Identification of Class Specific Discourse Patterns

Anup Kumar Chalamalla
IBM India Research Lab
ISID Campus, Vasant Kunj
New Delhi, 110070
achalama@in.ibm.com

Sumit Negi
IBM India Research Lab
ISID Campus, Vasant Kunj
New Delhi, 110070
sumitneg@in.ibm.com

L. Venkata Subramaniam
IBM India Research Lab
ISID Campus, Vasant Kunj
New Delhi, 110070
lvsubram@in.ibm.com

Ganesh Ramakrishnan
IBM India Research Lab
ISID Campus, Vasant Kunj
New Delhi, 110070
ganramkr@in.ibm.com

ABSTRACT

In this paper we address the problem of extracting important (and unimportant) discourse patterns from call center conversations. Call centers provide dialog based calling-in support for customers to address their queries, requests and complaints. A Call center is the direct interface between an organization and its customers and it is important to capture the voice-of-customer by gathering insights into the customer experience. We have observed that the calls received at a call center contain segments within them that follow specific patterns that are typical of the issue being addressed in the call. We present methods to extract such patterns from the calls. We show that by aggregating over a few hundred calls, specific discourse patterns begin to emerge for each class of calls. Further, we show that such discourse patterns are useful for classifying calls and for identifying parts of the calls that provide insights into customer behaviour.

Categories and Subject Descriptors

H.2.8 [**Database Applications**]: Text Mining, Knowledge Management Applications

; H.4.0 [**Information Systems Applications**]: Miscellaneous

General Terms

Knowledge Discovery, Text Mining

Keywords

Call Center Analytics and Applications, Text Mining, Classification and Clustering, Unsupervised Learning, Information Extraction

1. INTRODUCTION

Many companies today maintain call centers to present a single point of contact to their customers. At these call centers, customers interact with professional agents who address their queries, requests and complaints. Call centers handle hundreds of calls depending on the nature of the business. They range from technical support (help desk, customer care) to promotional (marketing, sales) to transactional (booking, rental).

There is a wealth of information hidden in the calls that could be useful to the organizations. Text analytics can play an important role in performing deep and insightful analysis of conversational transcripts. In this paper we address the problem of extracting important (and unimportant) discourse patterns from call center conversations. We show that by aggregating over a few hundred calls, specific discourse patterns begin to emerge for each class of calls. We also demonstrate that these discourse patterns also serve as useful features for call classification and clustering required in the tasks of call routing, obtaining call log summaries, agent assisting and monitoring, automatic domain model generation, system evaluation and modeling, business insight generation, *etc.*

Today's call centers handle a wide variety of domains such as computer sales and support, mobile phones, car rentals, apparels, and so on. It has been observed that within a domain, or within an instance of a domain, the interactions in a contact center follow specific, repetitive patterns. This is mainly because of the similar nature of the queries, requests, and complaints received from customers. For example, in a call center that handles car bookings, the call flow remains unchanged between calls. The agent starts out by introducing herself, gathers the customer's needs, suggests possible car options and finally makes a booking if the customer is satisfied. So a lot of phrases and discourse patterns get repeated.

We exploit this repetitive nature of the calls to extract key discourse patterns within each class of calls. We show that such patterns are class specific and identifying them result in useful business knowledge as well as extremely useful features for call classification and clustering. We argue that patterns consisting of sequences of non-consecutive phrases capturing contextual correlations are vital features for information extraction from natural language text. We show that intra sentence (phrasal) and inter sentence (discourse) long range patterns are present in the calls. We discuss methods to extract these discourse patterns that capture key knowledge about the calls.

But what are the patterns of interest and what is the value of extracting them? A snippet of an example interaction between a customer and an agent is given in Figure 1. The greeting segment and the conversation relating to the agent asking for the pick up and car

```
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.
CUST: Aah ok I need it from SFO.
AGENT: For what date and time.
CUST: I want to pick it up from SFO on August 3
and drop it in LA on August 6th.
.........
.........
AGENT: Maam so I have a 15 seater van available
for you at 300.58$.  This is with Taxes with
surcharges and with free unlimited mileage.
CUST: oh this quote seems to be on the higher
side.  Do you have a better rate for me
AGENT: Well maam are you a member of auto club?
.........
.........
AGENT: Thank you for calling CarCompanyA and you
have a great day good bye
```

Figure 1: Snippet of an Example Interaction

details are repeated across most of the calls. Consequently, this discourse pattern is present in most calls and is possibly uninteresting. However, the discourse relating to the agent presenting the rate and the customer raising an objection to it would not be present in all the calls. In this case it is also interesting to capture how the agent overcomes the objection to make the sale. Thus, while conversations common to all calls may be uninteresting, specific portions of conversations that differ from others provide valuable knowledge. Identifying such discourse patterns helps in capturing specific customer objections and agent best practices for handling customer objections. Also this serves as a summary of the hundreds of calls to a call center. Breaking up calls into familiar portions that are common to all calls and portions typical to a specific class of calls, allows for easier analysis. It is important to capture the business knowledge present in the calls as it allows the management to understand reasons for sale (no sale), evolve better agent training methods, understand customer needs, and so on.

2. BACKGROUND AND RELATED WORK

Call center analytics is a relatively new area. There is need to analyze huge amounts of customer agent interactions to derive deeper insight into the business processes, customer needs and agent capabilities. In this section we present some of the work we have built upon and extended in this paper.

Call Center Analytics: Call transcripts have been analyzed for topic classification [6], quality classification [20] and for estimating domain specific importance of call fragments [14]. It has also been shown that useful business intelligence can be obtained from customer agent conversations [16].

Extraction of Discourse Patterns: A call center conversation typically proceeds in the form of questions and answers. In a process like car rental booking, the questions are mostly asked by the agent as the call is agent driven in this case. The agent asks questions like '*what is the pick up location*', '*what is the pick up date*', *etc.* The first task in learning the discourse model of conversations of call center data is to identify the questions and their answers accurately. Question answer pairs can be identified in emails using lexical similarity and based on writing styles [15]. Identifying questions in conversations is difficult because features such as question marks are absent in spoken language. We use certain keyword based methods to identify questions. Identifying useful discourse patterns in conversations is typically based on clustering speaker turns [13]. Question-answer extraction and clustering them based on speaker turns helps in finding discourse patterns effectively.

