Homework 3
Abhijit Prakash Bhatnagar

1. Please include your code! However, you will not be specifically graded on it and your responses to the questions below should be clear enough to understand what you did.

   See attached folder

2. (2pt) Define the feature templates you used for implementing the structured perceptron classifier. Propose 3 additional feature templates that (1) will likely be useful for improving NER performance and (2) are not part of your implementation.

   Feature set used:

Feature Name	Features
syntactic chunk word featurizer	s_i & w_i
syntactic chunk tag featurizer	s_i & ti
word tag featurizer	w_i & t_i
1 gram tag featurizer	t_i
1 gram syntactic chunk featurizer	s_i
2 gram tag featurizer	t_i & t_{i-1}
2 gram forward tag featurizer	t_i & t_{i+1}
2 gram syntactic chunk featurizer	s_i & s_{i-1}
2 gram forward syntactic chunk featurizer	s_i & s_{i+1}
3 gram tag featurizer	t_i & t_{i-1} & t_{i-2}
3 gram forward tag featurizer	t_i & t_{i+1} & t_{i+2}
3 gram around tag featurizer	t_i & t_{i-1} & t_{i+2}
3 gram syntactic chunk featurizer	s_i & s_{i-1} & s_{i-2}
3 gram forward syntactic chunk featurizer	s_i & s_{i+1} & s_{i+2}
3 gram around syntactic chunk featurizer	s_i & s_{i-1} & s_{i+2}
is number featurizer	w_i == 0-9 ?
is capitalized featurizer	w_i isCapital?
is split featurizer	w_i contains "–"?

   w_i is the word at i^{th} position. s_i is the syntactic chunk tag at i^{th} position. t_i is the POS tag at i^{th} position.

   Feature set that might improve performance and are not part of the implementation:

Feature Name	Features
Previous word and current tag	w_{i-1} & t_i
Previous 2 words and current tag	w_{i-2} & w_{i-1} & t_i
Next word and current tag	w_{i+1} & t_i

Next 2 words and current tag	w_{i+2} & w_{i+1} & t_i

(more in figure 3 in Michael Collins paper)

In my implementation, I mostly covered combinations of previous words, syntactic chunks and tags. I did not consider cross-combinations such as currect tag and previous chunks, etc that Michael collins proposed.

3. (2pt) Write out the Viterbi decoding algorithm for the structured perceptron using one recursive equation.

$$\pi(i, s_i) = \max_{s_{i-1}} w \cdot \phi(x, i, s_{i-1}, s_i) + \pi(i-1, s_{i-1})$$

4. (2pts) Ablation Study: In Ablation studies, we remove a subset of features (i.e. a subset of feature templates) to understand how those features affect performance. Design your own ablation study and report how the performance varies with respect to both the dev set and the test set. Try no more than 4 feature vector variations, including your full model with complete feature set.

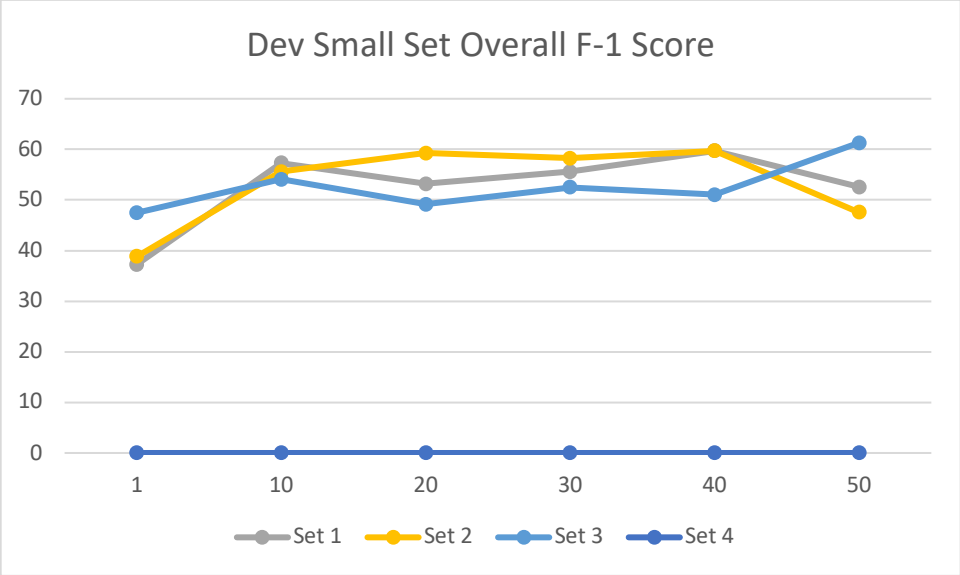
For this exercise, I used the dev-small and the test-small sets. Overall, I used 4 features sets.

