Portfolio- Named Entity Recognition

1. Description of Named entity recognition:

Name entity recognition (NER) is a significant subtask in the area of NLP. They are widely used in information retrievals (IR). For example, with respect to the application of knowledge graph (KG), Google incorporates many high-quality open resources of multi-lingual knowledge graphs to measure different entity relations, such as product attributes, locations, events, and people. Another example is that Netflix built relevant tags for each clip by NER techniques. This approach enables smooth content discovery for the users.

2. Related work and the task design:

In this assignment, taking into consideration the chosen conll files might come with inconsistent annotations and features, in order to build an efficient NER evaluation system, we first inspect if the NER annotations of our training and testing conll file are consistent with each other and convert them into the same format.

Afterwards, we extend our features and extract 8 different features., including current, preceding, next tokens and pos tags, and chunk labels, as well as capitals.

In the “result of testing” section below, we build 8 different system based on these features applying 3 different classifiers: logistic, SVM, and naïve bayes. We also applied 2 advanced model in system 7 and 8: CRF and BERT.

From system 1 to 3, we keep all the above-mentioned traditional features and concatenate them as one-hot encoding using Dictvectorizer. In terms of the setting of classifiers, we utilized Logistic Regression, Support Vector Machine (SVM), and Naive Bayes (NB), in order. Here, we want to observe the distinctive results between these three classifiers.

Aside from using traditional features with sparse representation, from system 4 to 6, we used pre-trained word embedding model to concatenate the preceding and current tokens as dense representation to see if we can get a better result. Moreover, in system 6, we combine sparse representation

(traditional features) with dense representation (word embedding) to see if this mixed system performs better than the other 2 systems (system 4 and 5).

In system 7, we classify our data using advanced model: CRF. We first load the original training file and testing file without removing their BIO labels. We then create a header for the output file and predict its performance using pandas and our self-made evaluation function.

In system 8, we classify our data using another advanced model: BERT. We simply follow the given instruction without changing the default setting of parameter in the notebook. However, we narrow down our training data to the one-third of its original numbers. By doing so, we accelerate the running process from 16 hours (2 epochs) to 5 hours (2 epochs).

2.1 Preprocessing

In the preprocessing py file, we first load in training (conll2003 train) and testing file (conll2003 dev) and import “deque module” to rotate the current tokens and pos tags. By doing so, we extract the preceding and next tokens and pos tags. We also create an if-else statement to append "ENDSENTX" to the row that is empty. By doing so, we can avoid the index / position error when reading or writing in the file.

In terms of NER annotations scheme, we can discover that the input tags contain “PER”, “LOC”, “ORG”, “MISC” and “O”, followed by the “BIO” schema. We again use if-else statement to remove the prefix of NER labels. For instance, we converted "B-MISC" or “I-MISC” to "MISC" and append the preprocessed NER labels to the list called “simple_ner” Afterward, we apply zip function to write these additional features and revised NER labels to a new file, with a header. By doing so, we can extract the column that contains NER labels easily using pandas in the basic evaluation py file.

```
if BIOner[0] == 'B' or BIOner[0] == 'I':
    sn = BIOner[2:]
else:
    sn = BIOner
simple_ner.append(sn)
```

In terms of the setting of advanced model: CRF, we used the given instruction to load the original training and testing file, without removing their BIO labels. We then create a header for the output file. By doing so, we can extract the column that contains NER labels easily using pandas in the basic evaluation py file.

In terms of the setting of advanced model: BERT, we simply follow the given instruction without changing the default setting of the parameters in the notebook. However, we narrow down our training data to the one-third of its original numbers. By doing so, we accelerate the running process from 16 hours (2 epochs) to 5 hours (2 epochs).

2.2 Feature extraction

In the feature generation task, we extracted 8 different features as followed. In this section, we simply describe the reason of using these traditional features as the representation. We'll further analyze the detail and provide more examples in the "Ablation analysis" and "error analysis" sections.

