Mining conversational text for procedures with applications in contact centers

Deepak Padmanabhan · Krishna Kummamuru

Received: 21 March 2007 / Revised: 22 June 2007 / Accepted: 7 August 2007
© Springer-Verlag 2007

Abstract Many organizations provide dialog-based support through contact centers to sell their products, handle customer issues, and address product-and service-related issues. This is usually provided through voice calls—of late, web-chat based support is gaining prominence. In this paper, we consider any conversational text derived from web-chat systems, voice recognition systems etc., and propose a method to identify procedures that are embedded in the text. We discuss here how to use the identified procedures in knowledge authoring and agent prompting. In our experiments, we evaluate the utility of the proposed method for agent prompting. We first cluster the call transcripts to find groups of conversations that deal with a single topic. Then, we find possible procedure-steps within each topic-cluster by clustering the sentences within each of the calls in the topic-cluster. We propose a measure for differentiating between clusters that are procedure-steps and those that are topical sentence collections. Once we identify procedure-steps, we represent the calls as sequences of procedure-steps and perform mining to extract distinct and long frequent sequences which represent the procedures that are typically followed in calls. We show that the extracted procedures are comprehensive enough. We outline an approach for retrieving relevant procedures for a partially completed call and illustrate the utility of distinct collections of sequences in the real-world scenario of agent prompting using the retrieval mechanism.

Keywords Conversation mining · Text mining · Clustering · K-Means · AprioriAll

1 Introduction

Contact centers (or, *Call Centers*) is a general term for help desks and customer service centers. They typically provide dialog-based support to the customers in the form of voice calls or web chat. Call centers have become widespread, especially because it allows companies to be in direct contact with the customers. A typical call center agent handles up to hundreds of calls a day depending on the complexities of the issues addressed. Advances in speech recognition systems and their widespread deployment in call centers for monitoring of agents, large volumes of dialog transcripts are being generated from the call centers. It may be noted that contact centers that provide support through web chat automatically generate dialog transcripts. This is in addition to the more traditional form of data such as email, work summaries by the agent, and customer satisfaction surveys. In the sequel, we use the phrase *conversational* (or simply, call) transcripts to include transcripts of any dialog (voice or Instant Messaging based chats) which is directed towards a specific objective such as resolution of the customer issue in the case of call centers. The call transcripts contain wealth of information, if it can be extracted. In this paper, we attempt to process the transcripts in an unsupervised way to extract various procedures followed by call center agents.

Extraction of procedures from conversational transcripts has quite a few applications. Call center agents typically use manually authored documents, or knowledge bases, to answer the calls. Because they are manually authored, these knowledge bases cannot quickly adapt to the various kinds of

This work is an extension of [16].

D. Padmanabhan (✉) · K. Kummamuru
IBM India Research Lab, EGL Business Park,
Domlur Inner Ring Road, Bangalore 560071, India
e-mail: deepak.s.p@in.ibm.com
URL: <http://www.research.ibm.com/irl>

K. Kummamuru
e-mail: kkumnamu@in.ibm.com

new problems, and newer ways of solving older problems that arise during the course of time. For example, an agent may be particularly efficient in solving an issue in a fashion that has not been documented. This is a particular case of knowledge gained from experience in dealing with various issues. It may be noted that call transcripts are the only sources where traces of such knowledge can be found. This work is thus the first step in identifying such knowledge which would automate the knowledge discovery task and partially automate the task of augmenting contact center knowledge bases. We illustrate a few more examples of applications of procedure extraction in the next section.

The transcripts that are obtained from call centers are typically noisy. Call transcripts generated by Automatic Speech Recognition (ASR) systems typically have a *Word Error Rate* of around 35–40% and hence are very noisy. IM transcripts also are very noisy because of unnatural synonyms and abbreviations that participants use in a usual IM conversation, coupled with the possibility of widely differing language styles of the participants. The extraction of procedures is also related to the popular topic of process mining [1]. In this paper, we attempt to bridge these areas by addressing the problem of procedure extraction from noisy text.

Collections of Call transcripts are typically very diverse in the kind of problems that they address. We initially cluster them to arrive at topical collections, which are collections of calls addressing a specific issue. Topical collections, being homogeneous, are better suited for our approach because the wide differences between calls dealing with problems of very diverse nature are hidden from the rest of the processes as they work with homogeneous collections. Each such topical collection is further split into two subsets, that of agent sentences and customer sentences. These sets are clustered separately to build clusters of sentences, each of which represent a specific sub-step in the process of solving the issue. These sub-steps may be at a very fine granularity such as “*right click and choose copy*”, or may abstract a series of real-world actions, as in the case of steps like “*reboot your machine to safe mode*”.¹ We quantify the sub-step nature of such clusters of sentences by defining a measure for the same and refining the clustering until we get good quality clusters. Once these clusters are obtained, calls can be represented as sequences of such clusters. Such sequences are subjected to frequent sequence mining to discover frequent procedural collections for a topical collection. These frequent sequences are clustered to find clusters of distinct sequences. In order to use the procedures derived from transcripts in agent prompting, we define a retrieval mechanism which outputs relevant procedures for a partially completed call. We illustrate the utility

of retrieving procedures with an example in a real-world scenario of agent prompting in call centers.

Organization of the Paper: In Sect. 2, we describe in more detail the motivation behind the extraction of sequences of procedural text sequences with some sample applications. Our approach is summarized in Sect. 3. Section 4 outlines a possible application in an agent prompting scenario. Details of the experiments performed comprised in Sect. 5. Section 6 summarizes our contributions and possible future work.

