

Mining Conversational Text for Procedures

Deepak P **Krishna Kummamuru**
IBM India Research Lab
Bangalore, India
{deepak.s.p, kkumnamu}@in.ibm.com

Abstract

With the advances in speech recognition systems, large volumes of call transcripts are being collected from call centers. Many organizations have started giving customer support through instant messaging tools. The agents who attend to calls in call centers typically use manually authored procedure-documents. Even with the advances in text analytics, there has not been much work in extracting useful information from call transcripts. In this paper, we consider any text derived from conversations like phone calls, web-chat and attempt to discover various procedures that are followed by agents during calls by analyzing the call transcripts in a semi-supervised manner. We also discuss how to use the discovered procedures for agent prompting and knowledge authoring. We first cluster the call transcripts in a given corpus to find the groups of conversations related to a single topic. Then, we analyze the calls within each topic-cluster for procedure-steps by again clustering agent and user transcripts separately. In this paper, we propose an approach for analyzing clusters to determine whether they describe topics or steps in a process. We formalize the differences between the two kinds of clusters and define a measure to differentiate between such clusters. Once, the steps in the calls are found, we find distinct frequent sequences of steps in the calls to discover the procedures.

1 Introduction

Call centers are centralized offices for the purpose of receiving and transmitting a large volume of requests by various communication channels like telephone, emails, and instant messages. A call centre is operated by a company to administer incoming product support or information inquiries from customers. This model is becoming more widespread because it allows companies to be in direct touch with their customers. A typical call center agent handles tens to hundreds of calls per day depending on the complexity of the issues that she addresses. With the advances in speech recognition technologies and their widespread deployment in call centers, huge volumes of data are produced everyday in

the form of call transcripts. This is in addition to the data that is produced as a part of the process such as e-mail, instant messages (IM), reports from the agent, customer satisfaction surveys etc. In this paper, we use the phrase *conversation (or simply call) transcripts* to include transcripts of conversations through various media like audio calls and IM/web-chat. In this paper, we focus on unsupervised analysis of transcripts of conversations at typical call centers to extract various procedures followed by call center agents.

There are quite a few applications that can use the procedures extracted from the conversational transcripts. We illustrate one example here and few more examples in the next section. Call center agents typically use manually authored documents (or knowledge bases) to answer the calls. Because they are manually authored, these knowledge bases can not quickly adapt to the various kinds of new problems that arise during the course of time. Moreover, manual augmentation of such knowledge bases would consume tremendous manpower and may not be exhaustive. The procedures discovered using our techniques can be used to partially automate authoring task.

There are two dimensions to the present work. One is the analysis of noisy text like conversation transcripts and the other is the extraction of procedures. Roy and Subramaniam have addressed the problem of extracting domain models from noise call transcripts in [Roy and Subramaniam, 2006]. The domain models consists of problem categories, typical customer issues and solutions, etc. This is one of the recent works on the topic of analysis of conversation transcripts. The extraction of procedures is related to the popular topic of process mining [Agrawal et. al. 1998]. In this paper, we attempt to bridge these areas.

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 more homogeneous, are better suited for our approach (Ref: Section 4.4). Each such topical collection of calls is further split into two subsets, that of agent sentences and customer sentences. The two subsets are further clustered to build clusters possibly containing sentences representative of

similar procedural steps, each cluster thus describing a sub-procedural step. We define a measure to quantify the quality of a sub-procedural cluster. We use this measure to refine clusters till we find good quality clusters. Once these clusters are obtained, each call in the topical collection can now be represented as a sequence of such clusters. The collection of calls, represented as sequences, are subjected to frequent pattern mining to discover frequent procedural sequences for a topical collection. We find representative distinctive procedural sequences from these frequent sequences using leader clustering algorithms like CAARD [Krishna and Krishnapuram, 2001].

Paper organization: Section 2 lays down the motivation behind the extraction of sequences of procedural text segments with some sample applications. Our approach is summarized in Section 3. The two main components of the proposed approach are measuring the quality of conversation clustering and finding desirable sequences of conversation clusters. We describe them in Section 4 and Section 5 respectively. Details of the experiments and their results comprise Section 6. Section 7 summarizes our contributions and possible future work.

