CHATTER: A Spoken Language Dialogue System for Language Learning Applications

by

Anna Darling Goldie

S.B. from Massachusetts Institute of Technology (2011)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements of the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2011
[June 2011]

Author .....
Department of Electrical Engineering and Computer Science
May 20, 2011

Certified by .....
Dr. Stephanie Seneff
Senior Research Scientist
Thesis Supervisor

Accepted by .....
Dr. Christopher J. Terman
Chairman, Masters of Engineering Thesis Committee

CHATTER:
A Spoken Language Dialogue System for Language Learning Applications

by

Anna Darling Goldie

Submitted to the Department of Electrical Engineering and Computer Science
on May 20, 2011, in partial fulfillment of the
requirements for the degree of
Masters of Engineering in Electrical Engineering and Computer Science

Abstract

The goal of this thesis is to build a Computer Aided Language Learning game that simulates a casual conversation in Mandarin Chinese. In the envisioned system, users will chat with a computer on topics ranging from relationship status to favorite Chinese dish. I hope to provide learners with more opportunities to practice speaking and reading foreign languages. The system was designed with generality in mind. The framework allows developers to easily implement dialogue systems to allow students to practice communications in a variety of situations, such as in a street market, at a restaurant, or in a hospital. A user simulator was also implemented, which was useful for the code development, as a tutor for the student, and as an evaluation tool. All of the 18 topics were covered within the 20 sample dialogues, no two dialogues took the same path, questions and remarks were worded differently, and no two users had the same profile, resulting in high variety, coherence, and natural language quality.

Thesis Supervisor: Dr. Stephanie Seneff
Title: Senior Research Scientist

Acknowledgments

First of all, I'd like to thank my advisor, Dr. Stephanie Seneff, for her enthusiasm, for her help in refining ideas and fixing bugs, for helping me to write this thesis, for believing in my ideas, for everything. It's been wonderful working with her over the past five years, and I can't even express how grateful I am for all of her support.

I'd also like to Yushi Xu for all of her help in getting the system online, for helping me to navigate the complexities of the Galaxy system, and her help in improving my Chinese and that of the system. Similarly, I am grateful to Seastar Lin and Jackie Lee for helping me to find disfluencies in my system's Mandarin. My fellow MEng students, Chris Varenhort, Jenny, Jieyun Fu, Alice Li have been the best of companions on this MEng journey, as have the rest of the Spoken Language System Group. I feel so lucky to have been a part of this lab.

Thanks go to Rosetta Stone, Fluenz for granting me evaluation versions of their software, allowing me to examine industry-standard language learning software.

I am also beholden to Gabriel Warshauer-Baker, James Goldie, Bob and Betsy from the Writing Center for helping me to make this thesis understandable to the reader. Finally, I'd like to thank all of my family and friends who were there for me throughout the process of this research project. I'm sad to be leaving this chapter of my life, but I'm happy to have had this amazing opportunity.

Contents

List of Figures

Chapter 1

Introduction

The number of people who wish to acquire a foreign language, particularly one such as Mandarin Chinese, far outstrips the number of competent teachers available. Attaining true fluency in another language requires hours and hours of interaction in the target language, but most teachers have too many students to spend much time interacting one-on-one with individual students. Computer Aided Language Learning (CALL) systems offer learners the low-stress, one-on-one interaction they need to become fluent in another language.

The idea for this dialogue system was the product of my experiences in China over two summers. As a foreigner, I found that I was asked the same questions over and over again: "Where are you from?," "Do you speak Chinese?" (silly, but very common), "Can you use chopsticks?," "Do you have a boyfriend?," "How old are you?," "What are you doing here?," "How long have you been here?," "What's your favorite Chinese food?," etc. Being able to answer these questions earned me a great deal of respect, lucrative deals in the marketplace, and opportunities for friendship and more sophisticated conversation.

Particularly in a country where native speakers are so eager to practice English, it is crucial to establish one's proficiency early. I have observed over and over again that, when two people interact, a natural default is to speak whichever common language is the strongest. For example, let's imagine that I am an American taking a trip to Beijing after having studied Mandarin for a couple of years. I have a decent vocabulary and am generally able to express myself, but I lack confidence and fluency. If I encounter a Chinese person with decent English, we will end up speaking English. Sadly, even if my Mandarin is slightly better than the other

person's English, we may still end up speaking English, due to my low confidence and the self-fulfilling perception that non-Asian people are incapable of learning Mandarin. The most insidious part is that, if the two of us speak English, my conversational partner's English will improve over time, while my Mandarin will remain constant (or even decline). Thus, this phenomenon reinforces itself by widening whatever (perceived) gap was present in the very first conversation. With this in mind, it is my goal to develop a system that gives users the experience and confidence to respond to typical questions, enabling them to establish a Mandarin-speaking rapport with Mandarin-speaking people abroad.

Additionally, by developing an intuitive and flexible framework for developers, I hope to promote the higher goal of facilitating the process of developing language learning dialogue systems in a variety of domains. As a result, my project consists of both a framework and a prototype built upon that framework.

The framework is my main contribution, consisting of several components discussed in Section 4.3, while the prototype is merely one instantiation, a proof of concept. Many other prototypes could be built upon this same framework, in many other scenarios such as ordering at a restaurant or booking a flight.

Chapter 2

Previous Work

For the purpose of surveying these systems and understanding where mine might fit into the larger picture, I have chosen a few example dialogue systems to place in a grid along two axes: the purpose of the dialogue system (functional or language learning) and the "strength" of the system's AI, with zero-AI audiotapes on one end of the spectrum and Turing Complete AI on the other.

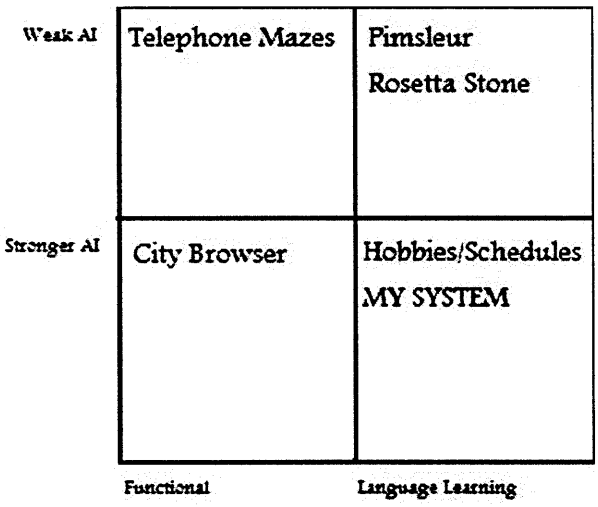

Figure 1: a depiction of two axes of dialogue systems: application and complexity

My system resides in the lower right quadrant, containing strong AI and language learning. This is unusual, since most "strong AI" systems have been confined to functional task completion scenarios. Few language learning solutions employ strong AI.

2.1 Functional Dialogue Systems

Although I am developing a dialogue system for language learning, I cannot ignore systems that were developed for more functional tasks, given the the similarity of the challenges involved. The simplest of such systems are telephone bots, which recognize single words and direct the user through a tree of information and human operators. As an example, a rough transcription of my interaction with the United Airlines' customer service line (800) 864-8331 is given below:

> ***System: Thank you for calling United Airlines. Your call may be monitored for quality assurance. Please select from the following four options: Departure and arrival information, Reservations, Mileage Plus, or other information.***
> *Me: Mileage Plus*
> ***System: I think you said "Reservations". Is that correct?***
> *Me: No*
> ***System: My mistake. Please tell me again.***
> *Me: Mileage Plus.*
> ***System: I think you said "Mileage Plus". Is that correct?***
> *Me: Yes*
> ***System: Please say or enter your 11 digit Mileage Plus number or say "I don't know."***
> *Me: I don't know*
> ***System: Please give me your last name and its spelling. For example, if your last name is Cusock, say "My last name is Cusock, "C" "U" "S" "O" "C" "K". Now give me your last name.***
> *Me: My last name is Goldie, "G" "O" "L" "D" "I" "E".*
> ***System: I'm sorry, I didn't quite catch that. Please spell your last name.***
> *Me: "G" "O" "L" "D" "I" "E"*
> ***System: I'm sorry, I still couldn't understand you. Please say it again.***
> *Me: "G" "O" "L" "D" "I" "E"*
> ***System: I'm sorry, I couldn't catch that. I'm sorry I am having so much trouble. I'm sorry. I will transfer you to a customer service representative.***

The Spoken Language Systems Group at the Computer Science and Artificial Intelligence Lab has been developing spoken language dialogue systems for information access for many years. Example systems include a weather domain [2], flight domain [3], and restaurant domain [4][5][6]. These domains are intended to provide access to information through human-

computer interaction, so they serve as good models for dialogue interaction for language practice. We have utilized many of the tools developed for these systems in the research described here.

2.2 Language Learning Dialogue Systems

Before developing this system, I spent a great deal of time surveying the existing language learning solutions so that I would know where mine fit into the space of solutions. I've observed a broad spectrum of Computer-Aided Language Learning (CALL), ranging from involved voice recognition and dialogue systems to far simpler, "lower tech" systems. Language learning dialogue systems tend to be less complex than their functional counterparts, simulating rather than implementing a true branching dialogue. Truly flexible dialogue systems tend to exist only in academia, since the added cost incurred by this additional complexity is not justified by the potential improvement in performance of the system.

Of all the language learning solutions I surveyed, Pimsleur is probably the oldest. It does not have the flashiest technology, but it is the one which I most consistently recommend to friends who wish to learn foreign languages. The Pimsleur Method™ involves listening and responding to audio prompts for thirty minutes a day. Most lessons culminate in a simulated dialogue between a recorded voice and the user. The "system" asks a question, pauses to give the learner a chance to respond, speaks the correct answer, and then pauses to give the learner time to repeat and reinforce the correct answer before continuing. When the pause is of an appropriate length, the illusion of interactive dialogue is surprisingly strong. Below is an excerpt I transcribed from Lesson 15 of Pimsleur's Italian I:

> *And now a conversation with the woman.*
> *Adesu una conversazione con la senora.*

You're in an agreeable mood, so confirm everything she asks you.
Woman: Buon giorno. Sta bene? | Good day. You are well?
Si, sto bene. | Yes, I am well.
Si, sto bene gracie. | Yes, I am well, thank you.
Woman: Vuole mangiare con me? | Would you like to eat with me?
Si, vorrei. | Yes, I would like to.
Gracie, vorrei. | Thank you, I would like to.
Vorrei mangiare con Lei. | I would like to eat with you.
Woman: Quando? Puo mangiare questa sera? Puo questa sera? | When? You can eat this evening? You can this evening?
Si, posso. | Yes, I can.
Posso questa sera. | I can this evening.
Woman: Puo alle sette? | Can you do seven?
Si, alle sette posso. | Yes, I can do seven.
Woman: Vuole fare qualcosa per me? | Would you like to do something for me?
Si, va bene. Voglio fare qualcosa per Lei. | Yeah, sure. I would like to do something for you.
Woman: Puo comprare del vino per favore? | Could you please buy me some wine?
Si, posso comprare del vino. | Yes, I can buy some wine.
Posso comprare del vino per Lei. | I can buy some wine for you.
Say, yes I can buy some for you.
Si, ne posso comprare per Lei. | Yes, I can buy some for you.
Si, ne posso comprare per Lei. | Yes, I can buy some for you.
Woman: Ha dei dollari? | Do you have American dollars?
Si, ne ho. | Yes, I have some.
Ne ho. | I have some.
Woman: Ne ha molti? | Do you have many of them?
Si, ne ho molti. | Yes, I have many of them.
Ne ho molti. | I have many of them.
Woman: E degli Euro? | And some Euros?
Si, ne ho anche. | Yes, I have some of those also.
Ho anche degli Euro. | I also have some Euros.
Ne ho anche. I have some of those also.
Woman: Allora puo comprare molto vino? | So you can buy lots of wine?
Si, ne posso comprare molto. | Yes I can buy lots of it.
Posso comprare molto vino. | Yes, I can buy lots of wine.
Woman: Allora puo mangiare con me. | Therefore you can eat with me.

The Pimsleur language program works amazingly well, considering that no natural language understanding has taken place. The effectiveness of the program all boils down to one concept called Optimal Interval Recall, in which vocabulary and syntactic structures are re-introduced at the interval which maximizes retention. Although the program is very strong as it

stands, the closer it could come to modeling the user's actual level, the stronger it would be. For example, the length of the pause after a prompt is a very important parameter. If the pause is too short, the user won't have an opportunity to formulate the given sentence before the answer is revealed, robbing him of the reinforcement and making the overall experience more stressful. On the other hand, if the pauses are too long, the student may either lose focus or repeat the response twice, potentially cementing an incorrect response or pronunciation. Given that users progress at wildly different levels, I feel sure that the program could benefit from incorporating information about the user's progress.

At the next higher level of complexity comes Rosetta Stone, the latest version of which (RosettaStone TOTALe™) ends each unit with an interactive dialogue. Unlike Pimsleur, Rosetta Stone employs actual voice recognition capability, telling a story with pictures and audio and prompting the user. If the user does not give the expected response, he or she will not be allowed to continue with the dialogue. Below is a dialogue I transcribed from RosettaStone TOTALe Mandarin Chinese:

> *Picture of a dog, a woman looking at water, a man and woman waving in a campsite*
> Woman: 你好| Hello.
> *Picture of your hand and speech bubble pointer to you*
> **You: 你好| Hello.**
> *Picture of a man pointing to a dog*
> Man: 你有狗吗? | You have a dog?
> *Picture of a dog being petted*
> **You: 有, 我有狗| Yes, I have a dog.**
> *Picture of the woman inviting you to sit on a log, picture of you gesturing to the woman*
> **You: 你在做什么? | What are you doing?**
> *Picture of the woman smiling and opening a book*
> Woman: 我在看书 | I am reading a book.
> *Gesturing toward metal thermos.*
> **You: 这是什么? | What is this?**
> *Woman offers the cannister*
> Woman: 这是咖啡 | This is coffee.
> Woman: 你有杯子吗? | Do you have a cup?
> *Picture of you producing a cup*

You: 有,我有杯子. | Yes, I have a cup.
Picture of you drinking from the cup, picture of the man gesturing to you
Man: 你有什么? | What do you have?
Picture of you producing bread.
You: 我有面包 | I have bread.
You: 你们有什么? | What do you guys have?
Picture of the woman looking in a bag, woman holding the bag,woman holding apples
Woman: 我们有平果 | We have apples.
Picture of apples being sliced, picture of plate being offered, picture of everyone eating bread, picture of dog eating bread
Man: 这只狗在吃什么? | What is this dog eating?
You: 这只狗在吃面包 | This dog is eating bread.
Picture of everyone laughing, you getting up, you hiking away, mountain view

Images are very powerful for conveying a story without language. While accountability is valuable, the system will only accept one particular wording, which is annoyingly narrow and does not encourage learners to use a variety of syntactic structures and vocabulary.

Next, we find systems such as Spoken Language System Group's Hobbies & Schedules system [1], where the user is presented with a schedule and a list of favorite activities and is instructed to coordinate a playdate with the system. Because the role of the system and the user are symmetric, building a simulated user required little effort beyond that which was expended to build the system's artificial intelligence. This simulated user was then used to provide help to the real user during gameplay. At the easiest "rehearsal" level, the user was given the simulated user's response in pinyin, which he or she could read aloud or push a button to have the simulated user say it. At higher levels of difficulty, the simulated user offered less help to the increasingly independent human user.

I modeled my help functionality after the Hobbies & Schedules system by building a simulated user that could play the role of the user. I should note, however, that the simulated user was more difficult to implement for my system, because the system and user have asymmetric roles. Fortunately, the ease of use and flexibility of the natural language processing components

of my framework greatly facilitated the development of my simulated user, and the simulated user, in turn, aided in the development of my dialogue system.

Chapter 3

Defining My Dialogue System

3.1 Dialogues

I would like to start off by distinguishing my system from the typical "chatbot", a text-based IM client which mimics language understanding. Vacuous dialogue systems can be surprisingly appealing to human users. Take the therapist bot Eliza (http://nlp-addiction.com/eliza/), for example. The developer, Joseph Weizenbaum, found that users often mistook Eliza for a human being and became quite emotionally involved in the dialogue before it became apparent that Eliza had no comprehension at all. Below is an excerpt from an interview I conducted with Eliza earlier today.

Eliza: Hello. My name is Eliza. How many I help you?
Me: Hello Eliza. I am trying to write my thesis, and I was hoping you might help me out by answering a few questions about yourself.
Eliza: Do you believe it is normal to be trying to write your thesis, and you was hoping me might help you out by answering a few questions about myself.
Me: Normalcy is irrelevant here, Eliza. My question is: to what extent do you truly understand natural language?
Eliza: Oh... truly understand natural language?
Me: Yes, just answer my question.
Eliza: Are you Sure?
Me: Absolutely.
Eliza: Tell me more...
Me: Stop evading my questions.
Eliza: Come, come elucidate your thoughts.

At which point, of course, my browser crashed. A most slippery character, this Eliza.

My goal in this thesis was to develop a dialogue system which takes a more active role in the conversation, asking the user questions and using the information gleaned to make further inquiries. Although my dialogue model is necessarily simplistic, it is still far deeper than a chat bot like Eliza.

For my system, I built a user simulator that can roleplay the user half of the conversation, following the example of the Hobbies & Schedules system. This simulator is very valuable both for helping the developer build the dialogue partners and helping the student learn what they might say next in the dialogue. Below is a dialogue between my system and the simulated user:

My System: 你 会 说 汉语 吗 | Can you speak Chinese?
User: 我 讲 中文 | I speak Chinese.
My System: 你 汉语 说 得 太 好 了! 你 在 这里 多久了 | You speak Chinese so well! How long have you been here?
User: 我 已经 在 这里 五 天 了 | I have been here five days already.
My System: 你 中文 真的 很 棒! 那, 你 会 用 筷子 吗 | Your Chinese is really great! Can you use chopsticks?
User: 我 会 用 筷子" | I can use chopsticks.
System: 想 不 到! 你 是 法国 人 吗 | Wow, really? Are you French then?
User: 加拿大 人 | Canadian
My System: 好 想 去 你 的 国家! 你 来 中国 旅游 吗 | I really want to go to your country! Did you come here on vacation?
User: 我 是 来 这里 旅游 的 | I came here on vacation.
My System: 好 啊! 你 是 单身 吗? | That's great! Are you single?
User: 我 结婚 了 | I'm married.
My System: 想 不 到! 你们 在 一起 多久 了 | Whoa! How long have you been together?
User: 我们 交往 37 年 了 | We've been together 37 years.
My System: 你 丈夫 很 帅 吗 | Is your husband very handsome?
User: 我 先生 不 是 个 帅哥 | My husband isn't a handsome man.
System: 你 先生 跟 你 一起 来 吗 | Did your husband come to China with you?
User: 他 没有 跟 我 一起 来 | He didn't come with me.
My System: 你 想念 你 丈夫 吗 | Do you miss your husband?
User: 我 想 他 | I miss him.
System: 糟 了! 你 觉得 这里 特别 吗 | That's too bad. Do you find it interesting here?
User: 我 觉得 这里 非常 特别 | I think it's extremely interesting here.
System: 我 很 高兴! 你 还 会 在 这里 待 多久 | I'm glad! How much longer are you

staying?
User: 我 还 会 在 这里 待 两 天 | I'll stay another two days.

