Gaining Experience with Structured Data: Using the Resources of Dialog State Tracking Challenge 2

Ronnie W. Smith
Department of Computer Science
East Carolina University
Greenville, NC 27858 USA
rws@cs.ecu.edu

Abstract

This paper describes a class project for a recently introduced undergraduate NLP course that gives computer science students the opportunity to explore the data of Dialog State Tracking Challenge 2 (DSTC 2). Student background, curriculum choices, and project details are discussed. The paper concludes with some instructor advice and final reflections.

1 Introduction

One of the consequences of the data explosion of the past twenty-five years is the likelihood that a large number of computer scientists and computing professionals will have the opportunity to develop analytical tools for processing large, structured datasets during their careers. This is of course especially true in the domain of NLP. A variety of NLP application domains can serve as an opportunity to provide undergraduate students with a chance to gain experience with the processing of structured data in order to solve an interesting "real world" problem. This paper describes a class project for an undergraduate NLP course that gives students the opportunity to explore the data of Dialog State Tracking Challenge 2 (DSTC 2). Student background, curriculum choices, and project details are discussed. The paper concludes with some advice for instructors who might be interested in incorporating a DSTC 2 based project into their course and final reflections about student feedback.

2 Background

At East Carolina University, a Natural Language Processing course was added to our curriculum during academic year 2015-2016. It is a junior/senior level course for which the prerequisites are data structures and introductory statistics. Significant factors that influenced the decisions about curriculum and pedagogy during the initial offerings of the course (spring semesters 2017 through 2019) include the following.

- While we also have a machine learning course, it is not a prerequisite. Consequently, the NLP instructor cannot assume all students have completed the machine learning course.
- While Python is a rapidly growing programming language of choice, our students have not had significant exposure to the language prior to the NLP course. We use Java and C++ in our introductory and data structures courses.
- Only a small percentage of our students choose graduate study and almost all of those who do only seek a terminal masters degree. Nevertheless, it is important to expose undergraduate computer science majors to the tools of research if for no other reason than to give them an awareness of how research advancements are made.
- It is this author's belief that an undergraduate degree in computer science is merely a "license to learn." It is essential that our students understand the necessity of and gain experience with self-directed learning before they graduate.

Based on these factors, *Natural Language Processing with Python* (Bird et al., 2009) was chosen as the textbook for the course. It had been the basis for a successful independent study course with a Duke University undergraduate during spring 2013. Notable advantages of this choice include the following.

1. Gives students the opportunity to engage in some self-directed learning of a new programming language (Python) and a new API (the Natural Language Toolkit (Bird et al., 2008)).
2. Provides a gentle introduction to machine learning techniques suitable for our students

Proceedings of the Fifth Workshop on Teaching NLP, pages 70–79

who had not yet taken the machine learning course.

3. Offers a low cost to students as it is available free of charge online.

4. Provides a rich collection of exercises by which students can begin to gain proficiency and confidence in working with collections of structured data.

A main disadvantage of this text is its limited discussion of linguistic theory, but that can be addressed by other assigned readings.

With the cooperation of the department chair, the class size was restricted (15 in 2017, 25 in 2018, and 29 in 2019). This enabled the opportunity to provide more individualized learning opportunities. The primary such opportunity was the class project using the DSTC 2 data that was assigned for the spring 2018 and spring 2019 offerings of the course. Discussion of this project will be the focus of the remainder of the paper.

3 DSTC 2 Overview

For DSTC 2 a general discussion of the challenge and challenge results are provided in (Henderson et al., 2014a). The ground rules for the challenge are specified in (Henderson et al., 2013). A summary of the details most relevant to its use for a class project is provided below.

3.1 Problem Domain

The dialog environment was a telephone-based dialog system where the user task was to obtain restaurant information. During data collection each system user was given a dialog scenario to follow. Example scenario descriptions extracted from two of the log files are given below.

- `Task 09825: You want to find a cheap restaurant and it should be in the south part of town. Make sure you get the address and phone number.`

- `Task 11937: You want to find an expensive restaurant and it should serve portuguese food. If there is no such venue how about north american type of food. You want to know the phone number and postcode of the venue.`

The basic structure of the dialogs has the following pattern.

1. Acquire from the user a set of constraints about the type of restaurant desired. Users may supply constraint information about area, food, name, and price range. This phase may require multiple iterations as user goals change (such as from portuguese food to north american food for task 11937).

