SupportNet: Neural Networks for Summary Generation and Key Segment Extraction from Technical Support Tickets

Vinayshekhar Bannihatti Kumar **Mohan Yarramsetty** **Sharon Sun** **Anukul Goel**
Amazon Web Services
Seattle, WA, USA
{vinayshk, ymohank, sharosun, anukul}@amazon.com

Abstract

We improve customer experience and gain their trust when their issues are resolved rapidly with less friction. Existing work has focused on reducing the overall case resolution time by binning a case into predefined categories and routing it to the desired support engineer. However, the actions taken by the engineer during case analysis and resolution are altogether ignored, even though it forms the bulk of the case resolution time. In this work, we propose two systems that enable support engineers to resolve cases faster. The first, a guidance extraction model, that mines historical cases and provides technical guidance phrases to the support engineers. These phrases can then be used to educate the customer or to obtain critical information needed to resolve the case and thus minimize the number of correspondences between the engineer and customer. The second, a summarization model that creates an abstractive summary of a case to provide better context to the support engineer. Through quantitative evaluation we obtain an F1 score of 0.64 on the guidance extraction model and a BertScore (F1) of 0.55 on the summarization model.

1 Introduction

It is of paramount importance to AWS Support organization to reduce the resolution time of customer cases to ensure their business runs seamlessly without any downtime. We have a unique challenge in that the customers' issues can be deeply technical and require technically skilled agents to resolve it. There is a rapid increase in the number of users of the services offered by our cloud company and it is important to improve tooling for Support Engineers (SEs) in order to scale. A significant portion of customers' cases are business critical and time-sensitive.

Cases are created by customers for several reasons such as guidance about a specific service or troubleshooting when a production service is down. A typical customer creates a case with a title of the case (case title) and a detailed correspondence on their issue as a part of the case (communication text). The agents then have to read this case, understand the customer's problem and suggest ways for them to resolve the issues. This requires the agents to spend a lot of time to completely read the case and then guide the customer to solve the problem. We have developed two systems that can use the inputs from the customer in the form of case text (case title + communication text) to speed up the agent's time to resolve a case. The first system is a summarization system that presents the customers problem to the agent to give them a head start in tackling the case. The second system provides them snippets from similar historical cases to reduce the time the agent takes to respond to the customer.

Prior work in the domain of customer support has focused on improving the time to resolve a case by improving routing and detecting the customer problems into one of several predefined categories (Gupta et al., 2013; Hui and Jha, 2000; Muni et al., 2017; Molino et al., 2018). While these methods reduce average total time to resolve issues throughout the case journey, they do not focus on reducing the active handle time by SEs, i.e., the time a SE has to invest to understand and respond to the customer. In our work, we address this gap by introducing two novel solutions as previously described.

SEs often look to similar previously resolved cases when beginning to tackle a new case according to a few internal studies. A previous similar case provides them troubleshooting resources, hints on root causes, and guidance material that they reuse on the new case. These resources from previous similar cases have been found to reduce the handle time by SEs, but it takes time for agent to browse through the results. Hence, we built a knowledge mining system based on NLP that allows SEs to efficiently look up historical cases without having to read the whole case.

Typical technical cases contain many conversations, and reading through them is a time-consuming process. In order to solve this problem, a system was introduced - State of the Case (SOC) - where SEs update a summary of the current state of a case. These updates are made manually by SEs while they hand the case back to the customer or to another internal team. In addition to serving as a smooth transition between SEs throughout the case lifecycle, the SOC of a case was intended to serve as its expert summary view, eliminating the vagueness and jargon that may be present in customers' case text. However, the additional manual effort to fill SOC resulted in a low adoption rate of 9.8% over time. We introduce the solution in Section 3.2 automating SOC generation based on customer communications. Table 1 provides a simulated example of the data we have. The SEs can use the SOC to get a head start in solving the case.

