

AN INTELLIGENT NATURAL LANGUAGE CONVERSATIONAL SYSTEM FOR
ACADEMIC ADVISING

By

EDWARD M. LATORRE-NAVARRO

A DISSERTATION PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

UNIVERSITY OF FLORIDA

2014

© 2014 Edward M. Latorre-Navarro

To my wife, family and friends

ACKNOWLEDGMENTS

I am thankful to Dr. John G. Harris, who encouraged my desire to become a Gator since I first approached him. Thanks to his vision and confidence in me, Albert is a smart, highly valued innovation that has been very rewarding to create.

I am appreciative of the professors on my PhD committee, whose courses directly influenced my path throughout this work. I thank Dr. Wind Cowles and her teachings in psycholinguistics and biological real-time language processing. I thank Dr. A. Antonio Arroyo for the introduction to NLP and machine intelligence. I thank Dr. Xiaolin Li for the experience in designing user-friendly cyber-physical systems and mobile applications.

I express my gratitude for the support of the Electrical and Computer Engineering Department of the University of Florida during my studies. I thank the staff of the ECE Student Services Office for their kind service working with Albert. I also thank my fellow comrades throughout all these years from the Hybrid Computational Group and Computational NeuroEngineering Laboratory. They have always offered a helping hand.

I thank my family and friends throughout Puerto Rico and the United States, for always believing in me and being there for me. I thank my neighbor Dr. José M.A. Almodóvar-Faría. I could not have asked for a better teammate.

I thank, with all my heart, my mother, my father, my brothers, sister, grandparents, uncles, aunts, cousins and in-laws. My greatest inspirations to be here, Ana I. Navarro-Vázquez, Pedro Navarro-Barros, Pedro L. Navarro-Vázquez, who now have a doctor in the family.

The person who is always by my side, with whom I share every experience and makes my world go round, I am grateful for my wife, Karen Vanessa Muratti-Rodríguez.

TABLE OF CONTENTS

| | <u>Page</u> |
|---|-------------|
| ACKNOWLEDGMENTS..... | 4 |
| LIST OF TABLES..... | 7 |
| LIST OF FIGURES | 8 |
| LIST OF ABBREVIATIONS..... | 10 |
| ABSTRACT..... | 12 |
| CHAPTER | |
| 1 INTRODUCTION..... | 14 |
| Academic Advising | 14 |
| Natural Language Processing | 16 |
| Natural Language Understanding..... | 17 |
| Conversational Agents | 19 |
| Question Answering Systems..... | 23 |
| Dissertation Topics | 24 |
| 2 A CONVERSATIONAL SYSTEM FOR ACADEMIC ADVISING..... | 26 |
| Academic Advising | 26 |
| The Role of the Academic Advisor | 26 |
| Academic Advising in ECE-UF | 27 |
| Fundamental Requirements for the Advising System | 28 |
| Albert – The Intelligent Academic Advising Conversational System..... | 32 |
| Advising Information..... | 33 |
| System Components..... | 34 |
| Advising Dialogue Manager | 37 |
| Academic Planning System..... | 38 |
| System Evaluation | 39 |
| 3 THE ADVISING DIALOGUE MANAGER | 41 |
| Natural Language Processing | 41 |
| Natural Language Generation | 42 |
| Natural Language Understanding..... | 42 |
| NLP Algorithm..... | 43 |
| Task Manager | 49 |
| Spell-Checking..... | 49 |
| Automatic Updates..... | 52 |
| Scaling-Up the Natural Language Scripts..... | 52 |

| | |
|--|-----|
| Statistical Data Collection..... | 54 |
| 4 THE ACADEMIC PLANNING SYSTEM..... | 56 |
| The Course Plan Analyzer..... | 57 |
| Obtaining the Academic Information..... | 57 |
| User Interface | 58 |
| The CoPA System..... | 58 |
| Albert's Advisor Website..... | 64 |
| Evaluating CoPA | 65 |
| 5 EXPERIMENTS AND ASSESSMENT | 66 |
| Experiments | 66 |
| Phase One | 68 |
| Phase Two | 69 |
| Phase Three | 71 |
| Analysis Summary..... | 84 |
| Current State of Albert..... | 85 |
| 6 CONCLUSION AND FUTURE WORK..... | 86 |
| Conclusion..... | 86 |
| Future Work..... | 87 |
| APPENDIX | |
| A TOPICS COVERED BY ALBERT | 88 |
| B SCREENSHOTS OF THE ALBERT WEBSITE..... | 89 |
| C SAMPLES FROM USER LOG FILES | 94 |
| D ADVISOR SURVEY QUOTES..... | 99 |
| LIST OF REFERENCES..... | 101 |
| BIOGRAPHICAL SKETCH..... | 107 |

LIST OF TABLES

| <u>Table</u> | | <u>page</u> |
|--------------|---|-------------|
| 5-1 | Results from the survey at the end of the CoPA process..... | 76 |
| 5-2 | Results from Albert's classification of the input statements..... | 79 |
| 5-3 | Results from the outcomes error estimation | 82 |
| 5-4 | System performance estimation results..... | 83 |

LIST OF FIGURES

| <u>Figure</u> | <u>page</u> |
|--|-------------|
| 1-1 Basic architecture of a dialogue system | 21 |
| 2-1 A model of the main components of Albert. | 35 |
| 2-2 A screenshot of the website for Albert in its initial state. | 36 |
| 3-1 Example of a procedure to match an input. | 45 |
| 3-2 Example of a procedure for an input..... | 51 |
| 3-3 Algorithm for course name submission..... | 53 |
| 3-4 Questionnaire required to submit the academic planning process | 55 |
| 4-1 A screenshot of Albert with the right side showing the first page of CoPA opened inside the frame. | 59 |
| 4-2 A segment of the template for entering the academic record in CoPA. | 60 |
| 4-3 A screenshot of Albert with the right side showing the second part of CoPA | 61 |
| 4-4 The template for choosing a course plan in CoPA, with an example of courses available to choose. | 62 |
| 4-5 Front page of the advisor website..... | 64 |
| 4-6 The advisor website, with an example student report. | 64 |
| 5-1 Input statement histogram for phase one. | 68 |
| 5-2 A screenshot of Albert, during phase two, showing the FAQ list on the right. | 70 |
| 5-3 Input statement histogram for phase two..... | 70 |
| 5-4 Statistics for the video tutorial..... | 73 |
| 5-5 Input statement histogram for phase three. | 74 |
| 5-6 Catalog year of each user who filled an academic record with CoPA. | 75 |
| 5-7 Survey results with a standard error bar for each question. | 76 |
| 5-8 Distribution of ON input statements. | 80 |
| B-1 Main webpage for Albert..... | 89 |

| | | |
|-----|---|----|
| B-2 | Main FAQ webpage..... | 90 |
| B-3 | Main webpage for Albert showing second FAQ page. | 90 |
| B-4 | Full list of questions in the second FAQ webpage. | 91 |
| B-5 | Main webpage for Albert showing page one of CoPA..... | 92 |
| B-6 | Main webpage for Albert showing an example page two of CoPA. | 92 |
| B-7 | Main webpage for Albert showing an example of the speech API..... | 93 |

