Joint Training of Classification and Similarity Models for Intent Detection in Task-specific Chatbot

Anonymous EMNLP submission

Abstract

Task-specific chatbot systems have gained many important applications, such as smart speaker, customer service system. One fundamental module behind them is detecting intent of a user's input, and can be modeled as a short text classification problem. However, in the early stage of building a chatbot, collecting enough labeled data for hundreds of thousands of user intents is expensive. Popular classification models, direct mapping a query to an intent, have a high precision, while depending on enough task-specific labels information. In comparison, similarity models, modeling similarity of two queries instead, can utilize additional out-of-domain data, while having a relatively lower precision, due to the discrepancy of similarity loss and real classification loss. In this work, we propose a novel model, called similarity model fused with classification model (SFC), to combine the merits of the two kinds of models in the framework of multi-task training. Our extensive experiments on 6 public and 1 private datasets demonstrate that our systems outperform very strong baselines (i.e., RoBERTa based pretrained model, joint model with NER), especially with insufficient data.

1 Introduction

Task-specific conversational chatbot (Wen et al., 2016) has been applied into many practical products. A popular one is the smart speaker, e.g. Alex, Siri, Google home. Another important one is the customer service system, which greatly help human agents handle miscellanious customer's questions. No matter these applications involves single- or multi-round conversations, a critical step is to identify the intent behind a user's question or response. The detected intent with its associated attributes is then mapped into a predefined dialog logic to obtain a suitable response to return to customers.

In the early stage of building a chatbot , sufficient labeled data is often expensive to obtain to render the system to achieve a strong performance. People thus have to filter out enough typical users' utterances from tons of real conversation logs, and label them with proper intents. In practice, this procedure can be improved by active-learning like operations. A basic intent detection system with very limited manually labeled data can be built first, and then it filters out a set of potential data with high confidences for humans to label. The new data is added to train a better system again. This procedure iterates until the system reach a high performance. Therefore, it is meaningful to address the challenge of short-text classification (Sriram et al., 2010; Chen et al., 2019a; Phan et al., 2008; Yan et al., 2009; Hua et al., 2015) problem under the few-shot setting (Yu et al., 2018).

Existent approaches for intent detection can be roughly categorized as text classfication model and text similarity model.

The first one, *text classification model*, includes a variety of work. From traditional machine learning models like SVM (Suykens and Vandewalle, 1999), boosting tree (Tu, 2005), to neural networks (Wen et al., 2016), such as convolutional neural networks (CNNs) (Kim, 2014; Zhang et al., 2015; Conneau et al., 2016) and long short term memory networks (LSTMs) (Mousa and Schuller, 2017; Liu et al., 2016), and then to the most popular pretrained language models based, such as BERT (Devlin et al., 2018), RoBERTa (Liu et al., 2019c) and etc. Especiallaly, pretrained models based (Vaswani et al., 2017) tends to be more helpful in the few-shot scenario (Yu et al., 2018; Madabushi et al., 2020) to alleviate the dearth of training data. Another two interesting lines of work, label-word joint models, and joint NER and classification are discussed in the related work of appendix.

The second one, *text similarity model*, is usually employed to calculate how similar between an input text and a historical text in the repository. The associated label of the most similar historical text is returned as the label of the text to query (Jafarpour et al., 2010; Leuski and Traum, 2011). A popular and effective methodology is adopting pretrained models to model the similarity calculation as a binary classification problem.

Despite plenty of success of these two methodologies, they still have some limitations in this task especially when data is insufficient. Regarding the *text classification model*, it learns a function that directly maps an input query into its expected label. Its training requires data in the form of a pair of a query and a task dependent label. It is quite hard to adopt labeled data from other domains, as their label definitions are often incompatible to each other. Though we can use some data augmentation methods to alleivate the paucity of data, e.g., translation based paraphrasing methods, the resulting benefit is still limited because of the data being homogeneous, in the case that our labeled data is very insufficient. Regarding the *similarity model*, the foremost advantage is it does not pose any restriction on the label definition of user intents, but aims to learn a function that measure how similar of two sentences are. Then in the intent detection task, we are only concerned which labeled query is the most similary one to the current input, and then use its label as the output. This results in a possibility that even though the labeled data in current domain is scarce, we may borrow additional data from another domain to help enhance the similarity model to make a better intent prediction. Nevertheless, just as every coin has two sides, the high flexibility of similarity model leads to its worse performance compared to a classification model, because its training loss differs from a classification loss corresponding to the intent detection goal. More, it requires an auxiliary model to narrow down candidate labeled queries to speed up the calculation, which makes it slow in the speed and tricky in the auxiliary model selection.

The above limitations motivate us to propose a system that may take both the high performance from a classification model and the ability of supporting out-of-domain data from a similarity model. We call our system as SFC, short for similarity model fused with classification model, shown in Fig. 1. In order to train effectively, we further borrow the multi-task learning (Caruana, 1993; Collobert and Weston, 2008; Liu et al., 2019a). Our basic idea come naturally, and the first impementation consists of two stages. In the first stage, we use an auxiliary model to select top-K most possible labels for an input. This model can be an elastic search (Divya and Goyal, 2013) or a text classification model trained on current domain. In the second stage, we build a classification model which is composed of several similarity modules. Then, this structure derives two goals to train towards, the task-specific classification loss , and the similarity loss on both in-domain and out-of-domain data. In this version, the two stages are independently optimized, so we call it *2-stage SFC*.