2. Once the constraints have been acquired, provide information about one or more restaurants that satisfy the constraints. Users may request that additional attributes about a restaurant be provided (such as address and phone number).

An example transcript of an actual dialog for completing task 11937 is provided in appendix A.

As described in the challenge handbook (Henderson et al., 2013), during each call, the dialog system logged detailed information that provides the needed input for a separate module for handling dialog state tracking. Further details about the data collection process are described next.

3.2 Data Collection Process

There were two different speech recognition (SR) models and 3 different dialog management models for a total of six different dialog systems that were used in the data collection process. Approximately 500 dialogs were collected using each of the six systems. There were a total of 184 unique users that were recruited using Amazon Mechanical Turk. Data using two of the dialog managers across both SR models (i.e., four of the six dialog systems) were used for training and development while data collected using the third dialog manager (1117 dialogs) was used as the test set for evaluation.[1]

4 Possible Activities with the DSTC 2 Data

For a group of students with sufficient technical background with Python and/or machine learning, a class project that allows students to develop their own dialog state tracking system is certainly feasible. Students could start from scratch and enhance

[1] The total number of dialogs was 3235. They were subdivided by the challenge organizers into a training set of 1612 dialogs, a dev set of 506 dialogs and then the test set of 1117 dialogs.

one of the rule-based baseline trackers that are provided in the DSTC 2 dataset or else develop or enhance a machine learning approach for tracking (e.g. (Williams, 2014), (Henderson et al., 2014b))[2]. In either case students should base their approach on a data-driven analysis of the nature of the dialogs.

However, this is not the only possible use of the data. In a circumstance where students do not have the necessary background to develop their own tracker, they can still engage with the data by developing their own analysis tools to glean information useful for studying other aspects of dialog processing such as miscommunication handling. Given the instructor's personal research interests and the students' background, this seemed to be the best way to proceed with a project as described next.

5 In-class Project Activities

About midway through the semester after completing the first five chapters of *Natural Language Processing with Python*, a week of class is taken to introduce students to dialog state tracking and the project. The goals for that week of class include the following.

- Introduce students to the natural language dialog problem. Resources used include a video of the Circuit Fix-it Shoppe dialog system (Hipp and Smith, 1993) and an introductory paper on natural language interfaces (Smith, 2005).
- Introduce students to the dialog state tracking problem using selected information from the first four sections of the DSTC 2 handbook (Henderson et al., 2013).
- Introduce students to the project requirements (see section 6).
- Provide a brief introduction to the structure of the raw data used in the project. The data is represented using JavaScript Object Notation (JSON). This introduction includes a template Python program that can be used as the basis for the software development activities of the students. Detailed study of appendix A in the challenge handbook is essential for successful completion of the project. This template program can access all dialogs based on a list of dialog ID's that are specified in a file whose name is specified as a command line argument. For each accessed dialog the template program extracts and displays the dialog acts of the system and speaker.
- Introduce students to other data resources available for the project. Besides the raw data, a supplemental resource that is provided are annotated transcripts of the dialogs. To keep students from being overwhelmed, each student is assigned the dialogs of a specific speaker. Several of the speakers interacted with all six of the dialog systems. Each student is assigned a different speaker.[3] The total number of dialogs in each set is between 15 and 20 dialogs per speaker. Besides the transcribed system output and Automatic Speech Recognition (ASR) input, other information provided in the annotated transcript includes the following.[4]
  1. The formal dialog act of the system utterance.
  2. The list of hypotheses provided by the Spoken Language Understanding (SLU) module including scoring.
  3. The actual transcription of what was spoken by the user.
  4. The chosen hypothesis of the SLU along with a comparison of that hypothesis to what was actually spoken.
  5. The current state of the dialog tracking process with respect to the informable attributes (area, food, name, and pricerange) for which information has been provided at some point in the dialog. Note that as in the case of the sample dialog for task 11937, the information for a specific goal can change (such as the food preference changing from "portuguese" to "north american".).

There are a few other in-class activities that relate more specifically to the project requirements.

[2]There are also several other papers related to DSTC 2 in the SIGDIAL 2014 proceedings (Georgila et al., 2014). The two specifically cited discuss the two best performing trackers in the original challenge.

