Zero to Spoken Dialogue System in One Quarter: Teaching Computational Linguistics to Linguists Using Regulus

Beth Ann Hockey
Department of Linguistics,
UARC
UC Santa Cruz
Mail-Stop 19-26, NASA Ames
Moffett Field, CA 94035-1000
bahockey@ucsc.edu

Gwen Christian
Department of Linguistics
UCSC
Santa Cruz, CA 95064, USA
jchristi@ucsc.edu

Abstract

This paper describes a Computational Linguistics course designed for Linguistics students. The course is structured around the architecture of a Spoken Dialogue System and makes extensive use of the dialogue system tools and examples available in the Regulus Open Source Project. Although only a quarter long course, students learn Computational Linguistics and programming sufficient to build their own Spoken Dialogue System as a course project.

1 Introduction

Spoken Dialogue Systems model end-to-end execution of conversation and consequently require knowledge of many areas of computational linguistics, speech technology and linguistics. The structure of Spoken Dialogue Systems offers a ready-made structure for teaching a computational linguistics course. One can work through the components and cover a broad range of material in a grounded and motivating way. The course described in this paper was designed for linguistics students, upper-division undergraduate and graduate, many having limited experience with programming or computer science. By the end of a quarter long course, students were able to build a working spoken dialogue systems and had a good introductory level understanding of the related computational linguistics topics.

When this course was first being contemplated, it became apparent that there were a number of somewhat unusual properties that it should have, and a number of useful goals for it to accomplish. The Linguistics Department in which this course is given had only sporadic offerings of computational courses, due in part to having no faculty with a primary focus in Computational Linguistics. linguistics students are very interested in having courses in this area, but even in the University as a whole availability is limited. A course on information extraction is offered in the Engineering School and while some linguistics students are equipped to take that course, many do not have sufficient computer science background or programming experience to make that a viable option.

This course, in the Linguistics department, needed to be for linguistics students, who might not have well-developed computer skills. It needed to fit into a single quarter, be self-contained, depend only on linguistics courses as prerequisites, and give students at least an overview of a number of areas of CL. These students are also interested in connections with industry; now that there are industry jobs available for linguists, students are eager for internships and jobs where they can apply the skills learned in their linguistics courses. Given this, it was also important that the students learn to program during the course, both to make engineering courses more accessible, and to attract potential employers.

In addition, since the department was interested in finding ways to expand computational linguistics offerings, it clearly would be good if the course appealed to the students, the department's faculty and to higher levels of the University administration.

Proceedings of the Third Workshop on Issues in Teaching Computational Linguistics (TeachCL-08), pages 80–86,
Columbus, Ohio, USA, June 2008.

2 Class Demographics

Students in the course are a mix of graduates and upper-division undergraduates with a solid background in syntax and semantics but are not expected to have much in the way of programming experience. Familiarity with Windows, Unix and some minimal experience with shell scripting are recommended but not required. Students have been very successful in the course starting with no programming experience at all. Because the Linguistics department is especially strong in formal linguistics, and the courses typically require extensive problem sets, linguistics students have good aptitude for and experience working with formal systems and this aptitude and skill set seems to transfer quite readily to programming.

3 Regulus Open Source Platform

The Regulus Open Source Platform is a major resource for the course. Regulus is designed for corpus-based derivation of efficient domain-specific speech recognisers from general linguistically-motivated unification grammars. The process of creating an application-specific Regulus recogniser starts with a general unification grammar (UG), together with a supplementary lexicon containing extra domain-specific vocabulary. An application-specific UG is then automatically derived using Explanation Based Learning (EBL) specialisation techniques (van Harmelen and Bundy, 1988). This corpus-based EBL method is parameterised by 1) a small domain-specific training corpus, from which the system learns the vocabulary and types of phrases that should be kept in the specialised grammar, and 2) a set of "operationality criteria", which control the specialised grammar's generality. The application-specific UG is then compiled into a Nuance-compatible CFG. Processing up to this point is all carried out using Open Source Regulus tools. Two Nuance utilities then transform the output CFG into a recogniser. One of these uses the training corpus a second time to convert the CFG into a PCFG; the second performs the PCFG-to-recogniser compilation step. This platform has been used the base for an number of applications including The Clarissa Procedure Browser (Clarissa, 2006) and MedSLT (Bouillon et al., 2005)