2 Motivation

A typical call consists of various phases. For example, a help-desk call broadly consists of the following three phases: agent introducing herself, seeking information about the problem, and providing solution to the problem. Even within, each of these phases of the call, there are some typical exchanges of information that would be same across all the calls that deal with the same topic. For instance, right-clicking ‘my computer’ and selecting properties, getting the SSN/Employee# of the caller, greeting the caller during the beginning etc are examples of such typical information exchanges. We call such units of information exchange as *sub-procedure text segments* (SPTS). It may be noted that calls may contain non-information portions like “yah”, “ic”, “OK”. SPTS from a corpus give very useful information about the calls in the corpus. In this paper, we attempt to extract SPTS from call transcripts by clustering similar sentences to SPTS clusters. There are quite a few applications that be built using SPTS. Specifically, we analyze the sequences of SPTS clusters that appear in calls of a particular type in order to discover various procedures that are followed in the calls. Here, we further motivate the present work by briefly describing various applications of SPTS. The exemplary applications are:

- One may be able extract characteristic phrases from SPTS. These characteristic phrases could be used for providing recommendations for the agent regarding the possible actions he could take for the kind of problem he is addressing.
- There may be different sequences of steps which lead to successful resolution of a given problem. This is intuitive since there are multiple ways to solve every real-world problem. When different

calls on a problem are tagged with the corresponding customer satisfaction data, analysis of STPS would enable us to identify the most “satisfying”, and hence the desirable way, of solving a problem.

- One can identify steps that lead to easy and successful completion of the calls, by analyzing the average proximity of a STPS to the end of the call, across various calls that have a representation in the STPS cluster. This would enable automatic generation of suggestions (of actions) which would enable faster resolution of the issue in question
- Similarly, one can also identify redundant loops (revisit of a cluster) in the calls which could be avoided. This could aid the under-performing agent (and agent who takes a longer time to resolve an issue than usual) to understand which among the steps that he usually resorts to (to solve the problem) have to be avoided.

3 Mining Conversational Text

In this section, we summarize the proposed approach to mine conversational text. A flow chart of the proposed approach is shown in Figure 1. First, we perform K-Means clustering on the corpus to obtain groups of conversations on a single topic, to extract coherent and meaningful SPTS from the conversation transcripts. For each such topical cluster, we collect the set of agent sentences and customer sentences separately, and cluster them adaptively until good quality SPTS are obtained. One of the main contributions of this paper is in defining a quality measure on SPTS clusters (Section 4.3). In the next section, we define the quality measure. In the experimental section (Section 6), we show that obtaining SPTS clusters from topic-clusters are better

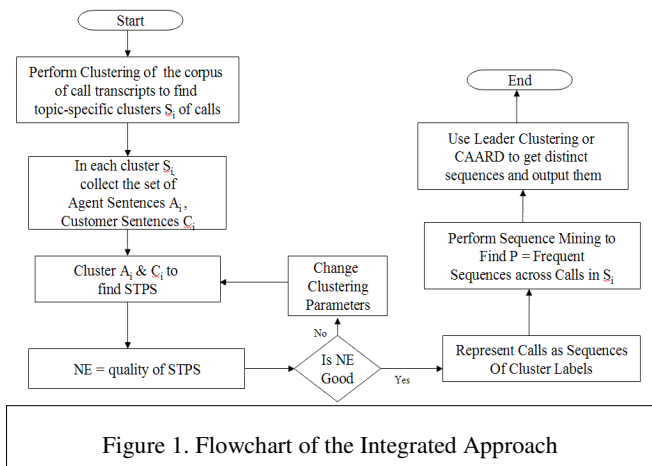


Figure 1. Flowchart of the Integrated Approach

than those obtained from the whole corpus.

