A framework for rapid development of conversational natural language call routing systems for call centers

Ea-Ee Jan, Hong-Kwang Kuo, Osamuyimen Stewart, and David Lubensky

IBM T.J. Watson Research Center, Yorktown Heights, NY
ejan@us.ibm.com, hkuo@us.ibm.com, ostewart@us.ibm.com, davidlu@us.ibm.com

Abstract

A framework for rapid development of conversational natural language call routing systems is proposed. The framework cuts costs by using only scantily prepared business requirements to automatically create an initial prototype. Besides clear targets (terminal routing classes), vague targets which are variations of users’ incomplete (semantically overlapping) sentences are enumerated. The vague targets can be derived from the confusion set of the semantic tokens of the clear targets. Also automatically generated for a vague target is a disambiguation dialogue module, which consists of a prompt and grammar to guide the user from a vague target to one of its associated clear targets. In the final analysis, our framework is able to reduce the human labor associated with developing an initial natural language call routing system from a few weeks to just a few hours. The experimental results from a deployed pilot system support the feasibility of our proposed approach.

Index Terms: spoken language processing, call routing application, speech recognition application

1. Introduction

Anecdotal evidence points to the cost of human labor as one of the biggest impediments in the wider adoption of speech and/or natural language technology in call center automation. The trend in the enterprise has been to try to match the web offering with the telephone channel. This has placed tremendous burden on the traditional telephone application involving touch-tone interactive voice response (IVR) menus because of the large inventory of products or services that should be presented to the user as menu options. For these applications that contain a large number of menu options, natural language call routing technology has become the preferred automation strategy. Natural language (NL) refers to applications that have no explicit menus (of items offered) but simply has an initial open-ended prompt asking the user the reason for their call and they are not instructed or constrained in terms of what they can or cannot say. The typical example of this opening prompt is “Hi, Welcome to XYZ Corporation, I am an automated assistant here to direct your call. How may I help you?” In response to the prompt, users may freely describe their requests in their own words.

This approach provides a natural human-machine interaction and reduces the burden on users to go through a lengthy tree structure of menu options, especially if the menu options contain over a hundred choices. However, there is now a growing tension between the increasing desire to replace the traditional menu-driven touch tone IVR with natural language call routing solutions and the cost of human labor associated with such implementation. There are, at least, six labor and cost intensive phases in the development of NL applications: (a) data collection, (b) writing specifications on how to label the data, (c) hiring and supervising annotators to label the data, (d) developing the NL model, (e) developing complementary disambiguation grammars (fixed grammars), and (f) writing the associated disambiguation dialogue prompts. There is convergence on Wizard of Oz as the strategy for handling (a) [1], and the approach to solving (d) is based on developing algorithms [3], however (b), (c), (e), and (f) are all still pretty much heavily manual and human labor intensive.

The problem that arises is that call center operators are often unwilling to invest in this laborious and expensive process. It is not unusual that a call center operator would provide a page or two describing the types of routing targets desired, and expect the engineer to build an initial system based solely on the short specification. In this case, when the topics involve a large number of classes spanning a complicated semantic space, one cannot easily and clearly define the classes of options due to semantic overlaps in the meanings of words and alternate short cuts through which callers can state their requests.

In this paper, we make a first attempt to deal with exactly this real-life problem and address the following two questions: How does one field an initial call routing system with practically no data, except the short specification from the customer? How can we automatically or semi-automatically generate disambiguation grammars and corresponding prompts? We propose a unified framework for creating class definitions tangential to the NL model, and for automatically generating grammars and associated prompts for vague classes. It will be shown that this approach reduces the human labor and consequent cost currently associated with the design and development of a deployed conversational natural language call routing application. The framework consists of two steps: 1. Designing clear and vague target classes effectively from the business requirement. 2. Automatic and/or semi-automatic creation of disambiguation prompts and grammars for each vague class.