3.2 Design Goals

When designing a large system, it is important to outline the properties that are most crucial. Otherwise, how can you make optimal design decisions? And how can you evaluate the extent to which your implementation fulfilled your original goals?

In the case of my system, the goals come in three layers: user-level goals, prototype-level goals, and framework-level goals, as shown in Figure 2.

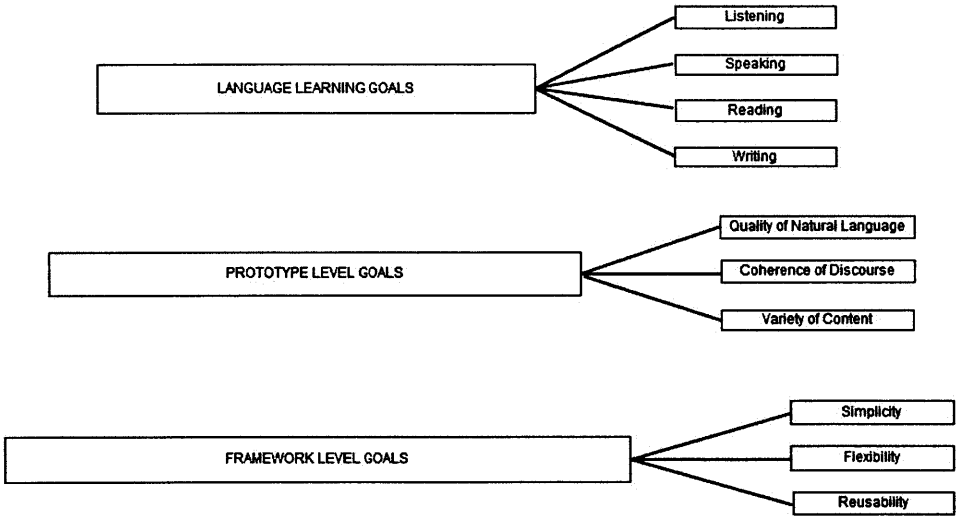

Figure 2: Diagram of the three layers of design goals

Each goal layer is a part of the larger goal, but represents a distinct and meaningful perspective. At the user-level, I formulate the goal in terms of its effect on the language learner. At the prototype-level, I express it in terms of the desired properties of the high-level application with which the user interacts. And finally, the framework-level goals are expressed in terms of

both their ability to support the prototype-level goals and transitively the user-level goals, but also in terms of the developer experience.

3.2.1 Language Learning Goals

These goals are more linguistic than they are related to computer science, but ultimately the most important for a language learning application:

GOAL 1: Language Learning Proficiency - user speaks/read/listens/writes better

At an abstract level, the goal of the system is to help users to attain a higher proficiency in the target language, however one defines proficiency.

3.2.2 Prototype Goals

Below I have enumerated the goals as related to the high level system with which language learning users will interact:

GOAL 2: Syntactic/Phonological - outputs high quality natural language
GOAL 3: Semantics - responds coherently
GOAL 4: Coverage – handles a variety of utterances and dialogues in the domain space

The first goal cannot be compromised, because the system will be used for language learning. It would be truly unacceptable to train people to speak with a non-native accent or incorrect grammar.

The second goal is more challenging, but also more flexible. In order to be engaging, the system must give an illusion of intelligence, of having understood the user's last utterance. However, a strange segue or odd question is not as harmful as an ill-formed utterance would be. In moderation, an occasional "well, how much do you weigh then?" might make the user laugh and help simulate culture shock.

In order to provide an interesting (not boring) dialogue, the system must cover the space of possible questions and transitions as evenly as possible, but with a stochastic component. This will expose the user to a wide variety of structures and queries, and make the game something that can be played over and over again without boredom.

3.2.3 Framework Goals

My goals in designing the framework were more abstract and therefore more difficult to express. The framework goals encompass those of the prototype in the sense that the goal of the framework is to enable the development of a system that provides high quality natural language, coherence, and variety. Rather than condense this high-level purpose into one meaninglessly vague goal, I have divided it into four framework design goals:

GOAL 5: Simplicity - minimizes complexity
GOAL 6: Flexibility - powerful enough to encode realistic and meaningful interaction
GOAL 7: Usability - intuitive interface for developers

Simplicity is the holy grail of all computer science. Wherever I could, I designed the system to have fewer components or to interact in cleaner and simpler ways.

Flexibility or computational power is achieved at the expense of simplicity, although a strong design can minimize the cost of this tradeoff. My goal here was to support just enough flexibility to support prototype level goals, so as to maintain as much simplicity as possible.

In terms of usability, I wrote all of the domain-specific languages with developers in mind, hoping that an intuitive interface would make it easier to extend and build new domains. I also designed modules such that they would be easier to debug.

3.3 Dialogue Loop

My dialogue model is quite constrained, but it allows me to be more flexible in the topics my system can cover, namely any topic I choose to add to my graph of features.

My conversation model is as follows:

A. System asks a question.
B. User answers this question.
C. System responds to user's answer. (Back to A)

The system greets the user, asks any number of questions, and then bids the user farewell.

The conversation loop can be modeled with the following Context Free Grammar:

Greeting (Question Answer [Remark])* Farewell

In order to generate example dialogues for the user and for debugging purposes, I implemented a full user simulator, which answers questions given the user's randomly generated persona.

Now let us work our way through one simple question.

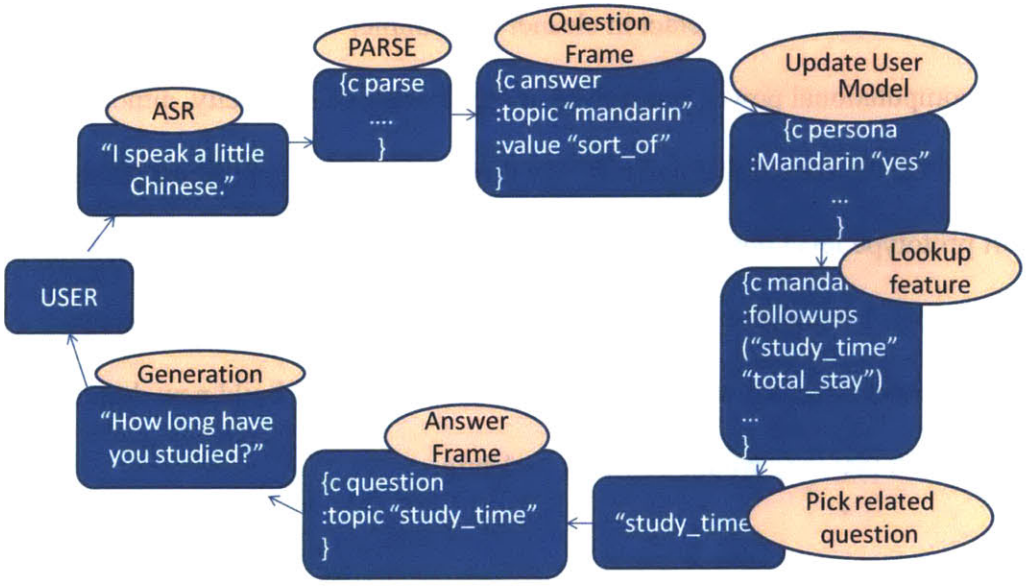

Figure 3: Block Diagram of the control loop for the dialogue system

3.3.1 Entering the Dialogue Loop

The dialogue is turn-based, opening with a question from the shopkeeper, such as "Can you speak Chinese?" At each subsequent turn, the shopkeeper will interpret the customer's response, update its model of the user and of the conversation state, and choose a new question to ask. The control loop is implemented in the form of a dialogue control "dctl" file.

3.3.2 Initialization

As the dialogue is initialized, a "persona" frame is created for the customer, specifying various pieces of personal information, such as gender, age, nationality, and marital status. A truncated instance of the profile is shown in Figure 4.

```
{c professional
 :total_stay {c total_stay :days 4 }
 :accompanied "with_coworkers"
 :nationality "England"
 :reason "business_trip"
 :stay_so_far {c stay_so_far :days 3 }
 :name "Sam"
 :gender "male"}
```

Figure 4: The persona above represents that of an Englishman named Sam who is traveling with coworkers on business. He's been in China for three days already, and plans to stay one more day.

The system also initiates a generic persona in which each field is left unspecified. As the system gains information about the user from his or her answers, it gradually populates its model of the user.

In addition to a model of the user, the system will also initialize a model of the dialogue. Information about what questions have been asked (initially none) and what questions have been

unlocked (e.g. "how many children do you have?" is unlocked if the user indicates that he/she is married) are stored in the form of vectors.

Finally, given the question-answer model for this dialogue, the system needs an initial question to jump start the loop. In this implementation, I chose to specify initial topics of conversation from which the system selects at random. In this example, the system will start off either by asking the user's name, his or her relationship status, or his or her nationality. (By the way, the "so" in ":so_here", ":so_hot", etc. refers to significant other, which may make the topics in Figure 5 easier to understand.)

```
{c eform
 :condList( "6" ":children" ":so_here" ":so_hot" ":so_how_long" ":how_met_so"
":miss_so" )
 :obvious (":gender")
 :background (":relationship_status")
 :initial_question  ( ":name" ":nationality" ":relationship_status")
 :frames_to_load (
        ":persona_file" "persona-frames"
        ":template_file" "chinese_templates.tree"
        ":feature_file" "feature-frames" )
}
```

Figure 5: This figure shows the initialization frame for the dialogue system

3.3.3 Generating a Question in Natural Language

After selecting the topic of its question, the system must present this question in natural language to the user. In this implementation, the system first generates an interlingual question frame from the topic string, encoding its topic and obvious background information that a human would be able to acquire just by looking (e.g. gender).

```
{c eform
  :query {c query
        :gender "male"
        :topic "name" } }
```

Figure 6: A query frame encoding a question about name addressed to a male user. In English this would translate best to "What is your name?"

We then convert the frame into a path which we will use to look up this utterance in a tree of templates. In this case, we look first at the topic (:NAME), then the fact that the utterance is a query (:query), and then that we know the user is male. (In this case the gender of the user is irrelevant, but it can matter when we begin querying topics such as relationship status.)

We look up the path ":NAME->:query->:male" in the structured template and produce the question "你叫什么名字" ["What is your name?"]. One such template for ":NAME" is shown below.

```
:NAME {c eform
        :query ( "你叫什么名字 | What is your name?" )
        :assumption ( "你叫:~name 吗| Your name is :~name?" )
        :answer ( "我的名字是:~name | My name is :~name"
                  "我叫:~name | I am called :~name"
                  "我是:~name | I am :~name" )
        :negation ( "我不叫:~name | I am not called :~name"
                    "我的名子不是:~name | My name is not :~name"
                    "我不是:~name | I am not :~name" )}
```

Figure 7: A generation template for questions and answers about name. For example, the value for ":query" shows how to ask a person's name in Chinese or English.

Note that the English in this example was added for the benefit of those who are not literate in Chinese, although I have collaborated on a project involving a form of bilingual templates which could easily be applied to this system [7].

3.3.4 Interpreting the User's Response

The user hears the system's natural language question, and then speaks an answer in natural language.

:user_utterance (0) "我是山姆" ["I am Sam."]

Given a transcript of the user's response and a record of the last question asked, the system parses the user's response using the TINA parser [8], producing the parse frame below.

```
{c cstatement
        :domain "Phrasebook"
        :input_string "我是山姆"
        :parse_status "FULL_PARSE"
        :topic {q pronoun
                :name "我" }
        :pred {p copular_vp
                :complement {q proper_person
                                :name "山姆" }
                        :copular "是" } }
```

Figure 8: a parse frame for the sentence "I am Sam". This structured representation is derived automatically from the parse tree

The system applies Genesis [9] rules to extract key-values pairs from the parse frame and into a simplified frame called an eform.

```
{c eform
        :agent "我"
        :clause "statement"
        :complement {q eform
                :person "山姆" }
        :predicate "copular" }
```

Figure 9: an eform for the sentence "I am Sam"

The system will flatten this eform using transduction rules stored in the feature frame, verify that the user answered the question, extract the user's answer and store that answer in its

model of the user. All of these steps are orchestrated by a feature frame for "name", a frame which describes various attributes of this particular topic, as well as its relation to other topics, as shown in Figure 10.

```
{c feature
        :feature "name"
  :assumable 0
  :generalizable 1
  :default {c default
        :remarks ("pretty_name")
        :followups ("nationality" "speak_chinese" "relationship_status" "reason" "stay_so_far")}
  :extraction ( (":complement" ":person" ":name" )
           (":*ANYWHERE*" ":loc_name" ":name" )
           (":*ANYWHERE*" ":agent" ":agent" ))
  :verification {c name
                 :requirements ( ":name" )
                 :name ( ">:*SELF*" )}}
```

Figure 10: A feature frame for the topic "name", describing how to extract meaning from the parsed user response ("extraction"), how to verify that he actually answered the question ("verification"), what comments to make after the user responds ("what a pretty name!"), and what followup questions to ask.

In order to extract the most relevant aspects of the parse frame, the system first transduces the utterance, using the topic's extraction list.

Each entry in the extraction list is a triplet, detailing what information to extract from the parse frame and how to store it. The first extraction entry, for example, is (":complement" ":person" ":name"), which tells the system to look for a subframe whose key matches the first element (e.g. ":complement"). Incidentally, "*ANYWHERE*" is a keyword indicating that the system should look for the second entry's value anywhere in this eform. In this case, the subframe in question is:

```
:complement {q eform
            :person "山姆" }
```

Figure 11: A subframe of the eform for "我是山姆| I am Sam"

Within that subframe, the system finds the key which matches the second element in the extraction list (e.g. ":person") and stores its value in the output frame under the name of the third element (e.g. ":name").

Each entry is executed in order until we arrive at the following condensed output frame.

```
{c eform
        :agent "我| I"
        :name "山姆| Sam" }
```

Figure 12: a condensed version of the parse frame

The system then checks to see whether the user has actually answered the question, using the feature frame's "verification" frame, in this case:

```
:verification {c name
            :requirements ( ":name" )
            :name ( ">:*SELF*" )}}
```

Figure 13: a verification frame for "name".

The ":requirements" field is a list of all fields that must be present in the answer for it to be considered a valid answer to the question. In this case, a ":name" field must be present. Furthermore, the ":name" entry in this verification frame has the entry ">:*SELF*"; this is a keyword indicating that the ":name" may have any value that is stored in the template file's entry for ":NAME" (In this case, it contains all values for "name" supported by the system, as shown in Figure 14).

```
:~name   ( "詹姆斯| James" "威廉| William" "山姆| Sam"
                                "安娜| Anna" "斯蒂芬尼| Stephanie" )
```

Figure 14: the generation template's entry for ":name", enumerating all valid values for ":name" in this implementation.

Satisfied that the user has answered the question, the system proceeds to extract the core of the user's answer (e.g. the fact that his name is Sam). The default is to extract the value of the key which was marked as a requirement in the extraction frame. In this case, ":name" is the only key which is required for the user's response to be considered a valid response, and as such this defaults to be the core of the user's answer.

Given this information, the system updates its model of the user and the conversation, storing "Sam" under ":name" in its generic persona and setting "name" as asked in its vector of "asked" questions. In this case, ":name" does not unlock any questions, but if it did, the system would also update its map of licensed questions.

3.3.5 Responding to the User's Answer

The system chooses randomly from a list of remarks contained within the current topic's frame, as shown in Figure 15. In this case, the only available remark is "pretty_name", so this complimentary remark will be converted into natural language using the Genesis language generation framework. [4].

```
:default {c default
        :remarks ("pretty_name")
        :followups ("nationality" "speak_chinese" "relationship_status" "reason" "stay_so_far")}
```

Figure 15: the default response frame for the "name" topic

Next, the system must choose a follow-up question to ask the user. In the case of name, the user's response is unlikely to affect the choice of follow-up question, so this feature has only a default follow-up frame. As shown in Figure 15, after asking the user's name, the system will choose between asking about "nationality", "Chinese ability", "relationship status, "reason for being in China", and "length of stay so far".

The selection of a next topic is at the heart of the system's artificial intelligence, allowing the system to branch to particular topics based on the user's answer. For other more complicated questions, such as relationship status, the system branches differently, as shown in Figure 16, depending on whether the user is married, in a relationship, or single.

```
:responses {c cond
        :single {c single
                :followups ("age" "looking") }
        :married {c married
                :followups ( "so_here" "age" "so_how_long" "children" "miss_so" )}
        :in_relationship {c relationship
                :followups ("so_hot" "so_here" "so_how_long" "miss_so")}
```

Figure 16: a response frame for "relationship_status", if the user says he/she is single, the system will follow-up with questions about age or whether he/she is looking for a partner. If the user says, he/she is married, the system will ask whether his/her spouse is in China, etc.

Given this topic, the system will repeat the generation process described in Section 3.3.3.

3.3.6 Termination

Currently, the system terminates when it runs out of unasked follow-up questions. However, there are other, equally valid configurations. The system could easily be set to timeout after some threshold number of questions, or switch to a new thread if the local follow-ups were exhausted. A topic switch could be implemented by having the system transition to the default frame when it runs out of follow-ups, and have the default frame contain valid thread-starting

topics (e.g. "about them Red Socks..."). Depending on the role of this dialogue system (i.e. whether it stands alone or is a segue into some other type of interaction), different termination strategies are more or less appropriate.

In this last section, I have described how a dialogue is initiated, taken the reader through one complete cycle of the dialogue, and described how a dialogue reaches its conclusion. In the following chapter, I will describe in detail the components powering this dialogue system.

Chapter 4

Implementation: the Framework and the Prototype

In this chapter, I will discuss the high-level architecture of this system, including how the system was ported online and outfitted with a recognizer and synthesizer. In addition, we will also explore the role of each of the major components designed to power this framework, including a feature frame to control topic transitions and processing, a template tree for generation, and a persona frame for creating random user profiles.

