Unsupervised Segmentation of Conversational Transcripts

Krishna Kummamuru*, Deepak P* Shourya Roy†, L Venkata Subramaniam†

Abstract

Contact centers provide dialog based support to organizations to address various customer related issues. We have observed that the calls received at contact centers mostly follow well defined patterns. Such call flows not only specify how an agent should proceed in a call, handle objections, persuade customers, follow compliance issues, etc but also help to structure the operational process of call handling. Automatically identifying such patterns in terms of distinct segments from a collection of transcripts of conversations would improve productivity of agents as well as track compliance to guidelines. Call transcripts from call centers typically tend to be noisy owing to the noise arising from agent/caller distractions, and errors introduced by the speech recognition engine. Such noise makes classical text segmentation algorithms such as TextTiling, which work on each transcript in isolation, very inappropriate. But such noise effects become statistically insignificant over a corpus of similar calls. In this paper, we propose an algorithm to segment conversational transcripts in an unsupervised way utilizing corpus level information of similar call transcripts. We show that our approach outperforms the classical TextTiling algorithm and also describe ways to improve the segmentation using limited supervision. We discuss various ways of evaluating such an algorithm. We apply the proposed algorithm to a corpus of transcripts of calls from a car reservation call center and evaluate it using various evaluation measures. We apply segmentation to the problem of automatically checking the compliance of agents and show that our segmentation algorithm considerably improves the precision.

1 Introduction

Many companies today maintain contact centers in which professional agents interact with the customers and vice versa through phone calls, online chat and/or emails to handle various types of customer related issues. These types range from technical support to promotional to transactional (travel booking, etc.) in nature. The contact centers typically handle hundreds of calls depending on the nature of the business of the organization. Because of speech recognition systems, the transcripts of the calls at the contact centers are available in large volumes. There is a wealth of information hidden in these transcripts that could be useful to the organizations if it is extracted and used appropriately.

Text analytics can play an important role in performing deeper and more insightful analysis of conversational transcripts. However, application of text analytics in contact center scenario is not very common because, contact centers do not have in-house text analytics capabilities. Also owing to the confidential nature of the contact center data, for example, the business information contained in emails, call logs, and call transcripts, the data can not be made public. Hence such data never reaches the text mining community. We have been interacting with services organizations including contact centers. In this paper, we will share our experiences of applying text analytics techniques on *real-life* data. Specifically, we identify a type of information, viz., segmented calls, that is specific to conversational text which can be used for various tasks by the organizations and propose an approach to extract the same.

It has been observed that within a domain, or within an instance of a domain, the interactions in a contact center follow some specific patterns. This is mainly because of similar nature of queries, requests, and complaints received at the contact centers. Moreover, agent supervisors encode the best practices in the form of a call flow to train agents on ways to handle different types of calls. Such a call flow can be thought of as a directed acyclic graph having a start and end state, where each state performs some specific function. Any call must follow one of the paths from start to end.

Consider the example of a typical call center which receives requests, queries, complaints etc about renting cars. In a typical conversation there, call center agents and callers take turns speaking to each other. As mentioned earlier, the agents are supposed to follow a call flow to address the needs of the customers. For example, agents are expected to greet the customer at the beginning, gather requirement(s) of the customer, provide options, handle customer objections, either confirm booking and/or thank customers for their interest and finally conclude the call. A representative snippet of an example interaction is given in Figure 1 and the call flow

*IBM India Research Lab, Bangalore, India. Email: {kkummamu,deepak.s.p}@in.ibm.com

†IBM India Research Lab, New Delhi, India. Email: {rshourya,lvsubram}@in.ibm.com

```
AGENT: Welcome to CarCompanyA. My name is Albert.
How may I help you?
.........
AGENT: Alright may i know the location you want
to pick the car from.
CUSTOMER: Aah ok I need it from SFO.
.........
AGENT: Wonderful so let me see ok mam so we
have a 12 or 15 passenger van available on
this location on those dates and for that your
estimated total for those three dates just
300.58$ this is with Taxes with surcharges and
with free unlimited free mileage.
.........
CUSTOMER: oh and one more question Is it just in
states or could you travel out of states
.........
AGENT: alright mam let me recap the dates you
want to pick it up from SFO on 3rd August and
drop it off on august 6th in LA alright
.........
AGENT: The confirmation number for your booking
is 221 384.
CUSTOMER: ok ok Thank you
AGENT: Thank you for calling CarCompanyA and you
have a great day good bye
```

Figure 1: Snippet of an Example Interaction

for the same domain is shown in Figure 2. Ocassionally, calls could deviate from the typical pattern, because of agent/caller distractions, unexpected responses etc.

Automatically identifying the segments in calls has many potential applications in call analysis. Viz.,

Call monitoring: In operational settings, once the call flow and guidelines are in place, *Quality Analysts* (QAs) manually analyze a small sample of the calls regularly. Their job is to ensure call compliance which effectively means the prescribed call flows are being followed. However, owing to the fact that plenty of such calls happen everyday (can go up to few thousand calls per day) and the calls can be of long duration (with calls as long as 30 minutes), the task of QAs are quite tedious. As a result, such manual analysis processes are *inconsistent* (many times one QA does not agree with another QA on the compliance of a call) and *slow*, (listening to long calls and reaching conclusions takes time). One of the tasks of Quality Analysis is compliance checking. For example, compliance rules could include requirements that agents should greet the caller, agents should check whether the customer has a valid driver license, etc. Intuitively, compliance

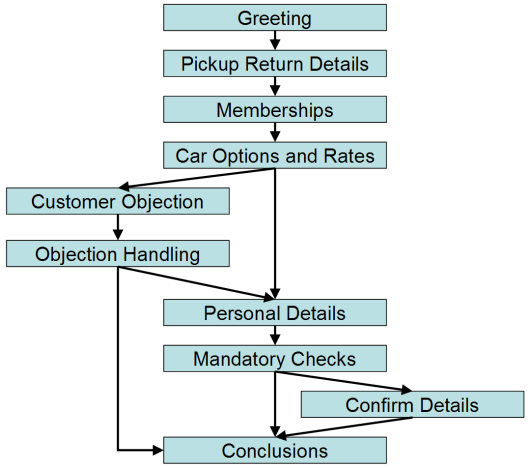

Figure 2: Structure of a Car Rental Process from a Call Center

checking can be identified with better precision when the calls are segmented. We have also observed the same in our experiments.

