CS 224N: MaxEnt Sequence Classifier & Treebank Parsing (PA3)

Sandeep Sripada(ssandeep) & **Venu Gopal Kasturi(venuk)**

1 Introduction

For Programming Assignment-3, the task was to implement named entity recognition and parsing systems. The aim was to examine whether pre-chunking of named entities can improve the performance of a statistical parser trained on a single or hybrid biomedical corpus when applied to the task of parsing another set of biomedical research articles. The maximum entropy classifier implemented was incorporated into a maximum entropy Markov model for doing named entity recognition on biomedical text. Also part of the assignment is the implementation of the CKY parsing algorithm for a broad coverage statistical treebank parser.

The best feature set achieves an accuracy of **90.3%**, precision of **71.08%**, recall of **60.12%** and F1-score of **65.14%**. The parser achieves a best score of **83.92%** over the **genia** dataset with **3-order markovization, unary rule marking** features. The performance on **bioie** is as shown in Table 1.

2 MaxEnt Classifier

The Maximum Entropy Classifier is a multi class classifier which would classify a token to one of the word classes. The classifier is trained using a set of tagged data on the 'genia' corpus. The features of each word are:

BASIC FEATURES: These are the basic features used for each word.

- WORD_NAME the current word name
- PREV_LABEL this acts like a minimalistic context dependence on the label of the previous word

TOP LEVEL WORD CHARACTERISTICS: These are features which correspond to the characteristics of the word at the top level.

- IS_NUMBER whether the word is a number
- IS_ALPHA_NUM whether the word is an alpha numeric word useful for words like VCAM-1, anti-CD3 etc.
- HAS_HYPHEN whether the word has a hyphen
- ABBREVIATION if the word is an abbreviation ie word is in upper case and length ¡ 5. This is helpful in recognizing that abbreviations like HUVEC, CEM etc can be entities.
- HAS_PARENTHESIS has parenthesis ie matches .*(.*).*
- END_WITH_HYPHEN ends with a hyphen
- BEGINS_WITH_HYPHEN begins with a hyphen
- UPPER_CASE_AFTER_HYPHEN if the char succeeding a hyphen is in upper case
- LOWER_CASE_AFTE_HYPHEN if the char succeeding a hyphen is in lower case
- UPPERCASE_NUMBER if the word is in uppercase followed by a number
- LOWERCASE_NUMBER if the word is in lower case followed by a number
- HAS_COMMA if the word contains a comma

SPECIALIZED FEATURES: These are features which are specific to the data we are using

- HAS_SSA if the word has SSA in it,this is particularly useful in tagging "protein" class
- END_STERONE if the word ends with "sterone", useful for protein class
- END_CYTE if the word ends with "cyte" or "cytes", useful for "cell_line" class
- HAS_GREEK if the word contains "alpha", "beta", "kappa", useful to identify non-zero classes and useful to identify "protein" class
- HAS_MONO if the word contains "mono", useful to identify "cell_type" class. But it is causing more misclassifications than it helps in classification
- ENDS_WITH_RNA if the word ends with "RNA", useful in identifying "RNA" class

CONTEXT DEPENDENT FEATURES: These features provide context information to the word we are looking at. But these features are extremely sparse. But they are extremely useful and have given a huge boost in the accuracy measure.

Table 1: Accuracy, Precision, Recall and F1 score on various datasets of the CKY parser.

Dataset	Features	Precision	Recall	F1	Exact match
bioie	-	70.47	57.66	63.42	0.00
bioie	VM 1	80.52	69.04	74.34	0.00
bioie	VM 2	83.36	73.05	77.87	0.00
bioie	VM 3	83.58	74.18	78.6	0.00
genia	-	76.47	66.90	71.36	4.92
genia	VM 1	82.97	73.16	77.75	4.92
genia	VM 2	83.68	79.22	81.39	8.20
genia	VM 3	83.92	79.51	82	8.20

- PREV_1_WORD The previous word if present
- PREV_2_WORD The 2nd previous word, if present
- PREV_3_WORD The 3rd previous word, if present
- NEXT_1_WORD The next word, if present
- NEXT_2_WORD The 2nd next word, if present
- NEXT_3_WORD The 3rd next word, if present

MISC FEATURES: These are some of the miscellaneous features that we have tried.

