CallAssist: Helping Call Center Agents in Preference Elicitation

Ullas Nambiar Himanshu Gupta Mukesh Mohania
IBM India Research Lab
New Delhi, India
{ubnambiar, higupta3, mkmukesh}@in.ibm.com

1. INTRODUCTION

The increasing complexity of products and services being offered by businesses has made providing customers with easy access to technical assistance an important business function. Therefore, most businesses operate *call centers* to respond to product related queries from consumers. An emerging model is to let a third-party to run the contact center for a business. *Preference elicitation* - the process of asking queries to determine preferences is a key function performed by call-center agents. In this paper, our focus is on helping call-center agents to efficiently elicit customer's preference.

Example: Suppose the customer query is: *"Hi, this is John. I wanted to know the status of DVDs I ordered from ABC.com."* □

To answer the above query, the call center agent will need additional information about the transaction. An obvious solution is to request for an unique identifier such as *customer_id* or *credit card* and look for recent transactions. Often a unique *transaction_id* might not be known to the customer during the call. The resulting query over the underlying database could then return a large number of transactions done by the customer at *ABC.com*. During discussions with managers we were informed that new agents often found it difficult to quickly identify the transaction of interest during such conversations. The difficulty increases as agents are often given a dumb terminal over which advanced search functions are not made available and also the ranking models are often decided by the back-end CRM systems. Suppose the process model and database schema of the example store are as described in Figure 1. Given the complexity of the processes and corresponding database schemas, even a well-trained agent would need much time to determine the transactions that are relevant to current conversation. Often, the agent will ask a series of queries to narrow down the list of transactions further. Formulating the right set of queries has many challenges. Foremost among them is the need to quickly identify the *attributes* for which either the customer has provided binding values or can easily provide them if asked. Therefore, only agents with extensive knowledge about the business process and back-end database can quickly and satisfactorily answer the queries posed by customers. Moreover, agent must formulate queries that progressively reduce the context of the conversation. However, the task of checking whether a query has reduced the context and then formulating a new query that further narrows the context can become a non-trivial task if a large number of transactions map to the conversation. This indirectly reflects in large agent training times and long ramp-up periods for new agents. In fact, a recent survey [1] shows that 12% of incoming calls are abandoned due to high wait times. With agent time being a premium commodity at a call center, techniques to reduce the call duration by helping agent to quickly extract the structured information relevant to the customer's query becomes necessary.

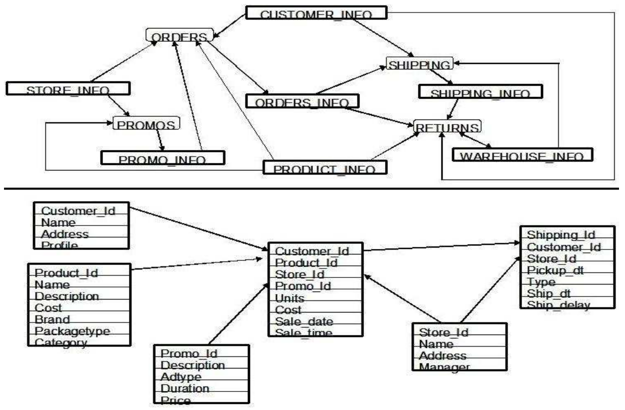


Figure 1: Example of a complex business process (top) and associated relational schema

While the problem of automating call center tasks has received some attention recently [4, 7, 1], much of the work looks at automated call routing or solution lookup. In this demo, we present *CallAssist* - a domain-independent system that *adaptively extracts relevant information from back-end databases and suggests queries to progressively narrow the context of the real-time conversation.*

The CallAssist Approach: We begin by assuming that all conversations handled by CallAssist will refer to a unique structured information stored in a back-end database. In other words, every conversation will contain the context of

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the VLDB copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Very Large Data Base Endowment. To copy otherwise, or to republish, to post on servers or to redistribute to lists, requires a fee and/or special permission from the publisher, ACM.
VLDB '07, September 23-28, 2007, Vienna, Austria.
Copyright 2007 VLDB Endowment, ACM 978-1-59593-649-3/07/09.