Agent monitoring: Certain agents perform better than other agents. Segmentation allows the call analysts to determine what good agents are doing in key segments so that all the agents can be trained to follow these positive traits. Also, call analysts try to identify patterns in problems and design corresponding solutions for the ease of training and grooming of new as well as existing agents. Identification of patterns in problems would be eased if the calls are presented to the analysts in segments.

Our contributions: The main contributions of this work are:

- A collection level technique to automatically segment call transcripts, and
- two novel evaluation measures to evaluate segmentation algorithms.

The proposed technique is unsupervised in nature and perform well without any manual intervention. This is important mainly because obtaining labeled data (segmented calls) to learn models is difficult, and many a times, it is impossible. At best, manual supervision in terms of *characteristic phrases* can be incorporated to improve the performance of the algorithm. The proposed technique is also shown to degrade gracefully even in the presence of noise introduced by Automatic Speech Recognition (ASR) system.

Organization of the paper: We describe related work in the next section. We describe the proposed algorithm

in Section 3. In Section 4, we describe a method to improve the segmentation using supervision derived from domain knowledge. We discuss various evaluation measures used to measure the quality of segmentation of conversational text in Section 5. Our experimental study is described in Section 6. Finally, this paper ends with summary and conclusions in Section 7.

2 Related Work

In the contact center scenario, we are not aware of any prior work trying to automatically identify segments (or states) in conversational data in an unsupervised way. This is the first work to raise the importance of the problem in addition to proposing techniques to solve the problem. The only part of the literature related to the present work is that of segmentation of text documents.

Current approaches to unsupervised text segmentation consider each document to be segmented in isolation. They work either by using coherence features to identify cohesive segments of text [6] [17] or by detecting topic shift indicators [14]. Certain techniques use keyword repetitions [13] or semantic networks [8] to detect topic boundaries. The well-known Topic Detection and Tracking (TDT) technique [1] also falls into this family of algorithms. Unlike TDT, our algorithm uses knowledge from a corpus of similar call transcripts to segment each transcript in the corpus. Supervised text segmentation techniques build a model from the training data and apply it to the test data; certain models work by extracting features that correlate with the presence of boundaries [2].

Text Analytics on call center dialogs have focused on clustering turns to gain insights into business processes [4] and to create domain models [15]. Call transcripts have also been analyzed for topic classification [5], quality classification [18] and for estimating domain specific importance of call center fragments [10]. Techniques which work well on clean text generally perform very badly on noisy text [3].

3 Segmentation of Conversational Text

3.1 Overview of the Call Segmentation Algorithm: The objective of call segmentation is to split calls into sequences of sets (segments) of contiguous turns, each such set corresponding to a logical state of the call flow. A turn is a contiguous sequence of words spoken by the same speaker (in a call). Our approach does call segmentation in an unsupervised setting, making use of a corpus of similar calls. There are two steps in unsupervised segmentation of call transcripts. Firstly, we identify characterizations of calls that could represent segments in the calls by analyzing the corpus of call transcripts and then, use these characterizations to segment a given call transcript.

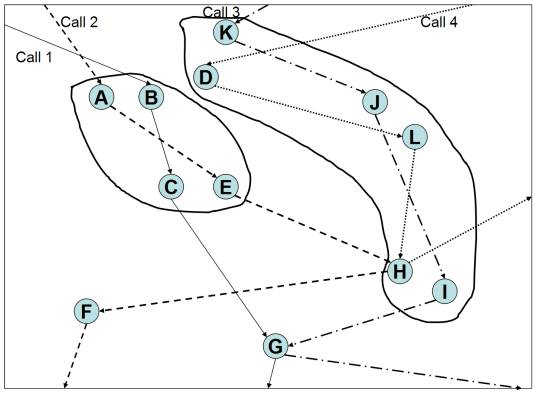

Figure 3: Section of a Call Flow Graph

Typically, textual documents are represented as a point in the vector space model [16] in which each dimension represents a word in the document. In case of conversational transcripts, we represent them as a piecewise directional line in the vector space where each turn is represent as a point and two consecutive turns are connected by a line as shown in Figure 3. *Call1* contains the sequence of turns B-C-G whereas *Call2* contains the sequence A-E-H-F and so on. Lexically C & E are closer than C & G or A & H. In this setting, we characterize the segments by collections of turns (points) which are lexically similar (close in the vector space) but also temporally occur together more often. For example, $(\{A, B\}, \{C, E\})$ could characterize a segment as $\{A, B\}$ and $\{C, E\}$ are lexically similar, and temporally occur close to each other on two (lexically) similar calls, $Call1$ and $Call2$. Similarly, $(\{K, D\}, \{J, L\}, \{H, I\})$ could also represent a segment. We represent lexically similar points by their centroid. The resulting characterization, viz., sequence of centroids, is referred to as a Representative Segment (RS) later in the paper.

In our implementation, we use the cosine similarity of term vectors to model lexical similarity and a pairwise neighborhood based similarity measure to model temporal proximity (which we detail in a later section). Note that clustering based on lexical similarity can be done by means of a linear algorithm whereas that using pairwise similarity measures demand a quadratic algorithm. Therefore, the proposed algorithm to find RSs is a two step clustering process in which clustering in the first step is done using cosine similarity (linear) followed by the second using the pairwise similarity (quadratic). This process optimizes the computational cost since in the second step, the quadratic algorithm works on an already clustered input. Finally, our approach to call

segmentation is summarized in the following steps:

1. Generate RSs
   (a) Cluster sentences from across calls using lexical similarity to arrive at sub-steps of a segment (Section 3.2)
   (b) Cluster the sub-steps from the previous step using temporal proximity based measures to arrive at RSs (Section 3.3)
2. Use RSs to segment individual calls (Section 3.4)
   (a) Represent each call as a sequence of RS indexes by replacing each turn by the index of the lexically closest segment summary and a confidence score using positioning information
   (b) Use a variant of agglomerative clustering, which is robust to noise, to merge adjacent segments to arrive at a segmentation of the call

3.2 Clustering Using Lexical Similarity Let $\mathcal{C}$ be the collection of calls $\{C_1, \ldots, C_N\}$ under consideration. Each call C_i is represented by a sequence of turns $\{t_1(C_i), \ldots, t_{|C_i|}(C_i)\}$ where $|C_i|$ is the number of turns in the call. Each turn is either spoken by the caller or the callee.

