JOINT TRAINING OF CLASSIFICATION AND SIMILARITY MODELS FOR INTENTION DETECTION IN TASK-SPECIFIC CHATBOT

Name of author

Address - Line 1
Address - Line 2
Address - Line 3

ABSTRACT

Task-specific chatbot systems have gained many important applications, such as Siri, Google home, and customer service systems, etc. One essential technique underlying these applications is to detect the real intention behind a user's input. Intention detection, assuming only one intention in a user's input here, can be regarded as a short text classification problem. In practice, the number of intentions can be hundreds of thousands, as opposed to a few number in the traditional text classification problem; creating enough training data from tons of raw user logs with miscellaneous information is not trivial, and generally only 10 to 20 samples per intention are available in the early stage. Thus, how to train an effective short-text classification model with limited labeled data in such a practical and important application is critical. In this work, we propose a novel model, called similarity model fused with classification model (SFC), which combines a classification model and a similarity model, and also borrows the power of multi-task training as well. We conduct extensive experiments on 4 public and 1 private datasets. Experimental results show that our new model outperforms very strong baselines (i.e., BERT, RoBERTa and ALBERT based pretrained models) by over 2 percentages in average F1 score.

1. INTRODUCTION

Task-specific conversational chatbot (Wen et al. 2016) has been applied into many practical products. One typical example is the chatbot that shares the working burden of human customer service agents responsible for customers' questions about certain products or services. Another one is the speech-based control in devices like intelligent speakers, such as Siri, Google home, Baidu Xiaodu, and smart home equipments, to name a few. Regardless of whether these conversations belong to single or multi-round, an essential technique underlying this type of chatbot is to identify the real intention behind a user's question or reply. The detected intention with its associated information is then mapped into a predefined dialog logic graph, from which a suitable response (e.g., conducting expected actions, replying texts or accomplishing some operations, etc) is returned. This is the main difference from a free chatbot.

Intention detection in task-specific chatbot is a challenging task due to some inherent properties. First, a customer's utterance within a conversation is usually quite short. Since it does not contain enough information, short texts (Song et al. 2014) are thought to be more ambiguous compared to long texts such as paragraphs or documents, posing a great challenge (Chen et al. 2019) for classification task (Phan, Nguyen, and Horiguchi 2008; Yan, Cao, and Li 2009; Hua et al. 2015). Second, in the initial stage of building a chatbot , it is often rather hard to collect sufficient labeled data to train a good model. Because developers have to examine tons of real system logs for typical user's utterances, and then label them with expected intentions. Therefore, to build a task-specific chatbot with high performance, it is essential to solve the challenge of short-text classification (Sriram et al. 2010) problem under few-shot setting (Yu et al. 2018).

Generally there are two kinds of approaches to address this problem, one is based on the *(text) classification model* that directly maps an input into a most likely label, and the other one is based on the *(text) similarity model* that searches a label whose associated data is most similar to an input.

The first one, *classification model*, includes classic long text classification and short text classification in this scenario. Besides traditional machine learning models like SVM (Suykens and Vandewalle 1999), boosting tree (Tu 2005), many work (Wen et al. 2016) choose neural network such as convolutional neural networks (CNNs) (Kim 2014; Zhang, Zhao, and LeCun 2015; Conneau et al. 2016) and long short term memory networks (LSTMs) (Mousa and Schuller 2017; Liu, Qiu, and Huang 2016) to accomplish the task of extracting semantic features from limited amount of words. Their common model structure is adding a softmax classifier as the final layer and mapping to labels. Another two interesting lines of work, label-word joint models, and joint NER and classification are discussed in the related work section. Afterwards, pre-trained language models on large corpus like BERT (Devlin et al. 2018) and RoBERTa (Liu et al. 2019c) has been proven more powerful in solving many NLP tasks including short-text classification (Madabushi, Kochkina, and Castelle 2020). Especially for few-shot scenarios (Yu et al. 2018), pre-trained model based on transformers (Vaswani et al. 2017) tends to be more helpful to reduce the negative effects brought by scarcity of training data.

