Ian MacDonald

CIS 530

Homework 8 Writeup

Feature Engineering

Features Tried

Word

The word itself, which is the simplest information capturable, and is the default feature provided in the code outline.

First Letter Capitalized (FLC)

Boolean value, 1 if only the first letter of the word is capitalized, 0 otherwise. Used because many named entities will be proper nouns, and proper nouns in Spanish are capitalized.

All Letters Capitalized (ALC)

Boolean value, 1 if all of the letters in the word are capitalized, 0 otherwise. Used because all capitalized words are likely to be abbreviations, and often times places, people, and organizations are written in their abbreviated format.

Part of Speech (POS)

The part of speech of the word. Logic is some parts of speech may commonly be used before/after named entities, and some parts of speech may be more likely to be a named entity.

Has Common Prefix (HCF)

Boolean, value is 1 if the first four letters of the word match any one of the top 150 most common 4-letter prefixes (among words with at least 5 letters in the training set that are named entities), 0 otherwise. Logic is that some named entities may share common prefixes that denote names, places, etc.

Has Common Suffix (HCS)

Boolean, value is 1 if the first four letters of the word match any one of the top 150 most common 4-letter suffixes (among words with at least 5 letters in the training set that are named entities), 0 otherwise. Logic is that some named entities may share common suffixes that denote names, places, etc.

Word Length (WL)

Length of the word. Different word lengths may be more or less likely to be a named entity.

Is Word a Single Character (SC)

Boolean, 1 if word is a single character, i.e. has a length of 1, otherwise 0. More general than using word length, as it only differentiates between word length of 1 and everything else. Logic here is that single character words are significantly less likely to be a named entity compared to most other word lengths. Only 1.15% of the length 1 training words are named entities, as compared to 5.28% for those of length 3, for example, with most other lengths having similar percentages.

Has Numbers (HN)

Boolean, 1 if the word contains any numbers, 0 otherwise. The presence of a number may affect likelihood of a word being a named entity.

All Numbers (AN)

Boolean, 1 if the word is a string of all numbers, 0 otherwise. If the entire word is actually just numbers, may affect the likelihood of the word being a named entity.

Performance and Feature Selection

Each of the features described above was added to the list of features extracted from the set of training words, and then used to train the default model, a basic Perceptron. The features were extracted from a window size of three around the target word (previous word, target word, following word). The window size and model were unchanged for all runs, in order to isolate the effect adding each extra feature had on the model's performance. If the model's performance improved, the feature was kept, and used in all subsequent models. If the model's performance did not improve, the feature was discarded. F1 Score on the development set was used to judge the impact of added features, with the tables below detailing how each set of features tried affected the model's performance.

Note: Including the detailed performance on both the training and development sets in one table made the table a bit cluttered, so they are broken up into two tables here. The first has a detailed look at the performance on the development set, with a single column showing the corresponding overall training score, and is the more useful table of the two. However, the mirrored version of the table showing detailed performance on the training set is also included.

Performance (Development Set Detailed)

Features	F1 Development					F1 Train
	LOC	MISC	ORG	PER	OVR	OVR
Word	63.68	28.54	53.13	51.93	52.53	**82.70**
Word, FLC	56.94	26.83	53.07	65.35	54.62	81.30
Word, FLC, ALC	63.72	27.62	59.02	**74.30**	61.12	82.05
Word, FLC, ALC, POS	61.28	**32.46**	58.12	71.73	59.74	81.79
Word, FLC, ALC, HCP	**63.83**	22.13	53.74	71.06	56.70	79.17
Word, FLC, ALC, HCS	63.41	28.67	**59.64**	71.38	**61.25**	82.51
Word, FLC, ALC, HCS, WL	62.01	26.32	51.98	66.65	57.67	71.59
Word, FLC, ALC, HCS, SC	52.71	23.92	47.29	57.25	49.15	74.61
Word, FLC, ALC, HCS, HN	62.60	27.37	56.44	70.80	58.92	81.19
Word, FLC, ALC, HCS, AN	62.38	27.13	53.47	66.96	56.70	78.45

Performance (Training Set Detailed)