4.1 System Architecture

At the highest level, the framework consists of various components orchestrated by a control file, as shown in Figure 17. The specific format of this control file is that of a dialogue control ("dctl") file, and the file used for my prototype is included in Appendix A

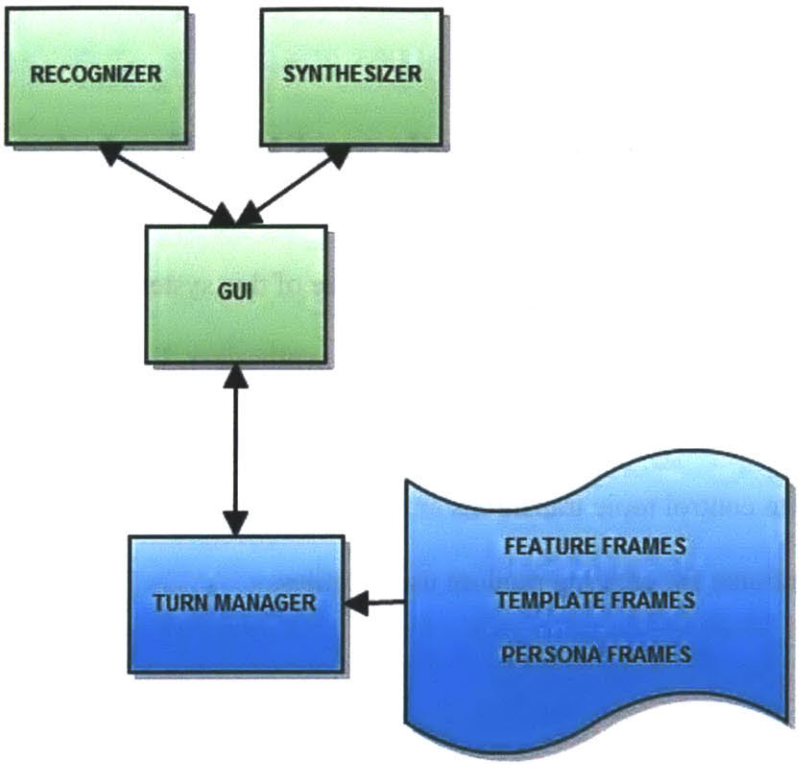

Figure 17: An overview of the system architecture with the front-end GUI, synthesizer and recognizer and the backend framework

Most of my work was done in the text-based backend, but in order to interface with human users, it was ported to an online system with a GUI, a synthesizer and a recognizer.

4.2 Simulated User

In addition to the core components of the framework, I implemented a simulated user which could step in at any time to play the role of the human user, as shown in Figure 17. The task of implementing a simulated user was greatly simplified by the template tree data structures that I designed for generating natural language utterances and promoted the higher level goal of language learning, as well as the lower level goal of facilitating debugging.

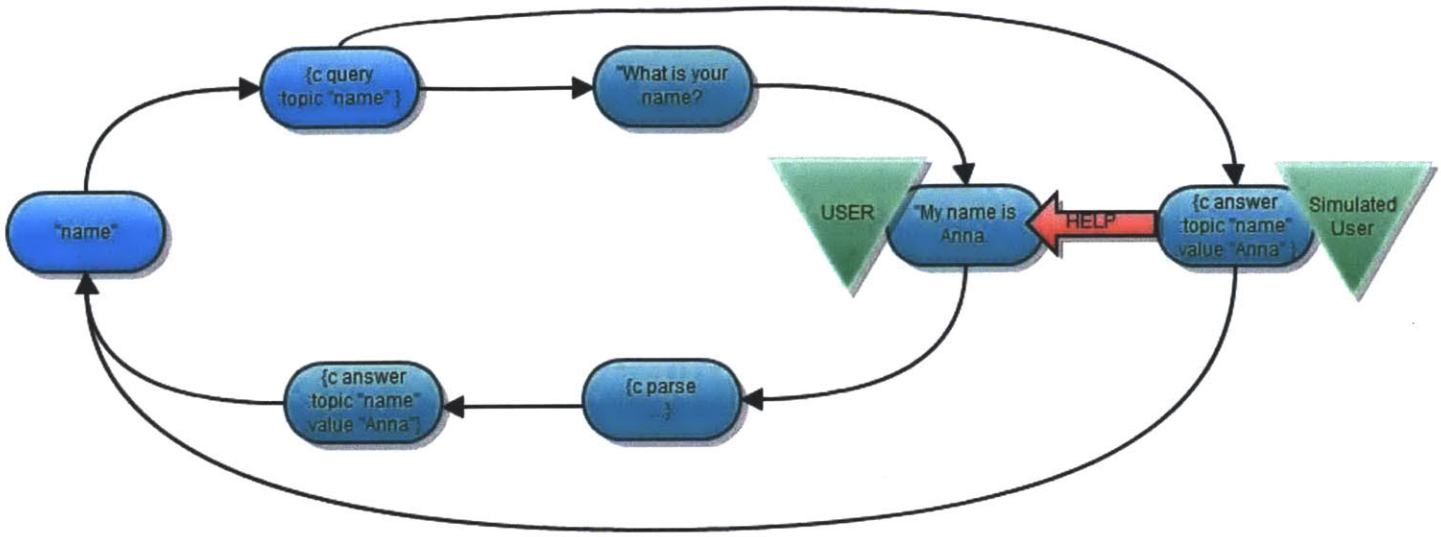

Figure 18: A block diagram depicting the flow of the dialogue involving a human user (inner loop) as well as the alternative simulated user driven path (outer loop)

In terms of language learning, we must remember that the relevant human users are non-native speakers of the target language Mandarin, and may therefore need some help or prompting, in order to successfully participate in the dialogue. Fortunately, the system is able to provide help to the user at any point in the dialogue, by simply eliciting the simulated user's response to the last query. In addition to turn-specific help, the system can also generate sample dialogues between the system and the simulated user, from which the human user might learn.

Furthermore, such generated dialogues are of great value in the debugging process. Equipped with a simulated user, the developer can test various stages of the dialogue cycle (e.g. generating, parsing, picking a new topic, transducing the user's input, etc.) without the tedium of entering each value by hand.

4.3 Major Components of the Framework

4.3.1 Persona User Model

Since the dialogue revolves around asking the user questions, we must either let the user respond based on his or her own profile or we must provide a new persona to the user. I chose to

constrain the responses (while enforcing variability) by having the user role play a randomly generated user. (e.g. "You are a 56-year-old business man, visiting China for a five day business trip. You do not speak the language, but you can use chopsticks. You have been married for twenty years and you have two children...") The underlying representation will be a frame of key-value pairs (e.g. :relationship_status "married"), from which an English/Mandarin paragraph is generated.

As the dialogue is initialized, a "persona" frame is created for the user, specifying various pieces of personal information, such as gender, age, nationality, and marital status. I have written a special compiler in C, which takes in a generalized persona frame (one with the same keys as an instance persona, but whose values are all possible instantiations of that key, as shown in Figure 19) and produces an instance persona (the identity of a particular customer, as shown in Figure 20). This frame language supports lists, arithmetic expressions, conditionals, and random numbers. The figures show only partial personas, so please see Appendix B for the full template and instance personas used in my implementation.

```
{c persona
:total_stay (
        "#if (:reason == business_trip) :brief_duration"
        "#if (:reason == internship) :medium_duration"
        "#if (:reason == job) :long_duration" )
:accompanied (
        "#if (:reason == business_trip) with_coworkers"
        "#if (:relationship_status == in_relationship) ( with_so with_family with_friends alone )"
        "#if (:relationship_status == married) ( with_so with_family with_friends alone )"
        "#if (:relationship_status == single) ( with_family with_friends alone )")
:nationality ("America" "Russia" "France" "Germany" "Sweden" "Korea" "Japan" "Canada"
        "Brazil" "Spain" "Italy" "Australia" "England" "Thailand")
:reason ( "business_trip" "internship" "job" )
:stay_so_far ":fraction * :total_stay"
:name (
        "#if (:gender == female) ( Anna Stephanie )"
        "#if (:gender == male) ( James William Sam )" )
```

```
:gender ( "male" "female" )}
```

Figure 19: A generic persona template

```
{c persona
:total_stay {c total_stay
        :days 3 }
:accompanied "with_coworkers"
:nationality "England"
:reason "business_trip"
:stay_so_far {c stay_so_far
                :days 4 }
:name "Sam"
:gender "male"}
```

Figure 20: A specified persona template of an English man named Sam

4.3.1.1 Lists

At the most basic level, the system supports random selection from among a list of strings. A good example of this is"nationality". The generic template stores a list of nationalities, and, during evaluation, one of these strings is selected at random (e.g. "American") and assigned as the value of "nationality" in the specified persona. List selection is a primitive operation in this domain-specific language, and may be embedded in more sophisticated constructions, such as conditionals.

4.3.1.2 Arithmetic

The system also supports basic arithmetic, including addition, subtraction, division, multiplication. This functionality is useful for generating numeric values, such as age, how long the user has studied Chinese, how many children he/she has, the duration of his/her stay in China, etc.

4.3.1.3 References

As can be seen from the persona template, one topic's value may depend on another's. For example, ":stay_so_far" is equal to ":fraction * :total_stay", i.e. some random fraction of the user's total stay in China. ":fraction" is one of the keywords discussed in section 5.1.5. Conditionals, which we will discuss in the next section, are another case in which a topic's value will vary, depending on that of other topics.

4.3.1.4 Conditionals

Conditionals are more powerful than references alone, and allow the developer to design more consistent user profiles. For example, although names are arbitrary, the persona's gender must match the gender of his/her name.

```
:name (
        "#if (:gender == female) ( Anna Stephanie )"
        "#if (:gender == male) ( James William Sam )" )
```

Here, if ":gender" is female, the system will set her name to either "Anna" or "Stephanie", otherwise, it will choose randomly between the male names "James" "William" and "Sam".

4.3.1.5 Keywords: durations, fractions

Finally, the persona language supports developer-defined keywords, such as ":fraction" or various durations. These are useful for generating random numbers or lengths of time.

4.3.1.6 Multiple Personas

The system allows the developer to store multiple persona templates, and choose randomly from them at runtime. The current implementation has three high-level profiles:

student, visitor, and professional. This allows for greater diversity without sacrificing the internal consistency of each persona instance.

4.3.1.7 Generality

The framework is general enough to be applied to a variety of scenarios, such as automatically generating a user's shopping list or other information relevant to the extensions discussed in Section 5: Future Work and Discussion.

4.3.1.8 Prototype

In the prototype, there are three generic personas: student, visitor, and professional. (To see the actual generic persona frames used in this prototype, please refer to Appendix B.) Having distinct high-level templates allows for greater variety without sacrificing the internal consistency of each user profile. For instance, students are still in school, and are in China to study abroad or do a summer internship. They have a major, but no salary and are generally younger and less likely to be married or have children. Professionals are those who are in China on business, and visitors are on vacation.

4.3.2 Feature Frames

The feature frames (pictured in Figure 21) are probably the most versatile component of the system. The goal of the feature frames was to gather all of the information specific to a feature in one convenient location. While this is makes information easier to look up, it does clump together information used for different stages of dialogue, such as generation and choosing a new topic.

```
{c feature
 :feature "relationship_status"
 :assumable 1
 :generalizable 0
 :extraction ( ( ":*ANYWHERE*" ":action" ":action" )
         ( ":*ANYWHERE*" ":agent" ":agent" )
         ( ":patient" ":noun" ":noun")
         ( ":complement" ":adj" ":adj" ) )
 :verification {c having_fun
                        :requirements ( ":action" ":adj" )
                :person ( "我" )
                :noun ( ":male_SO" ":female_SO" )
                :action ( "结婚" "有" )
                :adj ( "单身" ) }
 :translation {c translation
                         :结婚 "married"
         :单身 "single"
         :有 "in_relationship" }

 :default {c default
                :remarks ("wow_really" "too_bad")
 }
 :responses {c cond
                 :single {c single
            :followups ("age" "looking")
                 }
         :married {c married
                 :followups ( "so_here" "age" "so_how_long" "children" "miss_so")
                }
         :in_relationship {c relationship
                 :followups ("so_hot" "so_here" "so_how_long" "miss_so")}}}
```

Figure 21: feature frame for the more complex topic, relationship status, where the user's response to this question should strongly affect the follow up question

The most crucial role of the feature frames is to specify coherent transitions between topics. Specifically, given the last question asked and the answer given, the feature frame contains a list of appropriate followup questions. In this sense, the set of feature frames makes up a graph with the features as nodes and pointers to followup questions as transitions (as shown in Figure 22).

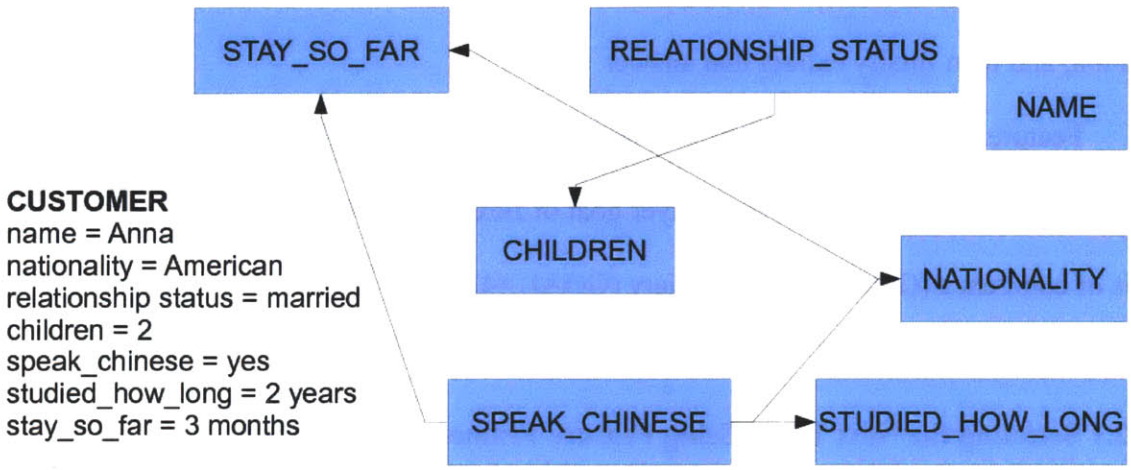

Figure 22: Visualization of feature frame topic graph with a simple persona

The feature frames also contain information about the types of questions that can be asked about each topic. For instance, ":so_here" (i.e. "Is your husband/wife/girlfriend/boyfriend here with you in China?") is a "yes no" question and therefore must be processed slightly differently from ":favorite_dish" (i.e. "What is your favorite Chinese dish?"). And to make the whole situation more complicated, many questions are "assumable", meaning that one may assume a given value, transforming the query into a "yes no" question, as in "Is your favorite Chinese dish Peking Duck?". Information about whether the topic is a "yes/no" question by default and whether or not the topic can be assumed to have a particular value is specified with key-value pairs in the feature frame (e.g. ":yesno 0" means that the topic does not elicit a "yes/no" question by default and ":assumable 0" means that one cannot assume a given value).

In addition to describing transitions and various properties of the topic's questions, the feature frames also contain information about how to process the user's response to queries on this topic. For instance, if the system asks about ":favorite_dish" and the user produces an utterance, the system will parse the utterance and then use its ":transduction" frame to extract the most relevant parts of the tree (as described in Section 3.3.4). Given the transduced frame, the

system will use its ":verification" frame to decide whether the user actually answered the question, and then finally extract that answer.

Feature frames play a variety of roles in the system, but ultimately were developed for the purpose of promoting the framework-layer goal of flexibility (GOAL #6) and the prototype-layer goals of coherence (GOAL #3) and variety (GOAL #4).

4.3.2.1 Selecting the Next Topic Using Feature Frames: An Overview

For some questions (but not others), the user's response should strongly influence which followup questions are appropriate. For example, after asking a person's name, there are some sets of questions which are frequently asked next, but in most cases, that set of followups is unaffected by whether the person's name turned out to be "Anna" or "Stephanie". However, if I ask a person's marital status, his or her response is likely to affect the questions I ask next (e.g. if the person is single, it would be strange to ask how he met his wife or how many children he had). Given this property of questions, I needed a mechanism that would allow me to specify sets of new topics based on either the user's response or the question alone. To this end, I designed the ":followups" and ":defaults" fields of the feature frame (as shown in Figure 21), where ":followups" are specific to the user's answer and ":defaults" are a list of questions that are often asked afterward, regardless of the user's answer.

4.3.2.2 AskMaps and CondMaps: Dialogue State as a Set of Bit Vectors

One of the most basic requirements for dialogue coherence is that the system does not repeat questions. The system could determine whether a given question had already been asked by examining its model of the user. However, for ease of processing, I decided to store this

information in the form of a bit vector named AskMap. As long as I use a global ordering for the supported topics (in this case, lexographic order), I can easily combine various dialogue states, using simple bitwise operators, such as AND and OR.

Similarly, I wished to store information about questions that have been "unlocked." For example, if I know a person to be married, I might randomly ask whether he had any children. In order to incorporate this subtlety, I store a list of initially locked questions in my dialogue.frame. During initialization, the system converts this list of "locked" features into a bit vector, setting each of the listed topics to 0 and the rest to 1. When a locked topic is seen in the ":followups" set of questions, its bit is set to 1.

Finally, I can condense ":followups" and ":defaults" into bit vectors of the same form. For ":defaults", I merely employ the same process as for converting the list of locked questions into a CondMap, setting listed features to 1 and the rest to 0. For followups, the system first uses the answer to extract the relevant set of topics, and then converts it into a bit vector of the same form.

4.3.2.3 Topic Selection

A coherent dialogue should favor ":responses" over ":defaults", since these are based on the user's response and the previous question, rather than the previous question alone. We must therefore encode this preference in the topic selection. Potential next topics can be divided into many categories (as shown in Figure 23).

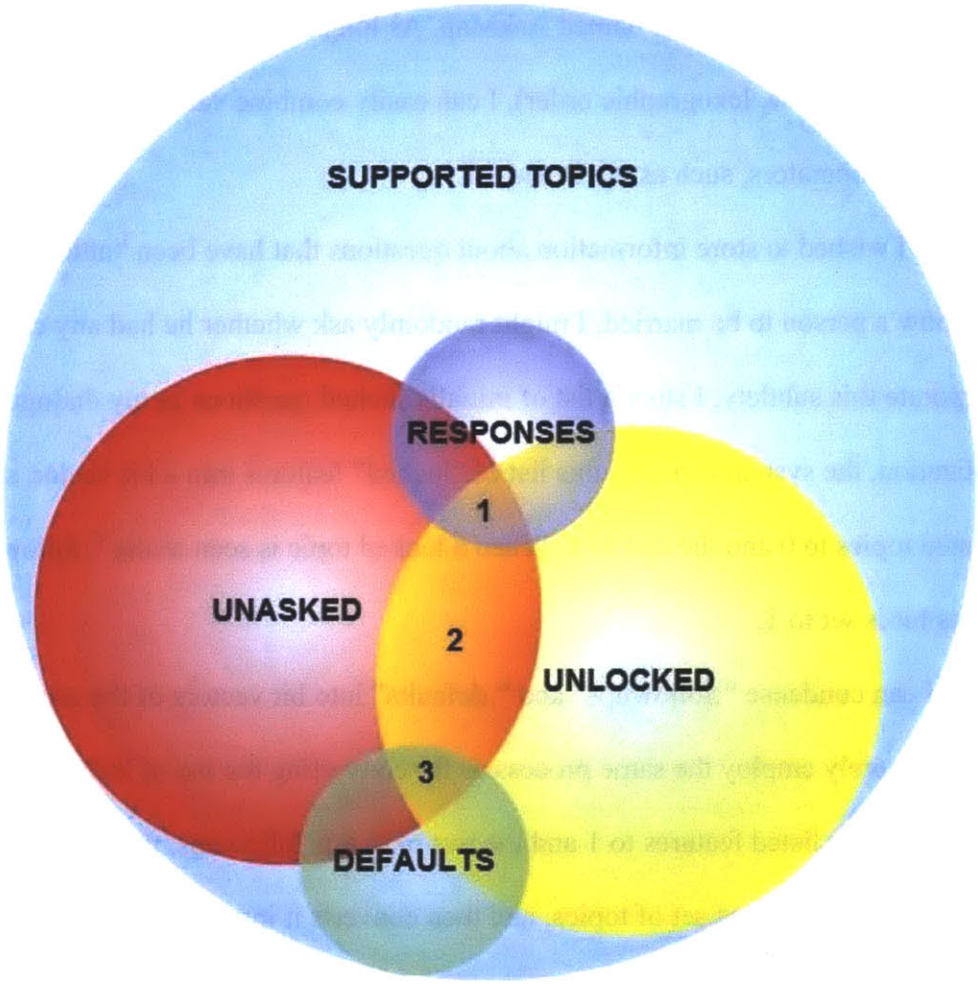

Figure 23: Venn diagram showing the status of all supported topics. Region 1 (unasked, unlocked questions based on user's response) represents the strongest candidates for next topics, Region 3 the next strongest, and Region 2 the weakest.

