Relation Extraction in Knowledge Graphs using Semisupervised Learning

Kartik
kartik@ucsc.edu

Abstract

With the rising use of virtual assistants and conversation agents, it is imperative to build systems that can automatically understand the meaning, intent and relations among the entities in the text. In this work, I propose a novel semi-supervised method for extracting knowledge-graph relations from utterances provided to conversational systems. I evaluate the approach with the conventional machine learning algorithms and various other techniques.

1 Introduction

Natural language Understanding (NLU) is central part for fulfilment of any task in conversation agents such as online help/chat bots, assistants in driverless vehicles and virtual assistants. This involves understanding hierarchical semantic frame and retrieving relevant information i.e. domain, intent, slots and relations. Knowledge Graph (KG) enables an abstract and convenient way to organize large amount of information on the web in the form of nodes and edges. The representations learned from KGs are being used to solve a wide variety of downstream tasks such as question answering, link prediction, entity classification and information retrieval. Fig.1 shows an example of a knowledge graph from the few movie samples in the dataset.

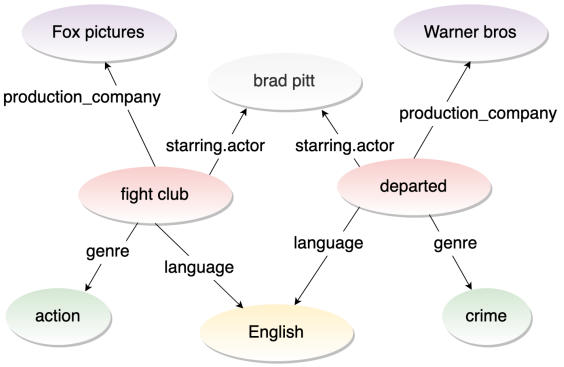

Figure 1: Knowledge graph for movies

Relation extraction is the task of extracting all the relations from an utterance. Fig. 2 shows few examples of an input utterance, its knowledge graph fragment, and the extracted relation.

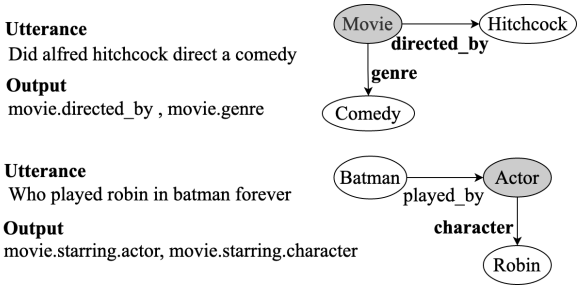

Figure 2: Relation Extraction from utterance

In this work, I provide a comprehensive comparison of conventional machine learning algorithms on a database $\mathcal{D}$ of utterances addressed to conversational system. The first section talks about the analysis of the dataset. Next, I formally define the task and discuss the methodologies used for relation extraction. Further, the experimental results are discussed.

2 Dataset

The Dataset consists of 3234 sentences with utterances including movie names, movie genres, actors, directors, producers, release dates, languages etc. The task is to extract relations from the utterances where each utterance can have multiple possible classes. The train and test set consists of 2253 and 981 samples respectively and a total of 19 classes.

As any real-world dataset, this dataset is highly imbalanced with the class `movie.starring.actor` having highest frequency of 340 samples and `movie.locations` with lowest count of only 3 samples. Fig.3 shows the frequency distribution of all classes in the dataset. Further, the different distribution of words in train and test poses another challenge as 35.7% words in test set are not in present in the train set.

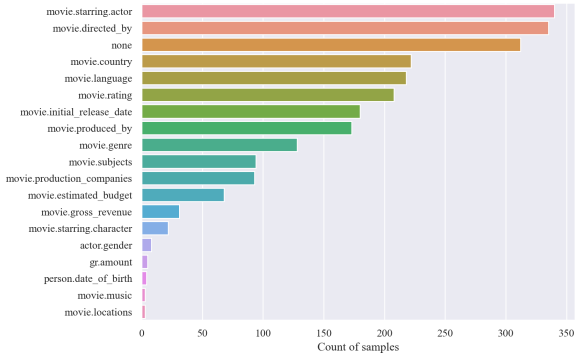

Figure 3: Frequency Distribution of classes in D

Further, I analyze the dataset for the popularity of words. Fig. 4 shows the wordcloud of the entire text where the fontsize represents the frequency of a word. Unsurprisingly, I observe that the word *movie* is the most popular word in the dataset followed by *show*, *produced*, *director* and *film* etc.

