HW5 Report

Task 2 LFs

The performance of my Label Functions:

```
L_dev = labeler.apply_existing(split=1)
tp, fp, tn, fn = gen_model.error_analysis(session, L_dev, L_gold_dev)
```

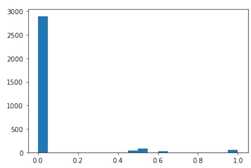
```
========================================
Scores (Un-adjusted)
========================================
Pos. class accuracy: 0.917
Neg. class accuracy: 0.97
Precision            0.524
Recall               0.917
F1                   0.667
----------------------------------------
TP: 11 | FP: 10 | TN: 324 | FN: 1
========================================
```

```
L_dev.lf_stats(session, L_gold_dev, gen_model.learned_lf_stats()['Accuracy'])
```

	j	Coverage	Overlaps	Conflicts	TP	FP	FN	TN	Empirical Acc.	Learned Acc.
LF_between	0	1.000000	1.000000	0.078035	8	15	4	319	0.945087	0.974917
LF_organization_right_movie_denial	1	0.225434	0.225434	0.002890	0	0	0	78	1.000000	0.671505
LF_person_right_movie_denial	2	0.147399	0.147399	0.011561	0	0	0	51	1.000000	0.647992
LF_ending_word	3	1.000000	1.000000	0.078035	10	11	2	323	0.962428	0.976840
LF_between_ending	4	0.028902	0.028902	0.008671	7	0	0	3	1.000000	0.545564
LF_usc	5	0.017341	0.017341	0.017341	1	0	0	5	1.000000	0.540134
LF_person_school_distance	6	0.023121	0.023121	0.008671	6	2	0	0	0.750000	0.545814
LF_HarryPotter	7	0.028902	0.028902	0.008671	8	2	0	0	0.800000	0.565138

I think my LFs are doing fairly well, because F1 is 0.67.

Marginal Distribution:



To me, this is a relatively good distribution, because the dataset is really unbalanced, it is hard to get a bimodal distribution with two even peaks.

Task 3 Distant supervision

performance

```
L_dev = labeler.apply_existing(split=1)
tp, fp, tn, fn = gen_model.error_analysis(session, L_dev, L_gold_dev)
```

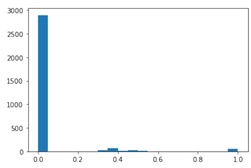
```
========================================
Scores (Un-adjusted)
========================================
Pos. class accuracy: 0.583
Neg. class accuracy: 0.985
Precision            0.583
Recall               0.583
F1                   0.583
----------------------------------------
TP: 7 | FP: 5 | TN: 329 | FN: 5
========================================
```

```
L_dev.lf_stats(session, L_gold_dev, gen_model.learned_lf_stats()['Accuracy'])
```

	j	Coverage	Overlaps	Conflicts	TP	FP	FN	TN	Empirical Acc.	Learned Acc.
LF_between	0	1.000000	1.000000	0.335260	8	15	4	319	0.945087	0.976884
LF_organization_right_movie_denial	1	0.225434	0.225434	0.080925	0	0	0	78	1.000000	0.669599
LF_person_right_movie_denial	2	0.147399	0.147399	0.046243	0	0	0	51	1.000000	0.641410
LF_ending_word	3	1.000000	1.000000	0.335260	10	11	2	323	0.962428	0.975397
LF_between_ending	4	0.028902	0.028902	0.008671	7	0	0	3	1.000000	0.543352
LF_usc	5	0.017341	0.017341	0.017341	1	0	0	5	1.000000	0.533304
LF_person_school_distance	6	0.023121	0.023121	0.008671	6	2	0	0	0.750000	0.537458
LF_HarryPotter	7	0.028902	0.028902	0.008671	8	2	0	0	0.800000	0.542449
LF_distant_supervision	8	0.320809	0.320809	0.300578	11	100	0	0	0.099099	0.347684

I think my LFs are doing fairly well, because F1 is still around 0.6 .

Marginal Distribution:



Task 4 End Extraction Model

The hyper-parameters I choose:

```
train_kwargs = {
    'lr':              0.001, # learning rate of the model
    'embedding_dim': 50,   # size of the feature vector
    'hidden_dim':    50,   # number of nodes in each layer in the model
    'n_epochs':      20,   # number of training epochs
    'dropout':       0.2,  # dropout rate (during learning)
    'batch_size':    70,   # training batch size
    'seed':          1701
}
```

End prformance:

```
tp, fp, tn, fn = lstm.error_analysis(session, test_cands, L_gold_test)
```

```
========================================
Scores (Un-adjusted)
========================================
Pos. class accuracy: 0.542
Neg. class accuracy: 0.978
Precision            0.406
Recall               0.542
F1                   0.464
----------------------------------------\
TP: 13 | FP: 19 | TN: 847 | FN: 11
========================================
```

I tried several different combinations of hyper-parameters, this one yield the best performance, with 0.46 F1 score.