Customer case text **Case Title:** Server down because of full volume **Case description:** We had our server go down this evening because the @gig volume of our EC@ server full. To avoid this in the future, I have two questions: - How can we know the amount of free space left in the Ec@ volume? - Is there a way we can setup alerts to monitor free usage? Thank you for your help in this matter. Have a good day, Instance ID(s): How can we track the storage / volume usage of our EC@ instances
State of Case(SOC) Case Summary: how can we track the free space remaining on the volume of an EC2 server ?@- how do we set up alerts at certain thresholds to know to act?

Table 1: The customer can describe their problem in several different ways. The state of the case summarizes the customer issue into something that is actionable.

We show that the summary version that a SE would write in SOC can be automatically generated using the state-of-the-art encoder (Bert (Devlin et al., 2018))-decoder (GPT-2 (Radford et al., 2019)) models with cross attention. We compare this model against recent baseline models such as Bart (Lewis et al., 2019) and on traditional encoder decoder models using LSTMs. We present findings that support using this model in production when compared to a more memory efficient model such as LSTM. Any conditional generation task requires a parallel dataset on which the model is trained on. Our dataset on the other hand has labeled data for the encoder text as well. This allows us to introduce a classification loss on the encoder to obtain better encoder representations that can be fed to the decoder. This also allows us to jointly train an encoder and a decoder with a simultaneous multi-task objective. We perform this novel experiment to show the efficacy of training the encoder with a cross entropy loss function while the decoder is trained using Maximum Likelihood Estimation (MLE).

Thus, the contribution of this work can be summarized as follows:

1. We present a model that is capable of mining technical guidance phrases from support cases.

2. We present a summarization model that generates a concise summary of customer problems from the communication text. We also compare several summarization models and discuss the potential impact of productionizing our model. In addition, we performed multi-task learning on the encoder to determine if it can improve decoder's performance.

3. We present the results from a human subject study to show the usefulness of our solution. Initial results suggest that the SEs considered the summary generated by our model as a good starting point to solve a case.

2 Related Work

As mentioned earlier, Molino et al. (2018) built systems that could categorize case issues into predefined categories. They also suggest predefined templates to SE. Specific details of the case are not taken into picture while suggesting these templates to make them more appropriate. Our summary model on the other hand ingests the context of the case and generates a personalized problem

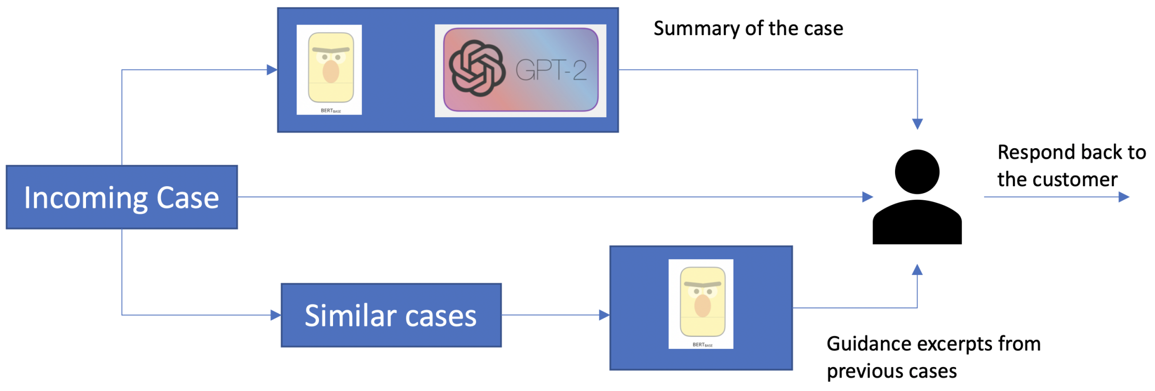
Summary of the case
GPT-2
Incoming Case
Respond back to the customer
Similar cases
Guidance excerpts from previous cases

Figure 1: (a) Component 1 reads the contents of the incoming case and sends a summary to the support agent to get a head start. (b) Component 2 reads the contents of the historical cases and provides guidance excerpts for the agents to respond back to the customer.

template that a SE can use to update the state of case.

