CS-BERT: a pretrained model for customer service dialogues

Peiyao Wang, Joyce Fang and **Julia Reinspach**
Amazon, Seattle, USA
peiyaow,joycfang,reinspac@amazon.com

Abstract

Large-scale pretrained transformer models have demonstrated state-of-the-art (SOTA) performance in a variety of NLP tasks. Nowadays, numerous pretrained models are available in different model flavors and different languages, and can be easily adapted to one's downstream task. However, only a limited number of models are available for dialogue tasks, and in particular, goal-oriented dialogue tasks. In addition, the available pretrained models are trained on general domain language, creating a mismatch between the pretraining language and the downstream domain launguage. In this contribution, we present CS-BERT, a BERT model pretrained on millions of dialogues in the customer service domain. We evaluate CS-BERT on several downstream customer service dialogue tasks, and demonstrate that our in-domain pretraining is advantageous compared to other pretrained models in both zero-shot experiments as well as in finetuning experiments, especially in a low-resource data setting.

1 Introduction

The introduction of large-scale pretrained transformers such has BERT and GPT-2 (1; 2) has led to SOTA results in a variety of NLP tasks, such as question answering (QA) (3; 4; 5), named-entity recognition (NER) (6), or text summarization (7). The pretraining of these models is performed on large unlabeled text datasets with self-supervised objective functions: auto-regressive language modeling (2; 5; 8), masked language modeling (1; 9), replaced token detection (10), or next sentence prediction (1; 11), to name a few. Afterwards, the pretrained model can be adapted to a specific task by finetuning on a much smaller, labeled, and task-specific dataset, thus eliminating the need for huge labeled datasets while still maintaining the power of large-scale transformer models.

Most of the published pretrained transformer encoders have been pretrained on large text corpora covering general topics and domains, such as the Wikipedia corpus or the books corpus. Hence, they do not represent the structure that is inherent to a dialogue, i.e., two or more speakers conversing on a topic, exchanging information, or working towards solving a problem. Recently, a few pretrained transformer encoders for dialogue tasks have been released, i.e., conversational-BERT (12) for chit-chat dialogues, and TOD-BERT (13) for goal-oriented dialogues. However, dialogue models for customer service are very domain-specific (e.g. check order status, request a refund), and as a result, even the few existing dialogue-specific pretrained models exhibit a language domain mismatch between the pretraining and the downstream task. This language mismatch leads to inefficiencies in the knowledge transfer from pretraining to finetuning, resulting in worse downstream task performance, and requiring larger amounts of labeled finetuning data.

In this work, we focus on developing retrieval-based task-oriented dialogue systems, which have been successfully deployed in real-world customer service applications (14; 15). In these systems, the dialogue model ranks a pre-defined pool of agent responses, given a customer's input and the conversation history. One of the biggest challenges of developing such models is the limited amount of training data (15): high-quality annotated dialogue data is needed, which is expensive to obtain. As one of the most effective solutions, in-domain pretraining has been shown to lead to performance gains under the low-resource setting (16; 17). More specifically, for customer service dialogue modeling, (15) showed that pretraining on a large, unannotated and noisy dataset with subsequent finetuning on a small, high-quality labeled dataset yields improved performance as compared to only training with the small labeled dataset from scratch. However, the authors only evaluated their pretraining-finetuning approach using a rather small pretraining dataset of ca. 380K dialogues, and using a small-scale transformer architecture. In addition, they did not consider advanced pretraining objectives such as masked language modeling (MLM) used in BERT, but their pretraining was done using the same training objective as the downstream task.

In this paper, we present CS-BERT, a BERT model pretrained on customer service dialogues, which efficiently reduces the need for expensive high-quality labeled dialogue task training data. In particular, we present the following novelties:

1. We develop CS-BERT, the first customer service domain-specific dialogue BERT model.
2. We present two novel BERT pretraining strategies specific for dialogue tasks, i.e., "masked speaker prediction" and "segment embeddings by speaker roles", which outperform models pretrained on the original BERT objectives (i.e., MLM and next sentence prediction).
3. We evaluate CS-BERT on several downstream tasks, and show that CS-BERT yields superior performance compared to BERT, conversational-BERT and TOD-BERT, in both a zero-shot setting as well as after finetuning, especially in a low-resource data setting.

The paper is structured as follows: Section 2 provides a literature overview of BERT and other available pretrained encoders. Section 3 describes the pretraining strategies that we used to develop CS-BERT. In Section 4 we describe the datasets and model training details for pretraining, as well as for finetuning on three specific downstream tasks. Section 5 follows with experimental results and discussions. We conclude the paper with a summary in Section 6.

2 Related literature

In this section, we review related literature. We first introduce BERT, then we discuss BERT variations that were developed to improve the original model. Lastly, we present the available pretrained models for dialogue tasks.

2.1 Pretrained transformer encoders: BERT

The original BERT model (1) has two self-supervised pretraining objectives: MLM and next sentence prediction (NSP). For the MLM objective, a small percentage ($\sim$15%) of the input subword tokens (18) are randomly masked, and the training objective is to predict the original token. For the NSP objective, 50% of the last sentences in an input document are replaced with a random sentence, and the training objective is to predict whether the last sentence belongs to the document or not. The last sentence is highlighted by a special separator in the input text, and by segment embeddings (segment A for the first part of the document, segment B for the last sentence).