[3]Besides acting as a limit on the amount of data a student had to consider, another reason for this was to generate data that could be used to study if speakers behaved differently with the different dialog systems.

[4]Details about the formal notation are provided in (Henderson et al., 2013).

They will be mentioned in the following section that discusses the project requirements.

6 Project Requirements

Students are required to submit a separate Python program for carrying out each of the following tasks.

1. Basic performance analysis of the speech quality.
2. Automatic annotation of dialog state change.
3. Automatic generation of dialog feature information for miscommunication detection.

More detailed discussion about each of these activities will follow.

6.1 Basic performance analysis of the speech quality

The intent of this requirement was to give students a gentle introduction to modifying the Python template program to access other elements in the raw dialog data. Their program was required to produce the following information for each user utterance.

- Number of words actually spoken by the human speaker.
- Number of words in the highest scored live speech recognition hypothesis.
- Total number of unique words found in the union of all the live speech recognition hypotheses.
- A label describing whether or not the utterance was understood.[5]

6.2 Automatic annotation of dialog state change

Dialog state change occurs when the user either supplies constraint information for possible restaurant recommendations (area, food, name, or price range) or else requests that additional attributes about a restaurant be provided (such as address and phone number). In this task students must implement a program that tracks the changes in these supplied values as the dialog proceeds. The algorithm for carrying out this task was discussed as part of the in-class activities. For each user utterance, the program had to specify the set of attributes for which (1) a new value had been supplied; (2) a modified value has been supplied; and (3) a value has been removed.

6.3 Automatic generation of dialog feature information for miscommunication detection

This part of the project gave students a chance to conduct their own analysis and offer their own insight into what readily computable features of the dialogs might be helpful to a dialog manager in determining that miscommunication occurred. Each student was required to propose three possibilities for feature detection, and in a one-on-one meeting, we would discuss the options and settle on a particular choice.[6] Details about chosen features and the results are presented in section 9.3.

This particular project requirement was used spring semester 2018 but not spring semester 2019. The reason for this was not due to any problem with the requirement other than the amount of time required by the instructor to meet with the students and then evaluate the work. Unfortunately for spring semester 2019 due to workload constraints the instructor did not have sufficient time to oversee and evaluate that requirement. Instead a standardized third requirement was used that asked students to conduct a performance analysis for the SLU module that looked at its performance as a function of the presence or absence of a "high-scoring" NULL semantic hypothesis where the definition of "high-scoring" was specified as a run-time parameter.

6.4 Project Report Requirements

Detailed requirements are provided in appendix B. The intent of the report requirement was to give students a chance to reflect on the dialog state tracking problem and its relationship to detection of dialog miscommunication. In earlier course assignments, students had been asked to reflect and write about the domain problem at hand. One such assignment was from the second chapter of the *Natural Language Processing with Python* textbook where students were asked to calculate word frequencies for words of their choice for five different genres in the Brown corpus. Students were asked to come up with words whose presence or

[5] This required using a function call to an instructor-provided Python module.

[6] Students were required to submit their proposal for advance review. Meetings were scheduled at 10 minute intervals.

absence might be typical for that genre. In their reflection, students were asked to explain their rationale for the genres they chose and to discuss the sequence of insights/lessons learned as different sets of words were tested. Students were specifically challenged to provide evidence of thoughtful inquiry—demonstrate a sequence of cycling through hypothesis, test, result, and refinement. It was hoped that prior experience with this mode of activity would be helpful while working on the project.

7 Project Assessment

7.1 Weightings

The three parts of the project were weighted as follows.

1. Speech quality performance (30%).
2. Dialog state change annotation (40%).
3. Generation of dialog feature information (2018)/SLU performance (2019) (30%).

For each part, code correctness/quality was weighted at 70% while report quality was weighted at 30%.

7.2 Evaluating code correctness/quality

For the first two parts of the project as well as the 2019 part 3 requirement to analyze SLU performance, it is certainly possible to base code correctness on automated testing. Unfortunately, that is likely to penalize excessively logic errors of a minor nature that could lead to large discrepancies in output results. Given the limited scope of each program, a checklist of features can be manually inspected reasonably quickly. A time estimate would be 20 to 30 minutes per student. As would be expected, correct solutions do not take as long to check.

Checking correctness of the 2018 part 3 requirement where students implemented their own feature generation algorithm is not as straightforward. While it was possible to steer the students towards a total of about five different dialog features rather than more than 20 different programs, it was still not a simple task.

