# A Multi-modal Approach to Concept Learning in Task Oriented Conversational Agents

**Toby Jia-Jun Li**
tobyli@cs.cmu.edu
Carnegie Mellon University
Pittsburgh, PA

**Marissa Radensky**
mradensky19@amherst.edu
Amherst College
Amherst, MA

**Tom M. Mitchell**
tom.mitchell@cs.cmu.edu
Carnegie Mellon University
Pittsburgh, PA

**Brad A. Myers**
bam@cs.cmu.edu
Carnegie Mellon University
Pittsburgh, PA

## ABSTRACT

A major challenge in designing conversational agents is to handle unknown concepts in user utterances. This is particularly difficult for general-purpose task-oriented agents, as the unknown concepts and the tasks can be outside of the agent's existing domain of knowledge. In this work, we propose a new multi-modal mixed-initiative approach towards this problem. Our agent PUMICE guides the user to recursively explain unknown concepts through conversations, and to ground these concepts by demonstrating on the graphical user interfaces (GUIs) of existing third-party mobile apps. PUMICE also supports the generalization of learned concepts to other different contexts and task domains.

## CCS CONCEPTS

• **Human-centered computing** → **Natural language interfaces**.

## KEYWORDS

task oriented conversational agents, multi-modal interfaces, concept learning, programming by demonstration, natural language programming

*Submitted to CHI 2019 Workshop on Conversational Agents: Acting on the Wave of Research and Development,*

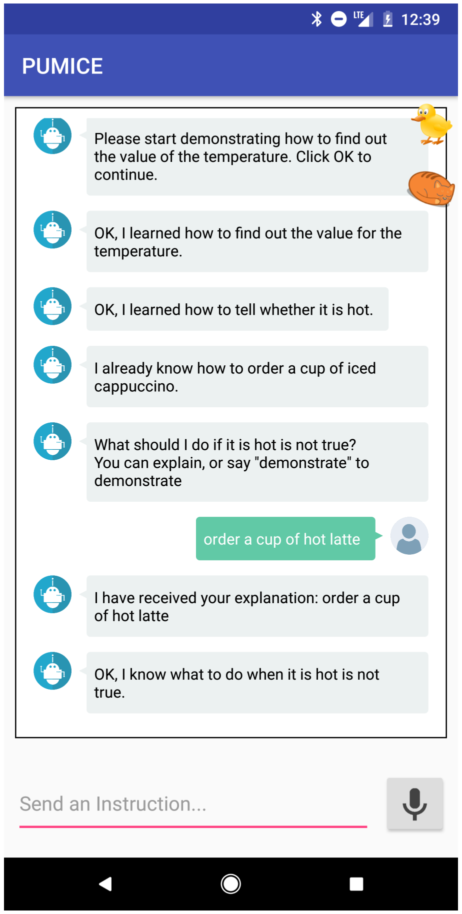


**Figure 1: A screenshot of the conversational interface for the current prototype of the PUMICE agent**

## INTRODUCTION

General-purpose task-oriented conversational agents such as Apple Siri, Google Assistant and Amazon Alexa, have become increasingly popular with end users. With the help of available third party "skills", they help users complete a wide range of tasks via a conversational interface. However, a major challenge for such agents is to handle out-of-domain or unknown concepts in user utterances [6]. These agents often have a set of knowledge (also known as the ontology) that contains concepts and procedures that they can understand and process. When they encounter an unknown concept or procedure, most existing agents perform a fallback action, such as a web search on the user utterance, or a semi-helpful action if the utterance is partially understood (e.g., Siri shows a list of nearby Starbucks stores on Apple Maps when asked to order a cup of coffee from Starbucks). These fallback strategies are not ideal, because they do not fulfill the user's original need, and do not enable the agent to learn the unknown concepts and procedures for future use.

A better approach is to have the agent learn unknown concepts and procedures from end users [6]. We have already made progress in developing agents that learn new procedures from user demonstrations and app usage traces (e.g., [4, 5, 7, 8]) and new concepts within constrained domains from user verbal explanations (e.g, [1, 3, 15]). However, these approaches are limited.

