CSE 517: Homework 3 - Structured Learning with Feature-Rich Models

Lilly Kumari

Problem 1. CFG Grammar Refinement

Solution 1.1 - a PCFG with rule probabilities

- N : {NP, VP, SBAR, V1, V2, COMP, ADVP }
- Σ: {John, said, that, Sally, snored, loudly, declared, Bill, ran, quickly, Fred, pronounced, Jeff, swam, elegantly}
- S: {S}
- R : Figure 1

NP ->	John	0.17
V1 ->	said	0.33
COMP ->	that	1.0
NP ->	Sally	0.33
V2 ->	snored	0.33
ADVP ->	loudly	0.33
V1 ->	declared	0.33
NP ->	Bill	0.17
V2 ->	ran	0.33
ADVP ->	quickly	0.33
NP ->	Fred	0.17
V1 ->	pronounced	0.33
NP ->	Jeff	0.17
V2 ->	swam	0.33
ADVP ->	elegantly	0.33

S ->	NP VP	1.0
VP ->	V1 SBAR	0.33
SBAR ->	COMP S	1.0
VP ->	VP ADVP	0.33
VP ->	V2	0.33

Figure 1:

Parse Tree – 1

S
NP Jeff
VP
V1 pronounced
SBAR
COMP that
S
NP Bill
VP
VP
V2 ran
ADVP elegantly

Parse Tree - 2

S
NP Jeff
VP
VP
V1 pronounced
SBAR
COMP that
S
NP Bill
VP
V2 ran
ADVP elegantly

Figure 2:

Jeff	pronounced	that	Bill	ran	elegantly
NP: 0.17	None	None	None	S: (NP,VP) 1*0.17*0.33*0.33*0.17*0.33^2	S: (NP,VP) 0.17*0.33*0.33*1*0.17*0.33^4 S: 0.17*0.33^4
	V1: 0.33	None	None	VP: (V1,SBAR) 0.33*0.33*0.17*0.33^2	VP: (V1,SBAR) 0.33*0.33*1*0.17*0.33^4
		COMP: 1.0	None	SBAR: (COMP,S) 1*1*0.17*0.33^2	SBAR: (COMP,S) 1*0.17*0.33^4
			NP: 0.17	S: (NP,VP) 1.*0.17*0.33*0.33	S: (NP,VP) 1*0.17*(0.33)^4
				V2: 0.33 VP: 0.33*0.33	VP: 0.33*0.33*0.33*0.33
					ADVP: 0.33

Figure 3: Parse Tree 1

1.1 - b

Figure 2, 3, 4 show the parse trees and the CKY charts of the two trees drawn from the given corpus.

Probability of first parse tree = $(0.17)^2 * (0.33)^6$

Probability of second parse tree = $(0.17)^2 * (0.33)^4$

1.1 - c I will try doing lexicalization where each non-terminal is annotated with its lexical head. Heads would propagate up in the tree.

From the collected corpus, we will have rules in the following form:

- V2 (ran, V2) -> ran (0.33)
- ADVP (elegantly, ADVP) -> elegantly (0.33)
- ADVP (quickly, ADVP) -> quickly (0.33)
- VP (ran, V2) -> V2 (ran, V2) (1)

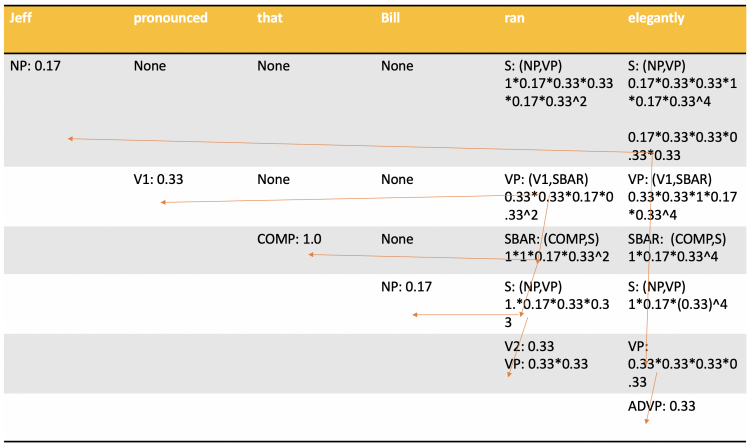

Figure 4: Parse Tree 2

- VP (ran, V2) -> VP (ran, V2), ADVP (quickly, ADVP) (1)
- V1 (pronounced, V1) -> pronounced (0.33)
- VP (pronounced, V1) -> V1(pronounced, V1), SBAR(that, COMP) (1)