2 Motivation and related work

A typical call in a customer service center consists of various steps such as “*greeting*”, “*getting details from the customer*”, “*resolving the issue*”, “*verification*”, and “*sign-off*”. Within these broad steps, there are some typical exchanges of information which would be characteristic of the topic or the procedure employed in the call. For instance, “*checking whether the wireless connection is working*” would be a sub-step which characterizes the topic of the call, which in this case is a wireless connection problem. “*Right-clicking My Computer*” and “*selecting properties*”, on the other hand, is an example of a sub-step which conveys information about the procedure used to resolve an issue. We call such information exchanges that stand for sub-steps in a call as *sub-procedure text segments* (SPTS). It may be noted that calls may contain non-information portions like *yah, I see, okay* etc. In this paper, we extract frequent and distinct collections of SPTS sequences from call corpus. The collection of frequent and distinct SPTS sequences, we conjecture, would represent various procedures followed in the calls and hence they are referred to as procedure collections hereafter. Here, we further motivate the present work by briefly describing various applications of procedure collections. The exemplary applications of SPTS and procedure collections are:

- *Using labeled information:* There may be different sequences of steps which lead to successful resolution of a problem, which may lead to varying degrees of satisfaction to the customer. The presence of labeled information such as *Customer Satisfaction Data* (which is usually logged on a per call basis and contains information about the extent to which the customer is satisfied by the way the agent answered the call) would enable us to identify the most satisfying procedure for resolution of an issue.
- *Identifying inefficiencies:* Representing calls as sequences of SPTS clusters allows identification of redundant loops in calls (revisits of a cluster in a call) which could be avoided. Such information can be used to aid the underperforming agent (an agent who takes a longer time to resolve an issue than usual) to understand which among

¹ In this paper, we use IT support-related examples to illustrate some of the concepts. However, the concepts are applicable to general customer service center calls.

the steps or loops in his usual sequence could be avoided to improve efficiency.

- *Aiding faster problem resolution*: One can identify steps that lead to easy and successful completion of the calls, by analyzing the average proximity of an SPTS to the end of calls, across various calls that have a representation of the cluster. Automatic suggestions based on these could aid the agent to jump to an SPTS that enables him to resolve the issue faster.
- *Agent prompting*: The usual method of reusing call collections in call centers is to enable the agent to query a collection of calls. This relies on the agent to express his queries well, and does not use such information as the context of the query, such as how far he has progressed in the call, the topic being dealt with etc. We may be able to use such information to prompt the agent with representative procedures he can use to solve the issue at hand.
- *Identifying agent experience/innovations*: Manually authored knowledge bases used by the agents cannot quickly adapt to the various kinds of new problems, and newer ways of solving older problems that arise during the course of time. Call transcripts are the only sources where traces of newer efficient ways of solving problems can be found. Automatic discovery of procedure collections can partially automate the process of knowledge base augmentation.

Clustering of call center dialogs has been employed to learn about similar dialog traces [2, 3]. Automatically assigning quality scores to calls in contact centers [4], mining call transcripts for trend analysis [5] and call-flow based analysis of call center transcripts [6] are interesting research topics in the contact center analysis. Segmentation of conversation transcripts has been attempted in the past [7] for information retrieval and summarization [8]. Noisy conversational text has been looked at for adding sentence boundaries [9]. The area of noisy text analytics has been receiving increased attention of late [10].

3 Mining conversational text for procedures

3.1 General philosophy of the algorithm

A procedure refers to a particular way of handling an issue in a directed conversation. Calls can be considered as sequences of information exchanges between the caller and the responder. Hence, our approach involves a two-level analysis—the first at the level of unit of information exchange (procedure sub-steps), and the second at the level of sequences of such exchanges. The first level analysis is best performed on a per topic basis, as that would enable segregation of similar sub-steps from widely differing calls addressing diverse

issues (refer Sect. 3.3.1). Thus, we first perform a clustering of the call corpus to obtain groups of conversations on a single topic. For each such topical cluster of calls, we collect the set of agent and customer sentences and cluster them adaptively until the clusters of sentences generated represent sub-steps and not sub-topics of conversations. Once SPTS clusters are obtained, each call can intuitively be represented as a sequence of SPTS clusters. Sequence mining [11] enables identification of frequent sequences across calls within a topical cluster. Further, leader clustering [12] is applied on these sequences to arrive at distinct collections of procedures per topical cluster. Further, we evaluate the utility of these procedure collections in an agent prompting scenario and show its effectiveness over traditional techniques such as information retrieval on the same call corpus. The multi-phase process of obtaining a set of procedures from call collections can be summarized as in Fig. 1.

3.2 Finding topical clusters of calls

Call centers typically handle very diverse issues. The internal customer support center catering to the needs of the employees was found to handle issues of varying complexity and diversity such as those ranging from using a web application to replication of emails in a Lotus Notes database. Clustering of sentences from such a widely varying collection would be error-prone. The sentence “*can you try reconnecting now*” may refer to manually plugging in an Ethernet cable in a call dealing with Ethernet connectivity, whereas it could refer to re-launch of an application in a Web Application-related issue. To abstract away such topical differences from the rest of the algorithm, we cluster the entire call corpus into topical collections using the K-Means algorithm (KMA) [13]. For this clustering, we consider each call as a document containing a concatenation of all the sentences in the call and represent the same as a vector of term frequencies. We set K to the number of different applications and issues that the concerned call center handles. Note that, it is good enough to make K equal to an approximate number rather than the exact number of applications/issues.