Prior work (Godse et al., 2018; Chess et al., 2007; Brittenham et al., 2007; Pathak and Khandelwal, 2017) has looked at providing better Information Technology Service Management (ITSM) to their customers by building resources that help the customers diagnose their own issue and find a solution for it. However, customers usually try to self-diagnose their issues before cutting a case. Hence, our solution focuses on helping the SE and improve their efficiency.

Other work (Gupta et al., 2013; Hui and Jha, 2000; Muni et al., 2017) has looked at the use of the support case text along with other metadata to classify the intent of the case and improve routing. We on the other hand analyze the case text to provide assistance to SE in their day-to-day tasks.

3 System and Model Overview

Figure 1 shows the overall architecture of the proposed system. There are two major components that we propose in this work. The first one uses a Bert and GPT-2 model to allow the SE to get a head start in solving the case. The second component runs on the case text of previously solved cases to provide the SE with guidance phrases. They can use the predicted guidance phrases to understand how the case can be solved and to also construct a response back to the customer.

In this section, we provide details of the models that were built in these components and the steps that we took to train them.

3.1 Guidance Extraction Model

Support cases that we receive from customers are filled with jargon rich text that takes highly skilled agents to read and understand. In order to train the models that can understand this text, we need large amounts of supervised data that is very expensive to obtain as it requires expert annotations. However, we can train large Language Models (LMs) with the vast amounts of self-supervised case text that enables the models to understand this jargon filled technical domain.

Following Lee et al. (2020); Beltagy et al. (2019) we continue the pre-training of the model presented by Devlin et al. (2018) . We continue the pre-training of the Bert model for another $60,000$ steps on the support cases that we have received in the period of $2019-20$. We call this model SupportBert. We show that this model outperforms the Bert base model trained on English Wikipedia and the Book corpus (Zhu et al., 2015) on our guidance phrase prediction task. We follow the standard procedure of fine-tuning this model on a labeled dataset of guidance excerpts. More about this dataset is presented in Section 4. We try several variants of the models while pre-training and the details of the experiments are presented in Section 5.

3.2 Summarization Model

We use a Seq2Seq (Sutskever et al., 2014) model with the cross attention as our baseline model. We use a Bert encoder and a GPT-2 decoder to summarize the case content. For every word W_i that belongs to the case description and the communication text, we pass that word through the

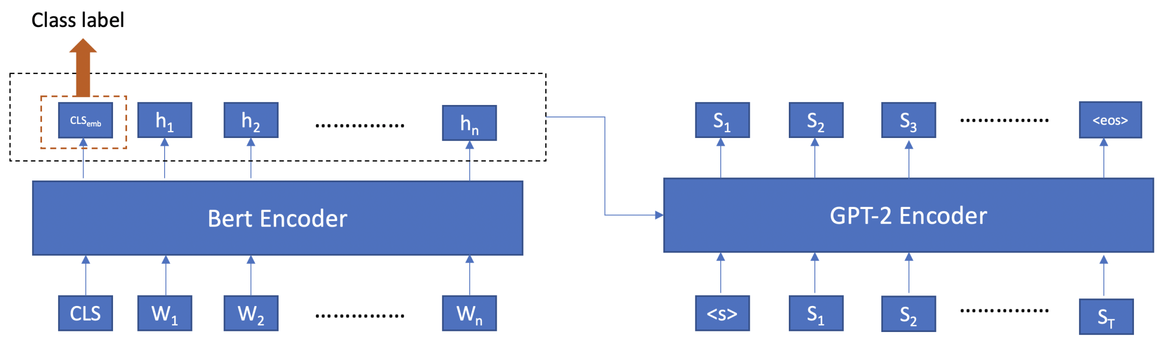

Figure 2: The complete setup of our summary model is shown above. We use the output of the GPT-2 decoder as a summary of the case description fed into the Bert Encoder. For the multi-task experiment we use the class label to add more gradients to the encoder.

