"Gimme the Usual" - How Handling of Pragmatics Improves Chatbots

Alessia Bianchini, Francesco Tarasconi,
Raffaella Ventaglio, Mariafrancesca Guadalupi
CELI Language Technology
{bianchini,tarasconi,ventaglio,guadalupi}@celi.it

Abstract

English. We provide our view on the components needed for both the development and further improvement of robust and effective chatbots. We focus on why Pragmatics is important in developing next generation chatbots by bringing a few generalizable examples. We report our current experience on the design and implementation of a task-oriented textual chatbot for a closed-domain Question Answering system, which tackles problems in Pragmatics.

Italiano. *Forniamo la nostra visione su quali sono i componenti necessari per realizzare e migliorare chatbot robusti ed efficaci. Ci concentriamo sul perché la pragmatica sia importante nello sviluppo di chatbot di nuova generazione portando esempi generalizzabili. Riportiamo la nostra esperienza nella progettazione e implementazione di un chatbot testuale task-oriented per un sistema di Question Answering a dominio chiuso che affronta problemi di pragmatica.*

1 Why Pragmatics Matters in Chatbots

Chatbot, chatterbot, natural-language interface, dialogue system are some of the terms used to refer to softwares that aim to carry on conversations with humans (Mauldin, 1994; Lester, Branting and Mott, 2004; Boualem, Casati and Toumani, 2004). We will not go into further details about the classification and definition of such softwares. We will use *chatbot* as if it was a hypernym of the above mentioned softwares instead.

Chatbots and Intent Understanding. The goal of an intelligent chatbot is to understand the user's intent (Yue, 2017) and behave accordingly. Such goal is quite complex to achieve, and beyond the capability of current state of the art chatbots. However, the hype around chatbots has raised awareness of what elements are needed for a chatbot to manage human-like interactions. It is generally agreed that to build effective and solid chatbots the following is needed:

Natural Language Processing (NLP): much of the intelligence needed to understand human intent lies in the processing of human language. Hence, the development and improvement of NLP algorithms is a necessary prerequisite for the creation of intelligent chatbots.

Machine Learning (ML): chatbot design should rely on ML for learning and automatically consolidating NLP rules by means of observation of past experience - i.e., past conversations and their outcomes (Perez-Marin, 2011). Current chatbot development, given enough annotated data, should consider adopting recently developed algorithms that are task-oriented (Bordes and Weston, 2016) or topic aware (Xing, Chen et al., 2017). Developments in reinforcement learning applications seem promising for task-oriented dialogue systems (Rieser and Lemon, 2011).

Context and State Awareness: depending on the purpose of the chatbot, the component responsible for the managing of the conversation (Dialogue Manager System - DMS) should take into account both context and states variables (Allen, Byron, Dzikovska, Ferguson, Galescu and Stent, 2001). From the DMS point of view, chatbots are usually classified as: stateless chatbots; semi-stateful chatbots; stateful chatbots (next generation chatbots). During the conversation, state transitions

depend on the information acquired before. As for the follow up action, it depends on the recognized context.

Natural Language Generation (NLG): NLG concerns what information and in what form it should be delivered (Breen, 2014). Dealing with "real" conversation requires being both proactive (e.g. suggest the best option; drive along the compilation of a form; remind planned activities; ...) (Owen et al., 2001) and adaptive (e.g. change style - both in written and spoken scenarios - according to domain, mood of the user, or sociolinguistic variables).

We argue here that, in addition to the above mentioned, moving from Semantics to Pragmatics plays a crucial role in building chatbots. This is because a lot of the knowledge human beings share during a conversation gets constructed along the conversation itself (Robyn, 2002; Pask, 1975). For instance, let's consider the following mock dialogue between Human (H) and Chatbot (C):

H lands on a money transfer service page.
H: Hello, I would like to make a transfer
C: Hello. Sure. Would you like to know more about: [FAQ menu about transfer service is shown]?
H navigates the FAQ menu
H signs up online to proceed with the transfer.
H: I would like to make a transfer
C: Sure. It only takes a couple of minutes
C starts the procedure to execute the transfer.