Once SPTS clusters are obtained, each call is then represented as a sequence of SPTS clusters. Sequence mining [Agarwal and Srikant, 1995] is done on these sequences of SPTS clusters to arrive at frequent sequences across calls within a topical cluster. From these frequent sequences of

SPTS clusters, distinct sequences are extracted using leader clustering [Jain and Dubes, 1988] or CAARD [Krishna and Krishnapuram, 2001], which would then be output. We explain mining for SPTS cluster sequences in Section 5.

In order to make the output more readable, each SPTS cluster is represented by sentences that characterize the cluster. These characteristic sentences can be obtained using one of text summarization algorithms like the one in [Krishna and Krishnapuram, 2001]. For the sake of quick experimentation in this paper, we use descriptive and discriminative words from the sentences in the cluster to describe the cluster. Descriptive words are those that contribute most to the average similarity among the calls in the cluster. Discriminative words are those that are more prevalent in the calls in the cluster as compared to the rest of the calls.

4 Quality of SPTS Clusters

4.1 Call Entropy: A Measure of Scatter of a Call

In this section, we define a measure that quantifies quality of a SPTS cluster and use this measure to drive the process of clustering agent and customer in conversations.

A typical call consists of a sequence of information exchanges. We assume that there exist some sentences in a conversation that reflect sub-procedures steps. The goodness of the SPTS cluster can be partially judged by the frequency of calls which contain sentences from each SPTS cluster and the number of calls the sentences in each SPTS cluster are scattered into. We quantitatively capture this intuition by defining entropy of a call with respect to the SPTS clusters.

Let, Call1 be represented by the sequence $\langle C_2, C_1, C_5, C_6, C_4 \rangle$ and Call2 $\langle C_3, C_5, C_5, C_3, C_5 \rangle$. As is obvious from the representation, Call1 is more scattered than Call2. We choose to use the well-studied entropy measure to measure the scatter of a call. Let, $E_C(c)$ represent entropy with respect to a given set C of SPTS clusters. That is,

$$E_C(c) = - \sum_i d_i \log(d_i), \quad (1)$$

where, d_i is the fraction of the call c in cluster i . For example, for the clusters numbered C_1 to C_6 , $E(\text{Call1}) = 0.6989$, and $E(\text{Call2}) = 0.29$.

4.2 Normalized Entropy: Adaptation to compare calls across Cardinalities

Entropy works well to compare calls of the same cardinality. Consider the case of Call3 which is $\langle C_1, C_2 \rangle$ and Call4 which is $\langle C_1, C_1, C_1, C_1, C_2, C_2, C_2, C_2 \rangle$, which have the same entropy. Intuitively, Call3 should have a higher score since it is scattered across as many clusters as it can be. Thus, we propose Normalized Entropy (NE) to be a better

measure of scatter of the call. NE of a call c with respect to a set C of clusters is defined as,

$$NE_C(c) = - (\sum_i d_i \log(d_i)) / \log(|c|), \quad (2)$$

where, $|c|$ is the length of the call c . Normalized Entropy would assume a value between 0 and 1 since $\log(|c|)$ is the maximum value that Entropy can assume. Call3 would have an NE value of 1.0 and Call4 would have an NE value of 0.333, which aligns well with our requirements.

4.3 A Quality Measure for a Clustering on a Collection of Calls

Now, we define quality measure of a given set C of clusters using the normalized entropy of all the calls in the corpus R , as the cardinality weighted average of NE values for calls:

$$NE_R(C) = (\sum_{c \in R} NE_C(c) * |c|) / (\sum_{c \in R} |c|) \quad (3)$$

Here are some important properties of the NE measure:

- NE 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 data elements, NE decreases with the increase in average call length. This is due to the increased probability of two steps (in a call) getting mapped to the same cluster with increase in the length of a call.

NE is parameterized by both the clusters and the corpus. Hence, one could compare clustering algorithms by their NE values on various corpora, and different corpora by their NE values when subjected to the same clustering technique. However, the characteristics of the NE function outlined in the previous paragraph lead to two important requirements for NE value based comparison. The clusters to be compared should have roughly same values for the following ratios

- Sentences ratio: Number of sentences to be clustered / Number of clusters
- Call length ratio: Average Call Length / Number of clusters

The reasons are obvious from the characteristics of NE function already discussed.