The second one, *similarity model*, is also called sentence-pair model. Its motivation originates from the idea of retrieval based chatbot (Jafarpour, Burges, and Ritter 2010; Leuski and Traum 2011). Given a Q-A (Question-Answer) pairs dataset and a user query Q, the retrieval based conversational system will search from the Q-A dataset the most matched pair (Q', A') through semantic analysis and return A' as the answer to

Q (Mnasri 2019). In this way, sentence-pair model pre-trained on large corpus can be employed to identify the class with highest semantic similarity to customer's query. A common model structure of pre-trained sentence-pair model is based on multiple cross-attention mechanism (Barkan et al. 2020). Due to the fact that many experiment results have shown that RoBERTa (Liu et al. 2019c) is an improved version of BERT (Devlin et al. 2018), and has achieved amazing results on both text classification and sentence-pairs semantic similarity tasks, we choose RoBERTa as one of the baselines in this paper.

Despite the success of these two kinds of approaches, they still have some limitations in task-specific chatbot scenario. As for the *text classification model*, it is quite hard to use data augmentation method to solve the data insufficiency challenge since the it is usually unfeasible to get large amount of domain-specific data when facing a new task. With respect to the *similarity model*, though we can obtain a large amount of data from semantically duplicate sentence pair identification domain for transfer learning (Sun et al. 2019), it is still quite hard to perform well on a new task-specific dataset, since their objective losses are totally different.

The above limitation motivates us to propose a joint system that is capable of taking advantages from both classification model and similarity model. We call it SFC, short for simialirty model fused with classification Model, shown in Fig. 1. To better train SFC, we further bring in multi-task learning (Caruana 1993; Collobert and Weston 2008; Liu et al. 2019b). Our basic idea came from two stages. In the first stage, we use an auxiliary model to select top-K most possible labels for a input. This model can be an elastic search (Divya and Goyal 2013) or a text classification model trained on current task-specific chatbot data. In the second stage, we build a classification model whose main modules are similarity models. Then, this structure derives two goals to train towards, the classification loss for our application, and the similarity loss for the main modules. When the two stages are independently optimized, we call *two-stage SFC*. We further find, the the quality of output from the first stage might limit the final performance of system, since the candidate labels for each user query is always fixed in the whole multi-task training process. This observation motivates us to further improve the two-stage SFC into a joint training, called *joint SFC*. In this way, the sentence pairs associated with the top-K classes also become dynamic, and the text classification model in the first stage will also be further optimized to provide better top-K result by the final training loss.

We summarize our contributions as follow:

1. We propose a novel joint system to makes full utilization of the advantages from both text classification model and similarity model.
2. To better train such a system, we bring in multi-task training to obtain reasonable performance.
3. Experiment results from 4 public and 1 private short-text classification datasets, show that our proposed SFC joint system can achieve significant and consistent improvements over strong baselines, especially in the low resource settings.

2. RELATED WORK

We supplement some work that is very different from classic classification models.

One line of work is based on label-word joint embedding, such as LEAM (Wang et al. 2018), MTLE (Zhang et al. 2017) and EXAM (Du et al. 2019). It poses extral requirements on label information. However in our case, many class labels are close to each other, and not well-defined. Though we can present the class label with one of user's utterances from that class, the label will become not only prolix but also biased.

Another line of interesting work is the joint training of sentence classification and NER, such as (Kruengkrai et al. 2020; Zhang et al. 2020; Hakkani-Tür et al. 2016; Liu and Lane 2016; Goo et al. 2018). For example, If both the city and date information are recognized, then they are good indicators that this query might be booking an ticket.

3. METHODOLOGY

In this section, we describe our proposed SFC. We propose two approaches to implement this idea, two-stage SFC, and its more compact version joint SFC.