Most of the topics are not licensed (e.g. because they have already been asked or have not yet been unlocked), but, of the licensed topics, an ordering should be applied. The best topics are those which are unasked, unlocked, and in the set of *response-specific* followups. The next best are those which were unasked, unlocked, and in the set of *default* followups, given the system's last question. Finally, there are a set of questions which do not appear in the ":defaults" or the ":followups", but which have been unlocked but not yet asked.

Now that we have established this hierarchy and we have all of the dialogue state stored in four bit vectors, the process of choosing a next topic can be distilled down to the expression:

```
(Wr * RESPONSE && ASKMAP)
        || (Wd * DEFAULT && ASKMAP && CONDMAP)
                || (Wa * ASKMAP && CONDMAP)
```

Depending on the weights set, the system can prefer or disallow any of the licensed sets.

4.3.2.4 Supporting Assumptions

One final complication is that of assumptions. Often a followup question based on the user's response will contain an implicit assumption. For example, if someone says he is American and I have some preconception that Americans like Kung Pao Chicken, I might want to ask "So, your favorite Chinese food is Kung Pao Chicken?" With the functionality I've described so far, the closest followup would be to ask "What is your favorite Chinese dish?", which loses much of the intended asssumption's meaning and the overall coherence it could lend to the dialogue.

The coherence and variety gained by supporting assumed questions is great enough that it was worth the simplicity lost. I therefore implemented assumptions as topics with the assumption appended (e.g. ":favorite_dish->kung_pao").

For the prototype implemented in this system, the actual feature frames can been seen in Appendix C.

4.3.3 Template Trees

Template trees store templates for generating questions and answers on every supported topic. By storing this information in a tree-like structure, as shown in Figure 24, it becomes easier to access during the dialogue, as well as providing a useful debugging tool and being capable of generating data to train the recognizer.

```
:RELATIONSHIP_STATUS {c eform
        :query {c eform
                        :male {c eform
                                        :single ( "你 是 单身 吗" )
                                        :!single ( "你 不 是 单身 吗" )
                                        :in_relationship ( "你 有 女 朋友 吗"
                                          "你 有 没 有 女 朋友" )
                        :!in_relationship ( "你 没 有 女 朋友 吗" )
                                        :married ( "你 结婚 了 吗"
                                          "你 结婚 了 没 有" )
                                        :!married ( "你 还 没 结婚 吗" ) }
                        :female {c eform
                                        :single ( "你 是 单身 吗" )
                                        :!single ( "你 不 是 单身 吗" )
                                        :in_relationship ( "你 有 男 朋友 吗"
                                                        "你 有 没 有 男 朋友" )
                        :!in_relationship ( "你 没 有 男 朋友 吗" )
                                        :married ( "你 结婚 了 吗"
                                                "你 结婚 了 没 有" )
                                        :!married ( "你 还 没 结婚 呢" ) }}
        :assumption {c eform
                                :female {c eform
                                                :single ( "你 是 单身 呢?" )
                :!single ( "你 不是 单身 呢?" )
                                        :in_relationship ( "你 :already :have :girlfriend" )
                                                :married ( "你 :am_married"
                                "你 :already :am_married" )}
                                :male {c eform
                                        :single ( "你 :now :am_single"
                                                        "你 还 :am_single" )
                                        :in_relationship ( "你 :already :have :boyfriend" )
                                        :married ( "你 :am_married"
                                "你 :already :am_married" )}}
        :answer {c eform
                                :female {c eform
                                                :single ( "我 现在 是 单身"
                                                "我 还 :am_single" )
                                        :in_relationship ( "我 :already :have :boyfriend 了" )
                                                :married ( "我 :am_married"
                                "我 :already :am_married" )}
                                :male {c eform
                                        :single ( "我 :now :am_single"
```

```
                                        "我 还 :am_single" )
                                :in_relationship ( "我 :already :have :girlfriend" )
                                 :married ( "我 :am_married"
                            "我 :already :am_married" )}}
:affirmation {c eform
                            :female {c eform
                                        :single ( "我 现在 是 单身"
                                        "我 还 :am_single" )
                                :in_relationship ( "我 :already :have :boyfriend 了" )
                                        :married ( "我 :am_married"
                            "我 :already :am_married" )}
                            :male {c eform
                                 :single ( "我 :now :am_single"
                                                "我 还 :am_single" )
                                :in_relationship ( "我 :already :have :girlfriend 了" )
                                 :married ( "我 :am_married"
                            "我 :already :am_married" )}}
:negation {c eform
:male {c eform
                                 :single ( "其实, 我 不 :am_single")
                                :in_relationship ( "其实, 我 :have_not :girlfriend" )
                                 :married ( "其实, 我 还 :am_not_married" )}
:female {c eform
        :single ( "其实, 我 不 :am_single")
                                :in_relationship ( "其实, 我 :have_not :boyfriend" )
                                 :married ( "其实, 我 还 :am_not_married" )}}}
```

Figure 24: template tree for the topic "relationship_status"

When generating a sentence from the template tree, the system looks up the given path, which includes information on topic, utterance type, and gender and relationship status of the addressee, and selects a template at random from the selected subtree. If the utterance is an answer or an assumption, the system can then insert the included value into the template's blank space, yielding a full utterance.

The power of the templates is in their versatility, doubling as generator for both the system and the simulated user, as well as a source of data for debugging or training the recognizer. Given the ease of debugging and the explicitness with which they are specified, a

developer can be reasonably confident that learners will not be exposed to ill-formed utterances, upholding the goal of quality. Furthermore, it is easy to provide a variety of syntactic structures and synonyms that can be interchanged at random, providing greater variety to the dialogue. Please look to Appendix D to see the full template tree for my prototype.

4.4 Evaluating the Prototype

Dialogue systems are difficult to evaluate in general, and the goals of this particular dialogue system differ enough that it is not effective to apply standard metrics, such as completion time, given that efficiency is not relevant here. Ultimately, the system could be evaluated in terms of the four design goals put forward in Section 3.2. In the worst case, I can have humans judge the quality of each system utterance and the coherence of each transition. While tedious, this approach is feasible with the help of services like Amazon Mechanical Turk. In terms of evaluating coverage, I can use a simulated user to generate batches of dialogue and evaluate the percentage of the space covered and its general distribution. The danger here, of course, is that a poor user simulation might constrain the dialogue paths. Finally, in terms of reusability, I can examine the cost of adding a new feature and the extent to which others can make use of the components of my system for other domains.

To this end, I gathered a sample of 20 dialogues (included in Appendix E) between the system and the simulated user. I will evaluate this prototype's dialogues according to the three prototype-layer goals: quality of natural language, coherence, and variety.

Quality of natural language is difficult for me to evaluate for the same reason that it was difficult for me to achieve: I am not a native speaker of Mandarin Chinese. I therefore enlisted the help of my coworkers and friends, Seastar Lin and Jacqueline Lee. Both are Taiwanese and

native speakers of Mandarin. Most dialogues contained at least one questionable statement, especially the remarks. Fortunately, given the simplicity of the framework, these lapses are quite easy to edit, and after receiving feedback, I quickly iterated on the system's remarks. Overall, the quality of the natural language was high, considering that the developer is a non-native speaker, and trivial to correct where lacking.

Because I chose to let the dialogues continue until the local topics had been exhausted, the samples were quite long with an average length of 17.1 utterances. Since this prototype only covers 18 topics (nationality, reason, relationship_status, children, so_how_long, so_here, so_hot, miss_so, so_how_long, stay_so_far, stay_to_go, total_stay, having_fun, can_use_chopsticks, speak_chinese, studied_how_long, name, age), an average dialogue exhausts nearly half the topics. All of the 18 topics were covered within the 20 sample dialogues, no two dialogues took the same path, questions and remarks were worded differently, and no two users had the same profile, resulting in high variety.

In terms of coherence, the illusion of intelligence was quite strong with relatively few bad topic transitions (only one or two per seventeen line dialogue). The remarks helped a great deal with the coherence, since they allowed the system to acknowledge the user's response before asking the next question. Thus, even if the system changed the topic, the next question didn't feel as abrupt as it would have without the system's comment first.

Chapter 5

Discussion and Future Work

Given the difficulty of building a coherent spoken language dialogue system and the necessarily limited scope of an MEng thesis, much work remains to be done. Below I discuss future work along several dimensions, including adding new features to the current implementation, extending into other domains, and evaluation.

5.1 Modifications to the prototype

At the shallowest layer, the prototype could be modified in many ways, potentially resulting in an application that better supported the application-layer goal of language learning. For example, an English paragraph is generated from the persona frame, but perhaps a Mandarin Chinese paragraph would also be useful to language learners, giving them a chance to practice reading in Chinese. Or perhaps it would be more useful to have both paragraphs shown side-by-side. Chinese literacy is challenging to achieve, and this feature is therefore worth considering.

Another application-layer modification would be to provide greater help functionality. The current prototype allows the user to ask the simulated user for its response at any point in the dialogue. However, perhaps it would be useful to be able to look up specific vocabulary words, or practice words that were particularly hard to pronounce. Also, if the user has trouble understanding the question posed, the simulated user's answer may not be enough to clarify the situation. Ideally, an instant translation feature could be added.

Certainly, the dialogue system would benefit from more topics and transitions, as well as a template tree with more ways of saying the same statements or questions. For example, the

system could ask about the user's favorite Chinese dish, whether he liked spicy or sweet food, whether he or she planned to return to China, etc. These structures could be endlessly enriched, supporting the goal of variety in the dialogues.

5.2 Natural Extensions: Bargaining

The chatting prototype described in this paper was designed to be a natural segue into other types of social interaction, such as bargaining, which could be implemented using my framework. Such multi-stage interactions could be implemented as a additional modules. I had originally hoped to include bargaining as part of the scenario, but it became clear that this was too ambitious a goal.

Although it is a crucial skill, those who travel to China are rarely prepared for the constant haggling. Almost nothing has a fixed price and it is expected that you will negotiate a fair price for yourself on items such as fruits and vegetables, fish, jewelry, clothing, books, DVDs, hotels, and taxi fares. If you don't bargain, you will be ripped off spectacularly, paying two to ten times what you ought to. Even if you don't mind being cheated constantly, there are a variety of reasons why it is a good idea to negotiate with a salesperson. First of all, bargaining is a chance for rich interaction and a great way to practice speaking Mandarin. Secondly, since haggling is an integral part of Chinese culture, you will have a much more enjoyable and authentic experience if you take part. Thirdly, being a good bargainer will earn you a great deal of respect from the Chinese people you encounter. Not haggling down the price is almost like buying a box of tictacs, throwing a $100 bill on the counter, and saying, "keep the change...". It's ridiculous to the point of being embarrassing or disrespectful. Besides, negotiation is an extremely valuable skill that is useful anywhere and everywhere you go.

Researchers at KTH in Sweden have set a precedent with the DEAL system [10]. The authors argue that a trading domain is very well-suited to a CALL system, because it is a restricted, universal domain, it is useful, it involves negotiation (complex interaction with rational/irrational components), it is easy to introduce new vocabulary, and the dialogue is well-known but still has surprise, social commitment and competition.

To give a clearer picture of the type of interactions this system would be modeling, an example dialogue taken from Beijing Travel Department's "Travel Tips" is shown below:

(Looking at a nice pair of fake Calvin Klein pants)
You: How much?
Seller: 500
(You look and decide the pants are worth about 100 to you.)
You: Cheaper!
Seller: How much [do] you want to pay?
(never answer this straight away, just look the pants over many times, ask why such a high price, remind them its[sic] a fake, find holes, etc.)
You: Cheaper!
Seller: 300
You: 40
Seller: No, no... these are real Calvin Klein pants, good quality... 280.
You: 50
Seller: 200
(Laughing)
You: 60
Seller: Ok, ok, last price, last price 160.
You: Ok, ok, last price 70.
Seller: These good pants must be at least 100.
You: 80
Seller: No, no, sorry, no profit.
(Start walking away)
You: Ok, sorry...
(calls you back, puts item in bag)
Seller: Ok 90!
You: Ok 85!
Seller: Ok sold... wow, you are [a] good negotiator.

If the seller is still smiling at the end, it is likely that you could have gotten the pants for less, maybe 50 yuan. However, the initial price (500) was still much higher than the actual price that was agreed upon (85).

Such an interaction could be implemented with extensions of the framework. A multi-stage dialogue could be modeled as a tree where at each turn the system decides whether to continue with the current conversation thread, change the subject, or transition to bargaining mode.

5.3 Extensions to the Framework

And then there are modifications to the framework itself. For instance, the dialogue model is quite simplistic, which supports the goal of simplicity, but limits the variety and coherence that I wish to provide the user. A more natural dialogue would allow each party to ask questions of the other. While I could easily provide a persona to the system, it would be more challenging to find a place for the user to ask a question.

Another modification would be to explicitly address lying. Although the system has access to the user's actual persona, in addition to its incomplete model of the user, the system does not currently use that information to verify that the user answered the question correctly. This was mostly to avoid "cheating", since I wanted the system to recognize the user's utterance on the merits of its own recognition, parsing, and transduction. Besides, I worried about biasing the system too strongly, given that that might cause false positives (e.g. the user answers incorrectly, but the system still "hears" the correct response). Therefore, at this point, the system is quite naive and will store whatever value the user gives into its model, regardless of its

veracity. An interesting extension would therefore be to call users on their lies (e.g. "You don't look 23... how old are you, really?" or "You don't sound American... are you from Russia?") This would add entertainment value to the user experience.

Finally, we might employ some machine learning to train a prototype. For example, by processing many such dialogues, the system could update its feature frames with the most natural topic transitions, train its recognizer, or discover more wordings to add to its template tree.

5.4 Evaluating Language Learning

Language proficiency and its differential are difficult to measure. Most agree that language proficiency consists of reading, writing, listening, and speaking abilities. Since language ability is important in these global times, there are standardized exams which measure each of these abilities, such as the Chinese HKN exam. One company had me take a technical interview in Mandarin and write some characters on a board. Others wished me to take a formal exam. At the end of the day, living a foreign country is the ultimate final exam. For the purpose of evaluating this system, I would probably take the same route as other language learning games from my lab and run user studies.

5.5 Highest Level Vision

At the highest level, I could imagine my system as part of an entire streetmarket full of language learning games, lined with stalls of haggling merchants, street food, barber shops, hospitals, jewelry, clothing, and electronics. Games could involve buying all the items on a shopping list with points awarded for having negotiated low prices.

5.6 Conclusion

My aim was to create a system that would help learners of Mandarin Chinese (or potentially other foreign languages) to achieve the threshold fluency needed to function in a Mandarin-speaking country and hold further conversation with native speakers.

This system was designed with the three tiers of goals in mind: (1) the application layer goal of language learning, (2) the prototype layer goals of correctness, coherence, and variety, and (3) the framework layer goals of simplicity, flexibility, and reusability.

My prototype's success in achieving the goals of correctness, coherence, and variety demonstrates the power of my framework. By designing an intuitive and extensible framework, I lay the groundwork for a whole series of CALL systems, giving language learners the practice and confidence they need to speak foreign languages.

Appendix A

Dialogue Control File

Below is the dialogue control specification for the simulation system. It controls both the user simulator and the dialogue partner. The last rule is a recursion back to the top of the rule list, representing the next turn in the dialogue.

The dctl file for the system that interacts with the user is somewhat more complicated, as it includes rules for processing an N-best list of utterances from the recognizer and rules for generating a synthesized utterance, as well as an option for typed input.

```
{c dctl
 :initial_frame {c turn_manager
   :domains "Persona sccYishu_streetmarket"
   :logdir "../LanguageLearning/streetmarket/logs"
   :languages "persona chinese_persona key_value"
        :note "We're assuming they're running out of the dctl directory"
        :static_prev_state 1
        :frame_database "../LanguageLearning/streetmarket/dialogue.frame"
        :always_copy_token 1
}
 :rules (
  {c rule
        :conditions "!:customer_model"
        :variables {c variables :output_key ":customer_model" :log
":customer_model" :ngenerate 1 }
        :operation "generate_persona"  }
        {c rule
        :conditions ":customer_model & !:shopkeeper_model"
        :variables {c variables :input_key ":customer_model" :output_key
":shopkeeper_model" :log ":shopkeeper_model" :ngenerate 1 }
        :operation "initialize_shopkeeper_model" }
        {c rule
        :conditions ":customer_model & shopkeeper_model & !:user_response"
        :operation "initiate_dialogue" }

  {c rule
        :conditions ":crazy_times & :customer_model & :shopkeeper_model & :user_response"
        :variables {c variables :output_key ":manual_topic" }
        :operation "read_input_line" }
```

```
        {c rule
        :conditions ":manual_topic"
        :variables {c variables :input_key ":manual_topic" :output_key ":system_query" }
        :operation "form_manual_query" }
        {c rule
        :conditions ":customer_model & :shopkeeper_model & :user_response"
        :variables {c variables :output_key ":system_query" :log ":system_query" }
        :operation "systems_turn" }
        {c rule
        :comment "recycle without models to trigger another conversation"
        :conditions ":dialogue_complete"
        :variables {c variables :rules ":rules" :in ":utterance_id :from_user_input " }
        :operation "hub_rule" }
        {c rule
        :conditions ":dialogue_complete"
        :variables {c variables :control ":return" }
        :operation "nop" }
        {c rule
        :conditions ":system_query"
        :variables {c variables
                :input_key ":system_query" :output_key ":system_utterance" :log_and_print
":system_utterance"
                :template_key ":template_file"
                :domain "Persona" :language "chinese_persona" }
        :operation "paraphrase_frame" }
 {c rule
        :conditions ":customer_model & :shopkeeper_model & :system_query
& :from_user_input"
        :variables {c variables :output_key ":user_utterance" }
        :operation "read_input_line" }

        {c rule
        :conditions  ":customer_model & :shopkeeper_model & :system_query &
( !:user_utterance | :user_utterance HELP )"
        :variables {c variables :output_key ":proto_response" :log ":proto_response"  }
        :operation "users_turn" }
        {c rule
        :conditions ":proto_response"
        :comment "this produces english from the simulated user (for human consumption only)"
        :variables {c variables
                :input_key ":proto_response" :output_key ":user_utterance" :log_and_print
":user_utterance"
                :template_key ":template_file"
                :domain "Persona"
                :language "chinese_persona" }
        :operation "paraphrase_frame" }
```

```
        {c rule
        :conditions ":user_utterance"
        :variables {c variables
                :domain "sccYishu_streetmarket"
                :input_key ":user_utterance" :output_key ":parse_frame" :log ":parse_frame" }
        :operation "create_frame"
        }
        {c rule
        :conditions ":parse_frame"
        :variables {c variables
                :domain "Persona"
                :language "key_value"
                :output_key ":key_value" }
        :operation "paraphrase_frame" }
 {c rule
        :conditions ":key_value"
        :variables {c variables
                        :simplify_kv_frame 1
                :input_key ":key_value"
                :output_key ":kv_frame" }
        :operation "generate_kv_frame"
        }
        {c rule
        :conditions ":kv_frame"
        :variables {c variables :input_key ":kv_frame" :output_key ":user_response"}
        :operation "transduce_user_input"
        }
        {c rule
        :conditions  ":customer_model & :shopkeeper_model & :system_query
& :user_response"
        :variables {c variables :counter ":utterance_id" }
        :operation "increment_counter" }
        {c rule
        :variables {c variables :rules ":rules" :in
":utterance_id :user_response :customer_model :shopkeeper_model :from_user_input " }
        :operation "hub_rule" } )
}
```