So, this way the VP from topmost S is always going to decompose into V1 (for *pronounced* and SBAR). Also, ADVP has attachemnt to VP annotated with a *verb* and V2. Hence, this lexicalization here would ensure zero probability to the high attachment of ADVP.

Problem 2. CFG Grammar Refinement II

Solution CKY parser with default PCFG will not completly learn to disambiguate correct structure related to PP attachment even after being provided with sufficient amount of annotated trees since PP attachment mostly happens to either NP or VP. Based on the probabilities that we get from the annotated corpus, the parser would be more inclined towards the one with the higher probability, i.e. either NP or VP attachment. PP attachment depends mostly on the affinity between the (noun term and the preposition) or the (verb term and preposition). Since a lexicalized version of PCFG focuses on propagating lexical heads (which based on the higher affinity to the preposition, would be noun or verb term) up in the tree, lexicalization might be better at resolving this ambiguity.

Problem 3. Named-Entity Recognition (NER) with Structured Perceptrons

Solution 3.3 - a - **Feature Engineering done so far:**

- Overall feature vector length = 4146
- lower case unigram merged with NER tag - when frequency of occurrence is greater than 3
- length of token merged with NER tag - when frequency of occurrence is greater than 3
- POS tag of current token merged with NER tag (when freq $>$ 3 in training data)
- Syntactic chunk tag merged with NER tag (when freq $>$ 3)

- 1st 2 letters (prefix) merged with NER tag (when freq > 3)
- last 2 letters (suffix) merged with NER tag (when freq > 3)
- 1st three letters (prefix) merged with NER tag (when freq > 3)
- last three letters (suffix) merged with NER tag (when freq > 3)
- Shape based features - for example, Bill - Xxxx (with and without NER tag)
- Count of shape features - for example, Bill - Xx (with and without NER tag)
- Number based features- has number, is number, is integer (with and without NER tag)
- Hyphen based features - has hyphen, or is hyphen (with and without NER tag)
- Period based features - has period, or is period (with and without NER tag)
- Apostrophe based features - has apostrophe or is apostrophe (with and without NER tag)
- Double Apostrophe based features - has or is ((with and without NER tag)
- Colon based features - has colon or is (with and without NER tag)
- Semicolon based features - has semicolon or is (with and without NER tag)
- All letters are capital or not (with and without NER tag)
- Is first letter capital or not (with and without NER tag)
- If the word contains just one letter and is capital (with and without NER tag)
- capital letters in between other letters (with and without NER tag)
- does the word end with s (with and without NER tag)
- does the word end with ing (with and without NER tag)
- does the word end with ed (with and without NER tag)

Additional feature templates that could improve NER performance:

- Context based word features - for example, is previous word capitalized, does previous word contain hyphen, is previous word numeric, is previous word *the*; is next word capitalized, does the next word end with *ing* and *ed*?
- Context based tag features - Is previous tag the beginning (B) of a NER tag, is next tag the Inside (I) of a NER tag
- Word clustering based features - applying clustering techniques on the set of corpus and learning word representations which could identify similarity (or affinity) between words and using those representations as features

- Gazetteer based knowledge - using some provided pre-compiled list of names of persons, organizations, cities, countries and building features based on the presence of a word in such existing knowledge-bases

- Idiom and Phrase based collocation scoring - using some existing idioms and typical phrases knowledge-base, one can use the probability of occurrence of a word within an idiom or phrase to construct new features (using a certain threshold on probability to make it a Boolean variable)

3.3 - b - Viterbi decoding Algorithm

We want to model this:

$$score(\mathbf{y}|\mathbf{x}) = w^\top \phi(\mathbf{y}, \mathbf{x}) \tag{1}$$

which decomposes to:

$$score(\mathbf{y}|\mathbf{x}) = w^\top \sum_i \phi\left(\mathbf{x}, i, y_i, y_{i-1}\right) \tag{2}$$

where i traverses along the length of the sentence.