- IS_COMMON: This features indicates if the word being considered is a Stop word ie a word is too frequent in the english language. We believed this would help us in identifying the non-entities.

2.1 Analysis of effectiveness of features

HAS_COMMA: This has given a good increase in the accuracy of the classifier. bis[4-[2,4-dioxo-5-thiazolidinyl)methyl]phenyl]methane was initially identified as Protein, but now it has been correctly identified as non-entity. Same is the case with phosphatidylinositol-3,4,5-trisphosphate and PI-3,4,5-P3 and "peptides". The output of the accuracy between the two runs is shown below.

Protein: precision: 56.30%; recall: 53.02%; FB1: 54.61 3902 (with HAS_COMMA feature)
Protein: precision: 54.05%; recall: 51.91%; FB1: 52.95 3980 (without HAS_COMMA feature)

BEGIN_WITH_CAP: This feature is ON if a word begins with a capital letter. It was able to correctly classify words like RelA, CD3epsilon, POU-specific, Jun.

CONTEXT DEPENDENT features: By adding the features PREV_WORD, .. PREV_3_WORD and NEXT_1_WORD .. NEXT_3_WORD, we have obtained a huge boost in the accuracy of the classifier.

Without Context features: processed 69051 tokens with 7119 phrases; found: 5308 phrases; correct: 2806. **accuracy: 84.15%; precision: 52.86%; recall: 39.42%; FB1: 45.16** DNA: precision: 38.94%; recall: 9.44%; FB1: 15.19 339 RNA: precision: 0.00%; recall: 0.00%; FB1: 0.00 0 cell_line: precision: 63.71%; recall: 14.96%; FB1: 24.23 124 cell_type: precision: 42.21%; recall: 43.40%; FB1: 42.80 943 protein: precision: 56.30%; recall: 53.02%; FB1: 54.61 3902

With Context features: processed 69051 tokens with 7119 phrases; found: 6021 phrases; correct: 4280. **accuracy: 90.03%; precision: 71.08%; recall: 60.12%; FB1: 65.14** DNA: precision: 69.98%; recall: 47.82%; FB1: 56.82 956 RNA: precision: 72.53%; recall: 50.38%; FB1: 59.46 91 cell_line: precision: 64.71%; recall: 50.00%; FB1: 56.41 408 cell_type: precision: 72.96%; recall: 64.45%; FB1: 68.44 810 protein: precision: 71.62%; recall: 64.91%; FB1: 68.10 3756

We can see that context sensitive information is very helpful in obtaining the correct labels for the classes. For example in the RNA class, IL-4, mRNA are previously misclassified but now it is being correctly classified. You can observe a huge jump in the % classification for that class in the stats mentioned.

2.2 Analysis of the sets of features

We performed the following experiments:

- BASIC features alone (1)
- BASIC + TOP LEVEL WORD characteristics features (2)
- (2)+SPECIALIZED FEATURES (3)
- (3)+CONTEXT FEATURES (4)
- (4)+MISC FEATURES
- Learning the model by increasing the number of iterations

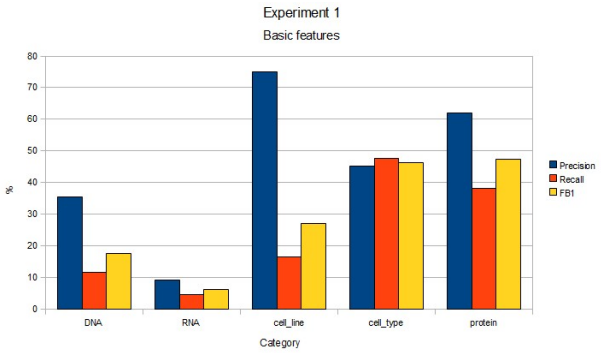

Figure 1: Experiment 1.

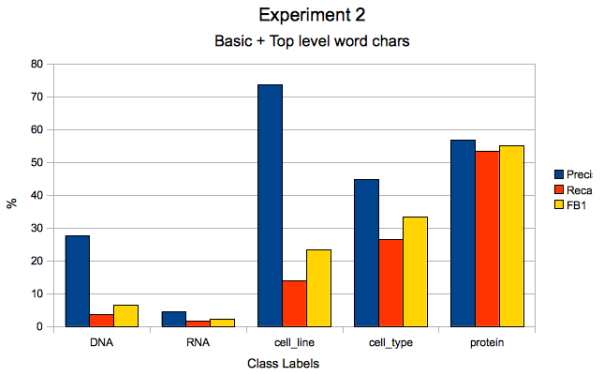

Figure 2: Experiment 2.