In order to work with a set of similar calls (on same topic), we first cluster $\mathcal{C}$ into K clusters using K-means algorithm [9]. It may be noted that if the corpus contains calls on the same topic, one can skip this initial phase of clustering (in this case, $K = 1$). Let, $\mathcal{T}_1, \ldots, \mathcal{T}_K$ be the topic clusters and

$$\mathcal{G}_i = \bigcup_{\forall l, C_j \in \mathcal{T}_i, Speaker(t_l(C_j))=caller} t_l(C_j),$$

be the set of turns spoken by callers, and

$$\mathcal{H}_i = \bigcup_{\forall l, C_j \in \mathcal{T}_i, Speaker(t_l(C_j))=callee} t_l(C_j)$$

be the set of turns spoken by receiving call center agents. $\mathcal{G}_i$s and $\mathcal{H}_i$s are clustered separately using the cosine similarity measure to obtain lexically similar clusters of turns, which we refer to as Sub-Procedure Text Segments (SPTS). Similar words and phrases are spoken by agents (and customers) in different calls due to the similar nature of interactions and SPTSs are expected to group such similar turns together. We use the K-means algorithm (KMA) to cluster $\mathcal{G}_i$s and $\mathcal{H}_i$s.

We determine number of clusters, K, of $\mathcal{G}_i$s and $\mathcal{H}_i$s by optimizing a measure which gives a quantification of the scatter of calls in the corpus across the clusters; we call this as the SCE (SPTS-Cluster-Entropy) measure. We describe the SCE measure in more detail in the next few paragraphs.

We say that a given clustering $\mathcal{S}$ is good if many calls in the corpus are scattered across many clusters in $\mathcal{S}$. That is, we define the SCE measure in terms of how each call is scattered across various clusters in $\mathcal{S}$.

DEFINITION 3.1. *Normalized entropy (NE) of a call C with respect to $\mathcal{S}$ is defined as,*

$$NE_{\mathcal{S}}(C) = -(\sum_i d_i \log(d_i))/\log(n),$$

where, d_i is the fraction of the call C in cluster S_i and n is the length of the call C. ■

NE would assume a value between 0 and 1 since $\log(n)$ is the maximum value that Entropy can assume for a call of length n.

EXAMPLE 3.1. *Let, the call C_1 be represented by the sequence of SPTSs $(S_2, S_1, S_5, S_6, S_4)$ and the call C_2 $(S_3, S_5, S_5, S_3, S_5)$. As is obvious from the representation, C_1 is more scattered than C_2. The entropy of d_i, viz., $E_{\mathcal{S}}(C) = -\sum_i d_i \log(d_i)$, captures this scatter. Because, when $\mathcal{S} = \{S_1, \ldots, S_6\}$, $E_{\mathcal{S}}(C_1) = 0.6989$, and $E_{\mathcal{S}}(C_2) = 0.29$.*

It may be noted that the NE measure works well to compare calls of different cardinalities unlike the entropy measure. Consider the case in which C_3 is (S_1, S_2) and C_4, $(S_1, S_1, S_1, S_1, S_2, S_2, S_2, S_2)$, 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. C_3 would have an NE value of 1.0 and C_4 would have an NE value of 0.333. ■

DEFINITION 3.2. *Let $\mathcal{C} = \{C_1, \ldots, C_N\}$ be the collection of calls and $\mathcal{S} = \{S_1, \ldots, S_M\}$ be the sets of SPTSs. Then, each C_i is represented by a sequence of S_js. Let C_i be represented the sequence $\{s_{i1}, \ldots, s_{in_i}\}$ where $s_{ij} \in \mathcal{S}$. Then, the SCE measure of $\mathcal{S}$ with respect to the corpus $\mathcal{C}$ is defined as*

$$SCE_{\mathcal{C}}(\mathcal{S}) = \left(\frac{\sum_{i=1}^{N} n_i NE_{\mathcal{S}}(C_i)}{\sum_i n_i} \right). \tag{3.1}$$

■

That is, SCE is the cardinality weighted average of NE values of the calls in the corpus. Properties of SCE measure are detailed in Appendix B.

Although maximizing scatter of calls (as quantified by the SCE measure) ensures better quality of SPTS

clusters, we would ideally want to have the smallest possible number of SPTS clusters which also has a reasonable scatter. It may be noted that these requirements are quite contradicting because the scatter tends to increase with the number of clusters. We accomplish our goal of finding a small number of clusters maximizing scatter by varying the two parameters of the K-means algorithm, namely the random seed and K by iterating over the following sequence of steps starting from a small value for K:

1. Let the current value of K be k. Vary the random seed t times for $K = k$ and $K = k + 1$, run the K-means algorithm for each such combination. Pick the best values for the SCE measure for both $K = k$ and $K = k + 1$
2. If the SCE value chosen in the above step for $K = k+1$ is not better than that chosen 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 enables reaching the termination condition early, but may merge many logical sub-steps. On the contrary, setting p to a low value risks splitting up of logical sub-steps.

3.3 Finding Segment Summaries As detailed in Section 3.1, we cluster SPTSs based on temporal coherence to arrive at segment summaries, referred to as Representative Segments (RSs). We define a similarity measure between SPTS and use the same in a relational-clustering algorithm to partition SPTS into a given number of clusters. The similarity measure is dependent on frequency of their co-occurrence.

DEFINITION 3.3. *Let χ_{ij} represent the number of times SPTS S_i and S_j occurred in the corpus of calls within a neighborhood of ω. That is,*

$$\chi_{ij} = |\{(k, l, m) : s_{kl} = S_i, s_{km} = S_j, |l - m| < \omega\}|.$$

where s_{ab} refers to the SPTS cluster associated with the b^{th} turn in the a^{th} call. Let $\Phi = \max_{ij} \chi_{ij}$. Then, the similarity, $\psi(S_i, S_j)$, between S_i and S_j is defined as

$$\psi(S_i, S_j) = \frac{\chi_{ij}}{\Phi}.$$

■

ψ measures the frequency of co-occurrence of pairs of SPTS in the corpus of calls within a specific neighborhood. We use Agglomerative Hierarchical Clustering [11] to cluster SPTSs using the above similarity measure. We denote the resulting clusters (i.e., the RSs) by $\Sigma_1, \ldots, \Sigma_L$. Note that each of these clusters is represented by the set of SPTSs that it contains. That is, $\Sigma_i \subseteq \mathcal{S}$ and $\Sigma_i \cap \Sigma_j = \emptyset, \forall i \neq j$.

