CSE 244 HW1: Movie Domain Relation Classification

Rishi Rajasekaran
rrajasek@ucsc.edu

Abstract

This report is an exploration of training different linear and deep neural network architectures for the task of extracting Freebase Knowledge Graph relations from text. I explore the training of the following model types — Multilayer Perceptron, multilayer deep networks, Convolutional Neural Networks and Recurrent Neural Networks.

1 Problem Statement

Relation extraction is a problem of recognizing entity-entity relations from free form text to solve the general problem of Knowledge Graph based question-answering.

A user's utterance can encode multiple relations or none. Therefore, it can be modeled as learning a multilabel predictive distribution of the form:

$$P(y_1, \ldots, y_n \,|\, x)$$

This can be achieved by learning a 1 vs all or a joint model across all the labels through a multilabel training procedure. We opt to do the latter due to it learning a more complete hypothesis instead of assuming conditional independence of labels.

2 Data Description

The input data provided comprises of $(utterance, relations)$ pairs where $utterance$ is free form text and $relations$ comprise of space separated FreeBase Entity-Relation-Entity triples. Example utterances are provided on Table 1. The utterances comprise of lower-cased free-form text which potentially represents ASR output. The relations number from 1 to 4 for each example along with a special `NO_REL` tag for utterances without any relation.

3 Feature Extraction

The general problem that is being solved is a example labelling problem — given an input x, we generate a data vector $\boldsymbol{y} = (y_1, \ldots, y_n)$ where $y_i = 1$ indicates the presence of the corresponding label and $y_i = 0$ indicates absence of the label.

The input vector can be of two types, depending on the type of model used:

1. A Bag-of-Words vector $\boldsymbol{w} = (w_1, \ldots, w_{|V|})$ where $|V|$ is the Vocabulary size. $w_i = 1$ indicates the word w_i of the vocabulary is present in the sentence and 0 otherwise.

2. A Word Sequence vector $\boldsymbol{w} = (w_1, \ldots, w_n)$ where w_i indicates the ith word in the utterance.

The vectorization logic is implemented in the `vectorizers.py` file through the classes `OneHotVocabVectorizer` and `SequenceVocabVectorizer`.

4 Modeling

The data was split in the ratio $(70 : 15 : 15)$ across train, validation and test sets. We use the same train, test, validation datasets for all models with the exception of the ensemble approach where each learner was trained on different shuffled splits of train-plus-validation set with a separate held out test set for verification.

We use the `AdamOptimizer` as the optimizer with parameters. Unless indicated otherwise, we use the `BCEWithLogitsLoss` as the loss function, learning rate of 0.001 and a training batch size of 16. The models were trained for around 100 epochs. The output layer of all models comprises of 46 sigmoid neurons each corresponding to a class label. The labels are predicted by thresholding the sigmoid output with a value of 0.5.

Utterance	Relations
who plays luke on star wars new hope	movie.starring.actor movie.starring.character
what are the names of female actors who played in lost	movie.starring.actor actor.gender

Table 1: Sample Utterance and Relations

The parameters that yielded the highest F1 score on the validation set during training were used for performing inference using the model.

4.1 Multi Layer Perceptron

The Multi Layer Perceptron model was implemented with a hidden layer size of 100 and was implemented as a multi-label classifier. The input provided is a BOW vector.

4.2 Deep Fully Connected Network

This comprises of a two-hidden layer neural network of hidden dimensions $(128, 64)$. The input is provided as a BOW vector.

4.3 Convolutional Neural Network

The CNN model comprises of the following layers:

- 1-D convolution kernel size 5, stride 1, 256 output channels (ELU activation)
- 1-D convolution kernel size 3, stride 2, 256 output channels (ELU activation)
- 1-D convolution kernel size 3, stride 2, 256 output channels (ELU activation)
- 1-D convolution kernel size 3, stride 2, 256 output channels (ELU activation)
- Linear Layer

The input is a padded sequence of one-hot vectors.

4.4 BiLSTM

The BiLSTM model was implemented using the following architecture:

- Learned embedding layer initialized with GloVe 6B 300-d weights.
- 2 layer encoder bidirectional LSTM with hidden state size 100
- Linear layer that takes the initial and final hidden state for each direction.

4.5 BiLSTM Ensemble

The BiLSTM ensemble comprises of 5 independently trained BiLSTM models where each model is trained on a different split of the dataset. We take the average of the output of each model and threshold it against a value of 0.5 to make the prediction.

5 Results

The results are reported for each model on Table 2. We report macro and micro-averaged **precision, recall** and **F-1** scores for each model to illustrate the performance with and without accounting for label imbalance. The severe label imbalance accounts for the high disparity between the macro and micro-averages.

Of the models trained, the CNN appears to have the highest micro-averaged training performance and returns very high macro averaged results. However, it did not yield good results on the Kaggle leaderboard — obtaining only 70% accuracy, so it very likely is overfitting on the data. An alternative approach I could have tried was to train CNN on word embeddings to evaluate if it can learn more stable hypotheses.

BiLSTM models seem to fit on rare labels better since they yield significantly higher macro-average scores. However, it doesn't seem to do well on the majority labels, possibly learning noisy hypotheses. However, they appear to have significantly better recall than the simpler models. Training an ensemble of LSTM models seems to yield the benefits of good precision and recall.The BiLSTM ensemble approach delivered the best performance among my submitted models in the both the public and private leaderboard and was ranked 4th overall for the competition.

6 Future Work

A potential avenue for improving the model performance is to add additional annotation information – named entity recognition, Part-of-Speech tags. Secondly, transfer learning from pretrained state-of-the-art models (such as BERT) could also yield better performance.

Model	Macro Average			Micro Average		
	Precision	Recall	F1	Precision	Recall	F1
MLP	0.45	0.37	0.40	0.88	0.76	0.81
DeepFC	0.44	0.37	0.39	0.84	0.75	0.79
CNN	0.58	0.55	0.56	0.93	0.90	0.92
BiLSTM	0.62	0.57	0.58	0.82	0.81	0.82
n-BiLSTM	0.46	0.39	0.42	0.89	0.83	0.86

Table 2: Test Data Results