LIST OF ABBREVIATIONS

| | |
|---------|---|
| AI | Artificial Intelligence |
| AIML | Artificial Intelligence Markup Language |
| API | Application Programming Interface |
| BSCEE | Bachelor of Science in Computer Engineering – Hardware emphasis |
| BSEE | Bachelor of Science in Electrical Engineering |
| CA | Conversational Agent |
| CoPA | Course Plan Analyzer |
| CS | Chatscript |
| DM | Dialogue Manager |
| DM | Dialogue Manager |
| ECE-SSO | Electrical and Computer Engineering Student Services Office |
| ECE-UF | Electrical and Computer Engineering – University of Florida |
| FAQ | Frequently Asked Questions |
| FN | False Negative |
| FP | False Positive |
| HCI | Human-computer interaction |
| IEEE | Institute of Electrical and Electronics Engineers |
| IR | Information Retrieval |
| ISIS | Integrated Student Information System |
| KB | Knowledge Base |
| KE | Keyword Extraction |
| LD | Lexical Disambiguation |
| LM | Literal Match |

| | |
|------|--------------------------------|
| LU | Lexical Unit |
| MT | Machine Translation |
| NLG | Natural Language Generation |
| NLP | Natural Language Processing |
| NLTK | Natural Language Toolkit |
| NLU | Natural Language Understanding |
| ON | Outcome Negative |
| OP | Outcome Positive |
| OS | Operating System |
| PM | Partial Match |
| POS | Part of Speech |
| POS | Parts-of-Speech |
| QA | Question Answering |
| TN | True Negative |
| TP | True Positive |
| WSD | Word Sense Disambiguation |

Abstract of Dissertation Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Doctor of Philosophy

AN INTELLIGENT NATURAL LANGUAGE CONVERSATIONAL SYSTEM FOR
ACADEMIC ADVISING

By

Edward M. Latorre-Navarro

May 2014

Chair: John G. Harris

Major: Electrical and Computer Engineering

Academic advisors assist students in academic, professional, social and personal matters. Successful advising systems increase student retention, improve graduation rates and help students meet educational goals. Research in academic advising shows a trend of integrating electronic technologies that advisors must adopt to increase their roles in student education, while resources are limited and the proportion of students increases. The main objective of this work is to deploy an expert-based natural language academic advising system, to manage multiple straightforward advising tasks and allow advisors to engage in non-mundane educative tasks.

This system features a conversational agent as the user interface, includes modules for data collection, user data management, an academic advising knowledge base and a web design structure for the implementation platform. This system also includes an expert-based system to manage academic tasks such as course sequence planning, and a method to allow the users to contribute to the academic knowledge base.

The system is operational for several hundred students of the Electrical and Computer Engineering Department of the University of Florida. This work covers the design, development, deployment and evaluation of this system. The students used the system during three experimental phases. For the third phase, the system performed well, obtaining close to 80%, on the traditional language processing measures of precision, recall, accuracy and F_1 score. Assessment from the constituencies showed positive and assuring reviews.

The development of this innovative application involves three research contributions to the field. Firstly, a comprehensive literature search and assessment of the academic advising field with recommended solutions based on the latest technological innovations. Secondly, the design of the first-known academic advising multi-task conversational system, using a robust and scalable linguistically inspired algorithm to resolve ambiguous references, suitable for conditions where large corpora of data are not available. The system contains measures for self-evaluation and for upgrading the knowledge base without involvement from the system developers. Thirdly, the implementation and evaluation of the system in a real-world scenario, which shows the viability of the application and initiated the development of a corpus for academic advising, valuable for the language processing and academic research communities.

CHAPTER 1 INTRODUCTION

Academic Advising

Since the late 1800s, higher education institutions have employed academic advisors to assist students in academic, professional, social and personal matters [1]. Academic advising was initially a response for a need to guide students in selecting their course sequence and later evolved to include assistance in personal matters. Later, in the 1970s, the advising practice became an examined activity as education institutions began comparing the methods and results of their advising systems [2]. Today, academic advising still serves the same basic objectives and peer reviewed research is still very active given the many important implications of a successful advising system such as student retention, graduation rates and student educational goals including academic engagement and performance, and career planning [1], [3]. Therefore, it is essential for academic programs to offer students an effective advising experience, which requires advisors to innovate at the speed of their students, who each year tend to have higher expectations from their education institutions and a stronger synergy with digital technology. Thus we see the innovation trend in advising has been towards the use of communication technologies such as email, instant messaging, social networking, course-management systems, podcasts, mobile applications, online videos and blogs [4]–[6]. A quick survey of university websites shows that many have introduced the concept of eAdvising, that is, utilizing electronic means, usually web-based, to offer advising to students [7]–[9].

In 2003, the authors of [10] suggested that interactive advising modules for personalized discussions could be created using artificial intelligence (AI) techniques. In

2008 Leonard [4], detailed the profound effect of technology use by advisors and referenced the idea of an expert-based interactive advising module as a possible future trend, but few institutions have shown interest in developing such a system. A fully automated system is a better solution for the innovation trend in advising and addresses the concern of not enough advisors, for too many students, in an economically challenged environment [1]. Such a system will lessen the burden of academic advisors from several mundane tasks and free up more of their time for the deeper aspects of advising, such as career planning or managing extraordinary personal situations.

There are several research publications describing expert-based advising systems for helping students with straightforward repetitive tasks such as choosing majors, adhering to an appropriate curriculum sequence or accessing degree audits [11]–[15]. This work is inspired by the belief that, following current technological trends, new interactive advising systems should also include a natural language interface to allow students to communicate as freely and openly as with their actual advisors. Such an innovative system could be much more attractive to students as it would allow them to easily ask a wider range of questions than those in previous expert-based systems and obtain immediate responses to these instead of waiting for peers or advisors to read and reply, as with current eAdvising methods.

To the author's knowledge, two publications exist that present advising systems with an expert-based system combined with natural language processing techniques for communication [16], [17]. The first system was limited to only yes/no type questions and a few phrases to manage state transitions [16]. The other system features an ontology-based information retrieval engine to guide students in searching for answers after

entering keywords [17]. The system developed for this research work allows unrestricted natural language communication utilizing state of the art natural language processing techniques.

Natural Language Processing

As the name implies, natural language processing (NLP) is the prime modern software technology for analyzing, understanding and interacting with humans in their natural language. The NLP field includes major research topics such as automatic summarization, information retrieval (IR), machine translation (MT), part-of-speech (POS) tagging, parsing, question answering (QA), sentiment analysis, and word sense disambiguation (WSD). In general, research in NLP comprises knowledge about linguistics, artificial intelligence, machine learning and to some extent, the psycholinguistic and psychological aspects of human language processing. Applications that use NLP involve human-computer interaction (HCI) through either computer natural language generation or computer natural language understanding [18].