Features	F1 Train					F1 Dev
	LOC	MISC	ORG	PER	OVR	OVR
Word	81.40	66.19	82.07	94.70	**82.70**	52.53
Word, FLC	79.91	60.48	80.58	**95.59**	81.30	54.62
Word, FLC, ALC	81.83	65.50	80.54	94.05	82.05	61.12
Word, FLC, ALC, POS	80.80	**66.47**	80.33	94.07	81.79	59.74
Word, FLC, ALC, HCP	82.67	57.86	76.33	92.96	79.17	56.70

Word, FLC, ALC, HCS	**83.21**	64.45	**80.85**	93.84	82.51	**61.25**
Word, FLC, ALC, HCS, WL	78.23	35.51	69.87	80.34	71.59	57.67
Word, FLC, ALC, HCS, SC	76.43	59.56	71.02	86.04	74.61	49.15
Word, FLC, ALC, HCS, HN	82.62	63.81	79.98	90.81	81.19	58.92
Word, FLC, ALC, HCS, AN	78.46	54.45	78.55	90.15	78.45	56.70

Final Feature Selection

As seen in the table above, a lot of the features that were tested actually ended up having a negative effect on the model's performance and were subsequently discarded. The first two added features, First Letter Capitalized and All Letters Capitalized helped improve the model's performance greatly, as expected, and were thus kept for subsequent runs. While Part of Speech a little surprisingly did not help at all, it did not bring the model's performance down too badly, but was removed nonetheless. The Has Common Prefix feature did not help, but the Has Common Suffix feature did, albeit a marginal increase. However, it was still an increase, so it was kept. The word length and numbers features all had varying negative effects on the model's performance, and all were discarded. That left the final features selected as Word, First Letter Capitalized, All Letters Capitalized, and Has Common Suffix. These features achieved a development F1 Score of 61.25 on the development set with the default model, the highest score for any feature combination.

Window Size Selection

After selecting features, the next step was deciding how large the window around the target word should be, as well as how the window should be situated. Below is a table showing the results of different window sizes (how many words around the target word are taken into account) and different window positionings (which words around the target word are taken into account, for example – a window size of 3 could be target word, previous word, and previous word of the previous word, or target word, following word, and following word of the following word) and their impact on model performance. For each window configuration, the feature combination noted above was used, as well as the default Perceptron model.

Note: Size column indicates window size, and positioning denotes which words in relation to the target word are taken into account (0 being the target word, -1 indicating the previous word, 1 indicating the following word, etc.). Additionally, two tables are provided, one for development one for training, for the same reasons provided above.

Windows Tried/Performance (Development Set Detailed)

Window Configuration		F1 Development					F1 Train
Size	Positioning	LOC	MISC	ORG	PER	OVR	OVR
3	-1, 0, 1	63.41	28.67	59.64	71.38	61.25	82.51
5	-2, -1, 0, 1, 2	62.89	30.66	64.01	70.24	61.66	91.27
7	-3, -2, -1, 0, 1, 2, 3	67.93	**38.19**	**65.04**	76.18	**66.16**	97.03
9	-4, -3, -2, -1, 0, 1, 2, 3, 4	**69.02**	35.26	64.10	75.66	65.53	97.82
11	-5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5	67.45	36.56	63.74	73.64	64.80	**98.35**
7	-4, -3, -2, -1, 0, 1, 2	61.65	33.82	64.16	70.88	61.87	96.51

7	-2, -1, 0, 1, 2, 3, 4	66.64	33.99	63.53	76.23	64.61	96.47
8	-4, -3, -2, -1, 0, 1, 2, 3	69.17	33.52	64.35	**76.61**	65.90	96.79
8	-3, -2, -1, 0, 1, 2, 3, 4	68.86	33.65	64.40	75.36	65.12	97.12
6	-3, -2, -1, 0, 1, 2	65.33	35.99	64.52	75.90	65.13	94.91
6	-2, -1, 0, 1, 2, 3	67.34	36.56	64.81	75.43	65.69	95.37

Windows Tried/Performance (Training Set Detailed)