Automatic Call-type Classification: A lot of work on automatic call type classification for the purpose of call routing ([10], [7]), obtaining call log summaries [5], agent assisting and monitoring [11] has appeared in the past. In most cases, authors have modeled this as a text classification problem. These approaches rely on finding key phrases, which are used as features. For manually transcribed calls, which do not have any noise, [11] a phrase level significance estimate is obtained by combining word level estimates that were computed by comparing the frequency of a word in a domain-specific corpus to its frequency in an open-domain corpus. In [18], phrase level significance was obtained for noisy transcribed data where the phrases are clustered and combined into finite state machines. Other approaches use n-gram features with stop word removal and minimum support ([10] [5]).

Unsupervised Clustering of Calls: Clustering call records for automatic domain model generation [14], system evaluation and modelling [2] and business insight generation is fairly common in literature related to Call-Center Analytics. Call centers typically handle queries from various domains such as computer sales and support, billing, car rental, etc. Each such domain generally has a domain model which contains common problem categories, typical customer issues and their solutions. These domain models, which are essential to handle customer complaints, are manually created over time. In the work [14], they propose an unsupervised technique based on call-record clustering to generate domain models automatically from call transcriptions. The TAKMI (Text Analysis and Knowledge Mining) [12] project, which has been successfully applied in the Call-Center domain to derive valuable business insights from call transcripts/logs, relies heavily on the quality of the call-clustering.

2.1 Our Contribution:

We present a method to identify useful discourse patterns by exploiting the redundancy in call center conversations. To do this, we first identify questions in conversations using a rule-based method (Section 3). Next, we cluster the questions using features that are frequently occurring patterns of non-consecutive words and named entities (henceforth called *phrasal* or *horizontal patterns*) extracted from the question collection. Section 4 is dedicated to discussing the details of the algorithm employed for determining frequent generic patterns of non-consecutive items. The motivation for named entity identification and an overview of the technique employed by us for this task is discussed in Section 5.1. It will be pointed out later, that unigram and n-gram features are just special cases of the more generic patterns of non-consecutive words. Clustering of questions gives us a way of canonically representing each question using the corresponding unique cluster label (or identifier). Based on this representation of questions, we mine frequent discourse patterns in the form of sequences of non-consecutive question cluster labels, named-entity annotations, and content words in the utterances (henceforth called *discourse* or *vertical patterns*). This process again makes use of the algorithm discussed in Section 4. Figure 2 illustrates the *phrasal* and *discourse patterns*. Note that the question cluster labels are assigned arbitrarily in the figure.

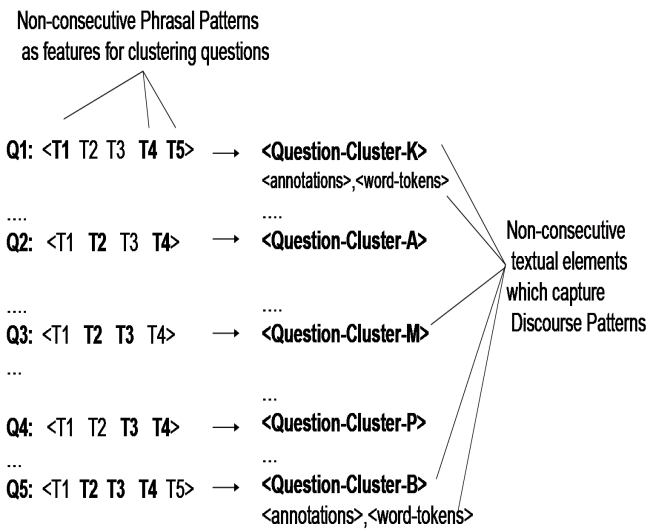

Figure 2: Phrasal and Discourse Patterns

Experimental results (Section 7) demonstrate three advantages of discovering the generic discourse patterns:

1. These discourse patterns, when employed as binary features in call classification drastically improve the quality of classification over simple unigram, and n-gram features (Section 7.2).
2. These discourse patterns can also be effectively used as binary features for call clustering to give reduction in entropy as well as an increase in purity over unigram and n-gram features (Section 7.3).
3. We show that the discourse patterns that we extract contain useful business knowledge (Section 7.4).

Finally, in Section 8, we present our conclusions and scope for future work.

3. EXTRACTION OF QUESTIONS

Most business-oriented call center conversations proceed more or less in an inquisitive fashion unlike other interactive dialogs. Both parties, *i.e.*, the agent and the customer, exchange information in the form of asking and answering questions relevant to completing the task at hand. Hence the discourse fragments in a dialog are generally centered around questions. Consequently, identifying questions becomes important. Here we describe a naive approach to question extraction from call center conversation data.

Unlike normal question answers from webpages like FAQs, *etc.*, identifying questions and answers in speech data is not straight forward. Punctuations are often missing and case information is not present. Also informal language is used by the speakers in expressing their inquiries. For example, agents may ask *your last name, please?* instead of asking the same question in a more formal manner. There can also be questions in declarative form, e.g. a customer says *I would like to inquire if I can get a 12 passenger car for rent.* The agents are usually trained to ask questions in an interrogative form. So agent questions are easier to extract.

have you rented from **us** in the **past**
have u rented a car from **us before**
have you rented with <**a_rental_agency**> **before**

Figure 3: Similar sentences

The crux of any question extraction system is to identify the asking point (how, what, when, where, etc.) in the text. Our question extraction method uses a similar approach, except for the fact that our *a*sking point dictionary consists of not only key phrases (how, what, when, have you, what's your etc.) but also special phrases which are specific to the domain. Examples of domain specific key phrases include *which location, which car, for how long*. Any utterance that begins with a key phrase would be recognized as a question. To extract queries which do not contain an asking point, we use an additional set of words which indicate that a sentence could be a question. *E.g.*, words such as, '*inquire*', '*inquiry*', '*please*' in a normal sentence indicate a question. This method performs quite well in extracting questions from the data we have experimented with (see Section 7.2).

4. EXTRACTION OF FREQUENT ITEM SEQUENCES

In this section we describe a variation of apriori algorithm for association rule mining applied on natural language text. This algorithm is the backbone of our method for extracting discourse patterns. Patterns that capture non-consecutive item (items could be words, entities, canonical question labels, *etc.*) sequences are important for natural language text. Though we will discuss the approach in the context of mining frequent word sequences from sentences (*horizontal patterns*), the approach applies equally well to the mining of frequent *vertical patterns* consisting of question labels, named entities and content words in utterances from calls.