Using Viterbi algorithm, we get y^* such that:

$$\mathbf{y}^* = \arg\max_{\mathbf{y}} \text{score}(\mathbf{y}|\mathbf{x}) \tag{3}$$

Hence the recursive algorithm to model it is following:

$$\pi\left(i, y_i\right) = \max_{y_{i-1}} w^T \cdot \phi\left(x, i, y_{i-1}, y_i\right) + \pi\left(i-1, y_{i-1}\right) \tag{4}$$

3.3 - c - BIO vs IO encoding

- IO encoding is kind of the most simplest encoding technique which just tags tokens as either being in (I) a particular entity type or not (O). That means that it lacks the demiliter property because it's not capabel of decoding subsequent entities of the same type. But since in language, subsequent entities belonging to the same type are quite rare, it almost shows comparable performance to BIO encoding. So, it depends on the data and task at hand because if the corpus contains very few cases of presence of subsequent entities of same type, then IO will perform almost similar to BIO. Also, since there are less number of tags that way, it becomes easy to distinguish one tag vs the pool (which would contain almost two-third of possible types for BIO). Hence, IO can perform better than BIO depending on the data and task.

- BIO encoding - pros: able to decode subsequent entities of same type. Cons: Since the decoding is quadratic in terms of number of tags which are (2N + 1) in case of BIO, the decoding can be slow and complex. Annotation of dataset with BIO tags is complicated here and needs good expertise.

- IO encoding - pros: decoding faster because of less number of tags (N + 1), Annotation of dataset with IO tags is rather simple. Cons: failure to decode subsequent entities belonging to the same type

3.3 - d - Error Analysis and Discussion

- Trained the perceptron algorithm by focusing on current tag and previous tag only (bigram order of tags considered for global feature decomposition).
- Since the feature vectors are very sparse, I used csr_matrix provided by scipy to construct both local and global feature vectors.
- didn't construct features based on context, for example, previous token or next token (word, pos, syn tag, ner tag). So, F1 - score is comparatively low as compared to competitive systems. Used only the small (reduced) datasets.
- Overall feature vector dimension is 4146 and the model was trained for 20 iterations since accuracy saturated after that.
- **Error Analysis** on dev data (small) is presented below along with the confusion matrix.

  Accuracy on Dev set = 92.7 % Accuracy on Test set = 90 %

  Figure 5 shows the confusion matrix obtained from predictions on Dev dataset.

  Figure 6 shows the confusion matrix obtained from predictions on test dataset.