- 40 iterations
- 80 iterations
- 120 iterations

The histograms for the experiments are as shown in Figure 1, Figure 2, Figure 3, Figure 4, Figure 5, Figure 6, Figure 7, Figure 8. These plots show the f1, precision, recall scores for various class labels.

3 Treebank Parsing

In this section, we describe the implementation of our CKY parser, provide an analysis of the performance, and discuss possible improvements to the parser that would correct frequently occurring errors.

3.1 Implementation and Optimizations

There were quite a few optimizations that we did in order to speed up the parsing.

- **Score and back structures:** Since there is no need to iterate over the score and back structures, there was no need to initialize a 3D array with the complete size. Instead, we used a Hashmap with the key formed by using the begin, split, end indices and the

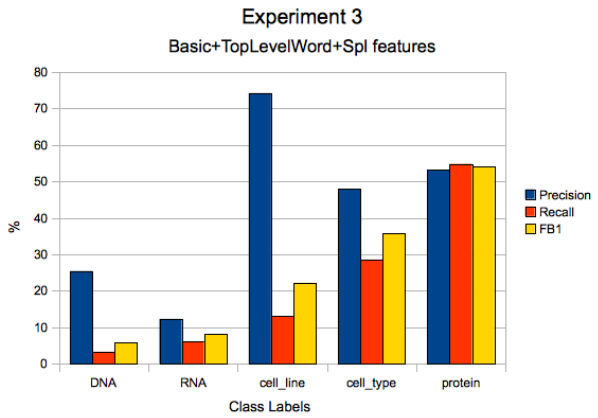

Figure 3: Experiment 3.

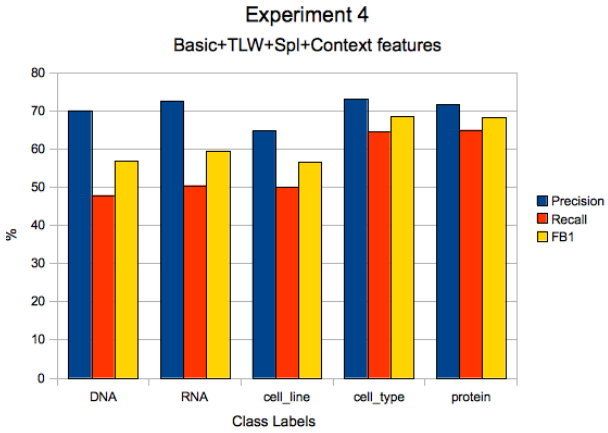

Figure 4: Experiment 4.

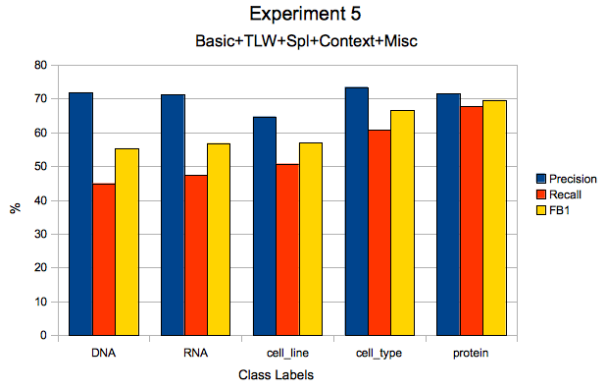

Figure 5: Experiment 6.

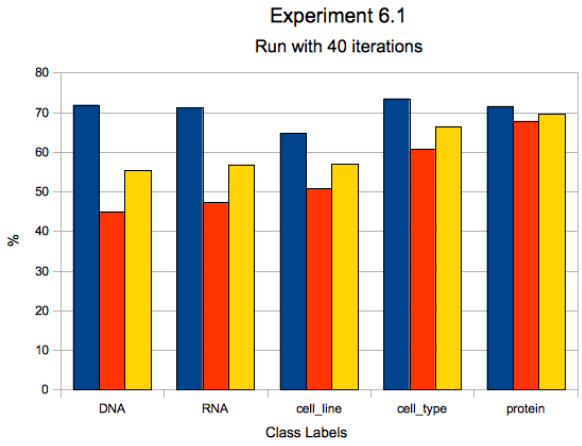

Figure 6: Experiment 6 - 40 iterations.