3.3 Obtaining sub-procedure text segment (SPTS) clusters

3.3.1 SPTS clusters

Let C be the collection of calls $\{C_1, C_2, \dots, C_N\}$. Each call C_i is represented by a sequence of turns (or sentences) $\{v_1(C_i), \dots, v_{|C|}(C_i)\}$ where $|C|$ is the number of turns in the call. Each turn is associated with the speaker of that turn, which is from the set $\{\text{“Caller”}, \text{“Responder”}\}$. Let $Speaker(v_i(C_j))$ be a function which returns the speaker of the i th turn in call C_j . Let the length of the call C_i be n_i , i.e., $|C_i| = n_i$ for $i = 1, \dots, N$. Let $T_1, \dots, T_k$ be a partition

Fig. 1 Overview of the approach

of C into K topic clusters. Let,

$$G_i = \bigcup_{\forall c \in T_i, \forall l, \text{Speaker}(v_l(c)) = \text{"Caller"}} v_l(c)$$

be the set of sentences spoken by the caller in calls in T_i , and

$$H_i = \bigcup_{\forall c \in T_i, \forall l, \text{Speaker}(v_l(c)) = \text{"Responder"}} v_l(c)$$

be the set of sentences spoken by those who receive the calls in T_i . G_i s and H_i s are clustered separately to obtain SPTS clusters. We use the simple K-Means algorithm (KMA) to cluster the G_i s and H_i s. The number of clusters in KMA is determined by optimizing a quality measure called SPTS-Cluster-Entropy (SCE) Measure. Given a set of SPTSs and a call, the latter can be represented by a sequence of SPTSs with as many elements in the sequence as there are sentences in the call, and the i th element of the sequence being that SPTS to which the i th sentence in the call bears a maximum similarity with. SCE measure is defined in terms of the scatter of calls in the topical call collection across the SPTSs for that collection. We describe the SCE Measure in more detail in the next few paragraphs.

Intuitively, a given clustering of sentences (collection of SPTS) S is good if many calls in the topical collection are scattered across many SPTSs. Thus, we define the SCE measure as a length-weighted average of the scatter of each call among the SPTSs. We first define a measure of scatter with respect to each call in the corpus.

Definition Normalized Entropy of a call C_j with respect to a collection of SPTSs S is defined as,

$$NE_S(C_j) = \frac{-\sum_i d_i * \log(d_i)}{\log(n)} \quad (1)$$

where d_i is the fraction of the C_j in SPTS S_i of S and n is the length of the call C_j .

It may be noted that NE would assume a value between 0 and 1 since $\log(|C_j|)$ is the maximum value that the entropy of the distribution can assume.

Example Let the call C_1 be represented by the sequence $(S_2, S_1, S_5, S_6, S_4)$ and the call C_2 be $(S_3, S_5, S_5, S_3, S_5)$. As is obvious from the representation, C_1 is more scattered than C_2 . The entropy of d_i , viz., $E_S(C) = -\sum_i d_i * \log(d_i)$ captures this scatter as $E_S(C_1) = 0.6989$ and $E_S(C_2) = 0.29$.

However, the entropy measure works well to compare calls of the same cardinality. Consider the case where C_3 is (S_1, S_2) and C_4 be $(S_1, S_1, S_1, S_1, S_2, S_2, S_2, S_2)$ which have the same entropy. Intuitively, C_3 should have a higher score since it is scattered across as many clusters as it can be. The normalization captures this notion as the NE value of C_3 would be 1.0, whereas C_4 would have an NE value of 0.333.

Definition Let $C = \{C_1, \dots, C_N\}$ be the collection of calls and $S = \{S_1, \dots, S_M\}$ be the set of SPTS for C . Then each C_i is represented by a sequence of S_j s. Then, the SCE measure of S with respect to the collection of calls, C is defined as,

$$SCE_C(S) = \frac{\sum_{i=1}^N n_i * NE_S(C_i)}{\sum_{i=1}^N n_i} \quad (2)$$

That is, SCE is the cardinality weighted average of the NE values in the corpus. Some important properties of the SCE measure include:

- SCE increases with the number of clusters because, there are more clusters for a given call to get scattered into.
- For a given number of clusters and an approximately equal number of sentences in the collection, SCE decreases with the increase in average call length. This is due to the increased probability of two sentences in a call getting mapped to the same SPTS which increase in length of the call.

It may be noted that, even though SCE is a corpus-based measure, the SPTS clusters are obtained using the similarity which is computed purely based on individual conversations. The SCE Measure is important because a better (higher) value for it implies that the SPTS clusters generated are closer to describing sub-steps in the call flow and vice versa.

The SCE value is parameterized by both the clustering and the corpus. Hence, one could compare call corpora by comparing their SCE values on the same clustering algorithm and different clustering algorithms by their SCE values on the same corpus. A call collection where the distinct sub-steps are more distinct from each other tends to give better SCE values over those where the sub-steps are similar, which is usually the desirable case (as we explain in a later section). However, the characteristics of the SCE value outlined above leads to some important requirements for comparing corpora or based on SCE values. The corpora to be compared should have comparable values for the following ratios:

- Sentences per SPTS: Average number of sentences per SPTS cluster
- Call Length Ratio: $\frac{\text{Average_Call_Length}}{\text{Number_of_SPTS_Clusters}}$

3.3.2 Adaptively obtaining better SPTS clusters

The collections of agent and customer sentences from each topical collection of calls C are clustered using KMA to obtain SPTS clusters S . We adaptively change the KMA clustering to arrive at a better (numerically higher) $SCE_C(S)$ measure. As observed in the previous section, SCE improves with the number of SPTSs in S and hence, a clustering which puts each sentence in its own cluster gives the best value for SCE. Our goal in obtaining SPTS clusters is to be robust to slight variations in the style of expressing the same sub-step across calls and agents and thus to obtain clusters that represent sub-steps at a higher level of generalization so that each sentence is an instance of the sub-step (SPTS cluster) that it is part of. This leads to the requirement of having sub-steps (SPTS Clusters) that

1. generalize well (have more sentences within) and,
2. have a good scatter of calls across sub-steps (SCE measure).

