# A Natural Language Conversational System for Online Academic Advising

**Edward M. Latorre-Navarro** and **John G. Harris**
Computational Neuro-Engineering Laboratory, Elec. and Comp. Eng. Dept., Univ. of Florida, Gainesville, FL
elatorre@ufl.edu, harris@ece.ufl.edu

**Abstract**

We have designed an academic advising online system to advise college students using natural language conversations. The system embeds knowledge of current and future teaching schedules, degree requirements, course prerequisites and various administrative procedures. While this information can be found by searching several university websites and catalogs, students continually ask human advisors these questions during their limited face-to-face time, which limits deeper developmental and educative advising that is only available from human advisors. Our system enhances the advising experience by offering a source for instant academic advice that does not require student training or additional human resources. The system contains a pattern-matching dialog management system with access via a web browser. We describe the motivation for our system, the design requisites, our approach for deployment, and analyze results from real-world field tests.

## Introduction and Motivation

Academic advisors assist students in personal, academic, professional and social matters. Successful advising programs increase student retention, improve graduation rates and help students meet educational goals (Gordon et al. 2011). Advising tasks are identified as prescriptive, providing expert advice, and developmental, where the advisor engages in a mutual learning process with the student, in order to help the student's problem solving, decision making and evaluations skills (Appleby 2008).

Academic institutions are also adopting learning-centered educative advising, to guide students on the philosophy of the curriculum and provide them with the skills needed for long-term educational planning (Melander 2005; Hagen and Jordan 2008). To help advisors manage these tasks, advising research has focused on technologies such as instant messaging, social networking and course-management systems (Leonard 2008; NACADA 2012). We propose the next generation of interactive advising systems should include a natural language interface to allow students to communicate as freely as they do with their advisors. Such a system would allow students to easily ask a wider range of questions than those in traditional expert-based systems and obtain immediate responses instead of waiting for peers or advisors to reply. This application also responds to a digital generation that thrives on immediate gratification through firsthand capabilities (Prensky 2001; Junco and Mastrodicasa 2007). In essence, the fundamental objective of this work is to provide students with an advising experience that is as close as possible to traditional human interaction.



## Related Work

Several research publications describe expert-based systems for helping students with straightforward repetitive tasks such as choosing majors and accessing degree audits, e.g. (Nambiar and Dutta 2010; Feghali et al. 2011). Two publications show an advising system combined with natural language processing (NLP) techniques for communication, (McMahan 2010) and (Leung et al. 2010), however, to our knowledge, there are no academic advising systems with a full natural language interface.

Some advising tasks of our system are similar to features available in conversational agent (CA) systems for e-learning environments, such as those examined in (Gandhe et al. 2009; Mori et al. 2013). The evolution of these CAs, shows examples of initially using pattern-matching techniques, followed by statistical methods as sufficient data becomes available. The success of many of these CAs suggests the viability of the advising system we propose.

## System Fundamentals

Kerly, et al. presents a series of questions for developers of CAs for e-learning (Kerly et al. 2009). We group these into four key subjects to address the system requirements.

(i) What approach to NLP should the developer adopt? Most practical dialog systems are corpus-based, while the

latest research trends are on statistical methods (Jurafsky and Martin 2000; Nadkarni et al. 2011). The enclosed domain of our application and the lack of advising data favor a corpus-based approach. We decided to select an advanced CA system within the open source alternatives.

(ii) Who controls the direction of the interaction? Users initiate and control the dialog, including usage of follow-up questions and contextual references. The system must respond with the correct answer or state that it is not available. The system output must be succinct and prevent unnecessary follow-up questions or invite off-topic dialog.

(iii) What will users say? The system must include the unique vocabulary of the user field, as well as technical terms such as, e.g., the course-code COMP1234, and C++, either an unofficial course name or a programming language. Many of these terms must be collected from the user field by implicitly crowdsourcing the users through either the Wizard-of-Oz technique (Jurafsky and Martin 2000) or by releasing a hand built prototype system that they can feed. We decided to hand build the input patterns based on how users could request each available answer.

The user data contains intricacies such as implicitly conveyed information, ambiguity, contextual references, and similar to traditional text messaging, will not contain formal writing features such as punctuation and capitalization. To determine the intention of the user from the linguistic expression, we need to distinguish every significant lexical unit (LU) and their relationships within the context, i.e., we need effective keyword extraction (KE) and context-based disambiguation structures. To overcome the lack of data, we follow the approach of (Hulth 2004; Bellotti et al. 2011), which showed that parts-of-speech (POS) tags, noun-phrase chunks and lexical relations are significant features for KE algorithms, independent of the usual statistical term selection methods.

