A Framework for Mining Speech-to-Text Transcripts of the Customer for Automated Problem Remediation

Prateeti Mohapatra[1], **Gargi Dasgupta**[2]
[1]IBM Research, India
[2]IBM Software, Yorktown Heights, USA
pramoh01@in.ibm.com, gargi.dasgupta@ibm.com

Abstract

Technical support services get several thousand voice calls every year. These calls vary across a range of technical issues or maintenance requests for a suite of hardware and software products. On receiving the call, a support agent creates a service request artifact that contains her interpretation of the customer's problem. This service request goes through the life cycle of the problem remediation process with the resolution also being recorded as part of the service request. It has been empirically observed that the actual complaint voiced by the customer is often different from the recorded interpretation in the service request. The service request created by support agents runs the risk of missing key information elements present in the customer voice records. In this paper, we build a framework that taps into voice calls and uses unsupervised and supervised learning methods to enrich the service requests with additional information. The enriched data is then used for automated problem resolution.

Introduction

While technical support services use multiple channels of communication, customers still prefer to contact support agents over the telephone. Big support companies employ several thousands of agents who continuously communicate with customers. Through the interplay of two processes of service request creation and problem remediation (Figure 1), customers' complaints get closed by support services agents. Technical support agents can be broadly classified into generic and expert agents, where generic agents are mainly customer-facing while expert agents help in solving the actual issues.

The service request creation process begins with a customer calling for technical support. A generic agent creates a service request containing the customer complaint, records the customer complaint as part of the description text, and forwards it for further processing in the problem remediation process. In the problem remediation process, the expert agent troubleshoots the problem based on the service request. The main objective of the agent is to close the service request as efficiently and as quickly as possible.

Several human-induced challenges can arise due to the interplay of these two processes:

- The generic agents fail to capture the conversation correctly due to a lack of time, technical knowledge, or clear understanding. Their summaries thus remain incomplete and sometimes erroneous, thereby not reflecting the concerns of the customer correctly.
- Because of incomplete information, an expert agent may have to engage in additional conversation(s).
- The time to resolve the problem gets longer as the customer gets directed to multiple agents.
- This in turn can frustrate the customer when she needs to repeat all the information to another agent.

For troubleshooting, the expert agent performs a search in the knowledge base (past service requests, knowledge articles, troubleshooting guides, etc) for known problems and their resolutions. The more specific a search is, the more relevant are the returned answers. Relevant answers can be sent to the customer, who can accept them and agree to close the request. In case relevant results are not found or the answer sent does not solve the issue, the customer is re-engaged to provide more details. This back-and-forth can continue for a while till closure is reached. By studying the service requests and by shadowing the expert agents as they troubleshoot, we had the following observations:

- Support conversations contain multiple sub-structures (*attributes*) within them: a description of the problem at hand (*symptom*), a description of the activities made to mitigate the problem (*activity*), and an explicit request from the customer (*intent*).
- The service requests created by agents capture partial symptoms most of the time. For long and complex problem descriptions, much of the symptom information is missed. Also, only $18 - 20\%$ of the time are the reported activities captured in the request.
- The final search that helped close the request was augmented by additional information obtained from the customer during the re-engagement process.

Based on observation 2, an investigation team looked at the speech-to-text (STT) transcripts of service requests that took over thirty minutes to solve. The study revealed that often times the voice call contained important information that could have significantly affected the closure time of the service requests. Table 1 gives examples of service request

Verbatim Client Questions	Documented Service Request
" ... the tape drive is dirty ... attempted cleaning ..."	"The tape drive is dirty."
"... functional test center does not support the power builder application ..."	"The software does not support the power builder application. "
"... some windows logs are being truncated and they're using the MSRPC agent less gathering technique ..."	"some windows logs are being truncated"
"... I am getting some errors in my console for the tape library unit I have ..."	"Getting errors on the console"
"... having problem with the SPSS modeler, the regression is not opening up ..."	"Problem with SPSS modeler"

Table 1: Examples of information missing in service requests but in audio transcript

excerpts that were different from what was present in the corresponding voice call. This is where our proposed framework comes in, augmenting service requests with enriched information extracted from voice calls. This enriched information helps focus the search on returning more relevant results.