2.2 Pretrained encoder flavors

After the initial publication of BERT, several pretrained transformer encoder flavors have been released. For example, RoBERTa (9) demonstrates improved performance by using a 10x larger pretraining dataset, and by optimizing training hyperparameters more carefully. ELECTRA (10) improves performance and pretraining time by a more efficient pretraining objective: instead of MLM, they introduce the "replaced token detection" task, thus making use of all the tokens in the input sequence. ALBERT (19) reduces model memory footprint by reducing the vocabulary embedding size and by parameter sharing. In addition, instead of using BERT's NSP task, ALBERT introduces a new and more difficult pretraining objective, i.e., "inter-sentence coherence loss", which helps model performance. Several works addressed BERT's issue of both long pretraining and inference time by reducing its quadratic computational complexity in the self-attention module, e.g., SqueezeBERT, Reformer, Performer and Longformer (11; 20; 21; 22). Other publications focus on reducing BERT's large memory footprint by model distillation, e.g., DistilBERT or DistilRoBERTa (23), which can benefit applications with limited computational resources.

2.3 Pretrained encoders for dialogue models

Most pretrained transformer encoders are pretrained on text corpora such as Wikipedia, News or Books. Thus, none of these represents the structure that is inherent to a dialogue, i.e., two or more speakers conversing on a topic, exchanging information, or working towards solving a problem. In the following, we briefly review the publicly available pretrained models that have been adapted for dialogue tasks.

Conversational-BERT (12) is a BERT model that was trained on several chit-chat type dialogue corpora, e.g., Twitter, Reddit, and Facebook News. The authors start from a cased BERT-base checkpoint, and then use BERT's standard objectives (MLM and NSP) for further pretraining on the dialogue data. Thus, while conversational-BERT is trained on dialogue-type datasets, it is neither trained on task-oriented dialogues, nor specific to any domain.

TOD-BERT (13) is a BERT model explicitly pretrained on goal-oriented datasets only. As in conversational-BERT, the authors start from an existing BERT-base checkpoint, and then pretrain TOD-BERT in a bi-encoder architecture, using a response contrastive loss in addition to the MLM objective. In addition, the authors also introduce two additional special tokens to indicate whether a user or system generated the text, i.e., "[USR]" and "[SYS]". While TOD-BERT was trained on task-oriented dialogues, it is also not specific to any domain.

In addition to the above mentioned dialogue transformer encoders, there also exist few pretrained generative models for open-domain conversations. Blenderbot (24) is an encoder-decoder architecture pretrained on Reddit data, while DialoGPT (25) is a decoder-only transformer model pretrained on Reddit data. However, in order to provide a fair comparison, we do not consider DialoGPT (25) or Blenderbot (24) for this work, since these are not transformer encoders.

3 CS-BERT pretraining strategies

In this section, we review the different pretraining strategies that we use to train CS-BERT. In particular, we use three pretraining strategies: (1) we use the same pretraining objectives as BERT, i.e., MLM and NSP/NTP (denoted as CS-BERT), (2) we add an additional objective, i.e., masked speaker prediction (MSP), to BERT's original pretraining objectives (denoted as CS-BERT-MSP), and (3) we use the same pretraining objectives as BERT, but use a different segment embedding strategy (denoted as CS-BERT-SSR).

3.1 Masked language modeling (MLM)

As in the original BERT paper, we pick 15% of tokens at random. We then replace 80% of these tokens with "[MASK]", 10% with another random token, and we keep 10% of the tokens unchanged. Then we compute the cross-entropy loss to predict the original token.

3.2 Next turn prediction (NTP)

Similar to the NSP task used in BERT, where the goal is to predict the next consecutive sentence following the context, the NTP task aims to predict the next consecutive agent turn following the dialogue context. We keep the actual agent turn following a dialogue history as a positive context-response pair, and we randomly sample one agent response from the pool of all available agent responses as a negative context-response pair.

3.3 Masked speaker prediction (MSP)

To enhance the model's ability to capture different roles from the dialogue, we incorporate the objective of MSP, which can be considered a special case of MLM applied to the three special dialogue tokens that indicate the speaker roles, i.e., "INITIALQUESTION", "AGENTSTART" and "CUSTOMERSTART" (see Table 1 for an illustration of the input format). During the training data generation, we randomly select 50% of these three tokens. We then replace 80% of them with "[MASK]", we replace 10% with another special dialogue token, and we keep 10% unchanged. The model then predicts whether the selected token is from the customer (i.e., if the original token is either "INITIALQUESTION" or "CUSTOMERSTART") or from the agent (i.e., if the original token is "AGENTSTART"). Binary cross-entropy loss is used for each masked token.

3.4 Segment embedding by speaker roles (SSR)

BERT uses binary segment embeddings to help the model distinguish between parts of the input sequence. Using BERT's default approach, the segment embeddings for the dialogue context are 0s, while the segment embeddings for the agent response (i.e., last turn in the dialogue) are 1s. For SSR, we develop a different segment embedding strategy motivated by the fact that in a dialogue there are alternating roles of customer and agent. In particular, we set the segment embeddings of all agent turns to 1s, and the segment embeddings of all customer turns to 0s. Compared to the standard segment embedding strategy, SSR focuses less on the NTP task, and more on the global contextual understanding of the dialogue.

4 Experiment setup

For all experiments in this paper, we use dialogues collected from the customer service of a large e-commerce company. We only use "chat" dialogues, i.e., written interactions between customer service agents and customers (as opposed to email conversations or transcribed spoken interactions, e.g., via phone support). We will distinguish between two dialogue datasets types:“non-annotated” dialogues and “annotated” dialogues. “Non-annotated” dialogues are free text conversations between a customer and an agent, and are inherently noisy. “Annotated” dialogues are conversations between customers and specialized agents, who provide annotations and only use a limited pool of approved agent responses (15). These annotated dialogues are of higher quality and are less noisy, and thus make it easier for a model to learn from. However, they are also more expensive to obtain.

4.1 CS-BERT pretraining

4.1.1 Pretraining dataset

As pretraining dataset, we use roughly 40,000,000 non-annotated customer-agent dialogues in English language. These dialogues come from all available customer service intents. The dialogues were filtered to have a minimum length of 1,000 words, and a maximum length of 80,000 words.