Feature Name	Description	Set 1	Set 2	Set 3	Set 4
isnumber featurizer	w_i == [0-9] ?	✓	✓	✓	✓
iscapitalized featurizer	w_i isCapital?	✓	✓	✓	✓
issplit featurizer	w_i contains "–"?	✓	✓	✓	✓
syntactic chunk word featurizer	s_i & w_i	✓	✓	✓	
syntactic chunk tag featurizer	s_i & ti	✓	✓	✓	
word tag featurizer	w_i & t_i	✓	✓	✓	
1 gram tag featurizer	t_i	✓	✓	✓	
1 gram syntactic chunk featurizer	s_i	✓	✓	✓	
2 gram tag featurizer	t_i & t_{i-1}	✓	✓		
2 gram forward tag featurizer	t_i & t_{i+1}	✓	✓		
2 gram syntactic chunk featurizer	s_i & s_{i-1}	✓	✓		
2 gram forward syntactic chunk featurizer	s_i & s_{i+1}	✓	✓		
3 gram tag featurizer	t_i & t_{i-1} & t_{i-2}	✓			
3 gram forward tag featurizer	t_i & t_{i+1} & t_{i+2}	✓			

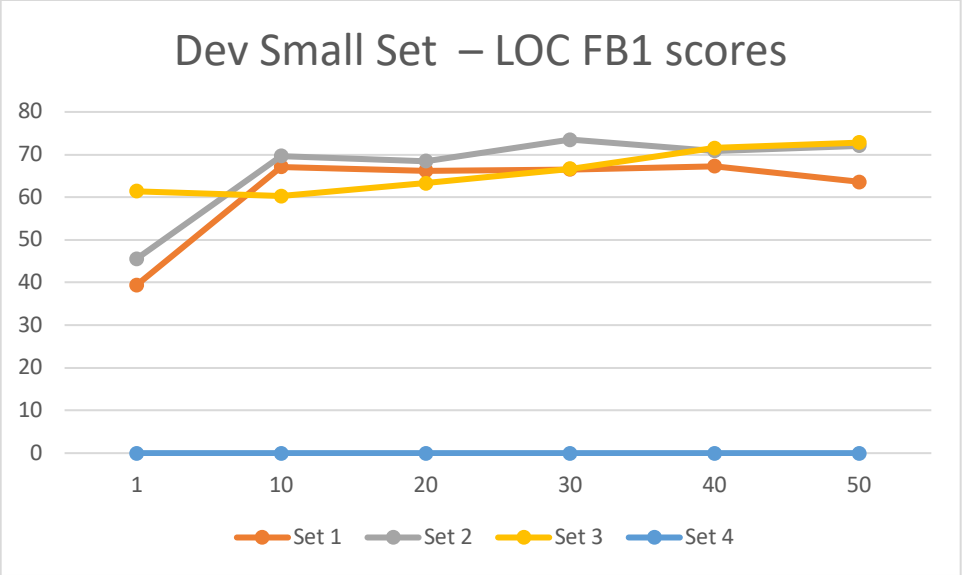
3 gram around tag featurizer	t_i & t_{i-1} & t_{i+2}	✓			
3 gram syntactic chunk featurizer	s_i & s_{i-1} & s_{i-2}	✓			
3 gram forward syntactic chunk featurizer	s_i & s_{i+1} & s_{i+2}	✓			
3 gram around syntactic chunk featurizer	s_i & s_{i-1} & s_{i+2}	✓			

w_i is the word at i^{th} position. s_i is the syntactic chunk tag at i^{th} position. t_i is the POS tag at i^{th} position.