We further find that the quality of outputs from the first stage might limit the final performance of the system, since those outputs are fixed in the whole system training procedure. This observation motivates us to continue improving the above 2-stage SFC into a joint training setting, and we call this second version as *joint-SFC*. In this way, the first stage model can be optimized by the optimization in the second stage so that it can provide better candidates to improve the performance of the second stage in turn.

Since SFC is in the framework of multi-task training, its inheriently supports adding additional tasks, such as NER, to gain further improvement.

2 Methodology

In this section, we describe our proposed SFC method, which includes the basic 2-stage SFC and its more compact version joint-SFC.

2.1 2-stage SFC

It consists of the following two independent stages in charge of different jobs.

Stage 1: classification model for providing top-K candidate class labels

We can use any auxiliary text classification model or tool, such as Term Frequency-Inverse Document Frequency (TF-IDF)retrieval, to provide top-K most related class labels. These selected class labels will later be used to sample sentence pairs of both positive and negative samples for stage 2.

In this work, we choose pretrained models to fine tune on our data. Due to the excellent performance of RoBERTa in this task, compared to other pretrained models, we use RoBERTa as the encoder of classification model in our setting.

Given a data point x_i with class label y_i from dataset $\mathcal{D}$, we take the final hidden state h_i of the first token [CLS] encoded by RoBERTa as the representation for the whole sequence of x_i. Then a

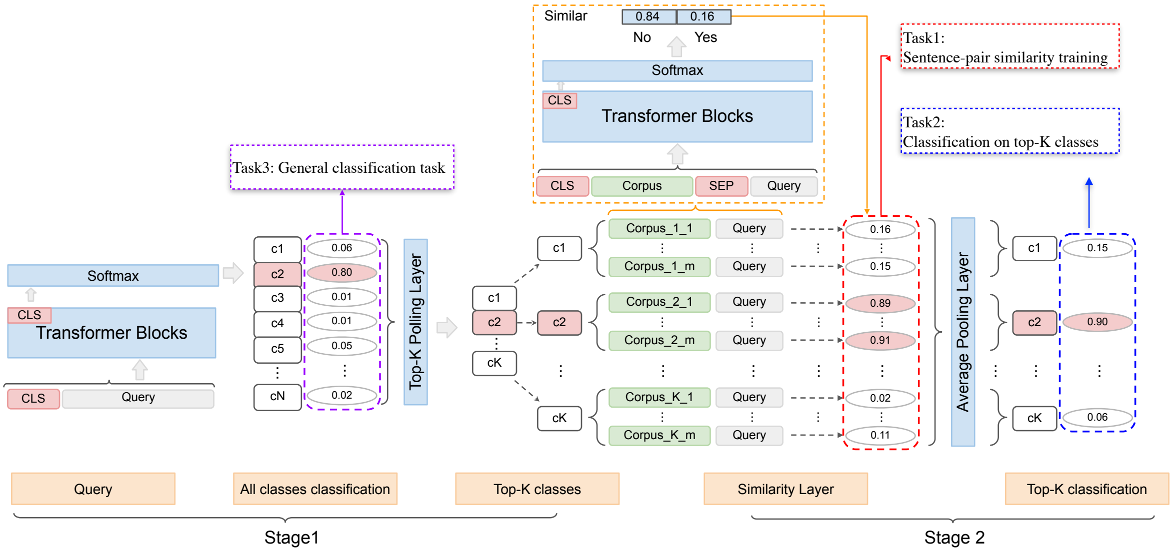

Figure 1: **Network Structure of SFC:** 2-stage SFC and joint-SFC are sharing the same network from stage 1 and stage 2, with the only difference whether two stages being jointly trained.

linear layer is followed to output probabilistic distribution of class labels, $softmax(W^C h_i + b^C) = softmax(\Phi_i^C)$, where W^C and b^C are trainable parameters. Then, the loss in stage 1 classification model is shown in Equ. 1,

$$\mathcal{L}^C = \frac{1}{N}\sum_{i=1}^{N} -log(\frac{exp(\Phi^C_{i,y_i})}{\sum_{j=1}^{C} exp(\Phi^C_{i,j})}) \tag{1}$$

Stage 2: sentence-pair similarity model based classification with multi-task learning

We continue choosing RoBERTa as the main module in this stage that identifies the class label with strongest semantic similarity to the user input sentence.

Suppose we have a training dataset $\mathcal{D} = \{x_i, y_i\}_{i=1}^N$ of N data points, in which x_i is the user input query and y_i is a single class label from a class label set of C classes in total. We generate a sentence pair dataset $\mathcal{D}' = \{[(x, x')_j, l_j]\}_{j=1}^M$ from $\mathcal{D}$, where $(x, x')_j$ is a pair of two history queries [1], and $l_j \in \{0, 1\}$ denotes the similarity label that only the ones from the same class label are considered similar. The size of $\mathcal{D}'$ would be up to $O(N^2)$ if no any sampling strategy is used. This is quite time-consuming for the training in stage 2 because N can be several hundreds or thousands even under few shot setting in task-specific chatbot scenario.