After splitting the dialogues in train and dev sets, we create turn-level training data: for each agent turn and its corresponding dialogue history (context), we use the actual agent turn as a positive example for the NTP task, and use randomly sampled agent responses as negative examples. For the dev dataset, we do not use all turns in the dialogues, but randomly sample a smaller subset of turns. This makes hyperparameter tuning and model evaluation more computationally efficient. Table 1 provides an example of a positive context-response pair that was created for pretraining CS-BERT. The pretraining dataset statistics are described in Table A.1 in the Appendix.

4.1.2 CS-BERT model training

To leverage the useful knowledge from the pretrained BERT, we do not train CS-BERT from scratch, but start from the pretrained BERT-base checkpoint. As in (1), we train with a total batch size of 256 sequences for 1,000,000 steps. However, we use a maximum sequence length of 256 throughout the whole process. We use the same Adam optimizer as in (1) with learning rate of $1e{-}4$, $\beta_1 = 0.9, \beta_2 = 0.999$, L_2 weight decay of 0.01, learning rate warmup over the first 10,000 steps, and linear decay of the learning rate. All our CS-BERT models were trained on 8 GPUs, and the total training time is around 14 days.

4.2 Response ranking

The first downstream task we evaluate is dialogue response ranking, the objective of which is to predict the most likely agent response from a predefined pool of agent responses, given the dialogue history (context) and (optionally) other dialogue metadata or features. The response ranking model is finetuned as a binary classification model, i.e., given a dialogue history and features, the model needs to predict whether the response is a valid next turn or not. During inference, the scores of all responses in the response pool are calculated, and then ranked to determine the most likely agent response.

4.2.1 Datasets for response ranking

While the non-annotated pretraining dataset described in Section 4.1.1 made use of all available customer service intents, in practice, we develop separate downstream models for specific intents. For this work, we use datasets from three different intents, i.e., “item-in-transit” (in-transit), “item-delivered-not-received” (DNR), and “start-return” (SR). For all intents, we use two types of dialogue data, as described below.

Annotated finetuning response ranking data We obtain high-quality labeled datasets for finetuning from conversations and annotations created by a pool of specialized, highly trained customer service agents (15). For all three intents, we collect these annotated dialogues for a ca. 3 month period (that is non-overlapping with the CS-BERT pre-training data time period), and then split the dataset into train and dev sets. We then further collect several weeks of annotated data to create a non-overlapping test dataset, which will be used for model evaluation. As for the pretaining data, we only use dialogues in English language.

After collecting the dialogues for the train/dev/test distributions, we create turn-level training data for the response ranking task. For each agent turn and its corresponding dialogue history (context), we use the actual agent turn a positive example. As a negative example, we randomly sample from the pool of available agent responses (i.e., the pool of responses that the model has available during inference). For the sake of computational efficiency, during hyperparameter tuning, we do not use all turns in the dialogue for the dev sets, but sample a subset of turns. For the test set that we use for

context	agent response
INITIALQUESTION Tracking says that my package may be lost. It was supposed to arrive on UCI_TOKEN. AGENTSTART Can you confirm the shipping address of the order? CUSTOMERSTART UCI_TOKEN	As I see, both the UCI_TOKEN and UCI_TOKEN are sold by our partner sellers and are shipped by Fedex so as much as I would love to send replacements for you, I don't have the option for that since we don't have the direct inventory of the items.

Table 1: An example of a positive context-response pair we use for pretraining. In particular, in the context, "INITIALQUESTION" is always added at the beginning of the dialogue to mark the customer's first utterance. "AGENTSTART" is always added before an agent's utterance; "CUSTOMERSTART" is always added before a customer's utterance (unless it's the customer's first utterance). Sensitive customer informations in the dialogue text are replaced by "UCI_TOKEN" to protect customer privacy.

model evaluation, we keep all turns in a dialogue.

For the response ranking finetuning data, we also collect metadata. Metadata are specific to each intent, and can be helpful for the model to predict the correct response. Table A.2 illustrates a positive context-response pair of the finetuning response ranking dataset, including metadata. The dataset statistics for the three intents can be found in Table A.3 in the Appendix.

Non-annotated finetuning response ranking data In addition to the annotated finetuning datasets, we also use general customer-agent dialogues for finetuning (i.e., dialogues from general customer service agents). We do that because we found that adding this data to the annotated finetuning data improves the response ranking model's generalization and performance. Please note that this is the same dialogue data type as used for CS-BERT pretraining. However, we filter by intent to match the intent of the annotated data[1]. The processing for this dataset is the same as the processing for the CS-BERT pretraining dataset (see Table A.4 in the Appendix for the dataset statistics). Since the non-annotated datasets are used for regularization purposes only, we do not report metrics on these.

4.2.2 Response ranking finetuning details

We finetune all response ranking downstream tasks on a 1:1 mixture of annotated:non-annotated data. We adopt the same cross-encoder architecture as in pretraining and use the NSP head for response ranking. We use the Adam optimizer with learning rate of 2e−5, a linear learning rate warmup for the first 1,000 steps, and then linear decay. All the other parameters are the same as in pretraining. We finetune with a batch size of 176 for 5 epochs, and we pick the model that yields the highest Recall@1 on the dev dataset to perform the evaluation on the test set.

[1]Noisy intent labels for this dataset were obtained by an intent classifier on the initial customer utterance.

4.3 Dialogue failure detection

Another downstream task we evaluate is the dialogue failure detection task. The goal of failure detection is to detect when the goal-oriented chatbot is not able to solve a customer's issue, given the limited number of agent responses in the predefined response pool. When a failure happens, the chatbot exits the conversation and transfers the customer to a general customer service agent. The failure detection model and the response ranking model work in combination to create a real-world dialogue system for customer service.