Automation of the problem-solving process in the support domain has remained an active area of research (Agarwal, Sindhgatta, and Sengupta 2012; Zhou et al. 2015). Most of these systems resort to information retrieval (IR) mechanisms that match the customer issue to historical content and bring them up as search results. Currently, the only source of input for these IR systems is the text captured in the service requests. When the data captured by generic agents is incomplete (for reasons cited above) the efficacy of the IR systems remains poor. In this paper, we also show how IR systems can be improved using STT transcripts in addition to agents summaries by accurately capturing the customers concerns, enriching the service request, and finally, improving the quality of response from an IR system. In the long term, this reduces re-engaging with the customer for information that she has already provided. We show how information retrieval systems can be improved using speech data in addition to text. Our goal in this work is to leverage the information from the voice conversations between the customer and the agent, to (a) accurately capture the customer's concerns, (b) enrich the service request, and finally, (c) improve the quality of response from an IR system. We show that our framework affects the information retrieval results positively, which leads to an improvement in the problem remediation process.

Related Work

Automatic Speech Recognition (ASR) systems have been trained for years using deep neural nets to create language models, acoustic models, and decoders Mohamed, Dahl, and Hinton (2012). However, the state of the art in large vocabulary speech recognition is far from perfect. Their error rates for phone conversations that have an open vocabulary are still high (Padmanabhan et al. 2002). This entails the need for further text processing on ASR transcripts. Speech summarization literature comprises sentence level (Christensen et al. 2004) or word level (Hori and Furui 2004) extraction. Summarizing into headlines and summaries on clean speech data like news works well with supervised methods (Maskey 2008). However, in the domain of noisy transcripts with a large technical vocabulary, these methods fail to scale.

There exists work in the speech-to-text literature on using the ASR output to extract "key" information from them. Roy et al (Roy and Subramaniam 2006) used the output of ASR systems for building domain models from call center telephonic conversations. The models comprise a taxonomy, common questions, and standard resolution procedures extracted from telephonic voice transcripts. Mishne et.al. (Mishne et al. 2005) introduced a method to identify the customer's problem. However, they used manually transcribed calls that did not have noise. The transcripts that they used contained speaker and time metadata. Also, they pre-computed the importance values for words offline and then found fragment-level importance values. In contrast, our framework incrementally gets to the focused problem description. This is all done online. Speech summarization techniques outlined in (Maskey 2008) fall in this category and work well when a few hundred annotation transcripts suffice. However, this approach does not work when handling the long tail of support questions. Also, assuming large amounts of labeled data is infeasible. Cailliau et. al. designed a method to select and rank critical conversations using linguistic text mining to detect sentiment markers on French automatic speech transcripts (Cailliau and Cavet 2013). More recently, Ghanny et al extracted named entities from French speech data (Ghannay et al. 2018). Sudarshan and Kumar perform emotion detection, banned and greeting words detection, and competitor name detection on manually transcribed voice transcripts to measure the performance of support agents and to get useful insights for business analytics (Sudarsan and Kumar 2019). Thaler et. al developed a method for capturing and sensing emotional data to detect whether a participant of a debating challenge was arguing for or against her/his conviction, using speech analysis (Thaler, Fauer, and Gewald 2021). These works are very different from what we are proposing which is enriching the service requests with information that might have been missed.

Attribute extractions are harder on ASR transcripts as they lack punctuation, sentence boundaries, capitalization, etc. Among structural features i.e. position of sentences, the speaker turns for extraction (Maskey 2008) are commonly extracted. However, none of the above literature exploits the sub-structure of problem descriptions for extraction tasks.

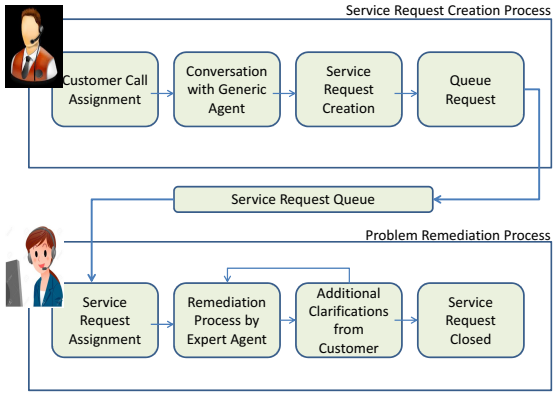

Figure 1: Interplay of Service Request Creation and Problem Remediation Processes

In our work, we also extract important attributes from combined service requests and extracted problem descriptions from STT transcripts.