In the following subsection, we describe various ways of using NE measure to appropriately split the corpus to get good SPTS clusters. Through the rest of the paper, we shall refer to the NE measure of the clustering of a collection of calls into sentence clusters as simply the NE measure of a collection of the call collection.

4.4 Analysis of Corpus Homogeneity using NE measure

A corpus of call collections would obviously contain calls pertaining to diverse issues. A clustering of agent and cus-

customer sentences of such a corpus, would mostly lead to topic-type clusters than SPTS clusters. A homogeneous corpus could be expected to give better SPTS clusters than a more diverse collection. This is based on the assumption that the larger dissimilarities (which are topic-level) have to be eliminated in order to expose the lesser (SPTS) dissimilarities to the clustering algorithm. Hence, we hypothesize that NE of the clusters increases with the homogeneity of the corpus of calls to be clustered. We will verify this hypothesis in the experimental study presented in Section 6.

5 Mining of SPTS Cluster Sequences

5.1 Calls as Cluster Sequences

Let, $\{C_1, \dots, C_n\}$ be the clusters of sentences corresponding to customer and $\{A_1, \dots, A_m\}$ be that to agents. Then, each call is represented by a sequence of those clusters to which sentences in the call belong to. For example, $\langle \text{START}, A_2, C_1, A_4, C_4, \text{END} \rangle$ represent a call where the first sentence in the call, which is in A_2 , was spoken by agent, second sentence, which is in C_1 , is spoken by customer, and so on. We assume, without loss of generality, that the call consists of sentences belonging to As and Cs alternatively. Consecutive sentences by the same person can be combined into one.

5.2 Frequent Sequences

This sub-section describes the technique of extracting useful information from the call sequences by means of frequent sequence mining. Mining sequential patterns from data [Agarwal and Srikant, 1995] is a well studied problem. One of the algorithms can be used to find frequent sequences from the set of call sequences. However, these algorithms result in many redundant sequences. Hence, one can use some summarization techniques like CAARD to find long and distinct sequences. Figure 2 summarizes the method extracting distinct frequent sequences.

Sequence Mining on Call Corpora
<pre> Corpus = Total Collection of Calls; Cluster Corpus by K-Means (K=k) to get Col- lections of Calls C = {C₁, C₂, ..., C_k} ; For (i=1; i<=k; i++){ DSA_i = { p ∃(c ∈ C_i) s.t. p ∈ C_{Agent} } DSC_i = { p ∃(c ∈ C_i) s.t. p ∈ C_{Customer} } Let A_i be the clusters obtained using K-Means on DSA_i with K=max(DSA_i /10, 10) Let C_i be the clusters obtained using K-Means on DSC_i with K=max(DSC_i /10, 10) For (every c ∈ C_i) { Assign a sequence S_c to c S_c = number of sentences in c; for(j=0; j < S_c ; j++){ s = jth sentence of c; if(s ∈ C_{Agent}) Sc[j] = cluster in A_i which contains s; else if(s ∈ C_{Customer}) </pre>

<pre> Sc[j] = cluster in B_i which contains s; } } SP = distinct & frequent sequences from the set of sequences { S_c ∀ c ∈ C_i } MinSupport = max(4 calls, 5% calls) Output <representative terms for C_i, SP> } </pre>
--

Figure 2. Sequence Mining Approach

6 Experimental Study

6.1 Data set

We used the data that was obtained from the IT internal helpdesk of a company. It consists of a set of transcripts of calls. 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 4000 calls containing about 68000 sentences.

6.2 SPTS Clusters and Corpus Homogeneity

In this subsection, we test the hypothesis laid down in Section 4.4. Given our aim of seeing how the homogeneity of the corpus affects the quality of SPTS clusters, we would like to cluster corpora which differ only in their homogeneity. We compare the NE values of pairs of corpora which have roughly the same number of calls and sentences using K-Means clustering as described in Figure 3.