For example, our previous SUGILITE [4] multi-modal agent can learn new procedures (e.g., order a cup of coffee from Starbucks) from users' demonstrations of tasks using third-party mobile apps, and generalize the procedures through parameterization (e.g., learn how to order different kinds of beverages available in the Starbucks app from a demonstration of ordering Cappuccino) based on users' verbal descriptions of tasks. However, SUGILITE can not learn declarative concepts involved in the control structures of tasks (e.g., the concept of *hot* in "if it is hot, order iced coffee"), nor can it generalize learned concepts to other task domains. SUGILITE also requires users to completely and accurately describe tasks in natural language before demonstrating for parameterization to work, which is not natural for end users according to our formative study (more details later).

The *learning from instruction* approach (also known as natural language programming [10, 11] in some research communities) used in LIA [1, 3] enables users to define declarative concepts and control structures (e.g., *important email* in "if an email is important, forward it to my assistant.") by verbal instructions (e.g., important emails are those from my supervisor or which contain the word "important" in the subject). However, this approach is limited to domains where the agent has prior knowledge in, since the agent needs to understand concepts used in the user's instructions (e.g., *sender* and *subject* of emails in the prior example). Thus this approach is not suitable for completely out-of-domain tasks where the agent has little prior knowledge, or for learning the "long-tail" of highly personalized concepts that are hard for users to explain using what the agent already knows.

In this position paper, we describe our ongoing work of designing and implementing a new conversational agent named PUMICE[1] to address limitations of the above approaches. In PUMICE, users first describe the desired tasks and control structures naturally in natural language from a high level, and then collaborate with the agent through conversations to explain and define any ambiguities, unknown concepts and new procedures in the initial descriptions in a top-down fashion. In this process, users can explain concepts by referring to either previously defined concepts, or contents from GUIs of third-party mobile apps, facilitating better reusability of existing knowledge. Users can also define new procedures through demonstrating with third-party apps.

[1]PUMICE is a type of volcanic rock. It is also an acronym for **P**rogramming in a **U**ser-friendly **M**ultimodal **I**nterface through **C**onversations and **E**xamples

## FORMATIVE STUDY

We took a *user-centered* approach [12] for designing a natural end-user development system [13], where we first studied how end users naturally instruct tasks with declarative concepts and control structures in natural language for various tasks in the mobile app context through a formative study on Amazon Mechanical Turk with 58 participants (41 of which are non-programmers; 38 men, 19 women, 1 non-binary person).

Each participant was presented with a graphical description of an everyday task for a conversational agent to complete in the context of mobile apps. All tasks had distinct conditions so that each task should be performed differently under different conditions, such as playing different genres of music based on the time of the day. Each participant was assigned to one of 9 tasks. To avoid biasing the language used in the responses, we used the Natural Programming Elicitation method [12] by showing graphical representations with limited text in the prompts. Participants were asked how they would verbally instruct the agent to do the tasks, so that it may understand the difference among the conditions and what to do in each condition. Each participant was also shown an example scenario and the corresponding example verbal instructions.

Participants were randomly assigned into 2 groups. For the experimental group, participants instructed tasks while looking at relevant mobile app GUIs. Each participant was presented with a mobile app screenshot with arrows pointing to the screen component containing information pertinent to the task condition. Participants in the control group were not shown app GUIs.

The participants' responses were analyzed using open coding [16]. 44 out of 174 responses (19%) were excluded from the analysis for quality control due to the lack of efforts in the responses, question misunderstandings or blank responses.

We report the most relevant findings for motivating the design of PUMICE below:

*App GUI Grounding Reduces Ambiguous and Fuzzy Concepts.* We analyzed whether each user's verbal instructions of the task provides a clear definition of the conditions in the task, and the logic for choosing different conditions. In the control group (instructing without seeing app screenshots),