Reducing k in KMA optimizes for the first requirement, whereas increasing it is favorable for the second requirement. Good generalization and generation of sub-step level clusters is achieved if the clusters generated are coherent, which is the criterion (squared error) that KMA tries to optimize for. The combined requirement translates to finding a clustering into k clusters (minimizing the squared error) where k is neither too low, nor too high. In terms of optimizing k , the first requirement tries to bring down the value of k , whereas the second requirement tries to optimize the SCE Measure. As combining these two measures to form a single combined function (and using traditional search algorithms to optimize that function) is not very straightforward, we find the least value of k where the returns (in terms of increase in SCE Measure) per unit increase in k start diminishing. We accomplish this by adaptively varying the two parameters of KMA, namely the random seed and the number of clusters to be generated by iterating over the following sequence of steps starting with small value for k :

- Vary the random seed t times for the current value of $K = k$ and for $K = k + 1$ and pick the clustering with the best SCE value among them for each K
- If the SCE value for $K = k + 1$ is not better than that for $K = k$ by at least $p\%$, output the clustering with $K = k$. Else, set $k = k + 1$ and continue.

Setting p to a high value tends to terminate the algorithm early and may merge sub-steps, whereas setting p to a low value leads to more exploration and may lead to splitting up of sub-steps.

3.4 Mining calls to get frequent, distinct and long sequences

Once we represent calls as sequences of SPTS clusters, we mine the collections of sequences for frequent sequential patterns across calls. Mining sequential patterns from data [11] is a well studied problem. We use the AprioriAll algorithm to mine for frequent sequential patterns from the call collection. It may be noted that a subsequence of a sequential pattern would have at least as much support² as the latter. This leads to collections of sequential patterns which include patterns of the following undesirable kinds:

- *Very short sequences*: Short sequences tend to have higher support than longer ones

² Support of a subsequence refers to the total number of calls having the subsequence.

- *Redundant sequences*: For each frequent sequence, all its subsequences would also be at least as frequent as the former

It may be appreciated that what we are interested in are collections of *long, distinct and comprehensive sequences*. Long sequences capture most of the procedure which contain them. The distinctness criterion prunes the sequence collection while preserving most of the information. We ensure that we get long and distinct sequences from the set of frequent sequences generated by the AprioriAll algorithm by the following sequence of steps:

- Remove all sequences of length less than *min* from consideration
- Remove all sequences which are subsequences of longer sequences in the collection from consideration
- Use KMA to cluster the collection considering each sequence as a bag of the component SPTS clusters
- Collect the longest *r* sequences from each cluster generated by KMA and form the new collection of sequences

The first step ensures that redundant sequences are eliminated. The clustering step ensures that we give adequate representation for each distinct set of sequences in the collection. The frequent sequence mining, coupled with these post-processing steps, ensures that we get a collection of *frequent, distinct and long* sequences. We hypothesize that this collection of sequences represents procedures in the call collection. We will show that these sequences concisely represent meaningful procedures and that they are comprehensive in a later section.

4 Using procedure collections for agent prompting

Agents in a contact center often query manually authored knowledge bases to find information on the issues related to the received calls. As mentioned in Sect. 2, the ability to retrieve relevant information depends on (1) the agents' ability to express their queries effectively, and (2) the quality of the knowledge base. Since knowledge base is manually authored, it is hard to quickly include in it the solutions to new problems or newer ways to solve old problems. In this paper, we propose a way to extract knowledge as a collection of procedures from historical call transcripts and append it to the existing knowledge base. In this section, a way is proposed to pro-actively prompt the agent at call-time by displaying relevant procedures that are suitable for the current call.

Given a partial call transcript, we convert it into a sequence of SPTS clusters, and use it to find relevant procedures which

would then be displayed to the agent. Extracted procedures may be presented to the agent as sequences of sets of keywords which best describe each step in the procedure. We describe these steps in the following sub-sections.

There are multiple advantages of agent prompting. Firstly, procedures, being concise generalizations of multiple similar calls, are much easier to assimilate than multiple whole call transcripts. Secondly, procedures are privacy preserving generalizations as it is hard to infer as to which agents or customers were involved in the calls that a procedure generalizes, given that characteristic styles of a person get filtered out in the procedure extraction process. Lastly, such an automatic prompting process no longer requires that the agent be able to summarize the call so far to an effective query phrase to query the knowledge base.

4.1 Representing a partial call as a sequence of SPTS clusters

Consider a call C_i , which has been transcribed by an ASR system until time t . We represent this partial transcript as $C_i^t = (Sent_1, Sent_2, \dots, Sent_{n_t})$ where n_t represents the length of the call transcript of Call C_i until time t and $Sent_i$ is the transcript of the i th turn in the call which could be either spoken by the *Agent* or the *Customer*. We represent C_i by a sequence of SPTSs $(S_1, S_2, \dots, S_{n_t})$ as follows:

$$S_k = \arg \min_{S_j} Dist(centroid(S_j), Sent_k),$$

where S_k is any SPTS cluster of sentences spoken by the person in the same role as the speaker of $Sent_i$, and $Dist$ is any distance function such as the cosine measure. We use the SPTS sequence representation of C_i in the rest of the process.

4.2 Retrieving relevant procedures for a partial call

Please note that, based on the variety of the calls received by a call center, the Interactive Voice Response (IVR) system throws up various options to choose from, to route the call to an appropriate expert. We assume that such routing information is known. In case this information is not known, the whole collection of procedures is considered.

