Slot Tagging in Knowledge Graphs using Seq2Seq Models

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 an encoder-decoder method for the task of intent detection and slot filling from utterances provided to conversational systems. I evaluate the approach with the conventional machine learning algorithms, deep learning approaches 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.

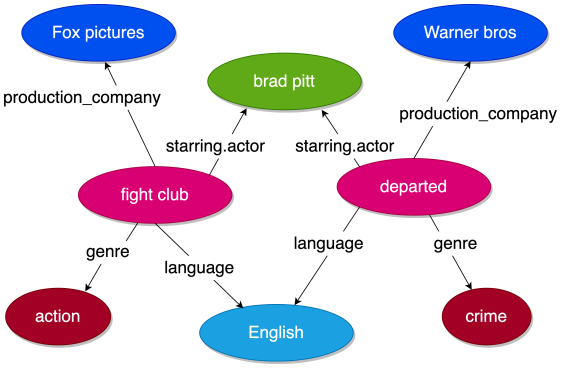

Figure 1: Knowledge graph for movies

Slot filling and Intent detection are two important tasks in any spoken language understanding systems. Intent detection aim to find the intent of a sentence (sentence-level classification problem) whereas slot filling classifies each word in a sentence to a slot-tag (sequence labelling problem). For example in Fig.2, the intent of the sentence *Play the rolling stone's brown sugar* can be "to play a song" whereas slot-filling identifies the important entities such as song, band, singer etc.

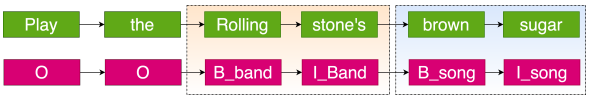

Figure 2: Slot filling in for a sentence (green)

In this work, I provide a comprehensive comparison of machine learning algorithms for slot-filling task 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 classify slot tags for each word in the utterances. The train and test set consists of 2253 and 981 samples respectively and a total of 27 slot tag classes.

As any real-world dataset, this dataset is highly imbalanced with the class `O` having highest frequency of 10243 words followed by `B-movie` with 1094 words and `B-location` with lowest count of only 2 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

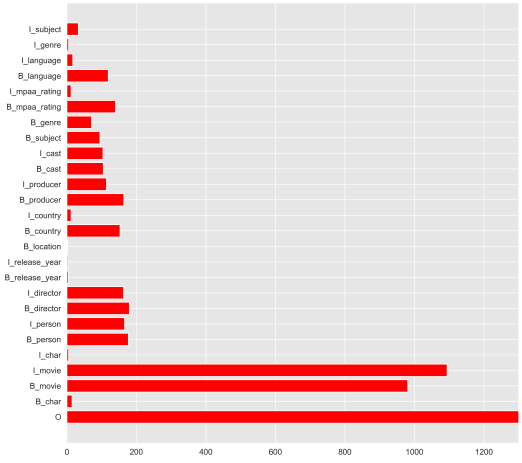

Figure 3: Frequency Distribution of slot-tags in $\mathtt{D}$

are not in present in the train set.

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.

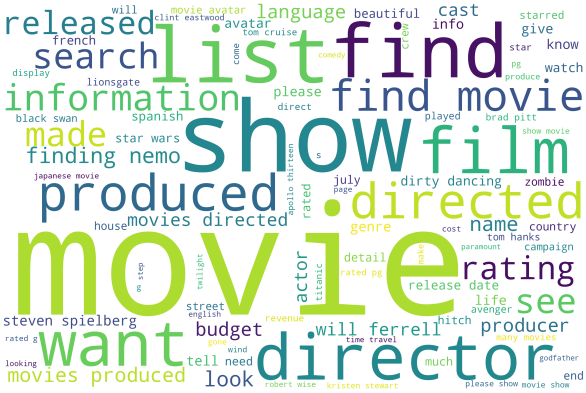

Figure 4: Wordcloud of the Dataset

3 Methodology

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

Task Definition: *Given a labelled training dataset* $\mathtt{D}$ *with a set of utterances where words* $\{w_1, w_2, \ldots, w_n \in u_i\}$ *are labelled with* $\{l_1, l_2, \ldots, l_m \in L\}$ *, the task is to learn a prediction/classification function for an unknown utterance* U *that predicts a label* $l_i \in L$ *for each word in* U*. Here* L *is a set of possible slot-tag labels.*

3.1 Text Processing

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

1. Sent2word: For some algorithms, sentences are split into individual words converting the sequence labelling task into word-level classification problem

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 depending upon the classification algorithm:

1. Word history (H): For word-level classification, features are prepared manually by considering the n-gram history of a word. Bigram and Trigram features are used for experiments. Fig.5 shows the features for the words in utterance *Who plays luke in star wars new hope*. For first word, the history is chosen as $<$ START $>$ token.

	prevprevprev_word	prevprev_word	prev_word	word
0	<START3>	<START2>	<START1>	who
1	<START2>	<START1>	who	plays
2	<START1>	who	plays	luke
3	who	plays	luke	on
4	plays	luke	on	star
5	luke	on	star	wars
6	on	star	wars	new
7	star	wars	new	hope
8	<START3>	<START2>	<START1>	show
9	<START2>	<START1>	show	credits

Figure 5: Trigram history of an utterance

2. Word future (F): Similar to n-gram history, I used future words for additional feature information in an utterance. For last word, the future is chosen as $<$ STOP $>$ token.

3. Tag count (C): A word with a certain slot-tags in the training dataset will be likely to have the same tag in an unknown utterance. For example: if the word *avengers* has the

slot *B-movie* in the train set, then its very less probable that in the test utterance it would have the slot tag *B-character*. Hence, I add the most common tag for each word as a feature.