The Regulus website (Regulus, 2008) makes available a number of resources, including compilers, an integrated development environment, a Regulus resource grammar for English, online documentation and a set of example dialogue and translation systems. These examples range from completely basic to quite complex. This material is all described in detail in the Regulus book (Rayner et al., 2006), which documents the system and provides a tutorial. As noted in reviews of the book, (Roark, 2007) (Bos, 2008) it is very detailed. To quote Roark, "the tutorial format is terrifically explicit which will make this volume appropriate for undergraduate courses looking to provide students with hands-on exercises in building spoken dialog systems." Not only does the Regulus-based dialogue architecture supply an organizing principle for the course but a large proportion of the homework comes from the exercises in the book. The examples serve as starting points for the students projects, give good illustrations of the various dialogue components and are nice clean programming examples. The more research-oriented material in the Regulus book also provides opportunities for discussion of topics such as unification, feature grammars, ellipsis processing, dialogue-state update, Chomsky hierarchy and compilers. Reviewers of the book have noted a potential problem: although Regulus itself is open source it is currently dependent on two commercial pieces of software, SICStus Prolog and Nuance speech recognition platform (8.5). Nuance 8.5 is a speech recognition developer platform that is widely used for build telephone call centers. This developer kit supplies the acoustic models which model the sounds of the language, the user supplies a language model which defines the range of language that will be recognized for a particular application. This dependance on these commercial products has turned out not to be a serious problem for us since we were able to get a research license from Nuance and purchase a site license for SICStus Prolog. However, beyond the fact that we were able to get licenses, we are not convinced that eliminating the commercial software would be an educational win. While, for example, SWI Prolog might work as well in the course the commercial SISCtus Prolog given a suitable port of Regulus, we think that having the students work with a widely used com-

mercial speech recognition product such as Nuance, is beneficial training for students looking for jobs or internships. Using Nuance also avoids frustration because its performance is dramatically better than the free alternatives.

4 Other Materials

The course uses a variety of materials in addition to the Regulus platform and book. For historical and current views of research in dialogue and speech, course sessions typically begin with an example project or system, usually with a video or a runnable version. Examples of system web materials that we use include: (Resurrected)SHRDLU (`http://www.semaphorecorp.com/misc/shrdlu.html`), TRIPS and TRAINS (`http://www.cs.rochester.edu/research/cisd/projects/trips/movies/TRIPS\_Overview/`), Galaxy (`http://groups.csail.mit.edu/sls//applications/jupiter.shtml`), VocalJoyStick (`http://ssli.ee.washington.edu/vj/`), and ProjectListen (`http://www.cs.cmu.edu/~listen/mm.html`)and NASA's Clarissa Procedure Browser (`http://ti.arc.nasa.gov/projects/clarissa/gallery.php?ta\=\&gid\=\&pid\=`).

Jurasfsky and Martin (Jurafsky and Martin, 2000) is used as an additional text and various research papers are given as reading in addition to the Regulus material. Jurafsky and Martin is also good source for exercises. The Jurafsky and Martin material and the Regulus material are fairly complementary and fit together well in the context of this type of course. Various other exercises are used, including two student favorites: a classic 'construct your own ELIZA' task, and a exercise in reverse engineering a telephone call center, which is an original created for this course.