Window Configuration		F1 Train					F1 Dev
Size	Positioning	LOC	MISC	ORG	PER	OVR	OVR
3	-1, 0, 1	83.21	64.45	80.85	93.84	82.51	61.25
5	-2, -1, 0, 1, 2	92.23	71.03	93.38	97.29	91.27	61.66
7	-3, -2, -1, 0, 1, 2, 3	97.22	91.74	97.15	99.31	97.01	**66.16**
9	-4, -3, -2, -1, 0, 1, 2, 3, 4	97.94	94.25	98.01	99.21	97.82	65.53
11	-5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5	**98.39**	**96.39**	**98.25**	**99.47**	**98.35**	64.80
7	-4, -3, -2, -1, 0, 1, 2	96.04	93.73	96.63	98.29	96.51	61.87
7	-2, -1, 0, 1, 2, 3, 4	96.39	92.41	96.35	98.83	96.47	64.61
8	-4, -3, -2, -1, 0, 1, 2, 3	97.74	88.33	97.30	99.15	96.79	65.90
8	-3, -2, -1, 0, 1, 2, 3, 4	96.67	93.85	97.63	98.41	97.12	65.12
6	-3, -2, -1, 0, 1, 2	94.08	90.97	94.48	98.59	94.41	65.13
6	-2, -1, 0, 1, 2, 3	95.17	87.12	96.07	98.71	95.37	65.69

Increasing the window size gradually while keeping it centered around the target word helped improve the model's performance up until window size 7, after which increasing the model size had a negative effect on performance. Additionally, sliding the window and increasing/decreasing it by one and then sliding all had varying negative effects on the model's performance. Therefore, a window of size 7 centered around the target word was chosen as the final window.

Model Selection

After selecting the features and window size, the final step was choosing a model. For each model type below, multiple different models were tested, each with different hyperparameters. The results below detail how each model performed. Each model was trained using the feature set and window sizes chosen above.

Logistic Regression

Logistic regression is a common model for classification problems, and was used to hopefully provide a better baseline value than that achieved by the basic Perceptron model, before moving on to slightly more complicated models. Various hyperparameters were tuned, including regularization strength (C), optimization algorithm (solver), and iterations (max_iter).

Performance (Development Set Detailed)

Hyperparameters	F1 Development					F1 Train
	LOC	MISC	ORG	PER	OVR	OVR
max_iter=500	67.73	35.66	67.15	75.91	66.74	92.63

max_iter=500, C=.5	66.34	31.54	65.73	74.05	65.00	86.76
max_iter=500, C=2	68.30	36.64	67.60	77.08	67.40	96.14
max_iter=500, C=3	68.63	36.96	67.95	77.47	67.73	97.54
max_iter=750, C=5	69.00	37.59	68.05	78.04	68.06	98.36
max_iter=1000, C=7	69.21	37.64	**68.11**	78.27	68.19	98.58
max_iter=1000, C=10	69.03	37.97	68.09	78.02	68.10	**98.72**
max_iter=1000, C=7, solver='liblinear'	**70.45**	**40.14**	68.04	**79.24**	**69.12**	97.83
max_iter=1000, C=7, solver='saga'	69.30	38.43	68.07	78.18	68.26	98.57

Performance (Training Set Detailed)

Hyperparameters	F1 Train					F1 Dev
	LOC	MISC	ORG	PER	OVR	OVR
max_iter=500	93.41	80.59	93.29	96.70	92.63	66.74
max_iter=500, C=.5	88.48	66.62	87.74	92.94	86.76	65.00
max_iter=500, C=2	96.53	89.81	96.23	98.73	96.14	67.40
max_iter=500, C=3	97.70	93.49	97.49	99.51	97.54	67.73
max_iter=750, C=5	98.38	96.01	98.23	**99.76**	98.36	68.06
max_iter=1000, C=7	**98.59**	96.55	98.47	**99.76**	**98.58**	68.19
max_iter=1000, C=10	98.74	**97.04**	**98.60**	**99.76**	98.72	68.10
max_iter=1000, C=7, solver='liblinear'	97.90	94.36	97.72	99.70	97.83	**69.12**
max_iter=1000, C=7, solver='saga'	**98.59**	96.49	98.47	**99.76**	98.57	68.26

Increasing the max iterations was necessary for the algorithm to converge, and was adjusted as needed while changing other hyperparameters. Decreasing the C value to 0.5 led to a decrease in model performance, likely because the regularization was too strong. For the model in general, it seemed that the regularization was too strong, as gradually increasing the C value continually led to better model performance, up until reaching a C value of 7, after which increasing the C value led to a very slight drop in performance. Changing the optimization algorithm to 'saga' led to a slight increase in model performance, but changing it to 'liblinear' had a more substantial performance increase. Overall, the best logistic regression model was max_iter=1000, C=7, solver='liblinear', with an overall F1 Score of 69.12 on the development set, beating the 66.16 benchmark of the basic Perceptron.