Text Processing Framework

The text processing framework has three main components.

Pre-processing: We use the BiRNN model proposed by (Pahuja et al. 2017) for punctuation and capitalization prediction by treating them as a correlated multiple sequence labeling problem. Given a speech transcript Q, which is comprised of a sequence of tokens - $Q = [q_1, q_2, .., q_n]$, we predict two labels (punctuation and a capitalization label) corresponding to that word. In the case of punctuation, the labels are Comma, Question Mark and Period. For capitalization, the labels are all-lowercase, all uppercase, mixed-case, sentence-case, and single-letter-word-case. The transcripts from the Intelligence Squared debating television show and from the machine translation task in IWSLT 2012 were used for training the model. The model was trained using standard backpropagation in TensorFlow (Abadi et al. 2016).

Phrase detection: This phase begins with keyword detection and then extracts relevant contextual segments around these keywords. These phrases are then weighted with respect to the technical corpus to get a final list of key phrases.

Keyword Extraction: The objective here is to extract keywords from the speech transcripts such that the keywords describe the component of concern of the user.

We use a Gated Recurrent Unit model to perform domain-specific keyword detection and extraction specifically for the support context (Chung et al. 2015). Formally, we use a GRU model trained for performing the task of keyword extraction from the problem reported by the user. The GRU was trained on user-reported problem descriptions along with the human agent's observations that captured the error code that was used to solve the problem. This was used to classify them into error categories or clusters of problems. Each of these error codes is usually associated with the main physical/logical entities. We revisit the following two equations from the GRU model used in computing the next hidden state recursively from the previous hidden state in the t^{th} iteration.

$$\mathbf{h'}_t = tanh(\mathbf{U}^h \mathbf{v}_t + \mathbf{W}^h(\mathbf{h}_{t-1} \circ \mathbf{r}_t))$$

$$\mathbf{h}_t = (1 - \mathbf{z}_t) \circ \mathbf{h'}_t + \mathbf{z}_t \circ \mathbf{h}_{t-1}$$

In these equations, the vectors $\mathbf{r}_t$ and $\mathbf{z}_t$ form an element-wise product with the previous hidden state. Here the new word vector is important when either of the vectors $\mathbf{r}_t$ or $\mathbf{z}_t$ have small values. This is the property that we leverage for entity extraction. Our methodology chooses the words in the query which are deemed important based on $\mathbf{r}_t$ or $\mathbf{z}_t$ and activities to create a short snippet using those chosen words. These words are the keywords. We define the importance $\gamma_s(w_t)$ of t^{th} word w_t in any sentence s as

$$\gamma_s(w_t) = 1 - \frac{1}{n}\|\mathbf{z}_t\|_1$$

The second term in the above equation is the mean of absolute values in $\mathbf{z}_t$. Thus, $\gamma_s(w)$ would be large for important words. Furthermore, we introduce a threshold τ so that any word w in a sentence s is considered important when $\gamma_s(w) > \tau$. Both τ and μ need to be set as design parameters.

This resulted in the extraction of the following keywords: machine, console, green, light, and blinking.

Phrase Completion Using Linguistic Patterns: The previous step extracted important keywords. It was observed that a significant number of descriptive phrases (i.e. adjectives, verbs) were being missed out. In our example, "not starting" is an important phrase that got completely missed out. This observation motivated us to post-process the textual segment to identify other more complete phrases.

Using Predicate Argument Structure and Slot Grammar (PAS), we extract other textual segments like complements and adjuncts of predicates. These add additional meanings to the predicates. Often, this translates to identifying the verbs and the adjectives mentioned in the segment along with the noun phrases. However, one would have to write a lot of rules to capture complements and adjuncts for all forms of the sentence when using dependency tree-based parsers like Stanford. In contrast, PAS preserves the same structure of the parse tree for different forms of sentences, e.g. "disk is making a loud noise" and "a loud noise is being made by the disk". Hence, the number of rules to be written to extract the words "loud noise" from the parse tree is much less.

The keywords extracted from GRUs, the keywords from the negation dictionary, and other textual segments from PAS are then combined to form a reconstructed textual segment. We reconstruct by starting from a keyword and applying, in order, unique keywords from all the dictionaries, and the textual segments from PAS. In the future, we are experimenting with different natural language generation techniques.

