', 13)]
('RBP', 9), ('NP', 11), ('VGF', 12), ('BLK', 13)]
```

Figure 5.6 Example for NEGP case

5.2.3 Word-level analysis

In this level of analysis, we understand how our model behaves for various words. The problem is reduced down to multi-classification of a particular word for the given classes.

Let's consider an example case where a negative particle occurs around a verb. It is to be grouped within verb group.

. ((binA_NEG bole_VM))_VGNF kAma ((nahIM_NEG calatA_VM))_VGF

"without" "saying" "work" "not" "happen"

binA kucha **bole** kAma **nahIM calatA**

"without" "something" "saying" "work" "not" "happen"

In the above sentence, the noun "kucha" is coming between the negative "binA" and verb "bole". Here, it is not possible to group the negative and the verb as one chunk. At the same time, "binA" cannot be grouped within an NP chunk, as functionally, it is negating the verb and not the noun. To handle such cases an additional NEGP chunk is introduced. If a negative occurs away from the verb chunk, the negative will be chunked by itself and chunk will be tagged as NEGP. However, our model treated "binA kucha" as an NP Chunk, because it couldn't train itself on any provided feature which could determine this NEGP case.

Another example of ambiguity is the word, "khaana", which appears in both Noun Phrase (NP) and Verb Phrase (VGF/VGNN).

Accuracy (correct predictions / total predictions) is computed for these kinds of words and compared against various feature sets. The analysis is depicted in Fig. 5.7.

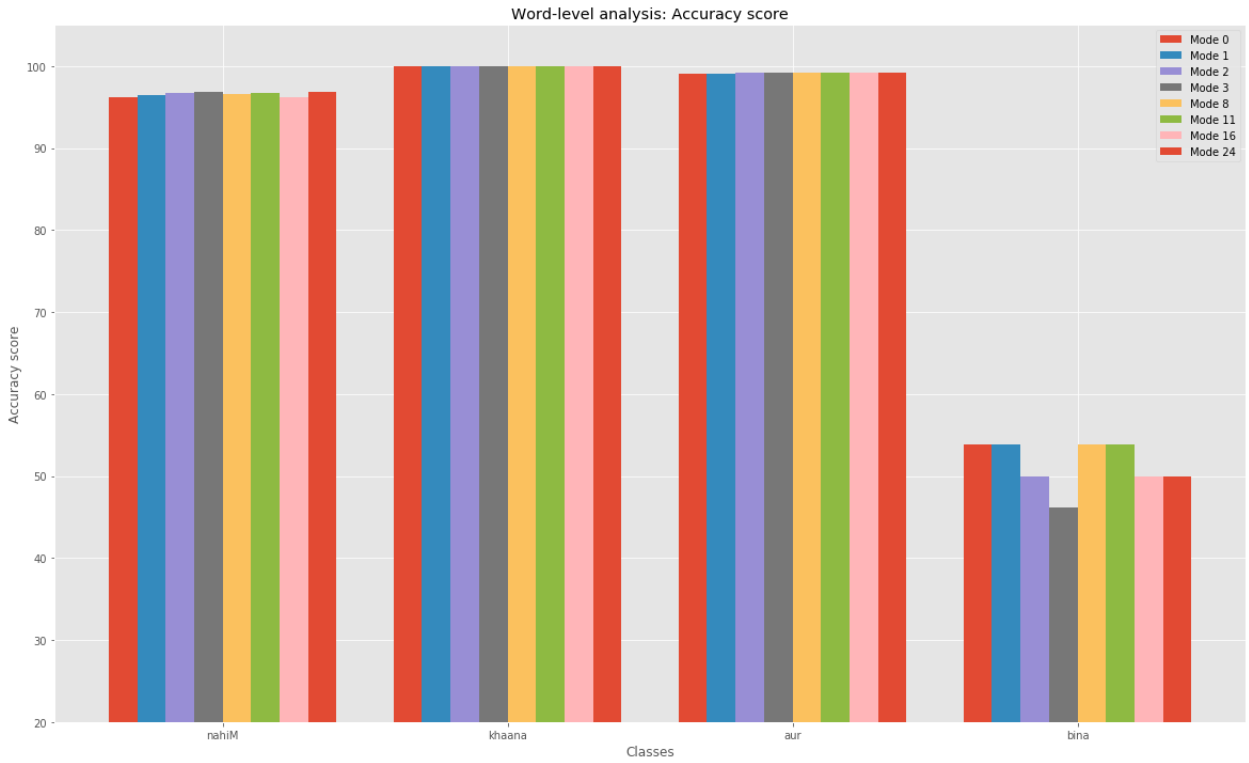

Figure 5.7 Accuracy comparison for 8 modes

5.2.4 Sentence-level analysis

Sentence level analysis is done on two measures: correct sequence of chunk tag prediction and correct boundary prediction of the chunk. At first, we plot heat maps for different modes, where each heat map depicts the accuracy of the prediction of the tag sequences. The heat maps are as shown in the Figures 5.8 - 5.15.