All tags and words are converted to indices based on the tag or word vocab respectively.

4 Experiments

4.1 Baseline Methods

I consider 6 conventional ML models for preparing baselines on the dataset.

1. Most frequent tag (MFT): To prepare the first baseline model, I calculate the frequency of all tags for each word in the training dataset. During testing, the most frequent tag for that word is selected as the label. For unknown words, the most frequent tag in the training dataset i.e. *O* tag is selected.

2. Decision Tree (DT): 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. The features here can be the history, and (or) future of a word. (See feature extraction)

3. 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. Long Short Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997): Previous works(Kurata et al., 2016; Goo et al., 2018) have used LSTM for sequence labelling tasks as they use a chain-like structure for processing a sequence. However, in order to retain long-term dependencies, LSTM use a gating-mechanism to remember useful information and forget useless ones. It consists of 3 gates: input gate, forget gate, output gate. For sequence labelling, each utterance is padded to a fixed size depending on batch max length. The input is a batch of shape (batch-size × sequence-length) and output is the shape (batch-size × sequence-length × tag-vocab). Cross-entropy is used as the loss function.

5. Encoder-Decoder LSTM (Sutskever et al., 2014) (Enc-Dec): Encoder-Decoder model uses an encoder to encode the input sentence utterance into a single context vector and decoder to decode labels sequentially one word at a time. The output of previous token is used as an input to the next word during testing.

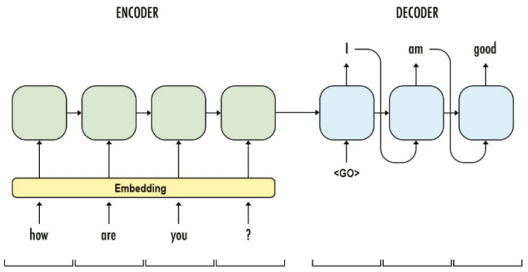

Figure 6: Encoder Decoder Working

6. Conditional Random Fields (CRF)(Lafferty et al., 2001): is a discriminative model that models the conditional probability of a sequence considering the fact that labels are dependent on each other. CRFs are primarily suited to our task (Huang et al., 2015) as BIO tags depend on each other. For ex: if the previous token in `I-movie` then it is very less likely that next token is `B-movie`. For implementation of CRF, I use this implementation. [1]

4.2 Implementation Details

Algorithms: All algorithms are implemented using Python3.7, Pytorch version 1.8 and open-source scikit-learn[2] (Pedregosa et al., 2011) library.

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

5 Results

In order to evaluate the performance of the models, we use *Accuracy* as the metric. In a sequence

[1] https://github.com/threelittlemonkeys/lstm-crf-pytorch

[2] https://scikit-learn.org/stable/

Model	Parameters
LogR	alpha: 1e-3 penalty:l2 learning-rate:optimal
DT	criterion:gini, min-samples-split:5
LSTM	lr:0.001, num-layers:2 embed-dim: 300 bs:32
CRF	lr:0.0002, num-layers:2 embed-dim: 300 bs:64
Enc-Dec	Enc/Dec-embed-dim:256 layers:1 dropout:0.5 lr:0.001

Table 1: Hyperparameters of the tuned-models

labelling problem, accuracy is calculated by word-level slot tag matching rather than sentence-level matching. The number of words now become 6415 for accuracy calculation.

Table 2 shows the results when as a word-level classification task. Words are converted to One Hot Vectors (OHV) resulting in a 6378 dimension vector for each word. Each word is classified into one of the 27 slot classes. Table X shows the performance of all models on the training and validation sets. We can see that the baseline MFT model shows the best performance on the validation. However, this approach does not work in test set where many unknown words are present. The next best model i.e. Logistic Regression with SGD performs best on the test set.

Model	Train Acc.	Val Acc.
MFT	98.4	**98.1**
DT+H	98.9	90.2
DT+H+F	99.8	91.8
LogR+SGD+H	97.5	94.1
LogR+SGD+H+F	**99.5**	95.8
LogR+SGD+H+F+C	99.4	95.8
LSTM+H+F	98.9	97.1

Table 2: Performance of word-level labelling models

Next, I approach the problem solely as a sequence labelling problem where the model predicts n slot classes for an utterance with n words. From Table. 3 we can observe the difference in performance of sequence labelling models. Surprisingly, LSTM gives the best performance with a validation accuracy of 97.6 which is comparable to the CRF model. The encoder-decoder LSTM model is unable to capture the dependencies between labels resulting in poor performance. This may be due to less availability of training data. Another reason might be the high variation in the vocabulary at train vs test time, implying most word tokens labelled as UNK during testing input.

Model	Train Acc.	Val Acc.
LSTM	**99.8**	**97.6**
CRF	99.7	96.1
Enc-Dec	95.1	93.1

Table 3: Performance of Sequential Labelling models

6 Conclusion

Slot Filling and Intent Detection are challenging problems 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. In this work, I show how traditional models can outperform deep-learning and label-dependent models such as LSTM, CRF and Encoder-Decoder in low-resource scenarios.

Future directions include use of attention based deep-learning approaches and transformer based pretrained embeddings to handle the sequence labelling problem.

References

Chih-Wen Goo, Guang Gao, Yun-Kai Hsu, Chih-Li Huo, Tsung-Chieh Chen, Keng-Wei Hsu, and Yun-Nung Chen. 2018. Slot-gated modeling for joint slot filling and intent prediction. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 753–757.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

Gakuto Kurata, Bing Xiang, Bowen Zhou, and Mo Yu. 2016. Leveraging sentence-level information with encoder lstm for semantic slot filling. *arXiv preprint arXiv:1601.01530*.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data.

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.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Raymond E Wright. 1995. Logistic regression.