An Anatomy for Artificial Conversation Generation in the Customer Service Domain

Chayan Chakrabarti and George F. Luger
Computer Science Department
University of New Mexico
Albuquerque, New Mexico

Abstract

Artificial conversations have been applied to the domain of customer service operations through virtual customer service chatter bots. One approach to artificial conversation generation, inspired by conversation theory and pragmatics, is combining content semantics with pragmatic semantics. Content semantics provide the background knowledge for a targeted purpose-driven conversation in a specific customer service situation. Pragmatic semantics provide the conversation engineering protocols as defined by certain existing social and practical conventions. A conversation architecture combining these semantics provides a robust and scalable means to generate artificial conversations. This work describes the low level details of the conversation generation process. Every step of the process, from preprocessing to model selection to utterance generation is examined in detail.

Introduction to Artificial Conversations

Contemporary chatter bots perform very well in tasks like question-answering and in single-pair utterance exchanges. However, they do not perform well at tasks where a specific context has to be maintained across a several utterance-exchange pairs (Chakrabarti and Luger 2013). A conversation, as conventionally understood, isn't merely a collection of utterance-exchange pairs. It is a process, grounded in a knowledge base, and modeled on specific social and practical conventions (Ginzburg 2008; Chakrabarti 2014). This is especially pertinent in customer-service situations, which involves short purposeful targeted conversations with a well-defined goal, and strictly defined conventions (Chakrabarti 2014).

GUS (Genial Understander System), one of the earliest works in conversation systems, was a virtual agent helping a customer make reservations. But it could handle only a very restricted set of questions, and the domain knowledge of the question-answer sequence had be encoded very precisely, which limited the scalability (Bobrow et al. 1977). The GALAXY Communicator system at MIT (Seneff et al. 1998; Polifroni and Seneff 2000) is a client-server architecture for communicating online information including weather and flight information and has several components including database access, speech synthesizer, speech recognizer, and a language understanding engine. It is not set up to build the knowledge base using facts, but in terms of anticipated questions (Filisko and Seneff 2003). The DARPA Communicator project (Levin et al. 2000) was an initiative to support advanced conversational capabilities including negotiation, plan optimization, and complex explanations. Some specialized techniques leverage dialogue structure in specific contexts to improve accuracy by encoding speech recognition patterns (Metallinou et al. 2013). Neural networks have also been used for deep-learning solutions to this problem (Henderson, Thomson, and Young 2013). Partially Observable Markov Decisions Processes (POMDPs) are also used to model conversations. They improve upon traditional conversational systems in that they can better handle ambiguity in changing domains (Gasic et al. 2013). Reinforcement learning techniques are also used for this problem (Rieser and Lemon 2013).

Although there has been progress made over the years in the design of conversational engineering systems, there is one major limitation to most of them. They do not make an explicit distinction to modeling the content required for the conversation and the semantics inherent in the conversation process. Most approaches either focus on only one of content modeling or conversation semantics, or sub-aspects of these, or incorporate both of them together without making an explicit distinction. This leads to blind spots in the application, in which either one has to encode content and semantics for a new domain from scratch, or the system has to undergo substantial remodeling to handle conversations of a different type.

Recent work in conversation architectures have demonstrated that combining the modeling of content semantics and pragmatic semantics can achieve good results in artificial conversation generation (Chakrabarti and Luger 2012; Chakrabarti and Luger 2013). Such an architecture has proved to particularly useful in customer service situations, modeling the chat interaction between a human customer and a virtual customer service representative (Chakrabarti 2014). This paper describes in detail the process of generating artificial conversations in the customer-service domain, examining the role each intermediate step plays in the process.

Conversation Architecture

Our architecture has a Knowledge Engine that models the content semantics, a Conversation Engine that models the pragmatic semantics, and a Chat Interface that performs pre-processing and post-processing tasks (Figure 1).