3.1. Two-stage SFC

Two-stage SFC is composed with 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 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

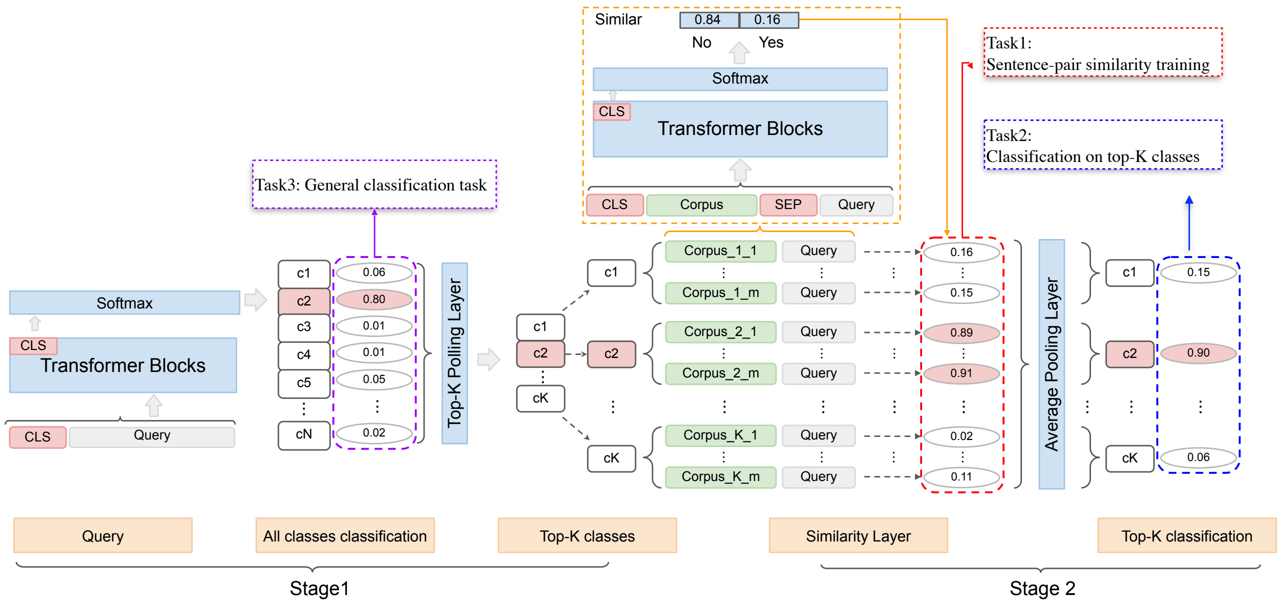

Fig. 1: **Network Structure of SFC:** two-stage SFC and joint-SFC are sharing the same network from stage 1 and stage 2, with the only difference whether two stages being jointly trained.

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 similarity 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 $\mathcal{N'}_i = \{[(x_i, x'_1), 0], ..., [(x_i, x'_{P \cdot K}), 0]\}$

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, Dandekar, and Csernai 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

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

Dataset	Domain	#Class	#Training Samples	#Samples/Class	Experimental Settings
ITG	FAQ chatbot	228	3938 * 0.7	17	3-fold
Amazon-670k	Product review	250	2658 * 0.7	8	3-fold
HWU64	intention detection	64	[320, 640, 960, 1280, 1920, 3200]	[5, 10, 15, 20, 30, 50]	data sampling
CLINC150	intention detection	150	[750, 1500, 2250, 3000, 4500, 7500]	[5, 10, 15, 20, 30, 50]	data sampling
BANKING77	intention detection	77	[385, 770, 1155, 1540, 2310, 3850]	[5, 10, 15, 20, 30, 50]	data sampling

Table 1: Statistics for all datasets and few shot settings.

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.

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

The second task is *classification on top-K classes*. As minimizing similarity loss is not our final goal in the intention 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.

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

With the average pooling result $\xi^{K,pos}$ and $\xi^{K,neg}$, we also generate a top-K class label $l_i^K \in [1, \ldots, K]$ 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.

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

where

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

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

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.

$$\mathcal{L} = \alpha_S \mathcal{L}^S + \alpha_K \mathcal{L}^K \qquad (7)$$

3.2. Joint SFC

In two-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.

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

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 2: F1 scores on five task-specific datasets 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.