Research in NLP is enjoying a peak in popularity after a widespread release of consumer and commercial HCI applications. Many well-recognized HCI-NLP applications are classified as artificial Conversational Agents (CA). In principle, these HCI-NLP applications are currently challenged in their ability to understand the intention of users within a wide range of demographics, regardless of their diversity in syntax, semantics and pragmatics. This challenge is described among fundamental problems in NLP and linguistics such as reference resolution, ellipsis, modifier attachment, conjunction and disjunction, and word-sense disambiguation. In NLP, these problems are within the sub-field of Natural Language Understanding (NLU). In a CA, the NLU is referred to as the unit where the system extracts meaning from the input text.

Natural Language Understanding

Natural Language Understanding (NLU) comprises everything related to machine reading comprehension. In HCI applications, the NLU component is where the natural language input is transformed into a semantic representation, such as first-order predicate calculus and lambda calculus, which allows the system to execute the required tasks. The NLU is where all the domain, discourse and world knowledge combines to determine the requested answer.

Within NLU, a fundamental research topic is resolving ambiguity, that is, disambiguating multiple alternative meanings from a linguistic expression. Resolving ambiguity is usually divided into two specific problems, lexical disambiguation, a difference in meaning, and syntactic disambiguation, a difference in syntactic category as e.g. determining if a word is used as a noun or a verb [18], [19].

Lexical or semantic disambiguation is prevalent in natural language and very much in English. In the WordNet 3.0 database [20], the average noun has 1.23 senses and the average verb 2.16 senses, where in WordNet sample texts the number of senses is roughly proportional to word frequency. Lexical disambiguation (LD) has always been a broad and challenging problem within NLP, cognitive science and the psycholinguistics fields [18]. LD tasks include the well-studied task of POS tagging, choosing the proper tag within context, and the more challenging WSD, selecting the correct sense for a word within context. The term WSD denotes only when the actual sense or meaning of a word is ambiguous, e.g. bass as in fish versus bass as in music. The WSD problem is viewed as a classification task, with research focused on methods known as lexical sample, using pre-selected target words with defined senses, and all-words, where the entire text is disambiguated using a full lexicon. Most proposals now

use unsupervised or semi-supervised learning techniques, after years of mostly supervised training methods using lexical samples [21].

Other LD tasks include:

- Reference resolution – determine the entity to which a linguistic expression is referring. Research for this task is divided between two subtopics: [18]
 - i) Anaphora resolution – finding the entity to which an expression refers. The expression is usually a pronoun and thus the subtask is referred to as pronominal anaphora resolution. Anaphora also includes use of possessive determiners and noun phrases.
 - ii) Coreference resolution – finding referring expressions in a text that refer to the same entity. This is the harder problem of the two.
- Ellipsis – identify from the conversation context an omitted verbal phrase from a sentence [22].
- Modifier attachment – identify the entity to which each modifier has to be attached [23].
- Conjunction and disjunction – when the word *and*, denotes disjunction rather than conjunction [23].
- Named entity recognition – detect and classify proper names.
- Relation detection and classification – detect and classify semantic relations among entities.

Resolving the lexical ambiguity difficulties will contribute to NLP research areas such as IR, MT, QA, sentiment analysis and text classification.

Syntactic disambiguation is the process of determining how to correctly parse a sentence from multiple alternatives, also called the process of resolving structural ambiguity. Structural ambiguity is less common than semantic ambiguity and the two are orthogonal, as a word can have either related or unrelated meanings while in different syntactic categories, for example “The professor will test the test on his teaching assistant” and “Can you pass me that can?” Local parsing algorithms and state

of the art syntactic parsing systems using dynamic programming are effective at dealing with this task [18].

Further complications with ambiguity in NLU may occur from user input of:

- Non-standard language – where the formal rules of capitalization, spelling and usage of characters are not followed. This predominantly occurs in applications where the users are allowed to write the input in natural language, such as text messages, blogs, discussion groups and social media websites.
- Idioms – expressions consisting of sequences of words that have a combinatorial literal meaning, distinct from the figurative meaning of the word combination.
- Neologisms, jargon and slang – newly coined terms, words, or phrases that have not been accepted into formal language. Bound to popularity, sometimes these terms are eventually accepted into the formal language.
- Segmentations – in English, segmentation issues may arise when a user inserts a hyphen between words to enhance meaning. They may also occur when a user invents abbreviations and uses entity names. In any case, most segmentation issues in English can be attributed to the previous mention of non-standard language.

These ambiguity difficulties occur often in HCI applications such as conversational agents, where the human is allowed to freely enter input into the system using natural language. The general concept is that people do not intentionally use ambiguity in every-day language. Psycholinguistic research suggests that when people perceive ambiguous words, they momentarily access their mental lexicons to rule out the unintended meanings [24], [25].

Conversational Agents

A conversational agent (CA) or dialogue system is a computer program designed to carry out an open and coherent conversation with a human [18]. Designing a CA that can fully emulate a human is one the most difficult tasks in NLP, but it's also the most publicly recognized NLP application after years of fictional AI characters made famous through movies, novels and television. In addition to computer science and engineering,

designing a CA requires studies in linguistics and psychology to create a realistic human interaction experience.

Present day computational hardware has allowed this field to explode in a wide variety of commercial applications such as:

1. Personal assistants
2. Help desk agents and customer service representatives
3. Travel agents
4. Technical support
5. Automatic call routing
6. Automatic tutoring systems
7. Gaming characters
8. Virtual friends and companions

Many of these applications are enhanced with additional technology at the input and output of the system, such as speech processing and avatars, and are known as spoken dialogue systems and embodied conversational agents respectively.

Deployment environments of commercial applications include application programming interfaces (API), webpages, robots, in-car systems and telephone-based systems.

Figure 1-1 shows the typical architecture of a basic dialogue system, which comprises of an NLU to receive the system input, a Dialogue Manager (DM) and a task manager to control the conversation, and a natural language generator to produce the output. The DM controls the architecture and structure of the dialogue, i.e., determines the output of the system given the input. It also keeps record of the state of the system and serves as interface with the task manager. The DM, NLU and natural language generation (NLG) components manage all the system's knowledge about the input language and knowledge about the domain of the application. The task manager represents the link between the DM, the domain knowledge base, and any other domain-specific components of the dialogue system.