1. token itself
2. previous token
3. next token
4. pos tag itself
5. previous pos tag
6. next pos tag
7. Capital
8. Chunk labels

Current, preceding, and next tokens:

Tokens themselves are commonly used in identifying the NERs. For instance, tokens, such as city, commission, department, and federation can help machine identify the NER annotations.

Current, preceding, next Pos Tags and Chunk labels

We extracted a combination of the preceding, current and the next POS tags to identify the syntactic class of the words, such as adjective, verb, and noun. This approach helps us figure out if a word belongs to a named entity or not, because the probability of a named entity being noun is high.

Capital

We use True or False to represent the word that has an initial capital letter. The reason of extracting this feature is because a word that contains NER annotation usually starts with a capital letter. (e.g., Tokyo - > LOC ; Tokyo Federal -> ORG)

2.3 Evaluation setup

Followed by the preprocessing steps and feature generation, in the basic evaluation py file, we used for-loop and default dict to calculate the overall TP, FP, and FN for each gold-system pair. For example, in the preprocessed output file, we have two columns that contain gold annotations (conll2003 dev) and predict annotations.

For each gold-predict pair in the preprocessed output file, if the system (predict) labels is equivalent to the gold labels, then the gold annotation obtains one "TP". If the system labels did not match the gold labels, then the gold labels got one "FN". If the corresponding labels appear in other class (e.g., for gold: 'O': system "O" appears in gold "ORG"), then the gold annotation obtains one "FP"

3. The results of the testing :

The illustrations below are the testing result of 6 systems adopting the following features: [token, preceding token, next token, pos, preceding pos, next pos, chunk label, capital]

System 1: features: [token, preceding token, next token, pos, preceding pos, next pos, chunk label, capital] , classifier: Logistic

System 2: features: [token, preceding token, next token, pos, preceding pos, next pos, chunk label, capital], classifier: SVM

System 3: features: [token, preceding token, next token, pos, preceding pos, next pos, chunk label, capital], classifier: Naïve Bayes

System 4: features: [word embedding current tokens], classifier: Logistic

System 5: features: [word embedding current, preceding tokens], classifier: Logistic

System 6: features: [word embedding current, preceding tokens, traditional features: pos tags, capital, chunk label], classifier: Logistic