Here, we have shown the values of tag pair sequence prediction to correct tag pair sequence, multiplied by a factor of 10 to magnify the difference between the colours, in order to make the results distinguishable.

As we can see, when we take only the basic template the results deviate more from the ideal value $\tilde{1}0$, however when we take other modes into consideration we can see that more examples are moving towards the ideal behaviour. Here, as we see when we use the *case* bit the result for several pairs of tags are better.

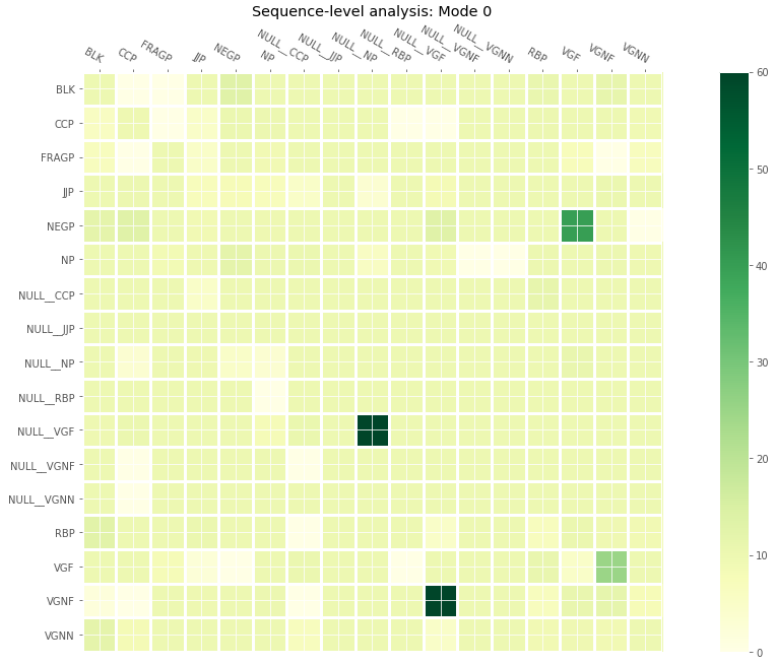

Figure 5.8 **Mode 0** - Predicted / Correct Distribution

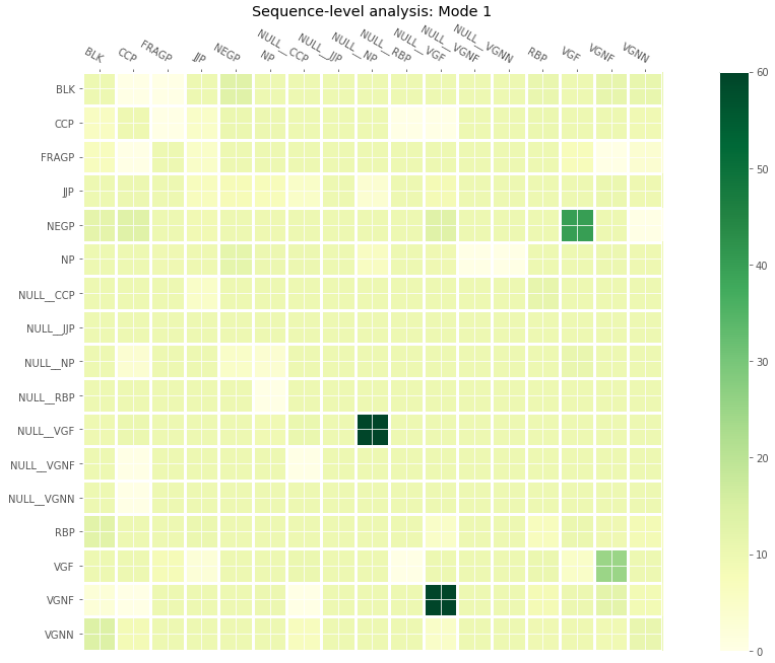

Figure 5.9 **Mode 1** - Predicted / Correct Distribution

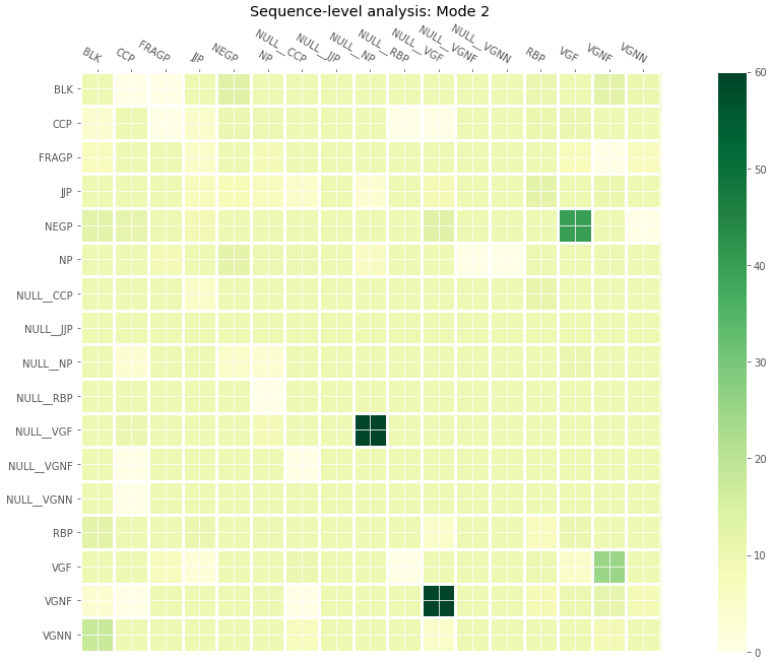

Figure 5.10 Mode 2 - Predicted / Correct Distribution

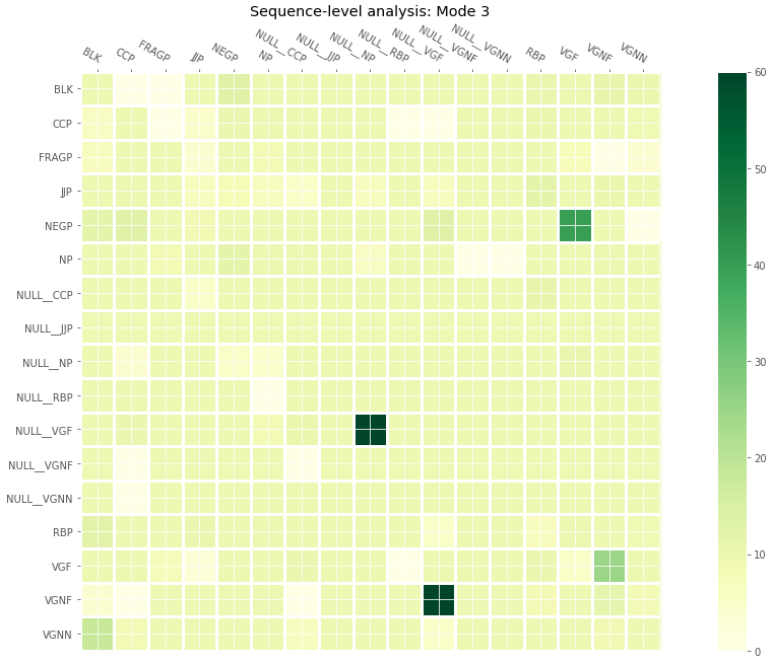

Figure 5.11 Mode 3 - Predicted / Correct Distribution