4. EXPERIMENTS

4.1. Datasets

Our experiments aim to study the short text classification in the low-resource environment popular in the task-specific conversational chatbot application. Hence we choose and configure set datasets to serve this study. Different from the datasets from other short text classification research, such as semantics classification with very limited semantics labels and relatively longer input, the datasets adopted in our experiments are typically having 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. Table 1 displays their statistics. All settings of four public datasets would be released to github soon.

1. *ITG*, is a proprietory 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.
2. *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 each one is associated with only 5 to 15 samples, or reviews, to mimic the chatbot scenario. From them, we sample 250 class labels as well as their samples to form a subset with 2658 samples.
3. *HWU64*, is a intention detection dataset designed for home robot scenario (Liu et al. 2019a). It aims at the specific task of capturing the intention 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.
4. *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).
5. *BANKING77*, is a intention detection dataset for bank customer services. The raw dataset contains 13,084 data points for 77 class labels (Casanueva et al. 2020).

Regarding the first two datasets, we conduct 3-fold cross validation experiments that 70 percent as training, 15 percent as validation and testing respectively, and report the averaged testing results. Regarding the last three datasets, we conduct experiments using a sampling method similar to that in (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, and 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 data more efficiently.

4.2. Baselines

We choose three kinds of system to construct our baselines.

1. *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.
2. *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 model is based on output 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, if both is applicable in some task. Our listing these non-pretrained model based systems here is to provide a comprehensive performance comparison on our datasets.

3. *Pre-trained model based similarity model*. We choose RoBERTa-large as our fundamental module as our SFC systems are based on RoBERTa too, and RoBERTa is empirically more effective than other pretrained models in the short text classification task in our experiments. 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.

Our SFC systems include two-stage SFC with ablation, trained with two tasks separately and both, and joint SFC with all three tasks.

4.3. Implementation Details

For our SFC systems, we use RoBERTa-large as the pretrained model for both classification task and sentence-pair similarity task. We fine-tune our SFC system on our datasets with max sequence length of 128, learning rate of 1.5e-5 for RoBERTa-large pretrained model and learning rate of 5e-4 for task-specific layer with polynomial decay and Adam optimizer.

In two-stage SFC, in equ. 7 the weights of two tasks are set as $\alpha_S = 0.125$ and $\alpha_K = 0.875$ in our setting. The reason for giving task 2 relatively more weights is that according to the ablation experiments we did in Section 4, task 2 contributes more to the overall performance. In joint SFC, in equ. 8, the weights of three tasks are set as $\alpha_S = 0.25$, $\alpha_K = 0.25$ and $\alpha_C = 0.5$ in our setting. Overall, the task weights are less sensitive in Joint SFC than in two-stage SFC.

For the experiments in Table 2, we empirically set the hyperparameter to be $K = 5$ and $P = 10$ for all SFC variants. In Table 4, we conduct extensive experiments to see the performance of SFC system under various hyperparameter settings.

4.4. Result Analysis

We report the F1 score [2] as the main evaluation measure for all experiments in Table 2.

The performance of each model is measured in F1 score on the fixed test set for each dataset. Overall, two-stage SFC systems in multi-task training deliver moderate improvements over corresponding baselines, and joint SFC deliver more significant and consistent improvements over the best baseline, RoBERTa based classification. In following content, we dive into more details.

Are multi-task and joint training working? First, we compare the four SFC variants to analyze the improvement brought by multi-task and joint-model structure.

[2]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.

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

Table 3: The average classification accuracy in percentage in stage 1 on all five dataset.

Comparing with training two-stage SFC with multi-task, training with only task 1, namely the sentence pair similarity model, degrades by 0.8 percent point on average, and training with only task 2, namely the top-K based classification task, degrades by 0.45 percent point on average. These degradation indicates multi-task training in two-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.

Joint SFC consistently outperforms two-stage SFC with multi-task training on 5 diverse datasets by 0.95 percents on average. This observation supports the idea that the joint model structure can alleviate the limitation brought by the lack of interaction between two stages. In following analysis, we will focus on the comparison between joint SFC and the other baseline models.