System 7: features: [tokens, pos tags, chunk labels], classifier: CRF

System 8: features: [tokens, pos tags, chunk labels], classifier: BERT

```
              precision    recall        f1
system1 O      0.990232  0.995914  0.993065
        ORG    0.874107  0.760038  0.813091
        LOC    0.884498  0.833811  0.858407
        MISC   0.908425  0.782334  0.840678
        PER    0.854661  0.937440  0.894139
\begin{tabular}{llrrr}
\toprule
        &     &  precision &    recall &        f1 \\
\midrule
system1 & O &   0.990232 &  0.995914 &  0.993065 \\
        & ORG &   0.874107 &  0.760038 &  0.813091 \\
        & LOC &   0.884498 &  0.833811 &  0.858407 \\
        & MISC &   0.908425 &  0.782334 &  0.840678 \\
        & PER &   0.854661 &  0.937440 &  0.894139 \\
\bottomrule
\end{tabular}

0.874106652006597
{'precision': 0.9023846375327228, 'recall': 0.861907562851955, 'f1_score': 0.8798760181632614}


              precision    recall        f1
system2 O      0.991907  0.996327  0.994112
        ORG    0.879392  0.801625  0.838710
        LOC    0.901549  0.861509  0.881074
        MISC   0.911426  0.811514  0.858573
        PER    0.889319  0.949190  0.918280
\begin{tabular}{llrrr}
\toprule
        &     &  precision &    recall &        f1 \\
\midrule
system2 & O &   0.991907 &  0.996327 &  0.994112 \\
        & ORG &   0.879392 &  0.801625 &  0.838710 \\
        & LOC &   0.901549 &  0.861509 &  0.881074 \\
        & MISC &   0.911426 &  0.811514 &  0.858573 \\
        & PER &   0.889319 &  0.949190 &  0.918280 \\
\bottomrule
\end{tabular}

0.9919072142640758
{'precision': 0.9147185700585487, 'recall': 0.8840331065749913, 'f1_score': 0.8981498085513474}



              precision    recall        f1
system3 O      0.988831  0.975613  0.982178
        ORG    0.682684  0.719885  0.700791
        LOC    0.680328  0.832378  0.748711
        MISC   0.874580  0.615931  0.722814
        PER    0.807141  0.911718  0.856248
\begin{tabular}{llrrr}
\toprule
        &     &  precision &    recall &        f1 \\
\midrule
system3 & O &   0.988831 &  0.975613 &  0.982178 \\
        & ORG &   0.682684 &  0.719885 &  0.700791 \\
        & LOC &   0.680328 &  0.832378 &  0.748711 \\
        & MISC &   0.874580 &  0.615931 &  0.722814 \\
        & PER &   0.807141 &  0.911718 &  0.856248 \\
\bottomrule
\end{tabular}

0.9888311230559105
{'precision': 0.8067126996672446, 'recall': 0.811105114551585, 'f1_score': 0.8021483763879568}
```

```
              precision    recall        f1
system4 O      0.977088  0.995479  0.986198
        ORG    0.738436  0.641013  0.686285
        LOC    0.829897  0.768863  0.798215
        MISC   0.834176  0.650631  0.731059
        PER    0.905080  0.859956  0.881941
\begin{tabular}{llrrr}
\toprule
        &     &  precision &    recall &        f1 \\
\midrule
system4 & O &   0.977088 &  0.995479 &  0.986198 \\
        & ORG &   0.738436 &  0.641013 &  0.686285 \\
        & LOC &   0.829897 &  0.768863 &  0.798215 \\
        & MISC &   0.834176 &  0.650631 &  0.731059 \\
        & PER &   0.905080 &  0.859956 &  0.881941 \\
\bottomrule
\end{tabular}

0.977088
{'precision': 0.8569354359512852, 'recall': 0.7831884810534534, 'f1_score': 0.8167395066920138}
```

```
              precision    recall        f1
system5 O      0.980036  0.995501  0.987708
        ORG    0.777295  0.684034  0.727689
        LOC    0.860759  0.811843  0.835586
        MISC   0.849950  0.665615  0.746572
        PER    0.927640  0.903779  0.915554
\begin{tabular}{llrrr}
\toprule
        &     &  precision &    recall &        f1 \\
\midrule
system5 & O &   0.980036 &  0.995501 &  0.987708 \\
        & ORG &   0.777295 &  0.684034 &  0.727689 \\
        & LOC &   0.860759 &  0.811843 &  0.835586 \\
        & MISC &   0.849950 &  0.665615 &  0.746572 \\
        & PER &   0.927640 &  0.903779 &  0.915554 \\
\bottomrule
\end{tabular}

0.9800363753075854
{'precision': 0.8791361242725053, 'recall': 0.8121545556968371, 'f1_score': 0.8426218929842275}
```

```
              precision    recall        f1
system6 O      0.989977  0.996066  0.993012
        ORG    0.790890  0.730402  0.759443
        LOC    0.862821  0.850048  0.856387
        MISC   0.855524  0.714511  0.778685
        PER    0.919780  0.953954  0.936555
\begin{tabular}{llrrr}
\toprule
        &     &  precision &    recall &        f1 \\
\midrule
system6 & O &   0.989977 &  0.996066 &  0.993012 \\
        & ORG &   0.790890 &  0.730402 &  0.759443 \\
        & LOC &   0.862821 &  0.850048 &  0.856387 \\
        & MISC &   0.855524 &  0.714511 &  0.778685 \\
        & PER &   0.919780 &  0.953954 &  0.936555 \\
\bottomrule
\end{tabular}

0.9899766698349607
{'precision': 0.8837983398845977, 'recall': 0.8489959898604967, 'f1_score': 0.8648164204464688}
```

```
       &       &  precision &    recall &        f1 \\
\midrule
system7 & O &   0.978607 &  0.992797 &  0.985651 \\
        & B-ORG &   0.804607 &  0.755406 &  0.779231 \\
        & B-LOC &   0.937500 &  0.841045 &  0.886657 \\
        & B-MISC &   0.926768 &  0.796095 &  0.856476 \\
        & I-MISC &   0.845283 &  0.647399 &  0.733224 \\
        & B-PER &   0.915835 &  0.797503 &  0.852583 \\
        & I-PER &   0.890244 &  0.949503 &  0.918919 \\
        & I-LOC &   0.621019 &  0.758755 &  0.683012 \\
        & I-ORG &   0.782178 &  0.736352 &  0.758573 \\
\bottomrule
\end{tabular}

0.9786071601466148
{'precision': 0.8557823696236343, 'recall': 0.8083171677366344, 'f1_score': 0.82825848623942}
```