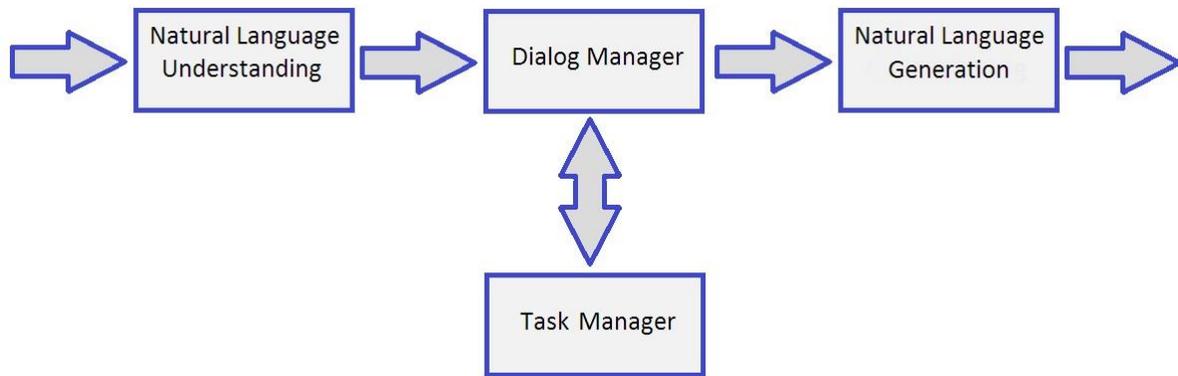


Figure 1-1. Basic architecture of a dialogue system

The two well-studied methods used in NLU systems are identified as pattern matching and statistical learning. Most practical applications utilize corpus-based methods while most research developments are on machine learning techniques, thanks in part to the availability of large corpora of data. Chapter 2 explains the motivation and requirements for choosing the method for the advising system.

Dialogue management is a decision task, typically modeled in state-space and implemented using knowledge-based or machine-learning techniques. Two common advanced architectures for dialogue management are information-state and plan-based [18], [26]. The term information-state architecture covers a wide range of knowledge-based and statistical methods that extract features from discourse context to determine the output. Plan-based methods use logic calculus to analyze within the knowledge base, input extracted features known as beliefs, desires and intentions. While distinct in principle, the two architectures require careful design for application constraints, maximizing feature extraction from the context of the input and many cycles of training and testing.

The NLG component hosts the process through which the output language is composed. Two well-known basic methods to generate language are template-based and generative-based [18]. For the first method, the simplest and thus most common, the dialogue designer preassembles sentences or prompts. The prompts usually contain parameters that the dialogue manager defines. The term generative-based refers to a process where an output sentence is constructed from words or even morphemes. Research in this area is focused on knowledge-based and statistical approaches that use the Internet and other large corpora as sources [27].

The evaluation of dialogue systems is an open-research problem, with most proposals requiring carefully weighted evaluation metrics and taxonomies [18], [28]–[30]. Nonetheless, dialogue systems are ultimately evaluated on the satisfaction and efficiency costs for the users. For this reason, the Turing test is still a highly regarded evaluation for any CA [31]. The Loebner Prize is the first and most renowned instantiation of a Turing Test [32]. Though not a standard within the academic research field, the open competition usually takes place at well-recognized universities and with participation by members of the academic and research field.

The CAs that compete in the Loebner prize are commonly known as chatterbots and are full production systems viable for commercial applications. Two recent multiple-year winners have released the main dialogue systems as the open-source languages Artificial Intelligence Markup Language (AIML) [33] and ChatScript [34]. Both systems utilize a template-based approach for the NLU and NLG, and a knowledge-based approach for the DM.

Released in 2001, AIML enjoys great success in the chatterbox community with thousands of members contributing to the knowledge base, releasing interpreters for multiple programming languages, and developing commercial releases and supplementary packages with advanced language features [35]–[37]. The advising system described previously, utilized AIML for its CA implementation [16].

ChatScript is an open-source scripting language for a rule-based dialogue system. ChatScript was released in 2011 with significant features not available in AIML, e.g., a knowledge-based information-state architecture for the NLU and NLG, WordNet integration, a query-based system for a dynamic database of tables and triples data structures, and analysis tools for user log files.

Another well-known development platform for NLP applications is the Natural Language Toolkit (NLTK), an open-source NLP platform written for the Python programming language [38]. Released in 2002, NLTK features WordNet integration and a strong NLU component. Whereas NLTK is used mostly for educational and research purposes, a few recent publications have used NLTK when developing dialogue systems [39], [40].

Question Answering Systems

When instead of a conversation, the system only responds to single queries independently, it is known as a QA system. Given strong commercial demand, QA systems are one of the oldest but also most active research topics in NLP. Because of the nature of many CA, these dialogue systems also contain a QA component.

The two basic QA models are IR, where answers are obtained through a literal search, usually from the web, and knowledge-based, where answers are built after a successful analysis of the parsed input. In knowledge-based systems, the information

for building the answers is collected from multiple documents. QA systems are usually designed to manage specific types of questions and answers, e.g., factoid questions and complex narrative questions. Most QA commercial applications deal with factoid questions, while narrative questions are common in research systems [18]. A system that uses a CA with a strong QA component must also manage elliptical questions [41]. Therefore, the advising CA must manage all three types of questions, given that its main task is to serve as a QA application.

Dissertation Topics

The main objective of this work is to construct and deploy a real-world expert-based academic advising system that allows the students to communicate freely in natural language. The system was designed to serve students of the Department of Electrical and Computer Engineering in the University of Florida (ECE-UF). The system is accessible online for public use at any time. After completion, this system can serve as a basis for deployment in other academic programs.

This system allows for training and testing of the algorithms developed in a well-defined, domain-specific, conversational question answering application, similar to other well-known commercial systems such as SIRI with Apple, Watson with IBM and Wolfram-Alpha. This work describes the production of a CA for academic advising, with the research focused on the NLU unit to develop a linguistically oriented algorithm suitable for this application, where there are no available corpora for machine training. The CA designed utilizes state of the art NLP algorithms and methods to enhance the system for utmost effectiveness and usability as evaluated by its users. The robustness of the algorithm allows for scaling up the input range of the system as the amount and

diversity of user interaction grows. Therefore, this work also introduces a methodology for allowing the system to scale up as new data and queries are obtained.

This dissertation is organized into the following chapters. Chapter 2 describes the task of academic advising, the user base for the system, introduces the conversational system for academic advising and specifies the components of the advising system. Chapter 3 describes the dialogue manager of the system, including the NLU, NLG and task managing components. Chapter 4 describes the academic planning system and the components for communicating information between the students and the advisors. Chapter 5 describes the field tests and the analysis of the results. Finally, Chapter 6 contains the conclusion, the contributions to the field and the plan of work for the future of the system.

The appendix section includes a list of topics this application contains, the screenshots of the web interface, performance evaluations by the ECE-UF advising personnel and examples of user log files from the experiments.

CHAPTER 2 A CONVERSATIONAL SYSTEM FOR ACADEMIC ADVISING

This chapter begins with a brief narrative of academic advising tasks, followed by a description of the advising situation in the academic department where the system is installed, in view of the motivation for the academic advising system developed. The second section details the objectives and specifications of the framework of this system. The third section introduces and describes the main components of this system.