[1]Available at http://www.incoming.com/statistics/performance.aspx

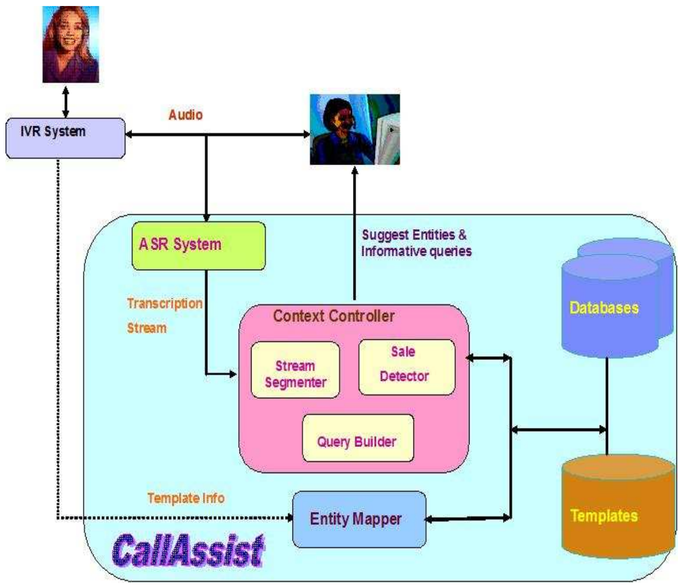


Figure 2: Architecture of CallAssist System

some entity [2] stored in a database. Hence, we start by identifying keywords - terms of interest appearing in the conversation. Our interest is in terms that might appear in a entity and since most attribute bindings are done using *noun phrases*, we select *noun phrases* as keywords. Accordingly, in the above example *John, DVD* and ABC.com are selected as keywords. Since the context defined as the entities containing a combination of the terms *John, DVD, ABC.com* is quite broad, a large number of entities will be relevant to the conversation. To narrow the list, we then formulate easy to answer yet highly classifying queries based on the extracted set. This is the *guiding principle behind our approach - measure the context of a conversation as the number of entities that can map to the conversation and then suggest queries whose answers will lead to a reduction in the mapped set of entities.* An added contribution of our system is the ability to extract relevant content *without requiring* the user to present `hard-to-remember` facts such as *transaction_ids* or share `sensitive` information like *social security* and *credit card* numbers. This will enable the call center to provide better privacy guarantees to its customers. Our solution is domain-independent and can be used by an agent with minimal understanding of the business. Thus, another advantage of using CallAssist is the reduction in agent training time, a major cost factor when inducting a new agent or relocating agent to a different business.

Challenges: The first challenge in realizing our approach is: *How to efficiently map relevant structured information to a given conversation?* Once we handle the above challenge, we can extract a set of entities that are ranked according to their relevance to the conversation. The extracted entity set could help the agent formulate future queries. However, if the entity set is large in size and the template has many features, then a number of queries can be formulated by the agent with not all being equally effective. This leads to our second challenge: *Which query when answered will lead to highest reduction in the context of the conversation?* The real-time nature of the conversation brings an additional challenge: *The entity extraction and relevant query suggestion tasks must be performed with near real-time response as a constraint.*

[2]An entity is a "thing" of significance, either real or conceptual about which the relational database holds information [3]

Solutions: Given a real-time audio call as input, CallAssist starts by converting the call into a text stream using an Automatic Speech Recognition (ASR) system. Then CallAssist maps keywords appearing in the stream to entities in the database. To perform the mapping, we extend the mapping techniques developed in the EROCS [2] system. EROCS mapped unstructured text from static and complete documents to entities stored in a database while in the call-center conversation transcripts are noisy and arrive in as a continuous stream from the ASR system. Since the input is a continuous stream, CallAssist ensures that the entities returned are updated based on the new keywords that are extracted from the stream. We continuously help the agent refine the context of the conversation by suggesting queries over attributes with highest information gain. To the best of our knowledge, there is no prior work that suggests context narrowing queries to a call-center agent.

2. THE CALLASSIST SYSTEM

The architecture of CallAssist and the flow of information through the system is illustrated in Figure 2. CallAssist consists of three main subsystems: an `Automatic Speech Recognition(ASR)` system, the `Context Controller` and `Entity Mapper`.

2.1 Extracting Relevant Structured Information

Entity: An entity is a "thing" of significance, either real or conceptual about which the relational database holds information [3].

Entity-template: An entity template specifies (a) the entities to be matched in the document and (b) for each entity, the context information that can be exploited to perform the match.