Natural language expressions allow inclusion of complex modifiers. The intervening modifiers improve the richness of expressions and are thus important to natural languages. However, these variations make certain highly correlated words non-consecutive. Figure 3, presents a set of example sentences. All are questions asking the same thing. However, they differ on the surface due to the use of modifiers and other intervening words. Patterns involving non-consecutive word and entity sequences are required as features or clues to detect "similar" questions and "similar" answers, so that they can be respectively grouped into clusters representative of their types.

We implemented an efficient algorithm described in [8] (an extension of the apriori algorithm described in [1]) for finding frequent patterns of non-consecutive tokens. The $minSup$ value in the apriori corresponds to the minimum number of times a non-consecutive n-gram should occur across all the sentences. So, the most frequent non-consecutive n-grams which exceed the threshold value $minSup$ are output by the system. Among all the non-consecutive patterns that can be extracted, the system generates the longest sequences. For example, from among the patterns #rented#car#before# and #rented#car# and #rented#before#, the system considers #rented#car#before# and leaves out the other two. While generating such patterns over text sentences we consider only the content words. Two parameters which control the generation of these patterns are the minimum threshold value ($minSup$) and the maximum token gap (δ) with which non-consecutive n-grams needs to be considered.

<company>**XYZ Ltd**</company> paid an advance of <currency>**10 mln dlrs**</currency> to <company>**ABC Inc.**</company> toward purchase of the latter's drilling unit. <company>**PQR News**</company> had announced <company>**XYZ**</company>'s acquisition plan in <date>**June 2005**</date>.

Figure 4: A sample annotated document D

We use the non-consecutive pattern mining approach in two places in our method, one to cluster the sentences based on non-consecutive N-grams, and other to generate discourse patterns over the entire conversations. We will reproduce here, some details of the pattern mining technique from [8] to set the context for the discussions that follow.

4.1 Algorithm for Extracting Frequent Patterns

A pattern can be equivalently represented as a *token-sequence*. For example, the pattern '<*company*>.* paid.* <*monetary amount*>.* to.* <*company*>' can be represented as the following token sequence: '[<*company*>, paid, <*monetary amount*>, to, *company*]'. We will denote a token-sequence comprising n tokens by s_n.

DEFINITION 1. *A document d is said to contain an instance of token-sequence s_n; $d \sqsupseteq s_n$;iff all tokens in s_n appear in d in the same order, with a fixed upper bound, δ, on the number of intervening words between every pair of successive tokens in s_n. Note that patterns with $\delta = 0$ correspond to the commonly used $n-$gram features.*

Let $\delta = 4$. The document D in Figure 4 has an instance of '[<company>, paid, <currency>]'. On the other hand, D does not have any instance of '[toward, unit]', because there are more than δ intervening words. Similarly, D does not have any instance of '[<currency>, toward]', since there is an intervening occurrence of 'company' type.

Let $\mathcal{D}$ be a set of training documents of size $|\mathcal{D}|$. $freq(s_n) = |\{d|d \in \mathcal{D}, d \sqsupseteq s_n\}|$ is the number of documents in $\mathcal{D}$ that contain instances of s_n. $freq(s_n)$ can also be counted at a granularity finer than document, such as at the sentence level. However, $freq(s_n)$ considers overlapping occurrences of s_n to be a single occurrence[1].

We define the support of s_n as $sup(s_n) = \frac{freq(s_n)}{|\mathcal{D}|}$. Let $minSup$ be a threshold on support. We define $\mathcal{S}_n$ as the set of all token sequence of length n that have support greater than or equal to the minimum support $minSup$, *i.e.*, $\mathcal{S}_n = \{s_n \,|sup(s_n) \geq minSup\}$. Let $\mathcal{S}_*$ be the set of token-sequences s_* such that each s_* has length between 1 and a threshold N and $sup(s_*)$ is above the threshold $minSup$. $\mathcal{S}_*$ is identified using the algorithm outlined in Figure 5. The algorithm is inspired by the a-priori [1] algorithm. The a-priori algorithm optimally and efficiently yields all item-sets or item-sequences, that have their support value above a given threshold value. We next prove that the algorithm given in Figure 5 optimally discovers all non-consecutive token sequences.

DEFINITION 2. *Let s_i and s_j be two token sequences where $j < i$. We say $s_i \supset s_j$; iff s_j is a contiguous subsequence of s_i.*

[1]If overlapping occurrences of a token sequence are considered as multiple occurrences, the monotonicity property of $freq(s_n)$ will not hold [19].

E.g. Token sequences $<t_1, t_2>$ and $<t_2, t_3, t_4>$ are contiguous subsequences of $s_4 = <t_1, t_2, t_3, t_4>$. On the other hand, $<t_1, t_3, t_4>$ is not a contiguous subsequence of s_4. Note that every document that has an instance of a sequence also has an instance of each of its contiguous subsequences, *i.e.* $\forall d \in \mathcal{D}, d \sqsupseteq s_n \Rightarrow \forall s_n \supset s_i, d \sqsupseteq s_i$. This implies $\forall s_n \supset s_i, sup(s_n) \leq sup(s_i)$.

THEOREM 1. *If $s_n \in \mathcal{S}_*$, then $\forall s_i | s_n \supset s_i, s_i \in \mathcal{S}_*$.*

Proof: We prove the theorem by contradiction. Let there be an $s_i, i < n$, *s.t.* $s_n \supset s_i$ and $s_i \notin \mathcal{S}_*$. This implies $sup(s_i) < minSup$. However, $sup(s_n) \leq sup(s_i)$. Therefore, $sup(s_n) < minSup$ which implies $s_n \notin \mathcal{S}_*$. This is a contradiction.

Using the result of Theorem 1, we iteratively build token-sequences of length $n+1$ from token sequences of length n.

Find $\mathcal{S}_1$, the set of all 1-item-sequences; $n = 1$
$\mathcal{S}_* = \{\}$
while $n \leq N$ **do**
  $\mathcal{S}_{n+1} = \{\}$
  for Each $s_n, s'_n \in \mathcal{S}_n$ **do**
    if s_n and s'_n have a subsequence of length $n-1$ in common **then**
      Merge s_n and s'_n to obtain s_{n+1}
      if $sup(s_{n+1}) \geq minSup$ **then**
        $\mathcal{S}_{n+1} = \mathcal{S}_{n+1} \cup \{s_{n+1}\}$
      end if
    end if
  end for
  for Each $s_n \in \mathcal{S}_n$ **do**
    if $\neg(\exists s_{n+1} \in \mathcal{S}_{n+1} | s_{n+1} \supset s_n)$ **then**
      $\mathcal{S}_* = \mathcal{S}_* \cup \{s_n\}$
    end if
  end for
  $n = n + 1$
end while

Figure 5: The algorithm for generating the set $\mathcal{S}_*$ of token-sequences with high support

The following corollary is important, as it implies that a set of patterns obtained with particular values of δ and N subsumes all sets obtained using lower values of the respective parameters.