We also approach failure detection as a binary classification task, i.e., given a dialogue history and metadata, we aim to predict whether the dialogue is a failure or not. During real-world inference, we call the failure detection model at each turn in the conversation, and transfer the customer to a human agent in the case of failure.

4.3.1 Dialogue failure detection data

We obtain the failure detection labels from the annotated response ranking finetuning data. Each turn in the conversation is labeled as non-failure if the conversation continues, and as failure if the conversation should be transferred to a human agent. To avoid target leakage, we only use the dialogue context to model failure detection, and exclude the selected response. See Table A.5 in the Appendix for an illustration of the model input. The full dataset statistics can be found in Table A.6 in the Appendix.

4.3.2 Failure detection finetuning details

We adopt the same cross-encoder architecture and its optimization setup as for the response ranking model described in Section 4.2.2. The failure detection model is finetuned for up to 5 epochs, with a batch size of 80. We evaluate all the models on the test sets at the epoch that yields the smallest loss on the dev set.

4.4 NER

We investigate the CS-BERT models for a NER downstream task. In particular, we build a NER model to identify personally identifiable information (PII) in customer and agent turns, which helps to protect the privacy information of our customers. We approach this as a token classification task. Given a turn from a customer or an agent, we finetune the NER model to predict the PII tokens within each turn. This is a multi-label classification task where we define a total of 14 PII entities.

4.4.1 NER data

The NER datasets are randomly sampled utterances from customer-agent chat dialogues that were labeled with PII entities by human annotators. We use $50,000$ agent or customer turns, where 12.2% of these contains at least one PII entity. Although we define 14 PII entities in the annotation task, the data is highly skewed toward common entities, such as name, phone number, email, physical address, and some of the entities are not even observed in the 50,000 turns. We split 70% of the data for training and validation, and the remaining 30% are used as test set for evaluation. Table A.7 in the Appendix shows the entity distribution of the NER dataset. We preprocess the data to inside, outside, beginning (IOB) format (26) before finetuning. Table 2 shows an example model input and corresponding output IOB labels.

4.4.2 NER finetuning details

The NER model uses each pretrained model with an additional linear layer and softmax activation for final token classification. We use the Adam optimizer for finetuning where the learning rate is 4e-5, $\beta = (0.9, 0.999)$, $\epsilon = 1e{-}8$, and the training batch size = 64. For each pretrained model we train the NER task for 5 epochs, and the model of the epoch with the highest F1 score on the validation set is selected for evaluation on the test set.

5 Results and discussion

5.1 Response ranking results

Throughout this paper the metrics we use for response ranking evaluation are mean reciprocal rank (MRR), recall@1 and recall@5. We evaluate model performance on the test set of the high-quality annotated response ranking dataset here. For completeness, we also report test results on the non-annotated response ranking datasets in the Appendix.

5.1.1 Response ranking zero-shot results

We first evaluate the pretrained models' performance on the high-quality annotated test sets for the three intents. We consider this a zero-shot inference task across all pretrained models, since none of them were pretrained using finetuning type training data.

To perform the zero-shot evaluation on the CS-BERT models, we use the NSP prediction head inherited from pretraining. For BERT, we use the NSP head inherited from the pretrained BERT model. For TOD-BERT we use the bi-encoder architecture, since this is how it was pretrained. We modify their original response contrastive loss head into a softmax cross entropy head to compute the scores for response ranking evaluation. For conversational-BERT, we use an NSP head as in the CS-BERT/BERT models, but with weights randomly initialized, because the authors did not release the model weights of the NSP head.

The results from the six pretrained models are summarized in Table 3. The CS-BERT pretrained models are consistently better than the non-CS-BERT models for all three intents. In particular, for in-transit and DNR, the MRR and recall@1 from CS-BERT-MSP evaluated on the test data are much better than those from the other two CS-BERT models. This demonstrates the usefulness of adding the MSP task into pretraining for dialogue downstream tasks.

5.1.2 Response ranking finetuning results

High-quality annotated finetuning data is time-consuming and expensive to obtain. Hence, we experiment with the low-resource data setup to evaluate the data efficiency of different pretrained models on the downstream response ranking task. In particular, we finetune the response ranking models with 5%, 10%, 20%, 50% and 100% of the available annotated finetuning data. The full results can be found in Table A.8 in the Appendix.

For illustration, in Figure 1, we plot MRR and recall@1 evaluated on the annotated finetuning test

context	**IOB label**
CUSTOMERSTART My name is Jane Doe.	O O O O B-NAME I-NAME O
I have an item delivered to Seattle WA.	O O O O O O B-ADDRESS I-ADDRESS O

Table 2: An example of the model input and label output for the finetuning NER task. The input to the model is one turn and the output are the entities in IOB format.

pretrained encoder	**MRR**	**Recall@1**	**Recall@5**	**MRR**	**Recall@1**	**Recall@5**	**MRR**	**Recall@1**	**Recall@5**
	in-transit			**DNR**			**SR**		
BERT	10.00%	0.60%	17.20%	5.50%	0.00%	1.20%	6.43%	1.31%	5.63%
Tod-BERT	13.30%	2.30%	18.40%	11.10%	1.60%	17.70%	9.19%	2.55%	10.56%
conversational-BERT	11.70%	4.90%	9.90%	8.00%	1.30%	10.00%	10.14%	3.68%	11.27%
CS-BERT	26.00%	6.00%	55.80%	33.40%	14.20%	**54.60%**	50.15%	30.97%	**75.62%**
CS-BERT-SSR	29.70%	6.50%	**58.80%**	31.30%	11.50%	51.10%	**50.41%**	**31.46%**	75.58%
CS-BERT-MSP	**33.10%**	**12.10%**	58.10%	**35.00%**	**19.70%**	52.90%	49.63%	30.95%	74.04%

Table 3: Zero-shot inference results from pretrained models on the annotated response ranking finetuning test sets.