Bert encoder to obtain the contextualized representations of the case content. We use Maximum Likelihood Estimation (MLE) to train the decoder on the case summary $(S_1...S_{T'})$. The case summaries entered by SEs on historical cases are used to train our model. Here S_i represents every word of the summary at a time step t'. T' represents the overall length of the summary. Our overall architecture is shown in Figure 2.

We first pass the words(W_i) into the Support-Bert model to get a contextualized representation of every word(h_{enc}).

$$h_{enc}^T = Bert\_encoder(W_1, W_2....W_N) \quad (1)$$

We then use these hidden states as the keys to the cross-attention units of the GPT-2 decoder. At each stage of the GPT-2 decoder we will see a probability distribution on the vocabulary(P_t^V).

$$P_t^V = GPT\_2(h_{enc}^T, S_{1...t}) \quad (2)$$

We use the SOC described in Table 1 for training the decoder with MLE. We want to maximize the log likelihood of the probability of the true word, in other words we want to minimize the negative log likelihood of the probability of the true word.

$$loss_t = -logP(W_t^{true}) \quad (3)$$

Total decoder loss is averaged cross entropy loss at each time step of the decoder.

$$\text{loss}_{decoder} = \frac{1}{T_{decoder}} \sum_{t=1}^{T_{decoder}} loss_t \quad (4)$$

During inference, we use the words generated by the decoder till time t and the Bert embeddings to produce the word at time $t+1$.

$$S_{t+1} = \arg\max_V [\text{softmax}(GPT\_2(h_{enc}^T, S_t))] \quad (5)$$

3.2.1 Multi-task training

In this variant of the summarization model, we also predict the issue category of the case text along with generating the summary. We have a unique corpus that has a label for the encoder text to train the encoder and SOC text to train the decoder. This enables us to jointly train the encoder and the decoder with both these loss functions. The encoder receives gradients from not only the MLE objective of the decoder but also the cross-entropy loss from issue categorization (241 pre-defined categories). Multi-task learning has shown to improve the performance of the Bert base model (Liu et al., 2019). However, each task head during this training phase is trained independently as parallel labels are not available. Also, there is a lack of a public corpus that enables us to jointly train the encoder on a classification task and the decoder on a text generation task. Our corpus allowed us to perform this experiment.

$$\text{loss}_{encoder} = \sum_{i=0}^{241} P(X)log(Q(X)) \quad (6)$$

$$\text{loss} = \text{loss}_{encoder} + \text{loss}_{decoder} \quad (7)$$

4 Datasets

4.1 Guidance Extraction

For the purpose of guidance extraction we asked annotators to label paragraphs from support cases as Technical Guidance (**T.G**) and Educational Guidance (**E.G**). Using this paragraph labeled case dataset, we want to identify if a paragraph is a guidance or not. As a first step we will combine both these guidance into one bucket and classify if a paragraph is a guidance para-

# Examples	# T.G	# E.G
2050	109	88

Table 2: The number of different types of guidance present in our guidance extraction dataset.

# Train	# Test	# Categories
104892	5000	241

Table 3: Number of training and testing samples used for the summary model. Total categories column indicates the number of categories that the encoder text could be classified into. This was used for training the encoder in a multi-task setup.

graph or not. The statistics of this dataset is shown in Table 2.

4.2 Summarization Dataset

We used an internally available parallel corpus for training our summarization model. Our dataset contains the case subject, first communication and the **Subject Matter Experts (SMEs)** annotated customer problem. We show the dataset statistics in Table 3. We use the case title and the first communication as the input to the model. The model summarizes the customer problem.