COROLLARY 1. *Let $S_*(\delta, N, minSup)$ be the set of all non consecutive token sequences up to a length of N such that a maximum of δ intervening words are permitted between any two successive tokens. Let the minimum support count be $minSup$. Then, $\forall N' \leq N, \delta' \leq \delta, minSup' \geq minSup$, we have $S_*(\delta', N', minSup') \subseteq S_*(\delta, N, minSup)$.*

Corollary 1 implies that larger values of δ and N and smaller values of $minSup$ should be preferred. There is however a trade off; large values of N and δ and small values of $minSup$ can result in a large number of patterns which might not be very significant. Moreover, the time required for mining patterns grows almost exponentially with increasing values of δ and N and decreasing values of $minSup$. Given this trade off, we experimented with sample data to decide on reasonable values of these parameters. These values will be reported in the experimental section.

5. IDENTIFYING PHRASAL PATTERNS

In this section we describe how we extract the sentence level patterns and use the patterns for clustering questions.

5.1 Named Entity Annotation

In this section we show how the task of named entity annotation is performed and why it is important in the perspective of extracting discourse patterns. We assert that named entity annotation improves the quality of *phrasal patterns*, thereby improving the quality of clustering of sentences based on these features. Effective sentence clustering in turn improves the extraction of *discourse patterns* from calls. Sequences of word tokens annotated as named entities are represented using the canonical form of their named entity types, so that the text segments which differ only in the named entity instances can be easily grouped together. As an example, the two sentences '*are you picking up the car at Cleveland?*' and '*are you picking up the car at Orlando?*' will look similar after annotation as '*are you picking up the car at LOCATION_CITY*'. Thus, named entity annotation is an important preprocessing step in generating discourse patterns.

We handle a few domain specific named-entities that are most frequently used in particular type of conversations, for example in a car rental process named-entities such as, CAR-MAKE (Chevrolet, Toyota, Mercedes), CAR-SIZE (mid-size, full-size, mini van), VEHICLE-TYPE (car, van, suv), LOCATION-NAME (e.g. *Chicago, Cleveland, Orlando, Los Angeles*), DATE and TIME (e.g., *February 28th, monday the 28th of feb, morning of 28th, 10 o clock, 10 p.m.*), AMOUNT (e.g. *$320.0, 344 dollars and 35 cents, 320.45 dollars*), CAR-TYPE (e.g. *luxury car, economy car*), DISCOUNT-TYPE (e.g. *AAA discount, military discount, sams club, AARP discount*) are very common.

We used a rule-based named entity annotator which has been developed in-house for annotating unstructured text. The annotator is written using Apache's open source annotator framework UIMA[2] (Unstructured Information Management Architecture). The working of the annotator is as follows. The input text is first passed through a sentence chunker which identifies sentence boundaries. Each sentence is tokenized based on a set of token-separator characters (*e.g., space,tab,-,*, etc.*). Tokens are then tagged with their dictionary attributes based on dictionary lookups.

Regular expressions over tokens and their dictionary and orthographic properties have been extensively used for named entity annotations. The Common Pattern Specification Language (CSPL)[3] specifies a standard for describing Annotators that can be implemented by a series of cascading regular expression matches. Our rules for entity identification were composed using a subset of CPSL and are similar to the syntax of rules used for named entity annotations in GATE [3]. The GATE architecture for text engineering uses the Java Annotations Pattern Engine (JAPE) [4] for its information extraction task. JAPE is a pattern matching language. Our rules support two classes of properties for tokens that are required by grammars such as JAPE: (1) orthographic properties such as an uppercase character followed by lower case characters, and (2) gazetteer (dictionary) containment properties of tokens and token sequences such as 'location' and 'person name'. The algorithm in the core annotator engine is independent of rules and dictionaries and it is very helpful in annotating different kinds of entities such as EMail, URL, Dates, Times, *etc*, provided the corresponding rules and dictionaries are framed.

[2] `http://incubator.apache.org/uima/`
[3] `http://www.ai.sri.com/~appelt/TextPro`

have you rented from **us** in the <T>past</T>
have you rented a car from **us** <T>before</T>
Extracted Feature: have#you#rented#us#TIME

Figure 6: Annotations Help Clustering

```
* Can I know the cost of a <CARTYPE>standard
size car</CARTYPE>?
* How much does it cost from
<PLACE>Richmond</PLACE> to
<PLACE>Cleveland</PLACE>?
```

Figure 7: Another Example Illustrating Importance of Annotations

5.2 Clustering Questions using Phrasal Patterns

In this step we try to capture questions that have similar meaning but different surface forms. In call centers the same questions get asked again and again in slightly different forms across calls. So for example, the questions in Figure 6, need to be identified as questions that have similar meaning. For this we perform clustering on the collection of all questions, using phrasal patterns (mined from the question collection) as features. It is necessary to cluster the questions across the conversations in an unsupervised way. A bag of words approach does not work because the vocabulary for a given call center task is limited. In a car rental task, for example, there are less than 500 content words. The difference is in how the words come together in a given sentence which changes what is being said. So, '*do you have a valid credit card*', '*do you have a valid aaa member card*' and '*do you have your credit card please tell me the number*' may be difficult to distinguish using bag of words. Also approaches based on consecutive n-grams fail because of the complex modifiers that natural language gives.

To make clustering most effective we annotate the questions. We observed that annotations result in better phrasal patterns. This is shown in the example in Figure 6. In this example we see that by annotating *past* and *before* as TIME the extracted feature's count across all sentences increases and makes it an important feature for clustering.