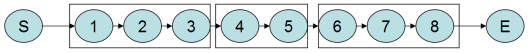

Figure 4: A call represented as a sequence of SPTS and a possible segmentation.

EXAMPLE 3.2. *In Figure 3, assume that the turns J & L fall in the same SPTS cluster S_1 by virtue of their lexical similarity. Similarly, $\{H, I\} = S_2$ and $\{C, E\} = S_3$ due to the same reason. Assuming that $\omega = 2$, χ_{12} would evaluate to 2 because S_1 and S_2 appear adjacent to each other in Call 3 and Call 4. On the other hand, χ_{23} would evaluate to 1 because there is only one call, Call 2, where S_2 and S_3 appear within a distance of 2. This leads to a higher chance that the SPTS cluster pair S_1 and S_2 gets merged into the same RS as compared to the pair of S_2 and S_3.*

3.4 Segmenting Calls using RS Given a new call C of length n, it is converted into a sequence of representative segment identifiers (RSIDs) by replacing each turn in the call by the identifier of the RS to which it is most similar lexically. Let $\{t_1(C), \ldots, t_n(C)\}$ be the sequence of turns of the call C and $\{r_1, \ldots, r_n\}$ be the corresponding sequence of RSIDs. That is, $\exists s_i \in r_i, t_i(C) \in s_i$ and $r_i \in \{\Sigma_1, \ldots, \Sigma_L\}, \forall i$ i.e., each $t_i(C)$ is closest to an SPTS which is contained within r_i. Segmentation is the process of dividing the call into sequences of contiguous groups of turns where each group is referred to as a *segment*. Pictorially this is represented in Figure 4. Here the call contains 8 turns and each turn is mapped to some RSID, r_j. An example segmentation of this call is shown with highlighted boxes: The first three sentences form the first segment, the next two form the second and the remaining sentences belong to the third segment.

We represent a segmentation J of C, as a sequence of numbers $(j_1, \ldots, j_d)$ where the first j_1 sentences of C belong to the first segment, the next j_2 sentences belong to the next segment and so on. It may be noted that $\sum_k j_k = n$, the length of the call C. We describe an algorithm to arrive at a segmentation of C from the sequence of RSIDs in the rest of the section.

DEFINITION 3.4. *Let a call C be of length n and an RS Σ occurred at a position j in the call sequence. Then, its relative position of occurrence is j/n. The average relative position of occurrence of Σ, $AV(\Sigma)$, in a corpus*

$\mathcal{C}$ is defined as the average of relative positions of each occurrence of the RS in $\mathcal{C}$. ■

We use the average relative positions of each Σ_i to estimate the genuineness of each sentence in C to enable robust segmentation. We will elaborate on this in the following paragraph.

DEFINITION 3.5. *The probability of jth RS, r_j in a call of length n, being a non-noisy occurrence is defined as*

$$PN(r_j) = 1 - |j/n - AV(r_j)|.$$

■

It is intuitive to expect that that the occurrence of an RS at a position very far from where it usually occurs (in a call), is likely to be a noisy occurrence. The probability of non-noise defined above captures this notion by assigning to every RS occurrence, a weight inversely related to its distance from the usual occurrence position, modeled as the average fraction of call elapsed before the RS occurs. This PN value is a measure of genuineness. It may be noted that the weight is linear with the distance from the average relative position. One can use other variations like those that depend on the position of the neighboring average relative positions.

DEFINITION 3.6. *Purity a segment $J_{kl} = (r_k, \ldots, r_{k+l})$ in a call C is given by*

$$Purity(J_{kl}) = \frac{1}{\sum_{i=k}^{k+l} PN(r_i)} \max_p \sum_{i=k, r_i=\Sigma_p}^{k+l} PN(r_i).$$

■

Purity of a segment is thus quantified as ratio of the presence of the maximally present RS to the cumulative presence of RSs; the presence of each RS being calculated as the sum of the non-noise probabilities of each occurrence of the RS in the segment.

We segment a call C into a sequence $j_1, \ldots, j_d$ such that the purity of each segment is greater than a threshold μ, $(0.0 \leq \mu \leq 1.0)$. We achieve the same by using a variant of Agglomerative Hierarchical Clustering. We start by assigning each sentence to it's own segment labeled with the RS identifier of the sentence. We merge these segments in a robust fashion iterating through the sequence of steps as follows.

1. Merge contiguous sequences of segments which have the same segment label to form a single segment.
2. Find the pair of adjacent segments (S_i, S_j), merger of which results in a segment of maximal purity, i.e., find a pair of adjacent segments (S_i, S_j) such that $Purity(S_i \cup S_j) \geq Purity(S_x \cup S_y)\ \forall x, y$. Ties are broken arbitrarily.
   (a) If the merger of S_i and S_j results in a segment of purity of less than μ, stop the agglomerative process and output the current segmentation.
   (b) Else, merge the segments and assign them the label of the RS that has maximal concentration on the merged segment. Go to Step 1.

It may be noted that this variant enforces the restriction that each segment may be merged only with one of its adjacent segments. This restriction makes the algorithm linear in the length of the call.

EXAMPLE 3.3. *Let the Call C_1 having 20 sentences be represented by the RS Sequence*

$$(\Sigma_2, \Sigma_2, \Sigma_2, \Sigma_2, \Sigma_3, \Sigma_2, \Sigma_4, \Sigma_4, \Sigma_4, \Sigma_4, \Sigma_4, \ldots)$$

where the 5th sentence is "agent: thank you for calling XYZ" with the AV values for Σ_2 and Σ_3 being 0.05 and 0.9 respectively. Notice that the 5^{th} sentence is a noisy occurrence of a "Sign-Off message", wherein the agent mistakenly assumes that he has completed the transaction. The probabilities of non-noise for the corresponding sentences can be calculated to be $(1.0, 0.95, 0.90, 0.85, 0.35, 0.75, \ldots)$. The probability of non-noise corresponding to the 5^{th} sentence is found to be 0.35, since this occurrence is very far-off in the call from the usual occurrence of similar sentences. The algorithm merges the first 4 turns upfront as they all are labeled with the same RS. The purity of the segment containing the first 6 turns would be 4.45 (the sum of the PN values for occur.

