Core Relation Extraction from Natural Language

Anuroop John Abraham
ajohnabr@ucsc.edu

1 Introduction

The problem statement is to determine knowledge graph relations in Freebase schema, which are invoked in user utterances addressed to a conversational system using conventional machine learning models. Since both the user utterance and its corresponding Core Relations are given, the problem falls under supervised machine learning approach.The goal is to identify the relationships of each utterance given by user.

2 Datasets

Since all of the relevant files where in text format, used pandas(Mckinney, 2011) library to do the data manipulation. Data from train_data.csv and test_data.csv where imported as a dataframe. The train dataset has contains 2253 user utterances and its corresponding core relations and the test data contains 981 user utterances. Each user utterance in train data is associated with one or more number of core relations. While framing the machine learning approach for this problem, "utterances",can be considered as the input and "Core relations" as output respectively.
E.g, for the utterance "who plays luke on star wars new hope" the Core relation associated to it are "movie.starring.actor" and "movie.starring.character".
The total number of unique words in the set of input utterances is 1117. In total there are 19 unique output labels and the distribution of them are given below

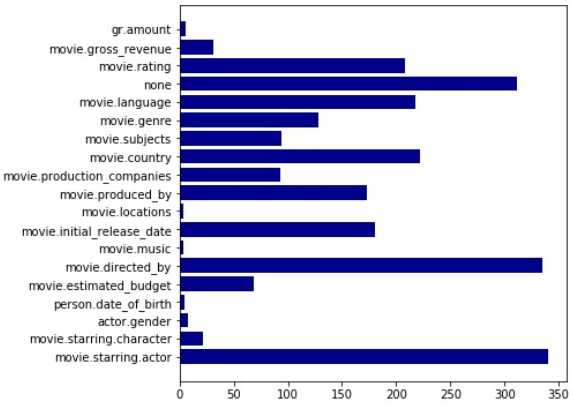

Figure 1: Frequency Distribution of Core Relations

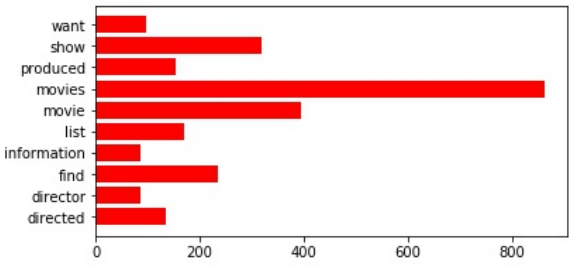

Figure 2: Frequency Distribution of Most Common Words

3 Models

3.1 Data Preprocessing

The utterances had several stop words. So in-order to remove them used NLTK(Loper and Bird, 2002) stopword corpus and removed the common words from utterance dataset and the stop word corpus. In order to reduce the number of features and to get better word mappings to the output label, each word was lematized. NLTK(Loper and Bird, 2002) builtin WordNetLemmatizer is used to get the lematized words.

3.2 Feature Extraction

Every machine learning model expects its input to be in numeric format. In order to convert the input text to vector representation, count vectorizer and tfidf vectorizer is used. A vocabulary of unique words are created at first then, its sorted and each of these words are given an index. Then for each documents in the dataframe, the words in sentences are replaced by a 1-dimensional matrix of length which

is equal to the total number words of the vocabulary. In the given problem, vocabulary size is 1117, hence each sentence in train dataset will be converted into a matrix of length 1117. For eg: for the sentence, “who plays luke on star wars new hope”, the index of each word in the vocabulary would be‘who’:1085, ‘plays’:735, ‘luke’:586, ‘on’:679, ‘star’:927, ‘wars’:1063, ‘new’:660, ‘hope’:458. So a sparse 1-D vector will be generated with 1 on the above mentioned indexes and 0’s on rest of the positions. Tfidf calculates, the term frequency-inverse document frequency of the words in each sentences. Hence the weight-age of each word can be computed. This is another way of understanding the magnitude of influence of each word in a sentence to its corresponding label.

Words	CountVectorizer Index	Tf-idf
who	1085	0.373
plays	735	0.507
luke	586	0.342
on	679	0.265
star	927	0.426
wars	1063	0.303
new	660	0.324
hope	458	0.190

3.3 Models

Machine Learning is vast domain enriched with different algorithms. Finding the algorithm, that best suits the problem is the most important part. For the given problem different machine learning models are applied to get the best results. Few of those are

1. Multinomial Naive Bayes
Multinomial Naive Bayes algorithm is a probabilistic learning method that is mostly used in Natural Language Processing. The algorithm is based on the Bayes theorem and predicts the tag of a text such as a piece of email or newspaper article. It calculates the probability of each tag for a given sample and then gives the tag with the highest probability as output. The hyper-parameters of the model are alpha, fit-prior and class-prior. "alpha" is the smoothing parameter, fit-prior enables the model to learn class prior probabilities and class prior is the prior probabilities of the classes.