	Entity level			**Word level**		
	Precision	**Recall**	**F1 score**	**Precision**	**Recall**	**F1 score**
BERT	87.73%	93.02%	90.30%	89.63%	92.06%	90.83%
Tod-BERT	90.52%	89.36%	89.94%	91.94%	87.89%	89.87%
conversational-BERT	91.13%	89.79%	90.46%	93.13%	87.97%	90.48%
CS-BERT	87.84%	93.64%	90.65%	88.39%	91.90%	90.11%
CS-BERT-MSP	89.10%	93.11%	**91.06%**	89.88%	91.82%	**90.84%**

Table 4: NER result

set as a function of data percentages used to finetune the response ranking models. For clarity, we only pick one CS-BERT model for these figures, since the curves for all three CS-BERT models are very similar. The plots clearly show that the CS-BERT model consistently outperforms the other three models in the low-resource data setting across three intents. This indicates a more efficient knowledge transfer from CS-BERT pretraining. As the amount of finetuning data gets larger, the metrics from different models start to converge. For SR, the conversational-BERT also achieves certain improvements over BERT and TOD-BERT, which illustrates that pretraining on general dialogues itself can help dialogue related downstream tasks. Even though TOD-BERT was pretrained using goal-oriented dialogues, it does not demonstrate a powerful knowledge transfer, probably due to the pretraining-finetuning model architecture mismatch.

5.2 Failure detection finetuning results

We run the same data efficiency experiments described in Section 5.1.2 for the failure detection task. As metrics we use area under precision-recall curve (AUC) scores. The full results can be found in Table A.9 in the Appendix.

For illustration, in Figure 2, we plot AUC evaluated on the failure detection test set as a function of data percentages used to finetune the failure detection models. For conciseness, we only plot one curve from CS-BERT for all three CS-BERT models. The CS-BERT model clearly outperforms the other three models, especially in the low-resource data setting. In particular, when using only 50% of annotated data or less, the CS-BERT demonstrates higher AUC than the non-CS-BERT models trained on 100% of the available annotated data, demonstrating the usefulness of in-domain pretraining. Moreover, TOD-BERT and conversational-BERT outperform BERT in the low-resource setting, which further agrees with our assumption that pretraining on general dialogues yields better performance on dialogue related downstream tasks.

5.3 NER finetuning results

We evaluate two CS-BERT models versus BERT, Tod-BERT and conversational-BERT for the NER task. We measure the model performance on both entity level and token level. For entity level evaluation, we count a true positive when the model correctly predicts the entire entity (i.e. the beginning and end of an entity). For word level evaluation, we count a true positive if a word is correctly labeled as its entity type, where we separate a sentence into words using spaces. Note that CS-BERT-SSR is not included in this experiment. Our NER data is single utterance-level, and thus is not expected to

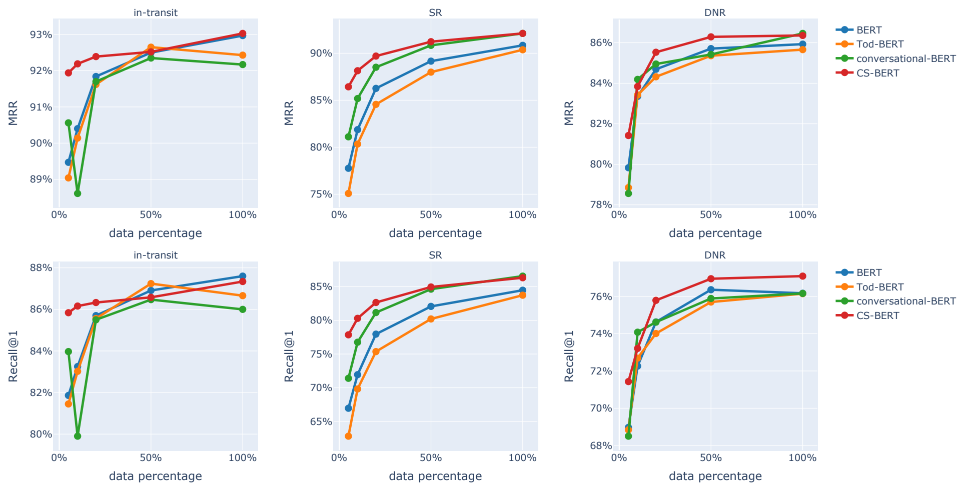
Figure 1: Recall@1 and MRR as a function of finetuning dataset percentage, evaluated on the response ranking task.

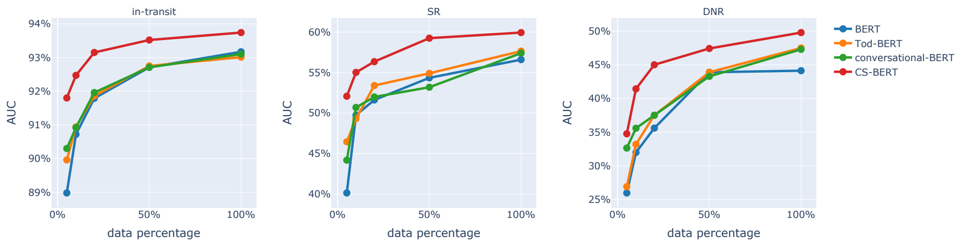
Figure 2: AUC as a function of finetuning dataset percentage, evaluated on the failure detection task.

benefit from the SSR embedding strategy.