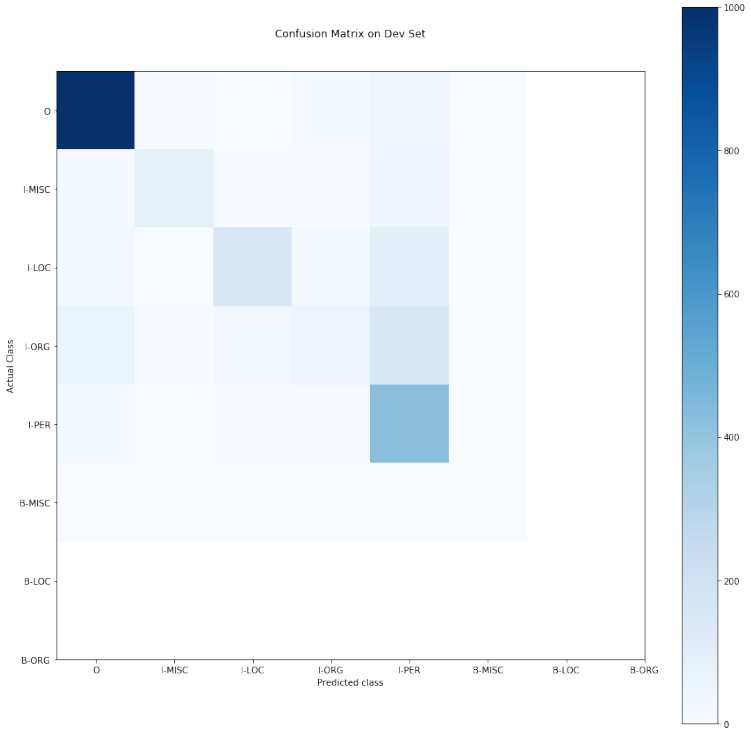

Figure 5:

Figure 7 shows the evaluation metrics obtained via running the conlleval.txt script on respective output text files.

- As it can be seen from Figure 5 that most of the I-ORG tags (has the lowest recall) are incorrectly labelled as I-PER or O. Similarily, I-LOC tags are also quite often misclassified as I-PER (not when all the letters are capitalized as can be seen in the figure 8), same goes

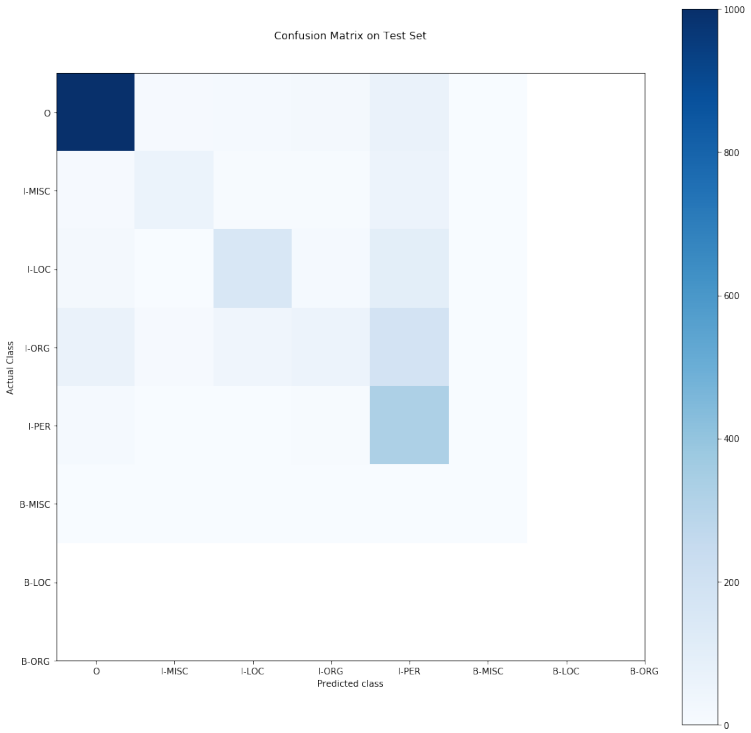

Figure 6:

for I-MISC. Since the dev set contains only 1 B-MISC tag, it's of no point to check for its correct classification. O and I-PER tags are mostly correctly classified (with a high recall).

- Figure 8 shows some example sentences in which the last two columns are true and predicted tags.
- The model does not perform well in distinguishing between I-PER and I-LOC when the first letter is capitalized. This could be improved by considering context based features and while decoding, looking at 2, 3 or 4 previous tags if possible.

On Dev Dataset

processed 7342 tokens with 817 phrases; found: 891 phrases; correct: 449.
accuracy: 92.75%; precision: 50.39%; recall: 54.96%; FB1: 52.58
LOC: precision: 76.40%; recall: 53.33%; FB1: 62.82 178
MISC: precision: 71.43%; recall: 54.05%; FB1: 61.54 84
ORG: precision: 25.84%; recall: 11.92%; FB1: 16.31 89
PER: precision: 42.59%; recall: 89.15%; FB1: 57.64 540