Academic Advising

The Role of the Academic Advisor

As described in Chapter 1, advising is an integral part of a successful academic department of an educational institution. Academic advising seeks to encourage student success through teaching and learning that leads to productive career and life planning skills [1]. The tasks advisors engage in are traditionally identified as prescriptive and developmental advising. As described by Appleby, the prescriptive advisor dispenses expert advice, whereas the developmental advisor engages in a mutual learning process with the student in order help the student's problem solving, decision-making and evaluations skills [42]. A study by Brown shows how the role of the advisor transitions from prescriptive to developmental throughout the academic years of the student [43]. The study also describes how advisors should be proactive with entry-level students.

Other studies support the idea of educative advising, where advisors are the teachers of the philosophy of the curriculum and the principles of how students learn [44]. As teachers, advisors need tools to encourage students to seek the information they need, so they can consult their findings and goals with them. An increase in

developmental and educative tasks will require an increase in resources for advising. The system developed for this dissertation provides advisors with an effective tool to streamline many prescriptive and educative tasks, thus allowing further developmental and educative tasks during their limited face-to-face time. With academic offices repeatedly struggling for resources, this automatic advising system is a favorable solution for the students, the advisors and the academic institutions [1].

Academic Advising in ECE-UF

The advising system was developed for students of the Department of Electrical and Computer Engineering of the University of Florida (ECE-UF). Undergraduate students of ECE-UF visit their academic advisor either by request of the advisor or to freely discuss a topic relevant to their studies. The first case is the most common, as these meetings must occur at least once per academic semester during a student's enrollment in the study program. During these meetings, most topics involve queries with accessible answers, i.e., prescriptive advising. These answers are usually obtained directly from facts or through logic analysis, and thus may be solved algorithmically through search algorithms and logic calculus respectively.

The other reason why students contact advisors is for the personal and professional issues that advisors must answer within their best judgment, i.e., developmental advising. The challenge for an automatic system lies in that any possible answer is greatly influenced by the personal circumstances of the student and very sensible to the context of the given situation. Since many of these questions arguably lack a definitive answer, it is reasonable to state that an advising system either will be biased when referring to such queries or must state that an answer is not practical for the system. The system either must anticipate every possible scenario or must

incorporate a decision-based generative method to create the best possible judgment for the scenario. Clearly, developmental tasks are better left for the human advisors. Nevertheless, while the main objective for this system does not include developmental advising, the system does manage a certain amount of the related topics.

Advisors and students of ECE-UF do not have a centralized database of academic advising information. This challenge affects training for the advisors, alternative resources of knowledge for the students and bookkeeping of all rules and guidelines. This is also a shortcoming in the effort to create a comprehensive knowledge base (KB) for the system; therefore, the knowledge for the system was obtained through meetings with the academic advisors and surveying multiple ECE-UF and UF documents. Results from this work will provide ECE-UF with a digital and easy to search through, advising KB. The data from the system will help construct an advising data corpus, currently unavailable in the research field.

Following Chapter 1 and the conditions just described, the design choice for the advising system was a template-based NLG and a knowledge-based QA model.

Fundamental Requirements for the Advising System

From a teaching perspective, some advising tasks of this system are similar to features available in conversational agent (CA) systems for e-learning environments, such as those examined by Kerley *et al.*, Gandhe *et al.* and Mori *et al.* [45]–[47]. The evolution of these CA applications shows an example of initially using pattern matching techniques and then statistical methods, as sufficient data becomes available. The success of many of these systems in education suggests the viability of the advising system here proposed. These systems also encourage additional non-NLP features for the advising system such as, an escalation mechanism to forward selected

conversations to advisors and adding a limited amount of non-topic conversations, or small-talk, to inspire user rapport with the CA. In contrast, e-learning systems have fundamental differences from the advising system here proposed, for instance, e-learning systems control the conversation with the users to allow for effective teaching and evaluation of the user's performance, while in the advising system the user controls the conversation at all times. Kerley *et al.* presents a series of questions that developers of e-learning systems must face [45]. To resolve the fundamental requirements for the development of the conversational advising system, these questions are grouped into the following nine major prerequisites.

What approach to NLU must the system adopt? The enclosed domain of the main topics for this application and the lack of any available corpus favor an approach to pattern matching techniques.

How does the developer select the software to implement the NLU techniques? A solution for rapid deployment involves selecting the most advanced available software within the free and open-source alternatives and building upon it. All components of the system are also built using open-source programming software.

Who controls the direction of the interaction, user or conversational agent? This design is meant to allow students to address their advising needs, therefore all the conversations are initiated and controlled by the users.

What freedoms do users have through the interface? User input determines the states of the CA. The system must recognize the situations that call for state transitions. For example, the user will access and manage the course planning system upon request, at any moment of a conversation. To make the most out of the

computational resources, the CA must have access to the computer's OS to interact with additional software and communication platforms.

Does the system structure the path along which the user must proceed?

For the advising application, traditional dialogue managing methods such as dialogue trees do not work because the intention is not to engage and expand a conversation, nor to participate in small talk with the user, but to respond to the user's questions with the information solicited and stay or leave the topic only upon user request. This requires very good reference resolution to easily determine when a linguistic expression is related to the same topic or starting anew. Therefore, the system must allow the user to explore the conversation along any dimension they choose, including some off-topic dialog, usage of follow-up questions and contextual references. For off-topic dialog, the system must quickly encourage returning to relevant tasks. As a service agent, the system must understand and help the users as efficiently and effectively as possible, that is, engage in the minimum possible dialog to understand the users and offer a straight-to-the-point easy to understand response. The system output must be succinct, while preventing unnecessary follow-up questions. The system shall not initiate off-topic dialog or return answers that could potentially misinform the user.

What will users say? Since there is no database of academic advising, the data was obtained through researching UF documents, advising documents from the web and through consultations with ECE-UF advisors. Allowing students to communicate naturally is akin to how users behave in the online and mobile communication media that are pervasive in our society. This behavior includes informal conversational natural language intricacies such as implicitly conveyed information, semantics, pragmatics,

ambiguity, contextual references and neologisms, which by social expectations, users presume NLP applications will understand. The system must also manage the input data without formal writing features such as punctuation and capitalization. It must also contain a spell checking method for the unique vocabulary and non-standard language of the user field, as well as the technical terms unique to each academic program, e.g. COMP1234 (course-code), C+ (grade), C++ (either an unofficial course name or a programming language) and gened (general education course). Many of these unique terminologies must be collected from the user field, thus data collection from the users is crucial for the success of this system.

Many CA developers initiate the data collection process using the Wizard-of-Oz technique, where a human, or wizard, simulates the behavior of the final system without the user knowing it is in fact a human at the computer's side [18]. For this work, time and user availability hinder a chance for such a study. To maximize input data and productivity, the best option is developing and deploying a preliminary version of the system and gathering the data through the promotion of this *beta* system.

How does the system handle unexpected responses elegantly? For unrecognized input, the system can suggest possible recognitions based on partial matches, request rephrasing or explicitly state that a response is not available. Another approach involves asking users to define the unrecognizable concepts and through semantic relations, map these definitions to the active KB. These alternatives are analyzed further in Chapter 5, where we examine a methodology to scale-up the CA.