Table 4 shows that CS-BERT-MSP performs slightly better than other models on entity and word level F1 score. However, we do not see significant difference in between the different pretrained models. We believe the reasons for this could be: (1) The NER annotated data are turn-level while the CS-BERT models are trained at the conversation level. (2) All pretrained models are sufficient enough to initialize the NER model, resulting in similar performance after finetuning. (3) Our human annotated data is noisy. We estimate that human level entity and word level F1 score are around 91.9% and 92.1%, respectively, based on 5,000 double labeled samples. Thus, we can see all models are reasonably close to human level performance. As a future work, we will improve our data quality by aggregating multiple annotations, and we plan to extend from turn level annotations to conversation level annotations.

6 Summary

In this work, we present three CS-BERT models pretrained on real-world customer service dialogues. We show that in zero-shot response ranking tasks, the CS-BERT models are superior to publicly available pretrained models, and benefit from our dialogue-specific objective function MSP.

We evaluate CS-BERT on three different downstream tasks, and demonstrate that CS-BERT consistently outperforms other non-CS-BERT models on the response ranking and failure detection task, especially in a low-resource data setting. We observe that CS-BERT is superior compared to BERT, as well as compared to two models pretrained on dialogue data, i.e., conversational-BERT and TOD-BERT, thus demonstrating the advantage of in-domain pretraining. However we also observe that the performance gains diminish with the size of the finetuning datasets, with some models reaching parity with CS-BERT when using all available finetuning data.

Due to customer data privacy concerns, we will not publicly release our CS-BERT models nor the datasets used in this work. However, we hope to inspire other researchers to experiment with pretraining their own, domain-specific BERT models. Our experiments demonstrate that these domain-specific pretrained models can improve performance on in-domain downstream tasks and can reduce the need for large amounts of expensive labeled datasets.

References

[1] Devlin, J., Chang, M.-W., Lee, K., et al. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*. 2019.

[2] Radford, A., Wu, J., Child, R., et al. Language models are unsupervised multitask learners. 2019.

[3] Reimers, N., Gurevych, I. Sentence-bert: Sentence embeddings using siamese bert-networks. In *EMNLP/IJCNLP*. 2019.

[4] Garg, S., Vu, T., Moschitti, A. Tanda: Transfer and adapt pre-trained transformer models for answer sentence selection. *ArXiv*, abs/1911.04118, 2020.

[5] Yang, Z., Dai, Z., Yang, Y., et al. Xlnet: Generalized autoregressive pretraining for language understanding. In *NeurIPS*. 2019.

[6] Li, J., Sun, A., Han, J., et al. A survey on deep learning for named entity recognition. *IEEE Transactions on Knowledge and Data Engineering*, 2020.

[7] Liu, Y., Lapata, M. Text summarization with pretrained encoders. In *EMNLP/IJCNLP*. 2019.

[8] Radford, A., Narasimhan, K. Improving language understanding by generative pre-training. 2018.

[9] Liu, Y., Ott, M., Goyal, N., et al. Roberta: A robustly optimized bert pretraining approach. *ArXiv*, abs/1907.11692, 2019.

[10] Clark, K., Luong, M.-T., Le, Q. V., et al. Electra: Pre-training text encoders as discriminators rather than generators. *ArXiv*, abs/2003.10555, 2020.

[11] Iandola, F. N., Shaw, A. E., Krishna, R., et al. Squeezebert: What can computer vision teach nlp about efficient neural networks? *ArXiv*, abs/2006.11316, 2020.

[12] https://huggingface.co/DeepPavlov/bert-base-cased- conversational.

[13] Wu, C.-S., Hoi, S., Socher, R., et al. Tod-bert: Pre-trained natural language understanding for task-oriented dialogue. In *EMNLP*. 2020.

[14] Lu, Y., Srivastava, M., Kramer, J., et al. Goal-oriented end-to-end conversational models with profile features in a real-world setting. In *NAACL-HLT*. 2019.

[15] Srivastava, M., Lu, Y., Peschon, R., et al. Pretrain-finetune based training of task-oriented dialogue systems in a real-world setting. In *NAACL*. 2021.

[16] Howard, J., Ruder, S. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*, 2018.

[17] Gururangan, S., Marasović, A., Swayamdipta, S., et al. Don't stop pretraining: adapt language models to domains and tasks. *arXiv preprint arXiv:2004.10964*, 2020.

[18] Wu, Y., Schuster, M., Chen, Z., et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *ArXiv*, abs/1609.08144, 2016.

[19] Lan, Z., Chen, M., Goodman, S., et al. Albert: A lite bert for self-supervised learning of language representations. *ArXiv*, abs/1909.11942, 2020.

[20] Kitaev, N., Kaiser, L., Levskaya, A. Reformer: The efficient transformer. *ArXiv*, abs/2001.04451, 2020.

[21] Choromanski, K., Likhosherstov, V., Dohan, D., et al. Rethinking attention with performers. *ArXiv*, abs/2009.14794, 2020.

[22] Beltagy, I., Peters, M. E., Cohan, A. Longformer: The long-document transformer. *ArXiv*, abs/2004.05150, 2020.

[23] Sanh, V., Debut, L., Chaumond, J., et al. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *ArXiv*, abs/1910.01108, 2019.

[24] Roller, S., Dinan, E., Goyal, N., et al. Recipes for building an open-domain chatbot. In *EACL*. 2021.

[25] Zhang, Y., Sun, S., Galley, M., et al. Dialogpt: Large-scale generative pre-training for conversational response generation. In *ACL*. 2020.

[26] Kudo, T., Matsumoto, Y. Chunking with support vector machines. In *Second Meeting of the North American Chapter of the Association for Computational Linguistics*. 2001.

A Appendix

dataset	dialogues	# turns
train	40,505,050	317,093,459
dev	151,960	250,423

Table A.1: Pretraining dataset.

context	response
INITIALQUESTION AGENTSTART Can you describe in a few words how I can help you with your recent order? CUSTOMERSTART want to return CUSTOMERSTART but hasn't been delivered yet PROFILESTART paymentmethod_credit_card shippingstatus_delayed carrier_usps	Ok, give me a minute to look into this.