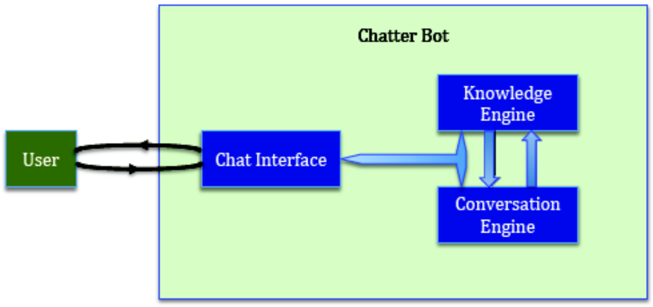

Figure 1: Architecture for the Chatter Bot

The Chat Interface contains modules for receiving user input, performing stemming, detecting speech acts, detecting topic, and interfacing with the Knowledge Engine and Conversation Engine (Chakrabarti 2014).

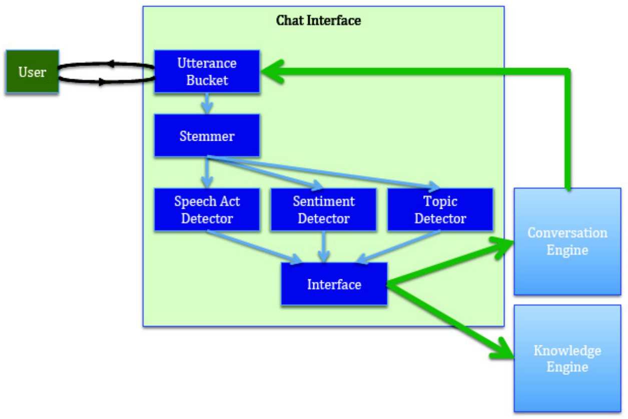

Figure 2: The Chat Interface directly interfaces with the user.

The Knowledge Engine identifies the specific speech act for the utterance and also the specific topic being discussed. A goal-fulfillment map (O'Shea, Bandar, and Crockett 2010) specifies the content semantics for the conversation. The specific goal-fulfillment map is selected from a double-key hash table, where the keys are the topic and the speech act (Chakrabarti 2014).

The Conversation Engine models four different types of conversations, procedural conversations, informational conversations, troubleshooting conversations, and dispute-resolution conversations using four different probabilistic finite state automata (Chakrabarti 2014). Figure 4. shows one such finite state machine for a troubleshooting conversation. Distinct finite state machines have been defined for other types of conversations as well.

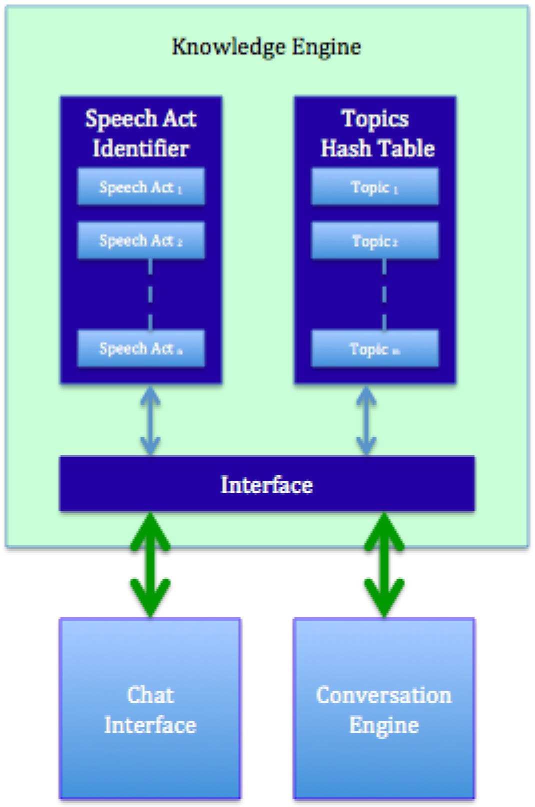

Figure 3: The Knowledge Engine models content semantics

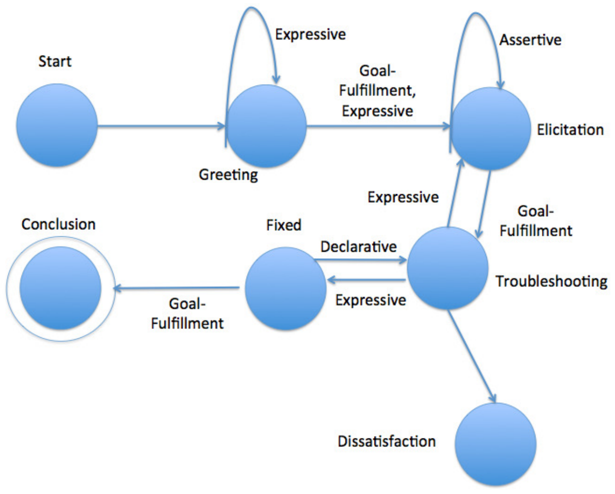

Figure 4: The Finite state automaton for Troubleshooting Conversations.