Is the fusion of classification and similarity models helpful? We compare joint SFC with classification and similarity based baselines respectively.

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

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

Does joint SFC improve the classification in stage 1? We compare the top-1 prediction and top-5 candidate labels in stage 1 from two-stage SFC and joint SFC in Table 3. We should note that two-stage SFC does not optimize the stage 1 model, which is a RoBERTa based classification model.

The joint training does not improve the classification performance in stage 1, measured by top-1 accuracy. However, it improves the quality of top-5 candidate class labels. It can be explained that, a classification model inherently optimizes the objective loss, 0-1 error of top-1 here; the sentence pair similairity model in the joint SFC poses more positive effect in optimizing the candidate labels.

How does training sample size influence SFC? We analyze the overall improvement trend of joint SFC under various few-shot settings, and show results in Fig. 2.

Our main focus is to study chatbot building in common situation during real-world application where only a few

Dataset	Model	$K = 3$	$K = 5$	$K = 10$	$K = 15$	$K = 20$
		$P = 20$	$P = 10$	$P = 5$	$P = 4$	$P = 3$
ITG	two-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	two-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 4: 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 sententence pair in stage 2.

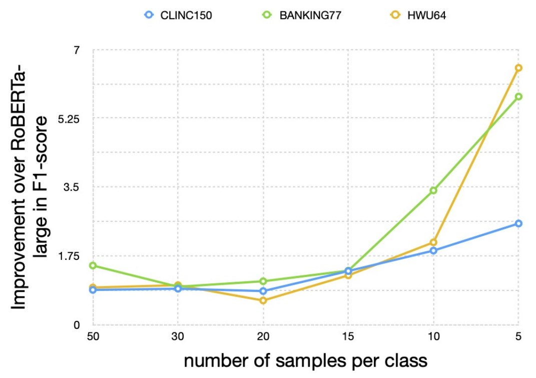

Fig. 2: Improvements from Joint SFC over RoBERTa based classification with different size of training data.

sample sentences are available for each class. The experimental results for the 3 diverse intention detection datasets (ClINC150, BANKING77, HWU64) under few-shot settings, 5 to 50 samples per class, indicate that our proposed joint SFC can achieve 2.47 percents average 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 few-shot setting with only 5 training samples per class our joint SFC achieves 4.97 percent points improvements on average over RoBERTa-large classification model. This observation indicates that SFC is especially well-performed when being applied during the initial stage of building task-specific chatbot system, where the amount of data sample for each class label is extremely scarce.

Furthermore, our proposed joint SFC can also achieve 2.04 percents average improvement over RoBERTa-large classification model on all the data settings including the ones with large scale, 30 to 50 samples per class. This indicates that joint SFC can still steadily outperforms pretrained model based baseline even if the data size grows bigger. However, the improvement is still more prominent when the data is scarce, which makes SFC an excellent model for low-resource chatbot scenario.

How does joint SFC relate to the label embedding based LEAM? LEAM performs consistently better than TextCNN by a large margin in all datasets. TextCNN is practically efficiently in task-specific chat applications, and LEAM shows its power in modeling labeling information.

LEAM does not perform as well as SFCs, since the first intuitive reason is it does not use pretrained model as encoding module; and another critical reason is in task-specific chat applications, it is common to have many intentions that are close to some others with a minor difference, and it is hard and even impossible to name each intention with a short clear name. One candidate solution is setting a most standard sample as the label of that class. When using non-pretrained models base LEAM, this is applicable. Yet when using pretrained model based, as the all labels can be hundreds of thousands, then this 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. In the scenario when there is no clear class label, 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.

Settings of hyperparameters In Table 4, we explore the influence from hyperparameters, which are the number of candidate class labels K in stage 1, and the number of sampled sentence pairs for each class P in stage 2, on ITG and Amazon-670K datasets. As we control the total number of sampled sentence pairs to about 50 to 60, we roughly obatin five combinations of hyperparameters.