For every partial Call C_i^t , we describe a method of retrieving a subset of procedures $P_{C_i^t}$ from the set of procedures $P = \{P_1, P_2, \dots, P_w\}$ of the relevant topical cluster. Let the average call length for calls in the appropriate topical cluster be l —this can be computed from the set of historical call transcripts. We define a boolean-valued function Relevance (P_j, C_i^t) which determines whether or not P_j is to be included in the set of procedures retrieved for the partial call

transcript C_i^t . We define the function as follows:

$$\text{Relevance}(P_j, C_i^t) = \begin{cases} \text{true}, & \text{if } \text{is SubSequence}\left(P_j^{\frac{m_j * n_t}{l}}, C_i^t\right) = \text{true} \\ \text{false}, & \text{otherwise} \end{cases} \quad (3)$$

where P_j^k denotes the prefix of P_j containing $\text{ceil}(k)$ elements and m_j denotes the length of the procedure, P_j . The method $\text{is SubSequence}(\dots)$ checks whether the sequence given as the first argument is contained in the sequence given as the second argument. We denote the set of relevant procedures from P for a partial call C_i^t as $P_{C_i^t}$.

As the average length of a call is l and the length of the partial call C_i^t is n_t , (n_t/l) is an intuitive estimate of the fraction of the call C completed at time t . We further assume that procedures span the entire length of the calls that they generalize. Thus, m_j being the length of procedure in question for calls of length l , $(m_j * n_t)/l$ is the *estimated minimum length* of the prefix of P_j that would be *relevant* to the partial call C_i^t . We include each procedure $P_i \in P$ in the set of retrieved procedures for C_i^t , if the prefix of P_i relevant to C_i^t , is contained in C_i^t . It may be noted that the occurrence of a k -length prefix of P_i in C_i^t automatically implies the occurrence of any shorter prefix of P_i in C_i^t and hence, it is important to exclude prefixes shorter than the estimated minimum length of the relevant prefix to prevent irrelevant procedures from being retrieved.

4.3 Evaluating utility of procedures in agent prompting

The utility of a procedure-based agent prompting application depends on three factors—relevance, understandability and comprehensiveness of the retrieved procedures. We analyze the understandability and comprehensiveness in Sects. 5.4.1 and 5.4.2, respectively. In this section, we outline a method to evaluate the relevance of the retrieved procedures.

The procedure collection as extracted using the method in Sect. 3 may contain multiple representative procedures per procedure type. This is because we allow the procedure collection to contain r longest procedures per procedure cluster (Ref: Sect. 3.4). A call may employ multiple variations of the same procedure type, and may even employ multiple procedures to solve the same issue. Thus, each call, upon completion, can be mapped to multiple procedures from the procedure collection.

Definition Let a call C_i , upon completion, be represented by the sequence of SPTS clusters $(S_1, S_2, \dots, S_n)$ where n is the length of the call. Let $P = \{P_1, P_2, \dots, P_w\}$ be the set of procedures extracted from a historical call corpus using the methods outlined in Sect. 3. We define P_C , the set of

procedures from P which are employed in the Call C_i as

$$P_{C_i} = \{P_j | (P_j \in P) \wedge \text{is SubSequence}(P_j, C_i) = \text{true}\} \quad (4)$$

At a given time t , using the partial Call C_i^t , a set of procedures $P_{C_i^t}$ can be extracted from the procedure collection using the *Relevance* function defined in Sect. 4.2. For a completed call C_i , we evaluate the relevance of the procedures retrieved at different points of time (before completion) in the call $(t_1, t_2, \dots, t_p)$ by measuring the correspondence between each of the sets $(P_{C_i^{t_1}}, P_{C_i^{t_2}}, \dots, P_{C_i^{t_p}})$ and the known set of relevant procedures for the completed call, P_{C_i} . We use the classical measures of *Precision*, *Recall* & *F-Measure* to evaluate this correspondence. $PREC_{C_i^t}^P$, $REC_{C_i^t}^P$, $F_{C_i^t}^P$, the *Precision*, *Recall* and *F-Measure* for the Partial Call C_i^t , using the procedure collection P are calculated as below:

$$\begin{aligned} PREC_{C_i^t}^P &= \frac{|P_{C_i^t} \cap P_C|}{|P_{C_i^t}|} \\ REC_{C_i^t}^P &= \frac{|P_{C_i^t} \cap P_C|}{|P_C|} \\ F_{C_i^t}^P &= \frac{2 * PREC_{C_i^t}^P * REC_{C_i^t}^P}{PREC_{C_i^t}^P + REC_{C_i^t}^P} \end{aligned}$$

The retrieval mechanism is good if the *Precision*, *Recall* and *F-Measure* assume high values for smaller values of t .

5 Experimental study

In this section, we present an experimental study of the proposed technique along with its utility in an agent prompting application. We start by describing the experimental setup and the dataset with a preliminary analysis of the issues discussed in the dataset. We evaluate the effect of corpus homogeneity on the SCE measure (and thus, the quality of resulting SPTS clusters). We go on to evaluate the extracted procedures for *understandability* and *comprehensiveness*. Having verified that the procedures are of good quality, we evaluate their utility in an agent prompting application. We evaluate the effectiveness of the retrieval mechanism proposed in Sect. 4.3 in guiding the agent to the right procedure quickly. This is followed by an analysis of frequent contiguous SPTS cluster-pairs and a discussion on how they can be made use of in an agent prompting application different from that presented in Sect. 4.

5.1 Experimental setup

We used the CLUTO toolkit [14] implementations of K-Means for our experiments. We set K in K-Means so that

Fig. 2 Cluster issue distribution

we get an average of 10–15 sentences per SPTS cluster. We set p to 15% for adaptively obtaining SPTS clusters, whereas we set min and r in the procedure extraction process to 7 and 3, respectively (Ref. Sect. 3.4). We use the *SLPMiner* [15] toolkit for frequent sequence mining using a minimum support of 4% or 4 calls, whichever is lesser.