Therefore, we adopt the idea of adversarial negative sampling strategy from the work (Bamler and Mandt, 2020). The key idea is to train with negative samples that are hard to be distinguished from positive samples. Now suppose we have a user input query x_i and top-K most related class labels set $\mathcal{C}' = \{c'_1, \ldots, c'_K\}$ provided by auxiliary model or tool in stage 1, where K is a hyperparameter to control candidate class number, we apply the adversarial negative sampling strategy into our sentence pair sampling process by the following two steps:

The first step is *positive sentence pair sampling*. During the training process, we first make sure the ground truth class label y_i for x_i is within the candidate class label set $\mathcal{C}'$. If $y_i \notin \mathcal{C}'$, we manually add y_i into $\mathcal{C}'$ by replacing c'_K with y_i, since c'_K is the least promising candidate label according to the auxiliary model in stage 1. Afterwards, we will randomly sample P sentences $\{x'_1, \ldots, x'_P\}$ with the same class label as y_i from dataset $\mathcal{D}$ to form the set of sentence pairs with positive label $\mathcal{P'}_i = \{[(x_i, x'_1), 1], \ldots, [(x_i, x'_P), 1]\}$, where P is also a hyperparameter set to control the number of sentences we should sample from each class.

The second step is *negative sentence pair sampling*: As for negative sentence pairs, we will also randomly sample P sentences for each class in the negative candidate class label set $\mathcal{C}' \backslash \{y_i\}$. The class labels in $\mathcal{C}' \backslash \{y_i\}$ are the set of most confusing class labels comparing to the ground truth y_i, so we assume that the sentence pairs with negative label grouped by user input query x_i and sentences with class labels in $\mathcal{C}' \backslash \{y_i\}$ are strong adversarial negative samples for sentence-pair sim-

[1] For simplicity, we use query to denote a user's question or a response.

ilarity model that can help enhance the training speed and performance. According to the same method as positive sentence pair sampling, we can obtain the set of sentence pairs with negative label as $\mathcal{N'}_i = \{[(x_i, x'_1), 0], ..., [(x_i, x'_{P.K}), 0]\}$

In this way, we generate the sentence pair dataset $\mathcal{D}'$ for stage 2 based on the top-K class label set $\mathcal{C}'$ provided by stage 1 as $\mathcal{D}' = \{\mathcal{P'}_i \cup \mathcal{N'}_i\}_{i=1}^{N}$, in which we have $M = N \cdot K \cdot P$ data points of sentence pairs in total.

Before starting to fine-tune our sentence-pair model on task-specific dataset, we first fine-tune RoBERTa on Quora dataset (Iyer et al., 2017), which contains 404,290 potential duplicate question pairs, for transfer learning. This is also the merit of similarity based model, as labeled task-specific classification data is hard to acquire, yet general sentence pairs with similar semantics can be much easier to acquire.

Multi-task based training

We continue to do multi-task training on sentence-pair dataset $\mathcal{D}'$ sampled from the top-K candidate class labels provided by stage 1. In stage 2, we have two tasks tuning our system.

The first task is *regular sentence-pair similarity task*. For sentence pair semantic similarity task, given a data point $(x, x')_i$ with similarity label l_i from data set $\mathcal{D}' = \{[(x, x')_i, l_i]\}_{i=1}^{M}$, Roberta takes the final hidden state h_i of the first token [CLS] as the representation for the sequence of packed sentence pair $(x, x')_i$. Let's suppose we have a linear layer as $\Phi_i^S = W^S h_i + b^S$, where W^S and b^S are trainable parameters, and the probability score for $(x, x')_i$ can be calculated as $p_i^S = softmax(\Phi_i^S)$.

Due to the fact that sentence pairs within the same class is much fewer than that from different classes, the dataset $\mathcal{D}'$ is quite imbalanced. Therefore, we will accommodate a weight variable $w^S = [K-1, 1]$ to the loss to eliminate the bias brought by data imbalance, shown in Equ. 2.

$$\mathcal{L}^S = \frac{1}{M}\sum_{i=1}^{M} -w_{l_i}^S \cdot log(\frac{exp(\Phi_{i,l_i}^S)}{\sum_{j=0}^{1} exp(\Phi_{i,j}^S)}) \quad (2)$$

The second task is *classification on top-K classes*. As minimizing similarity loss is not our final goal in the intent classification, we bring back classification loss again. In this task, the network is based on sentence pair similarity modules. The only difference is that we add a task-specific average pooling layer, shown in Fig. 1 to accomplish the classification task based on sentence-pair model.