This results in the following non-redundant textual segment: "machine not starting came console seeing green light

saying blinking". Note however that not every textual segment is of interest. For instance, "starting came console" has no meaning on its own. We, therefore, need to determine which phrases of the textual segments are important in the technical corpus and filter out the rest.

Phrase Filtering Using Technical Corpus: We used the following approach to estimate the (technical) importance level of phrases in the textual segment. This is different from the sentence and word level importance studied in (Maskey 2008) but is key for our use case of driving search. In order to create a technical corpus for a domain, problem-solving manuals, technical articles, and historical service request descriptions were taken to compose a text dump of around 30 GB. All tri-grams and bi-grams are extracted from the text dump and individual words are stemmed. They are then stored in a hash map to look up their frequency of occurrence. From the above reconstructed textual segment, we extract bigrams, trigrams, and 1-word skip bigrams. Phrases whose normalized count is above a threshold are considered valid technical phrases. This helps to remove noise (non-technical words) from the text. As the phrases have more meaning if they are ordered by their appearance, we merge the phrases in order. This process results in the production of clean and focused textual segments. The output obtained after this phase is machine not starting console seeing green light blinking".

Attribute Identification The important fragments as identified by the previous component combined with the agents' summaries are now passed through attribute identification. As discussed earlier, the problem description always contains one or more of the attributes of symptom, activity, and intent. We exploit this sub-structure of the voice calls by building a Conditional random field (CRF) model that extracts these attributes. Our intuition for using CRFs for attribute identification comes from our observation of the nature of problem descriptions, wherein the context of the words is very important. We pose the problem of text annotation as a sequence labeling problem as the problem descriptions talk about the problem symptoms, their activities in trying to resolve the problem, and finally their explicit requests. CRFs are graphical models that can capture such dependencies among input observations (Lafferty, McCallum, and Pereira 2001). A CRF model defines a conditional distribution $p(y|x)$ where *y* is the corresponding label sequence and *x* is the observed data sequence, in our case, the set of words. The observation *x* can be dependent on the current hidden label *y*, previous *n* hidden labels, and on any of the other observations in an *n* order CRF. For our case, the input sequence *x* corresponds to a series of words, while the label sequence *y* corresponds to the symptom, activity, intent, or other assigned to the input sequence. The probabilistic model of a label sequence given some sequence of words is mediated in this model through a set of weighted functions f_i:

$$p(y|x) = \frac{exp(\sum_i \sum_t w_i f_i(y_{t-1}, y_t, x, t))}{Z(x)}$$

where w_i are the weights assigned by the learning algorithm, and $Z(x)$ is a normalization factor overall label sequences.

Label	Count	Label	Count
Symptom	1773	Multiple Symptom	467
Activity	121	Multiple Activity	13
Intent	510	Multiple Intent	12

Table 2: Distribution of Symptom, Activity, and Intent

For the CRF training, we used a manually annotated set of support questions in order to generate labeled training data. We utilized the CRF model for extracting the symptoms, activities, and intents. We used the B-I-O encoding, where each word in the query is annotated to one of the following classes (labels): the beginning of a symptom (B-S), inside of a symptom (I-S), the beginning of an activity (B-A), inside of an activity (I-A), the beginning of an intent (B-I), inside of an intent (I-I) or other (O).

To conduct our experiments, we used the linear-chain Stanford CRF implementation (Finkel, Grenager, and Manning 2005). We used the current word, the previous word, the next word, the current word character n-gram ($n \leq 6$), the presence of the word in the left window (size = 4), the presence of the word in the right window (size = 4), the position of the word in the sentence, and current POS tag as features computed using the Stanford NER toolkit. These features were used to train and test CRF.

Evaluation

In this section, we evaluate our framework mainly on three dimensions: (1) Efficacy of its text processing framework to remove noise and extract key "enrichment" information (2) Accuracy of the CRF model on "attribute" extraction (3) Effectiveness of the extracted "enrichment" information to improve the quality of the agent's search.

For evaluation purposes, 240 real STT transcripts were downloaded from the technical support domain for a company. We had both the STT transcripts and the agents' service requests for these audio calls. For attribute extraction using CRF, we used 1972 support queries and asked subject matter experts (SMEs) to annotate them with symptom, intent, and activity labels. 7 SMEs participated in the labeling tasks. All of them were provided the 240 voice transcripts. For attribute identification, they were each provided with 270 support queries for labeling. The overlap of support queries among the annotators was 66%. Once the labeling was completed, those queries where annotators had disagreements were revisited to resolve the differences. During the disagreement resolution process, conflicts were resolved by taking a majority vote for the final label among the 7 annotators. We compute the inter-annotator agreement ratio in terms of the kappa coefficient (Cohen 1960). Due to the presence of sensitive information, we are unable to publicize the data.