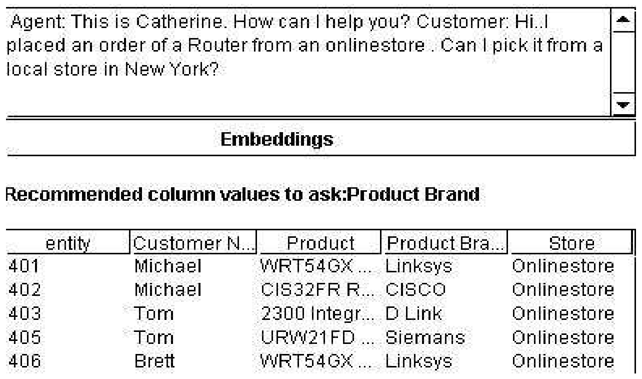
Formally, an entity template is a rooted tree with each node labeled using a table in the given relational database.The entities are extracted based on the entity-template associated with the audio call given as input. Based on the IVR input, we can classify the call into *new customer, tracking past order, filing complaint etc.* and also identify the entity-template to be used for processing the incoming call.

CallAssist assumes that an ASR system such as the IBM research prototype described in [8] is available for transcribing the input speech data. The output from ASR is *streamed* to `Entity Mapper` via the `Context Controller`. The output from `Stream Segmenter (SS)` - a subset of the audio conversation that has been buffered; is sent to `Entity Mapper` as the unstructured text to which entities have to mapped. The entities are defined by the *template* which is also given as input. To perform the mapping we extend the entity extraction framework described in [2]. The extensions are to overcome the challenges brought out by the streaming nature of input text, the inaccuracy introduced by ASR system and the need to provide answers in real-time. Once the best matching entities are identified, `Entity Mapper` returns the *top-K* relevant entities to the `Context Controller`. These entities are then shown to the agent.

2.2 Suggesting Context Narrowing Queries

An important contribution of CallAssist is that of suggesting context narrowing queries to the agent. The `Query Builder (QB)` identifies such queries based on the current state of the conversation (as captured by SS) and the relevant entities that are returned by the `Mapper`. Given our assumption of a single best entity, $\widehat{e}$, for every conversation, the set of entities E_t returned by `Entity Mapper` can be seen as a partial definition of $\widehat{e}$. Hence, QB must formulate a query that can quickly reduce the size of E_t. Essentially, this amounts to identifying the attribute that can classify E_t into the largest number of disjoint subsets. We use *information gain* [6] as the measure to decide the attribute over which to formulate the query. We avoid picking attributes such as *transaction ids, invoice numbers, etc* which would have large number of distinct values but would be difficult for the customer to provide. The task performed by QB is equivalent to identifying the attribute that might appear at the top of a decision tree built over E_t. Building the complete tree would help us identify a sequence of queries that when asked in order could identify a single entity belonging to E_t. However, for a certain E_t we cannot guarantee that $\widehat{e}$ is present in E_t since it is not based on the complete conversation. Therefore, building the complete decision tree and asking a series of queries may often lead to in-optimal use of time. Hence, in the current implementation QB identifies a single query for each distinct E_t.

Given the limited space available on the agent's desktop, we must ensure that only the most relevant parts of an entity are displayed to the agent. The information (set of attributes) displayed must explain why the given set of entities were chosen from all the entities available. Moreover, the queries suggested by QB should involve the attributes that are being displayed, so that the agent is aware of what answers to expect from the customer. Thus, deciding an output schema for the extracted entities at run-time is a non-trivial task. In fact, this problem quickly becomes *NP-Complete* as shown in [5]. Hence, in our current implementation, we assume that a *output template* corresponding to each *entity template* is given at design time. We plan to extend the heuristics identified in [5] to develop dynamic display solutions.

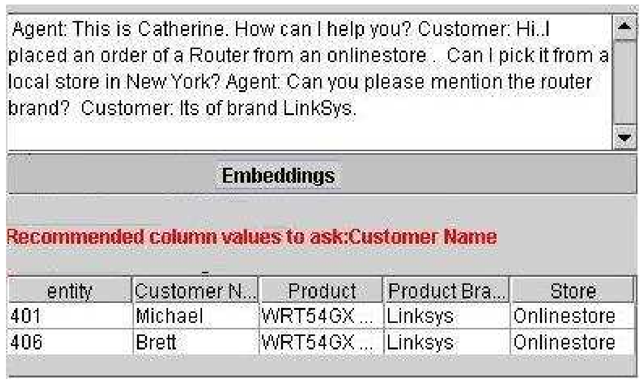

Agent: This is Catherine. How can I help you? Customer: Hi..I placed an order of a Router from an onlinestore . Can I pick it from a local store in New York?