We already know that the size of Φ^S in task 1 is $M \times 2 = (N \cdot K \cdot P) \times 2$, where N is the total number of data points in original training data $\mathcal{D}$, K is a hyperparameter that controls the number of candidate class labels provided by stage 1, and P is also a hyperparameter that controls the number of sentences we should randomly sample from each candidate class labels. Now, for the average pooling layer, we first reshape Φ^S into the size of $N \times K \times P \times 2$, and then split it into $\Phi^{K,pos}$ and $\Phi^{K,neg}$, and both of them will have the size of $N \times K \times P$. In this way, $\Phi^{K,pos}$ can represent the level of similarity for each sentence pair, and in the mean time, $\Phi^{K,neg}$ can represent the level of dissimilarity for each sentence pair. Then, we can do average pooling for each candidate class among the top-K classes as shown in Equ. 3.

$$\xi_{i,j}^{K,pos} = \sum_{p=1}^{P} \Phi_{i,j,p}^{K,pos} \qquad \xi_{i,j}^{K,neg} = \sum_{p=1}^{P} \Phi_{i,j,p}^{K,neg} \quad (3)$$

With the average pooling result $\xi^{K,pos}$ and $\xi^{K,neg}$, we also generate a top-K class label $l_i^K \in [1, \dots, K]$ for each $\xi_i^{K,pos}$ and $\xi_i^{K,neg}$. The loss for top-K classification task is shown in Equ. 4. In Equ. 5 and Equ. 6, the first term $\xi_{i,l_i^K}^{K,pos}$ and $\xi_{i,l_i^K}^{K,neg}$ encourage high level of similarity and low level of dissimilarity for prediction of the correct label l_i^K within the top-K candidate classes.

$$\mathcal{L}^K = \frac{1}{N}\sum_{i=1}^{N}(\mathcal{L}_i^{K,pos} - \mathcal{L}_i^{K,neg}) \quad (4)$$

where

$$\mathcal{L}_i^{K,pos} = -log(\frac{exp(\xi_{i,l_i^K}^{K,pos})}{\sum_{j=1}^{K} exp(\xi_{i,j}^{K,pos})}) \quad (5)$$

$$\mathcal{L}_i^{K,neg} = -log(\frac{exp(\xi_{i,l_i^K}^{K,neg})}{\sum_{j=1}^{K} exp(\xi_{i,j}^{K,neg})}) \quad (6)$$

Finally, the overall loss function for multi-task learning in stage 2 is shown in Equ. 7. The training objective of stage 2 is to minimize the weighted sum of task-specific losses. Here α_S and α_K are weights of task 1 and task 2 respectively.

Models	CLINC150						BANKING77						HWU64						ITG	Amazon-670k
	5	**10**	**15**	**20**	**30**	**50**	**5**	**10**	**15**	**20**	**30**	**50**	**5**	**10**	**15**	**20**	**30**	**50**	**3-fold**	**3-fold**
TextCNN (classification)	0.5318	0.6963	0.7609	0.8142	0.8526	0.8867	0.4408	0.6436	0.7366	0.7918	0.8228	0.8619	0.3112	0.4007	0.4823	0.5272	0.5782	0.6262	0.6624	0.4401
LEAM (classification)	0.7514	0.8203	0.8612	0.8802	0.9010	0.9180	0.5422	0.7812	0.8129	0.8280	0.8610	0.8727	0.4545	0.5554	0.5936	0.6599	0.6855	0.7046	0.7086	0.6091
BERT-large (classification)	0.8080	0.8904	0.9265	0.9334	0.9497	0.9595	0.5780	0.8004	0.8518	0.8827	0.8858	0.8982	0.4711	0.5963	0.6342	0.7010	0.7117	0.7424	0.7485	0.6658
ALBERT-xxlarge (classification)	0.8497	0.9008	0.9296	0.9297	0.9466	0.9578	0.5549	0.7981	0.8231	0.8530	0.8571	0.9096	0.4879	0.6116	0.6135	0.6996	0.7094	0.7376	0.7253	0.6893
RoBERTa-base (classification)	0.8732	0.9254	0.9363	0.9482	0.9558	0.9637	0.7305	0.8654	0.8808	0.9080	0.9061	0.9293	0.5831	0.6790	0.7064	0.7100	0.7320	0.7472	0.7734	0.6708
RoBERTa-large (classification)	**0.8974**	**0.9372**	**0.9508**	**0.9584**	**0.9621**	**0.9733**	**0.7690**	**0.8728**	**0.8966**	**0.9099**	**0.9227**	**0.9313**	**0.6044**	**0.7002**	**0.7129**	**0.7436**	**0.7493**	**0.7678**	**0.7990**	**0.7156**
RoBERTa-large (similarity)	0.8266	0.8861	0.9023	0.9084	0.9090	0.9407	0.757	0.8476	0.8614	0.8743	0.8749	0.8980	0.5425	0.6164	0.6503	0.6729	0.6947	0.7231	0.7418	0.6362
2-stage SFC (task1)	0.8979	0.9457	0.9517	0.9591	0.9610	0.9664	0.7975	0.8818	0.8962	0.9109	0.9187	0.9198	0.6477	0.7055	0.7200	0.7232	0.7484	0.7653	0.7972	0.7189
2-stage SFC (task2)	0.9162	0.9424	0.9530	0.9617	0.9633	0.9690	0.7997	0.8823	0.8945	0.9123	0.9236	0.9317	0.6498	0.6980	0.7202	0.7358	0.7498	0.7657	0.8020	0.7311
2-stage SFC (task1 + task2)	0.9167	0.9456	0.9571	0.9638	0.9658	0.9753	0.8135	0.8854	0.8931	0.9192	0.9257	0.9339	0.6525	0.7092	0.7168	0.7476	0.7519	0.7696	**0.8124**	0.7364
Joint-SFC	**0.9231**	**0.9560**	**0.9644**	**0.9669**	**0.9712**	**0.9821**	**0.8270**	**0.9069**	**0.9103**	**0.9209**	**0.9323**	**0.9463**	**0.6697**	**0.7211**	**0.7254**	**0.7497**	**0.7593**	**0.7772**	0.8114	**0.7445**