The essence of clustering is to increase the distance between dissimilar sentences and decrease the distance between similar sentences. Hence, it is also imperative to avoid dissimilar sentences getting clustered together based on words alone. Figure 7 shows examples of car type and place annotations. Such an annotation reduces the first question to '*Can I know the cost of a CARTYPE*' and second question to '*How much does it cost from PLACE to PLACE*'. So the word '*cost*' even though common in both sentences, because of the presence of canonical tokens like CARTYPE and PLACE the clustering algorithm performs well in grouping first sentence separately from the second sentence. Every question is now represented using the following features for clustering, *viz.*, (i) words and (ii) phrasal patterns. In generating the phrasal patterns using apriori, we take a small value for the token gap parameter and high value for minimum support count, the K value parameter. This is understandable since the sentences are small and we need good representative features for clustering. Next, the questions are clustered using K-Means clustering algorithm. After extracting and cluster-

```
AGENT: Sir how may I help you?
<#GreetingQuestion#>
CUST: Can I know the cost of a <CARTYPE>standard
size car</CARTYPE>?  <#CostQuestion#>
AGENT: Sir which city would you need the car in?
<#LocationQuestion#>
```

Figure 8: Example of Sentence Annotation

ing the questions we represent each occurrence of a question in a call by its canonical cluster label as determined by the clustering algorithm. An example of a call with labels assigned to its contents is shown in Figure 8[4].

6. IDENTIFYING DISCOURSE PATTERNS

Extracting discourse patterns involves searching for frequently occurring discourse fragments in the conversations. We define what is called discourse features in a conversation as text with a sequence containing content words, named-entities canonicalized by their types, questions canonicalized by their cluster-labels. The task is to extract the *discourse patterns* from all conversations. To perform this task, we apply the algorithm we described in Section 4 over a larger sequence of tokens comprising the whole conversation text with canonicalizations. Refer to Figure 2 which illustrates this pictorially.

As shown in Figure 8, many calls have #GreetingQuestion# followed by #CostQuestion# from a customer, which in turn is followed by #LocationQuestion# from the agent. This is an example of a frequent discourse pattern. The advantage of using non-consecutive pattern mining is that it makes our extraction robust to noise. In spontaneous conversations people do not follow a rigid format. In the above example the agent could say '*can you wait for a second sir*' before the #LocationQuestion#. Additionally, since there are many different irrelevant utterances like '*give me a moment sir*', used in conversations, we need to pick the most relevant sequences which capture the discourse information. Non-consecutive pattern mining allows us to handle all these variations. We choose a large value for the token gap parameter δ. This choice can be justified, since we had already shown that there can be irrelevant discourse fragments in between some important ones. Hence we need to relax the value of token gap to capture sequences of important discourse fragments separated by several other texts in between. In the following sections we illustrate how these extracted discourse patterns can be used as features for better classification of calls in call center domain. Also we explained some class specific discourse patterns which emerge as top features through classification.

7. EXPERIMENTS AND RESULTS

In this section, we present the experimental study of our technique. We start by describing the experimental setup and the data set. To bring out the value of the discourse patterns in call center conversations we present three sets of results. First, we show that the extracted discourse patterns when used for call classification result in improved performance compared to bag of words, unigram, bigram and trigram based techniques. This alludes to the fact that there are class specific discourse patterns present in the calls. Next we show unsupervised clustering of call-records using the discourse features which resulted in better entropy compared to using uni-grams and bi-grams. Next we show that we are indeed able to extract the key discourse patterns for a given class.

[4]For simplicity and effectiveness of understanding in the paper we manually renamed some arbitrary question-cluster-labels given by clustering tool weka to meaningful names in the figure 8.

7.1 Data and Experimental Setup

We collected 935 calls from a car rental help desk. We obtained automatic transcriptions of the dialogs using an Automatic Speech Recognition (ASR) system. The transcription server, used for transcribing the call center data, is an IBM research prototype. The speech recognition system was trained on 300 hours of data comprising of call center calls sampled at 8KHz. These calls were of three types, *viz.*, calls that resulted in booking (*booked*), calls that did not result in booking (*unbooked*) and service calls where customers were seeking information and not trying to make a booking (*service*). The calls that did not result in a booking (*unbooked*) were further divided into sub classes based on the reason for their not resulting in a successful booking. A call can become unsuccessful when the agent and customer do not reach common terms based on the customer's requirement. Specifically, we concentrated on the classes "rates too high", "unavailability of car" and "not meeting requirements". The first class corresponds to customers not making bookings because they thought the rate being quoted by the agent was too high. In the second, the car being asked by the customer was not available. And in the third, the customer did not meet one or more requirements for renting a car, such as payment option requirements, driving license requirements, *etc.*

The question extraction step extracted around 7000 questions from the data of which around 5000 are questions asked by the agent and 2000 are questions asked by the customer from a total of 935 conversations. By randomly sampling some calls, we estimated that the average total number of questions asked by the customer and agent within a "booked" conversation is 10, within an "unbooked" conversation is 8, and within a "service" conversation is 5. Based on these sampled numbers, we estimated that there should be around 7152 questions in the entire data. On the sampled calls, our question extraction method yielded precision, recall and F1 measures tabulated in Table 1. The definitions for precision, recall and F1-measure are given below.

$$Recall = \frac{\text{Number of instances correctly classified in a class}}{\text{Total number of instances expected in the class}} \quad (1)$$

$$Precision = \frac{\text{Number of instances correctly classified in a class}}{\text{Total number of instances classified in the class}} \quad (2)$$

$$F1\ Measure = \frac{2}{\frac{1}{Recall} + \frac{1}{Precision}} \quad (3)$$

Precision	Recall	F-Measure	Class
0.93	0.88	0.90	booked
0.91	0.94	0.92	unbooked
0.93	1.00	0.97	service

Table 1:

As can be seen from Table 1, the F1 measure for each category is above 0.9. Recall from Section 3, that question extraction is

not an end in itself, but is only a technique used for feature construction for the tasks of finding frequent discourse patterns, call classification and clustering. We report the representative numbers in Table 1 only to give an idea about the quality of our question extraction technique. The phrasal patterns for questions were derived by employing the algorithm (Figure 5) explained in Section 4, with the following parameter settings: (maximum token gap) $\delta = 5$, (minimum support threshold) $minSup = 5$ and (maximum number of items in a pattern) $N = 5$. We generated the question clusters for agents' and customers' questions separately using $k = 30$ in the k-means implementation in Weka [17]. We used the Weka toolkit [17] for clustering the questions. Subsequently, frequent discourse patterns were mined using the algorithm (Figure 5) explained in Section 4 with the following parameter settings: (maximum token gap) $\delta = 15$, (minimum support threshold) $minSup = 5$ and (maximum number of items in a pattern) $N = 7$.