2. SGDClassifier
SGD Classifier is a linear classifier(SVM, logistic regression, a.o.) optimized by the Stochastic Gradient Descent approach .Gradient descent is one of the most popular algorithms to perform optimization. Stochastic gradient descent (SGD) computes the gradient using a single sample. Different hyperparameters of SGD classifier are SGDClassifier loss, penalty, alpha, l1-ratio, fit-intercept, max-iter, epsilon, learning-rate etc.When the value of when loss function is set to “hinge”, the classifier act as a linear SVM, when “log”, it acts as logistic regression model. The penalty is also know as regularization term. Defaults to ’l2’ which is the standard regularizer for linear SVM models. ’l1’ and ’elasticnet’ might bring sparsity to the model.

3. Decision Tree
A decision tree is a map of the possible outcomes of a series of related choices. The goal is to create a model that predicts the value of a target variable based on several input variables. A decision tree is a simple representation for classifying examples. Various hyperparameters of decision trees are criterion, splitter, max-depth, min-samples-split etc. criterion is to measure the quality of a split. Supported criteria are “gini” for the Gini impurity and “entropy” for the information gain. splitter refers to the strategy used to choose the split at each node, max-depth is the maximum depth of the tree and min-samples-split minimum number of samples required to split an internal node

4. Logistic Regression
Logistic Regression is a popular statistical model used for binary classification. This model does predictive analysis. Logistic regression is used to describe data and to explain the relationship between one dependent binary variable and one or more independent variables. The hyperparameters to be tuned in this model are penalty, solver is the parameter the determines which Algorithm to use in the optimization problem, max-iter: Maximum number of iterations taken for the solvers to converge etc.

5. Support Vector Classifier
SVC is an extension of SVM algorithm.SVM maps training examples to points in space so as to maximise the width of the gap between the two categories. New examples are then mapped into that same space and predicted to belong to a

category based on which side of the gap they fall. The hyperparameters of SVC are C which is the Regularization parameter, kernel which Specifies the kernel type to be used in the algorithm. It must be one of 'linear', 'poly', 'rbf', 'sigmoid', 'precomputed' or a callable.

6. Random Forest

A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is controlled with the max-samples parameter if bootstrap=True (default), otherwise the whole dataset is used to build each tree.

7. Linear SVC

Similar to SVC with parameter kernel='linear', but implemented in terms of liblinear rather than libsvm, so it has more flexibility in the choice of penalties and loss functions and should scale better to large numbers of samples.

4 Experiments

The data-set was split into train and validation-set with 80:20 ratio. Hence the training dataset contains 1802 data samples for training and 451 data samples for testing the accuracy of the model. In-order to get few sample data of different category in both training and validation, the dataset is shuffled before splitting. Used Scikit-learn(Pedregosa et al., 2011) 'train-test-split' function to split the data. Random state was enabled, since the every time function train-test-split shuffles the data and gives different set of outputs. So the random-state will ensure that the train and validation set will give same results irrespective of the number of times in which the code is executed.

4.1 Hyperparameter Tuning

Scikit-learn's(Pedregosa et al., 2011) GridsearchCV and Pipeline is used to find the optimum hyper parameters for the models. The number of folds used for the experiments was 5.

[A]. Support Vector Classifier

For the SVC algo , there are several hyperparameters, out of which i tried {'C':[1,10,100,1000],'gamma':[1,0.1,0.001,0.0001], 'kernel':['linear','rbf']}, these combinations for the parameter and the total number of candidates for the experiment was 4*4*2 = 32 candidates. The gridsearch gave maximum validation accuracy

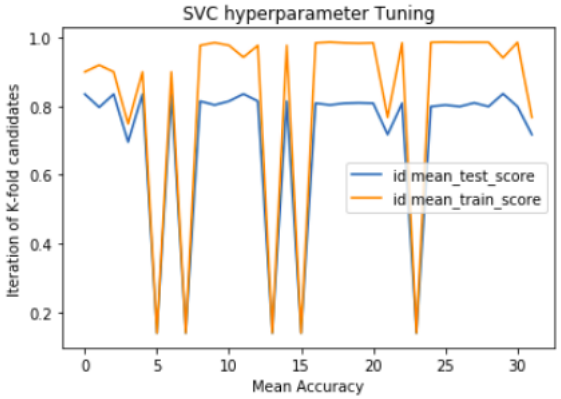

Figure 3: SVC Hyperparameter Tuning

score ie 0.836, while using the hyperparameter values C=1000, gamma=0.001 , kernal ="linear".

