Snorkel Workshop: Extracting Spouse Relations from the News

Part 4: Training our End Extraction Model

In this final section of the tutorial, we'll use the noisy training labels we generated in the last tutorial part to train our end extraction model.

For this tutorial, we will be training a fairly effective deep learning model. More generally, however, Snorkel plugs in with many ML libraries including TensorFlow, making it easy to use almost any state-of-the-art model as the end extractor!

In [ ]:
```
%load_ext autoreload
%autoreload 2
%matplotlib inline
import os
import numpy as np

# Connect to the database backend and initalize a Snorkel session
from lib.init import *
from snorkel.annotations import load_marginals
from snorkel.models import candidate_subclass

Spouse = candidate_subclass('Spouse', ['person1', 'person2'])
```

I. Loading Candidates and Gold Labels

In [ ]:
```
from snorkel.annotations import load_gold_labels

train_cands = session.query(Spouse).filter(Spouse.split == 0).order_by(Spouse.id).all()
dev_cands   = session.query(Spouse).filter(Spouse.split == 1).order_by(Spouse.id).all()
test_cands  = session.query(Spouse).filter(Spouse.split == 2).order_by(Spouse.id).all()

L_gold_dev  = load_gold_labels(session, annotator_name='gold', split=1, load_as_array=True, zero_one=True)
L_gold_test = load_gold_labels(session, annotator_name='gold', split=2, zero_one=True)

train_marginals = load_marginals(session, split=0)
```

II. Training a *Long Short-term Memory* (LSTM) Neural Network

LSTMs can acheive state-of-the-art performance on many text classification tasks. We'll train a simple LSTM model below.

In deep learning, hyperparameter tuning is very important and computationally expensive step in training models. For purposes of this tutorial, we've pre-selected some settings so that you can train a model in under 10 minutes. Advanced users can look at our Grid Search Tutorial for more details on choosing these parameters.

Parameter	Definition
n_epochs	A single pass through all the data in your training set
dim	Vector embedding (i.e., learned representation) dimension
lr,	The learning rate by which we update model weights after,computing the gradient
dropout	A neural network regularization techique [0.0 - 1.0]
print_freq	Print updates every k epochs
batch_size	Estimate the gradient using k samples. Larger batch sizes run faster, but may perform worse
max_sentence_length	The max length of an input sequence. Setting this too large, can slow your training down substantially

Please Note !!!

With the provided hyperparameters below, your model should train in about 9.5 minutes.

In [ ]:
```
from snorkel.learning.pytorch.rnn import LSTM

train_kwargs = {
    'lr':         0.001,
    'dim':        100,
    'n_epochs':   10,
    'dropout':    0.25,
    'print_freq': 1,
    'batch_size': 128,
    'max_sentence_length': 100
}

lstm = LSTM(n_threads=1)
lstm.train(train_cands, train_marginals, X_dev=dev_cands, Y_dev=L_gold_dev, **train_kwargs)
```

Now, we get the precision, recall, and F1 score from the discriminative model:

In [ ]:
```
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))
```

We can also get the candidates returned in sets (true positives, false positives, true negatives, false negatives) as well as a more detailed score report:

In [ ]:
```
tp, fp, tn, fn = lstm.error_analysis(session, test_cands, L_gold_test)
```

Finally, let's save our model for later use.

In [ ]:
```
lstm.save("spouse.lstm")
```