Appendix B

Persona Frames

Below are the complete specifications for three top-level persona types: a student, a business traveller and a tourist.

```
{c personal_info
 :persona (

{c student
 :nationality ("America" "Russia" "France" "Germany" "Sweden"
        "Korea" "Japan" "Canada" "Brazil" "Spain" "Italy"
        "Australia" "England" "Thailand")
 :reason ("summer_program" "conference" "study_abroad")

 :total_stay (
 "#if (:reason == summer_program) :medium_duration"
 "#if (:reason == conference) :brief_duration"
 "#if (:reason == study_abroad) :long_duration" )
 :stay_so_far ":total_stay - :fraction * :total_stay"
 :stay_to_go ":total_stay - :stay_so_far"
 :age  "18 years + :random * .5 years"
 :gender ("male" "female")
 :studied_how_long ":age * .3 * :fraction"
 :how_many_visits  ":fraction * :age / 2 years"
 :can_use_chopsticks ( "yes" "no")
 :speak_chinese ( "yes" "no" )
 :favorite_chinese_dish ("hui_guo_rou" "bao_zi" "gong_bao_ji")
 :relationship_status ("married" "in_relationship" "single")
 :children (
 "#if (:relationship_status == married) :random * .25"
 "#if (:relationship_status == in_relationship) 0"
 "#if (:relationship_status == single) 0")
 :name (
 "#if (:gender == female) ( Anna Stephanie )"
 "#if (:gender == male) ( Gabe William Sam )")
 :profession "student"
 :major ("chemistry" "biology" "physics" "computer_science" "chinese")
 :how_met_so (
 "#if (:relationship_status == in_relationship) ( school dancing bar )"
```

```
 "#if (:relationship_status == married) ( school dancing bar grocery_store )"
 "#if (:relationship_status == single) %NA%" )
 :so_here (
 "#if (:accompanied = with_so) yes"
 "#if (:relationship_status == in_relationship) ( yes no )"
 "#if (:relationship_status == married) ( yes no )"
 "#if (:relationship_status == single) %NA%" )
 :so_hot (
 "#if (:relationship_status == in_relationship) ( yes no )"
 "#if (:relationship_status == married) ( yes no )"
 "#if (:relationship_status == single) %NA%" )
 :miss_so (
 "#if (:so_here == yes) no"
 "#if (:relationship_status == in_relationship) yes"
 "#if (:relationship_status == married) yes"
 "#if (:relationship_status == single) %NA%" )
 :so_how_long (
 "#if (:relationship_status == in_relationship) :random * .2 years"
 "#if (:relationship_status == married) :age - 18 years * :random * .1"
 "#if (:relationship_status == single) %NA%" )
 :looking_for_someone (
 "#if (:relationship_status == in_relationship) no"
 "#if (:relationship_status == married) no"
 "#if (:relationship_status == single) ( yes no )" )
 :accompanied (
 "#if (:relationship_status == in_relationship) ( with_so with_family with_friends alone )"
 "#if (:relationship_status == married) ( with_so with_family with_friends alone )"
 "#if (:relationship_status == single) ( with_family with_friends alone )")
 :salary 0
 :having_fun (
 "#if (:speak_chinese == yes) yes"
 "#if (:accompanied == alone) no"
 "#else yes")
 :plan_return ":having_fun"
}

{c visitor
 :nationality ("America" "Russia" "France" "Germany" "Sweden" "Korea" "Japan" "Canada"
"Brazil" "Spain" "Italy" "Australia" "England" "Thailand")
 :reason ("honeymoon" "vacation")
 :total_stay ":short_duration"
 :stay_so_far ":total_stay - :fraction * :total_stay"
 :stay_to_go ":total_stay - :stay_so_far"
 :age "10 years + :fraction * 40 years"
 :gender ( "male" "female" )
 :how_many_visits ":fraction * :age / 2 years"
```

```
:studied_how_long ":age * .3 * :fraction"
:can_use_chopsticks ( "yes" "no" )
:speak_chinese ( "yes" "no" )
:favorite_chinese_dish ("hui_guo_rou" "bao_zi" "gong_bao_ji")
:relationship_status (
"#if (:reason == honeymoon) married"
"#else (married in_relationship single)" )
:children (
"#if (:relationship_status == married) :random * .25"
"#if (:relationship_status == in_relationship) 0"
"#if (:relationship_status == single) 0")
:name (
"#if (:gender == female) ( Anna Stephanie )"
"#if (:gender == male) ( Gabe William Sam )")
:profession (
"#if (:age <= 22) %NA%"
"#else ( computer_scientist homemaker violinist physicist )")
:major (
"#if (:age < 18) %NA%"
"#if (:age > 17 && :age < 22) ( physics biology chemistry )"
"#if (:age > 22) %NA%")
:how_met_so (
"#if (:relationship_status == in_relationship) ( school dancing bar )"
"#if (:relationship_status == married) ( school dancing bar )"
"#if (:relationship_status == single) %NA%" )
:so_here (
"#if (:accompanied = with_so) yes"
"#if (:relationship_status == in_relationship) ( yes no )"
"#if (:relationship_status == married) ( yes no )"
"#if (:relationship_status == single) %NA%" )
:so_hot (
"#if (:relationship_status == in_relationship) ( yes no )"
"#if (:relationship_status == married) ( yes no )"
"#if (:relationship_status == single) %NA%" )
:miss_so (
"#if (:so_here == yes) no"
"#if (:relationship_status == in_relationship) yes"
"#if (:relationship_status == married) yes"
"#if (:relationship_status == single) %NA%" )
:so_how_long (
"#if (:reason == honeymoon) :random * .2 years"
"#if (:relationship_status == in_relationship) :random * .2 years"
"#if (:relationship_status == married) :age - 18 years * :random * .1"
"#if (:relationship_status == single) %NA%" )
:looking_for_someone (
"#if (:relationship_status == in_relationship) no"
```

```
 "#if (:relationship_status == married) no"
 "#if (:relationship_status == single) ( yes no )" )
 :accompanied (
 "#if (:reason == honeymoon) with_so"
 "#if (:relationship_status == in_relationship) ( with_so with_family with_friends alone )"
 "#if (:relationship_status == married) ( with_so with_family with_friends alone )"
 "#if (:relationship_status == single) ( with_family with_friends alone )")
 :salary (
 "#if (:profession != %NA%) 10000 * :random"
 "#else 0")
 :having_fun (
 "#if (:speak_chinese == yes) yes"
 "#if (:accompanied == alone) no"
 "#else yes")
 :plan_return ":having_fun"
}

{c professional
 :nationality ("America" "Russia" "France" "Germany" "Sweden" "Korea" "Japan" "Canada"
"Brazil" "Spain" "Italy" "Australia" "England" "Thailand")
 :reason ( "business_trip" "internship" "job" )
 :total_stay (
        "#if (:reason == business_trip) :brief_duration"
        "#if (:reason == internship) :medium_duration"
        "#if (:reason == job) :long_duration" )
 :stay_so_far ":total_stay - :fraction * :total_stay"
 :stay_to_go ":total_stay - :stay_so_far"
 :age ( "#if ( :reason == job || :reason == business_trip ) 18 years + :fraction * 40 years
+ :stay_so_far"
          "#else 18 years + :fraction * 40 years" )
 :gender ( "male" "female" )
 :studied_how_long ":age * .3 * :fraction"
 :how_many_visits ":fraction * :age / 2 years"
 :can_use_chopsticks ("yes" "no")
 :speak_chinese ("yes" "no")
 :favorite_chinese_dish ("hui_guo_rou" "bao_zi" "gong_bao_ji")
 :relationship_status ("married" "in_relationship" "single")
 :children (
 "#if (:relationship_status == married) :random * .25"
 "#if (:relationship_status == in_relationship) 0"
 "#if (:relationship_status == single) 0")
 :name (
 "#if (:gender == female) ( Anna Stephanie )"
 "#if (:gender == male) ( James William Sam )" )
 :profession (
 "#if (:reason == internship) student"
```

```
"#else ( computer_scientist statistician businessman lawyer )" )
:major (
"#if (:reason == internship) ( chemistry biology physics )"
"#else %NA%")
:how_met_so (
"#if (:relationship_status == in_relationship) ( school dancing bar )"
"#if (:relationship_status == married) ( school dancing bar )"
"#if (:relationship_status == single) %NA%" )
:so_here (
"#if (:relationship_status == in_relationship) ( yes no )"
"#if (:relationship_status == married) ( yes no )"
"#if (:relationship_status == single) %NA%" )
:so_hot (
"#if (:relationship_status == in_relationship) ( yes no )"
"#if (:relationship_status == married) ( yes no )"
"#if (:relationship_status == single) %NA%" )
:miss_so (
"#if (:relationship_status == in_relationship) yes"
"#if (:relationship_status == married) yes"
"#if (:relationship_status == single) %NA%" )
:so_how_long (
"#if (:relationship_status == in_relationship) (:age - 18 years) * :random * .1"
"#if (:relationship_status == married) (:age - 18 years) * :random * .1"
"#if (:relationship_status == single) %NA%" )
:looking_for_someone (
"#if (:relationship_status == in_relationship) no"
"#if (:relationship_status == married) no"
"#if (:relationship_status == single) ( yes no )" )
:accompanied (
"#if (:reason == business_trip) with_coworkers"
"#if (:relationship_status == in_relationship) ( with_so with_family with_friends alone )"
"#if (:relationship_status == married) ( with_so with_family with_friends alone )"
"#if (:relationship_status == single) ( with_family with_friends alone )")
:salary (
"#if (:reason == internship) 1000 * :random"
"#else 10000 * :random")
:having_fun (
"#if (:speak_chinese == yes) yes"
"#if (:accompanied == alone) no"
"#else yes")
:plan_return (
"#if (:reason == business_trip) yes"
"#if (:reason == job) yes"
"#else :having_fun")
}
```

```
)
 :*special_keys*
{c special_keys
 :fraction  ":random * 0.1"
 :random ( "1" "2" "3" "4" "5" "6" "7" "8" "9" "10" )
 :coin_flip ( "0" "1" )
 :trinomial ( "0" ".5" "1")
 :brief_duration
 {c duration
        :min 1
        :max 7
        :units "days"
 }
 :short_duration
  {c duration
        :min 1
        :max 4
        :units "weeks"
  }
:medium_duration
 {c duration
        :min 1
        :max 12
        :units "months"
 }
 :long_duration
 {c duration
        :min 1
        :max 4
        :units "years"
  }
  :super_long_duration
  {c duration
        :min 4
        :max 20
        :units "years"
  }
 :time_units (
        {c base
                :name "days" }
        {c weeks
                :nbase "7" }
        {c months
                :nbase "30" }
        {c years                :nbase "365" } )}
```