5.2 Dataset used

We used the call transcripts obtained using an ASR system from the internal IT helpdesk of a company. The calls are about queries regarding various issues like *Lotus Notes*, *Net Client* etc. The prefixes of sentences in the transcripts are either “*Customer*” or “*Agent*”, depicting the role of the speaker. Calls are one-to-one conversations between an agent and a customer. The data set has about 4,000 calls containing around 68,000 sentences. The ASR system used for generating transcripts has an average Word Error Rate of 25% for Agent sentences and 55% for customer sentences. We present a preliminary analysis of the issues handled in the dataset in the following sub-section.

5.2.1 Cluster issue analysis

Representative words give an idea about the various issues addressed in each of the topical clusters obtained. A topic may be related to an issue with an application, or may be related to an application in itself, if most of the issues related to it have commonalities. We did a manual analysis of 50 topical clusters (obtained by setting $K = 50$ in K-Means clustering of the entire corpus) using only the top five representative words obtained from the *Cluto* clustering with the aim of *understanding the various issues addressed* in the corpus, and in *analyzing the quality of the topical clusters*

obtained. Those representative words returned by *Cluto* are either words that are abundant in the cluster, or words that help distinguish the cluster from the other clusters. From these representative words, we pick words that characterize the issue/application to which the cluster belongs. For certain issues like “*password issues*” and “*installation issues*”, being very generic, the issues across applications were seen to be clustered together. Broadly, the clusters seemed to address a single issue, and more encouragingly, issues were not split across multiple clusters on visual examination. We present an application-wise or issue-wise breakup of the clusters in Fig. 2. It may be noted that there is a high skew in the issue distribution with *Password* (14 clusters) and *Lotus Notes* (9 clusters) forming the biggest chunk of issues addressed. This is indicative of the repetitiveness of tasks that an agent performs in a call center, and supports our argument that more of call center tasks need to be automated.

5.3 Need for finding topical clusters

Here, we state and validate the assumption behind the initial clustering of call corpus to find issue-specific topical clusters. A call corpus such as the one obtained from a call center which handles general technical queries can be expected to be very diverse in the nature of issues handled. The clustering algorithm detects dissimilarities (using the distance function) and builds coherent clusters to minimize dissimilarities. It may be noted that topic-level dissimilarities are larger than sub-step level dissimilarities. For example, the dissimilarities between a sentence from a call addressing a Network issue and another addressing a Lotus Notes issue is greater than the dissimilarities between two sub-steps of a password recovery process. A clustering of agent and customer sentences of a heterogeneous (multi-topic) corpus would mostly lead to

topic-type clusters rather than sub-step clusters. Intuitively, the *larger topic-level dissimilarities have to be eliminated to expose the lesser sub-step level dissimilarities* to the clustering algorithm. Homogeneous corpora, being a collection of calls from the same topic, can be expected to be free from the topic-level dissimilarities, and hence can be expected to give better quality SPTS clusters.

Homogeneity is inversely related to the scatter of a dataset. The scatter can be measured as the average distance of each element from the centroid of the dataset, i.e., $\frac{1}{|D|} \sum_{d \in D} \text{dist}(d, \text{Centroid}_D)$ for a dataset D . This is precisely the measure that the K-Means Algorithm minimizes (for each cluster). Thus, the topical clusters generated by the K-Means algorithm can be expected to be more homogeneous than the entire corpus.

In this subsection, we validate the assumption that homogeneous corpus results in better SPTS clusters. We do this by comparing SCE measures of SPTS clusters obtained from a topical cluster (containing similar calls) and those from a set of random calls from the corpus. The set of random calls contains roughly the same number of calls each having roughly the same number of sentences so that the comparison is meaningful (refer Sect. 3.3.1).

For each topical collection of calls T^j from the entire corpus C , we arrive at a random collection of calls, D^j from C which has the same number of calls as in T^j and roughly the same number of sentences as in T^j . We compare the SCEs of the clustering of T^j and D^j by aggregating the values over different values of j and K using two methods:

- We take the weighted average of SCEs of T^j s, the SCE of each T^j weighted by the number of calls in it and denote it by SCE_T . The same is done across D^j s to obtain SCE_D .
- For each (T^j, D^j) pair, we compare the SCE values and assign the one with a higher value of SCE as the *winner*. We count the number of times that T^j wins and the number of times that D^j wins across all values of j .

Results and observations: The results in Table 1 validate the hypothesis that homogeneous clusters generate better SPTS clusters. The figures on aggregated SCE shows that SPTS clusters from homogeneous collections consistently perform better than their less homogeneous counterparts. The winner-based analysis further validates the hypothesis in that homogeneous collections win over their less homogeneous counterparts 80% of the time, on the average.

Table 1 Homogeneity analysis based on SCE measure

K in K-Means	SCE_T	SCE_D	% of T^j wins	% of D^j wins
50	0.842	0.795	82%	18%
100	0.787	0.731	80%	20%

5.4 Quality of extracted procedures

The extracted procedures can be assumed to be of good quality if they can be well *understood* (as they would be ultimately presented to the agent in an application such as agent prompting) and are *comprehensive* (i.e., span most of the calls that contain them). In the following subsections, we evaluate the understandability and comprehensiveness of extracted procedures.

5.4.1 Understandability of procedures

In this subsection, we evaluate the procedures in terms of their understandability. We analyze the topical clusters and the sequences mined from them by representing each topical cluster and each SPTS cluster by the set of representative keywords from them. Although manual analysis is a tedious process, in the absence of on-field verification, we looked into some random sequences and into calls that contain them, and assessed that they follow a similar procedure of solving the problem. One representative sequence from the *Lotus Notes* cluster is shown below. The cluster itself is represented as a set of topical keywords, and each SPTS cluster in the sequence is again represented as a *<role of speaker: {representative keywords} π manual summary>* tuples. The manual summaries were generated looking at the respective SPTS clusters.

