INF 558: Building Knowledge Graphs

Report of Homework5

Information Extraction II

Author: Zongdi Xu (USC ID 5900-5757-70)

Date: Mar 11, 2020

Task 1.1

(Refer to uploaded files)

Task 1.2

1.2.1 Labeling functions

```
from snorkel.lf_helpers import (
    get_left_tokens, get_right_tokens, get_between_tokens,
    get_text_between, get_tagged_text, contains_token
)


import random, sys

# TODO: Define your LFs here, below is a very simple LF
def LF_random(c):
    return round(random.random())

def LF_distance(c):
    return 1 if len(list(get_between_tokens(c)))<7 else -1

def LF_hash(c):
    return (hash(c.person.get_span())+hash(c.organization.get_span())+sys.maxsize) % 2 * 2 -1

def LF_right_detect(c):
    return 1 if contains_token(c, 'school') or contains_token(c, 'college') \
        or contains_token(c, 'university') \
        else -1

def LF_between_detect_refined(c):
    candidate_predicates = list(get_between_tokens(c))
    prepositions = {'at', 'from', 'to'}
    intransitive_predicates = {'graduated', 'studied', 'enrolled',  'went', 'returned', 'educa
    transitive_predicates = {'attended'}
    phrases = {'member', 'of'}
    if len(transitive_predicates.intersection(candidate_predicates))>0 or \
        len(prepositions.intersection(candidate_predicates))>0 and \
            len(intransitive_predicates.intersection(candidate_predicates))>0 \
        or len(phrases.intersection(candidate_predicates))>1:
        return 1
    return -1

def LF_combined(c):
    if LF_between_detect_refined(c)==1 and LF_right_detect(c)==1:
        return 1
    return -1

def LF_combined_refined(c):
    taboo = {'later', 'here', 'there'}
```

```
    if LF_combined(c)==1 and not len(taboo.intersection(get_between_tokens(c)))>0:
        return 1
    return -1
```

1.2.2 Performance

- Score of generative model

```
In [71]: L_dev = labeler.apply_existing(split=1)
         tp, fp, tn, fn = gen_model.error_analysis(session, L_dev, L_gold_dev)
```

```
Clearing existing...
Running UDF...
[========================================] 100%

========================================
Scores (Un-adjusted)
========================================
Pos. class accuracy: 0.778
Neg. class accuracy: 0.981
Precision            0.778
Recall               0.778
F1                   0.778
----------------------------------------
TP: 14 | FP: 4 | TN: 205 | FN: 4
========================================
```

- Detailed statistics about LFs learned by generative model

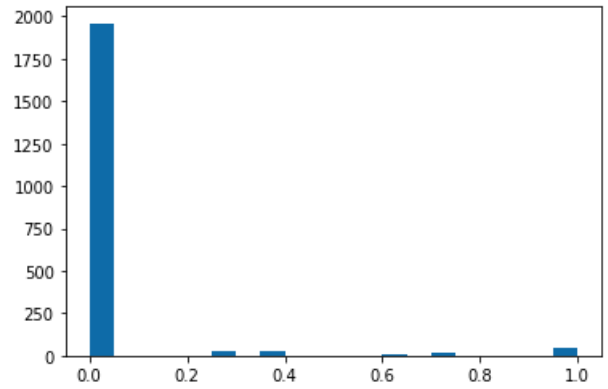
```
In [72]: L_dev.lf_stats(session, L_gold_dev, gen_model.learned_lf_stats()['Accuracy'])
```

Out[72]:

	j	Coverage	Overlaps	Conflicts	TP	FP	FN	TN	Empirical Acc.	Learned Acc.
LF_distance	0	1.0	1.0	0.713656	4	83	14	126	0.572687	0.679795
LF_hash	1	1.0	1.0	0.713656	11	103	7	106	0.515419	0.567580
LF_right_detect	2	1.0	1.0	0.713656	17	9	1	200	0.955947	0.973130
LF_combined_refined	3	1.0	1.0	0.713656	13	2	5	207	0.969163	0.976528

1.2.3 Distribution of training marginals

```
In [69]: import matplotlib.pyplot as plt
         plt.hist(train_marginals, bins=20)
         plt.show()
```



1.2.4 Comment of marginal distribution

It is rather good because it shows a clear differentiation between 0.0 and 1.0.

Task 1.3

1.3.1 Additional Labeling Function

```
from SPARQLWrapper import SPARQLWrapper, JSON


def LF_distant_supervision(c):
    if not LF_right_detect(c)==1:
        return -1
    sparql = SPARQLWrapper("http://dbpedia.org/sparql")
    sparql.setQuery(f"""
        PREFIX foaf: <http://xmlns.com/foaf/0.1/>
        PREFIX dbo: <http://dbpedia.org/ontology/>
        PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
        SELECT DISTINCT ?_name ?_edu
        WHERE {{
            [] a dbo:Person ;
                foaf:name ?name ;
                dbo:almaMater [ foaf:name ?edu ] .
            BIND(STR(?name) AS ?_name)
            BIND(STR(?edu) AS ?_edu)
            FILTER(REGEX(?_edu, "(school)|(university)|(college)|(academy)", "i"))
            FILTER(REGEX(?_name, "{'|'.join(list(map(lambda name: f'({name})', c.person.get_
            # FILTER(STR(?name) = "{c.person.get_span()}")
            FILTER(?_edu = "{c.organization.get_span()}")
        }}
        # LIMIT 10
    """)
    sparql.setReturnFormat(JSON)
    results = sparql.query().convert()
    return 1 if len(results["results"]["bindings"])>0 else -1
```