Efficacy of text processing framework: To evaluate the text processing framework and its ability to remove noise from the speech transcripts, we used the text similarity metrics of Cosine, Jaccard, and Jenson-Shannon. We compared the syntactic similarity between the extracted textual segment and the agent service request summaries after each of

Attribute	Precision	Recall	F1
Symptom	78.3%	71.20%	74.33%
Activity	70.00%	33.33	44.85%
Intent	93.74%	48.67%	77.41%

Table 3: Attribute Performance Score

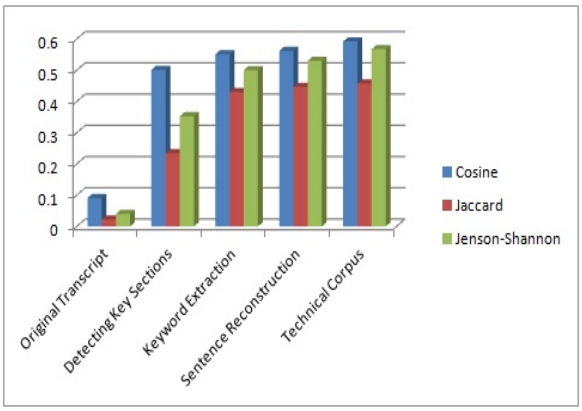

Figure 2: Evaluation Results

Algorithm 1: Diff Extractor

Input: Service request summary, Extracted Textual Segment
Output: Missing Information in the summary

```
for word i, w_i, in extracted textual segment do
    if w_i not in agent's summary then
        for word j, w_j, j = i + 1 in extracted textual segment do
            if w_j not in agent's summary then
                continue
            else
                output words w_i to w_{j-1} from the textual segment
                i = j + 1
                break
            end if
        end for
    end if
end for
```

the text processing phases. Figure 2 shows the average similarity scores for the 240 calls for the different phases starting with the original speech transcript. As shown, the average similarity scores improved at each stage with a maximum gain of $30\% - 40\%$ obtained from the section detection and the keyword extraction. The post-processing steps of using the PAS parser to find additional textual segments and stitching them together using the technical corpus phrases further improved the similarity by (5-10)%.

Next, we quantify how much enrichment our framework is able to achieve using the text processing framework. For this evaluation, we designed a *diff* extractor that compares the service request summaries and the information extracted using our framework and identifies any missing information in the summaries. It also can point out erroneous captures by the agent. The algorithm for obtaining the missing/erroneous information is given in Algorithm 1. The output of this algorithm on a few real summaries is shown in Table 4.

Using Algorithm 1, we compared our framework's extractions against 240 agents' service request summaries. In the absence of any ground truth for the dataset under consideration, we performed a manual evaluation of the extracted textual segments. We report that our framework was able to detect enrichment information that was not present in the summaries with high accuracy in 97 out of 240 of the records i.e. 40% improvement. Row 1 in Table 4 shows that the agent completely missed out on capturing an important entity, the **tape library unit** that our framework could extract. Row 3 shows that the agent added a piece of extra erroneous information, **windows**, not present in the original transcript.

Accuracy of the CRF Model on Attribute Extraction: We next evaluate the attribute extraction component. To create training data for CRF's sequence labeling, we worked with subject matter experts (SMEs) to annotate 1972 support queries with symptom, intent, and activity labels. The distribution of these labels in our training dataset is shown in Table 2. Longer more complex queries often had multiple symptoms, intent, and activity (as shown in Table 2). We trained a linear CRF on the annotated queries and tested the obtained models on 240 problem descriptions (a combined set of important fragments identified by our framework and the agents' summaries). We had subject matter experts in the domain verify the accuracy of extractions with the ground truth. Table 3 gives the performance score of the CRF model A strict scoring was followed where extraction is marked correct only when all the symptoms or activities or intents are identified in longer more complex queries.

