CS 445 Project 4 Report

This project's aim is to recognize name entities. There are location, person and organizations labels for entities. Also for non-entities, we have *"O"* label (out). There are 2 datasets. First one contains every word and their features. Second dataset contains name entities and their labels.

```
[102] df_features
```

	Row	Words	Feature
0	1	Müzik	müzik+Noun+A3sg+Pnon+Nom
1	1	Şenliği'ne	Şenlik+Noun+Prop+A3sg+P3sg+Dat
2	1	hazırlanın	hazırla+Verb
3	1	POZİTİF	pozitif+Adj
4	1	ve	ve+Conj
...	...	...	...
143702	9999	bir	bir+Det
143703	9999	ölçüde	ölçü+Noun+A3sg+Pnon+Loc
143704	9999	prim	prim+Noun+A3sg+Pnon+Nom
143705	9999	kazandırdı	kazan+Verb
143706	10000	Çünkü	çünkü+Conj

143707 rows × 4 columns

```
#create dataframe
df_labels = pd.DataFrame(IOLabels, columns=["Row", "Label", "Words"])
df_labels
```

	Row	Label	Words
0	1	I-ORG	POZİTİF
1	1	I-ORG	ve
2	1	I-ORG	Açık
3	1	I-ORG	Radyo
4	1	I-LOC	İstanbul
...	...	...	...
16868	9995	I-ORG	DSP
16869	9996	I-ORG	DSP
16870	9997	I-ORG	DSP
16871	9998	I-ORG	DSP
16872	9999	I-ORG	DSP

16873 rows × 3 columns

To create train set, we

need to bring labels and features together. In order not to loose non-entity words, datasets are merged according to *df_features* dataset. Then *nan* label values converted to *"O"* notation.

```
[104] #Merge dataframes witf based on df_features dataset
      merged_df = df_features.merge(df_labels, on = ["Row", "Words"], how="left")
```

```
[105] merged_df["Label"].fillna("O", inplace=True)
```

```
[106] merged_df
```

	Row	Words	Feature	Features2	Label
0	1	Müzik	müzik+Noun+A3sg+Pnon+Nom	[müzik, Noun, A3sg, Pnon, Nom]	O
1	1	Şenliği'ne	Şenlik+Noun+Prop+A3sg+P3sg+Dat	[Şenlik, Noun, Prop, A3sg, P3sg, Dat]	O
2	1	hazırlanın	hazırla+Verb	[hazırla, Verb]	O
3	1	POZİTİF	pozitif+Adj	[pozitif, Adj]	I-ORG
4	1	ve	ve+Conj	[ve, Conj]	I-ORG
...	...	...	...	...	...
145159	9999	bir	bir+Det	[bir, Det]	O
145160	9999	ölçüde	ölçü+Noun+A3sg+Pnon+Loc	[ölçü, Noun, A3sg, Pnon, Loc]	O
145161	9999	prim	prim+Noun+A3sg+Pnon+Nom	[prim, Noun, A3sg, Pnon, Nom]	O
145162	9999	kazandırdı	kazan+Verb	[kazan, Verb]	O
145163	10000	Çünkü	çünkü+Conj	[çünkü, Conj]	O

145164 rows × 5 columns

Next, following features are created.

{'inf': ['A3sg', 'Pnon', 'Nom'],
 'nom': True,
 'pnon': True,
 'postag': 'Noun',
 'word.isdigit': False,
 'word.istitle': True,
 'word.isupper': False,
 'word.lower': 'müzik'}

İnf is all inflecionals features that is taken from dataset.
Nom is noun case. If the Word is a noun, then it returns true.
Pnon is proper noun. If the Word is a proper noun, then it returns true.
Postag is type of the word.
Word.isdigit is true if the Word is a number.
Word.istitle is true if the Word begins with upper case letter.

Word.isupper is true if the Word is consisted of only upper case letters.
Word.lower is lower case of the Word.

In order to obtain train set, we need to create list of lists of dictionaries. Every dictionary represents a word in the text, and they contain features of the words. Also, every list represents sentences.

[[{feature1: xxx, feature2: True, …}, {}, {}, {}], [{}, {}, {}, {}], [{}, {}, {}, {}], …]

For training, there are 5 CRF model that are created. Each has combination of 4 folds of 5 folds for training set. Also they have a test set which is remaining from other 4.

After the training, the evaluation results are obtained from remaining folds. Accuracy is an important measure for ML models but for our dataset, but most of the labels are "O". Therefore, accuracy score mostly gives high scores because of high base line accuracy. However, entity labels (I-Per, I-Org, I-Loc) are more important rather than "O" labels. In this case, F1 score is a better measure of our model. It is harmonic mean of precision and recall. Because we can exclude "O" labels. Thus, F1 score shows that how accurately we estimate entity labels (I-Per, I-Org, I-Loc).

Accuracy Score
Avg → 0.97571754356771

F1 Scores
CRF1 → 0.8485571904592357
CRF2 → 0.8402884363601334
CRF3 → 0.8467095943830918
CRF4 → 0.8105135631724366
CRF5 → 0.845174444670488

Average → 0.8382486458090771

From classification reports I-ORG has lowest scores among three labels' precision and recall scores. It is between 0.7 and 0.8. However, I-loc and I-per range between 0.85 and 0.9.