2. Generation of clear target and vague target classes

Typically, a customer will provide a list of items (e.g. services or problems) that they support. In our framework, we assume that this list offers a definition of the call routing classes to be included in the NL model and use this list as the basis for extrapolating or generating the relevant information: clear and vague targets. Suppose, for example, that a system supports ‘Microsoft Windows password reset’, ‘Lotus Notes password reset’ and ‘Linux password reset.’ These three items are the clear target classes. Thus, a clear target contains the required semantic categories for expressing the caller’s request, e.g., “I want to reset windows password,” where we know the product or the system the caller is having a problem.
with “Windows” and a specification of the actual problem or solution or action desired “password reset.” Clear target classes are typically the terminal nodes of the tree describing the call routing options and they are commonly provided as business requirements. Continuing with this example, a vague class will be a request like “I need to reset my password” where the product or system’s name (“Windows”) is missing from the caller’s natural language input. According to our framework, this request is classified as “password reset” vague. So, a vague category only contains partial semantic information for fulfilling a caller’s request [2]. A vague target class is associated with multiple clear target classes. It can be, but is not limited to, the intermediate nodes in a menu tree. Vague target classes are artificial classes from the overlap of clear targets in a given semantic space. They are commonly manually designed and maintained by the business analyst and Voice User Interface (VUI) designer. This usually takes about 1 to 2 weeks of effort depending on the complexity of the application. Further adding to the complexity of handling vague targets is the fact that when there is a vague target, then a disambiguation dialogue module is activated—this allows the user to be prompted in a directed manner to identify and gather the missing piece(s) of information for routing the call. A disambiguation prompt and grammar contains an enumeration of all the possible clear target matches. In the example of “password reset” vague, the module will prompt for system name: Microsoft Windows or Lotus Notes or Linux?” In parallel, a disambiguation grammar is used to recognize user response from the list of options.

We will use the following example to explain our proposed framework and approach for clear and vague class generation. Assume a natural language call routing system is designed to support 8 FAQs containing the following issues:

1. Unable to create a data source for Excel
2. How to Unhide hidden columns for Excel 2002
3. How to Unhide hidden rows for Excel 2002
4. How to hide watermarks or background for Excel 2000
5. How to remove password from Workbook for Excel 2000
6. How to create a password for Microsoft Windows
7. How to reset password for Linux operation system
8. How to reset password for Lotus Notes

Q1 to Q8 denote the FAQ questions 1 to 8. By definition, each FAQ is a clear target class. To determine whether a vague class may potentially arise from the intersection of semantic concepts in two clear classes, it is useful to first define the concept of semantic tokens. A semantic token is a key word or phrase that is important to the definition of a class. These tokens can be extracted by different approaches. It can be as simple as a text processor that performs word stemming, combines words into phrases, and removes filler or stop words. In our example, the semantic tokens for the above FAQs are:

- (Q1) create data_source excel
- (Q2) how to unhide hidden columns excel 2002
- (Q3) how to unhide hidden rows excel 2002
- (Q4) how to hide watermarks or background excel 2000
- (Q5) how to remove password workbook excel 2000
- (Q6) how to create password microsoft_windows
- (Q7) reset password linux
- (Q8) reset password lotus_notes

Any semantic token appears in the definition of multiple clear classes, they can be used to identify vague classes. From the above example, the semantic token of “password” appears in the classes Q5, Q6, Q7, and Q8. One can then define the vague class of “password” and its associated confusion set of clear targets {Q5, Q6, Q7, Q8}. A dialogue module to disambiguate Q5, Q6, Q7, and Q8 is needed. In addition, “reset password” which is common to Q7 and Q8 is another vague class. (Note that “create password” is not a vague target, it refers to Q6 only.) The sequences of “excel 2002”, “excel 2002 unhide hidden” and “excel 2002 unhide hidden” refer to the same vague class which involves Q2 and Q3 and so only one vague class needs to be defined. We choose to use the largest set of semantic tokens common to the multiple clear targets in the confusion set to specify the vague class. Thus “excel 2002 unhide hidden” defines the vague class that includes Q2 and Q3.

In the above example, 6 vague classes, including “create” (Q1,Q6), “excel” (Q1,Q2,Q3,Q4,Q5), “excel 2002 unhide hidden” (Q2,Q3), “excel 2000” (Q4,Q5), “password” (Q5,Q6,Q7,Q8), and “reset password” (Q7,Q8) can be extracted. The vague classes automatically generated by the machine can then be checked and refined by the business analyst or user interface expert. For example, the “create” vague class would likely be deleted because it is unlikely someone would say they want to create something without saying what they want to create.

Based on this example, it should become obvious that for a human to directly define vague classes from the initial specification of the clear classes is extremely cumbersome and prone to omission. A semi-automated procedure such as our proposed framework is necessary. As has been noted in [2], the majority of users who call NL applications state their requests using vague utterances. Against this background, the accuracy of defining vague classes is essential to successful NL call routing applications. Subsequently, the derived clear classes consisting of all the clear and vague targets are used to train the Action Classifier, i.e., the NL algorithm that handles intent determination. Since a clear target is a terminal node, all clear targets are routed to the appropriate destination (automation or human specialist). In contrast, vague targets require further clarification or disambiguation to gather the full set of information required for routing the call. In the next section, we describe an attempt to automatically generate the prompts and grammars for the disambiguation dialogue to deal with vague classes. The generated prompts and grammars can be further edited by UI experts, for example to make prompts sound more natural. Having the automatically generated prompts and grammars for each vague class provides an organized framework for the UI expert and can save time and effort.