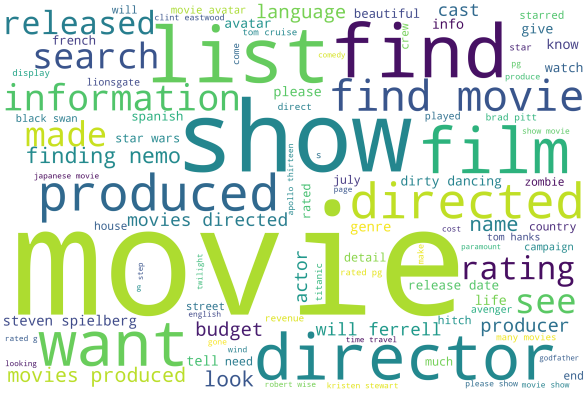

Figure 4: Wordcloud of the Dataset

3 Methodology

This section describes the process of extracting features from the sentences and methodology adopted for the multilabel classification task.

Task Definition: *Given a labelled training dataset D with a set of utterances* $\{u_1, u_2, \ldots, u_n\}$ *, the task is to learn a prediction/classification function for an unknown utterance* x *that predicts if a label* $l_i \in L$ *is present in it or not, where L is a set of possible relation labels. Note that a single utterance can have multiple possible labels.*

3.1 Text Processing

Although the provided train and test data was free of punctuation, all lowercase and cleaned, a number of processing methods were used for the experiments:

1. POS tagging: Identification of parts-of-speech such as verbs, nouns, adjectives, etc. can guide the training process in better classification. For example: Consider the utterance: *find* ***sandra bullock*** *movies*.
   Here the verb *find* is followed by multiple proper nouns whereas in this utterance:
   give me movies ***directed by*** *tyler perry*
   the verb "direct" is followed by preposition "by" which depends upon noun "tyler perry".
   Fig.5 shows the comparison between dependency trees of the sentences.

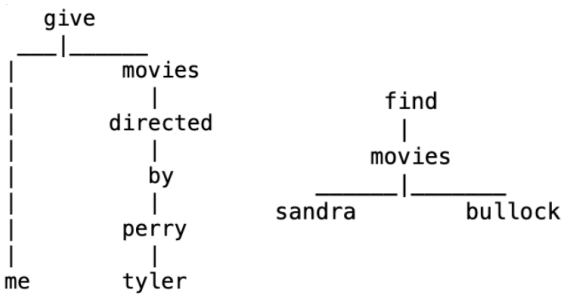

Figure 5: Dependency tree of label a) directed_by b) starring_actor

2. NER: As the task is to extract entity labels and relation between entities, Named Entity Recognition can help the model to classify an utterance.

3. Oversampling: Training an ML model on imbalanced dataset makes the predictions favorable towards the majority classes. In order to balance the dataset, I use RandomOverSampler[1] to randomly duplicate minority classes.

4. Out-of-domain Data: Additional data[2] was added for a few classes based on the utterance proporites of a class in original data. For example: Utterances with label *actor.gender* contains an occurrence of gender-specific noun such as male, female, woman. Around 200 sentences with such nouns were added with the same label so that the ML model learns to associate the nouns with labels.

[1] https://imbalanced-learn.org
[2] https://github.com/doomlab/Word-Norms-2

3.2 Feature Extraction

Representation of input textual data into meaningful features is a crucial component of any successful machine learning algorithm. The following features are used in the experiments:

1. *Count Vectors:* One of most popular and basic text feature methods is count vector. Count vectors just forms a $n \times n$ matrix of word counts for words in the corpus.

2. *Term-Frequency Inverse-Term Frequency (Tf-idf) vectors:* TFIDF calculates the importance of word using its frequency in documents. Most occurring words get lower importance and vice versa.

3. *Glove Embeddings:*(Pennington et al., 2014) These embeddings are obtained by unsupervised learning of word-word co-occurence on large data. Words close together in the semantic space have similar glove embeddings. For a sentence, embeddings are calculated by averaging the embedding of each word in that sentence.