Predicted	B-LOC	B-MISC	B-ORG	B-PER	I-LOC	I-MISC	I-ORG	I-PER	O
Gold									
B-LOC	1545	4	104	24	2	0	5	1	152
B-MISC	7	734	13	24	0	5	3	3	133
B-ORG	44	15	1013	46	1	1	20	3	198
B-PER	21	5	76	1469	1	5	13	8	244
I-LOC	4	0	1	0	195	1	26	19	11
I-MISC	1	18	0	5	4	224	11	26	57
I-ORG	18	1	17	4	17	5	553	45	91
I-PER	0	0	1	6	0	4	13	1241	42
O	8	15	34	26	94	20	63	48	42451

P: 0.8557823696236343 R: 0.8083171677366344 F1: 0.8282584862394202

	precision	recall	f1-score	support
LOC	0.93	0.91	0.92	615
MISC	0.75	0.78	0.77	218
ORG	0.79	0.89	0.84	384
PER	0.98	0.97	0.97	698
micro avg	0.89	0.91	0.90	1915
macro avg	0.86	0.89	0.87	1915
weighted avg	0.90	0.91	0.91	1915

```
INFO:root:  Test Loss: 0.07
INFO:root:  Precision: 89.34 || Recall: 91.49 || F1: 90.40
```

In the system 1 and system 2, we keep all the features mentioned above and used two different classifiers: logistic and SVM to test their recall and precision.

Considering SVM can find the best margin in high dimensional space and reduce risk of error on data by allowing misclassifications to happen (soft margin), our hypothesis here is that, system 2 (SVM) will perform slightly better than system 1 (logistic regression), which can find the decision boundary with different weights but are vulnerable to the overfitting (outliers).

We assume that both of the classifiers (logistic and SVM) will have better performance than Naïve bayes does, since naïve bayes assumes all the features are independent. In most cases, the set of predicators in our training and testing data are syntactically and semantically correlated. (e.g., Federal office [‘org’] v.s office [‘O’]) We’ll further explain this part in the “Ablation analysis” and “Error analysis”.

The result of system 1 to 3 successfully fits our hypothesis. System 2 (SVM) has the highest precision (0.91) and recall (0.88), which is slightly better than that of system 1 (logistic classifier with precision:0.90; recall:0.86). On the other hand, system 3 (Naïve bayes) ’s precision (0.81) and recall (0.81) are far behind. This gap might increase if we add more training data or combine more useful traditional features.