Experiment Methodology Corpus Homogeneity and Clustering Quality
<pre> Corpus = Total Collection of Calls; Cluster Corpus by K-Means (K=k) to get groups of Calls C = {C₁, C₂, ..., C_k} ; Build another set of groups of random calls (from Corpus) D = {D₁, D₂, ..., D_k} such that the total number of sentences (across calls) in D_i and C_i (for every i between 1 and k) are approximately the same; For (i=1; i<=k; i++){ DSC_i = {p ∃(c ∈ C_i) s.t. p ∈ c}, the collection of all sentences in C_i DSD_i = {p ∃(c ∈ D_i) s.t. p ∈ c}, the collection of all sentences in D_i Cluster DSD_i and DSC_i using K-Means, setting K to max(DSC_i /10, 10); NEC_i = NE of the clustering of DSC_i NED_i = NE of the clustering of DSD_i } NEC = (∑_i NEC_i* DSC_i) / (∑_i DSC_i); NED = (∑_i NED_i* DSD_i) / (∑_i DSD_i); </pre>

Figure 3. Corpus Homogeneity Experiments

The experiment takes as input a corpus of calls and uses K-Means to cluster them into multiple groups (C_i s). We build another set of groups (D_i s) containing calls randomly picked from the corpus such that each group, D_i has approximately the same number of calls and sentences as that of C_i . It may be noted that C_i and the corresponding D_i would have approximately the same values for Sentences ratio and Call length ratio defined in Section 4.3.

Given that clustering is a method of arriving at clusters of calls maximizing intra-cluster similarity and minimizing inter-cluster similarity, each of the C_i s can be expected to be more homogeneous than the corresponding D_i . Now, we compare the NE measures to test our hypothesis. The NE measure of a group of calls is computed as the NE measure of the clustering of the collection of sentences from those calls into sentence clusters (as mentioned in Section 4.3). As we would have as many pairs of NE values as the number of sub-collections, we obtain the cardinality weighted average of the collections so that we would have a single pair of values for the entire collection. This experiment was conducted for varying values of K and corpus subsets containing only agent sentences and customer sentences separately. We present representative results here.

Table 1. NEs against Homogeneity of Clusters

K	NE of Homogeneous Groups (NEC)	NE of Random Groups (NED)
50	0.842	0.795
100	0.787	0.731
K	NEC (Customer)	NED (Customer)
50	0.842	0.802
100	0.787	0.732
K	NEC (Agent)	NED (Agent)
50	0.840	0.7886
100	0.784	0.729

Table 1 above shows that the C collections (which are outputs of K-Means and hence more homogeneous) consistently score higher than the D collections (which are random). Yet another way to look at the comparison would be to compare the number of cases where NEC_i s are greater than NED_i s (NEC_i win) and vice versa.

Table 2. Homogeneity in terms of Wins

k	NEC_i wins	NED_i wins
50	82%	18%
100	80%	20%
k	NEC_i (Customer) wins	NED_i (Customer) wins
50	78%	22%
100	78%	22%
k	NEC_i (Agent) wins	NED_i (Agent) wins
50	84%	16%
100	80%	20%

Table 2 further validates the hypothesis. The homogeneous clusters score better over the random clusters with a prob-

ability of 0.8 (on an average). The results of the experiments validate the hypothesis viz., using homogeneous corpora improves the quality of the clustering for applications that require SPTS clusters.

6.3 Cluster Issue Analysis

Representative words give an idea about the homogeneity of the clusters obtained. A cluster could be addressing a problem specific to an application. We did a manual analysis of the 50 clusters (obtained by setting $K=50$ in K-Means) using only the top 5 representative words and tried to identify the specific problem addressed by each cluster. Figure 4 gives the problem against the number of clusters addressing the problem obtained.