How does the system handle synonymous expressions? In traditional pattern matching techniques, developers address this challenge by predefining a

mapping of synonymous expressions. For everyday English, the lexical database WordNet is a well-known resource [20]. The advising CA system employs a software platform that includes WordNet along with many additional predefined concepts, which allows easily dealing with this concern. Conversely, the definitions for the technical terminologies the users will use must be hand-coded.

How much testing will be required? Commercial CA systems in e-learning often undergo over two thousand conversations before reaching operational state [45]. With an additional data collection phase, the advising system should require a greater amount of conversations to obtain a comparable performance.

Three additional fundamental requirements for this system include, first, that the system be available to the students at all times through a comfortable instrument such as a webpage or mobile application. Secondly, the dialogue 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. Third, the system must obtain and store student academic data without compromising their private information. Specifically, in UF, student academic records are protected by the Family Educational Rights and Privacy Act – FERPA 20 U.S.C. § 1232g; 34 CFR Part 99, a U.S. federal law, and by the Florida Statute Section 1002.22 2007, a state law.

Albert – The Intelligent Academic Advising Conversational System

The system developed provides students with an academic advising service that reflects a human interaction experience through an online text application. The system does not require student training or additional human resources from the academic departments. This system enhances the academic advising experience by offering students a service that is available at any time. Albert includes multiple advising

services accessible from any device with web access. Albert respects the privacy of its users and encourages the students to become independent and take responsibility for making decisions. Students who effectively utilize Albert will have the awareness to focus their meetings with the advisors in the subjects of developmental and educative advising.

Advising Information

Albert includes knowledge about the academic programs and policies, answers to a wide range of frequently asked questions (FAQ) in academic advising, it offers recommendations for the development of a course plan that leads to degree completion and referrals to other academic services. Appendix A contains the list of academic topics included with Albert. Students can search all this available information using natural language for a comparable experience to communicating with their advisors.

Albert contains the course scheduling information for all ECE-UF courses, as posted on the UF Registrar's webpage. Once available, a Python script reads the online information and a cron job updates the KB daily. The information is stored indefinitely and always available for access when queried by date. This process assures the information is always up to date without dependency of human maintenance. For energy consumption reasons, updates for the information of each term are suspended two weeks after each term commences. The easy access to this information benefits all students from ECE-UF and other departments who want to enroll in courses from the department, faculty members, advisors and administrative personnel. All the other information in Albert is hand scripted in the KB of the system.

As a courteous advisor employed by an academic department, the system output reflects an advisor who is polite, friendly, maintains a positive affective state with its

users and demonstrates a personality trait intended to sympathize with the students of ECE-UF.

For this development of Albert, knowledge about degree programs and advising FAQ is mostly relevant to undergraduate students. While these students are the targets of all the advising challenges and benefits discussed previously, future developments could include information about graduate degrees. Albert does contain all ECE-UF undergraduate and graduate course information including the course schedules, pre-requisites, descriptions, etc. The target users are students from the ECE-UF two undergraduate degrees, Bachelor of Science in Electrical Engineering (BSEE) and Bachelor of Science in Computer Engineering – Hardware emphasis (BSCEE). The target users also include the undergraduates who are enrolled in the combined Bachelor and Master of Science program, where students get an early start on their Master's degree.

System Components

The main script of Albert is written in the Python programming language, version 2.7.5. This script controls all the functions of Albert and communication with each module including the web interface, the gateway interface for web communications, the user login routine, the dialogue manager, the expert system for academic planning and the database. Figure 2-1 shows a model of the Albert system. The dialogue manager refers to the complete dialogue system described in Chapter 1, namely, the NLU, NLG, DM and task manager.

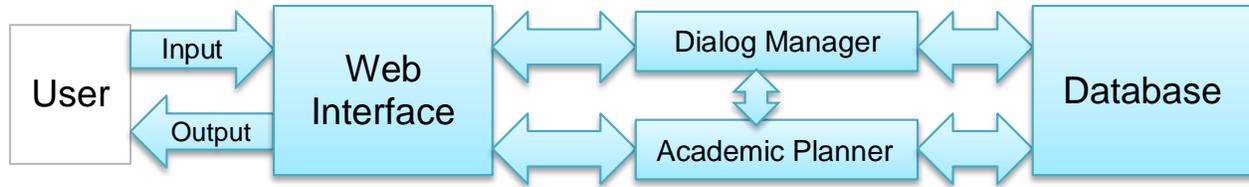


Figure 2-1. A model of the main components of Albert.

For students to effectively adopt the system, continual availability of Albert is required. This feature is accomplished by making Albert accessible through the web, specifically as a web-browser graphical user interface. This website is hosted in a desktop computer at ECE-UF facilities and accessible via the Internet address <http://advising.ece.ufl.edu>. The computer has an Intel Pentium 4 processor, 1.8 GB of RAM and is running the operating system (OS) Red Hat Enterprise Linux 6.4.

The website was built with scripts written in Python, PHP, JavaScript, HTML and CSS programming languages, in addition to Unix scripts to manage the daily tasks. Communication between these programs, the computer's OS and the web-based clients, is done through socket technology, which is widely used in web-based software. The website was designed for simplicity and speed, with a load time of approximately one second on contemporary versions of the web browsers Google Chrome, Mozilla Firefox and Apple Safari.

Users who access Albert using the Google Chrome web browser can also send messages using speech recognition software. The website includes an open-source speech recognition API that allows users with the required hardware and Chrome to input data using speech [48].

Figure 2-2 shows a screenshot of Albert in its initial state. The mid-upper left area shows the forms for logging into the system. Below these forms, the main window for communicating with Albert shows the instructions for connecting with the system. On the right of the main window, an independent web frame shows examples of FAQs the system contains. Screenshots showing additional states of the website are available in Appendix B, including a screenshot of the webpage using Chrome with the speech API.