We find that the best setting of hyperparameters is different from datasets of different data size. The class label number of ITG and Amazon-670k are similar in our experimental setting, which is around 250. However the total amount of data points of ITG is approximately 2 times larger than Amazon-670k. Therefore, the setting of $K = 5$ and $P = 10$ is good enough for ITG since most of the classes contains over 10 data points. However, for Amazon-670k, the amount of data sample for each class is scarce, thus, we may need more candidate classes to sample enough effective sentence pairs for sentence-pair model layer of SFC. In this way, the setting of $K = 10$ and $P = 5$ becomes a good choice.

However, in spite of the effect on evaluation performance brought by different settings of hyperparameters, joint SFC still steadily outperforms two-stage SFC by 0.41 percent points in overall average F1 score on ITG and Amazon-670k datasets.

References

Bamler, R.; and Mandt, S. 2020. Extreme Classification via Adversarial Softmax Approximation. *arXiv preprint arXiv:2002.06298* .

Barkan, O.; Razin, N.; Malkiel, I.; Katz, O.; Caciularu, A.; and Koenigstein, N. 2020. Scalable Attentive Sentence Pair Modeling via Distilled Sentence Embedding. In *AAAI*, 3235–3242.

Bhatia, K.; Dahiya, K.; Jain, H.; Prabhu, Y.; and Varma, M. 2016. The extreme classification repository: multi-label datasets & code. *URL http://manikvarma.org/downloads/XC/XMLRepository. html* .

Caruana, R. 1993. Multitask Learning: A Knowledge-Based Source of Inductive Bias ICML. *Google Scholar Google Scholar Digital Library Digital Library* .

Casanueva, I.; Temčinas, T.; Gerz, D.; Henderson, M.; and Vulić, I. 2020. Efficient Intent Detection with Dual Sentence Encoders. *arXiv preprint arXiv:2003.04807* .

Chen, J.; Hu, Y.; Liu, J.; Xiao, Y.; and Jiang, H. 2019. Deep short text classification with knowledge powered attention. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 6252–6259.

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

Conneau, A.; Schwenk, H.; Barrault, L.; and Lecun, Y. 2016. Very deep convolutional networks for text classification. *arXiv preprint arXiv:1606.01781* .

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* .

Divya, M. S.; and Goyal, S. K. 2013. ElasticSearch: An advanced and quick search technique to handle voluminous data. *Compusoft* 2(6): 171.

Du, C.; Chen, Z.; Feng, F.; Zhu, L.; Gan, T.; and Nie, L. 2019. Explicit interaction model towards text classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 6359–6366.

Goo, C.-W.; Gao, G.; Hsu, Y.-K.; Huo, C.-L.; Chen, T.-C.; Hsu, K.-W.; and Chen, Y.-N. 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)*, 753–757.

Hakkani-Tür, D.; Tür, G.; Celikyilmaz, A.; Chen, Y.-N.; Gao, J.; Deng, L.; and Wang, Y.-Y. 2016. Multi-domain joint semantic frame parsing using bi-directional rnn-lstm. In *Interspeech*, 715–719.

Hua, W.; Wang, Z.; Wang, H.; Zheng, K.; and Zhou, X. 2015. Short text understanding through lexical-semantic analysis. In *2015 IEEE 31st International Conference on Data Engineering*, 495–506. IEEE.

Iyer, S.; Dandekar, N.; and Csernai, K. 2017. First quora dataset release: Question pairs. *data. quora. com* .

Jafarpour, S.; Burges, C. J.; and Ritter, A. 2010. Filter, rank, and transfer the knowledge: Learning to chat. *Advances in Ranking* 10: 2329–9290.

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

Kruengkrai, C.; Nguyen, T. H.; Aljunied, S. M.; and Bing, L. 2020. Improving Low-Resource Named Entity Recognition using Joint Sentence and Token Labeling. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 5898–5905.

Lan, Z.; Chen, M.; Goodman, S.; Gimpel, K.; Sharma, P.; and Soricut, R. 2019. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942* .