Embeddings

Recommended column values to ask:Product Brand

entity	Customer N...	Product	Product Bra...	Store
401	Michael	WRT54GX ...	Linksys	Onlinestore
402	Michael	CIS32FR R...	CISCO	Onlinestore
403	Tom	2300 Integr...	D Link	Onlinestore
405	Tom	URW21FD ...	Siemans	Onlinestore
406	Brett	WRT54GX ...	Linksys	Onlinestore

Figure 3: CallAssist Interface Displaying Call Transcript, Relevant Entities and Suggested Query


Agent: This is Catherine. How can I help you? Customer: Hi..I placed an order of a Router from an onlinestore . Can I pick it from a local store in New York? Agent: Can you please mention the router brand? Customer: Its of brand LinkSys.

Embeddings

Recommended column values to ask:Customer Name

entity	Customer N...	Product	Product Bra...	Store
401	Michael	WRT54GX ...	Linksys	Onlinestore
406	Brett	WRT54GX ...	Linksys	Onlinestore

Figure 4: CallAssist Interface Displaying Reduced Entity Set based on Customer Feedback

3. DEMONSTRATION

In this demo we will showcase CallAssist's domain independent approach for efficiently generating preference elicitation queries to help a call-center agent. Figure 3 and Figure 4 show sequential screenshots of the interface displayed to the call-center agent during a conversation. We will give an end-to-end demonstration of the CallAssist system with focus on the coping with inaccuracies introduced by ASR during transcription and the need for continuously generating relevant suggestions under strict response-time constraints.

3.1 Dealing with Noisy Transcripts

The input to CallAssist is a noisy stream of text data. The ability of CallAssist to suggest entities of interest is therefore dependent on how well it can handle the noise in the input. Generally, ASR systems are known to have 30-40% error in transcription of call-center conversations. Since, CallAssist is only focussing on the noun phrases, only errors introduced in the noun phrase transcription would affect the system. Furthermore, only terms appearing in the database will be effectively used by CallAssist. Therefore, using a controlled vocabulary derived from the database to tune the ASR system can further reduce the inaccuracies in the system. We will demonstrate that higher transcription accuracy can be obtained using a controlled vocabulary using a dataset consisting of both manually and automatically generated transcripts from 900 calls received by a US Car Rental Agency. The ASR system used to transcribe the calls is trained over a controlled vocabulary consisting of model names, cities etc. We use CKER (Controlled Keywords Error Rate), measured as $\frac{S+D+I}{N} \times 100$, where N is the total number of controlled vocabulary terms in the manually generated transcript, and S, I and D are the total number of substitution, insertion, and deletions in the automatic transcript. Figure 5 shows that by considering CKER instead of WER (error over all words) the error is reduced to 30-35% from 40-45%. The error can be further reduced by using the vocabulary to fix spelling errors. The CKER+ plot in Figure 5 which measures the error after converting all terms within a predefined edit distance from a vocabulary term to the term further reduces average transcription error to around 25%.

3.2 Efficiency and Accuracy of CallAssist

Automatic transcription of call-center conversations results in generation of error-prone text. To help the call-center agent, CallAssist must efficiently identify the context of the conversation using the noisy transcript that is continuously streamed by an ASR system. Moreover, customers involved in a voice conversation with a call-center agent expect responses in real-time. Hence, CallAssist must provide informative suggestions to the agent in real-time and the suggestions must also take into account them latest information provided by the customer. Thus, CallAssist must also ensure that set of relevant entities mapping to a conversation are continuosuly updated based on the input. Moreover, the mappings must be done under strict time constraints.

In this demonstration, we will present CallAssist's ability to efficiently update the set of relevant entities over a noisy input stream and with varying response time constraints. Intuitively, both an increase in transcription error and/or reduction in response time should lead to reduction in accuracy of extracted entities and thereby affect the suggested queries. However, as evident from Figure 6 and Figure 7, CallAssist is able to maintain high levels of accuracy in the face of variation in response time and noise levels of input transcript. The suggested queries were manually checked for accuracy by volunteers and were found to be highly relevant.