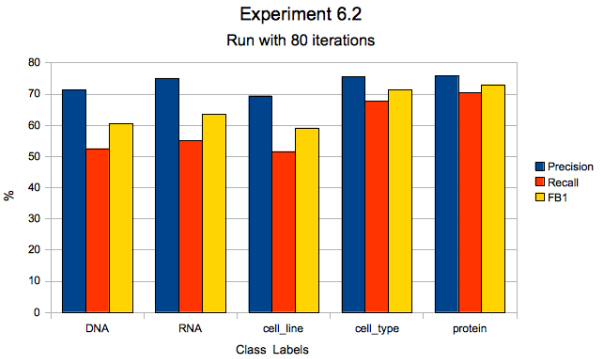

Figure 7: Experiment 6 - 80 iterations.

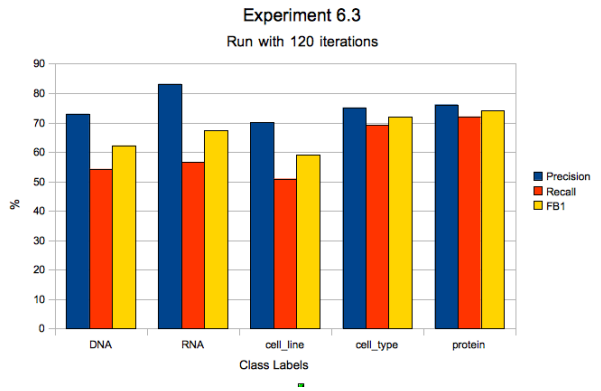

Figure 8: Experiment 6 - 120 iterations.

non-terminal symbols as a '|' separated string. This worked quite effectively as keys were only placed in the map if corresponding value exists and is directly fetched in O(1) time.

- **Non-terminal set:** The inner loops in the CKY algorithm iterate through all non-terminal symbols and proceed if the grammar rule formed by the chosen non-terminals is present. This is not required and instead of iterating in a 3-level nested for loop through all the non-terminal set, we ran the algorithm over all the rules present in the grammar as indexed by either the left/right children. This saved us a lot of time and eliminated the need to iterate $\sim$ 30 billion times (with around 3000 non-terminals).

- For rules of the type $A \rightarrow BC$ or $A \rightarrow B$, the loop was evaluated only if the score structure contained an entry for the non-terminals B, C (which also means that their value is > 0). This also helped us a lot as unnecessary log manipulations were not done when the score entries were zero.

- Instead of creating extra structures holding various temporary values, we used the structures already made available in the grammar and lexicon classes. Eg: binaryRulesByLeftChild to get the list of all binary rules (as used in the toString() method).

- To avoid underflow errors, we use log adds to deal with probability calculations.

- For both span 1 and span 2 or higher, unary rules were handled along with the other rules.

3.2 Analysis

As shown in Figure 9, one of the common errors made by the parser is with the assignment of singular noun (NN) tag to plural nouns which should have been marked NNS. This is a minor error but is frequent at times. Another common error as shown in Figure 10 and Figure 12, is with the expansion of the noun phrase tag. Instead of expanding all singular nouns (NNs) in under one noun phrase (NP) tag, the best parse sometimes expands NP− >NP NP and one of the singular nouns is attached to the last NP tag. This could be avoided by assigning all NNs to the first NP tag rather than giving the parser a chance to expand it using another NP tag. This might be done either by giving more weight to rules of the form NP− >NP NN NN or reducing the weight of NP− >NN once an earlier NN is already associated with a NP.

Another error that is quite frequent is the expansion of PP to give NP phrases. The part where PP expands to give IN and NP with NP in turn expanding to NP and NP is a common erroneous subtree. This is shown

in Figure 14. This figure also has the error mentioned earlier of the attachment of NN tags to a single NP rather than two different NP tags.