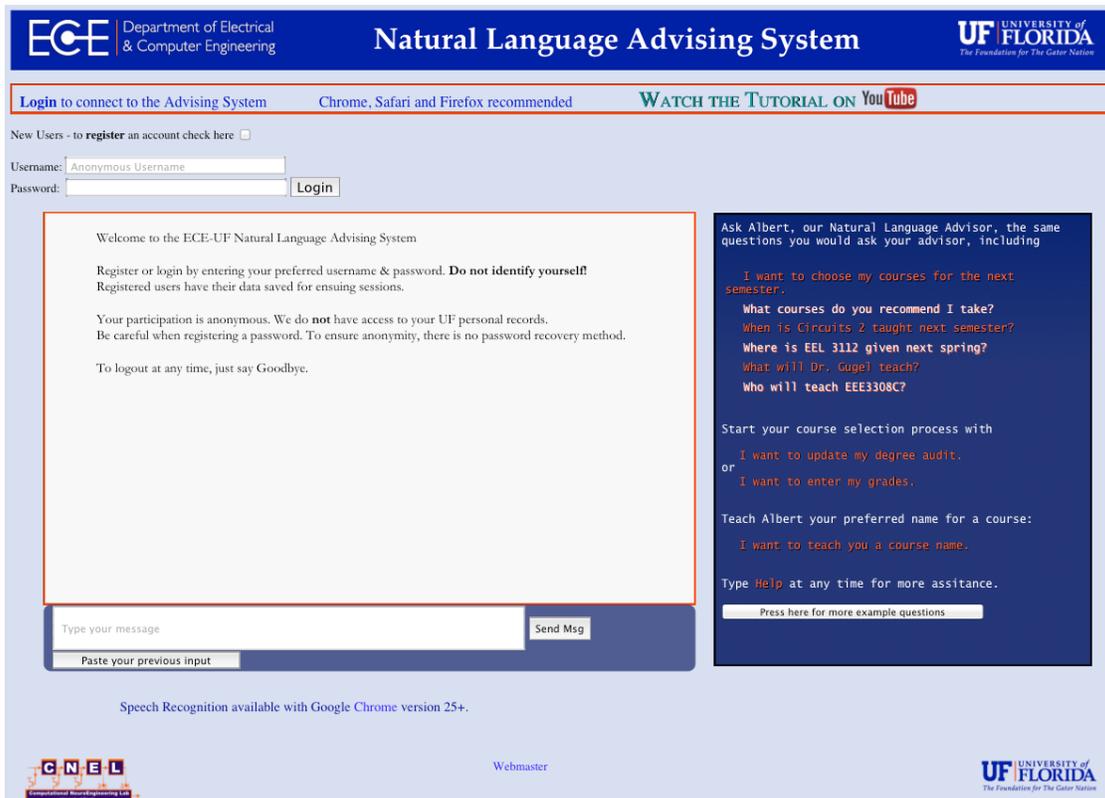


Figure 2-2. A screenshot of the website for Albert in its initial state.

To protect the information of the students, the system requires users to register an account using an anonymous username and password combination. This data is stored encrypted in the Albert web server using the standard secured hash algorithm

SHA-512 – 64 bits. Usernames and passwords are case sensitive; furthermore, the system does not allow repeated usernames entered in different letter case. These security features allow students to safely share their usernames with their advisors, yet keep their information secured through the password. Anonymous accounts also encourage students to communicate freely without repercussions, which is the best way to obtain sincere feedback about the system and the academic services. The account in Albert also allows storage of user information for recurrent advising sessions.

Advising Dialogue Manager

The heart of Albert is the NLP system that drives all the input, output and states of the system, i.e., the advising Dialogue Manager (DM). Following the objectives and constraints listed above and the options reviewed in Chapter 1, the best option to develop the DM is to build it around the ChatScript (CS) scripting language. This work uses CS version 2.0. In the DM, CS is the main component and it includes most of the NLU and NLG structures. The design of the DM allows a straightforward method for redrafting and distributing the system for other academic programs, by using variables in the input patterns for the proper nouns and the contents of the well-defined knowledge base. Specifically, the advising FAQ corpus for the input patterns is generic enough such that deploying in other academic departments requires editing the KB, but not reworking the existing input templates.

Since CS runs as a client of the DM, the DM controls communication of CS with additional task applications integrated with Albert. CS can also communicate directly with the OS through terminal commands at any moment. CS can read files from external applications via its internal database, subject to these applications creating the files in the required format. With these features, the DM integrates functions written in

CS, Python and PHP. For example, to manage the unique technical terms and proper nouns, an external spell-checker in Python was constructed using methods surveyed from the literature [18], [38], [49]. When CS finds a candidate for the spell-checker, e.g. a noun term similar to a course name, it sends the term to the Python spell checker, which returns the possible corrections to CS to determine the system response.

Chapter 3 covers the details about the DM and processing the natural language input and output using the CS system. Regarding computational performance, as CS is a production grade chat system, messages through the website took approximately one second when tested from computers connected through a local area network.

Academic Planning System

Albert includes an expert system designed to offer students guidance and recommendations when preparing their course plans for each term. This expert system was developed in Python. This system allows students to enter their academic record and receive recommendations on how to develop their course plan, based on the courses completed, the prerequisites approved, the courses offered each semester and related department and university rules. With this academic planning system, students can create their course plan for the next semester and send it electronically to their advisor for review. With course planning as the main feature, we refer to this system as the Course Plan Analyzer (CoPA).

CoPA runs independently from the DM allowing users to work on both systems simultaneously, analogous to a student filling up a form in the advisor's office. Albert controls communication between both systems and automatically sends the data to an advisor's database, allowing advisors access to student submissions. Development of this procedure required building an independent website for advisors to access this

database. This process is part of the objective of allowing advisors to integrate Albert into their daily advising practice.

Chapter 4 covers the details about the academic planning system and CoPA. Regarding computational performance, the webpages for the CoPA system are also within the approximately one second delay between events the rest of Albert offers, when tested from computers connected through a local area network.

System Evaluation

To evaluate Albert, following the academic advising guidelines of the National Academic Advising Association [6] and the Council for the Advancement of Standards in Higher Education [50], assessment must include direct and indirect evaluation, use qualitative and quantitative methodologies, and data collected must include responses from students and other constituencies. For quantitative analysis, Albert includes a numerical survey for the students built-in the CoPA process and through CS, keeps a record of log files from all users. The log files contain the information of the input-output messages, login events and message timestamps. Within the CS dialogue scripts, an original development allows to assess the number of questions that did not have an available answer, the questions responded to with nonspecific answers and other statistics that provide a measure for accuracy and task completion success. Chapter 3 covers additional details about the log files and their usage, and Chapter 5 shows the results obtained from these log files. For qualitative analysis, the measures include collecting feedback from the students and having consultations with the ECE-UF academic advisors for assessment reports.

The data from the log files is also used to compose the academic advising FAQ corpus currently under development. As the data is processed, the system continues to

grow with additional input patterns. This data also offers assessment to advisors with information and statistics about the user data, to learn the areas of advising that most concern students. As reviewed by Soria in [3], advisors need detailed assessments of the elements of satisfaction that matter most for students' integration and retention, and student satisfaction with their academic institution.

CHAPTER 3 THE ADVISING DIALOGUE MANAGER

Chapter 3 describes the advising dialogue manager (DM), namely, the Natural Language Understanding (NLU) system, the Natural Language Generation (NLG) system and the task manager.

Natural Language Processing

The NLU component of the DM produces a semantic representation of the natural language input that is appropriate for the dialogue tasks. The NLG consists of templates containing text, pointers, variables and other control functions. The main objective in the DM design was to respond to questions from academic advising topics. The NLU contains several templates and measures for responding to secondary topics such as information about the Albert system, control of the system, the persona of the Albert advisor, conversational systems, web browsers, mobile devices, time, dates, movies and other small talk. The structures in the NLU for these secondary topics use the same principles as the main topic, and the design calls for off-topic dialog to promptly converge to on-topic dialog. Accordingly, the focus of this section refers to development of the NLU for the advising topic only.