Random Forest

Random Forest is an ensemble classifier that takes advantage of the "bagging" method, where smaller samples of the training data are used to build a group of decision tree classifiers that are then collectively used to predict classes. In all of the previous models, the training accuracies were much higher than the development accuracies, and the Random Forest was used to hopefully try and combat some of that overfitting. The hyperparameters adjusted were the number of trees built (n_estimators), the splitting criterion (criterion), and the number of features used at each split (max_features)

Performance (Development Set Detailed)

Hyperparameters	F1 Development					F1 Train
	LOC	MISC	ORG	PER	OVR	OVR
n_estimators=30	**66.09**	**35.15**	63.79	68.39	63.09	**98.89**
n_estimators=40, max_features=.5	65.94	33.02	**66.96**	**71.49**	**64.71**	98.51
n_estimators=20, max_features=.5	65.20	32.79	66.69	70.15	64.01	97.26
n_estimators=40, max_features=.5, criterion='entropy'	64.89	29.60	66.63	70.26	63.54	98.54

Performance (Training Set Detailed)

Hyperparameters	F1 Train					F1 Dev
	LOC	MISC	ORG	PER	OVR	OVR
n_estimators=30	**98.90**	**97.48**	**98.80**	**99.75**	**98.89**	63.09
n_estimators=40, max_features=.5	98.53	96.56	98.51	99.45	98.51	**64.71**
n_estimators=20, max_features=.5	97.63	93.41	97.37	98.61	97.26	64.01
n_estimators=40, max_features=.5, criterion='entropy'	98.61	96.79	98.44	99.50	98.54	63.54

Ironically, the Random Forest models had some of the most overfitting of any of the models. Tuning the hyperparameters changed the accuracies a little bit, but they all performed worse than the default Perceptron and significantly worse than the best Logistic Regression model. Additionally, the Random Forest models ran extremely slow, which made tuning the hyperparameters a relatively cumbersome task. Given the slow running time and overall poor results from the first few models, it was unlikely that a Random Forest model was going to be found that outperformed the best model up to this point. Therefore, only 4 models were tried, with the best one being n_estimators=40, max_features=.5, and default criterion, with an F1 Score on the development set of 64.71.

XGBoost

While the results from the Random Forest models were poor, I still thought that an ensemble method could work, so I decided to give XGBoost a try. As the name implies, XGBoost takes advantage of the "boosting" method, where a series of weak classifiers are learned sequentially, with later classifiers putting more emphasis on previously misclassified values. These classifiers are then used for prediction, with the hope that they might generalize better than previous classifiers. The hyperparameters tuned were learning rate (eta), tree depth (max_depth), and number of estimators (n_estimators).

Performance (Development Set Detailed)

Hyperparameters	F1 Development					F1 Train
	LOC	MISC	ORG	PER	OVR	OVR
booster='tree'	62.05	23.73	64.34	68.43	61.15	73.31

booster='tree', n_estimators=200	64.98	28.16	64.68	70.64	62.94	77.28
booster='tree', n_estimators=500	66.81	**37.94**	66.01	73.71	65.44	82.42
booster='tree', n_estimators=500, max_depth=10	68.04	35.39	**67.94**	**74.27**	**66.28**	87.36
booster='tree', n_estimators=500, max_depth=10, eta=.1	66.46	34.55	65.78	72.70	64.75	87.36
booster='tree', n_estimators=1000, max_depth=10, eta=.2	**68.06**	36.27	67.09	74.22	66.03	**88.99**

Performance (Training Set Detailed)

Hyperparameters	F1 Train					F1 Dev
	LOC	MISC	ORG	PER	OVR	OVR
booster='tree'	75.31	44.55	76.10	79.29	73.31	61.15
booster='tree', n_estimators=200	79.56	50.53	79.45	83.38	77.28	62.94
booster='tree', n_estimators=500	84.63	58.03	84.27	88.59	82.42	65.44
booster='tree', n_estimators=500, max_depth=10	**89.49**	67.20	**88.83**	**92.47**	87.36	**66.28**
booster='tree', n_estimators=500, max_depth=10, eta=.1	83.55	57.97	82.95	87.31	81.37	64.75
booster='tree', n_estimators=1000, max_depth=10, eta=.2	88.99	**90.90**	71.19	90.31	**93.51**	66.03