3.3 Methodology

For the best performing model, I use a mix of supervised and unsupervised algorithm to train the model. This is based on the following observations:

1. Every model after training gets skewed towards the class with majority samples. This problem amplifies with classes less than 20 samples since the model has very limited data points to learn useful information about that class. Many classes fall under this problem. For the task I consider `actor.gender`. I obtain the glove embedding for both *actor* and *gender*. Next, from each utterance noun chunks are extracted using spacy followed by embedding average of words in each noun chunk. Now, an utterance is classified as actor.gender if the distance of any noun chunk in an utterance is less than a fixed threshold. See Fig.7 for detailed illustration.

2. As this is a multiclass multilabel problem, all ML model confuses between single label $L_1 = \{$`movie.starring.actor`$\}$ and multilabel $L_2 = \{$`movie.starring.actor, movie.starring.character`$\}$. Looking at the data, the structure of L_2 is $\{$___ *play* ___ *character* ___ *movie}*. The dependency graph of an utterance with this structure would have a word *play* with atleast 3 child nodes. For example:In sentence *who played cunningham in happydays*, the word *play* has 3 child nodes. whereas in *who played in happydays*, it refers only to *who* and *happydays*.

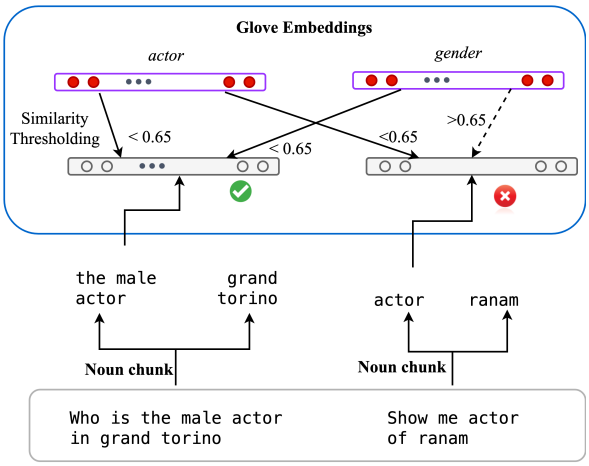

Figure 6: Similarity thresholding for actor.gender

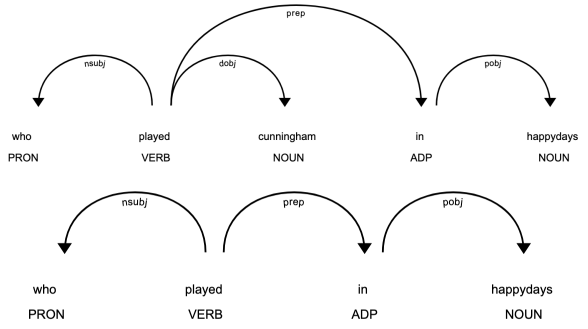

Figure 7: Dependency graph for actor+character vs actor

4 Experiments

4.1 Baseline Methods

I consider 5 conventional ML models for preparing baselines on the dataset.

1. Decision Tree (DT) (Quinlan, 1986): DT are a supervised and non-parametric tree-based method which predicts a label by travelling from root to the leaf. Splitting at any node is based on threshold value of a single feature. *Max-depth* controls the maximum depth till a tree can be split,min-sample-split has a lower

bound on number of samples before splitting and *criterion* gives an option of choosing either gini-impurity or entropy for information gain.

2. Random Forest (RF) (Breiman, 2001): Fits more than 1 decision tree on a subset of dataset in order to reduce overfitting. The parameter *n-estimator* controls the number of DTs. The rest of the parameters are similar to DT.

3. Multinomial Naive Bayes: Naive Bayes uses conditional independence of all features with each other in the Bayes theorem to classify text. It only has 2 parameters, first being the smoothing parameter *alpha* and *fit-prior* which gives option to learn class prior probabilities or use a uniform prior.

4. Support Vector Machine (SVM) (Cortes and Vapnik, 1995): is a supervised learning algorithm that finds a hyperplane dividing all the classes such that the margin between the plane and data points is maximized. The degree parameter defines the degree of polynomial kernel to be fitted.

5. Logistic Regression (LogR) (Wright, 1995): uses a logistic function to model a non-linear data in a linear way. This when combined with stochastic gradient descent calculates the gradient of loss for each sample and updates the parameters. The penalty parameter specifies the norm (L1, L2) and alpha controls the regularization. *Learning-rate* decides how much to change parameters according to the errors.