5 Programming languages

Prolog is used as the primary language in the course for several reasons. First, Prolog was built for processing language and consequently has a natural fit to language processing tasks. Second, as a high-level language, Prolog allows students to stay on a fairly conceptual level and does not require them to spend time learning how to handle low-level tasks. Prolog is good for rapid prototyping; a small amount of Prolog code can do a lot of work and in a one quarter class this is an important advantage. Also, Prolog is very close to predicate logic, which the linguistics students already know from their semantics classes. When the students look at Prolog and see something familiar, it builds confidence and helps make the task of learning to program seem less daunting. The declarative nature of Prolog, which often frustrates computer science students who were well trained in procedural programming, feels natural for the linguists. And finally, the Regulus Open Source System is written mainly in Prolog, so using Prolog for the course makes the Regulus examples maximally accessible.

Note that Regulus does support development of Java dialogue processing components, and provides Java examples. However, the Java based examples are two to three times longer, more complicated and less transparent than their Prolog counterparts, for the same functionality. We believe that the Java based materials would be very good for a more advanced course on multimodal applications, where the advantages of Java would be evident, but in a beginning course for linguists, we find Prolog pedagogically superior.

A potential downside to using Prolog is that it is not a particularly mainstream programming language. If the course was solely about technical training for immediate employment, Java or C++ would probably be better. However, because most students enter the course with limited programming experience, the most important programming outcomes for the course are that they end up with evidence that they can complete a non-trivial programming project, that they gain the experience of debugging and structuring code and that they end up better able to learn additional computer science subsequent to the course. The alternative these students have for learning programming is to take traditional programming courses, starting with an extremely basic introduction to computers course and taking 1-2 additional quarter long courses to reach the level of programming sophistication that they reach in one quarter in this course. In addition, taking the alternative route, they would learn no Computational Linguistics, and would likely find those courses much

less engaging.

6 Course Content

Figure 6, depicts relationships between the dialogue system components and related topics both in Linguistics and in Computational Linguistics and/or Computer Science. The course follows the flow of the Dialogue System processing through the various components, discussing topics related to each component. The first two weeks of the course are used as an overview. Spoken Dialogue Systems are put in the context of Computational Linguistics, Speech Technology, NLP and current commercial and research state of the art. General CL tools and techniques are introduced and a quick tour is made of the various dialogue system components. In addition to giving the students background about the field, we want them to be functioning at a basic level with the software at the end of two weeks so that they can begin work on their projects. Following the two week introduction, about two weeks are devoted to each component.

The speech recognition discussion is focused mainly on language modeling. This is an area of particular strength for Regulus and the grammar-based modeling is an easy place for linguists to start. Covering the details of speech recognition algorithms in addition to the other material being covered would be too much for a ten week course. In addition, the department has recently added a course on speech recognition and text-to-speech, so this is an obvious thing to omit from this course. With the Nuance speech recognition platform, there is plenty for the students to learn as users rather than as speech recognition implementers. In practice, it is not unusual for a Spoken Dialogue System implementer to use a speech recognition platform rather than building their own, so the students are getting a realistic experience.

For the Input Management, Regulus has implemented several types of semantic representations, from a simple linear list representation that can be used with the Alterf robust semantics tool, to one that handles complex embedding. So the Input Manager related component can explore the trade offs in processing and representation, using Regulus examples.

The Dialogue Managment section looks at simple finite state dialogue management as well as the dialogue state update approach that has typically been used in Regulus based applications. Many other topics are possible depending on the available time.

The Output Management unit looks at various aspects of generation, timing of actions and could also discuss paraphrase generation or prosodic mark up.

Other topics of a system wide nature such as N-best processing or help systems can be discussed at the end of the course if time allows.

7 Improvements for '08

The course is currently being taught for Spring quarter and a number of changes have been implemented to address what we felt were the weak points of the course as previously taught. It was generally agreed that the first version of the course was quite successful and had many of the desired properties. Students learned Computational Linguistics and they learned how to program. The demo session of the students' projects held at the end of the course was attended by much of the linguistics department, plus a few outside visitors. Attendees were impressed with how much the students had accomplished. In building on that success, we wanted to improve the following areas: enrollment, limiting distractions from the spoken dialogue material, building software engineering skills, making connections with industry and/or research, and visibility.