4 Using Supervision for Better Segmentation

In this section, we explore the usage of domain information and develop techniques to incorporate the same as supervision into the segmentation process. Segments in the transcripts of contact center calls can be distinctly identified by sets of phrases, commonly used by customers and agents. Domain information can also be procured as a manually segmented call collection. Generating a large collection of manually segmented calls is a laborious task whereas defining characteristic phrases is relatively easier for domain experts. Moreover, characteristic phrases per segment type can be re-used across processes for those segment types that are common across processes. Some examples of commonly used phrases obtained from a contact center is

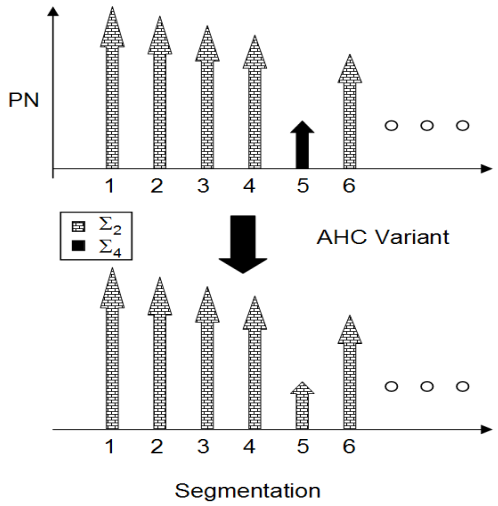

Figure 5: Example of Segmentation

given in Figure 6. There is another important benefit of using supervision in the segmentation process. Some representative segments obtained may not map to the states (as introduced in section 1) which have business significance whereas some segments may map to multiple of them. We use characteristic phrase collections to improve the set of RSs as well as to attach them with states having business significance. We will refer to this step as *RS repair phase* in the sequel.

Pickup Return Details: date and time, on what date, what date and, at what time, from which location, around what time, what date, date and, and time, which location, what time, on what, like to pick, to pick up, a car, to pick, i want to, you like to, time, would you like;
Car Options and Rates: includes all the, for the whole, includes all, unlimited,with all the, all the taxes, with all, including all the, all taxes and, charges, the taxes, the taxes and, all taxes, unlimited miles, with unlimited, taxes and, surcharges;

Figure 6: Examples of characteristic phrases of segments

Let $A = \{A_1, \ldots, A_M\}$ be the sets of phrases obtained from domain experts where each A_i corresponds to a pre-defined state or *state* G_i. We now measure the correspondence between an RS Σ_i and a pre-defined state G_j using $Score(\Sigma_i, G_j)$ which is defined as the total number of occurrences of phrases in the set A_j among the sentences in Σ_i. We use the scores to modify the collection of Σs to arrive at a new collection of Σs by the following sequence of operations.

- For an RS Σ_i, if the majority of occurrences of phrases are from the A_j, we assign Σ_i to G_j. If there is no such pre-defined state, we exclude Σ_i from the set of RSs.
- If multiple Σs get assigned to the same G_j, we merge them to form a single merged RS.

The above operations may decrease the number of RSs by deletion or merger. The usage of domain knowledge in this fashion seeks to arrive at a set of RSs which have a one-to-one mapping to the pre-defined states. The first step of deleting RSs which are not matched to a single pre-defined state helps to remove noisy RSs which may pollute the segmentation. Noisy RSs may be formed due to noisy sentences in the call, noise introduced by the clustering to SPTS clusters and/or by noise in the clustering of SPTS clusters to form RSs. The second step of merging RSs helps to remove any many-to-one mapping that exists between RSs and pre-defined states. It may be noted that the above sequence of steps is incapable of repairing any one-to-many mappings between RSs and pre-defined states. Thus, in the presence of domain knowledge, it helps to generate a larger number of RSs than the number of pre-defined states, so as to avoid any one-to-many mapping between the RSs and pre-defined states. The many-to-one mappings introduced due to the larger number of RSs would be eventually removed by the repairing step.

Finally, it may be noted that the above procedure heavily relies on the supervisory phrases and can not tolerate noise in the phrases. This problem can be easily addressed by assigning fuzzy membership values to the phrases in A_i and the function $Score(.,.)$ can be computed using these membership values.

5 Evaluation methodology

The best way to evaluate an automatic segmentation method is to compare the results of the segmentation with ground truth viz., manual segmentation. In this section, we provide two metrics for the same. The choice will depend on whether we have mapping between automatic segments and pre-defined states. We also discuss another evaluation metric with the emphasis on an application of the segmentation namely checking business compliance of the call-center agent.

5.1 Segmentation Quality Evaluation. Evaluation of call segmentation would involve checking the extent of match between segmentations. In this section,

we present two evaluation measures. In the manually segmented call, each segment in the call is assigned to a pre-defined state. On the other hand, the proposed unsupervised method generates segmentation where the segments do not have any semantic meaning - each segment is assigned to a unique number. However, in the presence of the RS repair phase, the algorithm does generate segments with pre-defined states. In the following subsections, we describe two evaluation methods for comparing pairs of call segmentations.

5.1.1 Subsequence Agreements. We propose a measure to compare two segmentations of a call by aggregating their *agreements on the segmentation of contiguous pairs of sentences in the call.* In the absence of correspondence between the segments in the automatic segmentation and states in the manual segmentation, we evaluate the quality of the former by aggregating the agreements between the segmentations over contiguous subsequences in the call.