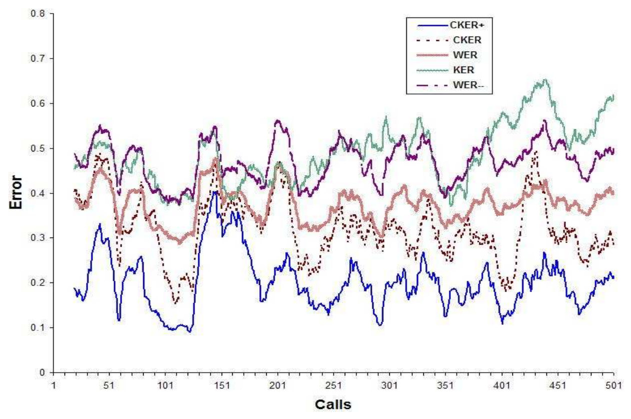


Figure 5: Comparison of WER, KER and CKER

4. SUMMARY

In this demonstration, we present CallAssist - a domain independent system for helping a call-center agent to elicit information from a customer during an audio conversation. CallAssist's contributions include - (1) an efficient and accurate approach for generating preference elicitation queries that depends on (2) an approach for continuously mapping relevant structured information to the streaming audio conversation using minimal contextual information.

Acknowledgments: We thank Shourya Roy and L. V. Subramaniam for helpful discussions on ASR systems and providing the transcripts. We thank Amitabh Ojha and Ashish Verma for helpful discussions and comments.

5. REFERENCES

[1] D. Carmel, M. Shtalhaim, and A. Soffer. eResponder: Electronic Question Responder. *7th International Conference on Cooperative Information Systems*, London, UK, 2000.

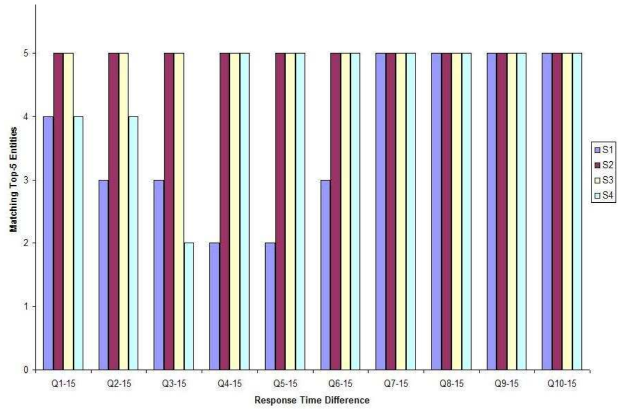


Figure 6: Entity Similarity between Response Time Restricted and Unrestricted Runs of CallAssist

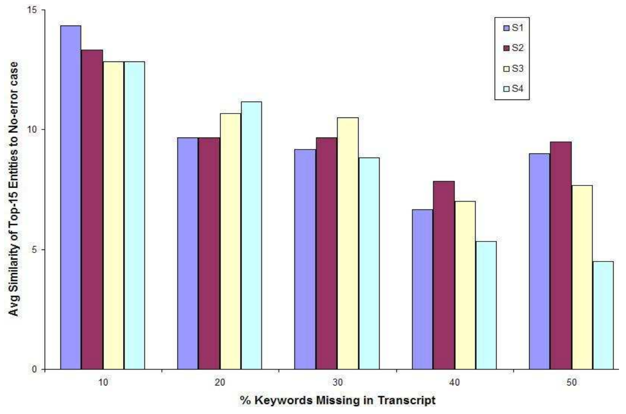


Figure 7: Similarity of Top-15 Entities Extracted from Transcripts with Varying Accuracy

[2] V. Chakaravarthy, H. Gupta, P. Roy, and M. Mohania. Efficiently Linking Text Documents With Relevant Structured Information. In *VLDB*, September 2006.

[3] P. Chen. The entity-relationship model: Toward a unified view of data. *ACM Transactions on Database Systems*, 1(1):9–36, 1976.

[4] J. Chu-Carroll and B. Carpenter. Vector-based natural language call routing. *Comput. Linguist.*, 25(3):361388, 1999.

[5] G. Das, V. Hristidis, N. Kapoor, and S. Sudarshan. Ordering the Attributes of Query Results. *SIGMOD*, 2006.

[6] T. Mitchell. Machine Learning. *McGraw Hill*, 1997.

[7] G. Riccardi, A. Gorin, A. Ljolje, and M. Riley. A spoken language system for automated call routing. *ICASSP 97*, Germany, 1997.

[8] H. Soltau, B. Kingsbury, L. Mangu, D. Povey, G. Saon, and G. Zweig. The IBM 2004 Coversational Telephony System for Rich Transcription. *IEEE ICASSP*, March 2005.

[9] M. Tang, B. Pellom, and K. Hacioglu. Call-type classification and unsupervised training for the call center domain. *Automatic Speech Recognition and Understanding Workshop*, November 2003.