In this interaction, the sentence "I would like to make a transfer" instantiates two different intents: informative intent, at first (H is looking for information about the transfer service); follow up intent, then (once H is satisfied with transfer service conditions, he/she wants to proceed with the transfer). Such *pragmatic disambiguation* involves taking knowledge from the conversational context into account, which is one of the most difficult tasks for a chatbot. We report below how we deal with this task in our task-oriented closed-domain chatbot.

2 Intent Understanding in Practice

Understanding intents implies handling both semantic meaning and pragmatic meaning. Roughly speaking, while semantics concerns the meaning of a sentence from the linguistic point of view, Pragmatics concerns the interpretation of the same sentence depending on extralinguistic knowledge (Grice, 1975). As mentioned before, a sentence can be ambiguous from the intent point of view. As for classifying intents, there seems to be no comprehensive literature about it, yet - not from the chatbot perspective, at least. However, based on our business experience, we would arrange intents as follows:

Informative Intent: the user is looking for information; e.g. *Question and Answer* (QA), FAQ browsing typically instantiate this intent.

Follow Up Intent: as in regular conversations, the user wants to "do things with words" (Austin, 1962), perform actions; e.g. "Call the call center", "Order pizza", "Turn on washing machine".

Dialogic Intent: the user uses *discourse markers* to connect, organize, manage the conversation; e.g.: greetings, farewells, turns markers, ...

Regular expressions, pattern matching and keyword recognition typically are not enough to achieve *real* intent understanding. This is because the more the interaction is *human-like*, the more complicated it becomes to figure out what the human really wants. Among business intent, real life cases we faced are: Onboarding, Question Answering and Education. In our applications, we break down the understanding process into subtasks. Namely: *intent classification* (e.g. "booking a flight"); *slot filling*, i.e. enriching the intent with more detailed information (such as "destination" and "departure time"); *context modeling*, i.e. keeping track of context to get to the correct meaning ("time" might refer to "flight departure time", "flight arrival time", "dinner time", etc...).

3 System Design and Architecture

Our task-oriented closed-domain financial textual chatbot, *Financial QA Chatbot*, aims to provide users with answers concerning banks and insurances, through a conversation in Italian. The type of answers that a user can obtain are similar to the ones found on a financial platform website[1]: this portal provides a search engine and FAQ section to satisfy the information need. Therefore, it is mainly a QA chatbot, although some additional follow up actions are available on top of providing an answer to questions, such as redirecting to specific websites or services. Financial QA is provided with a proprietary scoring algorithm to match the current user's questions to answers in a database A. In line with previous work (Quarteroni and Manandhar, 2007), we will review key design and architecture aspects, with emphasis on possible solutions to Pragmatics problems discussed in sections 1 and 2. In this sense, the most significant components are the Dialogue Manager and the Context Manager, which provide the scoring algorithm enriched information.
NLP functions such as normalization, tokenization, lemmatization, POS tagging, disambiguation and dependency parsing are made available through the CELI linguistic pipeline[2] (Tarasconi and Di Tomaso, 2015).

Dialogue Scenario: a QA session consists of *actions* that can be performed by the user or by the automated system, according to *Dialogue Management* logic.
User actions: greet, quit, ask a question q, acknowledge the previous utterance, ask for help/suggestions, browse the navigation menu.
System actions: greet, quit, present answer a, acknowledge the previous utterance, ask for clarifications, propose a follow up (question/action), reprimand for using swearwords, suggest questions, present or hide the navigation menu.

User's action classification: each user's utterance is classified into one of five action classes: greet, quit, ask a question, acknowledge, ask for help. This is accomplished using predefined dictionaries and automatic classifiers, which also consider discourse markers and *disfluencies*.
Although there is promising work done on dialog act detection with multi-level information (Rosset et al., 2008), in this step with adopt a simpler approach, leaving further refinements to subsequent components.