(iv) How much testing will be required? CA systems for e-learning often undergo over two thousand conversations before reaching operational state (Kerly et al. 2009).

Three additional key technical requirements include, first, that the system be available at all times through an online instrument such as a webpage or mobile application. Secondly, the system must obtain and store student academic data without compromising their private information. Third, the dialog system must systematically manage the complexities of multiple course sequences within an academic program and the multiple amendments that often occur in the academic field.

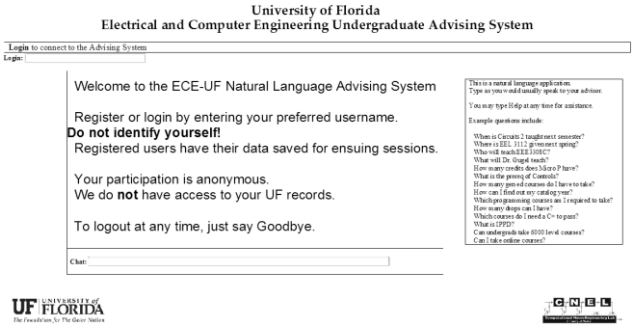
## The Instant Academic Advice System

We propose the INSTant Academic adVICE system (Instavice) to offer students the advising system described above. Instavice includes information about the academic programs, course schedules, answers to a wide range of academic FAQs, recommendations for the development of a course plan, and refers students to academic services. We created the knowledge base (KB) through university documents and reports from the academic advisors. We organized the KB as a list of triples containing a question, its answer and the topic to which we classified the answer.

For course schedule information, which is updated daily in the Registrar's webpage during enrollment period, we use an automated Python script to read the online data and update the system database. This process assures Instavice has the latest information without dependency of human maintenance. This information is available by specifying a term in a query, e.g., "Who taught C++ in Fall 2012".

The website was developed using Python, JavaScript, HTML, CSS and uses sockets for communication. The website is hosted on a desktop computer running on a Linux OS, with an Intel Pentium 4 processor and 1.8GB of RAM. The website was designed for speed with a load time under one second on contemporary versions of all major browsers. To protect user data, the system requires anonymous usernames. Anonymity also encourages students to provide feedback without fear of repercussions. Each account stores the user's data for ensuing sessions. Figure 1 shows a screenshot of the website.

The advising dialogue engine (ADE) in Instavice drives the input, output and states of the system. We built our ADE around ChatScript (CS), an open source scripting language for a rule-based dialog system available for download at (Wilcox 2011). We used CS version 2.0.




This is a natural language application.
Type as you would usually speak to your advisor.

You may type Help at any time for assistance.

Example questions include:

When is Circuits 2 taught next semester?
Where is EEL 3112 given next spring?
Who will teach EEE3308C?
What will Dr. Gugel teach?
How many credits does Micro P have?
What is the prereq of Controls?
How many gen ed courses do I have to take?
How can I find out my catalog year?
Which programming courses am I required to take?
How many drops can I have?
Which courses do I need a C+ to pass?
What is IPPD?
Can undergrads take 6000 level courses?
Can I take online courses?

*Figure 1: Above: screenshot of Instavice during the experiments. Below: The FAQ list on the right side frame of the screenshot.*

CS features a knowledge-based architecture, a query system for a dynamic database, it includes the WordNet ontology to manage synonymous expressions, and a client-server architecture that communicates using sockets. Our ADE runs as a client of the underlying Python script that manages Instavice, which allows integrating CS with external functions. For example, using methods in (Bird et al. 2009; Norvig 2007), we added a spell-checker using Python to manage the unique terms.

Similar to semantic grammars, scripts in CS comprise topics, patterns, concepts and facts. A topic is a collection of patterns for input matching. Patterns contain keywords and logic functions for conditioning a match. A concept is a collection of LUs. Facts, the elements of the database, are triples that contain LUs or other facts as field values.

To define the input patterns, we identified keywords, POS tags, noun-phrase chunks and lexical relations from our initial KB. We organized these features within the topics in which we pre-classified each question answer pair. For example, we defined the topic *course schedule information* to include the keywords "any course name", "a professor's name", the word *professor* and the word *teach*. This topic contains a pattern for an input such as, *who will teach C++ during the next semester*, where the extracted keywords are *who*, *teach*, *C++* and *next-semester*. Within CS, we generalized the keywords into *who*, "teach-word", "course-name" and "term-phrase", where

- Teach-word is any word that refers to a course being taught, e.g. *teach*, *teaches*, *lectures* and *give*.
- Course-name is a name from the list of all courses.
- Term-phrase is any phrase that refers to an academic term, e.g. *next semester*, *Fall 2012* and *next summer*.