The first representative sequence shown in Table 2 summarizes a distinct way of handling the issue of “*setting up archiving and scheduling automatic archiving*”. The first entry is about the agent instructing to *double click on the database* (the Lotus database), whereas the second entry is about the customer clarifying as to *how he should be setting up archiving*. The entire process of setting up takes some time and the third step is about the agent *asking to be prompted when the archiving is done*. The agent later asks to *restart notes* and answers the customer’s query about *scheduling the archiving* process. The customer further asks the agent to *go ahead and perform the scheduling* on his machine. The completion of the task is verified by checking the inbox for completion of the archiving process. We observed that this procedure is fairly understandable with the sequences of sets of keywords that describe it. We found that most procedures are fairly understandable in the presence of domain knowledge. Techniques to correlate these sequences of sets of keywords with knowledge bases in the contact center would aid much better understandability of these procedures.

5.4.2 Comprehensiveness of procedures

Comprehensiveness can be quantified as the fraction of the length of calls that the procedures span. Our retrieval mechanism assumes that procedures are comprehensive. This

Table 2 Representative sequence

Topical Cluster: Lotus Notes
Keywords: lotus, notes, archive, schedule
agent: double, click, database π Instructions to double click on database
customer: advanced, settings, archive π should I go to advanced settings
agent: desktop, know π let me know when you are back on your desktop
agent: restart, lotus, notes π asking to restart lotus notes
customer: scheduling, archive π query about scheduling archive
customer: ahead, go π go ahead and perform scheduling
agent: inbox, open π go to your inbox
Topical Cluster: Expense Reimbursement
Keywords: expense, reimbursement, application
customer: connect, application, error π complaint about error in connecting to the application
agent: connect, intranet π agent checks about intranet connectivity
agent: java, enterprise π agent checks for the java version and edition
customer: completed, close π customer informs completion of java installation
customer: working, close, thanks π customer informs that he is able to connect and signs off

assumption is the basis of the *relevant prefix length* calculation in Sect. 4.2. In this subsection, we validate this assumption. The *Span* of a procedure $P_j = (S^1, S^2, \dots, S^n)$ that is contained within a call C_i represented as an SPTS Cluster sequence $(S_1, S_2, \dots, S_m)$ is computed as

$$Span_{P_j}^{C_i} = \begin{cases} \left(\frac{\max_k \{k | S_k = S^n\} - \min_k \{k | S_k = S^1\}}{m} \right) & \text{if} \\ isSubSequence(P_j, C_i) = true \\ 0.0 & \text{otherwise} \end{cases}$$

Thus, the span is computed as the fraction of the call C that the procedure P_j covers. The span of a call C_i over a collection of procedures $P = \{P_1, P_2, \dots, P_k\}$ is computed as

$$Span_P^{C_i} = ArgMax_j \{Span_{P_j}^{C_i} | P_j \in P\}$$

To compute the span of a call collection (such as a topical cluster) over a set of procedures, we take the cardinality weighted average of the span of each call over the procedure collection (such as the collection of procedures extracted from the topical cluster) across the calls in the collection.

The results of the span analysis are presented in Table 3. The results show that procedure collections usually span at least 80% of the call. This validates our assumption that *procedure collections tend to span most of the call* that contain them and thus gives us more confidence in the retrieval algorithm (Sect. 4).

Table 3 Span analysis of procedures

Topical cluster description	Span of calls in the topical cluster
Lotus Notes	0.8617
Global Print	0.8144
Lotus Notes Accessibility	0.8770
Serial Number Software	0.8908
Asset Manager	0.8745

5.5 Agent prompting application

In this section, we evaluate the utility of extracted procedures in an agent prompting application. Firstly, we evaluate the effectiveness of the retrieval mechanism presented in Sect. 4.3 to drive the agent towards the correct procedure. Secondly, we demonstrate that pairs of frequently contiguous SPTS clusters are useful in agent prompting based on localized information.

5.5.1 Evaluation of a procedure-based agent prompting application

For each topical cluster, and for each call, we evaluate the *Precision*, *Recall* and *F-Measure* as defined in Sect. 4.3 on completion of 20, 40, 60 and 80% of the call. For each topical cluster, we take the cardinality weighted average of these measures across calls in that cluster.

The statistics for the topical cluster on *Lotus Notes* (Fig. 3) shows that our technique achieves a precision of up to 0.80 on completion of 80% of the call. In Fig. 4, we report the F-Measure values for the topical clusters shown in Fig. 2.

As can be seen from the figures, certain topical clusters achieve an F-Measure of around 0.55 even after seeing just 20% of the call. This shows that our method of retrieving procedures provides a very good accuracy. These results show

Fig. 3 Retrieval statistics for the topical cluster on Lotus Notes

Fig. 4 F-Measure across topical clusters

that the retrieval mechanism outlined in Sect. 4 work well for guiding the agent *quickly* to the desired procedure.

5.5.2 Using contiguous step-pairs for prompting

Sequence analysis using the AprioriAll algorithm yields sequences that may not be contiguous. We define $\langle S_1, S_2 \rangle$ as a contiguous cluster pair with score p (the support score) if there are at least p calls where S_2 immediately follows S_1 . These provide a different kind of information, that of typical question–answer pairs, which can be leveraged in the contact center scenario. As most of these pairs would be of the form $\langle \text{Customer SPTS}, \text{Agent SPTS} \rangle$ or $\langle \text{Agent SPTS}, \text{Customer SPTS} \rangle$, due to calls being alternating sequences of customer and agent sentences, this analysis would aid us in identifying typical customer responses to agent directives/instructions, and typical agent suggestions to customer responses. We present some frequent contiguous SPTS cluster IDs labeled with the topical cluster from which they have been extracted in Table 4.