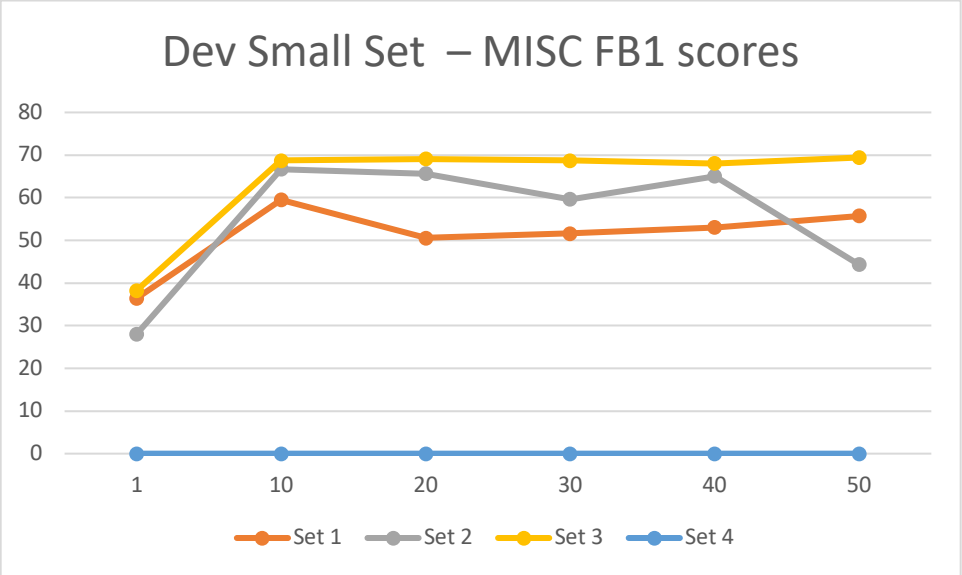
Performance Analysis using FB1 scores



Training Round	1	10	20	30	40	50
Set 1	37.25	57.33	53.18	55.61	59.68	52.56
Set 2	38.81	55.60	59.29	58.24	59.68	47.55
Set 3	47.47	54.08	49.11	52.49	51.06	61.31
Set 4	0	0	0	0	0	0