In the system 4, we used word embedding to represent tokens. In system 5, we represent current and preceding tokens using word embedding. In system 6, we concatenate dense representation (current + preceding word embedding tokens) with sparse representation (one hot encoding traditional features: capital, pos tags, chunk labels)

Our hypothesis here is that system 4 will have the lowest precision and recall, and system 6 will preforms slightly better than system 5, since system 6 combine more useful features to help machine identify the bias in our training data.

The result fits our assumptions, and we can also find another interesting result, where the score for “MISC”, “PER”, and “O” NER annotations almost remain the same in these three systems, whereas the score for “ORG” and “LOC” keeps increasing from system 4 to 6. In other words, “ORG” and “LOC” always get confused with one another, and therefore adding more features in system 5 (preceding tokens) and system 6 (capital + pos + chunk labels) can to some degree help machine solve this bias and increase their overall recall and precision.

If we look at our confile training data, we can find that bias (tokens) usually appears between the tokens that have either “ORG” or “LOC” labels. (e.g, London Newsroom (current + next token) -> ORG / London (current token) ->

LOC).We'll provide more similar examples and analysis in the "Ablation analysis" and "Error analysis".

In the system 7, the test result shows that the prevision and recall for our CRF model is 0.85 and 0.80. Compared to the other systems, this score is not ideal. However, we only apply original 3 features (tokens, pos tags, chunk labels) to this system. If we extend our features, we might get a better result. When we look at the confusion matric, it shows the similar result that we found in the previous systems, where "B-LOC", "B-ORG" and "B-PER" often get confused with one another.

Finally, in the system 8, we used BERT model to test the performance. Our hypothesis here is that BERT model will get the highest score, since transformer can read the entire texts sequentially and bidirectionally and use the masked LM (MLM) and next sentence prediction (NSP) technique to fine tune the features.

The result fits our assumptions that the recall (0.91) and precision (0.89) are surprisingly high. Actually, in order to accelerate the running process, I remove one third of the training data. As we know, having enough training data can yield higher accuracy in BERT. If we didn't remove the partial training data, we would definitely get a higher score, which will surpass all the other systems (1~7).

Feature Ablation Analysis:

In the ablation analysis, we investigated all the possible combination of features to see which feature contribute to the progress. We decided to present 6 experiments that have interesting comparable results. In the first experiment, we keep all the traditional features and test their initial performance using logistic regression. The result is surprisingly good and we therefore decided to remove different features in the following 5 experiments to explore which features contribute to our system most.

In the experiments 2-5, we remove different combinations of specific features and only adopt logistic classifier to test them. Our motivation toward this decision is that we want to see more difference in different experiments. (SVM can reduce the risk of the error on the data)

In the experiment 6, we use logistic as our classifier and word embedding to represent current and preceding tokens and concatenate them with 3 traditional features: capital, pos tags, and chunk labels, to see if combining dense representation with sparse representation can get a better result.

The overall traditional features we use: [token, preceding token, next token, pos, preceding pos, next pos, chunk label, capital]

Experiment 1: keep all the features. classifier: logistic

Experiment 2: First we remove the current tokens and further remove the preceding and next tokens as well; classifier: logistic

Experiment 3: Remove current, preceding, next pos tags; classifier: logistic

Experiment 4: Remove current, preceding, next pos, chunk labels; classifier: logistic

Experiment 5: Remove current, preceding, next pos tags and capital features; classifier: logistic

Experiment 6: Represent current and preceding tokens using word embedding and concatenate them with 3 traditional features: capital, pos, chunk labels; classifier: logistic

Experiment 1

Predicted	LOC	MISC	O	ORG	PER
Gold					
LOC	1718	20	115	89	152
MISC	41	991	107	49	80
O	12	29	45822	48	98
ORG	110	41	141	1598	202
PER	40	12	90	42	2965

P: 0.9028443896335979 R: 0.8606703508705407 F1: 0.8792637604004702

In this experiment 1, we keep all the features to test the initial performance on recall and precision. The result is surprisingly good. We get 0.90 in precision and 0.86 in recall. The overall F1 score even outperforms that in rest of the experiments. We'll further remove different features in the following 5 experiments to explore which features contribute to our system most.