5 Experiments:

5.1 Guidance Extraction Model

5.1.1 Experimental setup:

The majority baseline would produce an F1 score of 0.13 if recall is set to 1 by predicting everything as guidance phrase. This shows that the dataset is highly skewed and it is not easy to get good performance with random guessing. Since the number of examples in the test set is not very high (20% of all annotated data in Table 2), we decided to perform a 5-fold validation and average the model performance. For each fold of data, we stop training after 3 epochs similar to (Devlin et al., 2018). We then use the 20% test set for each fold to calculate the accuracy of the model prediction with the ground truth annotation. In order to control for variance, we repeat the experiment 5 different times and average the results. We compared the following 4 variants of the model:

Off the shelf Bert (OSB): We used Bert-base model that was trained on English Wikipedia and the Book Corpus and then fine-tuned the model against our own dataset. During fine-tuning, we used a batch size of 16, max length of 512, learning rate of 5e-5 with weight decay ($\epsilon = 1e-8$).

Model	P	R	F1
OSB	0.665	0.6114	0.6116
PB	0.7178	0.6268	0.6456
PBK	0.6576	0.5386	0.5636
BUL	0.6742	0.6204	0.6306

Table 4: Performance of different guidance extraction models. These performance results have been averaged with 5-fold validation and 5 different runs to control for variance.

Bert – Pre-trained with support case data (PB): Since the language in support cases is wildly different from that of Wikipedia and the Bookcorpus, we continued the pre-training of the Bert model on a corpus of support cases. We used a total of 236,354 cases to continue pretraining the Bert model. We then fine-tuned this Bert model with the same experimental setup as **OSB**.

Bert Pre-trained with Keywords in inputs (PBK): We observed that there were a lot of technical terms in our dataset that have a different contextual meaning (e.g. VPC, route). Hence, we hypothesized that we should add technical keywords to our model's vocabulary. To facilitate this, we trained a new WordPiece model (Wu et al., 2016) on our corpus of support cases. There were 1662 tokens in Bert's vocabulary after adding these words. The rest of the training pipeline was similar to **PB**.

Bert model with under sampling Limit Increase Cases (BUL): When our customers want to add more resources to their existing account they create a Limit Increase Case with us. These cases follow a very generic template (canned email) that might not be very useful to the model. In order to test this hypothesis, we under sampled those cases and re-ran the pre-training and fine-tuning experiment without adding additional keywords.

5.1.2 Experiments Discussion

From Table 4, we see that the best performing model on the guidance phrase dataset is the PB model. This observation is consistent with several other works (Beltagy et al., 2019; Lee et al., 2020). We see that the model performance significantly drops when we add the custom keywords. This is because the word embeddings for these words are trained from scratch while the word embeddings of the Bert base vocabulary have already been tuned. We do see a 2% increase in F1 by under sampling limit increase cases but the performance is still lower than the PB model.

5.2 Summary Model

5.2.1 Experimental Setup

We train the encoder-decoder model for 3 epochs on the training data mentioned above. For the encoder we use a Bert base model. The decoder is the GPT-2 model from OpenAI. The keys to the self-attention units of the GPT-2 decoder are the final hidden states of the Bert encoder. We used a batch of 16, max length of 512, Adam (Kingma and Ba, 2014) optimizer learning rate of $5e-5$ with weight decay of $(\epsilon = 1e-8)$. We experiment with several variants of the encoder-decoder model. They are discussed below:

LSTM Based encoder-decoder (LSTM): We implement an LSTM based encoder-decoder model with attention (Sutskever et al., 2014; Bahdanau et al., 2014). There are 512 units in both the encoder and the decoder with 2 layers. We use an Adam optimizer with learning rate of 0.3.

Bert encoder with GPT-2 decoder (BG): We implement a Bert encoder with a GPT-2 decoder. The final hidden states of the Bert base encoder act as the key to the self-attention blocks (Vaswani et al., 2017) of the GPT-2 module.

Pre-trained Bert encoder with GPT-2 decoder (PBG): This model is similar to the **BG** model. But we used the pre-trained Bert model from Section 5.1.1.