The XGBoost models actually did very well in combatting overfitting, as the gaps between the training F1 Scores and the development F1 Scores were much smaller than those of previous models. Unfortunately though, the trade off to avoid underfitting led to the model underperforming, at least compared to the best Logistic Regression model. The best XGBoost still performed better than both the best Random Forest and the basic Perceptron, albeit very marginally. Overall, these results aren't surprising, as XGBoost did help with the overfitting as expected, but brought the overall predictive power down too much, which is a common side effect. Ultimately, the best XGBoost model ended up using hyperparameters n_estimators=500 and max_depth=10, and had an F1 Score of 66.28 on the development set.

Model Comparison

Best Performance (Development Set Detailed)

Model	F1 Development	F1 Train

	LOC	MISC	ORG	PER	OVR	OVR
Perceptron	67.93	38.19	65.04	76.18	66.16	97.03
Logistic Regression (max_iter=1000, C=7, solver='liblinear')	**70.45**	**40.14**	**68.04**	**79.24**	**69.12**	97.83
Random Forest (n_estimators=40, max_features=.5)	65.94	33.02	66.96	71.49	64.71	**98.51**
XGBoost (booster='tree', n_estimators=500, max_depth=10)	68.04	35.39	67.94	74.27	66.28	87.36

Best Performance (Training Set Detailed)

Model	F1 Train					F1 Dev
	LOC	MISC	ORG	PER	OVR	OVR
Perceptron	97.22	91.74	97.15	99.31	97.01	66.16
Logistic Regression (max_iter=1000, C=7, solver='liblinear')	97.90	94.36	97.72	**99.70**	97.83	**69.12**
Random Forest (n_estimators=40, max_features=.5)	**98.53**	**96.56**	**98.51**	99.45	**98.51**	64.71
XGBoost (booster='tree', n_estimators=500, max_depth=10)	89.49	67.20	88.83	92.47	87.36	66.28

Overall, the Logistic Regression was the clear winning model, performing better across the board on the development set, and was therefore chosen as the final model.

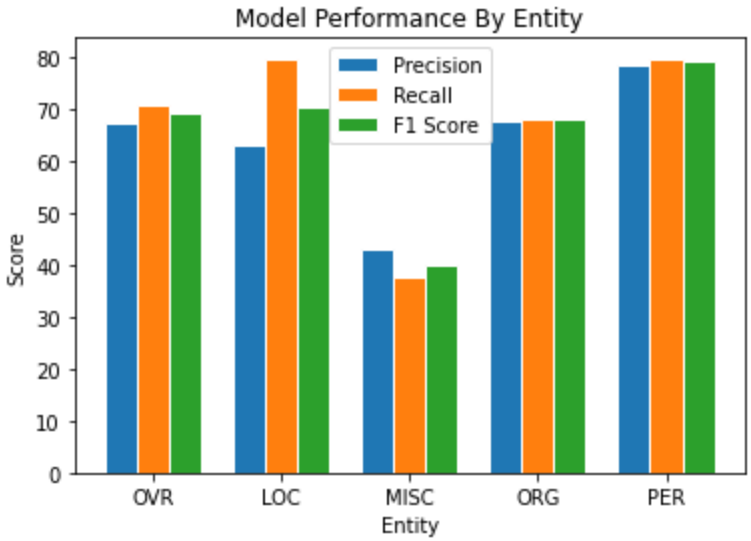
Best Model and Test Performance

Best Model	F1 Train	F1 Development	F1 Test
Logistic Regression	97.83	69.12	72.78

The Logistic Regression model performed well, surpassing the 68 F1 Score threshold for the train, development, and test data. There are some clear overfitting issues, as the training scores are 25+ points higher than the development and test scores, and the model still makes a lot of errors, which are explored below in the error analysis section of this writeup. Interestingly, the model performs more than 3 points better on the testing set as compared to the development set, so it might actually generalize a bit better than what would be expected based on the development scores. Otherwise, the testing data set may by chance just be a little closer to the training set than the development set was. Either way, the model still performed well overall.