Figure 2 shows the algorithm for this example. Clearly, this is not the only way to ask who will teach a specified course. The user could ask *Who is the professor of C++*, or if the request is within the context of a previous input, *Who is teaching it*. For the first case, we add a new pattern and map it to the pattern defined above. For the second case, we define a pattern that determines if the previous input was within the predefined context of course information.

Algorithm: **Match user request for a course offering**

**Input string S**: *who teaches C++ next semester*
**Desired output**: Name of whom teaches C++ in the next semester or *C++ is not offered next semester*

**If** S contains a keyword of the topic course_search
**If** S matches with a who-teaches pattern
  **If** S contains a term-phrase keyword
    Calculate the term T, using the date and phrase tense
  **Else**
    Use previous T value. Default value is the next term
  **If** the course C++ exists in the schedule of T
    **Find** the corresponding data element *Instructor*
      **Return** *C++ is taught by* Instructor *in* T
  **Else Return** *C++ not offered in* T

*Figure 2: Example of a procedure to match a user request.*

To determine the context, we use features such as the topic that previously matched, the current keywords and the state of the variables of potentially missing keywords.

In our CS script, the last two sequential topics manage the input queries that did not match to an available answer. The penultimate topic returns suggested questions related to matched keywords. The last topic contains default responses for unrecognized input. While many intricacies are involved in designing the close to 200 input patterns our CS script contains, a thorough description of all functionalities in CS is beyond the scope of this paper.

## Results and Analysis

We designed and deployed our system for undergraduate students of the Electrical and Computer Engineering Department of the University of Florida. Our system is accessible via http://advising.ece.ufl.edu. We advertised the website with a desktop computer setup in the student services office of the department during a six-week period between October and November 2012 and recorded data from 53 anonymous users who made at least three input entries. Users with less than three were discarded. A second testing period began when the department advisors emailed the students announcing our advising webpage. We recorded 103 users during this four-week period between March and April 2013. Students did not receive tutorials or demos on how to use the system, except for the instructions and example questions on the webpage.

Figure 3 shows example dialog extracted from multiple users. Responses by the system are curtailed for clarity and user syntactic errors are preserved for authenticity. For the first period, the average number of input entries per user was 16.0, where 49% had between four and ten entries. The second period results were 11.4 and 58% respectively.

While Instavice contains over 100 unique answers to academic FAQs, students did not seek many of these answers. Surveys from a sample of the users showed that they did not have a need for these answers at the time of the conversation or they lacked the academic experience to necessitate finding out the information. As seen in Figure 1, we have a short list of suggested questions as examples. Many students opted to start out with those questions, then try similar and follow-up questions. For these students, this exercise proved to be a useful academic informative session. We plan to extend that list of examples in order to inform the students of the system capabilities and incite them to learn the information.

Student feedback showed that the most requested features were personalized recommendations for course enrollment and to track their progress through the degree. Statistical analysis of the system responses was not included in this phase, as data acquisition and viability analysis were the main objectives, and not enough data was available for statistically significant conclusions.

| Albert is the Instavice persona for our implementation | |
|---|---|
| jer123: | *There is an error in my degree audit. How do I get my audit fixed?* |
| Albert: | The Registrar's office handles your degree audit… |
| dude: | *How many drops can I have* |
| Albert: | Students are given two drops prior to reaching ... |
| dude: | *What if I am out of drops?* |
| Albert: | A third drop must be petitioned. You may… |
| uhingb: | *What will Zmuda be teaching* |
| Albert: | Prof Zmuda, Henry is teaching in Fall 2013 EEL 4930 POWER SYSTEM ANALYSIS for 3 … |
| uhingb: | *tell me about EEL 4930* |
| Albert: | I'm working on getting a good synopsis of … |
| rlb1116: | *when is eel3701c in spring?* |
| Albert: | For Spring 2013 we have EEL 3701C is on … |
| rlb1116: | *what about next spring?* |
| Albert: | I don't have the Spring 2014 schedule yet. |
| jrs5269: | *what grade is considered passing for critical tracking courses?* |
| Albert: | You need to pass all the listed tracking courses … |
| tab7240: | *what if i drop and go below 12 credit hours* |
| Albert: | 12 credits is the minimum required to be a full … |
| slapeyre: | *Feedback* |
| Albert: | If you wanna leave me feedback, I'm listening. |
| slapeyre: | *put in all the course info and scheduling from the rest of the UF website pages* |

*Figure 3. Selected dialog extracted from multiple user log files. An ellipsis represents a response truncated for this illustration.*

## Conclusion and Future Work

We present natural language conversational system that provides instant academic advising for members of the academic community. Our system allows users to communicate analogous to how two people interact, thus not requiring a user learning curve. This system is an instrument to increase student integration, retention, satisfaction and performance, without requiring additional personnel, by allowing advisors to allocate resources to developmental and educative tasks, instead of repetitive time intensive tasks related to prescriptive advising.