Dialogue Management: the conversation proceeds along these logics.

1. An initial greeting (*greet* action), a request for help (*ask for help*) or a direct question q (*ask a question*) from the user.

2. The system, if asked for help, presents the user with a navigation menu, based on current context and on the given hierarchical classification of contexts or topics (see *Context Management* below). This menu can be browsed until a terminal node in the classification is reached, and, at that point, a predefined set of questions related to that topic is suggested. The user can select a question q from that list.

3. q is analyzed to detect *wh-type* (Huang et al., 2008) and whether it is elliptic or anaphoric. This information is passed along with q and the current context to the subsequent QA component.

4. The QA component searches for matches of the query according to the *QA Algorithm*. Each matching answer a_k is accompanied by a $relevance$ score r_k, $r_k \in (0, 1]$. If at least one match has relevance more than a fixed threshold T, only the best match (highest relevance) is returned. Otherwise, up to the top N_r highest results are returned by the QA component. In Financial QA's basic settings, $T = 0.75$ and $N_r = 5$.

5. The QA component results are processed: they can be a single answer or, because of low relevance scores, a list of answers. If a single answer is provided by the QA component, it is returned to the user (*answer* action). In the case of a list, the user is asked for clarifications, and a single answer is selected based on her additional input (*ask for clarifications* action, then *answer* action). After an answer is provided to the user, context is updated accordingly.

6. The system inquires whether the user is interested in a follow up session; if this is the case,

[1] gooruf.com
[2] www.celi.it

the user can enter a question again. Else, the system acknowledges.

7. Whenever the user wants to terminate the interaction, a final greeting is exchanged (*quit* action).

Context Management: intuitively, all the answers a in the knowledge base are grouped in disjoint topics of maximum granularity, which are then organized in a hierarchical structure, used to model context in this QA task.
Managing topic hierarchies can improve performance in a query matching system (Domingues et al., 2014). Formally, context elements are topics of conversation belonging to the finite set $C = \{C_1, \ldots, C_N\}$. Topics are arranged in a hierarchical classification structure, which can be represented as a tree $T = (C, E)$, where C is the set of nodes. Edges E express the "C_i has subclass C_j" relation. A context X is, in general, an arbitrarily ordered sequence of topics.
In our current implementation of Financial QA, we support only contexts of length 1, therefore the context X_s at step s of the conversation is the position C_s in T. We assume all interactions start at the root node C_0. X_s is meant to represent the current topic of conversation at step s, according to the last answer provided or the latest click on the navigation menu. By supporting contexts of length > 1, it is also possible to keep track of previous topics of conversation.
Each node C_i has a corresponding nonempty set of topic-related *keywords* W_i.
An important distinction is drawn between *terminal nodes* C^Ω and *nonterminal nodes* $C \setminus C^\Omega$. Each terminal node C_j^Ω has a (potentially empty) set of answers A_j corresponding to it. All the A_j sets are disjointed. Let A be the set of all answers: $A = \cup_{j \in 1,\ldots,\omega} A_j$.
In our current implementation, there are 35 classification nodes arranged on 3 levels, 25 of them are terminal ones; the number of answers in the knowledge base is 440, and growing
In the Financial QA chatbot, two types of moves between contexts in C are allowed:

1. To children nodes or root node: using the interactive navigation menu. Context is updated automatically according to the user's selection.
   Example: *You are in the "People" section. Ask me a question or choose one of the following topics:*
   (a) members
   (b) influencers
   (c) contact us
   (d) return to main menu

2. To any terminal node: after answer a_k is provided to the user by the system, new context becomes C_j, where A_j contains a_k.
   Example: after providing the answer *COST_OF_GOORUF = "Gooruf is free, only Premium Providers are required to pay"*, context is changed to *ROOT* $\rightarrow$ *services* $\rightarrow$ *info_about_gooruf*, *info_about_gooruf* being the terminal topic containing answer *COST_OF_GOORUF*.