Experiment 2

The two illustrations shown below are the testing result of removing only tokens (illustration 1) and preceding and next tokens together (illustration 2), in order.

Predicted	LOC	MISC	O	ORG	PER
Gold					
LOC	1452	62	192	124	264
MISC	87	826	160	101	94
O	159	68	45473	127	182
ORG	237	70	193	1291	301
PER	170	27	122	59	2771

P: 0.7970880130787182 R: 0.7660508218531835 F1: 0.7781099852043185

Illustration 1

Predicted	LOC	MISC	O	ORG	PER
Gold					
LOC	880	43	362	217	592
MISC	61	483	181	154	389
O	241	90	45071	166	441
ORG	238	61	362	543	888
PER	241	28	253	180	2447

P: 0.6270421417454625 R: 0.5634816310706235 F1: 0.5757207007323716

Illustration 2

In the experiment 2, we chose logistic regression and first removed only current tokens. Our hypothesis is that if we remove the tokens, the overall score will drop, since tokens, such as city, commission, department, and federal, can help machine recognize named entities. The result fits our hypothesis. Compared to the first experiment, both precision (0.90 -> 0.79) and recall (0.86 ->0.76) drop hugely.

We further remove the preceding and next tokens additionally to see if the score will continue to drop. Our assumption here is that having previous and next tokens can further help machine identify difference between NER annotations (e.g, London Newsroom (current + next token) -> ORG / London (current token) -> LOC) and if we remove them, the score will continue to drop.

The result fits our assumption. The overall precision (0.79 -> 0.62) and recall (0.76 -> 0.56) again hugely drop.

Experiment 3

Predicted	LOC	MISC	O	ORG	PER
Gold					
LOC	1724	15	152	97	106
MISC	35	972	107	56	98
O	18	19	45819	47	106
ORG	114	44	150	1589	195
PER	46	9	115	42	2937

P: 0.90353818625459 R: 0.8555947674394826 F1: 0.8768719191090233

In the experiment 3, we removed current, preceding and next pos tags to test if the overall precision and recall will drop as seen in the experiment 2. Our hypothesis is that pos tags, such as NN, NNP, or NNPS, can help machine identify NER annotations, such as ORG, MISC, or People.

Surprisingly, the result does not fit our assumption. The precision and recall are almost the same as seen in the experiment 1, which might be due to the presence of chunk labels, as chunk tags like noun phrase (NP) can also help identify named entities. To test our assumptions, we decided to remove chunk labels additionally in the experiment 4.

Experiment 4

Predicted	LOC	MISC	O	ORG	PER
Gold					
LOC	1720	15	148	98	113
MISC	34	973	108	56	97
O	11	19	45816	47	116
ORG	107	45	160	1589	191
PER	43	9	123	40	2934

P: 0.9044196815409226 R: 0.8551668746059689 F1: 0.8770175386532806

Interestingly, the result in the experiment 4 does not fit our aforementioned assumption. While we removed pos tags and chunk labels, the result for the precision and recall still remain the same as seen in experiment 1 (keep all the features) and 3 (remove previous, current, and next pos tags). This might be due to the presence of capital features, as words that have named entities in conll2003 dataset are usually capitalized. (e.g., Tokyo - > LOC; Tokyo Commodities Desk -> ORG)

Again, we decided to remove capital features and pos tags but keep the chunk labels to test our hypothesis.

Experiment 5

Predicted	LOC	MISC	O	ORG	PER
Gold					
LOC	1554	10	438	61	31
MISC	19	901	320	11	17
O	3	10	45914	38	44
ORG	62	28	555	1395	52
PER	12	2	808	8	2319

P: 0.9417261219510442 R: 0.7707747238543561 F1: 0.8438021437724041

In the experiment 5, we see that the result of removing capital features fits our assumption. Recall drops from 0.85 to 0.77. Interestingly, the precision score remains high and even surpass the result of the other previous experiments. We'll further dive into the question by analyzing the confusion matrix in the "error analysis" section.