**Table 1: The conversation transcript for the example scenario. "A" indicates the agent; "U" indicates the user.**

| | |
|---|---|
| U: | If it is hot, order a cup of Iced Cappuccino. |
| A: | How do I know whether it is hot? |
| U: | It's hot when the temperature is above 90 degrees Fahrenheit. |
| A: | How do I find out the temperature? |
| U: | Let me show you. |
| A: | Please show me how to find out the temperature using mobile apps. |
| | (User demonstrates in the AccuWeather app...) |
| A: | I have learned how to find out the temperature using AccuWeather. |
| A: | I have learned how to tell whether it is hot using the temperature. |
| A: | How do I order a cup of Iced Cappuccino? |
| U: | Let me show you. |
| A: | Please show me how to order a cup of Iced Cappuccino using mobile apps. |
| | (User demonstrates in the Starbucks app...) |
| A: | I have learned how to order a cup of something using Starbucks. |
| A: | I have learned if it's hot, order a cup of Iced Cappuccino using Starbucks. What should I do if it is not hot? |
| U: | Order a cup of Hot Latte. |
| A: | OK. I have learned to order a cup of Hot Latte if it is not hot. |

33% of the participants used ambiguous or fuzzy concepts in the instructions, such as "If it is daytime, play upbeat music..." although the example they saw had clearly defined conditions.

Interestingly, for the experimental group, where each participant was provided an app screenshot displaying specific information relevant to the task's condition, significantly fewer ($p < 0.05$) participants (9%) used ambiguous or fuzzy concepts, while the rest clearly defined the condition (e.g., before 7 am). The results suggest that end users naturally use ambiguous and fuzzy concepts when verbally instructing task logic. Showing users relevant mobile app GUIs with concrete instances of the values can help them ground the concepts, leading to less ambiguities and fuzziness in their descriptions. The implication is that a potentially effective approach to avoiding ambiguous or fuzzy utterances for agents is to guide users to explain them in the context of app GUIs.

*Assumption of Common Sense Knowledge Understanding.* We observed that the participants often assumed the understanding of common sense knowledge for the agent when instructing tasks. For example, one participant said, "if the day is a weekend". The agent would therefore need to understand the concept of "weekend" (i.e., how to know the day of the week for today, and what days count as "weekend") to resolve this condition.

However, the coverage in the understanding of common sense knowledge is not only lacking, but also impractical in current agents, especially when across diverse domains due to the spotty coverage and unreliable inference of existing common sense knowledge systems. Managing user expectation and communicating the agent's capability is also a long-standing challenge in interactive intelligent systems [9]. A feasible workaround is to enable the agent to ask users to explain new concepts when they come up, and to build up knowledge of concepts over time through its interaction with users.

## PUMICE

Motivated by the formative study results, we designed an early prototype of the Pumice agent for Android phones (Figure 1). It can handle natural language instructions with unknown or fuzzy concepts by allowing users to recursively define these concepts in a top-down process. Through Pumice, end users can start by describing the task and its logic at a high level in a natural style according to the formative study results. A conversational agent would guide the users to further articulate about concepts that require more clarifications or explanations through conversations using a combination of verbal definitions, references to previously defined concepts, and references to existing external app GUIs. This process allows the agent to learn reusable and domain-transferable concepts and procedures in a wide range of task domains without having any prior knowledge in the domain.

We are still in the early process of evaluating the effectiveness, user perception and usability of Pumice through empirical user studies. However, our own initial experiences with the Pumice prototype in different task scenarios suggest that Pumice's approach is promising in helping end users