On Test Dataset

processed 6288 tokens with 796 phrases; found: 890 phrases; correct: 381.
accuracy: 89.87%; precision: 42.81%; recall: 47.86%; FB1: 45.20
LOC: precision: 69.27%; recall: 53.63%; FB1: 60.45 192
MISC: precision: 63.38%; recall: 41.28%; FB1: 50.00 71
ORG: precision: 27.00%; recall: 11.34%; FB1: 15.98 100
PER: precision: 33.40%; recall: 87.56%; FB1: 48.35 527

Figure 7:

Misclassification- I-ORG

May NNP I-NP O O
17 CD I-NP O O
v SYM I-VP O O
Northampton NNP I-NP I-ORG O

August NNP I-NP O O
1-4 CD I-NP O O
v SYM I-VP O O
Somerset NNP I-NP I-ORG I-PER
((O O O
four CD I-NP O O
days NNS I-NP O O
)) O O O

Zycie NNP I-NP I-ORG I-PER
Warszawy NNP I-NP I-ORG I-PER
said VBD I-VP O O
the DT I-NP O O
new JJ I-NP O O
, , I-NP O O
open JJ I-NP O O
consortium NN I-NP O O
, , O O O
which WDT I-NP O O
also RB I-ADVP O O
included VBD I-VP O O
several JJ I-NP O O
listed VBN I-NP O O
Polish JJ I-NP I-MISC I-PER
firms NNS I-NP O O
, , O O O
would MD I-VP O O
on IN I-PP O O
Monday NNP I-NP O O
inform VBP I-VP O O
Privatisation NNP I-NP O I-PER
Minister NNP I-NP O O
Wieslaw NNP I-NP I-PER I-PER
Kaczmarek NNPS I-NP I-PER I-PER
of IN I-PP O O
its PRP$ I-NP O O
plans NNS I-NP O O
. . O O O

I-LOC

Romania NNP I-NP I-LOC I-PER
beat VBD I-VP O O
Lithuania NNP I-NP I-LOC I-PER
2-1 CD I-NP O O
((I-NP O O
halftime NN I-NP O O
1-1 CD I-NP O O
)) O O O
in IN I-PP O O
their PRP$ I-NP O O
European JJ I-NP I-MISC I-MISC
under-21 JJ I-NP O O
soccer NN I-NP O O
match NN I-NP O O
on IN I-PP O O
Friday NNP I-NP O O
. . O O O

SOCCER NN I-NP O O
- : O O O
ROMANIA NNP I-NP I-LOC I-LOC
BEAT NN I-INTJ O O
LITHUANIA NNP I-NP I-LOC I-LOC
IN IN I-PP O O
U-21 NNP I-NP O I-MISC
QUALIFIER NNP I-NP O O
. . O O O

I-MISC

- : O O O
Speaker NNP I-NP O O
of IN I-PP O O
parliament NN I-NP O O
Habib NNP I-NP I-PER I-PER
Boulares NNP I-NP I-PER I-PER
arrives VBZ I-VP O O
in IN I-PP O O
Tripoli NNP I-NP I-LOC I-PER
to TO I-VP O O
represent VB I-VP O O
President NNP I-NP O O
Zine NNP I-NP I-PER I-PER
al-Abidine JJ I-NP I-PER I-MISC
Ben NNP I-NP I-PER I-PER
Ali NNP I-NP I-PER I-PER
at IN I-PP O O
the DT I-NP O O
Libyan JJ I-NP I-MISC I-LOC
revolution NN I-NP O O
anniversary NN I-NP O O
celebrations NNS I-NP O O
. . O O O

I-PER

Hashan NNP I-NP I-PER I-PER
Tillekeratne NNP I-NP I-PER I-PER
, , O O O
Roshan NNP I-NP I-PER I-PER
Mahanama NNP I-NP I-PER I-PER
, , O O O
Kumara NNP I-NP I-PER I-PER
Dharmasena NNP I-NP I-PER I-LOC
, , O O O

Man NN I-NP O O
of IN I-PP O O
the DT I-NP O O
Match NNP I-NP O I-PER
: : O O O
Aravinda NNP I-NP I-PER I-PER
de NNP I-NP I-PER O
Silva NNP I-NP I-PER I-PER

Figure 8: Error Analysis on Dev set