Experiment 6

The illustrations shown below are the result of using word embedding to represent tokens (illustration 1), current and preceding tokens (illustration 2), and the combination of dense and sparse representation (illustration 3).

Predicted	LOC	MISC	O	ORG	PER
Gold					
LOC	1642	27	113	250	62
MISC	62	859	232	79	36
O	17	58	45794	125	15
ORG	213	66	316	1365	132
PER	53	17	315	67	2697

P: 0.856479475663009 R: 0.793173000309751 F1: 0.8224590065000175

Predicted	LOC	MISC	O	ORG	PER
Gold					
LOC	1766	26	103	161	38
MISC	71	883	210	82	22
O	17	59	45779	119	35
ORG	179	66	300	1456	91
PER	50	13	209	54	2823

P: 0.8779003504244741 R: 0.8254389959059492 F1: 0.8497084174848633

Predicted	LOC	MISC	O	ORG	PER
Gold					
LOC	1784	25	53	171	61
MISC	60	924	137	108	39
O	16	60	45816	94	23
ORG	171	74	188	1556	103
PER	44	13	60	60	2972

P: 0.8809977549419996 R: 0.8528094598171204 F1: 0.8659582210113369

Our hypothesis in the experiment 6 is that the model combining dense representation and sparse representation (traditional features) will perform better than the model using only tokens as its representation. Considering our pre-trained embedding model might contain bias information, adding the traditional linguistic features, such as pos tags and chunk labels, might improve the score. The result fits our hypothesis. The recall (0.79 - > 0.85) and precision (0.85 -> 0.88) increase, compared to the scores in illustration 1 and 2.

Error Analysis:

In this section, we'll further analyze the changes of recall and precision by looking at the confusion matric across some experiments as seen in the "ablation analysis".

Error Analysis 1:

Predicted	LOC	MISC	O	ORG	PER
Gold					
LOC	1452	62	192	124	264
MISC	87	826	160	101	94
O	159	68	45473	127	182
ORG	237	70	193	1291	301
PER	170	27	122	59	2771

P: 0.7970880130787182 R: 0.7660508218531835 F1: 0.7781099852043185

Illustration 1: remove current token.

Features we keep: [preceding token, next token, pos, preceding pos, next pos, chunk label, capital]

Predicted	LOC	MISC	O	ORG	PER
Gold					
LOC	880	43	362	217	592
MISC	61	483	181	154	389
O	241	90	45071	166	441
ORG	238	61	362	543	888
PER	241	28	253	180	2447

P: 0.6270421417454625 R: 0.5634816310706235 F1: 0.5757207007323716

Illustration 2: remove current, preceding, and next token

Features we keep: [pos, preceding pos, next pos, chunk label, capital]

In the experiment 2, we see that the result fits our hypothesis, where removing previous and next tokens will harm recall and precision, because these two features can help machine identify difference between NER annotations.

When we look at the confusion matrix in the illustration 1, it shows that location (LOC) always gets confused with “O”, organization (ORG), and People (PER), and this situation even gets worse in the illustration 2.

That is because country name, such as “London” or “Israel” might bring out different meaning when looking at its preceding or next tokens together.

For example, when we look at the NER annotations in confile 2003, we find that ‘London Newsroom’ (ORG) and ‘London’ (LOC) repeatedly appear in many rows. If we only look at the current token, the machine will recognize “London” as organization and location in different rows. (e.g., London Newsroom (current + next token) -> ORG / London (current token) -> LOC).