### 3. Prompt and grammar generation for dialogue module

As stated above, all vague targets (upon recognition) are sent to the disambiguation dialogue module. The disambiguation module consists of a prompt and a grammar for each vague class. The purpose is to guide the user to the final clear target class and subsequent call routing. Some vague targets may involve a small number of clear targets, so there are a limited number of options to pick from. Typically, this range between two and three options, and a simple directed dialogue sub-module would be sufficient. However, most vague targets involve a large number of clear targets, with four options or more. In such cases, it is just too
cumbersome, laborious and cognitively intensive to enumerate all the possible clear targets in the prompt. Thus, creating such prompts (and corresponding grammars) involve heavy customization. This situation is compounded further by the fact that the user’s response may lead to a subset of the vague target in the disambiguation process. This necessitates and justifies a recursive procedure rather than manual. In current practice, the prompts and grammars are designed by the VUI expert and speech scientist, respectively. Needless to say, tremendous effort is required for designing these prompts and grammars, especially when the number of vague classes is large. This underscores the need for a semi-automatic process for creating prompts and grammars, which is a nice derivative of our proposed framework which we will now illustrate.

Typically, a disambiguation prompt includes two parts. The first part contains an implicit confirmation which summarizes the user’s intent from the preceding query. The second part enumerates possible solutions to the user based on missing options or variables from the initial utterance. From the example in Section 2, suppose the user says: “I forgot my password. I cannot login. Could you help me to reset my password?” This query is routed to the “reset password” vague class. To generate the implicit confirmation, we propose a prompt generation procedure that uses the vague target description, e.g., since “reset password” is the core semantic token sequence used to define this particular vague class, this token can be used as the input to generate the first part of the prompt. A simple template, such as “OK, you need help on xxx”, (where xxx refers to the vague target) can be a good strategy.

To generate the second part of the prompt that enumerates the options to the user, we propose to extract the longest signature semantic tokens as the required features for the prompt. The term “signature” [4] refers to a subsequence of the semantic tokens in a class definition that uniquely identifies it. As an illustration, lotus notes and linux are signatures for Q6, Q5, respectively. In addition, remove, workbook and 2000 and any combination of these three tokens are signature to Q3 for “password” vague. We need to use the longest signature as features to create the prompts because it carries the most information. A simple template method can also be used to generate the second part of the prompt. As another example, if the user says: “show me how to unhide excel?”; this query maps to the vague class of “excel 2002 unhide hidden”. Q1 and Q2 are the clear targets associated with it, and the signature for Q1 and Q2 are columns and rows, respectively. The prompt can be as simple as “OK, you need help on excel 2002 unhide hidden, do you need rows or columns?” This becomes the input for automatic prompt generation. As mentioned earlier, after the automatic generation phase, a UI expert is allowed to easily modify any prompt that may sound a little awkward.

If the vague class leads to too many clear classes, it may be too cumbersome to list all the possible solutions. For example, based on our example, when the user says: “I need help with my password,” it maps to “password” vague, which contains 4 clear classes (Q5,Q6,Q7,Q8). Different strategies have been used today for this scenario. The system can reply with the fact that the user’s response may lead to a subset of the vague target. However, we have many solutions related to password issues. Can you be more specific?” [5] However, in our framework, we want to create a prompt leading to a maximum reduction of expected entropy in order to try and minimize the number of dialogue turns. To achieve this, the semantic tokens from all clear targets associated with the vague class are collected. One or two semantic token sequences with maximum length which separates these clear classes into several subspaces with similar sizes are chosen as the relevant features for creating the prompt. In other words, we propose to separate the vague classes into several sub-semantic spaces. For example, the resulting prompt may be: “OK, you need help on password. Is this password reset or something else?” [5] Which ever way the caller answers, the number of remaining possible clear targets will be reduced in half. Specifically, for the “password” vague class, using “password reset” as a sub-semantic new group is sufficient to separate “Q5, Q6, Q7, Q8” into “Q5, Q6” and “Q7,Q8”. This approach mitigates the cognitive load on the user who might be easily confused and irritated by another open-ended prompt.