Appendix C

Feature Frames

Below are the feature frames for all supported topics in this prototype. These frames describe how transition between topics as well as how to interpret user responses to given questions.

```
{c features

 :features (

{c units
 :feature "units"
 :extraction ( ( ":time_dur" ":天" ":day")
                          ( ":time_dur" ":月" ":month" )
         ( ":time_dur" ":年" ":year" )
         ( ":time_dur" ":岁" ":year" )
         ( ":time_dur" ":星期" ":week" ))}

{c feature
 :feature "speak_chinese"
 :assumable 0
 :generalizable 1
 :extraction ( ( ":*ANYWHERE*" ":modal" ":modal" )
                  ( ":*ANYWHERE*" ":action" ":action" )
     ( ":*ANYWHERE*" ":negate" ":negate" )
     ( ":*ANYWHERE*" ":noun" ":noun" )
     ( ":agent" ":person" ":person" )
        ( ":*ANYWHERE*" ":agent" ":agent" ) )
 :verification {c speak_chinese
                          :requirements ( ":modal" ":action" ":auxil" )
                          :modal (":can")
                          :action ( ":speak")
                          :person ("我")
                          :noun (":chinese")}
 :translation {c translation
                          :说 "yes" }
 :default {c default
                 :remarks ("amazing_chinese")
        :followups ("studied_how_long" "stay_so_far" "name")}}
```

```
{c feature
 :feature "children"
 :assumable 0
 :generalizable 1
 :extraction ( ( ":*ANYWHERE*" ":amount" ":amount" )
                 ( ":*ANYWHERE*" ":action" ":action" )
     ( ":*ANYWHERE*" ":negate" ":negate" )
        ( ":*ANYWHERE*" ":adj" ":adj" )
     ( ":*ANYWHERE*" ":noun" ":noun" )
     ( ":agent" ":person" ":person" )
        ( ":*ANYWHERE*" ":agent" ":agent" ) )
 :verification {c verification
                          :requirements ( ":amount" )
                          :units ("个")
                          :action ( "有")
                          :person ("我")
                          :noun ("孩子")
               :adj ( "小" )
               :agent ("我" )}
 :translation {c translation
                          :有 "yes" }
 :default {c default
               :remarks ("so_cute")
       :followups (":relationship_status" ":so_here" ":total_stay")}}

{c feature
 :feature "having_fun"
 :assumable 0
 :generalizable 1
 :yesno 1
 :extraction ( ( ":*ANYWHERE*" ":adj" ":adjective" )
                          ( ":at" ":dir" ":at" )
        ( ":agent" ":person" ":person" )
        ( ":*ANYWHERE*" ":degree" ":degree" )
        ( ":*ANYWHERE*" ":action" ":action" ) )
 :verification {c having_fun
                          :requirements ( ":adjective" )
               :adjective ( ">:fun" ">:boring" )
               :at ( "这儿" "这里" )
               :person ( "我" )
               :action ( "觉得" )}
 :valuation ( ":fun" )
 :default {c default
       :followups ("speak_chinese" "favorite_dish" "stay_so_far")}
```

```
 :responses {c responses
                :no {c no
                                :remarks ("too_bad" "why_not")
                            :followups   ("how_many_visits" "stay_to_go" "accompanied"
"relationship_status" "so_here" )
                            }
          :yes {c yes
                                :remarks ("glad" "of_course" )
                            :followups ("stay_so_far" "stay_to_go" "speak_chinese" "favorite_dish"
"accompanied" )
                            }}}

{c feature
 :feature "relationship_status"
 :assumable 1
 :generalizable 0
 :extraction ( ( ":*ANYWHERE*" ":action" ":action" )
          ( ":*ANYWHERE*" ":agent" ":agent" )
          ( ":patient" ":noun" ":noun")
          ( ":complement" ":adj" ":adj" ) )
 :verification {c having_fun
                            :requirements ( ":action" ":adj" )
                  :person ( "我" )
                  :noun ( ":male_SO" ":female_SO" )
                  :action ( "结婚" "有" )
                  :adj ( "单身" ) }
 :translation {c translation
                              :结婚 "married"
          :单身 "single"
          :有 "in_relationship" }

 :default {c default
                  :remarks ("wow_really" "too_bad")
 }
 :responses {c cond
                    :single {c single
               :followups ("age" "looking")
                    }
          :married_SAFE {c married
                    :followups ( "so_here" "age" "so_how_long" "children" "miss_so")
                   }
         :married {c married :followups ( "so_here" ) }
          :in_relationship {c relationship :followups ( "so_here" ) }
          :in_relationship_SAFE {c relationship
                    :followups ("so_hot" "so_here" "so_how_long" "miss_so")}}}
```

```
{c feature
 :feature "age"
 :assumable 0
 :generalizable 1
 :default {c default
                :remarks ("look_younger" "look_older")
         :followups ("reason" "relationship_status" "studied_how_long")
         }
 :responses {c cond
                 :teen {c teens
              :remarks ("young")
              :followups ("alone" "reason")
}}
 :extraction ( (":*ANYWHERE*" ":agent" ":agent" )
                          (":*ANYWHERE*" ":amount" ":amount" )
          (":*ANYWHERE*" ":time_dur" ":time_dur" )
          (":*ANYWHERE*" ":units" ":units" ))
 :verification {c age
                        :requirements ( ":amount" )
                :time_dur ( ">:*SELF*" )   }}

{c feature
 :feature "name"
 :assumable 0
 :generalizable 1
 :default {c default
                :remarks ("pretty_name")
         :followups ("nationality" "speak_chinese" "relationship_status" "reason" "stay_so_far")}
 :extraction ( (":complement" ":person" ":name" )
                          (":*ANYWHERE*" ":loc_name" ":name" )
          (":*ANYWHERE*" ":agent" ":agent" ))
 :verification {c name
                        :requirements ( ":name" )
                :name ( ">:*SELF*" )}}

{c feature
 :feature "nationality"
 :assumable 1
 :generalizable 1
 :default {c default
                :remarks ("want_to_go")
         :followups    ("total_stay" "reason" "name")
 }
 :extraction ( ( ":agent" ":person" ":person" )
                  ( ":*ANYWHERE*" ":action" ":action" )
          ( ":from" ":noun" ":nationality" )
```

```
        ( ":agent" ":noun" ":noun" )
    ( ":*ANYWHERE*" ":noun" ":noun" )
        ( ":pred" ":name" ":nationality" )
        ( ":*ANYWHERE*" ":country" ":nationality" ))
 :verification {c nationality
                        :requirements ( ":nationality" )
                        :action ( "来" "来自" "来 从" )
                        :nationality ( ">:*SELF*" )
                        :noun ( "人" ":*SELF*" "家乡")
                        :person ( "我" ) }
 :responses {c cond
                :usa {c usa
            :remarks ("wince" "love_media")
                :followups ("why_iraq")
                }
        :france {c france
            :remarks ("great_wine" "romantic")
                }}}
{c feature
 :feature "so_how_long"
 :assumable 0
 :generalizable 1
 :extraction ( (":*ANYWHERE*" ":agent" ":agent" )
                        (":*ANYWHERE*" ":at" ":at" )
        (":*ANYWHERE*" ":time_dur" ":time_dur" ) )
 :verification {c so_how_long
                        :requirements ( ":time_dur" ) }
 :valuation ( ":time_dur" )
 :default {c default
        :followups ("so_hot" "miss_so" "so_here")
 }}

{c feature
 :yesno 1
 :feature "so_hot"
 :assumable 0
 :generalizable 1
 :extraction ( ( ":agent" ":noun" ":actor" )
                        ( ":*ANYWHERE*" ":adj" ":adjective" )
    ( ":complement" ":noun" ":noun" )
        ( ":*ANYWHERE*" ":degree" ":degree" )
    ( ":*ANYWHERE*" ":person" ":person" )
    ( ":*ANYWHERE*" ":negate" ":negate" ) )
 :verification {c so_hot
                        :requirements ( ":adjective" ":noun" )
                        :adjective ( ":attractive" )
```

```
                        :noun (":attractive_noun" )
                        :person ( "他" "她" )
                        :actor ( ":SO" ) }
 :valuation ( ":attractive_noun" ":attractive" )

 :default {c default
        :followups ("so_how_long" "miss_so" "so_here")
 }}

{c feature
 :feature "so_here"
 :extraction ( ( ":agent" ":noun" ":actor" )
                          ( ":*ANYWHERE*" ":adj" ":adjective" )
      ( ":complement" ":noun" ":noun" )
         ( ":*ANYWHERE*" ":adv" ":adv" )
      ( ":*ANYWHERE*" ":person" ":person" )
      ( ":*ANYWHERE*" ":negate" ":negate" ) )
 :verification {c so_hot
                        :requirements ( ":action" ":adv" )
                        :actor ( ":SO" )
                :action ( "来 和" )
                :adv ( "一起" ) }
 :valuation ( "一起" "*EMPTY*" )
 :yesno 1
 :assumable 0
 :generalizable 1
 :add_default 0
 :default {c default
        :followups ("so_how_long" "miss_so")
 }
}

{c feature
 :feature "miss_so"
 :assumable 0
 :generalizable 1
 :extraction ( ( ":*ANYWHERE*" ":action" ":action" ) )
 :verification {c so_hot
                        :requirements ( ":action" )
                :action (":miss")}
 :valuation ( ":miss" )
 :translation {c translation
                 :想 "yes" }
 :default {c default
        :followups ("so_hot" "so_how_long" "so_here" "having_fun")
 }
```

```
 :responses {c responses
                :no {c no
                  :remarks ( "that's_weird" "why_not")
          :followups   ("stay_so_far" "so_here" "accompanied" "so_how_long" "so_hot"
"how_met_so" "having_fun" )}
         :yes {c yes
                  :remarks ( "of_course" "too_bad" "understand" )
           :followups ( "stay_to_go" "accompanied" "so_how_long" "how_met_so" "so_hot" )}}}


{c feature
 :feature "reason"
 :assumable 1
 :generalizable 1
 :extraction ( ( ":*ANYWHERE*" ":agent" ":agent" )
                          ( ":comp" ":action" ":action" )
          ( ":*ANYWHERE*" ":dir" ":dir" )
          ( ":patient" ":noun" ":noun" ))
 :verification {c eform
                          :requirements ( ":action || :noun" )
                :dir ( ":here" )
                :action ( ">:*SELF*" ) }
 :translation {c translation
                          :蜜月 "honeymoon"
                :旅游 "vacation"
                :实习 "internship"
                :留学 "study_abroad"
                :开会 "conference"
                :出差 "business_trip" }
 :default {c default
         :followups   ("total_stay" "stay_so_far" "stay_to_go" "relationship_status" "nationality")
 }
 :responses {c cond
                    :study {c study
              :remarks ("wow_really")
                    :followups ("major->chinese" "age" "studied_how_long")
                    :implication ("salary->%NA%")
                    }
         :job {c work
              :remarks ("wow_really")
                    :followups ("how_many_visits" "profession->english_teacher" "salary")
                    :implication ("major->%NA%")
                    }
         :vacation {c vacation
              :remarks ("how_nice")
                    :followups ("having_fun" "favorite_dish->gong_bao_ji" "can_use_chopsticks"
```

```
"speak_chinese" "how_many_visits")
                }}}


{c feature
 :feature "total_stay"
 :assumable 0
 :generalizable 1
 :yesno 0
 :derivable ( "stay_so_far" "stay_to_go" )
 :derivation ":stay_so_far + :stay_to_go"
 :default_regular {c default
        :followups   ("stay_so_far" "stay_to_go" "reason" "studied_how_long"
"how_many_visits" "speak_chinese" "having_fun" "can_use_chopsticks" "miss_so")
 }
 :default {c default
                :followups ("reason->vacation")
        :remarks ("好短")}
 :extraction ( ( ":at" ":dir" ":at" )
                ( ":agent" ":person" ":person" )
         ( ":agent" ":noun" ":noun" )
         ( ":*ANYWHERE*" ":amount" ":amount" )
         ( ":*ANYWHERE*" ":units" ":units" )
      ( ":*ANYWHERE*" ":time_dur" ":time_dur" ) )
 :verification {c stay_so_far
                :requirements ( ":time_dur" )
                :person ( "我" )
          :noun ( "行程" )
                :at ( "这里" "中国" ) }
 :responses_regular {c cond
                :brief {c brief
            :remarks ("so_short")
                :followups ("reason->vacation" "reason->business_trip")
                }
        :short {c short
            :remarks ("too_short")
                :followups ("reason->vacation")
                }
        :medium {c medium
            :remarks ("how_nice")
                :followups ("reason")
                }
                :long {c long
            :remarks ("wow_long")
                :followups ("reason->job" "reason->study")
                }
```

```
                :super_long {c super_long
            :remarks ("whoa")
                :followups ("reason->job" "can_speak" "can_use_chopsticks"
"studied_how_long")
                }}}

{c feature
 :feature "stay_so_far"
 :assumable 0
 :generalizable 1
 :derivable ( "stay_to_go" "total_stay" )
 :derivation ":total_stay - :stay_to_go"
 :extraction ( ( ":at" ":dir" ":at" )
                ( ":agent" ":person" ":person" )
     ( ":*ANYWHERE*" ":time_dur" ":time_dur" ) )
 :verification {c stay_so_far
                :requirements ( ":time_dur" )
                :person ("我")
                :at ( "这里" "中国" ) }
 :default {c default
              :remarks ("amazing_chinese")
       :followups   ("total_stay" "stay_to_go" "reason" "studied_how_long"
"how_many_visits" "speak_chinese" "having_fun" "can_use_chopsticks")
 }
 :responses {c cond
                :brief {c brief
            :remarks ("welcome")
                :followups ("reason->vacation" "reason->business_trip")
                }
        :short {c short
            :remarks ("welcome")
                :followups ("reason->vacation")
                }
        :medium {c medium
            :remarks ("how_nice")
                :followups ("reason")
                }
                :long {c long
            :remarks ("wow_long")
                :followups ("reason->job" "reason->study")
                }
        :super_long {c super_long
            :remarks ("whoa")
                :followups ("reason->job" "can_speak" "can_use_chopsticks"
"studied_how_long")
                }}}
```

```
{c feature
 :feature "stay_to_go"
 :assumable 0
 :generalizable 1
 :derivable ( "stay_so_far" "total_stay" )
 :derivation ":total_stay - :stay_so_far"
 :extraction ( ( ":at" ":dir" ":at" )
                ( ":agent" ":person" ":person" )
     ( ":*ANYWHERE*" ":time_dur" ":time_dur" ) )
 :verification {c stay_so_far
                :requirements ( ":time_dur" )
                :person ("我")
                :at ( "这里" "中国" ) }
 :default {c default
              :remark ("fate")
       :followups   ("plan_return" "meet_again" "total_stay" "stay_so_far" "reason"
"studied_how_long" "how_many_visits" "speak_chinese" "having_fun" "can_use_chopsticks")
 }
 :responses {c cond
                :brief {c brief
            :remarks ("too_bad")
                :followups ()
                }
       :short {c short
            :remarks ("too_bad")
                :followups ("reason->vacation")
                }
       :medium {c medium
            :remarks ("how_nice")
                :followups ("reason")
                }
                :long {c long
            :remarks ("wow_long")
                :followups ("reason->job" "reason->study")
                }
       :super_long {c super_long
            :remarks ("whoa")
                :followups ("reason->job" "can_speak" "can_use_chopsticks"
"studied_how_long")
                }}}
{c feature
 :feature "studied_how_long"
 :assumable 0
 :generalizable 1
 :extraction ( ( ":*ANYWHERE*" ":action" ":action" )
                        ( ":*ANYWHERE*" ":noun" ":noun" )
```

```
        ( ":*ANYWHERE*" ":time_dur" ":time_dur" )
        ( ":*ANYWHERE*" ":person" ":person" ) )
 :verification {c studied_how_long
                  :action ( "开始" ">学" )
         :person ( "我" )
         :noun ( ":chinese" )
         :requirements ( ":time_dur" )}
 :default {c default
                :remark ("amazing_chinese")
        :followups   ("stay_so_far" "reason" "how_many_visits" "can_use_chopsticks")
 }}


{c features
 :feature "how_many_visits"
 :assumable 0
 :generalizable 1
 :extraction ( ( ":*ANYWHERE*" ":units" ":units" )
         ( ":*ANYWHERE*" ":amount" ":amount" )
         ( ":*ANYWHERE*" ":action" ":action" )
         ( ":*ANYWHERE*" ":time" ":time" )
         ( ":*ANYWHERE*" ":nth" ":nth" ) )
 :verification {c studied_how_long
                   :action ( "来" )
          :amount ( ">:*SELF*" )
          :person ( "我" )
          :time ( "已经" )
          :units ( "次" )
          :nth ( ">:*SELF*" )
          :requirements ( "amount || nth" )}
 :default {c default
                :followups ("total_stay" "stay_so_far" "studied_how_long" "having_fun")
        }
 :responses {c responses
                :first_time {c first_time
                                :remarks ("amazing_chinese")
                        :followups   ("can_use_chopsticks")
                        }
        :once_before  {c once_before
                                :remarks ("no_wonder_speak")
                        :followups ("reason")
        }
        :many_times {c many_times
                                :remarks ("no_wonder_speak")
                        :followups ("reason")
                        }}}
```

```
{c features
 :feature "can_use_chopsticks"
 :assumable 1
 :yesno 1
 :extraction ( (":*ANYWHERE*" ":action" ":action" )
                    (":*ANYWHERE*" ":person" ":person" )
        (":*ANYWHERE*" ":modal" ":modal" )
        (":*ANYWHERE*" ":auxil" ":modal" )
        (":*ANYWHERE*" ":negate" ":negate" )
        (":*ANYWHERE*" ":noun" ":noun" ))
 :verification {c can_use_chopsticks
              :requirements ( ":modal" )
        :action ( "用" )
        :modal ( "会" )
        :person ( ">我" )
        :noun ( "筷子" )}
 :valuation ( "会" )
 :default {c default
            :followups ("favorite_dish" "stay_so_far" "having_fun" "reason" "nationality")
       }
 :responses {c responses
            :no {c no
                        :remarks ("must_be_hard")
                  :followups   ("how_many_visits")
                  }
       :sort_of  {c sort_of
                        :remarks ("naturally")
       }
       :yes {c yes
                        :remarks ("wow_really")
                  :followups ("stay_so_far")
                  }}}


{c features
 :feature "favorite_dish"
 :assumable 0
 :generalizable 1
 :extraction (( ":*ANYWHERE*" ":noun" ":noun" )
            ( ":*ANYWHERE*" ":degree" ":degree" )
    ( ":*ANYWHERE*" ":action" ":action" )
    ( ":*ANYWHERE*" ":person" ":person" )
    ( ":*ANYWHERE*" ":truth_value" ":truth_value" ) )
 :verification {c favorite_dish
                  :requirements ( ":noun" )
```

```
                        :noun (">:*SELF*")
                        :action (">:like" "吃")
                        :person ( "我" ) }
 :default {c default
                :remarks ("delicious")
         :followups   ("can_use_chopsticks" "nationality" "stay_so_far" "reason" "having_fun")
         }
 :responses {c responses
                :shui_zhu_yu {c shui_zhu_yu
                                :remarks ("whoa")
                        :followups ("like_spicy")}}}

{c feature
 :feature "plan_return"
 :assumable 0
 :generalizable 1
 :yesno 1
 :extraction ( ( ":*ANYWHERE*" ":modal" ":modal" )
                 ( ":*ANYWHERE*" ":action" ":action" )
      ( ":*ANYWHERE*" ":negate" ":negate" )
      ( ":*ANYWHERE*" ":noun" ":noun" )
      ( ":agent" ":person" ":person" )
         ( ":*ANYWHERE*" ":agent" ":agent" ) )
 :verification {c plan_return
                        :requirements ( ":action" )
                        :action ( "来")
                        :person ("我")}
 :valuation ( "来" )
 :default {c default
                :remarks ("glad")
         :followups ("relationship_status" "stay_so_go" "name")}}

{c features
 :feature "default"
 :assumable 0
 :generalizable 1
 :default {c default
                :remarks ("i_see")
         :followups ("favorite_dish" "name" "nationality" "stay_so_far" "reason")}})}
```