Table 1: F1 scores on five task-specific datasets for text classification in chatbot under low resource. For ITG, we keep the full dataset. For Amazon-670k, we randomly sampled 250 classes with training sample numbers within 5-15 samples per class. For CLINC150, BANKING77, HWU64, we set up various few-shot settings (5/10/15/20/30/50 samples per class) while keeping the test set to be fixed. The highest scores among all the baseline models and SFC variants for each data setting are both marked in bold.

$$\mathcal{L} = \alpha_S \mathcal{L}^S + \alpha_K \mathcal{L}^K \tag{7}$$

2.2 Joint-SFC

In 2-stage SFC, Stage 1 and stage 2 are separate that there is no deep interaction with each other during training. In this case, the performance of stage 1 may limit the potential of stage 2; meanwhile, stage 2 also cannot give training feedback back to stage 1 for fine-tuning. Therefore, to further improve the overall performance of SFC, we proposed a joint model structure, shown in Fig. 1.

In joint-SFC, classification model is being placed in lower layer level to dynamically provide top-K candidate class labels for the sentence-pair similarity model placed in higher layer level through a top-K pooling layer. In this way, classification model and similarity model can be merged into one single joint-model for multi-task training with sentence pairs sampled from varying candidate class labels, thus avoiding the limitation brought by 2 separate stage structure.

There are 3 tasks in total during the training process of joint-SFC, shown in Fig. 1, and the overall loss is also the weighted sum of all the task-specific losses, shown in Equ. 8. Here, α_S, α_K and α_C represent the weights for task 1 sentence-pair similarity in Equ. 2, task 2 top-K classification in Equ. 4, and task 3 general single sentence classification respectively in Equ. 1.

$$\mathcal{L} = \alpha_S \mathcal{L}^S + \alpha_K \mathcal{L}^K + \alpha_C \mathcal{L}^C \tag{8}$$

3 Experiments

3.1 Datasets

Our experiments aim to study the short text classification in the low-resource environment popular in the task-specific conversational chatbot application. The datasets adopted in our experiments, showed in 2 [2], typically contain comparatively large number of class labels, ranging from several dozens to hundreds, and each class label is associated with a handful of labeled queries with each query being usually one sentence.

ITG, is a proprietary FAQ dataset from real-world chatbot project, which is composed of question and answer pairs about online English teaching. It contains 3,938 sample questions for 228 class labels, and each class label corresponds to a unique answer.

Amazon-670K, is a customer product review dataset for text classification task from the extreme classification repository. The complete dataset contains 670,091 class labels, 285,176 training samples and 150,875 testing samples (Bhatia et al., 2016). As each sample may correspond to multiple class labels, we keep only the first one. We further filter out the class labels that only associated to training samples within the amount of 5 to 15 to mimic the few-shot chatbot scenario. From them, we sample 250 class labels as well as their training samples to form a subset with 2658 samples.

HWU64, is an intent detection dataset designed

[2] All settings of six public datasets would be released to github soon.

Dataset	Domain	#Class	#Training Samples	#Samples/Class	Settings
ITG	FAQ chatbot	228	3938 * 0.7	12	3-fold
Amazon-670k	Product review	250	2658 * 0.7	8	3-fold
HWU64	Intent detection	64	[320, 640, 960, 1280, 1920, 3200]	[5, 10, 15, 20, 30, 50]	data sampling
CLINC150	Intent detection	150	[750, 1500, 2250, 3000, 4500, 7500]	[5, 10, 15, 20, 30, 50]	data sampling
BANKING77	Intent detection	77	[385, 770, 1155, 1540, 2310, 3850]	[5, 10, 15, 20, 30, 50]	data sampling
FRAMES	Intent detection with NER	21	[208, 408, 984]	[10, 20, 50]	data sampling
ATIS	Intent detection with NER	21	[168, 303, 544]	[10, 20, 50]	data sampling

Table 2: Statistics for all datasets and few shot settings.

for home robot scenario (Liu et al., 2019b). It aims at the specific task of capturing the intent for different user requests to home robot and finding the corresponding answer. The raw dataset contains 25,716 data points for 64 class labels through crowdsourcing.

CLINC150, is a dataset designed for task-oriented systems with 23,700 queries that are short and unstructured for 150 intents, in the same style made by real users through crowdsourcing (Larson et al., 2019).

BANKING77, is an intent detection dataset for bank customer services. The raw dataset contains 13,084 data points for 77 class labels (Casanueva et al., 2020).

FRAMES, is a collection of multi-domain dialogues dealing with hotel bookings. The raw data is consists of 1369 human-human dialogues with slot filling information. There are 30,522 utterances in total for 21 intents and 49 slots (Asri et al., 2017).

ATIS, is a popular dataset for intent detection of flight reservations with slot filling(Tur et al., 2010). The raw data's training, development and test sets contain 4,478, 500 and 893 utterances, respectively. There are 120 slot labels and 21 intent types for the training set.