The placement of PP tags is another error where the PP tags are misplaced under the wrong NP tag. They are sometimes placed in the outer level NP tag rather than the inner level NP tag which leads to the outer Noun phrase now having two consecutive PP's. This can be avoided by placing a negative weight/score to such attachments. This is shown in Figure 11.

More rare occurrence of errors are with SYM tags as shown in Figure 13. The SYM tree is wrongly attached under the NML tag whereas it should have been under the PP tag. Also, the assignment of SYM to a '-' and not HYPH is another error which just occurred once because the symbol '-' itself occurred just once. This could have occurred because of the possibility of the rules having equal probability.

3.3 Enhancements done

Markovization: We have implemented vertical markovization with various orders. The order/level of annotation could be changed as a parameter and the appropriate parent annotation was done. The results of the parser showed a significant improvement after including a vertical markovization of order 2. We also tried with higher order tagging, but the improvements obtained were negligible.

Marking Unaries: We implemented a unary marking where all modes which have only one child were annotated with a 'U' tag. This simple inclusion also proved to be useful and contributed to improved scores.

4 Treebank Parsing combined with NER

NER tagging was useful for parsing in cases where the non-chunked version shows errors like wrong labeling of tags and wrong positioning of a sub trees. The chunked version for the above shown examples has one of the labeling errors but it gets one of the subtree placement right. The NP tag is correctly placed. In some of the cases, the NER chunking actually proved to reduce the parsing match.

Examples where chunking fails: Sentence: [Natural, variants, of, the, HIV-1, long, terminal, repeat, :, analysis, of, promoters, with, duplicated, DNA, regulatory, motifs, .] is transformed to chunked sentence: [Natural, variants, of, the, HIV-1_long_terminal_repeat, :, analysis, of, promoters, with, duplicated, DNA_regulatory_motifs, .]. The word is joined together (long terminal repeat) thereby reducing the accuracy completely as leaf nodes are not matched.

Another example where a similar thing happens: test sentence: [Mutations, in, the, Pit-1, gene, in, children, with, combined, pituitary, hormone, deficiency, .] is transformed to chunked sentence: [Mutations, in, the, Pit-1_gene, in, children, with, combined, pituitary, hormone, deficiency, .] where the words Pit-1 and gene are combined.

There were quite a few instances where these kinds of errors have reduced the parsing accuracy. So we cannot really say whether chunking helps or not. In some cases it helps and in some cases the performance was worsened.

5 Future Work and Improvements

- In the parsers, some of the rules might have equal probability at the end and the selection of the final rule depends highly on how the equality is handled. Once small change which could be greatly useful is to assign some kind of weighting scores for all rules so that ones with equal probability can be disambiguated. For eg: giving more weight to expected rules should break the equality and help in the most likely rule to be chosen. Note: This is after the CKY parser calculates the final values - during selection of which rule to take up. The weights could be set by maximizing on some validation data.

- As mentioned in the earlier cases, singular/plural noun errors and labeling errors could be reduced by using more training data.

- Another problem was with incorrect positioning of subtrees due to misinterpretation of phrases and their positions in the sentence. These can be avoided again by some additional training data which would enhance probabilities of the likely rules. Another way to improve upon these errors is to do a more semantic analysis like using Wordnet to get senses and disambiguating/or enhancing rule probability by giving more weight to likely senses.

- For max ent: The major problem has been that we are not training the model well enough, by cutting short the iterations at 40, this is leading to the fact that the Objective function is very large. This is negating the improvements that can be achieved by using good features. As initially the classifier tends to misclassify but when the iterations are increased, it learns the parameters well and gives us better results. This can be seen in the learning curve when the iterations are increased.

- Since we have noticed that adding context information is giving us a huge boost in the accuracy, it would be interesting to see the behavior when the previous label is also added to the feature list. Currently we are only using the immediate preceding label alone.

6 Contributions

Sandeep Sripada: Worked on parsing and improvements. Worked on the report for corresponding part.
Venu Gopal Kasturi: Worked on MaxEnt classifier and improvements. Worked on the report for corresponding part.