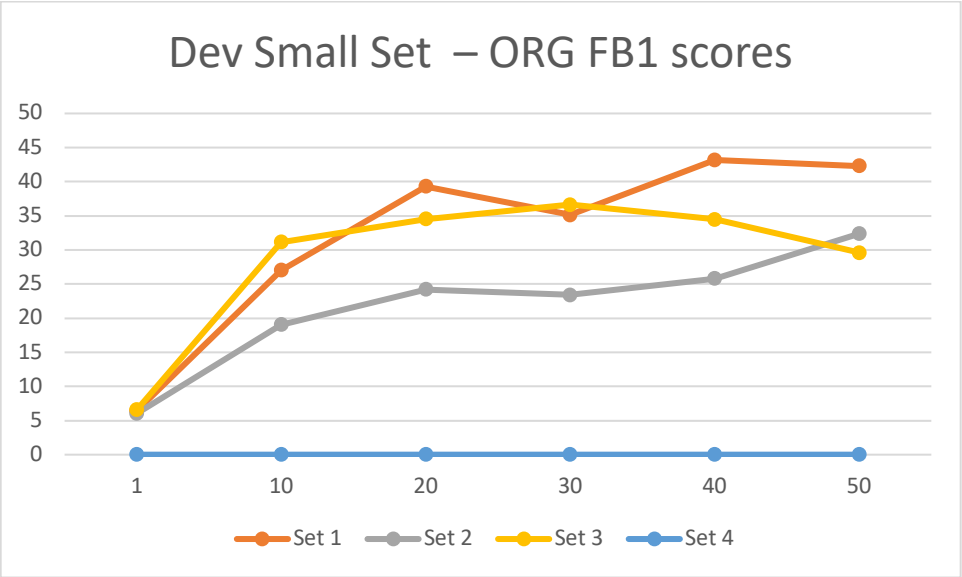


Training Round	1	10	20	30	40	50
Set 1	39.44	67.11	66.23	66.55	67.24	63.61
Set 2	45.60	69.66	68.44	73.52	70.88	72.06
Set 3	61.44	60.29	63.31	66.67	71.52	72.80
Set 4	0	0	0	0	0	0

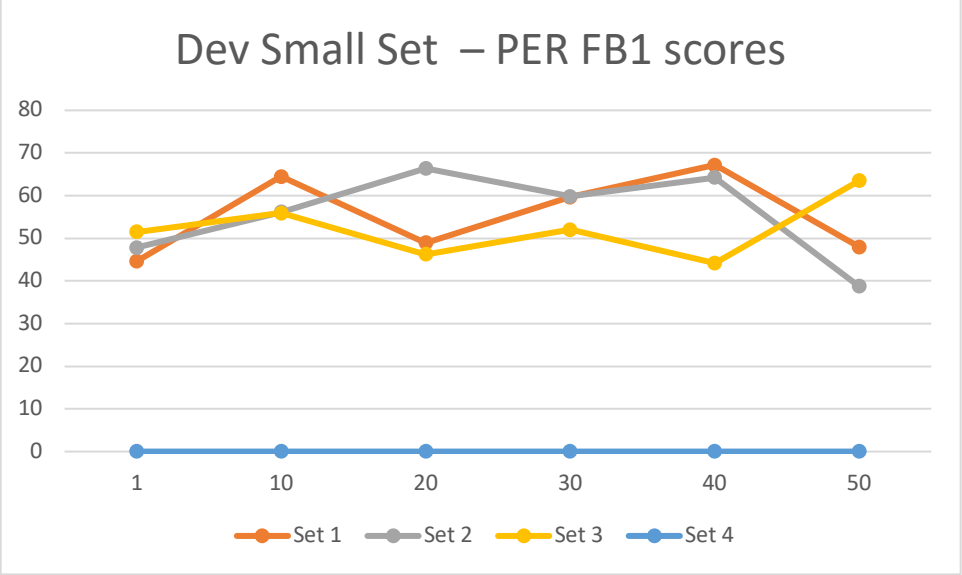


Training Round	1	10	20	30	40	50
Set 1	36.45	59.49	50.54	51.61	53.02	55.72
Set 2	28.07	66.67	65.62	59.66	65.02	44.31
Set 3	38.30	68.72	69.07	68.72	68.06	69.43