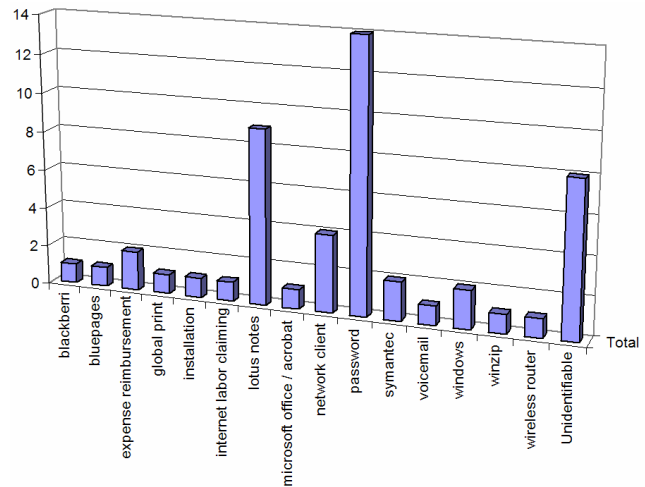


Figure 4. Cluster Issue Distribution

Fourteen clusters contained calls relating to some password issue, whereas nine clusters contained lotus notes issues. The topics of seven clusters could not be identified precisely. That the subjects of most of the clusters were identifiable only by looking at the top five representative words shows that most of the clusters are homogeneous, at least to some extent.

6.4 Sequence Analysis Results

We used a toolkit, called SLPMiner [Seno and Karypis, 2002], that implements well-known sequence mining algorithms, to find frequent sequences of clusters. We used AprioriAll algorithm with a minimum support of the larger of either 4 calls or 5% of the calls to perform sequential pattern mining.

The sequence analysis results in a set of <representative words, frequent sequences> tuples for each of the call clusters. We provide representative results from the sequence mining experiments below. Each homogeneous topic cluster

of calls and some SPTS clusters in each of these topic clusters are shown in table 3. The clusters are shown in the form of sets of most frequent words in them which aids visualization of the calls to some extent. Each sequence that we present is of the form <Start, ..., End>. Further, the prefix of each cluster represents whether it is of the “Customer” or “Agent” type. The categories for the clusters have been obtained from the experiment on cluster issue analysis described in the previous section.

Table 3. Sequence Analysis Results

Cluster 1 (Category: <i>Lotus Notes</i>) 56 Calls {archive, document, database, inbox, folder, notes, lotus}	
C_1	{notes, lotus, database, archive, don, click, workspace, mail, template, file}
A_1	{lotus, notes, open, click, archive, back, close, workspace, window, workstation}
Sequence 1 (41%)	<START, C_1 , END>
Sequence 2 (16%)	<START, A_1 , C_1 , END>
Cluster 2 (Category: <i>Expense Reimbursement</i>) 16 calls {don, expense, click, reimbursement, dot, error, application, working, launch}	
C_1	{expense, reimbursement, application, don, launch, fine, click, time, page, type}
A_1	{calling, speaking, call, stuff, today, didn, kind, don, number, contacting}
A_2	{expense, reimbursement, number, dot, launch, hold, correct, give, thing, time}
Sequence 1 (69%)	<START, C_1 , END>
Sequence 2 (50%)	<START, A_1 , A_2 , END>
Cluster 3 (Category: <i>Global Print</i>) 76 calls {printer, print, password, job, connect, global, click}	
A_1	{printer, working, give, problem, fox, don, information, hold, network, printers}
A_2	{printer, install, print, installing, installed, don, give, number, access}
Sequence 1 (28%)	<START, A_1 , END>
Sequence 2 (25%)	<START, A_2 , END>
Cluster 4 (Category: <i>Winzip/Compression</i>) 34 calls {zip, compress, file, password, winzip, license, connect}	
C_1	{windows, file, don, thousand, window, ve, explorer, kind, zip, size}
C_2	{click, double, file, open, zip, don, programs, files, program, start}
Sequence 1 (40%)	<START, C_1 , END>
Sequence 2 (24%)	<START, C_1 , C_2 , END>
Cluster 5 (Category: <i>Wireless Connectivity</i>) 81 calls {wireless, connect, router, network, password}	

C_1	{network, wireless, connection, find, connections, networks, open, click, status, access}
Sequence 1 (27%)	<START, C_1 , END>
Sequence 2 (16%)	<START, C_1 , C_1 , END>