Error Analysis

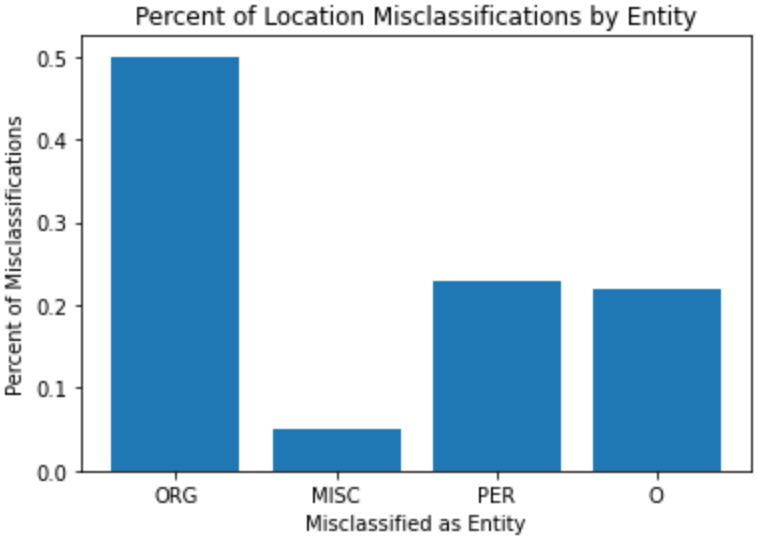
Overall Performance Errors



As seen in the graph above, each entity had varying degrees of success. Miscellaneous performed the worst by a substantial amount, likely because many of the Miscellaneous tags the model saw during training were specific to the training set. The model performed best on People, which makes sense as many names may be commonly used, and names in general show up in predictable places in sentences. The model also seems to predict Location frequently, as the recall was high and the precision was low. The precision, recall, and F1 Score for Organization were nearly identical, showing the model had good balance when predicting that entity.

Errors By Entity

Location



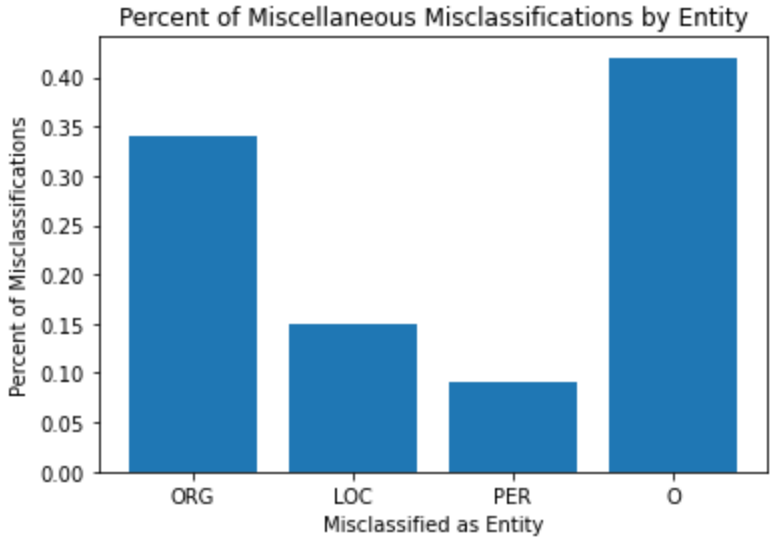
Examples:

Actual: B-LOC, Predicted: B-ORG
Word: Cuba

Actual: B-LOC, Predicted: B-ORG
Word: Israel

Almost 50% of the misclassified locations were classified as organizations, about 20% each for people and non-entities, and very few as miscellaneous. It's unclear why the distribution is so skewed in favor of organizations, but I would guess the model has difficulty with locations it has not seen before, and it may be easy to mistake them for organizations based on how each of them are used in sentence structure. A closer look into some examples shows pretty clear misses, as Cuba and Israel are both obviously locations. Furthermore, it is surprising to see Cuba, a predominantly Spanish speaking country, be misclassified, given that the data sets used are all in Spanish.