The first time the course was given, enrollment was six students. This level of enrollment was no doubt in part related to the fact that the course was announced relatively late and students did not have enough lead time to work it into their schedules. The small size was intimate, but it seemed as though it would be better for more students to be able to benefit from the course. For the current course, students knew a year and a half in advance that it would be offered. We also had an information session about the course as part of an internship workshop, and apparently word of mouth was good. With addition of a course assistant the maximum we felt we could handle without compromising the hands-on experience was twenty. Demand greatly exceeded supply and we ended up with twenty two students initially enrolled. As of the deadline for dropping without

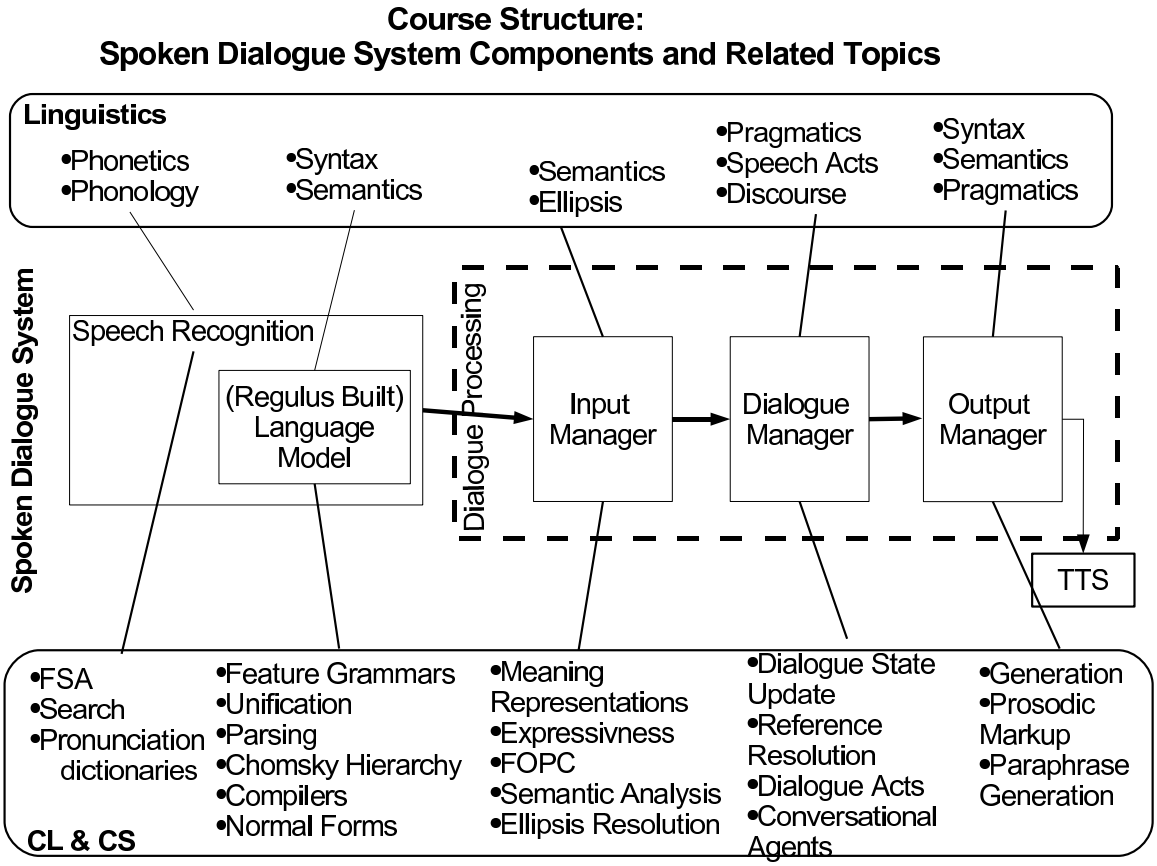

Figure 1: Course Schematic: Architecture of Dialogue System with associated linguistic areas/topic at above and Computational Linguistics and/or Computer Science topics below

penalty course enrollment is 16. The course is currently scheduled to be taught every other year but we are considering offering it in summer school in the non-scheduled years.