We draw some sample inferences from table 4 to illustrate the utility of such sequences. The typical call flow for certain types of problems on Lotus Notes involves closing down the application, and then archiving the mail to a database. These are the steps illustrated by Sequence 2 of Cluster 1. The expense reimbursement in the company (that we collected the data from) is handled by means of a java web application. Sequence 1 in Cluster 2 says that people have troubles in launching the application. Sequence 2 in Cluster 3 talks about the instructions regarding giving the number of the printer to be installed using the web application for installing printers (“*Global Print*”). Sequence 2 in Cluster 4 reveals that most people having problems related to Winzip/Compression have *Windows 2000* on their machines. The second cluster in the sequence contains customer sentences on how they open a *file compression application*. Sequence 2 in Cluster 5 says that the workflow in handling a *wireless connectivity problem* involves checking the status of the connection twice. The first cluster in Sequence 2 of Cluster 2 is a *beginning sequence*; it contains only greeting messages. As can be seen, frequent sequences provide interesting insights to the workflow involved in solving a problem represented by each of the homogeneous clusters.

6.5 Contiguous Step-Pair Analysis

Sequence Analysis using the AprioriAll algorithm gives sequences that may not be contiguous. < a , c > is a contiguous cluster pair with support k if there are k calls which pass directly from cluster a to cluster c . For instance, a contiguous pair <Start, G> or <G, End> with a very high support may lead to the inference that G is possibly a “greeting” or “sign-off” kind of cluster respectively. This analysis is motivated by the results of the AprioriAll analysis. The first cluster in sequence 2 of cluster 2 was found to be a “greeting” cluster, and hence, the information conveyed by that cluster regarding the workflow particular to solving the problem is minimal. Identifying such clusters is a straightforward application of identifying contiguous step-pairs. Further, as most such pairs are bound to be of the type <customer-sentence, agent-sentence> or <agent-sentence, customer-sentence> given that each call is an alternating sequence of customer and agent clusters, it would aid us in identifying typical (customer or agent) responses to a kind of (agent or customer) query or directive.

We illustrate the utility of such an analysis by means of one of the homogeneous collections of the *Expense Reimbursement* category. Figure 5 is the call graph for the collection with every edge labeled with the number of calls passing through the pair of clusters that it connects. The figure is the

subset of the call graph which enables us to identify contiguous step-pairs supported by at least 3 calls.

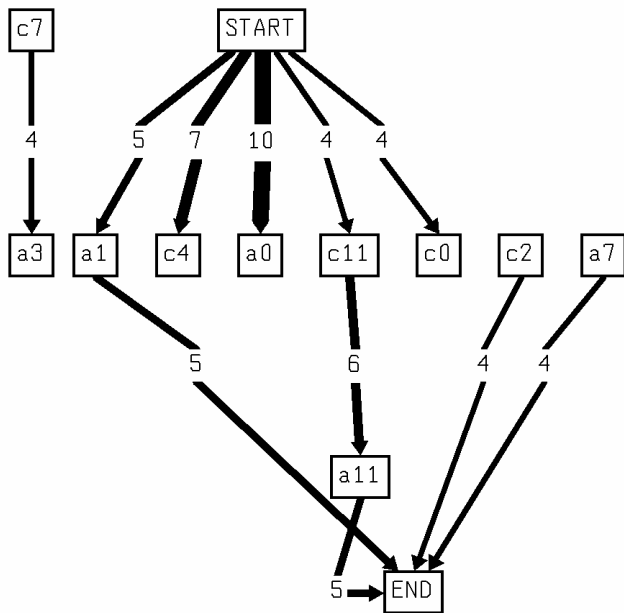


Figure 5. Call Graph (Edges < 4 pruned)

Due to lack of space, we present herewith table 4 describing a subset of nodes in the figure. These descriptions were arrived at by manual analysis of the sentences associated with the clusters.