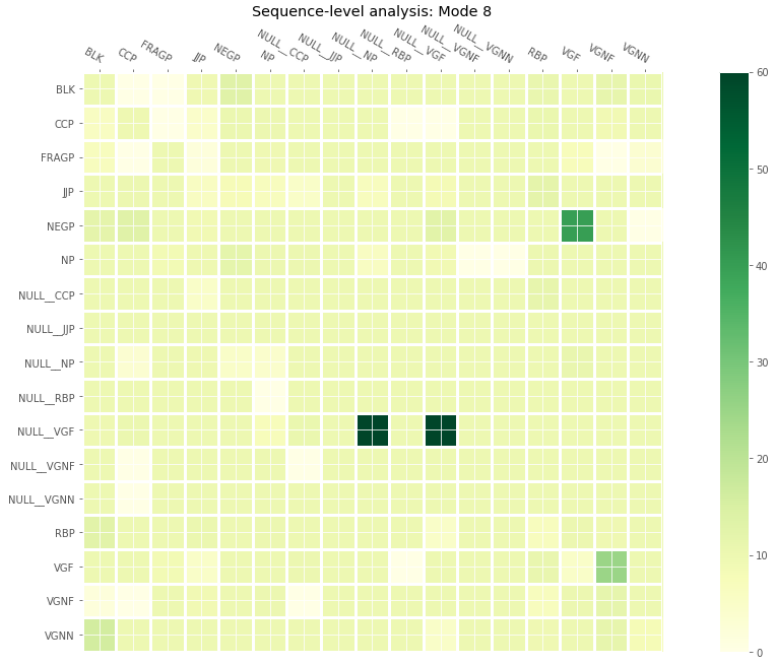

Figure 5.12 **Mode 8** - Predicted / Correct Distribution

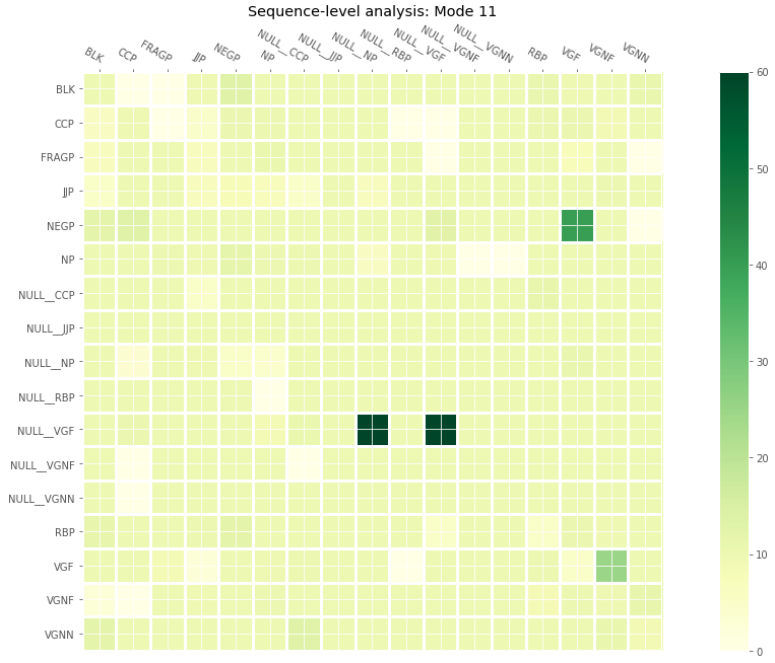

Figure 5.13 **Mode 11** - Predicted / Correct Distribution

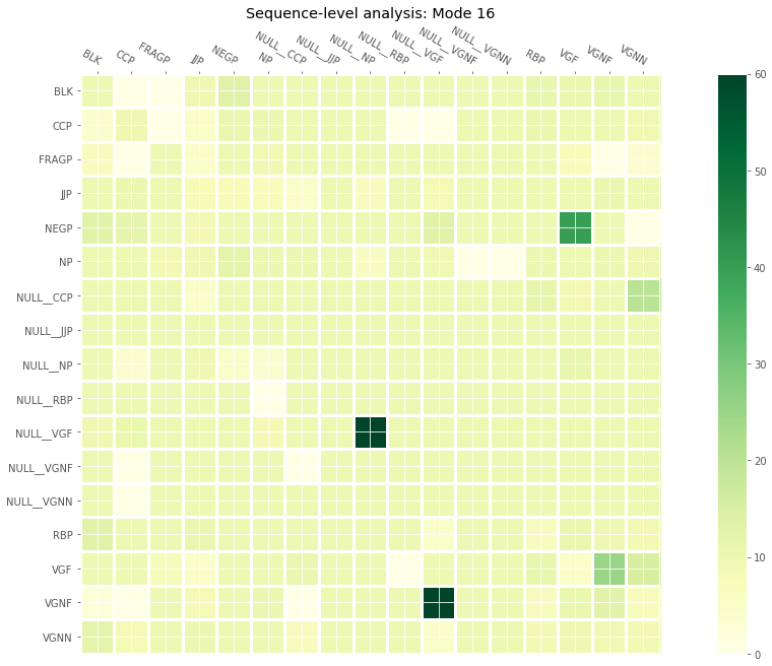

Figure 5.14 Mode 16 - Predicted / Correct Distribution

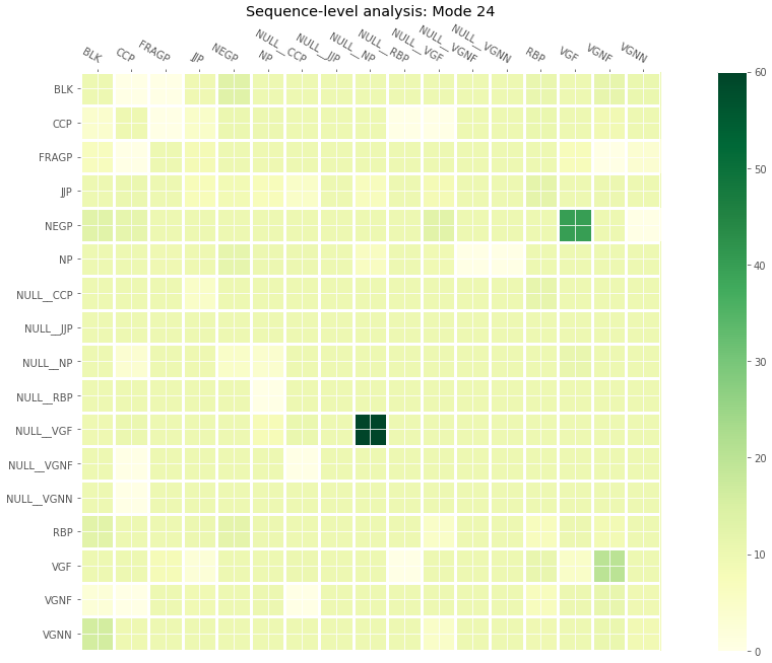

Figure 5.15 Mode 24 - Predicted / Correct Distribution

Next, we have done analysis on the prediction of correct chunk boundaries. Here, in

?? we have shown two plot - one plot shows the tags where extra boundaries have been predicted and one where some boundaries have been skipped. This case arises when a particular chunk tag occurs consecutively in a sentence. For example, if a sentence constitutes of tags NP, NP, NP, VP however in this case, the model can predict the sequence either as NP, NP, VP (boundaries skipped) or as NP, NP, NP, NP, VP (extra boundaries created).