QA Algorithm Design: question q, its wh-type h, and its current context C_s are passed to the algorithm. Keywords $\mathbf{w}_q$ are extracted from q. If q is anaphoric or elliptic, the algorithm evaluates whether to expand W_q to $\dot{\mathbf{w}}_q$ by using keywords W_s corresponding to C_s. The final representation of q is:

$$R(q) = (C_s, h, \dot{\mathbf{w}}_q).$$

Answers $a \in A$ are described by the following feature vector:

$$F(a) = (C_a^\Omega, H_a, W_a)$$

where C_a^Ω is the classification terminal node corresponding to a, W_a the corresponding keywords and H_a a set of related wh-type (for example a user might inquire about Gooruf by referring to it as a *what* or a *who*).
Relevance r_k for each answer a_k is computed, by considering the classification structure T as well, therefore:

$$r_k(q) = \rho(R(q), F(a_k), T).$$

To compute ρ, scores are calculated separately by comparing contexts (using proximity in T between C_s and C_a^Ω in T), wh-types (h and H_a) and a dense semantic representation of keywords ($\dot{\mathbf{w}}_q$ and W_a) obtained using a Word2Vec model for Italian language (Mikolov et al., 2013); before these partial scores are weighted and summed.

Example: we provide below an example of Financial QA interaction which shows how managing hierarchical context helps in accomplishing

the question answering task.
A subtree *taxes* of T models Italian taxes-related topics:

- *taxes* $\rightarrow$ *city_taxes*
  - *TARI*
  - *TASI*
- *taxes* $\rightarrow$ *income_taxes*
  - *IRPEF*
  - *IRAP*

Individual taxes are represented as terminal nodes in T. Let $C^{\Omega}_{\text{taxes}} = \{$*IRPEF, IRAP, TARI, TASI*$\}$. Each $t \in C^{\Omega}_{\text{taxes}}$ has associated answers: "*HOW_TO_PAY* t", "*WHERE_TO_PAY* t", "*AMOUNT_TO_PAY* t".
The interaction could go as follows:

H: How can I pay city taxes?
QA Algorithm detects wh-type *how*, keywords matching the *city_taxes* node, finds two relevant answers in children nodes and Chatbot asks for clarification.
C: Did you mean TARI or TASI?
H: the second one
Chatbot presents answer *"HOW_TO_PAY TASI"*
Context is now *TASI*.
H: and where can I pay it?
QA Algorithm detects wh-type *where* and completes the question with context knowledge.
Chatbot finds a single relevant answer and presents answer *"WHERE_TO_PAY TASI"*.
H: how much IRPEF should I pay?
Chatbot presents *"AMOUNT_TO_PAY IRPEF"*.
Context is now *IRPEF*.
H: where can I pay it?
Chatbot presents *"WHERE_TO_PAY IRPEF"*.

4 Conclusions and Further Work

We are currently in the process of evaluating Financial QA according to a framework based on PARADISE (Walker et al., 1997; Rieser and Lemon, 2011), which considers, among the others, the following indicators: Task Ease, NLU Performance, Expected Behavior, Presentation, Verbal Presentation, Future Use. We plan to finalize our evaluation in the next months.
NLP is crucial for the development of robust chatbots; since extra-linguistic elements are potentially very important in intent understanding, moving from semantics to pragmatics is a necessary step to develop next-generation chatbots. We have shown how Dialogue Management can support a more robust handling of context, at least in closed-domain QA tasks.
Further work is required to handle more business cases and a broader definition of context, such as history of activities conducted by the same user, which can be especially useful in chatbots with recommender functions (Lombardi et al., 2009).
We would like to thank Andrea Bolioli for his help in the review phase and all of our CELI colleagues for their invaluable work and support.

References

John L. Austin. 1962. *How to Do Things With Words*. Oxford University Press.

James F. Allen, Donna K. Byron, Myroslava Dzikovska, George Ferguson, Lucian Galescu and Amanda Stent. 2001. *Toward conversational human-computer interaction*. *AI magazine*, 22(4):27–39.