Larson, S.; Mahendran, A.; Peper, J. J.; Clarke, C.; Lee, A.; Hill, P.; Kummerfeld, J. K.; Leach, K.; Laurenzano, M. A.; Tang, L.; et al. 2019. An evaluation dataset for intent classification and out-of-scope prediction. *arXiv preprint arXiv:1909.02027* .

Leuski, A.; and Traum, D. 2011. NPCEditor: Creating virtual human dialogue using information retrieval techniques. *Ai Magazine* 32(2): 42–56.

Liu, B.; and Lane, I. 2016. Attention-based recurrent neural network models for joint intent detection and slot filling. *arXiv preprint arXiv:1609.01454* .

Liu, P.; Qiu, X.; and Huang, X. 2016. Recurrent neural network for text classification with multi-task learning. *arXiv preprint arXiv:1605.05101* .

Liu, X.; Eshghi, A.; Swietojanski, P.; and Rieser, V. 2019a. Benchmarking natural language understanding services for building conversational agents. *arXiv preprint arXiv:1903.05566* .

Liu, X.; He, P.; Chen, W.; and Gao, J. 2019b. Multi-task deep neural networks for natural language understanding. *arXiv preprint arXiv:1901.11504* .

Liu, Y.; Ott, M.; Goyal, N.; Du, J.; Joshi, M.; Chen, D.; Levy, O.; Lewis, M.; Zettlemoyer, L.; and Stoyanov, V. 2019c. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692* .

Madabushi, H. T.; Kochkina, E.; and Castelle, M. 2020. Cost-Sensitive BERT for Generalisable Sentence Classification with Imbalanced Data. *arXiv preprint arXiv:2003.11563* .

Mnasri, M. 2019. Recent advances in conversational NLP: Towards the standardization of Chatbot building. *arXiv preprint arXiv:1903.09025* .

Mousa, A.; and Schuller, B. 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*, 1023–1032.

Phan, X.-H.; Nguyen, L.-M.; and Horiguchi, S. 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*, 91–100.

Song, G.; Ye, Y.; Du, X.; Huang, X.; and Bie, S. 2014. Short text classification: A survey. *Journal of multimedia* 9(5): 635.

Sriram, B.; Fuhry, D.; Demir, E.; Ferhatosmanoglu, H.; and Demirbas, M. 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*, 841–842.

Sun, C.; Qiu, X.; Xu, Y.; and Huang, X. 2019. How to fine-tune bert for text classification? In *China National Conference on Chinese Computational Linguistics*, 194–206. Springer.

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

Tu, Z. 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, 1589–1596. IEEE.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention is all you need. In *Advances in neural information processing systems*, 5998–6008.

Wang, G.; Li, C.; Wang, W.; Zhang, Y.; Shen, D.; Zhang, X.; Henao, R.; and Carin, L. 2018. Joint embedding of words and labels for text classification. *arXiv preprint arXiv:1805.04174* .

Wen, T.-H.; Vandyke, D.; Mrksic, N.; Gasic, M.; Rojas-Barahona, L. M.; Su, P.-H.; Ultes, S.; and Young, S. 2016. A network-based end-to-end trainable task-oriented dialogue system. *arXiv preprint arXiv:1604.04562* .

Yan, R.; Cao, X.-b.; and Li, K. 2009. Dynamic assembly classification algorithm for short text. *Acta Electronica Sinica* 37(5): 1019–1024.

Yu, M.; Guo, X.; Yi, J.; Chang, S.; Potdar, S.; Cheng, Y.; Tesauro, G.; Wang, H.; and Zhou, B. 2018. Diverse few-shot text classification with multiple metrics. *arXiv preprint arXiv:1805.07513* .

Zhang, H.; Xiao, L.; Chen, W.; Wang, Y.; and Jin, Y. 2017. Multi-task label embedding for text classification. *arXiv preprint arXiv:1710.07210* .

Zhang, L.; Ma, D.; Zhang, X.; Yan, X.; and Wang, H. 2020. Graph LSTM with Context-Gated Mechanism for Spoken Language Understanding. In *AAAI*, 9539–9546.

Zhang, X.; Zhao, J.; and LeCun, Y. 2015. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, 649–657.