Table 4. Contiguous Step-Pair Analysis Results

Cluster	Description
a0	This contains sentences containing a greeting and asking as to what kind of problem is being experienced
c11	This contains sentences wherein the customer complains that he/she is unable to access the Expense Reimbursement application
a11	This contains sentences where the agent says that there is a problem with the infrastructure supporting expense reimbursement
a7	This contains sentences where the agent assigns a ticket number and says a person would call back the customer

A lot of interesting inferences could be drawn out of such observations. a0 is hence confirmed as a “greeting” cluster (appears in the pair <Start,a0> with a support of 10 calls) and a7 is a kind of “sign-off” cluster. A pair of interest is <c11,a11>. From the descriptions presented in table 4, it can be identified to be representing a particular call flow where there is an infrastructure problem which is the cause of the outage that the customer complains off.

7 Summary and Future Work

We define SPTS clusters and their applications in knowledge extraction from call center transcripts. The proposed entropy based quality measure provides a method of quantifying the goodness of a clustering in terms of scatter of the calls among the clusters generated. Our studies show that homogeneous collections lead to better SPTS clusters. Our initial results presented here show that distinct and frequent sequences of SPTS clusters are representative of frequent workflows. We also show that contiguous pairs of SPTS clusters provide useful information regarding typical responses, starting and ending SPTS clusters. It may be noted that we *did not* attempt to quantify, characterize or eliminate the noise in the data.

As mentioned above, this paper presents only a preliminary set of results of our mining approach. There are various issues which need to be addressed to apply the proposed technique to practical scenarios.

- Automatically learning the number of clusters in a given corpus of calls utilizing the temporal information from calls and NE measure would enable us to make the method largely unsupervised.
- Although we have chosen discriminative and descriptive words to represent the SPTS clusters, using more sophisticated techniques such as leader clustering or CAARD to find the most representative sentences or phrases from SPTS clusters would improve understandability of the step that SPTS clusters represent.
- A comparative study on the relative effectiveness of partitioning algorithms and hierarchical clustering algorithms and their flexibility in taking the temporal information (manifested as the order of sentences in the calls) into account would provide insights regarding directions to proceed.
- An algorithm for noise removal could be plugged into an appropriate step in the process to improve the quality of the results.
- Doing a feature selection on the collection of SPTS clusters to remove those clusters which do not contribute to the process of solving the problem would also add value to the proposed approach.
- Pairs of clusters could be mined to identify <question, answer> pairs or <directive, response> pairs which could be used in automating the answering of certain portions of the call before transferring it to a specialized agent.

References

- [Agarwal and Srikant, 1995] Rakesh Agarwal, Ramakrishnan Srikant, “Mining Sequential Patterns”, in *Proceedings of the 11th International Conference on Data Engineering*, Taiwan, 1995
- [Agrawal et. al. 1998] R. Agrawal, D. Gunopulos, and F. Leymann, “Mining process models from workflow logs,” In the *Proceedings of the Sixth International Con-*

ference on Extending Database Technology, pages 469-483, 1998.

- [Jain and Dubes, 1988] A. K. Jain and R. C. Dubes, *Algorithms for Clustering Data*, Prentice-Hall, Englewood Cliffs, N.J., 1988.
- [Krishna and Krishnapuram, 2001] K. Krishna and Raghuram Krishnapuram, "A Clustering Algorithm for Asymmetrically Related Data with its Applications to Text Mining," *CIKM2001*, Atlanta, USA, November 2001. Pages 571-573.
- [Liu et. al., 2003] Liu, Liu, Chen, Ma, "An Evaluation of Feature Selection for Text Clustering", in Proceedings of the *International Conference on Machine Learning, ICML*, 2003
- [MacQueen, 1967] MacQueen JB, "Some methods of classification and analysis of multivariate observations," Proceedings of the *5th Symposium on Math, Statistics and Probability, Berkeley, CA*, 1967
- [Roy and Subramaniam, 2006] Shourya Roy and L Venkata Subramaniam, "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*, 2006.
- [Seno and Karypis, 2002] Masakazu Seno and George Karypis, "SLPMiner: An Algorithm for Finding Frequent Sequential Patterns Using Length Decreasing Support Constraint," in Proceedings of the *2nd International Conference on Data Mining*, 2002.
- [Zhao and Karypis, 2001] Ying Zhao and George Karypis, "Criterion Function for Document Clustering: Experiments and Analysis," Department of Computer Science, University of Minnesota, UMN CS 01-040, 2001