Bert encoder with GPT-2 decoder + encoder loss (BGE): This model is similar to the **BG** above. But we also include a multi-task objective that classifies the input text into one of the 241 categories. We want to evaluate if there is any advantage to either the text generation phase or the classification phase by performing this multi-task learning.

Pre-trained Bert encoder with GPT-2 decoder + encoder loss (PBGE): This model is a combination of the PBG model and BGE model. We pre-train the Bert model and use the multi-task loss to get better representations of the input text.

5.2.2 Experiments Discussion

Since Rouge is considered as the industry standard metric for summary tasks (Zhang et al., 2020; Lewis et al., 2019), we compare different models based on ROUGE-L (Lin, 2004) metric. This comparison is consistent with all the other text generation metrics presented in Table 5. We observe that the best performing model is the BG model. The performance is almost the same as the PBG model. However, when we compare the models using BertScore (P, R, F1) (Zhang et al., 2019) we see that PBG model slightly outperforms the BG model. BertScore has been found to have more correlation with human judgment. Further human evaluation is required to get the nuanced differences between the BG model and the PBG model. Several works have investigated the use of multi-task learning for classification (Liu et al., 2019, 2017) and have found that training the model with several losses increases model performance. In our case we see that the multi-task objective deteriorates the model text generation metrics. We hypothesize that both the encoder and the decoder try to modify the Bert representations to suit their task at hand leading to this degradation in the performance. We also looked at the outputs from the Bart (Lewis et al., 2019) model, but observed that the Bart model did not perform abstraction and merely copied the words from the customer text in the decoder response.

We also investigate if we can achieve higher performance on the encoder classification if we train with the multi-task objective. From Figure 3 we see that all the models achieve a similar test performance after 3 epochs of training. However, it is interesting to see that the classifier achieves better performance in the initial steps of training with both the PBGE and the BGE models. The comparison is made with respect to taking an off the shelf Bert model and training with the encoder objective described in Section 3.2.1.

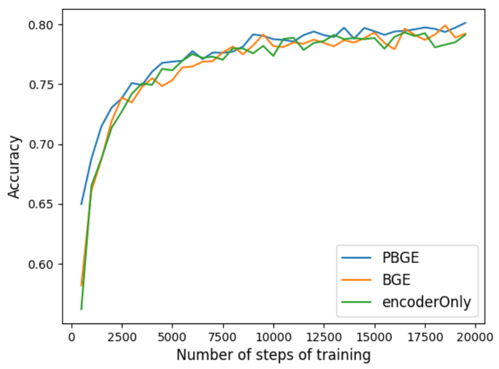

Figure 3: Comparisons of different encoder classifier accuracies against the number of steps the model is trained on.

Model	Bleu-1	Bleu-4	Cider	ROUGE	DIST-1	DIST-2	DIST-3	P	R	F1
PBGE	0.130	0.063	0.190	0.225	0.043	0.170	0.319	0.529	0.512	0.515
LSTM	0.066	0.032	0.062	0.113	0.015	0.119	0.228	0.479	0.415	0.439
BGE	0.131	0.066	0.206	0.294	0.045	0.211	0.415	0.556	0.527	0.535
BG	0.143	0.080	0.220	0.401	0.064	0.275	0.470	0.568	0.539	0.546
PBG	0.132	0.074	0.227	0.436	0.062	0.271	0.470	0.580	0.535	0.551

Table 5: Comparison of all the models. Based on the ROUGE score we see that the BG model performed as well as the PBG model. The results obtained from BertScore(P, R, F1) closely resembles that of our human evaluation

The performance of a simple model like LSTM is not close to the transformer based model. Even though the LSTM models can give a significant gain in inference time while deploying the model in production, from the model performance above we can see that we need to have infrastructure in place to productionize the transformer based models for our use case. The transformer based models took a total time of 15 minutes to decode the test set of 5000 examples which adds an average latency time of 180ms to summarize each example.