4.2 Implementation Details

Algorithms: All algorithms are implemented using Python3.7 and open-source scikit-learn[3] (Pedregosa et al., 2011) library. For POS tagging of utterances, nltk[4] (Bird and Loper, 2004) is used. To capture the dependency between words in an utterance spacy[5] dependency parser is used.

Parameter Settings: All experiments are performed on 80% train set and hyperparameters are tuned using Grid Search on the remaining 20% data as validation set. The random seed is set to 1. Table 1 shows the best parameter values of all experiments.

[3] https://scikit-learn.org/stable/
[4] https://www.nltk.org/
[5] https://spacy.io/

Model	Parameters
RF	max-depth:10, n-estimators:50, criterion:entropy, max-depth:10, min-samples-split:5
LogR	alpha: 1e-3 penalty:l2 learning-rate:optimal
SVM	degree:2, kernel:linear
DT	criterion:gini, min-samples-split:5
MultiNB	alpha:0.2, fit-prior: True

Table 1: Hyperparameters of the tuned-models

5 Results

In order to evaluate the performance of the models, we use *Accuracy* as the metric. In a multiclass multilabel problem, a prediction is correct if all the labels are correctly identified for an utterance.

Table 2 shows the results when solved as a multiclass multilabel problem for each class. Labels are converted to One-Hot Vectors (OHV) where each column belongs to a single class i.e. 19 columns. The labels which are present are marked as 1 for each utterance and passed to the BinaryRelevance function of sklearn which trains each model as a binary classification task. Table X shows the performance of all models on the training and validation sets.

Model	Train Acc.	Val Acc.
RF+Tfidf	99.9	72.5
LogR+SGD+Count	97.1	78.7
SVM+Count	84.3	69.4
DT+Count	99.8	73.9
MultiNB+Tfidf	94.6	**83.3**

Table 2: Performance of Multilabel Multiclass Classification models

Next, I approach the problem solely as a multilabel classification problem where the model predicts 1 class for an utterance. This was based on the assumption that the test set has the exact same combination of classes as the train set. Multiple classes in an utterance are considered as a single class which converts the 19 classes into 47 classes. Classes with less than 20 samples are discarded and models are trained on the remaining. Table 3 shows the comparison of various models. Logistic Regression with SGD training performs the best with 90.6% validation accuracy. This is due to the reason that stochastic gradient descent (SGD) tries

to find the optimal parameters using iterative minimisation of gradient function resulting in faster convergence.

Model	Train Acc.	Val Acc.
RF+Tfidf	82.1	75.1
LogR+SGD+Tfidf	99.0	**90.6**
SVM+Count	99.6	88.9
DT+Count	98.1	83.5
MultiNB+Tfidf	94.6	83.3

Table 3: Performance of Multilabel Classification

Oversampling the data to form equal distribution outperform the best model by 6.7%. Predictably, undersampling did not perform well as it reduced the count of already limited available data. Similar results were seen when sentences were concatenated with the NER and POS tags. Out-of-domain data helped in improving performance of that particular class but not the overall model performance.

Model	Train Acc.	Val Acc.
OverSampling	99.2	**96.7**
Undersampling	99.9	86.8
NER	98.2	89.4
POS	99.0	89.9
Out-of-Domain	99.0	89.9

Table 4: Performance of Text processing on best model

6 Conclusion

Relation Extraction is a challenging problem given the limitations of traditional machine learning models and limited availability of annotated data. Skewed class distribution poses another problem as it induces bias while training. A combination of supervised and unsupervised approach is proposed i the work to solve the task of relation extraction. Further, a number of techniques such as out-of-domain data, NER, Over/Undersampling were used to finetune the best model.

Future directions include use of deep-learning based approaches to handle the multiclass multilabel problem without splitting into multiple models.

References

Steven Bird and Edward Loper. 2004. NLTK: The natural language toolkit. In *Proceedings of the ACL Interactive Poster and Demonstration Sessions*, pages 214–217, Barcelona, Spain. Association for Computational Linguistics.

Leo Breiman. 2001. Random forests. *Machine learning*, 45(1):5–32.

Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine learning*, 20(3):273–297.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

J. Ross Quinlan. 1986. Induction of decision trees. *Machine learning*, 1(1):81–106.

Raymond E Wright. 1995. Logistic regression.