DEFINITION 5.1. *Consider two segmentations, $J = (j_1, j_2, \ldots, j_d)$ and $K = (k_1, k_2, \ldots, k_f)$ for a call C. Assume a function f_i^J which gives the segment identifier to which the i^{th} sentence of a call C is mapped under the segmentation J. We define the measure of agreement A_{JK}^2 between the segmentations J and K for C as,*

$$A_{JK}^2 = \left(\frac{\sum_{i=1}^{n-2} I(I(f_i^J, f_{i+1}^J), I(f_i^K, f_{i+1}^K))}{\sum_{i=1}^{n-p}(1)} \right), \tag{5.2}$$

where $I(.,.)$ returns 1 if both the arguments are equal and 0 otherwise. It may be noted that A_{JK}^2 always assumes a value between 0 and 1. ■

The superscript 2 denotes the usage of pairs of contiguous sentences. This measure can be generalized for contiguous subsequences of length greater than 2. In general, the measure A_{JK}^p measures the fraction of contiguous subsequences of length p which have equivalent segmentations (according to the segmentations in question). To measure the agreement over collections of calls, we take the length-weighted average of the A_{JK}^p measure across calls in the collection. It may be noted that this measure does not require that the segment identifiers for the different segmentations come from the same space.

EXAMPLE 5.1. *Consider two segmentations $J = (5, 2, 7)$ and $K = (5, 1, 1, 7)$ of the call C of length 14. The segmentations are obviously not equivalent, nor do they have the same number of segments. But, as can be noted, the contiguous subsequence of the first 6 sentences from the call have the same segmentation of $(5, 1)$ from both the segmentations. So do the next 8 sentences from the call. The 3^{rd} segment break in K being spurious led to the two segmentations being non-equivalent. Intuitively, these segmentations have a very high match as they differ only in one segment break. The A_{JK}^2 estimates the match between these segmentations at a high value of 0.92. This shows how evaluating agreement over contiguous subsequences in the call captures the approximate match between the segmentations even in the presence of noise.*

5.1.2 Segment Type Matching. When the segment identifiers for the segmentations in question come from the same space, we can get a more intuitive measure of the quality of the automatic segmentation. In the presence of the RS repair phase, we obtain a unique mapping from each automatic segment to a pre-defined state.

DEFINITION 5.2. *Let J and K be the automatic and manual segmentations of the call C. Assume that each automatic segment s be mapped to a unique predefined state under the mapping $map(s)$. We define the quality measure B_{JK} for the automatic segmentation J of a call C against the manual segmentation K as,*

$$B_{JK} = \left(\frac{\sum_{i=1}^{n} I(map(f_i^J), f_i^K)}{n} \right). \tag{5.3}$$

Here, $I(.,.)$ and f_i^J are the same functions as defined in Section 5.1.1. ■

B_{JK} measures the fraction of sentences from the call that are mapped to the same segment type in both the automatic and manual segmentations.

5.2 Segmentation in Compliance Checking. In this subsection, we consider the problem of checking the compliance of call-center agents with guidelines. One way of automatically checking the compliance is to find whether predefined phrases relating to the guidelines are present in the call transcripts. For example, to check if the agent has confirmed that the customer has a valid driver license we can look for the key words "driver license", "valid", etc. However, the occurrences of key words for a mandatory check may occur in various contexts in a call, whereas we wish to extract only those occurrences which occur as part of the mandatory check that we are looking for. Therefore, we apply segmentation on call transcripts and check whether the predefined phrases are present in appropriate segments of the call. For example, confirming if the customer has a clean driving record should be present in *Mandatory Checks* segment, or any segment that maps to *Mandatory Checks*. In this case,

we evaluate segmentation by comparing the key word matches in the entire call and in a specific segment.

6 Experimental Study

In this section, we present an experimental study of the segmentation technique. We start by describing the experimental setup and the data set. We compare the proposed technique against other techniques and results confirm the superiority of our technique. We then analyze the effect of various factors such as presence of named entity information, supervision, and noisy transcripts and show that our technique improves with the former two factors, whereas it degrades gracefully with noise in the transcripts. Finally, we evaluate the proposed segmentation algorithm in the context of compliance checking application.

6.1 Experimental Setup and Data set. We used the CLUTO toolkit [7] implementations of K-Means and AHC for our experiments. Based on the empirical analysis, we set p, the minimum percentage of improvement in SCE measure when K is increased by 1 (defined in Section 3.2), to 15 in the clustering to find SPTS clusters. This led to an average of 12 turns per SPTS cluster. We use a window size of 2 to compute the similarity between SPTS (refer Definition 3.3). The AHC phase of clustering SPTS to form RSs was stopped when each RS has around 2-3 SPTSs on the average. The threshold for purity in the segmentation phase was chosen at 0.85 on an empirical basis.

We collected 137 calls from a car rental help desk and manually transcribed them. All these 137 calls had resulted in the customer making a car reservation. As all these calls are reservation calls from the same domain, they follow a similar call flow. We passed them through a named-entity tagger [12] and replaced each of the named-entity types by unique codes. This is expected to improve the performance of the segmentation process as the differences between the different instances of the same type such as *airport names*, *city names*, *dates* etc. get hidden from the segmentation process. From these reservation calls, we randomly picked 74 calls and manually segmented them. We also obtained a corpus of 30 calls from the same car rental process using Automatic Speech Recognition (ASR). The ASR transcripts are very noisy and have an average Word Error Rate (WER) of 40%, with a split up of 25% WER on agent sentences, and 55% WER on customer sentences. The ASR dataset helps us to evaluate the extent of noise tolerance of our technique.

In manual segmentation of these calls, a domain expert defined ten states and assigned each turn in the call to one of the ten states. The ten categories are: greeting, customer requirement, details (of pick up and return), membership, rates, personal details, mandatory checks, verify (the details), objection handling (and customer objection), conclusion (refer Figure 2). An example of a manual segmentation on the call shown in Figure 1 is given in Figure 7. Apart from this call dataset, we use domain knowledge in the form of characteristic phrases for pre-defined states (as in Figure 6) to evaluate the value-add that can be obtained using the technique presented in Section 4.

```
AGENT (Greeting):  Welcome to CarCompanyA. My
name is Albert.  How may I help you?
.........
AGENT (Pickup Return Details):  Alright may i
know the location you want to pick the car from.
CUSTOMER (Pickup Return Details):  Aah ok I need
it from SFO.
AGENT (Pickup Return Details):  For what date and
time.
.........
AGENT (Car Options and Rates):  Wonderful so let
me see ok mam so we have a 12 or 15 passenger van
available on this location on those dates and for
that your estimated total for those three dates
just 300.58$ this is with Taxes with surcharges
and with free unlimited free mileage.
.........
CUSTOMER (Customer Objection):  oh and one more
question Is it just in states or could you travel
out of states
.........
AGENT (Confirm Details):  alright mam let me
recap the dates you want to pick it up from SFO
on 3rd August and drop it off on august 6th in LA
alright
.........
AGENT (Conclusions):  The confirmation number for
your booking is 221 384.
CUSTOMER (Conclusions):  ok ok Thank you
AGENT (Conclusions):  Thank you for calling
CarCompanyA and you have a great day good bye
```

Figure 7: A Manually Segmented Dialog (partial).