The conversation planner maintains a workspace of four types of conversations, and increases or decreases a heuristic score for each type depending on how the conversation unfolds. A successful conversation is one in which only one type of conversation remains in the workspace, and the probabilistic finite state automaton associated with that conversation reaches a defined accepting state. A failed conversation is one in which the conversation reaches a defined unescapable dissatisfaction state for one of the probabilistic finite state automation, or all four probabilistic finite state automata associated with the four conversation types are dropped from the workspace (Chakrabarti 2014). Figure 5. shows an overview of the conversation planner.

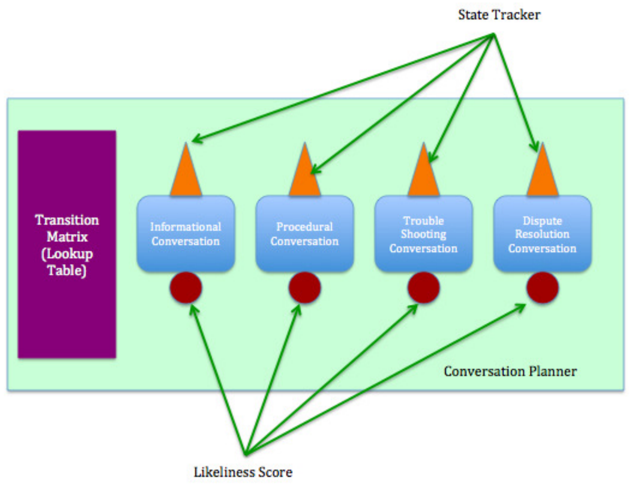
State Tracker
Transition Matrix (Lookup Table)
Informational Conversation
Procedural Conversation
Trouble Shooting Conversation
Dispute Resolution Conversation
Conversation Planner
Likeliness Score

Figure 5: The Conversation Planner consisting of the transition matrix lookup table, the state tracker, the likeliness score variable, and the conversation solutions in the workspace.

(Chakrabarti 2014) shows in detail how the Knowledge Engine in conjunction with the Conversation Engine engineers artificial conversations. The architecture models how a human would generate a conversation and incorporates well defined ideas from conversation theory, speech act theory, and the theory of pragmatics.

Corpus and Parameter Learning

We used a corpus of chat transcripts between a human customer and a human customer service agent working for an online electronic trading portal. The corpus consisted of 2,886 distinct conversations. Each conversation was in the form of an Excel file and was clearly tagged by a unique conversation identifier. For example, in each conversation, the utterances were tagged by who was delivering it, either the customer or the customer service agent. An utterance is everything that is said by either the customer or the representative in a single turn. It consists of one for more sentences. We assume that each utterance belongs to a single context.

A series of successive utterance pairs on the same context constitutes a conversation. The shortest conversation had 5 distinct utterances. The longest conversation had 82 distinct