[B]. SGD Classifier

The list hyper parameters values used for SGDClassifier are {'loss':["hinge", "log", "squared_hinge", "modified_huber"],'alpha':[0.0001, 0.001, 0.01, 0.1], 'penalty':["l2", "l1", "none"]}. In case of SGD model, when the loss function is set to "hinge", the classifier act as a linear SVM, when "log" is used – it acts as logistic regression model. Total number of candidates during Grid search = 48 candidates. The gridsearch gave maximum

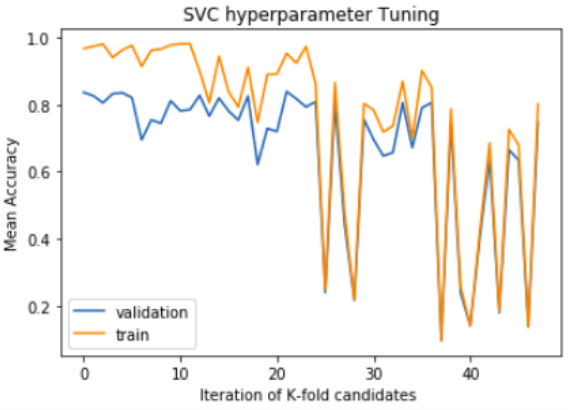

Figure 4: SGD Hyperparameter Tuning

valdation accuracy score ie 0.853, while using the hyperparameter 'alpha'= 0.0001, 'epsilon'= 0.1, 'loss'= 'hinge', 'max_iter'= 1000 and 'penalty'= 'l2'.

[C]. Decision Tree

The list hyper parameters values used for Decision Tree Classifier are 'max_depth': [2, 3, 5, 10, 20], 'min_samples_leaf': [5, 10, 20, 50, 100],'criterion': ["gini", "entropy"].So the total number of candidates during Grid search = 50 candidates.

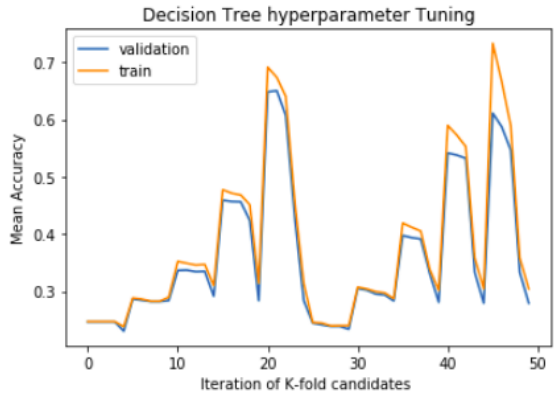

Figure 5: Decision Tree Hyperparameter Tuning

The gridsearch gave a maximum valdation accuracy score i.e , 0.836, while using the hyperparameter values 'criterion'='gini'='max_depth'=20, 'min_samples_leaf= 5.

[D]. Logistic Regression

The list hyper parameters values used for Logistic Regression are 'solver':['newton-cg', 'lbfgs','liblinear'], 'penalty': ['l2'], 'C':[0.1,1,0.01]. Hence, the total number of candidates during Grid search will be 9 candidates. The gridsearch gave maximum valdation accuracy

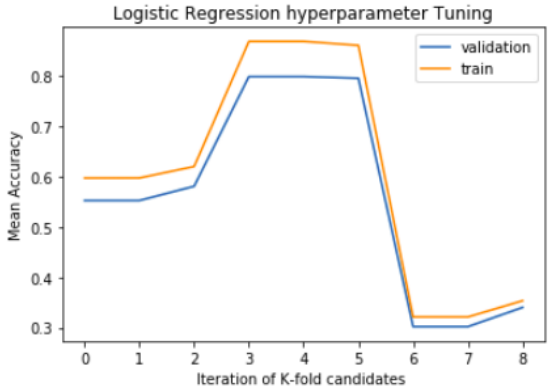

Figure 6: Logistic Regression Hyperparameter Tuning

score ie 0.799, while using the hyperparameter 'C'= 1, 'penalty'= l2, 'solver'= 'newton-cg'.