Set 4	0	0	0	0	0	0

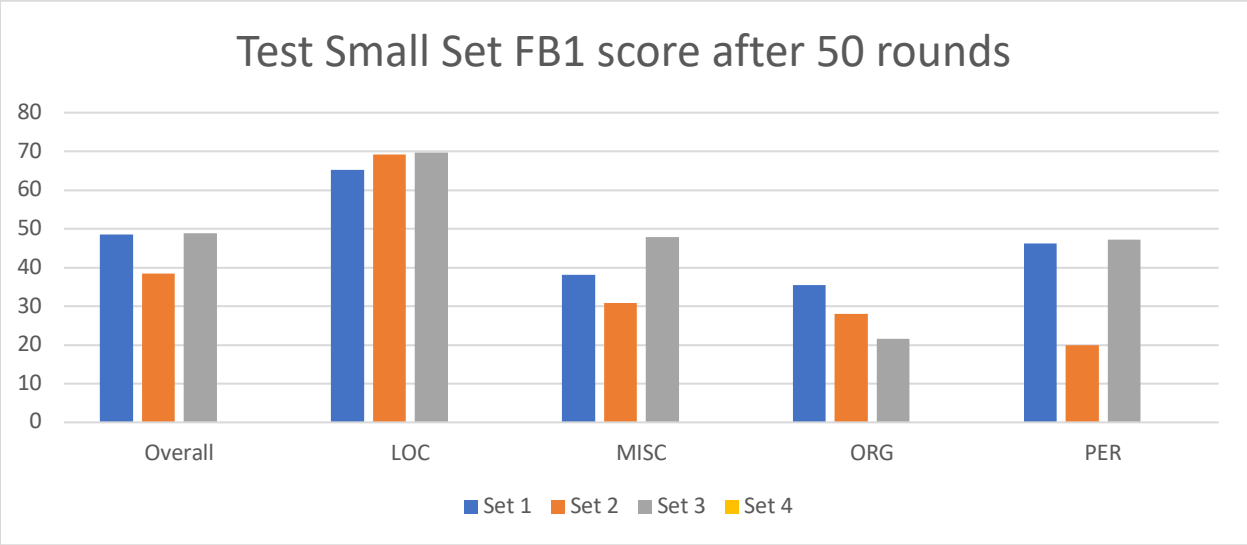


Training Round	1	10	20	30	40	50
Set 1	6.43	27.04	39.31	35.13	43.18	42.29
Set 2	6.04	19.05	24.22	23.40	25.81	32.37
Set 3	6.57	31.13	34.50	36.65	34.49	29.60
Set 4	0	0	0	0	0	0



Training Round	1	10	20	30	40	50
Set 1	44.63	64.50	48.96	59.70	67.21	47.98

Set 2	47.82	56.17	66.38	59.81	64.23	38.74
Set 3	51.47	55.91	46.27	52.04	44.18	63.49
Set 4	0	0	0	0	0	0



Training Round	Set 1	Set 2	Set 3	Set 4
Overall	48.46	38.51	48.80	0
LOC	65.25	69.26	69.62	0
MISC	38.14	30.85	47.78	0
ORG	35.46	27.99	21.62	0
PER	46.15	20.00	47.21	0

5. (2pt) While BIO encoding is the most common, there are a number of other encoding schemes (https://lingpipe-blog.com/2009/10/14/coding-chunkers-as-taggers-io-bio-bmewo-and-bmewo/) also used in practice. In fact, some studies have reported that IO encoding, which does not differentiate 'B' from 'I', often results in comparable (or even better) performance depending on the data and the task. Why might this be the case? Discuss the potential pros and cons of BIO and IO encoding schemes for NER.

    - The result in the referenced paper suggests that although IO tagging achieves the highest F1 score, its main limitation is the ability to recognize consecutive entities. BIO is useful if the intent is to be able to differentiate between 2 separate entities in the text.
    - BIO encoding worked better than IO encoding for Russian texts. This might be due to the structure of the Russian language in which the named entities are usually located next to each other. This also highlights the weakness of IO in case of consecutive named entities. Different languages may benefit from specific annotation schemes.

    Reference: https://www.sciencedirect.com/science/article/pii/S1110866520301596

6. (2pt) Error analysis and discussion of results. Including and discussing specific error cases you found will be useful here!
    1. I noticed that just the word-properties featurizer contributed did not contribute to any significant improvement of the accuracies, precision and FB1 scores.
    2. Most error cases predict O in-place of I-MISC
    3. Just predicting ”O” for every-tag (use only the word-properties feature set) results in 83.91% accuracy with 0 precision, recall and FB1 score.
    4. Seems different feature sets are able to identify different NERs more accurately than others. For example, Set 3 is able to identify LOC almost as well as other sets 1 and 2, but performs much worse for PER and ORG. Set 3 consistently works better than Sets 1 and 2 despite having the least number of features (only unigram and word properties)
    5. Trigram and bigram properties do not necessarily improve the FB1 scores.