7.2 Classification Using Discourse Patterns

We show that classification of call-records, which is the underlying theme in applications such as call routing, call log summary generation, agent assisting and monitoring can significantly benefit from the use of discourse patterns as proposed in our work. The following tables show the results obtained on classification of the calls into the three classes discussed previously. We randomly split the data into test and train set. 80% of the data is used for training the classifier and the rest 20% is used for testing. Results are reported as averages over 5 random train-test splits. We illustrate the results using two types of classification methods– the Naive-Bayes Classifier and Support Vector machines implemented in Weka [17]. We measure the results in terms of precision, recall and F1-measure of the classifier.

Classification of "unbooked" calls is important from the call center company point of view, for understanding the reasons behind calls being unsuccessful or ending abruptly. In this task we achieve significant improvement in the results when our approach is employed compared to classification using unigrams, bi-grams and tri-grams. Table 2 shows the results by using unigrams and bi-grams. Table 3 shows the results using a combination of unigrams, bi-grams and tri-grams. Tables 5 and 4 shows the results for our method; using the discourse patterns as features for classification along with unigrams and bi-grams.

We asserted in the beginning that there are discourse segments in the conversation which are common across all the calls such as the greeting part, the agent asking for details of car reservation, date and time of picking, etc. These are the "unimportant" discourse patterns that are identified by our algorithm. These are redundant as far as classification is concerned. Further, they occur the most number of times since they are present in all the calls irrespective of class. We can find the top-K most frequently occurring unimportant patterns and remove them, K value can is dependent on the data. This is known as feature selection. After doing this, the top distinguishing features in the classification (Table 5) begin to emerge; the precision, recall and F1-measure of the classes improved significantly as shown in the Table 5 compared to the results in Table 4. The summary of classification results is shown in 6. All these results are when using Naive Bayes classifier.

The results for support vector classification for the same data are given in the tables 7 and 8. Table 7 gives results for unigrams and bigram features using support vector classification. Table 8 gives results for the combination of unigram, bigram and discourse features with feature selection using support vector classification. We find that the precision, recall and F-measure using SVM are slightly better compared to those generated by Naive Bayes Classifier. With SVM, the precision, recall and F-measure on combination of unigrams, bigrams and discourse features with feature selection is much better compared to using unigrams with bigrams.

Precision	Recall	F-Measure	Class
0.48	0.65	0.55	rates
0.37	0.21	0.267	unavailability of car
0	0	0	not meeting requirements

Table 2: Classification Results with Unigram and Bigram Features

Precision	Recall	F-Measure	Class
0.48	0.65	0.55	rates
0.34	0.26	0.29	unavailability of car
0	0	0	not meeting requirements

Table 3: Classification Results with uni-grams, bi-grams, and tri-gram Features

Precision	Recall	F-Measure	Class
0.533	1	0.696	rates
1	0.273	0.429	unavailability of car
0	0	0	not meeting requirements

Table 4: Classification Results with uni-grams, bi-grams and discourse pattern Features

7.3 Unsupervised Clustering of Call-Records using Discourse Patterns

To validate the claim that clustering applications can significantly benefit from our method of call record representation using phrasal and discourse patterns, we conducted clustering experiments on a call-transcript corpus from the car-rental domain. The corpus contains 233 transcriptions from 4 business categories explained earlier. ("rates too high", "Unavailiabilty of cars", "not meeting requirements", "specific car requirements not met"). The clustering results are as shown in Figures 10 and 11. The clustering was performed using CLUTO [9] and evaluated using its entropy and purity functions. Cluster purity indicates the degree to which a cluster contains concepts from one class only (perfect purity would be 1). Cluster entropy indicates whether concepts of different classes are represented in the cluster (perfect entropy would be 0). The mathematical formulas for entropy and purity are given in the Figure 9. We find that the results on using discourse features are better compared to using the uni-grams and bi-grams (Figures 10 and 11). The summarized results are shown in 9.

7.4 Examples of Important Discourse Patterns

In this section we illustrate the class specific and unimportant discourse patterns extracted by our algorithm. Figure 12 shows an example of a discourse that is found commonly in all calls. This discourse fragment is an instance corresponding to the most common discourse pattern that our algorithm extracts from the data. The extracted pattern looks like this:
#GREETING-QUESTION#VEHICLE-QUERY#
PICKUP-DATE-QUERY#DATE-ENTITY#

Unigram+ bigram features	Unigram+bigram+ trigram features	Unigrams+discourse pattern features	Unigrams+filtered discourse pattern features	Category
0.55	0.55	0.696	**0.82**	rates
0.267	0.29	0.429	**0.71**	unavailability of car
0	0	0	**0.47**	not meeting requirements

Table 6: Summary of class-wise F1 results from tables 2 through 5. The best F1 for each class is reported in bold font; it can be easily seen that "*unigrams+filtered discourse pattern features*" give far better performance than the others.

Precision	Recall	F-Measure	Class
0.7	1	0.82	rates
0.857	0.6	0.71	unavailability of car
0.6	0.39	0.47	not meeting requirements

Table 5: Classification Results using Naive-Bayes Classifier with uni-grams, bi-grams and discourse pattern features after feature selection by removing unimportant discourse patterns

Precision	Recall	F-Measure	Class
0.7	1	0.82	rates
0.77	0.9	0.83	unavailability of car
1	0.222	0.364	not meeting requirements

Table 7: Classification Results with only Unigrams and Bigrams using Support Vector Machines

Precision	Recall	F-Measure	Class
0.774	1	0.873	rates
0.818	0.818	0.818	unavailability of car
1	0.444	0.615	not meeting requirements

Table 8: Classification Results using Support Vector Classification with uni-grams, bi-grams and discourse pattern features after feature selection

Entropy	Purity	Representation
0.782	0.536	uni-grams+bi-grams
0.441	**0.777**	Discourse pattern features+uni-grams+bi-grams

Table 9: Summary of clustering results from tables 10 and 11. Higher the purity or lower the entropy measure, better is the clustering. The best measures for each representation is reported in bold font; it can be easily seen that "*uni-grams+bi-grams+discourse pattern features*" give far better performance than just '*uni-grams+bi-grams*' .