Another important metric that one should look at is the number of distinct phrases that the Seq2Seq model is able to generate. Seq2Seq models are known to suffer from the dull response problem (Gupta et al., 2019a). If the DIST-* metrics are high, that shows that the model is able to generate more diverse outputs when presented with different input scenarios (Li et al., 2015; Gupta et al., 2019a). From the table we can see that the DIST-3 metric which captures the number of unique trigrams is highest for the BG model. The next closest model is the PBG model. We also observe that adding encoder loss does not improve the distinct scores.

5.2.3 Human Evaluation

Since the PBG model had the best BertScores and was pre-trained on support cases we used the summaries from this model for human evaluation. While automatic evaluation metrics captures many aspects of the ground truth with the generated text they have been criticized for the lack of semantic understanding (Gupta et al., 2019b; Sulem et al., 2018). In order to evaluate the efficacy of the summaries generated by our models and to validate its usefulness, we conducted a human subject study with the 2 subject matter experts. We randomly sampled 50 cases from the test set and asked them to provide feedback on the accuracy of the summary and if the summary generated would form a good starting point to write the summary of the current case. We observed that 23/50 were considered as a useful summary from both the SEs. 10/50 were considered as not useful by both. The other 17 had mixed ratings. If we consider a summary useful if either of the individual annotated it as useful, then 40/50 were useful summaries.

5.2.4 Qualitative Analysis:

We show some of the summaries generated by our model in Table 6. As we can see from the model generations, the model has learned to predict the technical language of our domain. In row 1 and 2 of Table 6 we see that the model has learned to predict the customers' problem well based on the communication text. This is because of the knowledge from the historical cases where in other customers had similar problem. We can also see from example 3 that the model has learned to associate CPU spikes with Internet Information Server. When we manually inspected a few samples (e.g. row 4) from the human subject study and spoke to our participants, we found that the model was penalized because it tried to generate a very specific customer problem that might not make sense semantically. We leave the introduction of semantic knowledge into these models as future work.

6 Application of the models to Support Businesses

Extraction of guidance phrases from case text of previous cases can provide quick tips to SEs and make it easier for them while working on a case. Every SE is required to write a reply to the customer with the steps they can take to resolve the issue. These guidance phrases will help them reuse the text enabling them to spend less time on the reply email.

The summary models we have built can be used in a variety of SE tasks. We can use the automated summary to provide the context of the

Customer Text	**Ground truth**	**Generated Summary**
Transfer t3 instance from one account to anotherHello, There is a t@.large reserved instance in account @ and we would like to transfer it to the other account that we have	Cx would like assistance on transferring the t2.large Reserved instance from account @ to account @	Cx wants to transfer instance from one account to another
Server down because of full volumeHello, Our Servers went down this evening because the volume of our EC@ server was completely full. I have a couple of questions to avoid this in the future: - How do I know the amount of free space left in the Ec@ volume? - How do I setup alarms to monitor free usage? Have a good day,	how can we track the free space remaining on the volume of an EC2 server?@- how do we set up alerts at certain thresholds to know to act?	Customer wants to know how to monitor the disk usage of the EC2 instance
Performance issue cpu spikesWe have noticed CPU spikes on an instance that is running an IIS site up to @We stopped the site and cpu is still spiking up to @ when running with few services installed. Issue occurs only when our monitoring services are running @am to @pm. I need help trying to understand what is causing this spike with http. Instance ID(s): i-@f@e@a Timestamp (YYYY-MM-DD HH:MM:SS UTC): @/@/@	We would need help with monitoring http traffic going in/out of the box and more details about cpu load by process name.	Customer experienced high CPU spikes on an instance that is running on IIS and wanted to know the cause
Chat: S@ access denied I am getting an error with IAM user or role experiencing the issue: arn:aws:iam::@:role/ with this role. Not sure what the problem is. Could you help?	Cx was facing ""lambda access error: lambda: GetAccountSettings"" error and needed assistance with the same	Customer was getting access denied error while trying to access S3 bucket via IAM user

Table 6: Examples of a few qualitative results produced by our model. Please note we have simulated the queries in the table to preserve customer privacy.

case to the next SE who picks up the case. When presenting similar cases to the SE, we can provide the summaries to let them quickly identify if the case is useful. The summaries that we generate can also be used as a starting point when the SEs are tasked with completing the SOC.