Regarding the first two datasets, we conduct 3-fold cross validation experiments that treat 70 percent as training, 15 percent as validation and testing respectively, and report the averaged testing results. Regarding the last five datasets, we use a sampling method similar to that from (Casanueva et al., 2020) yet in a more sophisticated few-shot settings. We fix a test data for each one, and examine their performances using 5, 10, 15, 20, 30, or 50 samples per class label respectively for training. This is important, as in practice a task-specific chatbot usually starts with merely a handful of labeled data available in the early stage. Besides, in the active learning framework, building an effective auxiliary system with limited resource is also quite important to help developers label new data more efficiently.

3.2 Baselines

Our baselines comprise of *four* styles of systems.

The first one, *non-pretrained model based classification system*, which consists of a typical CNN based TextCNN (Kim, 2014), and a label-embedding based LEAM (Wang et al., 2018). Regarding TextCNN, the input is from RoBERTa tokenization, and the kernels are set as 1, 2, 3, 4, 5. Regarding LEAM, a literal class label is required, which is available only in HWU64, CLINC150, BANKING77. Thus, for other two datasets, we have to use its class number instead. Empirically, a non-pretrained model based system are not performing as well as a pretrained models based in many NLP tasks. Our listing these non-pretrained model based systems here is serving a comprehensive performance comparison on our datasets.

The second one, *pretrained model based classification system*, which consists of BERT (Devlin et al., 2018), RoBERTa (Liu et al., 2019c), and ALBERT (Lan et al., 2019) based. These pretrained models are practically proven to achieve outstanding performances in classification task and other NLP tasks.

The third one, *pretrained model based similarity model*. As RoBERTa is empirically found more effective than other pretrained models in the short text classification task in our experiments, we choose RoBERTa-large based implementation. In inference, we use an elastic search to find a set of potential candidate labels for the similarity model, to guarantee a reasonable running time.

The fouth one, *pretrained model based joint text classification and NER model*. We choose the work from (Chen et al., 2019b) as our baseline system, which utilizes a strong RoBERTa-large model to jointly train two losses. Our SFC framework is also conveniently supportive of extra NER loss that in stage 1 a fourth task with a NER loss is trivially derived based on the output states from a pretrained

Models	FRAMES			ATIS		
	10	20	50	10	20	50
RoBERTa-large (classification)	0.3456	0.4043	0.4262	0.9349	0.9757	0.9800
RoBERTa-large (classification w/ NER)	0.3520	0.4088	0.4353	0.9560	0.9706	**0.9832**
Joint-SFC (w/o NER)	0.3843	0.4130	0.4390	0.9618	0.9761	0.9825
Joint-SFC (w/ NER)	**0.3925**	**0.4420**	**0.4456**	**0.9639**	**0.9784**	0.9826

Table 3: F1 scores for joint sentence classification and NER training. We iterate various sample number per intent and test on original test set.

F1 score	2-stage SFC	joint-SFC	Gap
top-1 accuracy	**81.50**	81.07	-0.47
top-5 accuracy	94.01	**94.30**	+0.29

Table 4: The average classification accuracy in percentage in stage 1 on all five dataset.

model.

All pretrained model baselines are fine-tuned on our datasets. Especially, the similarity model baseline and all similarity layers in SFCs use extra Quora dataset (Iyer et al., 2017) to enhance system performance. This is also one of merits in SFC, as it support adopting out-of-domain data in comparison to a classification based model.

Our SFC implementations include three 2-stage SFCs using task 1 and 2 with ablation in Table 1; one joint-SFC trained on all three tasks in Table 1; one joint-SFC with an additional fourth NER task in Table 3.

All setting details can be found in the appendix due to the space limitation.

3.3 Result Analysis

We report the F1 score [3] as the main evaluation measure for all experiments in Table 1 and Table 3.

Multi-task and joint training

Comparing with training 2-stage SFC with multi-task Table 1, training with only task 1, namely the sentence pair similarity model, degrades by 0.8 percentage point on average, and training with only task 2, namely the top-K based classification task, degrades by 0.45 percentage point on average. These degradation indicates multi-task training in 2-stage SFC is helpful for the system performance. Besides, task 2 plays a relatively more important role in multi-task learning, and this aligns with their optimal weight settings.

[3] In the multi-class and multi-label case, this is the average of the F1 score of each class with weighting depending on the numbers in each class.

Dataset	Model	$K=3$ $P=20$	$K=5$ $P=10$	$K=10$ $P=5$	$K=15$ $P=4$	$K=20$ $P=3$
ITG	2-stage SFC	0.8034	0.8124	0.8008	0.7967	0.7934
	joint-SFC	0.7986	0.8114	0.8010	0.7972	0.7918
Amazon-670k	2-stage SFC	0.7278	0.7364	0.7366	0.7328	0.7204
	joint-SFC	0.7334	0.7445	0.7516	0.7344	0.7373

Table 5: We show the performances of SFC from different settings of hyperparameters, K denoting the candidate class number from stage 1, P denoting the number of sampled sentence pair in stage 2.

Joint-SFC consistently outperforms 2-stage SFC with multi-task training on 5 diverse datasets by 0.95 percentage point on average. Though a joint model structure may bring extra complexity to the system, it does alleviate the error propagation from the first stage and improves the final performance.

In following analysis, we will focus on the comparison between joint-SFC and the other baseline models.