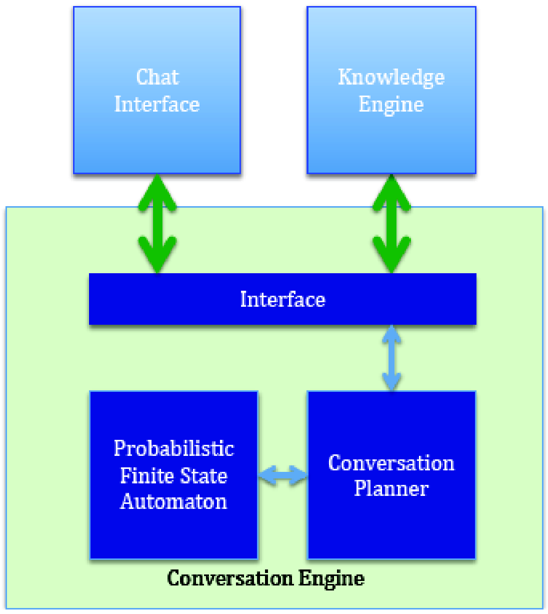

Figure 6: The Conversation Engine models pragmatic semantics

utterances. The median was 26 utterances and the average was around 22 utterances. The utterances were mostly interleaved, i.e., alternating between the customer and the representative. Most of the conversations were related to a single context. The conversations that were not interleaved and related to more than one context were not analyzed.

We used a bag-of-words based latent-semantic algorithm to tag each utterance in each conversation in the corpus with a speech act. We also used a bag-of-words based latent-semantic algorithm to tag each conversation in the corpus with one of the topics (Chakrabarti 2014). The transition probabilities for the four finite state automaton corresponding to the 4 types of conversations were also learned from the corpus.

Generation of Artificial Conversations

This section shows in detail how an artificial conversation is generated in a step-by-step manner.

1. The conversation starts with a human making a comment.

   ```
   Customer : I would like to open a new
   account for day trading.
   What are my options?
   ```

   This message is entered from the standard terminal. The Utterance Bucket directly collects the text in the form of a string. A standard spellchecker and grammar checker autocorrects the spelling and grammatical errors in the sentence if any

2. The correct sentence, free of spelling and grammatical error, is sent to the Stemmer. Using Porter's Stemming algo-

rithm, the following stems are obtained, "account", "day trade", "open", and "options".

3. The entire stemmed sentence is then passed on simultaneously to the Speech Act Detector, the Sentiment detector, and the Topic Detector. The following events then take place.
   * The Speech Act Detector uses Latent Semantic Analysis to determine that the type of speech act is "Expressive", since the bag of words included "would" and "like".
   * The Sentiment Detector detects that the sentiment is neutral, since none of the words from the positive or negative bag of words is encountered.
   * The Topic Detector determines using Latent Semantic analysis that the topic is "new account" using bag of words "new", "account", and "open".
4. The output of the Speech Act Detector, the Sentiment Detector, and the Topic Detector is then sent to the interface. The Interface combines these into an array list, and sends the array list to the Conversation Engine and the Knowledge Engine simultaneously.
5. In the Knowledge Engine, the following steps take place.
   * The Interface of the knowledge engine receives the array list and sends it to the Speech Act Identifier. This module selects the correct speech act from the list as "expressive".
   * The interface also sends the bag of words to the topic hash table. The hash table retrieves the topic as "new account". The appropriate context map is then pulled out. This context map lists the steps for the encoded knowledge for opening a new account in the form of a goal-fulfillment map. The appropriate goal-fulfillment map, shown in Figure 5.17, is then put in to the workspace and sent to the interface.
   * A goal-fulfillment algorithm is initiated. A counter is initiated to keep track of the progression of goals in the map.
6. In the Conversation Engine, the following steps take place.
   * In the Probabilistic finite State Automata, initially all four possible solutions are maintained. This is because initially the probabilities of each conversation type will be nearly equal. A counter is initialized to maintain the current state of the conversation in each solution.
   * The Conversation Planner will calculate the probabilities of transition from one state to another depending upon the Speech Act being uttered. These transitions are learned from the corpus and are stored in a lookup table. The Conversation Planner is responsible for advancing the counter indicating the current state of the conversation.
7. The information is sent back to the Chat Interface. The Utterance Bucket corrects spelling (unlikely) and grammatical errors, and then outputs the response of the chatter bot to the standard terminal.