The following four examples could further explain why ‘LOC’ will get confused with ‘ORG’, ‘PER’ and ‘O’, when we remove preceding and next tokens:

1. Reuter’s Television -> ORG; Television -> “O”
2. Toronto Dominion - > PER; Toronto -> LOC
3. Taiwan -> LOC; Relations across the Taiwan strait -> ORG
4. Federal office -> ORG; office -> ‘O’

Error analysis 2:

Predicted	LOC	MISC	O	ORG	PER
Gold					
LOC	1724	15	152	97	106
MISC	35	972	107	56	98
O	18	19	45819	47	106
ORG	114	44	150	1589	195
PER	46	9	115	42	2937

P: 0.90353818625459 R: 0.8555947674394826 F1: 0.8768719191090233

Illustration 1: remove current, preceding, and next pos tags

Features we keep: [Current token, preceding token, next token, chunk label, capital]

Predicted	LOC	MISC	O	ORG	PER
Gold					
LOC	1554	10	438	61	31
MISC	19	901	320	11	17
O	3	10	45914	38	44
ORG	62	28	555	1395	52
PER	12	2	808	8	2319

P: 0.9417261219510442 R: 0.7707747238543561 F1: 0.8438021437724041

Illustration 2: remove capitals

Features we keep: [Current token, preceding token, next token, pos, chunk label]

In the experiment 3, we remove preceding, current, and next pos tags. The result did not fit our assumption, where the precision and recall are almost the same as seen in the experiment 1.

We assumed that the presence of capital features, to some degree, helps machine recognize NER annotations without pos tags. (e.g., General Motor (NNP + NNPS + Capitals) -> ORG / General Motor (Adjective + NNP) -> ‘O’) The first “General Motor” refers to a real organization, whereas the second “General Motor” is just a capitalized word without NER labels that presents in the beginning of the sentence.

Therefore, we further remove the capital features in the experiment 5. The result successfully fits our assumption, where recall drops from 0.85 to 0.77.

When we look at the confusion matrix between experiment 3 (illustration 1) and experiment 5 (illustration 2), we find that Location (LOC), Organization (ORG), and People (PER) always get confused with one another.

This is probably because the name of an organization (ORG) or a person (PER) might be attached to a place's name (LOC) or follow on a token that has "O" tags. (e.g., Federal office for motor vehicle -> organization / motor vehicle -> "O"). Another example could be that a country or place (LOC) is followed by a token that could be either NNP or adjective, under different contexts. (e.g., New Delhi NNP + NNP / adj + NNP).

Once we remove the pos tags and capital features together, machine could not distinguish if the preceding token is an adjective that describe a country or a noun that is part of country's name.

Discussion and Conclusion

In this assignment, we build 8 different systems with a combination of different features and classifiers. When we look at the overall test performance, BERT model (system 8) yields the best result (recall:0.91, precision:0.89). If we did not remove one-third of its training data, the result would be even higher.

When we look at the models adopting sparse representations (extensive traditional features), system 2 (SVM) yields the second-best outcome (recall:0.88, precision, 0.91), which demonstrates that SVM usually performs slightly better than logistic regression and way better than Naïve Bayes.

When we look at the models adopting dense representation, system 6 (dense representation + sparse representation) performs better than system 4 and system 5, which shows that adopting traditional features (e.g., pos tags, chunk labels, capitals) with word embedding tokens based on linguistic information, to some degree, can help improve recall and precision.

When we look at the confusion matric of different systems, "LOC", "ORG", "PER" usually get confused with one another. This could be due to the reasons as followed:

1. country name, such as "London" or "Israel" might bring out different meaning when looking at its preceding or next tokens together.

(e.g., London Newsroom (current + next token) -> ORG / London (current token) -> LOC).

2. Name of an organization (ORG) or a person (PER) might be attached to a place's name (LOC) or follow on a token that has "O" tags, and presence of capital features, to some degree, helps machine recognize NER annotations without pos tags.

(e.g., General Motor (NNP + NNPS + Capitals) -> ORG / General Motor (Adjective + NNP) -> 'O')