Effect of enrichment information on information retrieval: For evaluating the effect of our framework's enrichment extraction, we compared the agent search result (the agent summary and the topmost link that was returned) with a modified search result where the agent summary was enriched with our framework's extracted data. Table 5 gives a few examples of agents' service request summaries, the enriched information extracted using our framework, the top search results using the service request summaries (column 3), and the top result using the enriched information (column 4). The enrichment data was fed into the search using relevance scores along with the keywords.

Relevance scores for the enrichment text were either increased or decreased. When the enrichment information reported a missing piece of context (for example, multiple symptoms) or an explicit intent, the relevance score is increased. For example, row 4 shows that the "Errors console tape library unit" is missing context. These keywords had a boosted relevance score in the new query. When the enrichment information represents an activity that the user has already performed, then the score is decreased. For example, row 1 reports that the customer had "attempted cleaning" the drive, but this had not been captured in the service

Transcript	AgentSummary	FrameworkInfo	ErrorType
... I'm getting some errors on the web console for the tape library unit that I have ...	Getting errors on the WE console	Errors console tape library unit	Omission Error
... in the error state system disk drive error there's a suspected jet five error correct and ...	Issue is not fix	state system disk drive error	Commission and Omission Error
... need to replace give this a disk failure correct ...	There is a failed drive the windows time for the replacement	Really need replace disk failure	Commission Error

Table 4: Examples of Missing/Erroneous Information in Agent's summary

ServiceReq	Framework	ServiceReqTop	FrameworkTop
Symptom: Dirty Tape Drive	**Activity**: Attempted Cleaning	Cleaning the tape drive	Removing a tape from the tape drive
Symptom: Tape Drive is defined and not responding	**Activity**: Tried hard reset	FI00141	Cleaning the tape drive
Symptom: Tape stuck in the tape drive	**Activity**: Tried diag cmd holding eject button	Eject button functions on the tape drive	Removing a tape from the tape drive
Symptom: Getting errors on the WE console	**Symptom**: Errors console tape library unit	Application deployment problems	Tape library reference codes
Symptom: There is a failed drive the windows time for the replacement	**Intent**: Need replace disk failure	Starting disk service	Replace a disk drive
Symptom: Tape drive will not eject the tape	**Activity**: Tried reset	Reset the drive	Removing a tape from the tape drive

Table 5: Examples of Search Accuracy Improvement

request. To give lower preference to cleaning links returned from the search, we decrease the relevance scores of these keywords. Hence, the "attempted cleaning" keywords were given a negatively boosted relevance. Note that without this modified relevance score, the search results continue to bring up the resolution document of "cleaning tape drive" as the topmost link. The enriched search results were evaluated by SMEs. In the SME evaluation, an overall accuracy improvement of 55% in the problem resolution process was achieved by using the enriched information extracted from the speech transcripts using our framework.

To Be Deployed

The proposed framework is being deployed in a test system to assist the agents in the effective and faster remediation of various problems. The objective is to identify and extract enriched information from STT transcripts. The integration of the proposed framework with the software support system enables agents to evaluate the retrieved results and provide feedback in real time. The ongoing feedback turned out to be highly useful, especially in the present conditions where the lack of adequate ground truth inhibits proper performance evaluation. In addition, feedback obtained from agents can also be used to improve the system through various strategies, such as continuous learning, and reinforcement learning. The framework is invoked whenever a user calls to report a problem. The results are displayed in a test system where agents can provide feedback on the retrieved results, and their feedback will be used for further enhancement of the modules.

Conclusion

In this paper, we describe a framework to identify and extract enriched information from STT transcripts to improve the efficiency of the problem remediation process. Our framework enhances the ability to automate a large part of this work, and, in turn, assist agents in accurately capturing customers problems. The promising results have received great feedback from the agents who believe that this can also lead to their further training in capturing customer complaints more effectively. We are in the process of collecting more transcripts with their corresponding service requests. The scope for future work is: (1) With the baseline CRF model in place, we have started work on leveraging neural networks for Attribute Identification. (2) We will focus on using different natural language generation and summarization techniques for grammatically correct sentence reconstruction. (3) We also plan to explore how well this framework supports other service areas and how to make it easier to expand to other domains with reduced cold start issues. (4) Relatively small dataset for evaluation; the solution may be overfitted to one domain.