Miscellaneous



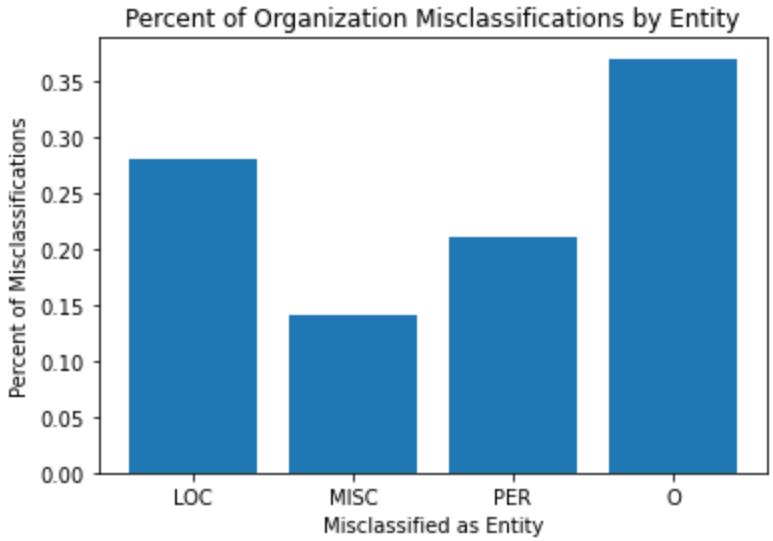
Actual: B-MISC, Predicted: B-ORG
Word: GFBITAL

Actual: B-MISC, I-MISC, Predicted: B-LOC, I-LOC
Words: San, Cristóbal

The miscellaneous misidentifications were predominantly predicted as non-entities. This makes sense since the miscellaneous words are likely training/development set specific, and the model's predictive power on those words likely doesn't generalize well, causing it to default often to just saying the word is a non-entity. A closer look at some misclassifications that were not predicted as non-entities shows some interesting results. GFBITAL was predicted as an organization, likely because the model assumed it was an acronym of some sort. San Cristóbal was predicted as a location, and a Google search reveals that it is actually a location, so this may have been mislabeled. Otherwise, there may be an edge

case where San Cristóbal referred to something other than the location. Either way, it was hard to fault the model too much for these specific misclassifications, showing how difficult it is to get the miscellaneous labeled entities correct.

Organization



Actual: B-ORG, Predicted: B-LOC
Word: Brasil

Actual: B-ORG, Predicted: B-LOC
Word: Colombia

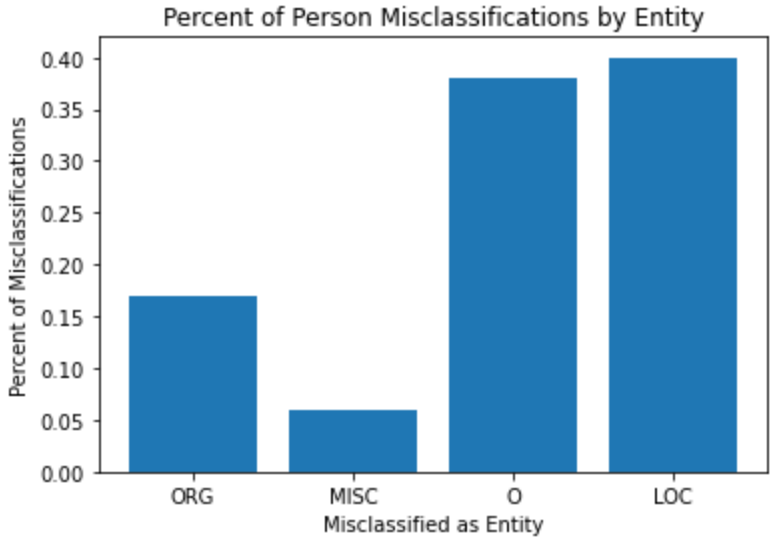
Actual: B-ORG, Predicted: B-LOC
Word: Paraguay

Actual: I-ORG, I-ORG , I-ORG, I-ORG, Predicted: I-PER, I-LOC, I-LOC, I-MISC
Word: Río, de, la, Plata

Actual: I-ORG, I-ORG, I-ORG Predicted: I-PER, I-PER, I-MISC
Word: New, York, Times

Organizations were mostly misclassified as non-entities, but overall the distribution was somewhat even across the board. There were some misses that did not make sense at first, as Brasil, Colombia, and Paraguay are all “incorrectly” classified as locations, even though they are all obviously locations. However, given that this data is from around 2002, when the FIFA World Cup was happening, these could all be referring to soccer teams, hence the organization tag. Regardless, it’s again hard to fault the model here. However, the other errors show the model with some pretty clear misses.