Antoine Bordes and Jason Weston. *Learning End-to-End Goal-Oriented Dialog*. 2016. arXiv:1605.07683 [cs.CL].

Benatallah Boualem, Fabio Casati, and Farouk Toumani. 2004. *Web service conversation modeling: A cornerstone for e-business automation*. *IEEE Internet computing*, 8(1):46–54.

Andrew Breen 2014. *Creating Expressive TTS Voices for Conversation Agent Applications*. *International Conference on Speech and Computer*. Springer.

Marcos Aurélio Domingues, Marcelo Garcia Manzato, Ricardo Marcondes Marcacini, Camila Vaccari Sundermann and Solange Oliveira Rezende. 2014. *Using contextual information from topic hierarchies to improve context-aware recommender systems*. *Pattern Recognition (ICPR), 2014 22nd International Conference on*. IEEE, 3606–3611.

Paul Grice. 1975. *Logic and conversation*. New York Academic Press, 41–58.

Zhiheng Huang, Marcus Thint and Zengchang Qin. 2008. *Question Classification Using Head Words and Their Hypernyms*. *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

James Lester, Karl Branting and Bradford Mott. 2004. *Conversational agents*. *The Practical Handbook of Internet Computing*, 220–240.

Sabrina Lombardi, Sarabjot Singh Anand and Michele Gorgoglione. 2009. *Context and customer behaviour in recommendation.*

Michael Mauldin. 1994. *ChatterBots, TinyMuds, and the Turing Test: Entering the Loebner Prize Competition*. *AAAI Press - Proceedings of the Eleventh National Conference on Artificial Intelligence*.

Tomas Mikolov, Kai Chen, Greg Corrado and Jeffrey Dean. 2013. *Efficient Estimation of Word Representations in Vector Space.* arXiv:1301.3781 [cs.CL].

Rambow Owen, Srinivas Bangalore and Marilyn Walker. 2001. *Natural language generation in dialog systems. Proceedings of the first international conference on Human language technology research.*

Carston Pask. 1975. *Conversation, cognition and learning*. New York: Elsevier.

Diana Perez-Marin. 2011. *Conversational Agents and Natural Language Interaction: Techniques and Effective Practices: Techniques and Effective Practices.* IGI Global.

Silvia Quarteroni and Suresh Manandhar. 2007. *A Chatbot-based Interactive Question Answering System*. *Proceedings of the 11th Workshop on the Semantics and Pragmatics of Dialogue.* Edited by Ron Artstein and Laure Vieu.

Verena Rieser and Oliver Lemon. 2011. *Reinforcement learning for adaptive dialogue systems: a data-driven methodology for dialogue management and natural language generation*. Springer Science & Business Media.

Carston Robyn. 2002. *Thoughts and Utterances: The Pragmatics of Explicit Communication*. Oxford Blackwell.

Sophie Rosset, Delphine Tribout and Lori Lamel. 2008. *Multi-level information and automatic dialog act detection in human–human spoken dialogs*. *Speech Communication*, 50(1):1-3.

Francesco Tarasconi and Vittorio Di Tomaso. 2015. *Geometric and statistical analysis of emotions and topics in corpora*. *IJCoL - Italian Journal of Computational Linguistics*, Vol.1 n. 1. Accademia University Press.

Chen Xing, Wei Wu, Yu Wu, Jie Liu, Yalou Huang, Ming Zhou, and Wei-Ying Ma. 2017. *Topic Aware Neural Response Generation*. *AAAI*, 3351–3357.

Gu Yue. 2017. *Speech Intention Classification with Multimodal Deep Learning*. *30th Canadian Conference on Artificial Intelligence, Canadian AI 2017, Proceeding*. Springer.

Marilyn A. Waler, Diane J. Litman, Candace A. Kamm, and Alicia Abella. 1997. *PARADISE: A framework for evaluating spoken dialogue agents*. *Proceedings of the eighth conference on European chapter of the Association for Computational Linguistics*. Association for Computational Linguistics.