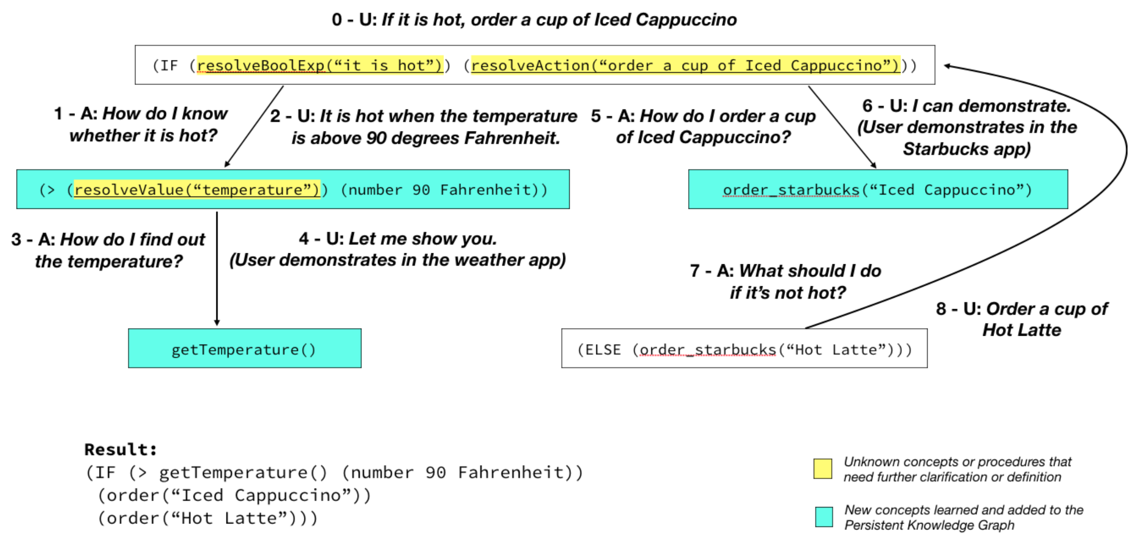


**Figure 2: Example structure of how PUMICE learns the concepts and procedures in the command "If it's hot, order a cup of Iced Cappuccino." In this process, the agent learns getting the current temperature, ordering a drink (any kind) from Starbucks, and the generalized concept of "hot" as "the temperature is above a certain value".**

teach conversational agents unknown or fuzzy concepts and procedures in various task domains. Our past evaluations with conceptually related systems [4, 5] also suggested that PUMICE's high-level multi-modal approach (verbal instructions + demonstrations on existing apps) should be usable for end users without much programming expertise.

### Example Scenario

Figure 2 shows an example structure of how PUMICE learns the concepts and procedures in the command "If it's hot, order a cup of Iced Cappuccino" **with zero prior knowledge** about weathers or coffee. Table 1 shows the corresponding conversation script. At first (Utterance 0 in Figure 2), the agent can only recognize the conditional structure in the user input, due to the lack of knowledge in relevant domains. However, based on the conditional structure, it recognizes that "it's hot" should represent a Boolean expression, while "order a cup of Iced Cappuccino" should be an action.

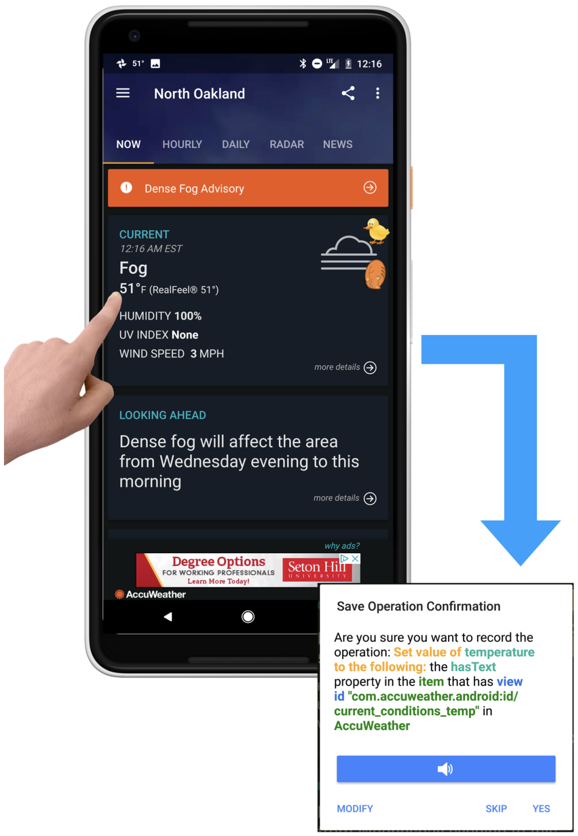


**Figure 3: The user demonstrates how to query for the value of "temperature" on the GUI of the AccuWeather app**

PUMICE recursively resolves these unknown procedures and concepts through conversations, as shown in Utterance 1 in Figure 2. In this case the user provides a definition of "it's hot", which is parsed into a comparison expression. However, PUMICE still does not know the concept of "temperature" nor does it know how to obtain its value. It only knows that "temperature" should be a value that is comparable to "90 Fahrenheit". Thus is asks the user how to find out the temperature.

