Slot Tagging and Intent Prediction using Transfer Learning

Kartik

1 Task Description

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 joint-finetuning model for the task of intent detection and slot filling from utterances provided to conversational systems. I evaluate the approach with a variety of transformer models with and without joint learning procedure.

2 Introduction

Natural language Understanding (NLU) is a 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. Slot filling (SF) and intent detection (ID) are useful for task-oriented dialogue system. For example, for an utterance like *"Buy a concert ticket for Coldplay in Seattle"*, intent detection works on sentence-level to indicate the task is about buying a concert ticket, while the slot filling focus on words-level to figure out which concert and what place to book ticket for.

In this work, I provide a comprehensive comparison of large pretrained models like BERT, DistilBert, AlBert, etc. for slot-filling and relation prediction 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.

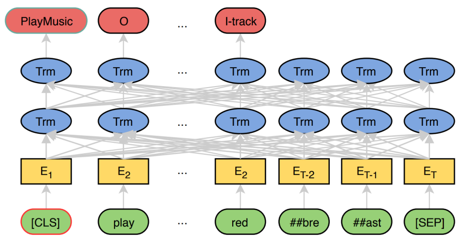

Figure 1: Example from Dataset

3 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.

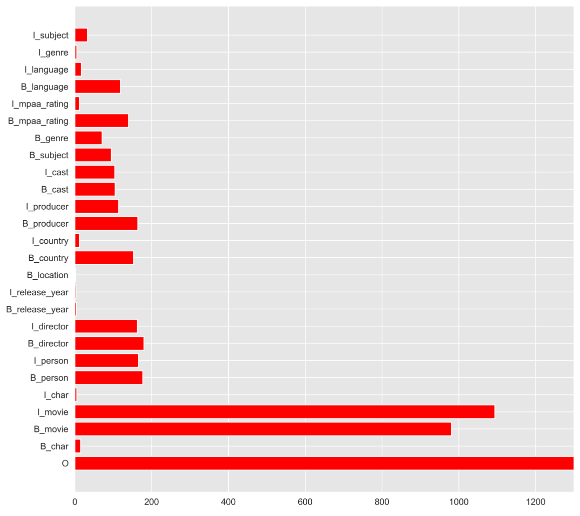

Figure 2: Frequency Distribution of slot-tags in D

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. Similarly for relations, class `movie.starring.actor` having highest frequency of 340 samples and `movie.locations` with lowest count of only 3 samples. Fig.2 and 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.

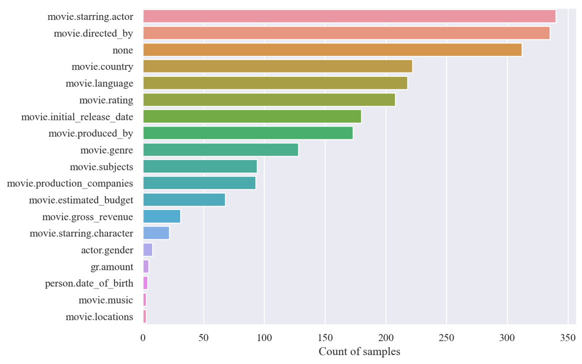

Figure 3: Frequency Distribution of core relations in `D`

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.

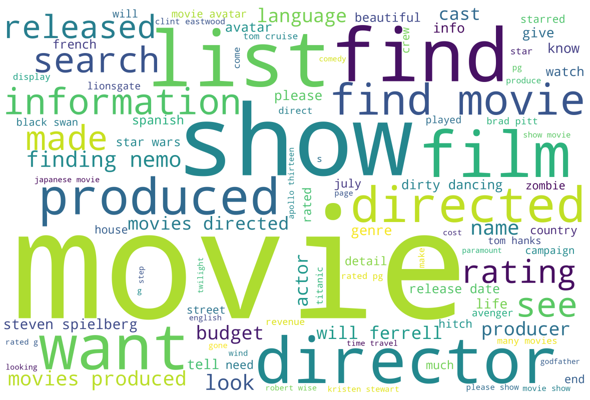

Figure 4: Wordcloud of the Dataset

The input of the model would look like this: **star of twilight**. The output of the model would look like this:

1. Slot labels: O O B-movie
2. Relation: movie.starring.actor

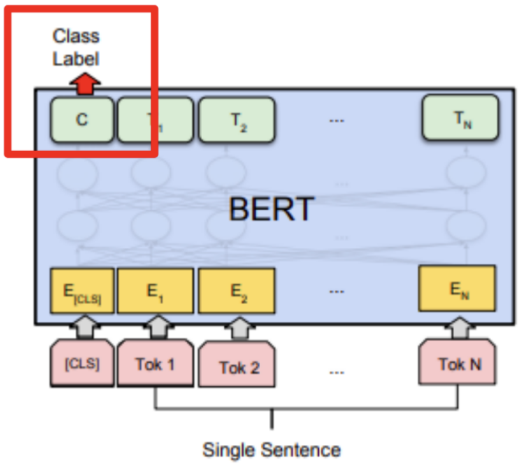

Figure 5: Bert for task-1

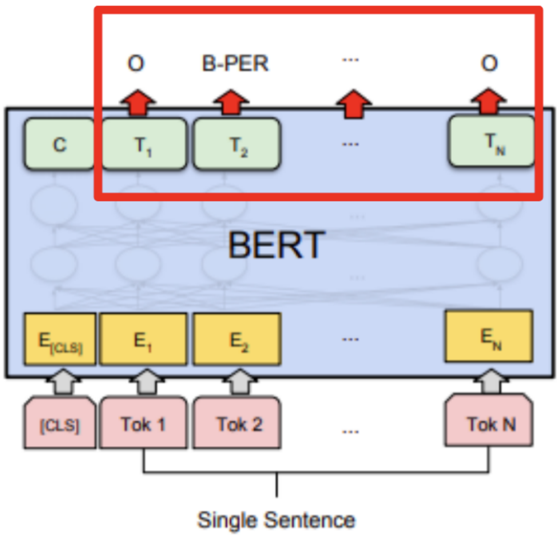

Figure 6: Bert for task-2

4 Methodology

This section describes the process of extracting features from the sentences and methodology adopted for both tasks.

Task Definition: *Given a labelled training dataset* D *with a set of utterances* $\{u_1, u_2, \ldots, u_n\}$ *and a pretrained model* P *, the task is to finetune the model* P *to learn two classification functions, one for predicting sentence labels* $r_i \in R, i \in [1, |R|]$ *and other for predicting word labels* $l_j \in L, j \in [1, m]$ *in an unknown utterance, where* R *is the set of core relations,* L *is the set of slot labels,* m *is the number of words in utterance. Note that a single utterance can have multiple possible core relations.*

Text Processing : As the provided train and test data was free of punctuation, all lowercase and cleaned no text preprocessing is done on the dataset.

Tokenization : This involves splitting the sentences into words or subwords depending upon the tokenization scheme. For this assignment, subword tokenization is used which splits a each word in the sentence into multiple words depending on the frequency of words in pretraining dataset. For example, the word *short* is broken into *short* and *##cut*. # is used to indicate the prefix of a word.

5 Models

The following pretrained models are used for implementation:

BERT : BERT (Devlin et al., 2018) is a bidirectional transformer encoder model trained with a masked language model objective to predict masked words in a sentence and a sentence-level language model objective to predict next sentence given current sentence. Fine-tuning bert just requires us to add a linear layer out of the final CLS embeddings (for CRP task).

DistilBERT (Sanh et al., 2019): is a lighter and faster version of BERT obtained by knowledge distillation of BERT using a student teacher training method. This model can give 95% of bert's performance with 40% less parameters and 60% faster speed.

Albert (Lan et al., 2019): This model is also a smaller version of BERT but with different training objective than DistilBERT. It uses a pretraining loss to predict the ordering of two consecutive segments. Further, Albert uses 2 parameter reduction techniques for scaling: a factorized embedding parameterization and cross-layer parameter sharing.

6 Experiments

All experiments are performed using Huggingface library [1] using Pytorch version 1.10.2 on Nvidia GeForce RTX 3080. Further, open source libraries such as pandas and sklearn are used during the training and inference pipeline. The dataset is split into 80% train and 20% validation set using a fixed random seed of 1234 for all experiments.

Evaluation: For performance testing of all models for both tasks, metrics such as Accuracy, F1, Precision and Recall is used. Since BERT is trained on MLM objective, the output of CLS token is used for relation prediction and hidden state outputs is used for slot prediction. For DistilBERT, last hidden state is used to project into the label dimension and make corresponding predictions based on the task. The following training objectives are experimented:

Single-label Multiclass Classification (SMC): In this, label of each utterance is considered as a single label even if has multiple labels annotated with it. For example, the label of *who plays luke on star wars new hope* is represented as one label joined with a # i.e. *movie.starring.actor#movie.starring.character*. This when done on complete dataset leads to 47 unique labels. Although on a large dataset, this objective cannot scale due to two reasons: first, that the combination of possibilities can be exponentially large and second, that the model can only predict the combination present in the dataset. Table. 1 shows the results on 3 models.