References

Abadi, M.; Barham, P.; Chen, J.; Chen, Z.; Davis, A.; Dean, J.; Devin, M.; Ghemawat, S.; Irving, G.; Isard, M.; Kudlur, M.; Levenberg, J.; Monga, R.; Moore, S.; Murray, D. G.; Steiner, B.; Tucker, P.; Vasudevan, V.; Warden, P.; Wicke, M.; Yu, Y.; and Zheng, X. 2016. TensorFlow: A System for Large-Scale Machine Learning. In *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, OSDI'16, 265283. USA: USENIX Association. ISBN 9781931971331.

Agarwal, S.; Sindhgatta, R.; and Sengupta, B. 2012. SmartDispatch: Enabling efficient ticket dispatch in an IT service environment. In *Knowledge Discovery and Data Mining*.

Cailliau, F.; and Cavet, A. 2013. Mining Automatic Speech Transcripts for the Retrieval of Problematic Calls. In *Proceedings of the 14th International Conference on Computational Linguistics and Intelligent Text Processing - Volume 2*, CICLing'13, 8395. Berlin, Heidelberg: Springer-Verlag. ISBN 9783642372551.

Christensen, H.; Kolluru, B.; Gotoh, Y.; and Renals, S. 2004. From Text Summarisation to Style-Specific Summarisation for Broadcast News. volume 2997, 223–237. ISBN 978-3-540-21382-6.

Chung, J.; Gulcehre, C.; Cho, K.; and Bengio, Y. 2015. Gated Feedback Recurrent Neural Networks. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, 20672075. JMLR.org.

Cohen, J. 1960. A Coefficient of Agreement for Nominal Scales. *Educational and Psychological Measurement*, 20(1): 37–46.

Finkel, J. R.; Grenager, T.; and Manning, C. 2005. Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, 363–370. Ann Arbor, Michigan: Association for Computational Linguistics.

Ghannay, S.; Caubrire, A.; Estve, Y.; Camelin, N.; Simonnet, E.; Laurent, A.; and Morin, E. 2018. End-To-End Named Entity And Semantic Concept Extraction From Speech. In *2018 IEEE Spoken Language Technology Workshop (SLT)*, 692–699.

Hori, C.; and Furui, S. 2004. Speech Summarization: An Approach through Word Extraction and a Method for Evaluation. *IEICE Transactions*, 87-D: 15–25.

Lafferty, J. D.; McCallum, A.; and Pereira, F. C. N. 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, 282289. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc. ISBN 1558607781.

Maskey, S. R. 2008. *Automatic Broadcast News Speech Summarization*. Ph.D. thesis, USA. AAI3333404.

Mishne, G.; Carmel, D.; Hoory, R.; Roytman, A.; and Soffer, A. 2005. Automatic analysis of call-center conversations. In *Proceedings of the 14th ACM International Conference on Information and Knowledge Management*, CIKM '05, 453–459. New York, NY, USA: Association for Computing Machinery. ISBN 1595931406.

Mohamed, A.-r.; Dahl, G.; and Hinton, G. 2012. Acoustic Modeling Using Deep Belief Networks. *Audio, Speech, and Language Processing, IEEE Transactions on*, 20: 14 – 22.

Padmanabhan, M.; Saon, G.; Huang, J.; Kingsbury, B.; and Mangu, L. 2002. Automatic speech recognition performance on a voicemail transcription task. *IEEE Transactions on Speech and Audio Processing*, 10(7): 433–442.

Pahuja, V.; Laha, A.; Mirkin, S.; Raykar, V. C.; Kotlerman, L.; and Lev, G. 2017. Joint Learning of Correlated Sequence Labeling Tasks Using Bidirectional Recurrent Neural Networks. In *INTERSPEECH*.

Roy, S.; and Subramaniam, L. V. 2006. Automatic Generation of Domain Models for Call-Centers from Noisy Transcriptions. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, 737–744. Sydney, Australia: Association for Computational Linguistics.

Sudarsan, V.; and Kumar, G. 2019. VOICE CALL ANALYTICS USING NATURAL LANGUAGE PROCESSING. In *ICTACT JOURNAL ON SOFT COMPUTING*.

Thaler, F.; Fauer, S.; and Gewald, H. 2021. Using NLP to analyze whether customer statements comply with their inner belief. In *arXiv preprint arXiv:2107.11175*.

Zhou, W.; Li, T.; Shwartz, L.; and Grabarnik, G. Y. 2015. Recommending ticket resolution using feature adaptation. In *2015 11th International Conference on Network and Service Management (CNSM)*, 15–21.