With the data collected, we have originated a corpus of academic advising FAQs that will allow including statistical NLP techniques into future developments. As future work, we will integrate statistical measurements from the log files content and include indirect evaluations by the constituencies. The evaluation data will offer advisors documented assessment of the areas of advising that most concern students. Additional work includes adding an expert system for academic enrollment planning, a mechanism to forward selected conversations to advisors and allowing users to add lexical definitions.

## Acknowledgments

We appreciate the support from participating students, advisors and personnel of the student services office.

## References

Appleby, D.C. 2008. Advising as teaching and learning. *Academic advising: A comprehensive handbook*. San Francisco, CA. 2: 85-102: Jossey-Bass

Bellotti, F.; et al. 2011. Towards a conversational agent architecture to favor knowledge discovery in serious games. In *Proceedings of the 8th Int Conf on Advances in Computer Entertainment Technology*. 17.

Bird, S.; Klein, E. and Loper, E. 2009. *Natural language processing with Python*. O'Reilly Media, Incorporated.

Feghali, T.; Zbib, I. and Hallal, S. 2011. A web-based decision support tool for academic advising. In *International Forum of Educational Technology & Society*. Athabasca University, Canada: 14:1, 82-94.

Gandhe S.; et al. 2009. An Integrated Authoring Tool for Tactical Questioning Dialogue Systems. In *Proceedings of the 6th Workshop on Knowledge and Reasoning in Practical dialogue Systems*, Pasadena, CA: AAAI Press.

Gordon, V.N.; Habley, W.R. and Grites, T.J. eds. 2011. *Academic Advising: A Comprehensive Handbook*. Manhattan, KS: Wiley.

Hagen, P.L. and Jordan, P. 2008. Theoretical foundations of academic advising. *Academic advising: A comprehensive handbook*. San Francisco, CA: 2: 17-35: Jossey-Bass.

Hulth, A. 2004. *Combining machine learning and natural language processing for automatic keyword extraction*. Department of Computer and Systems Sciences, Institutionen för Data-och systemvetenskap.

Junco, R. and Mastrodicasa, J. 2007. *Connecting to the Net. Generation: What Higher Education Professionals Need to Know about Today's Students*. NASPA, Student Affairs Administrators in Higher Education.

Jurafsky, D. and Martin, J.H. 2000. Speech and language processing. *Speech and language processing*. Prentice Hall.

Kerly, Alice, Richard Ellis, and Susan Bull. 2009. Conversational Agents in E-Learning. *Applications and Innovations in Intelligent Systems XVI*. 169-182: Springer.

Leonard, M.J. 2008. Advising delivery: Using technology. *Academic advising: A comprehensive handbook*. San Francisco, CA: 292-306. Jossey-Bass

Leung, C.M.; et al. 2010. Intelligent Counseling System: A 24 x 7 Academic Advisor. *EDUCAUSE Quarterly* ERIC. 33:4.

McMahan, B. 2010. An Automatic Dialog System for Student Advising. *J. of Undergraduate Research*, Minnesota State University, Mankato.

Melander, ER. 2005. Advising as Educating: A Framework for Organizing Advising Systems. *NACADA Journal;* 25:2 84-91.

Mori D.; et al. 2013. An easy to author dialogue management system for serious games. *J. Comput. Cult. Herit.* 6:2 10:1-10:15.

NACADA – National Academic Advising Association. 2012. Advising Technology Innovation Awards. 1998–2012.

Nambiar, A.N. and Dutta, A.K. 2010. Expert system for student advising using JESS In *Proceedings of the International Conf on Educational and Information Technology*. 1:312.

Nadkarni P.M.; Ohno-Machado L. and Chapman W.W. 2011. Natural language processing: an introduction. In *Proceedings J Am Med Inform Assoc*; 18:5 544-551.

Norvig, P. 2007. How to write a spelling corrector. Unpublished. Available: http://norvig.com/spell-correct.html

Prensky, M. 2001. Digital natives, digital immigrants Part 2: Do they really think differently? *On the horizon*. 9:6 1-6: MCBUP Ltd

Wilcox, B. 2011. ChatScript project. Documentation and source code available for download at http://chatscript.sourceforge.net