6.2 Comparison with TextTiling and Support Based Segmentation. We evaluated our approach against the classical TextTiling algorithm [6] and a baseline technique, viz., support based segmentation described in Appendix A. TextTiling evaluates the statistical probability of each sentence boundary to be a segment break using the text on either side of it. We

used a window size as twice the length of the average length of sentences in the call. Support based segmentation is an intuitive way of utilizing sequence information and corpus-level information in segmentation. Our experiments show that our algorithm outperforms both the techniques by a good margin on the A^2_{JK} measure (whose numeric value is proportional to the quality). The results are presented in Table 1.

The support based segmentation algorithm was very slow in performance, the execution time varying from 2 seconds to 41 seconds for calls having a length of around 20, at the same time yielding a very low A^2_{JK} score of 45%. For certain highly structured processes, which have a well-defined workflow, sequences across segments may have a very high-support. This effects the performance of the support based technique whereas our algorithm is highly immune to varying levels of structure. Our algorithm outperforms TextTiling too by a difference of 0.11 in the A^2_{JK} score. TextTiling constructs document vectors making it very sensitive to noise; hence, we expect that its performance would deteriorate when applied to a collection of more noisy data such as ASR transcripts. This also shows the value addition caused by the usage of information from a corpus of similar calls (as our algorithm does) to segment each call.

Approach	A^2_{JK}	Time Per Call
RS-Based	0.6820	0.03 s
Support-Based	0.4512	19.2 s
TextTiling	0.5807	0.03 s

Table 1: Comparison with Support-Based Segmentation and TextTiling.

6.3 Effect of NE-Tagging. Typically calls have widely different ways of expressing the same concept or referring to the same artifact. Certain concepts that give information about the process, are at a much higher level than their actual occurrences in the call. For example, an occurrence of an airport name provides just as much information as another instance of location. These variations induce a lot of errors in the generation of SPTS clusters since the clustering relies on the scanty information from the sentences in the call. We cancel out such errors by replacing every named entity by its type. This pre-processing step resulted in an improvement in the A^2_{JK} by 5.4% to **0.7186**. We use the NE-tagged dataset in all the experiments hereon.

Type of Transcript	# Calls	A^2_{JK}	B_{JK}
Manual	74	0.7512	0.85
ASR	30	0.6915	0.62

Table 2: Comparison between ASR and Manual Transcripts.

6.4 Effect of Supervision. We were able to procure collections of characteristic phrases for 8 pre-defined states used in the car-rental process. Out of these, 4 segments had relatively error free phrases, whereas the other 4 had noise in the form of very generic phrases. Although our RS Repair phase is highly sensitive to noise in the supervision phase, we used all the 8 segment types as the A^2_{JK} measure is useful only when both the segmentations for a given call contain roughly the same number of segments. Despite the noise in the supervision, its usage improved the A^2_{JK} measure by 4.6% to **0.7512** from 0.682. The presence of predefined states enables quality assessment using the B_{JK} measure. This measure shows a very high-accuracy of 0.85 for the usage of the non-noisy supervision (4 segments), whereas the accuracy dropped to 0.52 with the usage of the noisy supervision (8 segments).

6.5 Effect of Noisy Data. The effectiveness of noise tolerance of our technique can be most effectively evaluated by observing it's performance on noisy data. We apply our techniques on the collection of ASR transcripts. It may be noted that our ASR data has only 30 calls, as opposed to 74 in our manual transcript collection. Thus, any deterioration in performance is to be attributed to the lesser amount of data as well as the noise in the data, and not solely to one factor. We use only non-noisy supervision (4 segment types) in our experiments on ASR transcripts. The results on ASR transcripts are compared against those in manual transcript collections in Table 2. We observe a deterioration of 8% in the A^2_{JK} measure whereas there is a deterioration of 27% in the B_{JK} measure. This shows that our techniques are relatively noise tolerant as they still achieve good performance while using highly noisy ASR transcripts with 40% WER.

6.6 Segmentation for Compliance Checking. Compliance checking involves verifying that an agent has performed all the mandatory checks that are prescribed by the contact center process. As mentioned in Section 5.2, to evaluate the quality of segmentation, we search for the key words, corresponding to a specific mandatory check, in the segment where the key task is

Check	Segmented Calls		Raw Calls	
	Precision	Recall	Precision	Recall
Drivers License	1.00	0.94	0.93	1.00
Credit Card	1.00	0.95	0.88	1.00
25 years	1.00	0.97	0.78	1.00

Table 3: Segmentation for Compliance Checking.

supposed to be present (for example, confirming if the customer has a clean driving record should be present in *Mandatory Checks* segment, or any segment that maps to *Mandatory Checks*) and compare the result with the keywords matches in the entire call. In this section, we consider three mandatory questions that an agent has to ask within a car-reservation call namely: asking if the customer has a valid driver license, asking if the customer holds a major credit card, confirming whether the customer is at least 25 years old.

In a segmented call, we check for the presence of the checks only in those segments where we expect to find them whereas, they are checked for in the entire call in the unsegmented scenario. We have labeled data of calls in which the labels correspond to occurrences of the mandatory checks. We evaluate the utility of segmentation in this task by reporting numbers for precision and recall. We summarize these results for each of the checks in Table 3.

The results show that segmentation has been particularly successful in reducing the number of false positives and thus has a much higher precision over the same task on the raw calls. From the low value of precision for the compliance checking task on raw call, we can see that there are a large number of instances which are wrongly detected as containing the key task (false positives). This is because the keywords that are characteristic of a mandatory check are likely to occur in other segments also. For example, consider the mandatory check of verifying that the customer has a "major credit card". Characteristic keywords for this task, such as "credit", "debit", "card" etc. are likely to occur in other segments in other contexts too. So, by looking at the entire call, it is not possible to capture the information if the agent has performed a particular key task or not due to the large number of false positives. This shows that segmentation can significantly improve agent monitoring processes such as compliance checking.