[E]. Random Forest

Different hyperparameter values used in tunning of Random Forest are'n_estimators':[10, 100, 1000], 'max_features': ['sqrt', 'log2']. and the total number of candidates during Grid search is 6 candidates. The gridsearch gave maximum

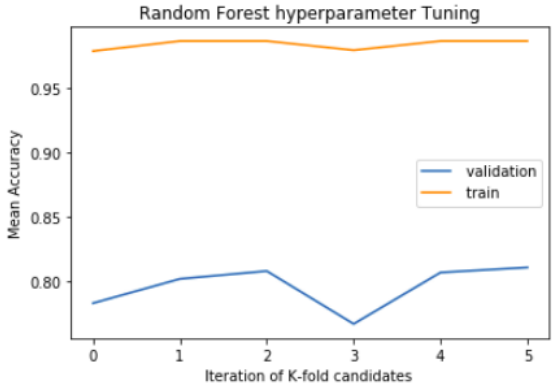

Figure 7: Random Forest Hyperparameter Tuning

valdation accuracy score ie 0.811, while using the hyperparameter 'max_features'= 'log2', 'n_estimators'= 1000'.

4.2 Modelling approach

[A]. Multiclass-single Label approach

In this approach multiple labels of each sentences were computed as a single label. For eg: The label for the sentence "who plays luke on star wars new hope" - ['movie.starring.actor', 'movie.starring.character'] is treated as a single label, ie 'movie.starring.actor movie.starring.character'. Hence the total number of labels in the training data is now 47.

[B]. Multiclass-multilabel approrach

In this approach all the original 19 labels were treated as the outputs. Used Sckit-learn's OneVsRestClassifier to implement the multi-class multi-label approach. In case of OneVsRest, for each classifier, the class is fitted against all the other classes, that means that problem of multiclass/multilabel classification is broken down to multiple binary classification problems.

[C]. Combination of multiclass-multilabel approach

Combined both the single-label and multi-label approaches to get better accuracy and generalizations. First all the inputs are passed through the multi-class multi-label model and then then the

null predictions in which the model is unable to predict any values were replaced by the predictions from the single-label model.

The performance of each model was evaluated based on the results of confusion matrix, classification report and validation accuracy score. Each row of the confusion matrix represents the instances in an actual class while each column represents the instances in a predicted class, or vice versa. Classification report however generates the precision, recall, f1score of each model.

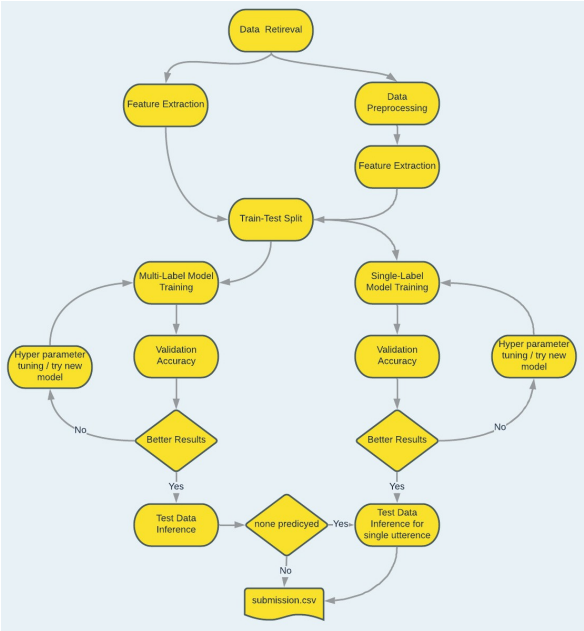

Figure 8: Random Forest Hyperparameter Tuning

5 Results

Both the single label and the multi-label approach has its on pros and cons. The single label approach gave a validation accuracy of 85% percentage, however its not always guaranteed that the same combination of labels will appear in the test data. Considering the outliers in real world scenario the total number combinations which can be formed by the labels are 19!, out of which only 47 combinations are present in the training data. Hence for the generalization purpose multilabel is best suited. The disadvantage of multi-label classifier is that, due data imbalance the model performance is very poor. For many instance the model didn't predict any labels. Hence to get the best out of both models, an approach is created with the combination of both. First the data will be passed through the multi-label model. Then the null predicted outputs were passes through the single label model to get its prediction.

Models	Train	Validation	Test
Single Label	0.96	0.85	0.74
Multi Label	0.97	0.80	0.72
Combined	0.97	0.89	0.77

References

Edward Loper and Steven Bird. 2002. Nltk: the natural language toolkit. *CoRR*, cs.CL/0205028.

Wes Mckinney. 2011. pandas: a foundational python library for data analysis and statistics. *Python High Performance Science Computer*.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2011. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12(85):2825–2830.