7.3 Evaluating report quality

Two main questions were examined.

1. Did the student address each of the report requirements?
2. Does the report exhibit evidence that the student seriously looked at the results of running the software and base their observations on that.

As might be expected, there was a wide disparity in the quality of the efforts. Some excerpts from the reports are the following.

- "Part 3 of this assignment seems to work when it wants to."
- "... much of our language is fluff."

In contrast, some students wrote multiple paragraphs analyzing details and making tangible proposals for how to use the results to help with handling dialog miscommunication. One student who was concurrently enrolled in an information retrieval course and had learned about Naive Bayes classifiers chose to explore using the data produced from part 1 (speech quality), and implement a classifier to see how it would perform (unsurprisingly, not well).

Fortunately in general, many students engaged in meaningful self-discovery of principles for effective human-computer dialog interaction such as the usefulness of careful design in the phrasing of questions to the user. Several students also noted that excessive user terseness is not always helpful.

8 Classroom reinforcement of their research efforts

Given the challenging nature of this project to our students, it would have been unwise to have spent the final weeks of the semester with excessive presentation of new material. To reinforce the goal of student exposure to the process of research and to connect their work to the research community, two 50 minute class periods were used looking at four papers from the 2014 SIGDial conference where the work from DSTC 2 was presented. Class time was spent watching the videos of the original conference presentations of these papers ((Henderson et al., 2014a), (Smith, 2014), (Williams, 2014), and (Henderson et al., 2014b)). These are available at `https://www.superlectures.com/sigdial2014/`. This classroom activity was conducted two weeks prior to the end of the

semester. The final week was spent being available for additional consultation about projects as well as spending class time on answering questions about NLP that were posed by students at the beginning of the semester. This was a chance to help them see what they had learned during the semester, and to understand better what remains an unsolved problem.

9 Student Outcomes

9.1 Part 1: Speech quality performance

This requirement served its purpose beautifully. One problem that exists in undergraduate computer science study is the pervasive belief that almost any programming question is answerable by an Internet search—you just have to submit the magic words to get the answer to appear. To meet this requirement and do the rest of the project, students really had to study the handbook and apply the information it contained to the provided template program. This was mentioned several times in student project reports in response to the "what was your biggest challenge and how did you overcome it?" question.

9.2 Part 2: Dialog state change annotation

The algorithm presentation in class did not go into the details of how to access the needed data. Again, students had to apply the lessons learned from part 1 as well as further investigate the handbook to understand how to access needed data. Between the data access and the required data structures needed in their implementation, students because much more capable of identifying the differences in use between Python lists and dictionaries. This part of the project tended to be the most challenging for the students.

9.3 Part 3: Generation of dialog feature information (2018)

Besides setting a flag to indicate repetition of a system response, the primary choice of students was to drill deeper into the ASR and SLU data. The most common approaches are given below.

- Counting the occurrences of an <attribute,value> pair (e.g. <food,italian>) in the different SLU hypotheses.
- Calculating cumulative confidence in an <attribute,value> pair over the various SLU hypotheses.
- Combining SLU score with utterance length in some fashion.
- Detecting the presence or absence of the NULL (no interpretation) hypothesis for the SLU.
- Detecting the presence or absence of <attribute,value> information in the SLU that appeared in the ASR.

Given the technical challenges required, the final item was only attempted by a few students.

A key misconception that arose in several students initial proposals for feature generation was a confusion over what is available at the time the dialog is occurring. Several students tried to propose using information only available after the dialogs had completed (i.e. using the correctness annotations for what was actually spoken and what the correct SLU hypothesis updated dialog state tracking values should be). While this information can be helpful in reassessing the performance of dialog system modules, it cannot be directly used in detecting miscommunication during an ongoing dialog. I believe that their work on part 2: dialog state change annotation led to this confusion since they were using the post-dialog annotations to complete that task. An extra 15 minutes of class time addressing this issue prior to student proposals should have cleared up this misconception.

9.4 Part 3: SLU performance (2019)

Most students implemented this correctly. This was one place where it might have been helpful to have the students also run their solution on the entire DSTC 2 dataset as well as the dialogs of their assigned speaker to see if the SLU performance on their speaker was representative of overall performance.