1.3.2 Performance

- Score of generative model

```
In [19]: L_dev = labeler.apply_existing(split=1)
         tp, fp, tn, fn = gen_model.error_analysis(session, L_dev, L_gold_dev)

         Clearing existing...
         Running UDF...
         [========================================] 100%

         ========================================
         Scores (Un-adjusted)
         ========================================
         Pos. class accuracy: 0.778
         Neg. class accuracy: 0.986
         Precision            0.824
         Recall               0.778
         F1                   0.8
         ----------------------------------------
         TP: 14 | FP: 3 | TN: 206 | FN: 4
         ========================================
```

- Detailed statistics about LFs learned by generative model

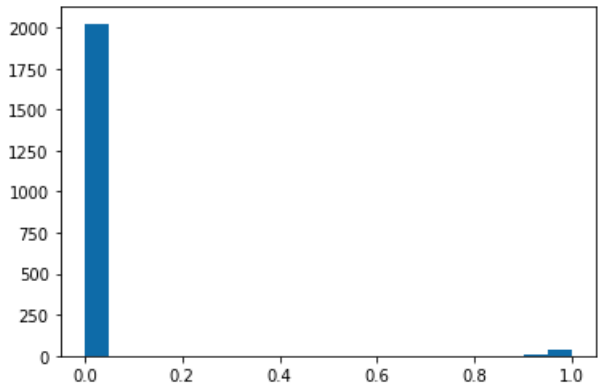
```
In [20]: L_dev.lf_stats(session, L_gold_dev, gen_model.learned_lf_stats()['Accuracy'])
```

Out[20]:

	j	Coverage	Overlaps	Conflicts	TP	FP	FN	TN	Empirical Acc.	Learned Acc.
LF_distance	0	1.0	1.0	0.748899	4	83	14	126	0.572687	0.672184
LF_hash	1	1.0	1.0	0.748899	10	109	8	100	0.484581	0.523581
LF_right_detect	2	1.0	1.0	0.748899	17	9	1	200	0.955947	0.974169
LF_combined_refined	3	1.0	1.0	0.748899	13	2	5	207	0.969163	0.977456
LF_distant_supervision	4	1.0	1.0	0.748899	2	1	16	208	0.925110	0.977791

1.3.3 Distribution of training marginals

```
In [16]: import matplotlib.pyplot as plt
plt.hist(train_marginals, bins=20)
plt.show()
```



1.3.4 Comment of marginal distribution

It is even better because it maintains a clear differentiation between 0.0 and 1.0 like that in `Task 1.2`, while removes some ambiguit y near 0.4 or 0.6.

Task 1.4

- Best tuned parameters

```
train_kwargs = {
  'lr':           0.009, # learning rate of the model
  'embedding_dim': 70,   # size of the feature vector
  'hidden_dim':    60,   # number of nodes in each layer in the model
  'n_epochs':      11,   # number of training epochs
  'dropout':       0.2,  # dropout rate (during learning)
  'batch_size':    70,   # training batch size
  'seed':          281
}
```

- Best F1 score: `0.483`

```
lstm = LSTM(n_threads=None)
lstm.train(train_cands, train_marginals, X_dev=dev_cands, Y_dev=L_gold_dev,
```

```
[LSTM] Training model
[LSTM] n_train=2074  #epochs=11  batch size=70
```

```
/Users/crxon/558/env/lib/python3.7/site-packages/torch/nn/functional.py:1351
eprecated. Use torch.sigmoid instead.
  warnings.warn("nn.functional.sigmoid is deprecated. Use torch.sigmoid inst
```

```
[LSTM] Epoch 1 (4.98s)  Average loss=0.162053   Dev F1=0.00
[LSTM] Epoch 2 (10.52s) Average loss=0.116270   Dev F1=0.00
[LSTM] Epoch 3 (16.60s) Average loss=0.116469   Dev F1=0.00
[LSTM] Epoch 4 (22.75s) Average loss=0.113216   Dev F1=0.00
[LSTM] Epoch 5 (29.10s) Average loss=0.090246   Dev F1=0.00
[LSTM] Epoch 6 (35.39s) Average loss=0.049010   Dev F1=19.05
[LSTM] Epoch 7 (41.54s) Average loss=0.032666   Dev F1=57.14
[LSTM] Epoch 8 (47.82s) Average loss=0.029918   Dev F1=53.85
[LSTM] Epoch 9 (54.20s) Average loss=0.026681   Dev F1=57.14
[LSTM] Epoch 10 (60.37s)        Average loss=0.021817   Dev F1=60.00
[LSTM] Model saved as <LSTM>
[LSTM] Epoch 11 (66.57s)        Average loss=0.021602   Dev F1=51.61
[LSTM] Training done (66.87s)
[LSTM] Loaded model <LSTM>
```

Report performance of your final extractor

In [25]:
```
p, r, f1 = lstm.score(test_cands, L_gold_test)
print("Prec: {0:.3f}, Recall: {1:.3f}, F1 Score: {2:.3f}".format(p, r, f1))
```

```
Prec: 0.609, Recall: 0.400, F1 Score: 0.483
```