7 Conclusion

It is important for us to scale the support business with our rapid user growth. We describe two components of a system that aims to reduce the time spent by SE in resolving a support case. The aim of this work is to promote research at the intersection of NLP and support business. Using our models, we were able to achieve an F1 score of 0.64 on the guidance extraction problem and BertScore (F1) of 0.55 on the summarization problem. These promising results shows us that they can be deployed in production, create impact and help SEs in their day-to-day tasks. We hope this contribution can lead to better tools that can improve the tooling necessary for support agents to provide a rich customer experience.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. Scibert: A pretrained language model for scientific text. *arXiv preprint arXiv:1903.10676*.

Peter Brittenham, R Russell Cutlip, Christine Draper, Brent A Miller, Samar Choudhary, and Marcelo Perazolo. 2007. It service management architecture and autonomic computing. *IBM Systems Journal*, 46(3):565–581.

David M Chess, James E Hanson, John A Pershing, and Steve R White. 2007. Prospects for simplifying itsm-based management through self-managing resources. *IBM Systems Journal*, 46(3):599–608.

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*.

Neha Atul Godse, Shaunak Deodhar, Shubhangi Raut, and Pranjali Jagdale. 2018. Implementation of chatbot for itsm application using ibm watson. In *2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)*, pages 1–5. IEEE.

Narendra Gupta, Mazin Gilbert, and Giuseppe Di Fabbrizio. 2013. Emotion detection in email customer care. *Computational Intelligence*, 29(3):489–505.

Prakhar Gupta, Vinayshekhar Bannihatti Kumar, Mukul Bhutani, and Alan W Black. 2019a. Writerforcing: Generating more interesting story endings. *arXiv preprint arXiv:1907.08259*.

Prakhar Gupta, Shikib Mehri, Tiancheng Zhao, Amy Pavel, Maxine Eskenazi, and Jeffrey P Bigham. 2019b. Investigating evaluation of open-domain dialogue systems with human generated multiple references. *arXiv preprint arXiv:1907.10568*.

Siu Cheung Hui and G Jha. 2000. Data mining for customer service support. *Information & Management*, 38(1):1–13.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2020. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2015. A diversity-promoting objective function for neural conversation models. *arXiv preprint arXiv:1510.03055*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2017. Adversarial multi-task learning for text classification. *arXiv preprint arXiv:1704.05742*.

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

Piero Molino, Huaixiu Zheng, and Yi-Chia Wang. 2018. Cota: Improving the speed and accuracy of customer support through ranking and deep networks. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 586–595.

Durga Prasad Muni, Suman Roy, Yeung Tack Yan John John Lew Chiang, Antoine Jean-Marie Viallet, and Navin Budhiraja. 2017. Recommending resolutions of itil services tickets using deep neural network. In *Proceedings of the Fourth ACM IKDD Conferences on Data Sciences*, pages 1–10.

Ramesh C Pathak and Pankaj Khandelwal. 2017. A model for hybrid cloud integration: With a case study for it service management (itsm). In *2017 IEEE International Conference on Cloud Computing in Emerging Markets (CCEM)*, pages 113–118. IEEE.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Elior Sulem, Omri Abend, and Ari Rappoport. 2018. Bleu is not suitable for the evaluation of text simplification. *arXiv preprint arXiv:1810.05995*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27:3104–3112.

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.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter Liu. 2020. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In *International Conference on Machine Learning*, pages 11328–11339. PMLR.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2019. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pages 19–27.