10 Advice for Instructors

Besides in an introductory undergraduate NLP class for computer science majors with limited background in machine learning and Python, I believe a project based on the DSTC 2 data can be used in a variety of contexts.

- In an advanced NLP class where students already have a machine learning background.
- In an advanced undergraduate data structures class. The project could be a means to get students interested in NLP based on a common

activity—having a conversation with someone else.

- In an accelerated summer camp environment for talented undergraduates.

If interested in using the DSTC 2 data for a class project, I would suggest the following steps.

1. Go to https://github.com/matthen/dstc and download the following files at a minimum.
   - `handbook.pdf`
   - `dstc2_traindev.tar.gz` - Train and development datasets for DSTC 2.
   - `dstc2_test.tar.gz` - Test dataset for DSTC 2.
   - `dstc2_scripts.tar.gz` - Evaluation scripts, baseline tracker and other tools.
   - `HWU_baseline.zip` - Alternative baseline tracker, provided by Zhuoran Wang.
2. Install the downloaded files and make sure you can run one of the supplied trackers on the entire dataset.
3. Streamline the baseline tracker code and/or scoring code to access and process a subset of the features from the datasets. Alternatively, you may wish to consult the teaching materials web site for this workshop to access the demo scripts that should be made available.
4. Explore the `/scripts/config` subdirectory within the installation to understand the use of the `flist` files for listing the specific dialogs to be processed during the execution of a script.

Once you've successfully completed these tasks, you should be well on your way to developing your own project with this data. Feel free to consult the author of this paper as needed.

11 Lessons Learned

Student feedback was largely anecdotal and quite favorable. While it is certainly the case that in response to the "What would you change to improve the course?", the most common answer was to have more time spent on classifiers/machine learning, students did enjoy working on the project. The most interesting piece of feedback that was acquired was from a student who originally submitted an incomplete project during the first offering of the assignment (spring 2018). The student was allowed an extra couple of weeks to complete the project. At some point during fall semester 2018 the student took the time to communicate with me to let me know that the student had used the project as part of a job interview when asked to give a technical presentation about a previous project the student had completed. The student said, "...the final project in your NLP class was by far the most interesting one that I've been assigned in college ...".

While this was gratifying feedback, the more important outcome is the belief that the course and project did achieve the goals of giving students experience with self-directed learning and engagement with the tools of research. We are fortunate in NLP to have a wide variety of interesting problems available to us that will naturally intrigue our students regardless of their original interest in NLP. If one wishes to use dialog processing as the interesting problem, the DSTC 2 data is a valuable resource for use in the classroom. Regardless, I would encourage instructors to pick a task that excites them. Our teaching is much stronger when we are demonstrating the need for self-directed learning ourselves. Our research provides such a means.

12 Acknowledgements

I would like to thank my computer science department chair, Dr. Venkat Gudivada, for allowing me to teach the initial offerings of the course with a restricted enrollment. This enabled me to offer more individualized exploratory research opportunities with our undergraduates.

None of this would have been possible without the existence of the DSTC 2 challenge. A special thanks goes to the DSTC 2 organizers, Matthew Henderson, Blaise Thomson, and Jason Williams as well as to the University of Cambridge, Microsoft Research, and SIGDIAL (Special Interest Group on Discourse and Dialogue) for their sponsorship and support of the challenge.

A final thanks goes to the anonymous reviewers who helped me better understand what would be of most value to the readers of this paper.

References

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit*. O'Reilly, Beijing. Current version of the book is available at http://www.nltk.org/book/.

Steven Bird, Ewan Klein, Edward Loper, and Jason Baldridge. 2008. Multidisciplinary instruction with the natural language toolkit. In *Proceedings of the Third Workshop on Issues in Teaching Computational Linguistics*, pages 62–70, Columbus, Ohio. Association for Computational Linguistics.

Kallirroi Georgila, Matthew Stone, Helen Hastie, and Ani Nenkova, editors. 2014. *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*. Association for Computational Linguistics, Philadelphia, PA, U.S.A.

M. Henderson, B. Thomson, and J. Williams. 2013. *Dialog State Tracking Challenge 2 & 3*. https://github.com/matthen/dstc/blob/master/handbook.pdf.

M. Henderson, B. Thomson, and J. Williams. 2014a. The second dialog state tracking challenge. In *Proceedings of the SIGdial 2014 Conference*, pages 263–272, Philadelphia, U.S.A. Association for Computational Linguistics.