Here the user can choose to demonstrate how to find out the temperature using an app on the phone that has the temperature displayed in the GUI, as shown in Figure 3. From the demonstration, PUMICE records the procedure to go to where the temperature is displayed and fetch the value, and saves it as a reusable query `getTemperature()` that can be used for querying the temperature in other scripts. At the end, the concepts of "it's hot" and "temperature" are stored in the persistent knowledge base of PUMICE.

For the next step, PUMICE similarly resolves the action "order a cup of Iced Cappuccino" by having the user demonstrate performing the task. Through its underlying SUGILITE framework for handling demonstrations, PUMICE recognizes "Iced Cappuccino" as a parameter in the task, and learns how to order all other available kinds of beverages in the Starbucks app by analyzing the GUI structure of the app. This procedural knowledge is stored persistently as a parameterized procedure `order_starbucks(drinkName)`. Finally, PUMICE proactively asks the user if it should do anything when the condition "it's hot" is not true, because our formative study suggested that end users would often omit *else* statements in commands with conditional structures. To respond, the user can either demonstrate what to do in the *else* condition, or verbally instruct what to do using concepts and procedures that PUMICE has already learned (as shown in the example).

## DISCUSSION

A main takeaway of this position paper should be our new approach for handling unknown concepts in conversational agents – PUMICE guides users to use a combination of verbal instructions and demonstrations to "teach" the concepts, and generalizes them so they can be used in other contexts. In particular, PUMICE's approach provides interesting implications in utilizing pointing and direct manipulation modalities to complement verbal inputs in the design of conversational agents.

### Multi-modal Interaction in Concept Learning for Conversational Agents

While conversational/speech interfaces have many advantages compared to conventional graphical interfaces, such as naturalness, low learning barrier, and support for various ubiquitous computing contexts, they also have limitations that make them ineffective in some scenarios. As a result, speech interfaces have been used in conjunction with other input modalities for a long time, dating back to early pioneer systems like Put-that-there [2] to overcome the limitations.

For example, in our case, supporting out-of-domain concept learning solely from verbal instructions would be very difficult. When a user is asked to explain a concept or a procedure to an agent, the prior knowledge, the internal state and the natural language understanding capability of the agent are invisible to the user. As a result, the user may overestimate the agent (e.g., expect understanding of common sense knowledge, use complex or fuzzy logic, or refer to information that the agent has no access to), or find it challenging to explain concepts using the limited prior knowledge of the agent.

We found that GUIs of existing apps are a good medium through which users can *ground* their instructions. Existing GUIs cover a wide range of task domains for general-purpose task-oriented agents. On the one hand, users are familiar with them, so they are likely able to find appropriate GUI references for concept instructions. In many cases, it is easier for an user to point to something than only verbally explaining. On the other hand, GUIs also provide rich structural and semantic information about the task, which helps the agent understand task contexts and generalize learned knowledge. The other way around, conversational contexts also help enhance the understanding of GUI demonstrations through the pattern of *mutual disambiguation* [14], as shown in our prior work [5]. For Pumice, our design goal is to support seamless coordination between the two modalities, so that users can choose the most natural and the most effective modality for different parts of tasks.

Unlike other popular approaches for supporting multi-modality in conversational agents, like the use of in-dialog cards, our usage of external app GUI demonstration would bring user attention away from the conversational context. So a core challenge would be to provide appropriate guidance to users during demonstrations, so that they can provide more useful inputs. We are currently exploring design opportunities in this area, such as the use of new dialog strategies and visual aids.

## FUTURE WORK

Besides plans on addressing challenges in multi-modal interaction, as covered in the Discussion section, we will conduct various kinds of in-lab and in-situ evaluations to measure the effectiveness, user perception and usability of Pumice's approach. We plan to further extend Pumice's mechanism for generalizing learned knowledge, to raise the ceiling of supported instruction expressiveness so that Pumice can learn about tasks with more complex logic and structures, and to explore issues in the cross-user sharing of learned knowledge. We are also interested in leveraging existing sources of world knowledge (e.g., ConceptNet and Wikidata) to assist users in their instructions of new concepts and new procedures.

## ACKNOWLEDGMENTS

This research was supported in part by Verizon and Oath through the InMind project, and in part by NSF grant IIS-1814472. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.