7 Summary and Conclusions

In this paper, we proposed an algorithm for automatic unsupervised segmentation of conversational transcripts. Our algorithm differs from the earlier approaches in the use of lexical coherence, textual proximity and position information. This makes our algorithm tolerant to noise in the transcripts. Further, we have outlined a step wherein supervision in the form of characteristic phrases can be used to improve the segmentation. We use a comprehensive set of evaluation measures which can handle approximate matches. We compare with an existing text segmentation algorithm. Our experimental results show that our algorithm performs better.

This is the first step toward the larger goal of providing an integrated framework for automatic segment level analysis of call transcripts. We show that unsupervised segmentation of calls is a plausible thing to do, and good accuracies can be attained even without using supervision.

As a possible extension to the present work, it is interesting to investigate the reusability of the proposed segmentation technique and it's adaptability across widely varying processes. Our technique uses a neighborhood clustering mechanism wherein the chosen neighborhood value is sensitive to the average size of the segments. Larger values for neighborhood yield larger segments although the vice versa is not true. One may need to devise techniques to automatically determine the value for the neighborhood using the corpus of calls and the domain information provided.

References

[1] J. Allan. *Introduction to Topic Detection and Tracking.* Kluwer Academic Publishers, MA, USA, 2002.

[2] D. Beeferman, A. Berger, and J. Lafferty. Statistical models for text segmentation. *Machine Learning*, 34:177–210, 1999.

[3] A. Clark. Pre-processing very noisy text. In *Workshop on Shallow Parsing of Large Corpora*, 2003.

[4] P. Deepak and K. Kummamuru. Mining conversational text for procedures. In *IJCAI-2007 Workshop on Analytics for Noisy Unstructured Text Data, Hyderabad, India, January 8, 2007*, 2007.

[5] A. Gilman, B. Narayanan, and S. Paul. Mining call center dialog data. In *Data Mining 5 (Information and Communication Technologies volume 33)*. Wessex Institute of Technology Press, 2004.

[6] M. A. Hearst. Multi-paragraph segmentation of expository text. In *Annual Meeting of the Association for Computer Linguistics (ACL)*, pages 9–16. Association for Computer Linguistics, 1994.

[7] G. Karypis. *CLUTO - A Clustering Toolkit.* Technical Report 02-017, Dept. of Computer Science, University of Minnesota, 2002.

[8] H. Kozima. Text segmentation based on similarity between words. In *Annual Meeting of the Association*

for Computer Linguistics (ACL), Columbus, OH, USA, 1993. Association for Computer Linguistics.

[9] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In *Fifth Symposium of Maths, Statistics and Probability, Berkeley*, 1967.

[10] G. Mishne, D. Carmel, R. Hoory, A. Roytman, and A. Soffer. Automatic analysis of call-center conversations. In *Intl. Conference on Information and Knowledge Management (CIKM)*, 2005.

[11] F. Murtagh. A survey of recent advances in hierarchical clustering algorithms. *Comput. J.*, 26(4):354–359, 1983.

[12] G. Ramakrishnan. *Bridging Chasms in Text Mining Using Word and Entity Associations.* PhD thesis, Indian Institute of Technology, Bombay, 2005.

[13] J. C. Reynar. An automatic method of finding topic boundaries. In *Annual Meeting of the Association for Computer Linguistics (ACL)*, pages 331–333. Association for Computer Linguistics, 1994.

[14] K. Ries. 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.

[15] S. Roy and L. V. Subramaniam. Automatic generation of domain models for call-centers from noisy transcriptions. In *Annual Meeting of the Association for Computer Linguistics (ACL)*. Association for Computer Linguistics, 2006.

[16] G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Commun. ACM*, 18(11):613–620, 1975.

[17] J. P. Yamron, I. Carp, L. Gillick, S. Lowe, and P. van Mulbregt. A hidden markov model based approach to text segmentation and event tracking. In *IEEE Intl. Conference on Acoustics, Speech and Signal Processing*, 1998.

[18] G. Zweig, O. Siohan, G. Saon, B. Ramabhadran, D. Povey, L. Mangu, and B. Kingsbury. Automated quality monitoring in the call center with asr and maximum entropy. In *IEEE Intl. Conference on Acoustics, Speech and Signal Processing*, 2006.

A Support-based Segmentation

Support-based segmentation is an alternate approach to find segments using the set of sequences SPTSs derived from the original corpus of calls. In this approach, *support* of a segment $(s_1,s_2,\ldots,s_l)$ is computed as the number of times the sequence of states have appeared in the corpus of calls. Given the corpus of calls, we follow the following 3 steps:

1. Identify SPTS clusters using the algorithm mentioned in Section 3.2,
2. Represent each call as sequence of SPTSs, and
3. Compute support of each SPTS subsequence.

Once the supports are calculated, one obvious good segmentation would be the one which has maximum total support across all possible segmentations. *Total support* is computed as sum of supports of constituent segments. The rationale behind this is the fact that the states (or sentences) within a segment would be occurring in sequence more often than states (or sentences) across segments and thus would have higher support. As a sequence of n consecutive elements is always present in the sequence of $n+1$ elements (those n elements plus the next element), support of a sequence is multiplied with its length to nullify the advantage enjoyed by short segments. Given a call, total support for all possible segmentations are computed and the one with maximum total support is identified. It may be noted that this is a computationally expensive task.

B Properties of the SCE Measure

The SCE measure quantifies the scatter of calls in a clustering. But, other properties of a clustering also affect the scatter of calls. In this section, we discuss the interplay between SCE measure and other properties of a clustering, especially, the number of clusters in a clustering. 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 data elements, SCE 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.

SCE is parameterized by both the clustering and the corpus used. Thus, one could compare clustering algorithms using their SCE values on different corpora, and different corpora (for comparing homogeneity) by applying the clustering algorithm. However the characteristics of the SCE measure outlined above require that the clusterings to be compared have approximately equal values for the following ratios

- Sentences Ratio: Number of turns per Cluster
- Call Length Ratio: Average Length of Calls per Cluster