Two activities in the first incarnation of the course were time-consuming without contributing directly to learning about CL and Dialogue Systems. First, students spent considerable time getting the software, particularly the Nuance speech recognition software and the Nuance text-to-speech, installed on their personal machines. The variability across their machines and fact we did not at that time have a good way to run the software on Machintoshes contributed to the problem. This made the speech aspects seem more daunting than they should have, and delayed some of the topics and exercises.

For the current course, we arranged to have all of the software up and running for them on day one, in an instructional lab on campus. Mandatory lab sessions were scheduled for the course in the instructional lab, starting on the first day of class, so that we could make sure that students were able to run the software from the very beginning of the course. These arrangements did not work out quite as smoothly as we had hoped, but was still an improvement over the first time the course was taught.

Rather than being completely dependent on students' personal machines, the labs, combined with a strategy we worked out for running the software from external USB drives, provide students with a way to do their assignments even if they have unsuitable personal machines. In the labs, students are able to see how the software should behave if properly installed, and this is very helpful to them when installing on their personal machines. We refined the installation instructions considerably, which seemed to improve installation speed. The Macintosh problem has been solved, at least for Intel Macs, since we

have been successful in running the software with BootCamp. The twice weekly lab sessions also give students a chance do installation and practical lab exercises in an environment in which the course assistant is able see what they are doing, and give them assistance. Observing and getting help from more computationally savy classmates is also common in the labs. Athough the measures taken to reduce the software installation burden still leave some room for improvement, students were able to use Regulus and Nuance successfully, on average, in less than half the time required the first time the course was taught.

The other distracting activity was building the backend for the course projects. Spoken Dialogue Systems are usually an interface technology, but the students in the first offering of the course had to build their projects end to end. While this was not a complete loss, since they did get useful programming experience, it seemed as though it would be an improvement if students could focus more on the spoken dialogue aspects. The approach for doing this in the current course is to recruit Project Partners from industry, government, academic research projects and other university courses. Our students build the spoken dialogue system components and then work with their Project Partner to connect to the Project Partner's system. The students will then demonstrate the project as a whole, that is, their dialogue system working with the Project Partner's material/system, at the course end demo sessions. We have project partners working in areas such as: robotics, telephone based services, automotive industry, and virtual environments. There are a number of potential benefits to this approach. Students are able to spend most of their time on the spoken dialogue system and yet have something interesting to connect to. In fact, they have access to systems that are real research projects, and real commercial products that are beyond what our students would be capable of producing on their own. Students gain the experience of doing a fairly realistic software collaboration, in which they are the spoken dialogue experts. Project partners are enthusiastic because they get to try projects they might not have time or resources to do. Industry partners get to check out potential interns and research partners may find potential collaborators. In the previous version of the course, half of the students who finished the course subsequently worked on Spoken Dialogue oriented research projects connected with the department. One of the students had a successful summer internship with Ford Motors as a result of having taken the course. The research and industry connection was already there, but the Project Partner program strengthens it and expands the opportunities beyond projects connected with the department.

One enhancement to students' software engineering skills in the current version of the course is that students are using version control from day one. Each student in the course is being provided with a Subversion repository with a Track ticket system hosted by Freepository.com. Part of the incentive for doing this was to protect Project Partners' IP, so that materials provided by (particularly commercial) Project Partners would not be housed at the University, and would only be accessible to relevant student(s), the Project Partner, the instructor and the course assistant. The repositories also support remote collaboration making a wider range of organizations workable as project partners. With the repositories the students gain experience with version control and bug-tracking. Having the version control and ticket system should also make the development of their projects easier. Another way we are hoping to enhance the students software skills is through simply having more assistance available for students in this area. We have added the previously mentioned lab sections in the instructional labs, we have arranged for the course assistant to have substantial time available for tutoring, and we are posting tutorials as needed on the course website.