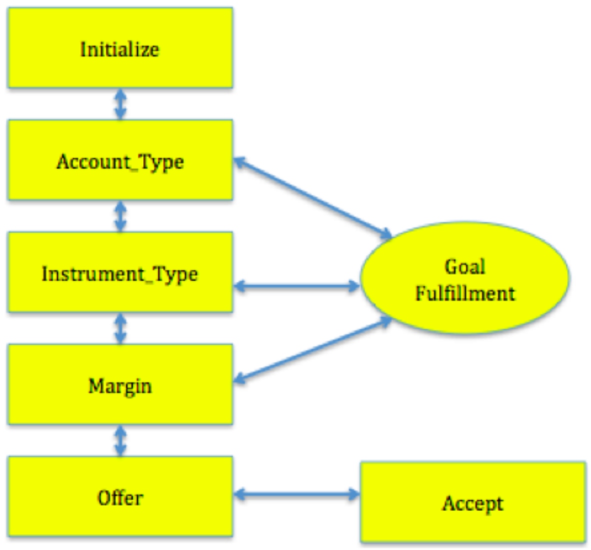

Figure 7: Goal-fulfillment map selected by the Knowledge Engine in the anatomy of a conversation.

```
Chatter Bot : Do you have an existing
trading account or would you like to
open a new one?
```

8. This process is repeated until the end of the conversation is indicated by the Conversation Planner counter being in an accepting state.

Conversation Creation

The next step is to actually generate the artificial conversation using the chatter bot architecture. The conversations are generated by a person, by interacting with the chatter bot architecture via a standard terminal. These are the steps to generate a conversation.

1. Play the role of the customer of the online electronic trading website. Pick out an issue from the list in 6.1.2. "Know" the responses to all the customer-side details. For example, know that the account can have two different modes and two different trading configurations.
2. Begin a conversation with the chatter bot by typing on the standard terminal.
3. The bot will then initiate a question. It will be displayed on to the terminal window. This will almost always be "small talk" at the beginning of the conversation. Answer the questions the bot asks by typing back into the terminal window.
4. The conversation will be lead by the bot, i.e.,
   - the bot will either ask the question to which the customer will respond (when did you put in the buy order?), or

- the bot will instruct the customer to perform some action (change the configuration of the account) to which the customer will answer affirmatively that he / she has completed the action, or answer negatively that he / she is unable to perform the action with a qualifier (I am unable to access the reset password form. I do not have my customer relationship number.) or
- The bot will ask a question that will require a Yes or No answer.

5. The responses of the customer has to be an exact match with the expected answer in the goal fulfillment map, irrespective of the response that the customer choses. For example, in response to a query from the bot: ”Do you remember what kind of orders you placed?”
   - The customer can either answer negatively ”No, I do not remember” or
   - The customer can answer ”Yes, they were buy orders” or ”Buy orders” or ”Yes, buy orders”
   - The customer can answer ”Yes, they were sell orders” or ”Sells orders” or ”Yes, sell orders”

   But the customer cannot answer ”Very unlikely they were buy orders, but I am not really sure”. This is because sentence similarity hasn't been implemented in this architecture. Sentence similarity is the area of research that reduces a range of semantically similar sentences into a root sentence (O'Shea et al. 2004; O'Shea, Bandar, and Crockett 2009). Hence for this dissertation, the responses need to have the exact words with only a slight change in grammar.
6. The transcript of the conversation is written to a file, and is tagged with the customer utterance and bot utterance. These transcripts can then be analyzed.

Results, Conclusions, and Future Directions

We generated 48 artificial conversations using this technique. Out of these, 42 conversations reached a conclusion state, and 6 conversations failed. Thus, we had a success rate of 87.5%. The transcripts of all 48 conversations are available at `www.cs.unm.edu/~cc/artificial_conversations/transcripts/`.

We demonstrate a modular, robust, and scalable architecture for chatter bots. The specific concepts of pragmatics, speech acts, and dialogue acts are well known in the field of conversation theory. However, this research is the first example of computationally modeling these specific concepts to realize pragmatic semantics for chatter bots. Similarly, specific concepts like goal-fulfillment maps have been explored previously in the knowledge representation literature. But this work is the first example of using goal-fulfillment maps for modeling content semantics for chatter bots in the form of a series of sub-contexts. In addition, this work is the first example of combining pragmatic semantics and content semantics to generate artificial conversations.

There are several exciting directions in which this work might be extended. Incorporating richer knowledge representation and retrieval techniques, such as ontologies, might make the architecture work with even less situation specific contextual conversations. We considered only four types of conversations, i.e., Procedural, Informational, Troubleshooting, and Dispute Resolution. Other types of conversations can be defined and the modeling and analysis can be extended to these types.The conversations were modeled using stochastic finite state automata, which worked well in narrow situational contexts.