Fusion of classification and similarity models

Regarding the classification models, joint-SFC outperforms RoBERTa-large based, the strongest among all baselines, by 2.04 percentage points on average F1 score over 5 datasets. Especially, ALBERT.xxlarge based is rather unstable in this short sentence task. Thus, we run multiple times with different settings and report the best ones here. Comparatively, RoBERTa based is the most stable and has the best performance almost all experiment settings.

Regarding the similarity model, as analyzed above, we only choose the RoBERTa-large based implementation as our baseline. Joint-SFC achieves more improvement of 7.09 percentage points on average F1 score. This is also understandable, since RoBERTa-large based similarity model does not and also can not train towards the task specific goal.

The above analysis illustrates that suitable fusing these two kinds of models can take their advantages.

Joint-SFC improves stage 1?

We explore the quality of predicted candidate labels in stage 1 from 2-stage SFC and joint-SFC respectively. We should note that 2-stage SFC does not optimize the stage 1 model, namely a RoBERTa based classification model.

From Table 4, the joint training does not improve the top-1 classification accuracy in stage 1, yet it improves the top-5 accuracy. The reason is that, a classification model inherently optimizes the objective loss, 0-1 error of top-1 here; the sentence pair similarity model in joint-SFC poses more positive effect in optimizing the candidate labels.

Supportive of extra NER loss

Many previous work shows joint training with a NER loss improves the intent classification task as well, observed again in our baselines and SFCs in Table 3. We choose the state-of-the-art RoBERTa-large model with NER loss as a strong baseline.

We can see that our joint-SFC without NER loss still outperforms RoBERTa with NER loss baseline by 0.86 percentage points on average. This indicates that the fusion of classification and similarity models can even works better than fusing with additional NER information under few-shot chatbot scenario.

Moreover, the adding NER loss to joint-SFC further achieves improvement of 1.66 percentage points over RoBERTa with NER loss baseline. Especially on FRAMES dataset, which is a much more difficult task in comparison with ATIS based on performance of all systems on it, our joint-SFC with NER loss achieves a bigger improvement of 2.8 percentage points on average.

Different training sample size

We analyze the overall improvement trend of joint-SFC under various few-shot settings, and show results in Fig. 2.

Our main focus is on studying chatbot construction in real world applications, where each class has only a small number of example sentences. The experimental results for the 3 diverse intent detection datasets (ClINC150, BANKING77, HWU64) under few-shot settings, 5 to 50 samples per class, indicate that our proposed joint-SFC can achieve average 2.47 percentage points improvement over one of the most powerful baseline models, which is RoBERTa-large classification model. Moreover, the improvement in F1 score becomes more and more prominent with the decrease of data sample number per class.

Especially, in the most extreme setting with only 5 training samples per class, joint-SFC achieves 4.97 percentage points improvements on average over RoBERTa-large classification model. Joint SFC is a better choice applied into the early stage of building a task-specific chatbot system, since label data is extremely scarce.

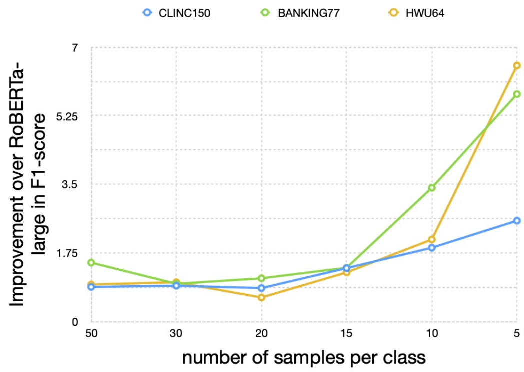

Figure 2: Improvements from joint SFC over RoBERTa based classification with different training size.

Furthermore, joint SFC also achieves average 2.04 percentage points improvement over RoBERTa-large classification model on all the data settings. However, the improvement is still more prominent when the data is scarce, which makes SFC an excellent model for low-resource chatbot scenario.

Relation to the label embedding based LEAM

LEAM shows its superiority over TextCNN in all datasets in our experiments, by modeling labeling information.

LEAM does not perform as well as SFCs, since the first intuitive reason is it does not use a pretrained model as its input encoding module; another critical reason is in task-specific chat applications, it is common to have many similar intents, and it becomes hard and even impossible to name each intent with a short clear name to feed into LEAM model. One candidate solution is setting a most standard sample as the label of the class. When using non-pretrained models base LEAM, this is applicable. Yet when using a pretrained model based, as all labels can be hundreds of thousands, then the training can not be accommodated in a poplar 32 G Tesla V100 GPU.

Actually, joint SFC can be kind of understood as a generalization form of LEAM. When there is no clear class labels, the relationship between a sample and a class label is implicitly encoded as that between a sample and another sample from the same class, and this turns into a sentence pair similarity model.

References

Layla El Asri, Hannes Schulz, Shikhar Sharma, Jeremie Zumer, Justin Harris, Emery Fine, Rahul Mehrotra, and Kaheer Suleman. 2017. Frames: A corpus for adding memory to goal-oriented dialogue systems. *arXiv preprint arXiv:1704.00057*.

Robert Bamler and Stephan Mandt. 2020. Extreme classification via adversarial softmax approximation. *arXiv preprint arXiv:2002.06298*.

Kush Bhatia, Kunal Dahiya, Himanshu Jain, Yashoteja Prabhu, and Manik Varma. 2016. The extreme classification repository: multi-label datasets & code. *URL http://manikvarma. org/downloads/XC/XMLRepository. html*.