In the skipped boundary plot we can see that, the NP, JJP are the most skipped in the prediction. Apart from these tags, tags like RBP,VGNN, VGNF, VGF and BLK show some skipping.

In the extra boundary plot, although NP chunk tag show significant error in the result, however NULL_VGF becomes much more prominent in this case. Apart from this, JJP, RBP, VGF, VGNF, VGNN show low levels of extra boundary generation. The error mostly arises when the model predicts a verb group as a combination of a verb group and NULL_VGF.

At last, we come to the analysis of the last plot where we have plotted the difference of predicted boundary and correct boundary. This, in general is the measure of errors that occur in the boundary prediction of the tags. The inference that we can make from this is that small errors in boundary predictions occur with almost all tags. These are mainly due to the erroneous labelling of the data.

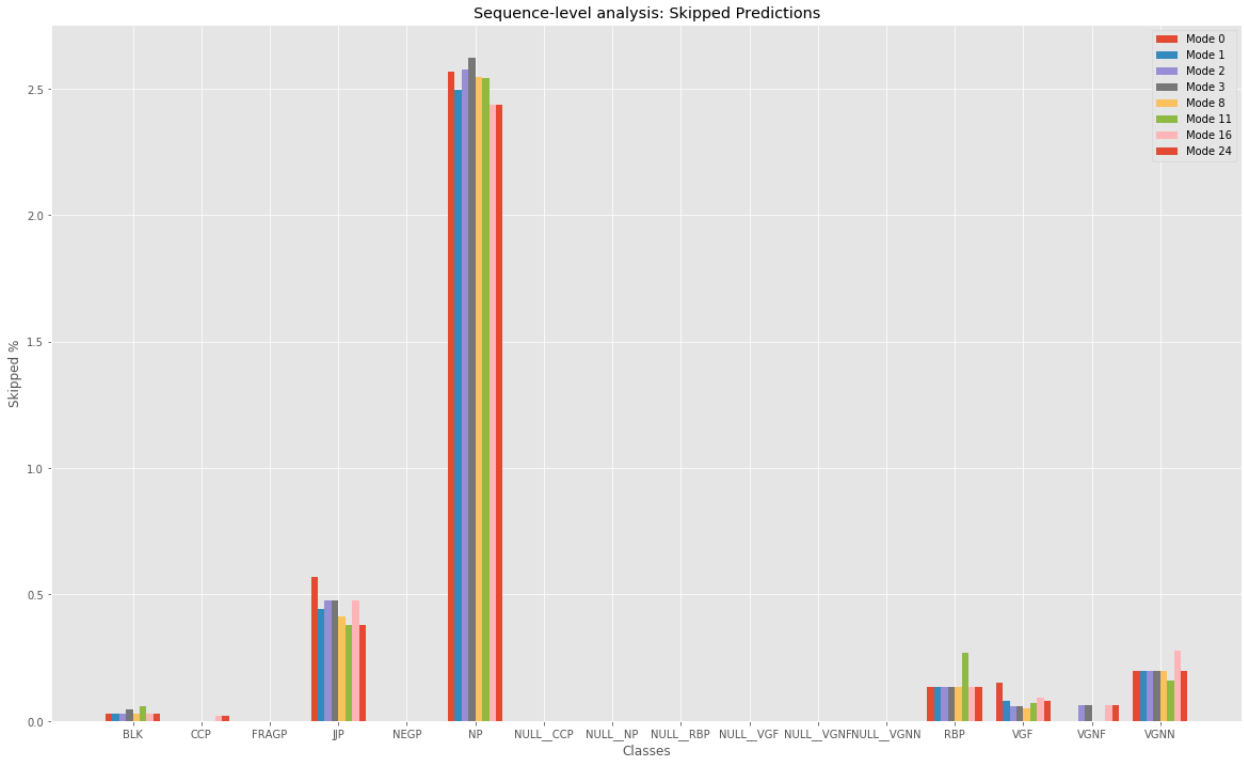

Figure 5.16 Skipped Boundary Predictions comparisons

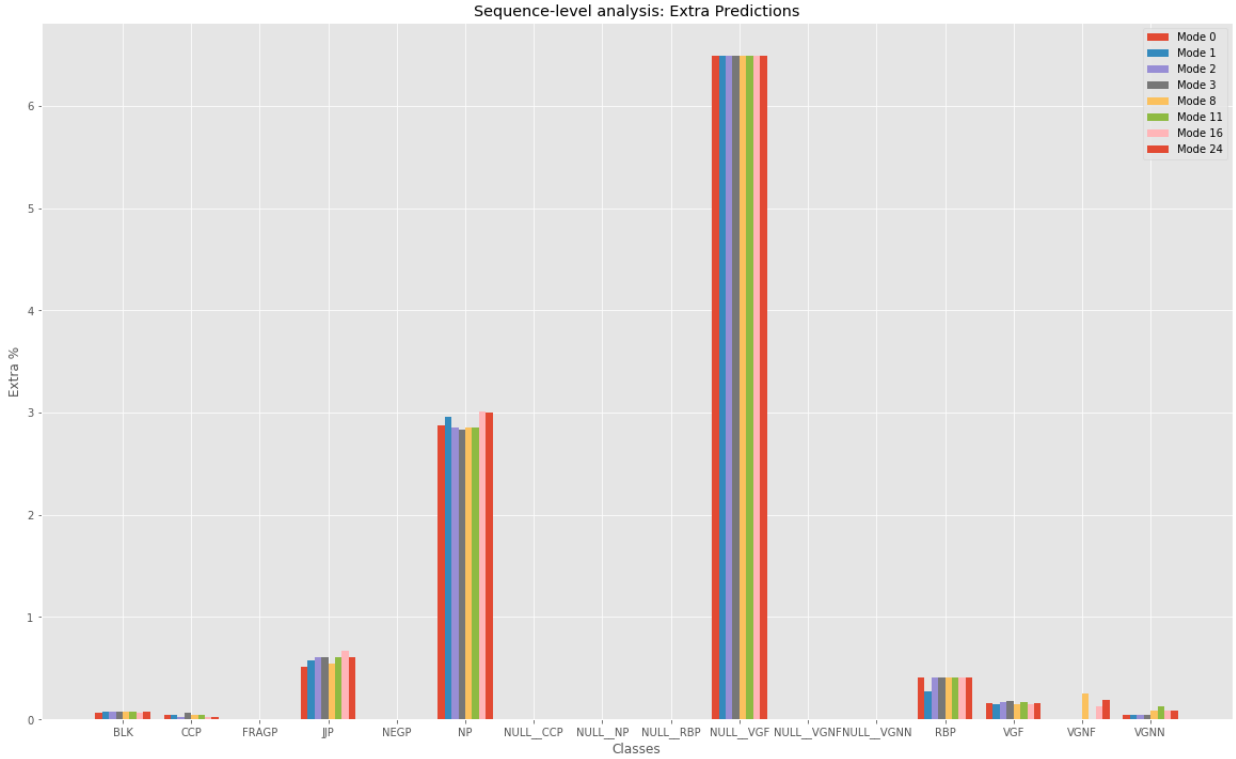

Figure 5.17 Extra Boundary Predictions comparisons

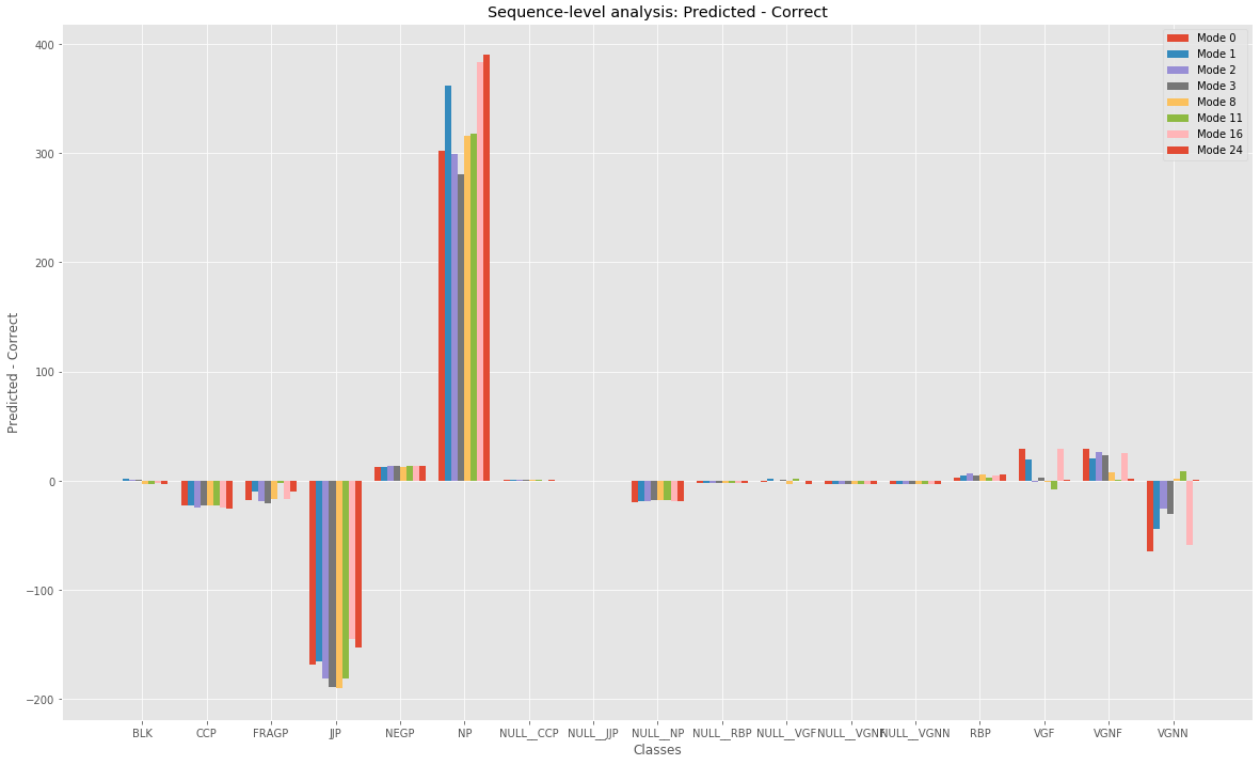

Figure 5.18 Predicted - Correct Boundary values

Chapter 6

Conclusion

After doing error analysis on various levels we come to the conclusion that, there occurs pattern in the dataset which are more prone to error, and certain feature sets perform distinctly better than the others.

- Predicting NP as NP + FRAGP and vice-versa is a very common error that occurs. Apart from that, prediction of NEGP tag is also error prone.
- Use of 'case' in the feature set shows better results in prediction of the results. Suffixes can also play a role in predicting the values correctly.
- Some words like 'binA' are not predicted that well by the model.
- At many places, NP and JJP boundaries are both skipped and created(extra). Although overall sequence order is maintained but such errors occur much more frequently.

References

[1] Charles Sutton, An Introduction to Conditional Random Fields for Relational Learning

[2] Himanshu Agrawal, 2016, Part of Speech Tagging and Chunking with Conditional Random Fields, IIIT Hyderabad

[3] skLearn-crfsuite © Copyright 2015 Mikhail Korobov