References

[Bobrow et al. 1977] Bobrow, D. G.; Kaplan, R. M.; Kay, M.; Norman, D. A.; Thompson, H.; and Winograd, T. 1977. Gus: A frame-driven dia—og system. *Artificial Intelligence* 8:155–173.

[Chakrabarti and Luger 2012] Chakrabarti, C., and Luger, G. 2012. A semantic architecture for artificial conversations. In *The 13th International Symposium on Advanced Intelligent Systems*. Kobe, Japan: IEEE Press.

[Chakrabarti and Luger 2013] Chakrabarti, C., and Luger, G. 2013. A framework for simulating and evaluating artificial chatter bot conversations. In *The 26th International Florida Artificial Intelligence Research Society Conference*. St. Pete Beach, FL: AAAI Press.

[Chakrabarti 2014] Chakrabarti, C. 2014. *Artificial Conversations for Chatter Bots Using Knowledge Representation, Learning, and Pragmatics*. Ph.D. Dissertation, University of New Mexico, Albuquerque, NM.

[Filisko and Seneff 2003] Filisko, E., and Seneff, S. 2003. A context resolution server for the galaxy conversational systems. In *Proc. Eurospeech*.

[Gasic et al. 2013] Gasic, M.; Breslin, C.; Henderson, M.; Kim, D.; Szummer, M.; Thomson, B.; Tsiakoulis, P.; and Young, S. 2013. Pomdp-based dialogue manager adaptation to extended domains. In *SigDial Metz France*.

[Ginzburg 2008] Ginzburg, J. 2008. *Semantics for Conversation*. King's College, London: CSLI Publications.

[Henderson, Thomson, and Young 2013] Henderson, M.; Thomson, B.; and Young, S. 2013. Deep neural network approach for the dialog state tracking challenge. In *SigDial Metz FranceMetz France*.

[Levin et al. 2000] Levin, E.; Narayanan, S.; Pieraccini, R.; Biatov, K.; Bocchieri, E.; Fabbrizio, G. D.; Eckert, W.; Lee, S.; Pokrovsky, A.; Rahim, M.; Ruscitti, P.; and Walker, M. 2000. The att-darpa communicator mixed-initiative spoken dialog system. In *ICSLP*.

[Metallinou et al. 2013] Metallinou, A.; Bohus, D.; ; and Williams, J. D. 2013. Discriminative state tracking for spoken dialog systems. In *Proceedings of Annual Meeting of the Association for Computational Linguistics (ACL), Sofia, Bulgaria*.

[O'Shea, Bandar, and Crockett 2009] O'Shea, K.; Bandar, Z.; and Crockett, K. 2009. A semantic- based conversational agent framework. In *The 4th International Conference for Internet Technology and Secured Transactions (ICITST-2009), Technical Co- Sponsored by IEEE UK?RI Communications Chapter*, 92–99.

[O'Shea, Bandar, and Crockett 2010] O'Shea, K.; Bandar, Z.; and Crockett, K. 2010. A conversational agent framework using semantic analysis. *International Journal of Intelligent Computing Research (IJICR)* 1(1/2).

[O'Shea et al. 2004] O'Shea, K.; Bandar, Z.; Crockett, K.; and Mclean, D. 2004. A comparative study of two short text semantic similarity measures. *Lecture Notes on Artificial Intelligence* 4953:172.

[Polifroni and Seneff 2000] Polifroni, J., and Seneff, S. 2000. Galaxy-ii as an architecture for spoken dialogue evaluation. In *LREC*.

[Rieser and Lemon 2013] Rieser, V., and Lemon, O. 2013. *Reinforcement Learning for Adaptive Dialogue Systems: A Data-driven Methodology for Dialogue Management and Natural Language Generation*. Springer.

[Seneff et al. 1998] Seneff, S.; Hurley, E.; Lau, R.; Pao, C.; Schmid, P.; and Zue, V. 1998. Galaxy-ii: A reference architecture for conversational system development. In *Proc. ICSLP, Sydney, Australia*.