The final area of improvement that we wanted to address is visibility. This is a matter of some practical importance for the course, the addition of CL to the department's offerings, and the students. Visibility among students has improved with word of mouth and with the strategically timed information session held the quarter prior to holding the course. The course end demo session in the first offering of the course did a good job of bringing it to the attention of the students and faculty in the Linguistics Department. For the current course, the Project Partner program provides considerable visibility for students, the department, and the University, among industry, government and other Universities. We are

also expanding the demo session at the end of the course. This time the demo session will be held as a University wide event, and will be held at the main UC Santa Cruz campus and a second time at the University's satellite Silicon Valley Center, in order to tap into different potential audiences. The session at the Silicon Valley Center has potential for giving students good exposure to potential employers, and both sessions have good potential for highlighting the Linguistics department.

8 Summary and Conclusion

The course presented in this paper has three key features. First it is designed for linguistics students. This means having linguistics and not computer science as prerequisites and necessitates teaching students programming and computer science when they may start with little or no background. Second, the course takes the architecture of a Spoken Dialogue System as the structure of the course, working through the components and discussing CL topics as they relate to the components. The third feature is the extensive use of the Regulus Open Source platform as key resource for the course. Regulus material is used for exercises, as a base for construction of students' course projects, and for introducing topics such as unification, feature grammars, Chomsky hierarchy, and dialogue management. We have found this combination excellent for teaching CL to linguistics students. The grammar-based language modeling in Regulus, the use of Prolog and relating linguistic topics as well as computational ones to the various dialogue system components, gives linguistics students familiar material to build on. The medium vocabulary type of Spoken Dialogue system supported by the Regulus platform, makes a very motivating course project and students are able to program by the end of the course.

We discuss a number of innovations we have introduced in the latest version of the course, such as the Project Partner program, use of instructional labs and subversion repositories, and expanded course demo session. Since we are teaching the course for the second time during Spring Quarter, we will be able to report on the outcome of these innovations at the workshop.

Acknowledgments

We would like to thank Nuance, for giving us the research licenses for Nuance 8.5 and Vocalizer that helped make this course and this paper possible.

References

Johan Bos. 2008. A review of putting linguistics into speech recognition. the regulus grammar compiler. *Natural Language Engineering*, 14(1).

P. Bouillon, M. Rayner, N. Chatzichrisafis, B.A. Hockey, M. Santaholma, M. Starlander, Y. Nakao, K. Kanzaki, and H. Isahara. 2005. A generic multi-lingual open source platform for limited-domain medical speech translation. In *In Proceedings of the 10th Conference of the European Association for Machine Translation (EAMT)*, Budapest, Hungary.

Clarissa, 2006. http://www.ic.arc.nasa.gov/projects/clarissa/. As of 1 Jan 2006.

D. Jurafsky and J. H. Martin. 2000. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition*. Prentice Hall Inc, New Jersey.

M. Rayner, B.A. Hockey, and P. Bouillon. 2006. *Putting Linguistics into Speech Recognition: The Regulus Grammar Compiler*. CSLI Press, Chicago.

Regulus, 2008. http://www.issco.unige.ch/projects/regulus/, http://sourceforge.net/projects/regulus/. As of 1 Jan 2008.

Brian Roark. 2007. A review of putting linguistics into speech recognition: The regulus grammar compiler. *Computational Linguistics*, 33(2).

T. van Harmelen and A. Bundy. 1988. Explanation-based generalization = partial evaluation (research note). *Artificial Intelligence*, 36:401–412.