Description	Entropy	Purity
Single Cluster	$E(S_r) = -\frac{1}{\log q}\sum_{i=1}^{q}\frac{n_r^i}{n_r}\log\frac{n_r^i}{n_r}$	$P(S_r) = \frac{1}{n_r}\max_i(n_r^i)$
Overall	$Entropy = \sum_{r=1}^{k}\frac{n_r}{n}E(S_r)$	$Purity = \sum_{r=1}^{k}\frac{n_r}{n}P(S_r)$

Figure 9: Entropy and Purity in CLUTO. S_r is a cluster, n_r is the size of the cluster, q is the number of classes, n_i^r is the number of concepts from the i^th class that were assigned to the r^th cluster, n is the number of concepts, and k is the number of clusters.

where tokens ending with "QUERY" are question clusters and DATE-ENTITY is the annotation for date (*3rd November*) [5]. This discourse pattern defines an instance of a discourse in which GREETING-QUESTION (e.g. *how may i help you*) of the agent is followed, with some token skips, by VEHICLE-QUERY (e.g. *do you have a 12 passenger van*) by customer, which is followed by PICKUP-DATE-QUERY (e.g. *what date and time you want to pick the car*) followed by the DATE-ENTITY. This is an example of an unimportant discourse pattern since it covers a common discourse found in most calls.

Now we show some important discourse patterns. Figure 13 shows an example of a discourse from the "Rates too high" call. The prominent class-specific discourse feature extracted by our algorithm is:
#AMOUNT-ENTITY#NAME-QUERY#better#price#
, where AMOUNT-ENTITY indicates the amount annotation of *$421.07* in the text, NAME-QUERY indicates query of agent asking for customer's name (e.g. *can i have your last name*) which is followed, with some skips, by two content words *better* and *price*. This indicates that the extracted feature covers the discourses from data which indicate the reason behind unbooked nature of the call because of customer not being satisfied with the price quoted by agent.

Figure 14 shows an example of a discourse from the "Unavailability of car" class. The pattern extracted here is:

[5]For simplicity and effectiveness of understanding in the paper we renamed the arbitrary question-cluster-labels generated by clustering tool weka, to meaningful names in the above representation of discourse feature

15-way clustering: [l2=8.51e+01] [233 of 233], **Entropy: 0.782, Purity: 0.536**

cid	Size	ISim	ISdev	ESim	ESdev	Entpy	Purty	rate	unav	not	spec
0	11	+0.232	+0.065	+0.033	+0.009	0.548	0.727	8	1	2	0
1	10	+0.157	+0.025	+0.032	+0.006	0.880	0.500	5	2	2	1
2	16	+0.155	+0.028	+0.034	+0.009	0.875	0.500	8	4	2	2
3	13	+0.143	+0.022	+0.031	+0.009	0.869	0.462	6	2	4	1
4	11	+0.142	+0.018	+0.035	+0.010	0.747	0.455	5	2	4	0
5	17	+0.138	+0.026	+0.033	+0.006	0.867	0.471	8	4	4	1
6	15	+0.142	+0.012	+0.037	+0.007	0.836	0.533	8	1	3	3
7	14	+0.131	+0.019	+0.030	+0.008	0.845	0.500	2	7	4	1
8	13	+0.124	+0.016	+0.028	+0.009	0.787	0.538	7	1	4	1
9	16	+0.131	+0.015	+0.041	+0.010	0.710	0.562	9	3	4	0
10	20	+0.121	+0.019	+0.035	+0.005	0.703	0.550	11	6	3	0
11	22	+0.126	+0.016	+0.039	+0.006	0.821	0.455	10	8	1	3
12	18	+0.120	+0.015	+0.038	+0.010	0.842	0.556	10	3	2	3
13	19	+0.104	+0.029	+0.026	+0.011	0.596	0.737	14	3	1	1
14	18	+0.108	+0.006	+0.036	+0.007	0.806	0.500	9	6	1	2

Figure 10: Clustering Results with text features (uni-grams and bi-grams)

15-way clustering: [l2=8.66e+01] [233 of 233], **Entropy: 0.441, Purity: 0.777**

cid	Size	ISim	ISdev	ESim	ESdev	Entpy	Purty	rate	not	unav	spec
0	10	+0.277	+0.058	+0.031	+0.010	0.234	0.900	0	9	0	1
1	14	+0.186	+0.050	+0.030	+0.009	0.716	0.500	2	5	7	0
2	10	+0.180	+0.023	+0.037	+0.006	0.441	0.700	3	7	0	0
3	11	+0.166	+0.023	+0.033	+0.010	0.655	0.636	2	0	7	2
4	14	+0.156	+0.025	+0.032	+0.006	0.367	0.857	1	0	12	1
5	11	+0.143	+0.022	+0.030	+0.008	0.746	0.636	2	1	7	1
6	14	+0.140	+0.015	+0.034	+0.008	0.367	0.857	12	1	1	0
7	18	+0.140	+0.024	+0.036	+0.007	0.307	0.889	16	1	1	0
8	19	+0.125	+0.024	+0.033	+0.006	0.315	0.842	16	3	0	0
9	17	+0.120	+0.018	+0.030	+0.009	0.437	0.706	0	5	12	0
10	22	+0.126	+0.019	+0.040	+0.008	0.542	0.773	17	3	1	1
11	15	+0.124	+0.012	+0.038	+0.005	0.177	0.933	14	0	1	0
12	17	+0.107	+0.029	+0.024	+0.010	0.692	0.588	0	3	4	10
13	23	+0.115	+0.018	+0.037	+0.008	0.339	0.870	20	1	0	2
14	18	+0.105	+0.016	+0.030	+0.006	0.402	0.833	15	2	0	1

Figure 11: Clustering Results with discourse patterns, uni-grams and bi-grams

#CARTYPE-QUERY#CARTYPE-ENTITY#not#available#, where, CARTYPE-ENTITY is annotation type for *economy car*, CARTYPE-QUERY indicates the agent's query *What type of car would you like to go for*.

Figure 15 shows an example of a discourse from a call belonging to "Not meeting requirements" class. The top distinguishing discourse feature extracted by our algorithm for this class is:
#CC-QUERY#RTT-QUERY#valid#credit#card#
where CC-QUERY indicates the credit card query (*e.g.*, '*Do you have a valid credit card*') and RTT-QUERY indicates the query '*do you have round trip travel ticket*'. A domain expert independently identified, by manually going through the calls, the discourse snippet of figure 15 as important to identify the reason for unsuccessful nature of the call in this class.

In this section we showed how some of the most common discourse patterns automatically extracted by our algorithm and classification cover relevant discourse fragments from the calls. These results also show that extraction of relevant class-specific discourse patterns has been made possible by accurate question extraction and clustering.