The grammar for each vague class is created by the algorithm in [4]. The target class definition is used as the input data to the algorithm. Here we propose an improvement to this algorithm by taking only the non-fillers, i.e., all the content words, from the class definition to calculate unique signature and confusion set. With this improvement, the grammar is more concise with better performance. In our offline experiments with alternative approaches where the grammar contains filler words, we observed that the grammars were heavily (negatively) biased to the fillers. Continuing further, the main body of the grammar associated with “password” vague class, which contains Q3,Q4,Q5 and Q6, is illustrated in Table 1. The natural language cues, such as “I need help on”, “could you help me with” and “please”, can be added upon this main body for better grammar coverage. From Table 1, this grammar supports “reset” or “reset password,” which leads to a vague class containing Q5 and Q6. “something else” supports a new vague class of Q3 and Q4. With this approach, prompts and grammars can be generated to interact with the user recursively until a clear class or maximum number of re-tries has been reached. This grammar is intentionally over generated to support many possible queries related to Q3,Q4,Q5 and Q6.

<table>
<thead>
<tr>
<th>Table 1. disambiguation grammar for “password” vague class</th>
</tr>
</thead>
<tbody>
<tr>
<td>reset :Q8::Q7</td>
</tr>
<tr>
<td>something else :Q5::Q6</td>
</tr>
<tr>
<td>reset password :Q8::Q7</td>
</tr>
</tbody>
</table>
4. Generation of language model and call routing model

A variety of methods may be used to bootstrap the language model (LM) used by the speech recognizer and the call routing model for classifying the intent of the caller to one of the target classes. Such methods include model adaptation of a background domain independent model with a small amount of domain dependent data. We will not discuss this topic much since it is not a major focus of this paper. We note only that we implemented a simple scheme whereby semantic tokens extracted from the class definitions in the business requirement specification are used to bootstrap a statistical call classifier. The call classifier was used to filter a large pool of LM training data to reject unrelated sentences and keep sentences that contain some of the key semantic tokens. Natural language grammars are also created to enumerate additional text data for LM. Interpolation with a general domain-independent LM may improve robustness. A standard 3-garm language is used in our experiments.

5. Results and discussion

A deployed conversational IT technical help desk call routing application was used to evaluate our approach. The given business requirement contains a total of 63 clear target classes. Each class came with one or two sentences of class definition to describe the scope of the class. 80% of the clear target classes severely overlap with each other in semantic space as many of them involved multiple products. As a benchmark, only 25 vague classes were created manually by a VUI designer for this data set. Our proposed approach extracted 315 semantic tokens from the clear and vague class descriptions and yielded 52 vague classes, of which 15 perfectly matched the human labeled vague classes. 6 vague classes out of the 52 were over generated. Two vague classes were missed entirely by the human expert, but not by the automatic approach; the human expert inspected and verified that the two vague classes were valid. The mismatches with the remaining 29 vague classes stemmed from insufficient context in the business requirement, although they were within the scope of the remaining 10 human labeled vague classes. This process took just a few hours compared to the 1-2 weeks required by the VUI expert who would typically do this manually.

A call classifier, supporting these generated classes (63 clear and 52 vague) was developed. The description of classes supporting multiple products was broken into multiple sentences to make one sentence for each product. It came to a total of 228 sentences; thus 1 to 2 sentences per class. To avoid unnecessary bias, text was stemmed and all filler words were removed before training the call classifier. The test set was collected by asking some subjects to write down their text queries based on descriptions of IT problems for each class definition, including clear and vague classes. The call classifier yielded approximately 22% errors using this text test data. Many errors are from the vague class and those clear targets with multiple products as expected. This performance was sufficient to support pilot of the application.

A pilot system was deployed using this call classifier along with the LM (not discussed here), grammars and prompts, although the last two were created manually. The performance was monitored and reviewed for about 2 months. A total of 18000 calls were analyzed. Approximately 80% of the calls were routed to a clear target. The rest of the calls were abandoned for various reasons.

In the future, additional detailed analysis by validating audio transcription and application logs will be conducted to determine the call routing accuracy of this live system. Meanwhile, this system was sufficient for a pilot deployment and data has been collected for future tuning. Finally, since the automatic grammar and prompt generation approach proposed in this paper was not used, the quality of automatic prompt and grammar generation remains to be evaluated independently.

6. Conclusion and future work

It is sometimes difficult to have a huge up front investment to develop a robust call routing application prior to full deployment. We can deploy a sufficiently robust pilot system, using just a terse business requirement specification containing a list of routing categories to analyze customer behavior and to generate class definitions for the call classifier. Our proposed method enables us to develop NL applications with much less effort compared to the conventional manual human labor intensive approach. In particular, the vague target creation and prompt/grammar creation saves time and labor expended by business analysts and VUI designers.

Many components described in this paper can be improved in future work. For example, in this paper, a simple text pre-processing was used to extract semantic tokens from the class definitions. Manual effort was required to handle “change password” and “reset password” as equivalent concepts. In the future, a POS tagger and WordNet will be explored for better semantic token extraction. A statistical natural language generation using the longest signature and confusable semantic tokens can also be explored for more natural prompt creation.

7. References