Person



Actual: B-PER, Predicted: I-MISC
Word: Elián

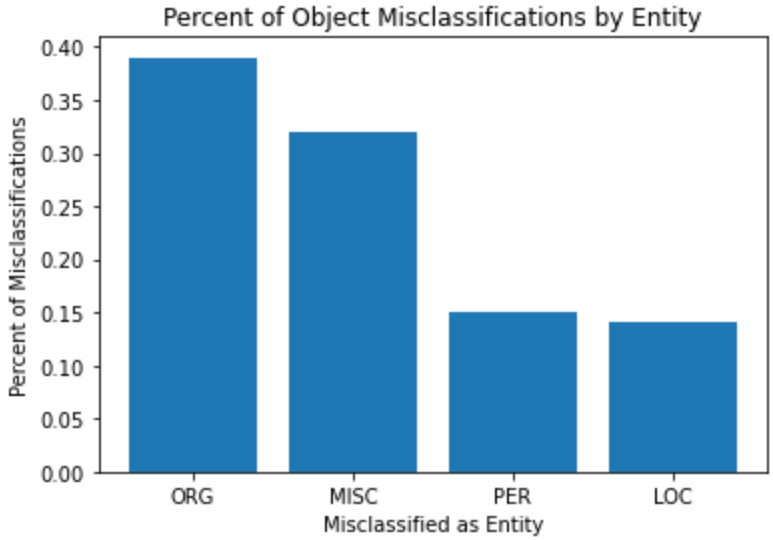
Actual: B-PER, Predicted: B-ORG
Word: Zamorano

Actual: B-PER, Predicted: O
Word: Gelsinger

Actual: B-PER, Predicted: B-ORG
Word: Sartor

People were mostly misclassified as non-entities or locations, and it was hard to tell exactly why the model missed people, other than that there's a lot of names out there and it can be difficult to tell what is and is not a name. The misclassifications examples above did not have a clear pattern, just general misclassifications.

Non-Entity

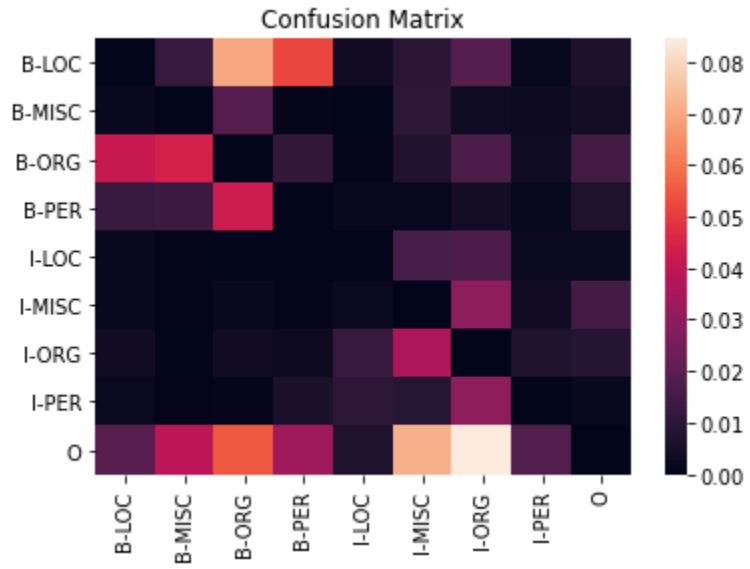


Actual: O, O, Predicted: B-PER, I-PER
Words: Empresas, Valor

Actual: O, Predicted: I-MISC
Word: de

Most non-entities that were misclassified as named entities were classified as organizations or miscellaneous, with people and location farther behind. There wasn't much of a pattern given the examples shown here.

BIO Encoding Level Confusion Matrix



The confusion matrix above shows the percentage of the total misclassifications for each pairing. Most of the misclassifications came from predicting something as an entity of some sort when it

was really a non-entity. Additionally, a lot of errors came from predicting “B-LOC” as “B-ORG” or “B-PER”. In general, a lot of the errors seemed to come from the model confusing the beginnings of entities with the beginning of other entities, or confusing the insides of entities with the insides of other entities. Rarely was an actual inside entity confused with a beginning entity, but confusing beginning entities with inside entities was a little bit more common.