Rich Caruana. 1993. Multitask learning: A knowledge-based source of inductive bias icml. *Google Scholar Google Scholar Digital Library Digital Library*.

Iñigo Casanueva, Tadas Temčinas, Daniela Gerz, Matthew Henderson, and Ivan Vulić. 2020. Efficient intent detection with dual sentence encoders. *arXiv preprint arXiv:2003.04807*.

Jindong Chen, Yizhou Hu, Jingping Liu, Yanghua Xiao, and Haiyun Jiang. 2019a. Deep short text classification with knowledge powered attention. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6252–6259.

Qian Chen, Zhu Zhuo, and Wen Wang. 2019b. Bert for joint intent classification and slot filling. *arXiv preprint arXiv:1902.10909*.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167.

Alexis Conneau, Holger Schwenk, Loïc Barrault, and Yann Lecun. 2016. Very deep convolutional networks for text classification. *arXiv preprint arXiv:1606.01781*.

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*.

Manda Sai Divya and Shiv Kumar Goyal. 2013. Elasticsearch: An advanced and quick search technique to handle voluminous data. *Compusoft*, 2(6):171.

Wen Hua, Zhongyuan Wang, Haixun Wang, Kai Zheng, and Xiaofang Zhou. 2015. Short text understanding through lexical-semantic analysis. In *2015 IEEE 31st International Conference on Data Engineering*, pages 495–506. IEEE.

Shankar Iyer, Nikhil Dandekar, and Kornél Csernai. 2017. First quora dataset release: Question pairs. *data. quora. com*.

Sina Jafarpour, Christopher JC Burges, and Alan Ritter. 2010. Filter, rank, and transfer the knowledge: Learning to chat. *Advances in Ranking*, 10:2329–9290.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

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*.

Stefan Larson, Anish Mahendran, Joseph J Peper, Christopher Clarke, Andrew Lee, Parker Hill, Jonathan K Kummerfeld, Kevin Leach, Michael A Laurenzano, Lingjia Tang, et al. 2019. An evaluation dataset for intent classification and out-of-scope prediction. *arXiv preprint arXiv:1909.02027*.

Anton Leuski and David Traum. 2011. Npceditor: Creating virtual human dialogue using information retrieval techniques. *Ai Magazine*, 32(2):42–56.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2016. Recurrent neural network for text classification with multi-task learning. *arXiv preprint arXiv:1605.05101*.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019a. Multi-task deep neural networks for natural language understanding. *arXiv preprint arXiv:1901.11504*.

Xingkun Liu, Arash Eshghi, Pawel Swietojanski, and Verena Rieser. 2019b. Benchmarking natural language understanding services for building conversational agents. *arXiv preprint arXiv:1903.05566*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019c. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Harish Tayyar Madabushi, Elena Kochkina, and Michael Castelle. 2020. Cost-sensitive bert for generalisable sentence classification with imbalanced data. *arXiv preprint arXiv:2003.11563*.

Amr Mousa and Björn Schuller. 2017. Contextual bidirectional long short-term memory recurrent neural network language models: A generative approach to sentiment analysis. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 1023–1032.

Xuan-Hieu Phan, Le-Minh Nguyen, and Susumu Horiguchi. 2008. Learning to classify short and sparse text & web with hidden topics from large-scale data collections. In *Proceedings of the 17th international conference on World Wide Web*, pages 91–100.

Bharath Sriram, Dave Fuhry, Engin Demir, Hakan Ferhatosmanoglu, and Murat Demirbas. 2010. Short text classification in twitter to improve information filtering. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 841–842.

Johan AK Suykens and Joos Vandewalle. 1999. Least squares support vector machine classifiers. *Neural processing letters*, 9(3):293–300.

Zhuowen Tu. 2005. Probabilistic boosting-tree: Learning discriminative models for classification, recognition, and clustering. In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, volume 2, pages 1589–1596. IEEE.

Gokhan Tur, Dilek Hakkani-Tür, and Larry Heck. 2010. What is left to be understood in atis? In *2010 IEEE Spoken Language Technology Workshop*, pages 19–24. IEEE.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Guoyin Wang, Chunyuan Li, Wenlin Wang, Yizhe Zhang, Dinghan Shen, Xinyuan Zhang, Ricardo Henao, and Lawrence Carin. 2018. Joint embedding of words and labels for text classification. *arXiv preprint arXiv:1805.04174*.

Tsung-Hsien Wen, David Vandyke, Nikola Mrksic, Milica Gasic, Lina M Rojas-Barahona, Pei-Hao Su, Stefan Ultes, and Steve Young. 2016. A network-based end-to-end trainable task-oriented dialogue system. *arXiv preprint arXiv:1604.04562*.

Rui Yan, Xian-bin Cao, and Kai Li. 2009. Dynamic assembly classification algorithm for short text. *Acta Electronica Sinica*, 37(5):1019–1024.

Mo Yu, Xiaoxiao Guo, Jinfeng Yi, Shiyu Chang, Saloni Potdar, Yu Cheng, Gerald Tesauro, Haoyu Wang, and Bowen Zhou. 2018. Diverse few-shot text classification with multiple metrics. *arXiv preprint arXiv:1805.07513*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657.