## REFERENCES


[1] Amos Azaria, Jayant Krishnamurthy, and Tom M. Mitchell. 2016. Instructable Intelligent Personal Agent. In *Proc. The 30th AAAI Conference on Artificial Intelligence (AAAI)*, Vol. 4.

[2] Richard A. Bolt. 1980. "Put-that-there": Voice and Gesture at the Graphics Interface. In *Proceedings of the 7th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '80)*. ACM, New York, NY, USA, 262–270.

[3] Igor Labutov, Shashank Srivastava, and Tom Mitchell. 2018. LIA: A Natural Language Programmable Personal Assistant. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Association for Computational Linguistics, 145–150. http://aclweb.org/anthology/D18-2025

[4] Toby Jia-Jun Li, Amos Azaria, and Brad A. Myers. 2017. SUGILITE: Creating Multimodal Smartphone Automation by Demonstration. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI '17)*. ACM, New York, NY, USA, 6038–6049. https://doi.org/10.1145/3025453.3025483

[5] Toby Jia-Jun Li, Igor Labutov, Xiaohan Nancy Li, Xiaoyi Zhang, Wenze Shi, Tom M. Mitchell, and Brad A. Myers. 2018. APPINITE: A Multi-Modal Interface for Specifying Data Descriptions in Programming by Demonstration Using Verbal Instructions. In *Proceedings of the 2018 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC 2018)*.

[6] Toby Jia-Jun Li, Igor Labutov, Brad A. Myers, Amos Azaria, Alexander I. Rudnicky, and Tom M. Mitchell. 2018. Teaching Agents When They Fail: End User Development in Goal-oriented Conversational Agents. In *Studies in Conversational UX Design*. Springer.

[7] Toby Jia-Jun Li, Yuanchun Li, Fanglin Chen, and Brad A. Myers. 2017. Programming IoT Devices by Demonstration Using Mobile Apps. In *End-User Development*, Simone Barbosa, Panos Markopoulos, Fabio Paterno, Simone Stumpf, and Stefano Valtolina (Eds.). Springer International Publishing, Cham, 3–17.

[8] Toby Jia-Jun Li and Oriana Riva. 2018. KITE: Building conversational bots from mobile apps. In *Proceedings of the 16th ACM International Conference on Mobile Systems, Applications, and Services (MobiSys 2018)*. ACM.

[9] Henry Lieberman, Hugo Liu, Push Singh, and Barbara Barry. 2004. Beating Common Sense into Interactive Applications. *AI Magazine* 25, 4 (Dec. 2004), 63–63. https://doi.org/10.1609/aimag.v25i4.1785

[10] Rada Mihalcea, Hugo Liu, and Henry Lieberman. 2006. NLP (Natural Language Processing) for NLP (Natural Language Programming). In *Computational Linguistics and Intelligent Text Processing (Lecture Notes in Computer Science)*, Alexander Gelbukh (Ed.). Springer Berlin Heidelberg, 319–330.

[11] Lance A. Miller. 1981. Natural language programming: Styles, strategies, and contrasts. *IBM Systems Journal* 20, 2 (1981), 184–215.

[12] Brad A. Myers, Andrew J. Ko, Thomas D. LaToza, and YoungSeok Yoon. 2016. Programmers Are Users Too: Human-Centered Methods for Improving Programming Tools. *Computer* 49, 7 (July 2016), 44–52. https://doi.org/10.1109/MC.2016.200

[13] Brad A. Myers, Andrew J. Ko, Chris Scaffidi, Stephen Oney, YoungSeok Yoon, Kerry Chang, Mary Beth Kery, and Toby Jia-Jun Li. 2017. Making End User Development More Natural. In *New Perspectives in End-User Development*. Springer, Cham, 1–22. https://doi.org/10.1007/978-3-319-60291-2_1

[14] Sharon Oviatt. 1999. Mutual disambiguation of recognition errors in a multimodel architecture. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*. ACM, 576–583.

[15] Shashank Srivastava, Igor Labutov, and Tom Mitchell. 2018. Zero-shot Learning of Classifiers from Natural Language Quantification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, 306–316. http://aclweb.org/anthology/P18-1029

[16] Anselm Strauss and Juliet M. Corbin. 1990. *Basics of qualitative research: Grounded theory procedures and techniques.* Sage Publications, Inc.