8. CONCLUSIONS AND FUTURE WORK

In this paper we proposed a method for automatic extraction of important discourse patterns from call center conversations using frequent sequences' pattern mining approach. We presented a unified approach for speech data mining by doing question-answer extraction, named entity recognition, extracting patterns of non-consecutive items (such as words, named entities, question labels, *etc.*), clustering and classification using these features. The extraction of the patterns of non-consecutive items is done along two dimensions, (i) on the words at the sentence level and other (ii) on the sequence of speech utterances in conversations. This enables us to extract important *discourse patterns* from the calls. In most

```
AGENT: My name is adrian, how may i help you
CUST: Do you have a 12 passenger van for rent
AGENT: on what date and time you want to pick
the car
CUST: I want it for 3rd November at 5 PM
.....
```

Figure 12: Snippet of a General Discourse Pattern

of the previous works [13] the generation of such patterns was performed on the whole transcript, without much discrimination between speech utterances. Because of the noise encountered in this form of data, the results on classification using discourse features have not been found to be very promising. After abstracting portions of text with question-clustering and named-entity-labeling we were able to achieve better class accuracy as well as specifically identify discourse patterns in the calls which are typical of an issue being addressed such as "rates being too high", "unavailability of car", "not meeting requirements".

In all the stages, care is taken such that our methods are domain independently applicable. For the question extraction phase, the domain specific key phrases dictionary is a plug in. This ensures that the question extraction technique is applicable to call center conversational data which proceeds mainly by questions and answers. Our methods though are not applicable to call conversations which are not significantly question-driven. But fortunately this method is still powerful enough for a huge class of contact center data. The annotator framework expects a set of dictionaries specific to the domain and a set of rules. Also the technique we employed is time-effective since it does not use parsing, natural language discourse analysis techniques and does not require large corpora for training.

9. REFERENCES

```
AGENT: The total cost including taxes and
surcharges comes for you at $421.07.  Should
i go ahead with the booking
CUST: OK was that 421
AGENT: yes 421.07.  can i have your last name.
so that i can lock this price for you.
CUST: Ahh I had a better price somewhere else.
But thank you.
```

Figure 13: Snippet of a “Rates too high" class

```
AGENT: What type of car would you like to go for
CUST: an economy car
AGENT: Oh economy car is not available for today
would you like to take a standard?
CUST: How about for tomorrow
AGENT: tomorrow also we are all sold out, would
you like a standard?
CUST: No, I am not interested ....
```

Figure 14: Snippet of an “Unavailability of Car" class

[1] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *VLDB*, 1994.

[2] F. Bechet, G. Riccardi, and D. Hakkani-Tur. Mining spoken dialogue corpora for system evaluation and modeling. In *EMNLP 2004: Empirical Methods in Natural Language Processing*, 2004.

[3] H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan. GATE: A framework and graphical development environment for robust NLP tools and applications. In *Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics*, 2002.

[4] H. Cunningham, D. Maynard, and V. Tablan. JAPE: a Java Annotation Patterns Engine (Second Edition). Research Memorandum CS–00–10, Department of Computer Science, University of Sheffield, Nov. 2000.

[5] S. Douglas, D. Agarwal, T. Alonso, R. Bell, M. Gilbert, D. Swayne, and C. Volinsky. Mining customer care dialogs for ’daily news’, 2005.

[6] A. Gilman, B. Narayanan, and S. Paul. Mining call center dialog data. In *Data Mining 5*, 2004.

[7] P. Haffner, G. Tur, and J. Wright. Optimizing svms for complex call classification. In *In Proceedings ICASSP’03, 2003.*, 2003.

[8] S. Joshi, G. Ramakrishnan, S. Balakrishnan, and A. Srinivasan. Information extraction using non-consecutive word sequences. In *Proceedings of TextLink 2007, The Twentieth International Joint Conference on Artificial Intelligence*, 2007.

[9] G. Karypis. CLUTO - a clustering toolkit. Technical Report #02-017, nov 2003.

[10] H.-K. J. Kuo and C.-H. Lee. Discriminative training in natural language call routing. In *In Proceedings of ICSLP-2000.*, 2000.

[11] G. Mishne, D. Carmel, R. Hoory, A. Roytman, and A. Soffer. Automatic analysis of call-center conversations. In *CIKM ’05: Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 453–459, New York, NY, USA, 2005. ACM.

[12] T. Nasukawa and T. Nagano. Text analysis and knowledge mining system. *IBM System Journal*, 40, 2001.

[13] D. Padmanabhan and K. Kummamuru. Mining conversational text for procedures with applications in contact centers. *International Journal on Document Analysis and Recognition*, 10(3-4):227–238, 2007.

[14] S. Roy and L. V. Subramaniam. Automatic generation of domain models for call-centers from noisy transcriptions. In *ACL-COLING: Joint conference of the Association for Computational Linguistics and the International Committee on Computational Linguistics*, 2006.

[15] L. Shrestha and K. McKeown. Detection of question-answer pairs in email conversations. In *COLING: Proceedings of the 20th international conference on Computational Linguistics*, 2004.

[16] H. Takeuchi, L. V. Subramaniam, T. Nasukawa, S. Roy, and S. Balakrishnan. A conversation-mining system for gathering insights to improve agent productivity. In *CEC-EEE: IEEE Joint Conference on E-Commerce Technology and Enterprise Computing, E-Commerce and E-Services*, 2007.

[17] I. H. Witten and E. Frank. Weka - a machine learning workbench for data mining. In *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2005.

[18] J. H. Wright, A. L. Gorin, and G. Riccardi. Automatic acquisition of salient grammar fragments for call-type classification. In *EUROSPEECH-1997*, pages 1419–1422, 1997.

[19] M. Zhang, B. Kao, D. W. Cheung, and K. Y. Yip. Mining periodic patterns with gap requirement from sequences. In *SIGMOD ’05: Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, pages 623–633, New York, NY, USA, 2005. ACM.

[20] 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 *ICASSP: IEEE Intl. Conference on Acoustics, Speech and Signal Processing*, 2006.


```
AGENT: Do you have a valid credit card
CUST: No but do you accept debit cards
AGENT: Maam do you have a round trip travel
ticket
CUST: I have a one way will that work
AGENT: Maam sorry you need to have a round trip
ticket or a valid credit card in your name
```

Figure 15: Snippet of a “Not Meeting Requirements" class