Table 4 Sample frequent contiguous SPTS pairs

Topical cluster	#Calls	SPTS cluster description
Lotus Notes	15/63	Customer says that he has reached a certain step of the mail archiving process Agent informs him of the next step to be performed
Lotus Notes	22/73	Agent asks the customer to open the local mail file copy Customer says that he is unable to do so
Global Print	10/74	Customer says he is done with the step of installing the printer Agent asks the customer to try to print a test page
Intranet Password	20/97	Agents tells customer that he suspects that the password may have expired, and asks him to sign into the intranet Customer says that he is unable to login

The above contiguous step-pairs have applications in multiple scenarios. An example is that of prompting the agent with local information, which is a different prompting task as compared to the one presented in Sect. 4. This may be used to partially automate the contact center operations as well. An information retrieval query could be performed using the customer sentence, on the set of historical customer sentences, so that the frequent agent replies could be retrieved and presented to the customer as a ranked list of responses. This is particularly useful to partially automate the execution of a multi-step process such as “*installing and testing a printer*” which usually is a well-defined flow of steps wherein the agent usually guides the customer as to which step to perform next. Further, in the partially automated case, the agent can choose from multiple suggested steps based on which step would lead to a favorable reply from the agent; once again, using the contiguous step-pair based analysis.

6 Summary and future work

In this paper, we propose a method to extract procedural information from contact center transcripts. We define SPTS clusters and the SCE measure as a quality measure to obtain better SPTS clusters. We study the effect of corpus homogeneity on the quality of the SPTS clusters generated. We outline an algorithm to extract long, frequent, distinct and comprehensive sequences of SPTS clusters which represent various procedures in the call transcript corpus. We show that these sequences are useful in guiding an agent to the relevant procedure in an agent prompting application. We also show that analysis of frequent contiguous pairs of SPTS clusters provides useful information to the agent.

This paper presents a first step towards extracting knowledge from contact center transcripts. The most common application of historical call transcripts in contact centers is that of *call monitoring*, which is used for agent evaluation. This is usually done by selecting a random sample of

calls and analyzing them manually. In such an analysis, we may well miss many calls where agent uses unacceptable practices because such calls would be very few. We conjecture that summarizing call corpora by procedure collection, and annotating each call by the procedure employed within it would help to analyze the entire corpus of calls rather than a small random sample. Such an annotated corpus can be used for identifying best procedure for a given problem/issue, identifying domain experts and help in placing agents at different sites to improve diversity of expertise. The extracted procedures may be used for automating the process of knowledge base creation for contact center agents. This involves comparing the extracted procedures to textual knowledge bases. Correlating the procedures with *Customer Satisfaction (CSAT)* data may lead to interesting conclusions on what factors contribute to a better *CSAT*. Using the contiguous step-pair analysis to partially automate multi-step processes would free up the agent from doing very repetitive tasks, wherein such time could be used in resolving a different issue with another customer (in a scenario like web-chat based support where the agent handles multiple issues at the same time).

References

1. Agrawal, R., Gunopulos, D., Leymann, F.: Mining process models from workflow logs. In: Proceedings of the 6th International Conference on Extending Database Technology, 469–483 (1998)
2. Bechet, F., Riccardi, G., Hakkani-Tur, D.: Mining spoken dialogue corpora for system evaluation and modeling. In Proceedings of EMNLP, Barcelona, Spain (2004)
3. Roy, S., Subramaniam, L.V.: Automatic generation of domain models for call centers from noisy transcriptions. In: ACL-2006 (2006)
4. Zweig, G., Siohan, O., Saon, G., Ramabhadran, B., Povey, D., Mangu, L., Kingsbury, B.: Automated quality monitoring for call centers using speech and NLP technologies. In: Proceedings of ICASSP (2006)
5. Douglas, S., Agarwal, D., Alonso, T., Bell, R.M., Gilbert, M., Swayne, D.F., Volinsky, C.: Mining Customer Care Dialogs for Daily News. *IEEE Trans. Speech Audio Process* **13**(5), pp. 652–660 (2005)
6. Abella, A., Wright, J., Gorin, A.L.: Dialog trajectory analysis. In: Proceedings of ICASSP (2004)
7. Ries, K.: Segmenting conversations by topic, initiative and style. In: Proceedings of ACM SIGIR 01 Workshop on Information Retrieval Techniques for Speech Applications, New Orleans, LA, USA (2001)
8. Mittendorf, E., Schauble, P.: Document and passage retrieval based on hidden markov models. In: Proceedings of ACM SIGIR, Dublin, Ireland (1994)
9. Nasukawa, L., Diwakar, P., Roy, S., Subramaniam, L.V., Takeuchi, H.: Adding sentence boundaries to conversational speech transcriptions using noisily labelled examples, and 2007 workshop
10. Analytics on noisy unstructured text data. In: Conjunction with IJCAI (2007)
11. Agarwal, R., Srikant, R.: Mining sequential patterns. In: Proceedings of the 11th International Conference on Data Engineering, Taiwan (1995)
12. Jain, A.K., Dubes, R.C.: *Algorithms for Clustering Data*. Prentice-Hall, Englewood Cliffs (1988)
13. MacQueen, J.B.: Some methods for classification and analysis of multivariate observations. 5th Symposium on Maths, Statistics and Probability (1967)
14. Steinbach, M., Karypis, G., Kumar, V.: A comparison of document clustering techniques text mining workshop. KDD 2000. Boston, MA, August 2000
15. Seno, M., Karypis, G.: SLPMiner: An algorithm for finding frequent sequential patterns using length decreasing support constraint. 2nd international conference on data mining (ICDM 2002) (2002)
16. Deepak, P., Kummamuru, K.: Mining conversational text for procedures. IJCAI 2007 Workshop on Analytics for Noisy Unstructured Text Data, AND 2007, Hyderabad, India