Table A.2: An example of a positive context-response pair created of the annotated response ranking finetuning dataset. Unlike the non-annotated pretraining data, we also use metadata for finetuning. We add "PROFILESTART" into the context as a separator between the dialogue history and the metadata. Each key-value pair in the metadata is concatenated by "_".

dataset	intent	dialogues	# turns
train	in-transit	58,515	174,434
dev	in-transit	4,815	5,000
test	in-transit	6,210	35,794
train	DNR	29,755	123,956
dev	DNR	4,674	5,000
test	DNR	3,104	19,828
train	SR	29,697	295,788
dev	SR	4,636	5,000
test	SR	7,473	95,879

Table A.3: High-quality annotated response ranking finetuning datasets for the intents "item-in-transit" (in-transit), "item-delivered-not-received" (DNR), and "start-return" (SR).

dataset	intent	# dialogues	# turns
train	in-transit	2,928,223	16,661,901
dev	in-transit	9,986	10,000
test	in-transit	9983	10,000
train	DNR	285,520	1,496,534
dev	DNR	9,876	10,000
test	DNR	9,833	10,000
train	SR	328,841	2,024,866
dev	SR	9,872	10,000
test	SR	9,871	10,000

Table A.4: Non-annotated response ranking finetuning dataset that was used to supplement the annotated dataset, for the intents "item-in-transit" (in-transit), "item-delivered-not-received" (DNR), and "start-return" (SR).

context	Failure label
INITIALQUESTION I'm still waiting for a pick up AGENTSTART Can you describe in a few words how I can help you with your recent order? CUSTOMERSTART I'm still waiting for pick up of the return item PROFILESTART membership_active shippingstatus_delivered carrier_fedex	1

Table A.5: An example of the model input we use for finetuning the failure detection task. The model's response pool was not able to handle this customer's problem and thus this conversation turn was labeled as failure. We do not use the predicted response as model input to prevent target leakage.

dataset	intent	# dialogues	# turns
train	in-transit	58,515	282,348
dev	in-transit	4,815	5,000
test	in-transit	6,210	35,794
train	DNR	29,755	164,919
dev	DNR	4,674	5,000
test	DNR	3,104	19,828
train	SR	29,697	391,052
dev	SR	4,636	5,000
test	SR	7,473	95,879

Table A.6: Annotated failure detection finetuning dataset for the intents "item-in-transit" (in-transit), "item-delivered-not-received" (DNR), and "start-return" (SR).

Label	# of entities	Percentage
Name	6304	89.2%
Physical address	168	2.4%
Phone fax	161	2.3%
URL	152	2.2%
Email	128	1.8%
Credit card number	92	1.3%
Unknown PII	47	0.7%
Membership ID	8	0.1%
CVC code	3	0.0%
Bank account	2	0.0%
Date of birth	2	0.0%

Table A.7: NER dataset entity distribution.