Appendix D

Template Trees

The template tree included below is the main generation engine of this system, describing how to produce questions and answer on all supported topics.

```
{c templates

 :FAREWELL ( "再见"
             "你 慢 走" )

 :RELATIONSHIP_STATUS {c eform
       :query {c eform
                    :male {c eform
                                 :single ( "你 是 单身 吗" )
                                 :!single ( "你 不 是 单身 吗" )
                                 :in_relationship ( "你 有 女 朋友 吗"
                                   "你 有 没 有 女 朋友" )
                    :!in_relationship ( "你 没 有 女 朋友 吗" )
                                 :married ( "你 结婚 了 吗"
                                   "你 结婚 了 没 有" )
                                 :!married ( "你 还 没 结婚 吗" ) }
                    :female {c eform
                                 :single ( "你 是 单身 吗" )
                                 :!single ( "你 不 是 单身 吗" )
                                 :in_relationship ( "你 有 男 朋友 吗"
                                                    "你 有 没 有 男 朋友" )
                    :!in_relationship ( "你 没 有 男 朋友 吗" )
                                 :married ( "你 结婚 了 吗"
                                          "你 结婚 了 没 有" )
                                 :!married ( "你 还 没 结婚 呢" ) }}
       :assumption {c eform
                          :female {c eform
                                        :single ( "你 是 单身 吗?" )
             :!single ( "你 不是 单身 呢?" )
                                 :in_relationship ( "你 :already :have :boyfriend 吗" )
                                        :married ( "你 :am_married 吗"
```

```
                    "你 :already :am_married 吗")}
                    :male {c eform
                            :single ( "你 :now :am_single 吗"
                                        "你 还 :am_single 吗" )
                           :in_relationship ( "你 :already :have :girlfriend" )
                            :married ( "你 :am_married"
                    "你 :already :am_married" )}}
:answer {c eform
                    :female {c eform
                                :single ( "我 现在 是 单身"
                                "我 还 :am_single" )
                           :in_relationship ( "我 :already :have :boyfriend 了" )
                                :married ( "我 :am_married"
                    "我 :already :am_married" )}
                    :male {c eform
                            :single ( "我 :now :am_single"
                                        "我 还 :am_single" )
                           :in_relationship ( "我 :already :have :girlfriend" )
                            :married ( "我 :am_married"
                    "我 :already :am_married" )}}
:affirmation {c eform
                    :female {c eform
                                :single ( "我 现在 是 单身"
                                "我 还 :am_single" )
                           :in_relationship ( "我 :already :have :boyfriend 了" )
                                :married ( "我 :am_married"
                    "我 :already :am_married" )}
                    :male {c eform
                            :single ( "我 :now :am_single"
                                        "我 还 :am_single" )
                           :in_relationship ( "我 :already :have :girlfriend 了" )
                            :married ( "我 :am_married"
                    "我 :already :am_married" )}}
:negation {c eform
:male {c eform
                            :single ( "其实, 我 不 :am_single")
                           :in_relationship ( "其实, 我 :have_not :girlfriend" )
                            :married ( "其实, 我 还 :am_not_married" )}
:female {c eform
        :single ( "其实, 我 不 :am_single")
                           :in_relationship ( "其实, 我 :have_not :boyfriend" )
                            :married ( "其实, 我 还 :am_not_married" )}}}
```

```
:SO_HERE {c eform
    :answer {c eform
                :female {c eform
                :married {c eform
                    :yes ( "我 :husband :with 我 一起 来"
                    "他 :with 我 一起 来" )
                                :no ( "我 :husband 没有 :with 我 一起 来"
                    "他 没有 :with 我 一起 来" )}
            :in_relationship {c eform
                     :yes ( "我 :boyfriend :with 我 一起 来" )
                                :no ( "我 :boyfriend 没有 :with 我 一起 来"
                    "他 没有 :with 我 一起 来" )}}
    :male {c eform
                :married {c eform
                                                :yes ( "我 :wife :with 我 一起 来"
                                            "她 :with 我 一起 来" )
                                :no ( "我 :wife 没有 :with 我 一起 来"
                    "她 没有 :with 我 一起 来" )}
            :in_relationship {c eform
                     :yes ( "我 :girlfriend :with 我 一起 来"
                                            "我 :girlfriend :with 我 一起 来" )
                                :no ( "我 :girlfriend 没有 :with 我 一起 来"
                    "她 没有 :with 我 一起 来" )}}}

    :query {c eform
                :female {c eform
                    :married ( "你 :husband 跟 你 一起 来 吗" )
            :in_relationship ( "你 :boyfriend 跟 你 一起 来 吗" )}
                :male {c eform
                    :married ( "你 :wife 跟 你 一起 来 吗" )
            :in_relationship ( "你 :girlfriend 跟 你 一起 来 吗" )}}}


:MISS_SO {c eform
    :query {c eform
    :female {c eform
                :married ( "你 :miss 你 :husband 吗" )
                :in_relationship ( "你 :miss 你 :boyfriend 吗" )}
    :male {c eform
                :married ( "你 :miss 你 :wife 吗" )
```

```
                    :in_relationship ( "你 :miss 你 :girlfriend 吗" )}}
        :answer {c eform
:female {c eform
                    :married {c eform
                                :yes ( "我 :miss 我 :husband"
                    "我 :miss 他" )
                                    :no  ( "我 :miss_not 我 :husband"
                                "我 :miss_not 他" ) }
                    :in_relationship {c eform
                                :yes ( "我 :miss 我 :boyfriend"
                                "我 :miss 他" )
                                    :no  ( "我 :miss_not 我 :boyfriend"
                                "我 :miss_not 他" ) }}
        :male {c eform
                    :married {c eform
                                :yes ( "我 :miss 我 :wife"
                                "我 :miss 她" )
                                    :no  ( "我 :miss_not 我 :wife"
                                "我 :miss_not 她" ) }
                    :in_relationship {c eform
                                :yes ( "我 :miss 我 :girlfriend"
                                "我 :miss 她" )
                                    :no  ( "我 :miss_not 我 :girlfriend"
                                "我 :miss_not 她" ) }}}}

 :HOW_MANY_VISITS {c eform
                    :answer ( "我 来 过 :~how_many_visits 次 了" )
                :query ( "你 来 过 几 次 了" ) }

 :SO_HOT {c eform
                :answer {c eform
                    :female {c eform
                    :married {c eform
                                                :yes ("我 :husband  :is_male_attractive"
                                                "我 :husband 是 个 :male_attractive_noun" )
                                        :no  ( "我 :husband :is_not_male_attractive"
                                            "我 :husband 不 是 个 :male_attractive_noun" )}
                :in_relationship {c eform
                                            :yes ( "我 :boyfriend :is_male_attractive"
                                            "我 :boyfriend 是 个 :male_attractive_noun" )
                                            :no  ( "我 :boyfriend :is_not_male_attractive"
                                            "我 :boyfriend 不 是 个 :male_attractive_noun" )}}
                    :male {c eform
```

```
                    :married {c eform
                                        :yes ( "我 :wife :is_female_attractive"
                                        "我 :wife 是个 :female_attractive_noun" )
                                        :no  ( "我 :wife :is_not_female_attractive"
                                    "我 :wife 不是个 :female_attractive_noun" )}
                :in_relationship {c eform
                                    :yes ( "我 :girlfriend :is_female_attractive"
                                        "我 :girlfriend 是个 :female_attractive_noun" )
                                        :no  ( "我 :girlfriend :is_not_female_attractive"
                                        "我 :girlfriend 不是
个 :female_attractive_noun")}}}
        :query {c eform
                    :female {c eform
                    :married ("你 :husband :is_male_attractive 吗" )
                :in_relationship ("你 :boyfriend :is_male_attractive 吗" )}
                    :male {c eform
                    :married ("你 :wife :is_female_attractive 吗" )
                    :in_relationship ("你 :girlfriend :is_female_attractive 吗" ) }}}


 :CHILDREN {c eform
                :query ( "你有几个 :child_noun" )
        :answer (
        "我有 :~children 个 :child_noun"
        )}

 :HAVING_FUN {c eform
        :query ( "你觉得这里 :fun 吗" )
        :answer {c eform
                    :yes ( "我觉得这里 :very :fun" )
                    :no  ( "我觉得这里 :very :boring" ) }}

 :PLAN_RETURN {c eform
        :query ( "你以后还会来吗" )
        :answer {c eform
                    :yes ( "我打算以后再来一次" )
                    :no  ( "我 :surely 不会再来了" ) }}

 :SPEAK_CHINESE {c eform
        :query ( "你 :can :speak :chinese 吗" )
        :answer {c eform
                    :yes ( "我 :can_speak :chinese" )
                    :no  ( "我 :cannot_speak :chinese" )}}
```

```
 :CAN_USE_CHOPSTICKS {c eform
:query ( "你 :can 用 :chopsticks 吗" )
        :answer {c eform
                        :yes ( "我 :can 用 :chopsticks" )
                        :no  ( "我 :cannot 用 :chopsticks" )}
        :affirmation {c eform
                        :yes ( "我 当然 :can 用 :chopsticks" )
                        :no  ( "我 :cannot 用 :chopsticks" )}
        :assumption {c eform
                        :yes ( "那, 你 :can 用 :chopsticks 吗" )
                        :no  ( "那, 你 :cannot 用 :chopsticks 吗" )}
        :negation {c eform
                        :yes ( "我 当然 :can :chopsticks" )
                        :no  ( "其实 我 :cannot :chopsticks" )}}

 :NAME {c eform
         :query ( "你 叫 什么 名字" )
  :assumption ( "你 叫 :~name 吗" )
  :answer (
  "我 的 名字 是 :~name"
  "我 叫 :~name"
  "我 是 :~name"
  )
  :negation ( "我 不 叫 :~name"
        "我 的 名子 不 是 :~name"
                "我 不 是 :~name" )}


:NATIONALITY {c eform
 :query ( "你 是 从 什么 地方 来 的" )
 :assumption ( "你 是 :~nationality 人 吗" )
        :answer (       "我 是 从 :~nationality 来 的"
                        "我 是 :~nationality 人"
                ":~nationality 人")
 :negation ( "我 不 是 :~nationality" ) }

 :REASON {c eform
                :query ( "你 来 中国 作 什么" )
        :assumption ( "你 来 中国 :~reason 吗" )
        :answer ("我 是 来 这里 :~reason 的"
                "我 来 这里 :~reason"
```

```
            "来 :~reason")
        :negation ( "我 不 是 来 中国 :~reason" )}


:TOTAL_STAY {c eform
                :query ( "你 一共 会 在 这里 待 多久" )
        :answer (
        "我 一共 会 在 这里 待 :~total_stay"
          "我 的 行程 一共 :~total_stay"
          "一共 :~total_stay"
          ":about :~total_stay"
          ":~total_stay :post_about"
          )}

:STAY_SO_FAR {c eform
                 :query ( "你 :already 在 这里 多久 了" )
         :answer (
                 "我 :already 在 这里 :~stay_so_far 了"
        ":already 来 :~stay_so_far 了"
        ":already :~stay_so_far 了"
        "我 :~stay_so_far 前 来 这里 的"
        )}

:STAY_TO_GO {c eform
                :query ( "你 还 会 在 这里 待 多久" )
        :answer (
                "我 还 会 在 这里 待 :~stay_to_go"
        "我 还 打算 再 多 留 :~stay_to_go"
        "再 过 :~stay_to_go 我 就 要 回去 了"
        "我 :~stay_to_go 后 回去"
        "还有 :~stay_to_go"
        "再 :~stay_to_go"
        )}

:MAJOR {c eform
          :query ( "你 的 专业 是 什么" )
   :answer (
   "我 的 专业 是 :~major"
   "我 是 学 :~major 的"
   "我 读 :~major"
   )}

:PROFESSION {c eform
                :query ( "你 的 职业 是 什么" )
```

```
        :answer (
        "我 是 :~profession"
        "我 是 一 个 :~profession"
        )}

:SALARY {c eform
                :query ( "你 薪水 多少" )
        :answer (
        "我 的 薪水 是 :~salary 美金"
        "我 每年 赚 :~salary 美金"
        "一年 :~salary 美金"
        )}

:AGE {c eform
        :query ( "你 今年 几 岁" )
 :answer (
 "我 今年 :~age 岁"
 "我 :~age 岁"
 )}

:STUDIED_HOW_LONG {c eform
                        :query ( "你 :chinese 学 了 多久 了" )
                :answer (
                "我 :~studied_how_long 前 开始 学 :chinese"
                "我 学 :chinese 学 了 :~studied_how_long 了"
                "我 已经 学 了 :~studied_how_long 的 :chinese"
                )}

:SO_HOW_LONG {c eform
                :query ( "你们 在 一起 多久 了" )
        :answer (
        "我们 在 一起 :~so_how_long 了"
        "我们 交往 :~so_how_long 了"
        "我们 认识 :~so_how_long 了"
        ":about :~so_how_long 了"
        )}

:HOW_MET_SO {c eform
                :query ( "你们 怎么 认识 的" )
        :answer (
        "我们 在 :~how_met_so 认识 的"
        )}

:ACCOMPANIED {c eform
```

```
            :query ( "你 和 谁 一起 来 呢" )
        :assumption ( "你 跟 :~accompanied 一起 来 了 吗" )
        :answer (
        "我 :with :~accompanied 一起 来"
        "我 :alone 来"
        "我 自己 一个人 来"
        "我 的 :~accompanied 也 来 了"
        )
        :affirmation {c eform
              :yes ("我 :with :~accompanied 一起 来")
              :no ("我 没 有 跟 :~accompanied 来" )}}

:FAVORITE_DISH {c eform
               :query ( "你 最 爱 吃 的 中国 菜 是 什么" )

         :assumption ( "你 最 喜欢 吃 :~favorite_dish 吗" )
         :answer (
         "我 最 喜欢 的 中国 菜 是 :~favorite_dish"
         "我 很 喜欢 吃 :~favorite_dish"
         ":~favorite_dish 是 我 最 喜欢 的 中国 菜"
         "我 最 爱 吃 :~favorite_dish"
         )
         :affirmation {c eform
              :yes ( "我 最 喜欢 吃 :~favorite_dish" )
               :no ( "我 不 喜欢 吃 :~favorite_dish " )}}

:~name   ( "詹姆斯" "威廉" "山姆" "安娜" "斯蒂芬尼" )
:~nationality ( "瑞典" "韩国" "日本" "德国" "澳大利亚" "美国" "加拿大" "英国" "新西兰" "
挪威" "俄罗斯" "非洲" "欧洲" "墨西哥" "法国" "印度" "泰国" "以色列" "阿根廷" "西班牙" "
巴西" "意大利")
:you ( "你" )
:I ( "我" )
:have          ( "有" )
:have_not ( "没 有" )
:SO ( ":male_SO" ":female_SO" )
:male_SO   ( ":boyfriend" ":husband" )
:husband   ( "先生" "丈夫" "爱人" )
:boyfriend ( "男朋友" )
:female_SO ( ":wife" ":girlfriend" )
:wife   ( "太太" "爱人" )
:girlfriend ( "女朋友" )
:can_speak ( ":can :speak" ":speak" )
:cannot_speak ( ":cannot :speak" ":cannot" )
```

```
:speak          ( "说" "讲" )
:can            ( "会" )
:cannot         ( "不 会" )
:~reason        ( "出差" "工作" "开会" "度假" "玩" "留学" "实习" "旅游")
:chinese   ( "中文" "汉语" "普通话" )
:~how_many_visits ( ":number" )
:~total_stay ( ":duration" )
:~stay_so_far ( ":duration" )
:~stay_to_go  ( ":duration" )
:~so_how_long ( ":duration" )
:pairs ("一两" "两三" "三四" "四五" "五六" "六七" "七八" "八九" "九十")
:number         ( "一" "两" "三" "四" "五" "六" "七" "八" "九" "十" )
:now            ( "目前" "现在" )
:duration  ( ":number :time_unit" )
:~age           ( ":number" )
:~children  ( ":number" )
:time_unit ( "天" "个 星期" "个 月" "年" )
:child_noun ( "孩子" "小 孩" "小 孩子")
:chopsticks ( "筷子" )
:is_female_attractive ( ":very :female_attractive" ":female_attractive :post_very" )
:is_male_attractive ( ":very :male_attractive" ":male_attractive :post_very" )
:is_not_female_attractive ( "不 :female_attractive" "不 是 :very :female_attractive" "不
太 :female_attractive" )
:is_not_male_attractive ( "不 :male_attractive" "不 是 :very :male_attractive" "不
太 :male_attractive" )
:very           ( "很" "非常" )
:post_very  ( "极了" )
:attractive ( ":female_attractive" ":male_attractive" )
:female_attractive ( "好看" "漂亮" "美丽" "可爱" )
:male_attractive ( "帅" )
:female_attractive_noun ( "美女" "美人" )
:male_attractive_noun ( "帅哥" )
:attractive_noun (":male_attractive_noun" ":female_attractive_noun")
:miss ( "想" "想念" )
:miss_not ( "不 想" )
:how_met ( "学校" "酒吧" "公司" "海滩" )
:already ( "已经" "" )
:am_single ( "是 单身" )
:am_married ( "结婚 了" )
:am_not_married ( "没有 结婚" )
:dish ( "包子" "小笼包" "牛肉面" "麻婆豆腐" "北京烤鸭" "宫保鸡" "回锅肉" )
:fun    ( "好玩" "新鲜" "特别" "舒服" )
```

```
:boring ( "无聊" )
:with  ( "和" "跟" )
:~major ( "中文" "化学" "生物" "物理" "数学" "历史" "机械" )
:~profession    ( "律师" "商人" "医生" "计算机 的 工程师" "数学家" )
:~accompanied    ( "家人" "朋友" "同事" )
:~salary    ( ":number 万" )
:alone    ( "一个人" )
:surely    ( "肯定" "一定" "应该" )
:otherwise    ( "不然" "否则" )
:about    ( "大概" "大约" "差不多" "" )
:post_about    ( "左右" "" )
:here    ( "这儿" "这里" "中国" )
}
```

Appendix E

20 Sample Dialogues

The twenty dialogues were the third iteration of samples generated to evaluate the prototype. The first dialogue is annotated to show the underlying topic transitions.

Dialogue 1:
:system (nationality) "你 是 从 什么 地方 来 的 "
:user (nationality->thailand) "我 是 从 泰国 来 的 "
:system (want_to_go | reason->conference) "好 想 去 你 的 国家! 你 来 中国 开会 吗 "
:user (reason->study_abroad) "我 是 来 这里 留学 的 "
:system (wow_really | relationship_status->single) "真的 吗?! 你 现在 是 单身 吗"
:user (relationship_status->married) "我 已经 结婚 了"
:system (wow_really | so_how_long) "我 的 天 啊! 你们 在 一起 多久 了 "
:user (so_how_long -> 15 years) "我们 在 一起 15 年 了 "
:system (so_hot) "你 爱人 非常 好看 吗"
:user (so_hot -> no) "我 爱人 不 是 个 美女"
:system (so_here) "你 太太 跟 你 一起 来 吗"
:user (so_here -> no) "她 没有 跟 我 一起 来"
:system (miss_so) "你 想 你 爱人 吗"
:user (miss_so -> yes) "我 想 她"
:system (too_bad | having_fun) "糟 了! 你 觉得 这里 特别 吗"
:user (having_fun -> yes) "我 觉得 这里 非常 舒服"
:system (glad | stay_so_far) "我 很 高兴! 你 已经 在 这里 多久 了 "
:user (stay_so_far -> 3 years) "已经 来 三 年 了 "
:system (amazing_chinese|can_use_chopsticks->no) "你汉语说得太好了! 那,你不会用筷子吗"
:user (can_use_chopsticks->no) "我 不 会 用 筷子"
:system (must_be_hard | speak_chinese) "怎么 可能?! 你 会 说 普通话 吗"
:user (speak_chinese -> yes) "我 会 讲 汉语"
:system (amazing_chinese | studied_how_long) "你 汉语说得太好 了! 你普通话 学了多久了"
:user (studied_how_long -> 1 year) "我 学 普通话 学 了 一 年 了 "

Dialogue 2:
:system (0) "你 在 这里 多久 了 "
:user (0) "已经 两 年 了 "
:system (0) "你 汉语 说 得 太 好 了! 你 还 会 在 这里 待 多久 "
:user (0) "再 七 个 月 "
:system (0) "好 啊 ! 那, 你 会 用 筷子 吗"

:user (0) "我 不 会 用 筷子"
:system (0) "怎么 可能?! 你 是 西班牙 人 吗 "
:user (0) "加拿大 人 "
:system (0) "好 想 去 你 的 国家! 你 叫 什么 名字 "
:user (0) "我 叫 山姆 "
:system (0) "这 真 是 个 好 名子! 你 来 中国 留学 吗 "
:user (0) "来 留学 "
:system (0) "想 不 到! 你 今年 几 岁 "
:user (0) "我 23 岁 "
:system (0) "啊, 你 看起来 更 老! 你 现在 是 单身 吗"
:user (0) "我 结婚 了"
:system (0) "想 不 到! 你 有 几 个 小 孩子 "
:user (0) "我 有 一 个 小 孩子 "
:system (0) "so_cute! 你 一共 会 在 这里 待 多久 "
:user (0) "我 的 行程 一共 三 年 "
:system (0) "好短! 你 觉得 这里 好玩 吗"
:user (0) "我 觉得 这里 非常 好玩"
:system (0) "我 很 高兴! 你 会 讲 汉语 吗"
:user (0) "我 会 讲 普通话"
:system (0) "你 中文 真的 很 棒! 你 中文 学 了 多久 了 "
:user (0) "我 三 年 前 开始 学 汉语 "

Dialogue 3:
:system (0) "你 来 中国 作 什么 "
:user (0) "我 来 这里 留学 "
:system (0) "我 的 天 啊! 你 是 俄罗斯 人 吗 "
:user (0) "意大利 人 "
:system (0) "好 想 去 你 的 国家! 你 叫 什么 名字 "
:user (0) "我 叫 安娜 "
:system (0) "这 真 是 个 好 名子! 你 已经 结婚 了 吗"
:user (0) "我 结婚 了"
:system (0) "糟糕! 你 想念 你 先生 吗"
:user (0) "我 不 想 他"
:system (0) "怎么 可 能?! 你 先生 跟 你 一起 来 吗"
:user (0) "他 跟 我 一起 来"
:system (0) "你们 在 一起 多久 了 "
:user (0) "大约 18 年 了 "
:system (0) "你 爱人 帅 极了 吗"
:user (0) "我 先生 是 个 帅哥"

Dialogue 4:
:system (0) "你 来 中国 作 什么 "

:user (0) "我 是 来 这里 留学 的 "
:system (0) "我 的 天 啊! 你 是 澳大利亚 人 吗 "
:user (0) "法国 人 "
:system (0) "好 想 去 你 的 国家! 你 一共 会 在 这里 待 多久 "
:user (0) "两 个 月 左右 "
:system (0) "好 啊 ! 你 觉得 这里 舒服 吗"
:user (0) "我 觉得 这里 非常 舒服"
:system (0) "当然! 你 会 讲 中文 吗"
:user (0) "我 会 说 普通话"
:system (0) "你 中文 真的 很 棒! 你 中文 学 了 多久 了 "
:user (0) "我 学 中文 学 了 一 年 了 "
:system (0) "你 在 这里 多久 了 "
:user (0) "我 0 天 前 来 这里 的 "
:system (0) "你 汉语 说 得 太 好 了! 那, 你 不 会 用 筷子 吗"
:user (0) "我 会 用 筷子"

Dialogue 5:
:system (0) "你 是 从 什么 地方 来 的 "
:user (0) "澳大利亚 人 "
:system (0) "好 想 去 你 的 国家! 你 来 中国 旅游 吗 "
:user (0) "我 来 这里 旅游 "
:system (0) "好 啊 ! 那, 你 会 用 筷子 吗"
:user (0) "我 不 会 用 筷子"
:system (0) "怎么 可能?! 你 觉得 这里 好玩 吗"
:user (0) "我 觉得 这里 非常 无聊"
:system (0) "你 已经 在 这里 多久 了 "
:user (0) "已经 来 两 个 星期 了 "
:system (0) "欢迎 来 中国! 你 一共 会 在 这里 待 多久 "
:user (0) "大约 三 个 星期 "
:system (0) "好短! 你 普通话 学 了 多久 了 "
:user (0) "我 一 年 前 开始 学 中文 "

Dialogue 6:
:system (0) "你 会 讲 中文 吗"
:user (0) "我 讲 普通话"
:system (0) "你 中文 真的 很 棒! 你 在 这里 多久 了 "
:user (0) "我 两 年 前 来 这里 的 "
:system (0) "你 待 好 常 时间! 你 还 会 在 这里 待 多久 "
:user (0) "还有 两 年 "
SEG FAULT

Dialogue 7:

:system (0) "你 会 说 汉语 吗"
:user (0) "我 讲 中文"
:system (0) "你 汉语 说 得 太 好 了! 你 在 这里 多久 了 "
:user (0) "我 已经 在 这里 五 天 了 "
:system (0) "你 中文 真的 很 棒! 那, 你 会 用 筷子 吗"
:user (0) "我 会 用 筷子"
:system (0) "想 不 到! 你 是 法国 人 吗 "
:user (0) "加拿大 人 "
:system (0) "好 想 去 你 的 国家! 你 来 中国 旅游 吗 "
:user (0) "我 是 来 这里 旅游 的 "
:system (0) "好 啊 ! 你 是 单身 吗?"
:user (0) "我 结婚 了"
:system (0) "想 不 到! 你们 在 一起 多久 了 "
:user (0) "我们 交往 37 年 了 "
:system (0) "你 丈夫 很 帅 吗"
:user (0) "我 先生 不 是 个 帅哥"
:system (0) "你 先生 跟 你 一起 来 吗"
:user (0) "他 没有 跟 我 一起 来"
:system (0) "你 想念 你 丈夫 吗"
:user (0) "我 想 他"
:system (0) "糟 了! 你 觉得 这里 特别 吗"
:user (0) "我 觉得 这里 非常 特别"
:system (0) "我 很 高兴! 你 还 会 在 这里 待 多久 "
:user (0) "我 还 会 在 这里 待 两 天 "
:system (0) "糟糕! 你 汉语 学 了 多久 了 "
:user (0) "我 已经 学 了 两 年 的 普通话 "

Dialogue 8:
:system (0) "你 会 讲 普通话 吗"
:user (0) "我 不 会 说 普通话"
:system (0) "你 中文 真的 很 棒! 你 汉语 学 了 多久 了 "
:user (0) "我 已经 学 了 一 年 的 普通话 "
:system (0) "你 来 中国 工作 吗 "
:user (0) "我 是 来 这里 工作 的 "
:system (0) "你 是 从 什么 地方 来 的 "
:user (0) "我 是 俄罗斯 人 "
:system (0) "好 想 去 你 的 国家! 你 一共 会 在 这里 待 多久 "
:user (0) "我 一共 会 在 这里 待 一 年 "
:system (0) "你 待 好 常 时间! 你 觉得 这里 新鲜 吗"
:user (0) "我 觉得 这里 非常 新鲜"
:system (0) "我 很 高兴! 你 已经 在 这里 多久 了 "

:user (0) "已经 六 个 月 了 "
:system (0) "好 啊！ 你 会 用 筷子 吗"
:user (0) "我 不 会 用 筷子"

Dialogue 9:
:system (0) "你 来 中国 作 什么 "
:user (0) "来 实习 "
:system (0) "你 是 韩国 人 吗 "
:user (0) "泰国 人 "
:system (0) "好 想 去 你 的 国家! 你 一共 会 在 这里 待 多久 "
:user (0) " 一 个 月 "
:system (0) "好短! 你 汉语 学 了 多久 了 "
:user (0) "我 五 年 前 开始 学 汉语 "
:system (0) "你 在 这里 多久 了 "
:user (0) "我 在 这里 三 天 了 "
:system (0) "欢迎 来 中国! 你 还 会 在 这里 待 多久 "
:user (0) "我 四 个 星期 后 回去 "
:system (0) "糟 了! 你 会 用 筷子 吗"
:user (0) "我 不 会 用 筷子"
:system (0) "怎么 可能?! 你 会 说 中文 吗"
:user (0) "我 会 说 汉语"
:system (0) "你 中文 真的 很 棒! 你 叫 什么 名字 "
:user (0) "我 的 名字 是 山姆 "
:system (0) "这 真 是 个 好 名子! 你 已经 有 女朋友"
:user (0) "我 有 女朋友"
:system (0) "真 倒 霉! 你 女朋友 跟 你 一起 来 吗"
:user (0) "我 女朋友 跟 我 一起 来"
:system (0) "你 想念 你 女朋友 吗"
:user (0) "我 想念 她"
:system (0) "完 了! 你 觉得 这里 新鲜 吗"
:user (0) "我 觉得 这里 非常 舒服"

Dialogue 10:
:system (0) "你 已经 在 这里 多久 了 "
:user (0) "我 0 天 前 来 这里 的 "
:system (0) "欢迎 来 中国! 你 一共 会 在 这里 待 多久 "
:user (0) "一 个 星期 "
:system (0) "好短! 你 觉得 这里 新鲜 吗"
:user (0) "我 觉得 这里 非常 好玩"
:system (0) "我 很 高兴! 你 会 说 汉语 吗"
:user (0) "我 不 会 普通话"
:system (0) "你 中文 真的 很 棒! 你 中文 学 了 多久 了 "

:user (0) "我 学 汉语 学 了 三 年 了 "
:system (0) "那, 你 会 用 筷子 吗"
:user (0) "我 不 会 用 筷子"
:system (0) "怎么 可能?! 你 是 日本 人 吗 "
:user (0) "我 是 俄罗斯 人 "
:system (0) "好 想 去 你 的 国家! 你 来 中国 旅游 吗 "
:user (0) "我 来 这里 旅游 "
:system (0) "好 啊 ! 你 还 是 单身 吗"
:user (0) "我 结婚 了"
:system (0) "太 可惜 了! 你 今年 几 岁 "
:user (0) "我 今年 14 岁 "

Dialogue 11:
:system (0) "你 会 讲 普通话 吗"
:user (0) "我 不 会 普通话"
:system (0) "你 汉语 说 得 太 好 了! 你 在 这里 多久 了 "
:user (0) "已经 两 天 了 "
:system (0) "你 中文 真的 很 棒! 你 会 用 筷子 吗"
:user (0) "我 会 用 筷子"
:system (0) "想 不 到! 你 是 英国 人 吗 "
:user (0) "我 是 瑞典 人 "
:system (0) "好 想 去 你 的 国家! 你 一共 会 在 这里 待 多久 "
:user (0) "一共 六 天 "
:system (0) "好短! 你 普通话 学 了 多久 了 "
:user (0) "我 已经 学 了 14 年 的 汉语 "
:system (0) "你 来 中国 作 什么 "
:user (0) "我 是 来 这里 出差 的 "
:system (0) "你 有 女朋友"
:user (0) "我 已经 有 女朋友"
:system (0) "想 不 到! 你 女朋友 跟 你 一起 来 吗"
:user (0) "她 没有 跟 我 一起 来"
:system (0) "你 想 你 女朋友 吗"
:user (0) "我 想念 我 女朋友"
:system (0) "糟糕! 你 觉得 这里 新鲜 吗"

Dialogue 12:
:system (0) "你 会 讲 中文 吗"
:user (0) "我 不 会 讲 中文"
:system (0) "你 中文 真的 很 棒! 你 普通话 学 了 多久 了 "
:user (0) "我 学 中文 学 了 三 年 了 "
:system (0) "你 在 这里 多久 了 "
:user (0) "已经 两 年 了 "

:system (0) "你 待 好 常 时间! 你 一共 会 在 这里 待 多久 "
:user (0) "我 的 行程 一共 三 年 "
:system (0) "你 待 好 常 时间! 你 觉得 这里 新鲜 吗"
:user (0) "我 觉得 这里 很 无聊"

Dialogue 13:
:system (0) "你 会 说 汉语 吗"
:user (0) "我 会 说 普通话"
:system (0) "你 中文 真的 很 棒! 你 中文 学 了 多久 了 "
:user (0) "我 五 年 前 开始 学 普通话 "
:system (0) "那, 你 会 用 筷子 吗"
:user (0) "我 不 会 用 筷子"
:system (0) "怎么 可能?! 你 是 澳大利亚 人 吗 "
:user (0) "我 是 从 韩国 来 的 "
:system (0) "好 想 去 你 的 国家! 你 来 中国 作 什么 "
:user (0) "来 旅游 "
:system (0) "好 啊 ! 你 一共 会 在 这里 待 多久 "
:user (0) "一 个 星期 左右 "
:system (0) "好短! 你 觉得 这里 新鲜 吗"
:user (0) "我 觉得 这里 很 舒服"
:system (0) "我 很 高兴! 你 还 会 在 这里 待 多久 "
:user (0) "再 六 天 "
:system (0) "太 可惜 了! 你 已经 在 这里 多久 了 "
:user (0) "我 一 天 前 来 这里 的 "

Dialogue 14:
:system (0) "你 是 从 什么 地方 来 的 "
:user (0) "我 是 从 西班牙 来 的 "
:system (0) "好 想 去 你 的 国家! 你 来 中国 留学 吗 "
:user (0) "我 来 这里 开会 "
:system (0) "我 的 天 啊! 你 一共 会 在 这里 待 多久 "
:user (0) "我 的 行程 一共 四 天 "
:system (0) "太 短 了! 你 觉得 这里 好玩 吗"
:user (0) "我 觉得 这里 非常 新鲜"
:system (0) "我 很 高兴! 你 已经 在 这里 多久 了 "
:user (0) "我 已经 在 这里 两 天 了 "
:system (0) "欢迎 来 中国! 你 会 用 筷子 吗"
:user (0) "我 不 会 用 筷子"
:system (0) "怎么 可能?! 你 会 说 中文 吗"
:user (0) "我 讲 普通话"
:system (0) "你 汉语 说 得 太 好 了! 你 叫 什么 名字 "
:user (0) "我 叫 山姆 "

:system (0) "这 真 是 个 好 名子! 你 还 是 单身 吗"
:user (0) "我 结婚 了"
:system (0) "真 倒 霉! 你 今年 几 岁 "
:user (0) "我 今年 23 岁 "
:system (0) "真的 马? 你 看起来 年轻 多 了! 你 普通话 学 了 多久 了 "
:user (0) "我 已经 学 了 两 年 的 中文 "

Dialogue 15:
:system (0) "你 来 中国 作 什么 "
:user (0) "我 是 来 这里 开会 的 "
:system (0) "想 不 到! 你 一共 会 在 这里 待 多久 "
:user (0) "我 一共 会 在 这里 待 一 天 "
:system (0) "好短! 你 中文 学 了 多久 了 "
:user (0) "我 学 汉语 学 了 四 年 了 "
:system (0) "那, 你 会 用 筷子 吗"
:user (0) "我 会 用 筷子"
:system (0) "想 不 到! 你 是 韩国 人 吗 "
:user (0) "俄罗斯 人 "
:system (0) "好 想 去 你 的 国家! 你 叫 什么 名字 "
:user (0) "我 叫 斯蒂芬尼 "
:system (0) "这 真 是 个 好 名子! 你 有 男朋友 吗"
:user (0) "我 还 是 单身"
:system (0) "真的 吗?! 你 今年 几 岁 "
:user (0) "我 今年 19 岁 "

Dialogue 16:
:system (0) "你 是 从 什么 地方 来 的 "
:user (0) "意大利 人 "
:system (0) "好 想 去 你 的 国家! 你 来 中国 作 什么 "
:user (0) "来 工作 "
:system (0) "你 一共 会 在 这里 待 多久 "
:user (0) "一共 两 年 "
:system (0) "你 待 好 常 时间! 你 觉得 这里 新鲜 吗"
:user (0) "我 觉得 这里 非常 舒服"
:system (0) "当然! 你 还 会 在 这里 待 多久 "
:user (0) "我 还 会 在 这里 待 一 年 "
:system (0) "你 待 好 常 时间! 你 会 讲 汉语 吗"
:user (0) "我 会 讲 中文"
:system (0) "你 中文 真的 很 棒! 你 叫 什么 名字 "
:user (0) "我 叫 威廉 "
:system (0) "这 真 是 个 好 名子! 你 结婚 了"
:user (0) "我 有 女朋友"

:system (0) "想 不 到! 你们 在 一起 多久 了 "
:user (0) "差不多 12 年 了 "
:system (0) "你 女朋友 非常 美丽 吗"
:user (0) "我 女朋友 不 美丽"
:system (0) "你 女朋友 跟 你 一起 来 吗"
:user (0) "她 没有 跟 我 一起 来"
:system (0) "你 想 你 女朋友 吗"
:user (0) "我 想 我 女朋友"

Dialogue 17:
:system (0) "你 还 没 结婚 吗"
:user (0) "我 目前 是 单身"
:system (0) "真的 吗?! 你 今年 几 岁 "
:user (0) "我 今年 21 岁 "
:system (0) "真的 马? 你 看起来 年轻 多 了! 你 来 中国 留学 吗 "
:user (0) "我 是 来 这里 留学 的 "
:system (0) "想 不 到! 你 是 巴西 人 吗 "
:user (0) "韩国 人 "
:system (0) "好 想 去 你 的 国家! 你 一共 会 在 这里 待 多久 "
:user (0) "三 个 月 "
:system (0) "好短! 你 普通话 学 了 多久 了 "
:user (0) "我 学 普通话 学 了 六 年 了 "
:system (0) "你 会 用 筷子 吗"
:user (0) "我 会 用 筷子"
:system (0) "我 的 天 啊! 你 觉得 这里 舒服 吗"
:user (0) "我 觉得 这里 很 特别"
:system (0) "我 很 高兴! 你 已经 在 这里 多久 了 "
:user (0) "我 两 个 月 前 来 这里 的 "
:system (0) "好 啊 ! 你 还 会 在 这里 待 多久 "
:user (0) "我 三 个 星期 后 回去 "
:system (0) "糟 了! 你 会 说 普通话 吗"
:user (0) "我 讲 中文"
:system (0) "你 中文 真的 很 棒! 你 叫 什么 名字 "
:user (0) "我 叫 山姆 "

Dialogue 18:
:system (0) "你 来 中国 作 什么 "
:user (0) "我 来 这里 旅游 "
:system (0) "好 啊 ! 你 一共 会 在 这里 待 多久 "
:user (0) "大概 两 个 星期 "
:system (0) "好短! 你 普通话 学 了 多久 了 "
:user (0) "我 七 年 前 开始 学 普通话 "

:system (0) "你 已经 在 这里 多久 了 "
:user (0) " 两 个 星期 了 "
:system (0) "欢迎 来 中国! 那, 你 会 用 筷子 吗"
:user (0) "我 不 会 用 筷子"
:system (0) "怎么 可能?! 你 觉得 这里 新鲜 吗"
:user (0) "我 觉得 这里 很 好玩"
:system (0) "当然! 你 会 讲 汉语 吗"
:user (0) "我 不 会 讲 汉语"
:system (0) "你 汉语 说 得 太 好 了! 你 叫 什么 名字 "
:user (0) "我 是 斯蒂芬尼 "
:system (0) "这 真 是 个 好 名子! 你 是 瑞典 人 吗 "
:user (0) "美国 人 "

Dialogue 19:
:system (0) "你 是 从 什么 地方 来 的 "
:user (0) "巴西 人 "
:system (0) "好 想 去 你 的 国家! 你 来 中国 留学 吗 "
:user (0) "我 来 这里 留学 "
:system (0) "我 的 天 啊! 你 还 会 在 这里 待 多久 "
:user (0) "再 三 年 "
:system (0) "你 待 好 常 时间! 你 在 这里 多久 了 "
:user (0) "我 0 天 前 来 这里 的 "
:system (0) "你 中文 真的 很 棒! 你 觉得 这里 新鲜 吗"
:user (0) "我 觉得 这里 非常 无聊"
:system (0) "你 会 讲 中文 吗"
:user (0) "我 不 会 汉语"
:system (0) "你 中文 真的 很 棒! 你 普通话 学 了 多久 了 "
:user (0) "我 已经 学 了 八 个 月 的 中文 "
:system (0) "那, 你 会 用 筷子 吗"
:user (0) "我 会 用 筷子"

Dialogue 20:
:system (0) "你 会 说 汉语 吗"
:user (0) "我 说 普通话"
:system (0) "你 汉语 说 得 太 好 了! 你 叫 什么 名字 "
:user (0) "我 叫 威廉 "
:system (0) "这 真 是 个 好 名子! 你 是 巴西 人 吗 "
:user (0) "我 是 从 俄罗斯 来 的 "
:system (0) "好 想 去 你 的 国家! 你 来 中国 留学 吗 "
:user (0) "来 留学 "
:system (0) "我 的 天 啊! 你 普通话 学 了 多久 了 "
:user (0) "我 已经 学 了 七 个 月 的 汉语 "

:system (0) "那, 你 会 用 筷子 吗"
:user (0) "我 会 用 筷子"
:system (0) "我 的 天 啊! 你 觉得 这里 新鲜 吗"
:user (0) "我 觉得 这里 非常 好玩"
:system (0) "当然! 你 还 会 在 这里 待 多久 "
:user (0) "我 一 年 后 回去 "
:system (0) "你 待 好 常 时间! 你 一共 会 在 这里 待 多久 "
:user (0) "两 年 左右 "

Bibliography

[1] S. Seneff, "Web-based Dialogue and Translation Games for Spoken Language Learning," Proc. of the Speech and Language Technology in Education (SLaTE) Workshop, Farmington, Pennsylvania, October 2007.

[2] Victor Zue, Stephanie Seneff, James R. Glass, Joseph Polifroni, Christine Pao, Timothy J. Hazen, and Lee Hetherington, "JUPITER: A Telephone-Based Conversational Interface for Weather Information. IEEE Transactions on Speech and Audio Processing", Vol. 8, No. 1, January, 2000.

[3] S. Seneff, "Response planning and generation in the MERCURY flight reservation system," Computer Speech and Language (2002) 16, 283–312

[4] J. Liu, Y. Xu, S. Seneff, and V. Zue, "CityBrowser II: A Multimodal Restaurant Guide in Mandarin," Proc. ISCSLP, Kunming, China, December 2008.

[5] A. Gruenstein, J. Orszulak, S. Liu, S. Roberts, J. Zabel, B. Reimer, B. Mehler, S. Seneff, J. Glass, and J. Coughlin, "City Browser: developing a conversational automotive HMI," CHI, 2009.

[6] A. Gruenstein, "Toward Widely-Available and Usable Multimodal Conversational Interfaces." PhD thesis, MIT Department of Electrical Engineering and Computer Science, 2009.

[7] Y. Xu, A. Goldie, S. Seneff. Automatic Question Generation and Answer Judging: A Q&A Game for Language Learning," Proc. SIGSLaTE, Warwickshire, England, September 2009.

[8] S. Seneff, "TINA: A Natural Language System for Spoken Language Applications," Computational Linguistics, Vol. 18, No. 1, pp. 61--86, 1992.

[9] L. Baptist and S. Seneff, "Genesis-II: A Versatile System for Language Generation in Conversational System Applications," Proc. ICSLP '00, Vol. III, pp. 271--274, Beijing, China, Oct 2000.," Speech Communication, Special Issue on Applications to Language Learning, 2009.

[10] P. Wik, A. Hjalmarson, J. Brusk, "DEAL A Serious Game for CALL Practicing Conversational Skills in the Trade Domain, " Proc. SIGSLaTE, 2007.

Sources

Learn Mandarin Chinese: Bargaining Lesson (http://www.youtube.com/watch?v=rRzARrixa5Y)
Chinese Lessons with Serge Melnyk
nciku
Mandarin by Anthony Gamaut
Short-Term Spoken Chinese Elementary and Intermediate
Streetwise Mandarin Chinese: Speak and Understand Everyday Mandarin: The Practical Guid to Contemporary Slang and Colloquial Expressions in Today's China Rongrong Liao, David Y. Dai, Jack Franke
A Course in Chinese Colloquial Idioms
Hanyu Jingdu Keben
Learn Chinese Easily
http://www.beijingtraveltips.com/tips/shopping_1/bargaining_howto.htm
http://chinesepod.com/lessons/%E6%9D%80%E4%BB%B7%E9%AB%98%E6%89%8B/dialogue
Chinesepod Buying Jade: http://chinesepod.com/lessons/buying-jade/dialogue
Chao, Evelyn. Niubi The Real Chinese You Were Never Taught in School. Penguin Books Limited. 2009