Matthew Henderson, Blaise Thomson, and Steve Young. 2014b. Word-based dialog state tracking with recurrent neural networks. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 292–299, Philadelphia, PA, U.S.A. Association for Computational Linguistics.

D. R. Hipp and R. W. Smith. 1993. A demonstration of the "circuit fix-it shoppe". In *Video Proceedings of the AAAI Conference*.

Ronnie Smith. 2014. Comparative error analysis of dialog state tracking. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 300–309, Philadelphia, PA, U.S.A. Association for Computational Linguistics.

R.W. Smith. 2005. Natural language interfaces. In *Encyclopedia of Language and Linguistics*, 2 edition, pages 496–503. Elsevier Limited, Oxford.

Jason D. Williams. 2014. Web-style ranking and SLU combination for dialog state tracking. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 282–291, Philadelphia, PA, U.S.A. Association for Computational Linguistics.

A Sample Dialog Transcript

Figure 1 is the transcript of an actual dialog for completing Task 11937 (description listed in section 3.1). SYS denotes utterances by the computer system and ASR is the speech returned by the speech recognizer. This sample dialog was used during class presentation about DSTC 2. It was used in order to make clear to the students that what ASR returns is not always what was actually said, but it is close enough in this example so that the intent is still understood. A more detailed transcript including what was actually said is also made available to the students.

B Sample Project Report Requirements

This information comes from the second offering of the dialog state tracking project during spring semester 2019. The report requirements for the previous year (where part three of the project allowed students to choose a dialog miscommunication feature to extract) is identical except the final paragraph was omitted.

B.1 Report Organization

Your report should have three sections, one section for each required investigation.

Section 1: Basic performance analysis of the speech quality

Section 2: Automatic annotation of dialog state change

Section 3: Basic performance analysis of the SLU module

Each section should contain the following.

- A list of any known deficiencies in the software. If you have no known deficiencies, explicitly say so.
- A description of the biggest challenge you faced in successfully completing the software for the given investigation. Did you overcome the challenge and if so, how?
- A discussion of your most interesting observation based on the data produced by the software. A quality discussion will not only present the results, but also present some thoughtful analysis that relates to the challenge of dialog state tracking. Possible contexts in which to frame the discussion include the following.

1. An observation based on differences as a function of the dialog system being used (recall that each `S[0-2]A[0-1]` combination represents a different dialog system).
2. An observation based on differences as a function of dialog type (goal change vs. non-goal change).
3. An observation that provides insight about one or more areas of exploration that could lead to improved performance by dialog state trackers, especially in regards to detecting possible miscommunication.

NOTE: You are not required to use the same context (or any of these contexts for that matter) for each section. Use whatever seems appropriate based on your observations as it relates to the specific topic.

IMPORTANT: A quality discussion of your most interesting observation for Section 3 should also include a proposal for either extending the program you wrote, or proposing a completely new software tool (i.e. program) for deeper investigation of the relevant issues you have identified for improving performance in dialog state tracking.

```
    Session: data/Mar13_S2A0/voip-72e50baa85-20130326_053416

Turn 0
SYS  > Hello , welcome to the Cambridge restaurant system. You can ask for
restaurants by area , price range or food type . How may I help you?
ASR  > i don't mind an expensive restaurant that serves portuguese food

Turn 1
SYS  > I'm sorry but there is no restaurant serving expensive portuguese food
ASR  > what about north american food

Turn 2
SYS  > I'm sorry but there is no restaurant serving expensive portuguese food
ASR  > north american food

Turn 3
SYS  > Could you please repeat that?
ASR  > what about north american food

Turn 4
SYS  > gourmet burger kitchen serves north american food in the expensive
price range
ASR  > what is the phone number

Turn 5
SYS  > The phone number of gourmet burger kitchen is 01223 312598 .
ASR  > what is the post code

Turn 6
SYS  > The post code of gourmet burger kitchen is C.B 2, 1 A.B
ASR  > bad

Turn 7
SYS  > I am sorry but there is no other north american restaurant in the
expensive price range
ASR  > thank you been bad

Turn 8
SYS  > gourmet burger kitchen serves north american food in the expensive
price range
ASR  > can you get back

Turn 9
SYS  > Can I help you with anything else?
ASR  > thank you good bye
```

Figure 1: Sample dialog for completing Task 11937