Model	Acc	F1	Loss
DistilBERT	**96.9**/91.2	**95.7**/89.3	0.16/0.48
BERT	96.5/**92.4**	95.5/**90.75**	**0.16/0.40**
Albert	96.6/90.7	95.4/88.6	0.18/0.52

Table 1: Performance of Singlelabel Multiclass Classification models (Train/Val

Model	Parameters
SMC	batch-size:4, dropout:0.1, criterion:cross-entropy, n-epochs:10, learning-rate:1e-5
MMC	batch-size:16, dropout:0.1, patience:3 , criterion:BCEwithLogits, n-epochs:20, learning-rate:1e-4
Slot prediction	batch-size:16, warmup-ratio:0.1 criterion:cross-entropy, n-epochs:15, learning-rate:1e-4
Mutlitask	batch-size:1 criterion:cross-entropy, n-epochs:10, learning-rate:0.001, dropout:0.25

Table 2: Hyperparameters of the tuned-models

Multilabel Multiclass classification (MMC): In this setup, instead of treating multiple labels of an utterance as single label, the labels of all utterances are one-hot encoded using sklearn Multilabel Bina-

[1] https://huggingface.co/

rizer. Each label is now a one hot vector of size 19, with 1 at places where the corresponding label is true and 0 otherwise. For modeling, the feature extractor class of Huggingface (AutoModel) is used and a linear layer is used to project the outputs to vector length (19). Instead of using cross entropy and softmax to get the best output, BCEWithLogitsLoss is used with first passes the logits to sigmoid and then calculates CE loss. During prediction, all logits with positive value (or sigmoid > 0.5) is treated as 1 and 0 otherwise. Table. 3 shows the results on 3 models.

Model	Train Acc	Val Acc	T-Loss	V-Loss
DistilBERT	**98.0**	**93.5**	2.63	6.67
BERT	97.0	94.4	3.1	7.8
Albert	89.5	95.0	8.2	5.2

Table 3: Performance of Multilabel Multiclass Classification models

Slot prediction : This task involves predicted a sequence of labels (BIO tags) for each word in the sequence. Hence, for each utterance the number of words should equal number of slot predictions. Since, the models used here do subword tokenization, the predictions would mismatch the required slots. Also, during training each subword needs to be assigned a label. For experiments, I assigned the label of the main word to all subwords. Assigning -100 label to consecutive subwords apart from first led to slightly lower results hence, this strategy is not used in experiments. Table. 5 shows the results on different models. All models are run using AutoModelforTokenClassification in Huggingface.

Model	Acc	F1	Loss
DistilBERT	**96.9**/91.2	**95.7**/89.3	0.16/0.48
BERT	99.1/98.5	95.7/**94.2**	**0.04/0.09**
Albert	98.3/97.0	93.8/90.2	0.05/0.17

Table 4: Performance of Slot tagging models

Multitask learning: In this training objective, rather than training each task seperately, both slot prediction and relation prediction are done at the same time using a single model end-to-end. For this experiment, only BERT model is used as it provides a CLS token and hidden states corresponding to each token as well. The CLS token encodes the sentence level information while the hidden states

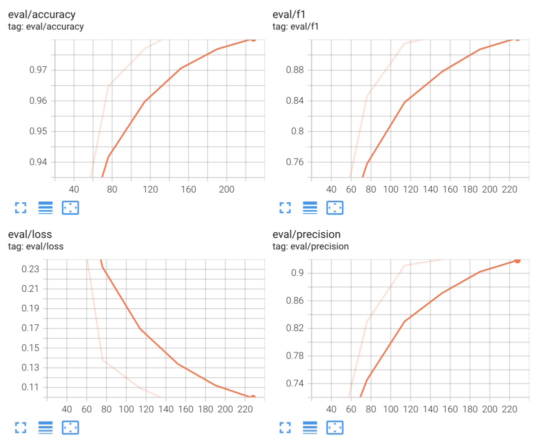

Figure 7: Tensorboard plots for BERT Slot prediction

contain information for each token. Hence, CLS token is used for relation prediction and hidden states are used for slot prediction. CrossEntropy loss is used for both at token level for slots and at sentence level for intent. Both losses are added and optimized using retain-graph parameter. Table **??** shows the results for experiment after s

Model	Acc	Loss
BERT-Slot	95.6/93.5	0.06/0.22
BERT-Relation	94.6/90.2	0.07/0.31

Table 5: Performance using multitask learning

7 Analysis

After thorough analysis of the experimental results on various models and tuning with different hyperparameters, surprisingly, single model performed better than joint models. Comparing SMC vs MMC, SMC resulted in better validation scores maybe due to less size of the training dataset.

References

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.