Adding Gazetteer Feature
To support I-ORG, gazetteer from 1st project is added as feature. However, adding gazetteer did not create significant contribution. Precision and recall of I-ORG is increased a little. This is maybe because of small gazetteer dataset. Enlarging it may contribute more.

Adding Word Shape Feature

Upper case → X
Lower case → x
Digits → d
Adding word shape feature increased average f1 score from 0.838 to 0.843

Using IOB labelling
Initially IO labelling is used for CRF models. Finally, IOB labelling is tried with same features. However, average f1 score decreased from 0.838 to 0.798 and average accuracy did not change remarkably. It shows that we choose right measure type, and our model cannot predict IOB labels as accurately as IO labelling. The reason behind that is dependency of IOB labels. The first entity begins with "B" and other following labels begin with "I". To support our model, we can add previous word features.

Adding Previous word features
Prev_label → previous word's label
Prev_istitle → is previous word title
Prev_isupper → is previous word upper case only
Prev_lower → Lower case of the previous word
Prev_postag → Previous word's postag

After adding features and training our model, average f1 score increased from 0.798 to 0.854. As before, average accuracy did not change significantly.

As a conclusion, we can see that adding features increase the accuracy. As stated in the NER lecture, average F-measure which is stated in Yeniterzi, Tür, Şeker GA papers was around %90. Our model reached 0.854 with IOB labelling and 0.886 with IO labelling which is not bad. Also, changing labelling format can affect f1 scores. With IOB tagging, our model could reach lower f1 score but name entities are represented better. On the other hand, choosing right evaluation measure is important to understand how well our model performs. In this way, f1 score without "O" labels gives a better measure to understand how accurately our model gives labels to name entities.

Appendix

```
[420] avg_f1_score = (crf1_f1_score + crf2_f1_score + crf3_f1_score + crf4_f1_score + crf5_f1_score)/5
      avg_f1_score

      0.8547129381666501

      avg_acc = (crf1_acc + crf2_acc + crf3_acc + crf4_acc + crf5_acc)/5
      avg_acc

      0.9799966175953212
```

```
[406] crf1_matrix = metrics.flat_classification_report(
          y_train_1, pred1, digits=3, labels=sorted_labels)
      print(crf1_matrix)
```

```
                  precision    recall  f1-score   support

           B-LOC      0.883     0.855     0.869       550
           I-LOC      0.702     0.688     0.695        48
           B-ORG      0.921     0.789     0.850       535
           I-ORG      0.858     0.852     0.855       283
           B-PER      0.907     0.842     0.874       920
           I-PER      0.823     0.864     0.843       339

       micro avg      0.884     0.835     0.859      2675
       macro avg      0.849     0.815     0.831      2675
    weighted avg      0.886     0.835     0.859      2675
```

```
[409] crf2_matrix = metrics.flat_classification_report(
          y_train_2, pred2, digits=3, labels=sorted_labels)
      print(crf2_matrix)
```

```
                  precision    recall  f1-score   support

           B-LOC      0.880     0.872     0.876       538
           I-LOC      0.720     0.581     0.643        31
           B-ORG      0.926     0.805     0.861       467
           I-ORG      0.878     0.902     0.890       255
           B-PER      0.904     0.816     0.858       934
           I-PER      0.804     0.859     0.830       368

       micro avg      0.882     0.837     0.859      2593
       macro avg      0.852     0.806     0.826      2593
    weighted avg      0.884     0.837     0.859      2593
```

```
crf3_matrix = metrics.flat_classification_report(
    y_train_3, pred3, digits=3, labels=sorted_labels)
print(crf3_matrix)
```

```
              precision    recall  f1-score   support

       B-LOC      0.913     0.886     0.899       606
       I-LOC      0.893     0.543     0.676        46
       B-ORG      0.931     0.837     0.881       497
       I-ORG      0.946     0.868     0.905       281
       B-PER      0.898     0.841     0.869       982
       I-PER      0.769     0.837     0.802       369

   micro avg      0.892     0.848     0.869      2781
   macro avg      0.891     0.802     0.839      2781
weighted avg      0.895     0.848     0.869      2781
```

```
crf4_matrix = metrics.flat_classification_report(
    y_train_4, pred4, digits=3, labels=sorted_labels)
print(crf4_matrix)
```

```
              precision    recall  f1-score   support

       B-LOC      0.895     0.802     0.846       646
       I-LOC      0.759     0.759     0.759        83
       B-ORG      0.839     0.761     0.798       548
       I-ORG      0.883     0.866     0.875       262
       B-PER      0.920     0.847     0.882      1011
       I-PER      0.790     0.832     0.811       358

   micro avg      0.874     0.818     0.845      2908
   macro avg      0.848     0.811     0.828      2908
weighted avg      0.875     0.818     0.845      2908
```

```
crf5_matrix = metrics.flat_classification_report(
    y_train_5, pred5, digits=3, labels=sorted_labels)
print(crf5_matrix)
```

```
              precision    recall  f1-score   support

       B-LOC      0.887     0.856     0.872       606
       I-LOC      0.841     0.757     0.797        70
       B-ORG      0.907     0.787     0.843       483
       I-ORG      0.824     0.838     0.831       241
       B-PER      0.898     0.801     0.847      1088
       I-PER      0.778     0.809     0.793       398

   micro avg      0.871     0.814     0.841      2886
   macro avg      0.856     0.808     0.830      2886
weighted avg      0.873     0.814     0.841      2886
```