References

[Lectures] Statistical parsing lectures.
Chris Manning.

```
Guess:
(ROOT
  (S
    (NP (DT These) (NNS results))
    (VP (VBP indicate)
      (SBAR (IN that)
        (S
          (NP (JJ alpha-lipoic) (NN acid))
          (VP (MD may)
            (VP (VB be)
              (ADJP (JJ effective))
              (PP (IN in)
                (NP (NN AIDS) (NN therapeutics))))))))
    (. .)))

Gold:
(ROOT
  (S
    (NP (DT These) (NNS results))
    (VP (VBP indicate)
      (SBAR (IN that)
        (S
          (NP (JJ alpha-lipoic) (NN acid))
          (VP (MD may)
            (VP (VB be)
              (ADJP (JJ effective))
              (PP (IN in)
                (NP (NN AIDS) (NNS therapeutics))))))))
    (. .)))
```

Figure 9: Common mistake with NNS tag.

```
Guess:
(ROOT
  (NP
    (NP (CD One) (NN HIV-1))
    (NP (NN variant)))
  (VP (VBD was)
    (VP (VBN detected)
      (VP (VBG containing)
        (NP
          (NP (DT an) (JJ additional) (VBG binding) (NN site))
          (PP (IN for)
            (NP (DT the) (NN transcription) (NN factor) (NN Sp1)))))))
  (. .))

Gold:
(ROOT
  (S
    (NP (CD One) (NN HIV-1) (NN variant))
    (VP (VBD was)
      (VP (VBN detected)
        (S
          (VP (VBG containing)
            (NP
              (NP (DT an) (JJ additional) (VBG binding) (NN site))
              (PP (IN for)
                (NP (DT the) (NN transcription) (NN factor) (NN Sp1))))))))
    (. .)))
```

Figure 10: Common mistake with the expansion of NP tag.

```
Guess:
(ROOT
  (NP
    (NP (JJ Specific) (NN PCR) (NN amplification))
    (PP (IN for)
      (NP
        (NP (NN N-ras) (NNS mutations))
        (PP (IN in)
          (NP (JJ neoplastic) (JJ thyroid) (NNS diseases))))))
  (. .))

Gold:
(ROOT
  (
    (NP
      (NP (JJ Specific) (NN PCR) (NN amplification))
      (PP (IN for)
        (NP (NN N-ras) (NNS mutations)))
      (PP (IN in)
        (NP (JJ neoplastic) (JJ thyroid) (NNS diseases)))
      (. .))))
```

Figure 11: Common mistake with PP tag attachment. PP is sometimes associated to the wrong branch.

```
Guess:
(ROOT
  (NP
    (NP (DT Another) (NN LTR) (NN size))
    (NP (NN variation)))

Gold:
(ROOT
  (S
    (NP (DT Another) (NN LTR) (NN size) (NN variation))
```

Figure 12: Common mistake with the expansion of NP tag. (Only the mistake is shown, rest of the tree is truncated)

```
Guess:
          (NP (DT a)
            (NML (NN G) (SYM -->) (NN C))
            (NN substitution)))
        (PP (IN in)
          (NP (NN codon) (CD 13))))
      (. .))))

Gold:
          (NP (DT a)
            (NML
              (NML (NN G))
              (PP (SYM -->)
                (NP (NN C))))
            (NN substitution))
          (PP (IN in)
            (NP (NN codon) (CD 13))))))
      (. .))))
```

Figure 13: Common mistake the association of SYM tag.

```
Guess:
(ROOT
  (S
    (PP (IN In)
      (NP (NN contrast)))
    (, ,)
    (NP
      (NP (JJ antisense) (NN inhibition))
      (PP (IN of)
        (NP (NN Tax) (NN itself))))
    (VP (VBD had)
      (NP
        (NP (DT no) (JJ apparent) (NN effect))
        (PP (IN on)
          (NP (NN cell) (NN growth)))))
    (. .)))

Gold:
(ROOT
  (S
    (PP (IN In)
      (NP (NN contrast)))
    (, ,)
    (NP
      (NP (JJ antisense) (NN inhibition))
      (PP (IN of)
        (NP
          (NP (NN Tax))
          (NP (PRP itself)))))
    (VP (VBD had)
      (NP (DT no) (JJ apparent) (NN effect))
      (PP (IN on)
        (NP (NN cell) (NN growth))))
    (. .)))
```

Figure 14: Common mistake with the expansion of PP to give NP phrases.