pretrained encoder	in-transit					DNR					SR				
	epoch	**Accuracy**	**MRR**	**Recall@1**	**Recall@5**	**epoch**	**Accuracy**	**MRR**	**Recall@1**	**Recall@5**	**epoch**	**Accuracy**	**MRR**	**Recall@1**	**Recall@5**
5%															
BERT	3	98.43%	89.47%	81.86%	98.78%	3	97.86%	79.83%	68.97%	96.09%	3	98.04%	77.76%	66.96%	91.40%
TOD-BERT	3	98.28%	89.04%	81.45%	98.33%	3	97.61%	78.85%	68.84%	95.67%	3	97.27%	75.08%	62.82%	90.85%
conversational-BERT	3	98.67%	90.56%	83.97%	98.93%	3	98.07%	78.56%	68.50%	95.86%	3	98.23%	81.11%	71.40%	93.35%
CS-BERT	2	98.80%	91.94%	85.84%	99.41%	3	98.10%	81.42%	71.43%	97.65%	3	98.65%	86.43%	77.85%	97.18%
CS-BERT-SSR	2	98.71%	91.97%	86.01%	99.37%	3	98.21%	81.26%	71.18%	97.78%	3	98.61%	86.29%	77.62%	97.18%
CS-BERT-MSP	2	98.76%	92.18%	86.33%	99.41%	3	98.18%	81.29%	71.02%	97.40%	2	98.56%	86.21%	77.43%	97.30%
10%															
BERT	3	98.58%	90.40%	83.25%	99.15%	3	98.24%	83.35%	72.26%	96.59%	3	98.24%	81.87%	71.94%	94.41%
TOD-BERT	3	98.51%	90.14%	83.02%	98.98%	3	98.05%	83.42%	72.69%	97.34%	3	97.74%	80.34%	69.82%	93.92%
conversational-BERT	3	98.47%	88.61%	79.90%	99.01%	3	98.33%	84.19%	74.08%	97.73%	3	98.67%	85.18%	76.75%	95.83%
CS-BERT	3	98.86%	92.19%	86.16%	99.50%	3	98.26%	83.84%	73.21%	98.42%	3	98.83%	88.14%	80.29%	97.86%
CS-BERT-SSR	3	98.80%	92.00%	85.79%	99.51%	3	98.26%	84.67%	74.57%	98.27%	2	98.73%	87.91%	79.87%	97.94%
CS-BERT-MSP	3	98.81%	92.15%	86.13%	99.51%	3	98.24%	82.93%	71.21%	98.36%	3	98.86%	88.29%	80.54%	97.86%
20%															
BERT	3	98.73%	91.84%	85.70%	99.31%	3	98.51%	84.69%	74.63%	97.29%	3	98.60%	86.26%	77.95%	96.61%
TOD-BERT	3	98.69%	91.62%	85.56%	99.21%	3	98.29%	84.32%	74.01%	97.22%	3	98.48%	84.56%	75.36%	96.18%
conversational-BERT	3	98.83%	91.70%	85.50%	99.36%	3	98.45%	84.95%	74.62%	97.86%	3	98.83%	88.51%	81.15%	97.66%
CS-BERT	3	98.89%	92.39%	86.33%	99.66%	3	98.59%	85.53%	75.79%	98.09%	3	98.91%	89.70%	82.64%	98.42%
CS-BERT-SSR	3	98.87%	92.35%	86.34%	99.65%	3	98.61%	85.95%	76.14%	98.20%	3	98.77%	89.20%	81.78%	98.37%
CS-BERT-MSP	3	98.82%	92.43%	86.50%	99.62%	3	98.52%	85.60%	75.83%	98.13%	2	98.94%	89.54%	82.33%	98.37%
50%															
BERT	3	98.81%	92.50%	86.91%	99.49%	3	98.66%	85.71%	76.36%	98.25%	3	98.92%	89.16%	82.06%	97.94%
TOD-BERT	3	98.82%	92.65%	87.24%	99.45%	3	98.61%	85.36%	75.70%	98.35%	3	98.63%	87.99%	80.20%	97.72%
conversational-BERT	3	98.84%	92.35%	86.47%	99.57%	3	98.63%	85.42%	75.89%	98.37%	3	99.08%	90.84%	84.62%	98.47%
CS-BERT	3	98.90%	92.52%	86.58%	99.64%	3	98.72%	86.29%	76.95%	98.65%	3	99.04%	91.23%	84.94%	98.84%
CS-BERT-SSR	3	98.87%	92.94%	87.44%	99.68%	3	98.74%	86.08%	76.85%	98.73%	3	99.03%	91.08%	84.72%	98.80%
CS-BERT-MSP	3	98.83%	92.23%	86.06%	99.67%	3	98.71%	86.15%	76.86%	98.66%	3	99.07%	91.07%	84.70%	98.81%
100%															
BERT	3	98.83%	92.97%	87.60%	99.57%	3	98.66%	85.93%	76.17%	98.34%	3	99.02%	90.84%	84.46%	98.62%
TOD-BERT	3	98.83%	92.43%	86.66%	99.50%	2	98.60%	85.66%	76.15%	98.33%	3	98.93%	90.37%	83.72%	98.45%
conversational-BERT	2	98.78%	92.17%	86.00%	99.58%	3	98.71%	86.46%	76.16%	98.12%	3	99.17%	92.11%	86.55%	98.89%
CS-BERT	3	98.98%	93.03%	87.34%	99.70%	2	98.75%	86.36%	77.09%	98.68%	3	99.14%	92.11%	86.31%	99.10%
CS-BERT-SSR	3	98.93%	92.29%	86.09%	99.70%	2	98.70%	86.41%	76.84%	98.67%	3	99.12%	92.15%	86.39%	99.11%
CS-BERT-MSP	3	98.96%	92.93%	87.17%	99.74%	3	98.70%	86.46%	76.63%	98.65%	3	99.08%	92.07%	86.27%	99.07%

Table A.8: Full results from finetuning the response ranking task on the six pretrained models at different dataset percentage levels, evaluated on the annotated response ranking test sets.

pretrained encoder	**in-transit**		**DNR**		**SR**	
	epoch	**AUC**	**epoch**	**AUC**	**epoch**	**AUC**
5%						
BERT	3	88.98%	4	25.97%	2	40.13%
Tod-BERT	3	89.96%	4	26.90%	3	46.45%
conversational-BERT	3	90.30%	4	32.64%	2	44.18%
CS-BERT	3	91.80%	4	34.76%	3	52.08%
CS-BERT-SSR	3	91.90%	4	38.85%	2	51.97%
CS-BERT-MSP	3	92.08%	4	40.47%	3	52.55%
10%						
BERT	2	90.72%	3	31.99%	3	49.74%
Tod-BERT	2	90.94%	3	33.20%	3	49.31%
conversational-BERT	2	90.92%	3	35.58%	3	50.69%
CS-BERT	2	92.47%	3	41.43%	3	55.02%
CS-BERT-SSR	2	92.65%	3	40.88%	3	56.67%
CS-BERT-MSP	2	92.54%	3	42.81%	3	53.95%
20%						
BERT	2	91.79%	2	35.60%	2	51.62%
Tod-BERT	2	91.86%	2	37.51%	2	53.40%
conversational-BERT	2	91.96%	2	37.51%	2	51.99%
CS-BERT	2	93.15%	2	45.01%	2	56.35%
CS-BERT-SSR	2	92.90%	2	45.75%	2	56.67%
CS-BERT-MSP	2	92.92%	2	45.65%	2	56.13%
50%						
BERT	2	92.71%	2	43.90%	2	54.35%
Tod-BERT	2	92.75%	2	43.90%	2	54.91%
conversational-BERT	2	92.71%	2	43.29%	2	53.19%
CS-BERT	2	93.52%	2	47.43%	2	59.25%
CS-BERT-SSR	2	93.59%	2	47.70%	2	59.48%
CS-BERT-MSP	2	93.53%	2	48.60%	2	58.94%
100%						
BERT	2	93.17%	1	44.13%	2	56.60%
Tod-BERT	2	93.01%	2	47.50%	2	57.65%
conversational-BERT	2	93.10%	2	47.30%	2	57.40%
CS-BERT	2	93.74%	1	49.79%	2	59.94%
CS-BERT-SSR	2	93.79%	1	49.26%	2	59.95%
CS-BERT-MSP	2	93.77%	1	48.76%	2	60.32%

Table A.9: Full results from finetuning the failure detection task on the six pretrained models at different dataset percentage levels, evaluated on the failure detection test set.