The advising knowledge base (KB) in Albert contains a finite amount of answers that the system should return in a conversation. In contrast, the system input domain comprises infinitely possible styles in which users can request each answer. To create the KB, the output information was obtained through university documents and interviews with the academic advisors. The input templates were derived from a Standard English subset of the input domain. As designed, after deploying the system, the KB was continuously updated through analyzing the user log files. The KB has the

information organized as a list of triples containing a question, its answer and the topic to which the answer was classified.

Natural Language Generation

As a template-based system, the NLG mostly responds with strings of answers taken straight from the KB. The answers are constructed using as tools logic functions, random generators, control structures and queries. Queries include course-scheduling information from the UF Registrar's Office webpage and student academic information saved with the course plan analyzer (CoPA).

The system also contains a trivial amount grammar rules, mostly for handling unrecognized input and extending off-topic dialog. For example, when responding to unrecognized questions, the NLG may either change the verb tense or rearrange the POS to let the user know the answer to that question is not available.

Natural Language Understanding

The system input is an informal conversational natural language with intricacies such as implicitly conveyed information, semantics, pragmatics, ambiguity and contextual references. Therefore, the system requires an effective reference resolution process to determine the intention of the user from the linguistic expression and its context. The NLU must classify every significant lexical unit (LU) and their relationships within the context of the conversation. A practical solution is to build an effective keyword extraction (KE) mechanism with context-based ambiguity management. Most current KE systems use similarity measures with features such as term frequency, inverse document frequency, relative position of the first occurrence and caps first ratio [51]–[54]. To be effective, these measures need either sufficient training data or data with formal grammar, none of which are available for Albert. Moreover, even with

sufficient data, most input queries for this system are too short for most frequency analysis techniques. Nevertheless, the work of Hulth and Bellotti et al., shows 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 [45], [51].

To successfully classify LUs, the NLU must resolve the ambiguity problems identified in Chapter 1 namely, word disambiguation, reference resolution, modifier attachment, named entity recognition, relation detection and classification, the use of non-standard English, and elliptical questions. NLP applications usually address ambiguity by adding probabilities to the grammar; however, this requires more data than is available. In addition, data from the initial experiments showed that students expect the system to understand their messages written with informal grammar, incorrect sentence structure, e.g. fragments similar to computer commands, and without the formal writing features of punctuation and capitalization. The overall initial reaction of users was to request information akin to using a typical online search engine, i.e., entering isolated keywords. Appendix C shows user data from the latest log files.

NLP Algorithm

The NLU and NLG components are built with the ChatScript (CS) scripting language. CS resembles a knowledge-based information-state architecture, features WordNet integration, a query-based system with a dynamic database of tables and triples data structures, and a client-server architecture that communicates using sockets. CS was created on the basis that understanding natural language requires pattern matching and a method to infer from facts. Similar to semantic grammars, a script in CS comprises topics, templates, concepts, facts and logic functions.

A topic is a collection of templates for input-output matching. Each template contains the keywords, logic functions and control structures for input matching, and the corresponding output message. A concept is defined as any collection of LUs such as a sentence, phrase, word or parts of words. CS includes predefined concepts with WordNet *synsets*, which are sets of cognitive synonyms. Facts in CS are the elements of the query-based database, with field values written as triples containing LUs or other facts. Facts are useful to build records, arbitrary graphs, data arrays, tables and similar database components.

The input template design involved identifying the keywords, POS tags, noun-phrase chunks and lexical relations of each input statement and selecting the most significant features, to define the keywords of the template and any respective topic keywords. For example, the topic *course_schedule_information* includes as keywords, the list of all course names, the list of all professor names, the word “*professor*” and the word “*teach*”. The keywords for each input template, the words that convey the message, are extracted using the KE features referred to previously. For example, in the input “*Who will teach C++ during the next semester*”, the keywords are *who*, *teach*, *C++* and *next-semester*, where *who* and *teach* must appear in that order. Using CS concepts and facts, the keywords are generalized to “*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 a word or phrase that refers to an academic term, e.g. next semester, next term, Fall 2012 and summer.

Figure 3-1 shows the algorithm for the example above. Clearly, this example is not the only way to ask who will teach a specified course. For example, the user could ask “*Who is the professor of C++*”, or if the request is within the context of the previous input, the user could ask “*Who is teaching it*”. The first case requires building a new template, which can be mapped to the output to the template defined above. The second case requires adding a new template that includes a method to determine the context of the previous input. The system identifies the context using features such as the current and previous input keywords, the keywords for the topics matched, the tense of any verb and the state of the variables representing potentially missing keywords.

| |
|--|
| <p>Algorithm: Respond to user request – Who teaches course</p> <p>Input string S: <i>Who teaches C++ next semester?</i></p> <p>Desired output: <i>C++ is taught next semester by Name of whom teaches C++ the next semester</i> or <i>C++ is not offered next semester</i></p> <p>If S contains a keyword of the topic <i>course_schedule_information</i></p> <p>If S matches with a Who-Teaches pattern</p> <p> If S contains a Term-Phrase keyword</p> <p> Calculate the Term value</p> <p> Else</p> <p> Use currently stored Term value</p> <p> If the course C++ exists in the schedule of the Term</p> <p> Find the corresponding data element <i>Instructor</i></p> <p> Return <i>C++ is taught by Instructor in Term</i></p> <p> Else</p> <p> Return <i>C++ is not offered in Term</i></p> |
|--|

Figure 3-1. Example of a procedure to match an input requesting who teaches a specified course during the next semester.

Recognizing context is also a requirement for resolving lexical ambiguity. Resolving the ambiguity problems required a combination of solutions. For reference resolution and elliptical questions, in addition to the method described above for

recognizing context, the system uses the CS feature of rejoinders. Rejoinders are input templates, which follow parent templates that elicit some expected user response. Rejoinders also allowed some input templates at the end of topics to assume certain keywords were implied. The entity recognition problem is managed by defining case-insensitive CS concepts with pre-classified POS tags. In addition, the system has concepts defined for all the technical terms, neologisms, slang and significant LUs not available in the CS or WordNet dictionaries. The system processes modifier attachment, semantic relations and entries in nonstandard English, on a per-template basis using logic functions and control structures.

The system also has measures to deal with nonlinguistic ambiguity that the users inadvertently convey. In some cases, the best solution was to return all the probable answers, like in the example, “*Who is teaching EEL4995?*” where EEL4995 is a collection of courses with individual instructors. In other cases, the best solution was to request more information from the user. Nonetheless, in most cases it is still preferable to convey some relevant information, and allow the user to either read it or try a different option. This user preference was palpable during the field tests; see Chapter 5. For instance, when a user asks, “*Who teaches Circuits in the next term?*” the meaning of next term depends on the current date and during the spring semester, it could refer to either the summer or the fall term. For this case, the system will decide on a value for the *term* variable, determine the response and return the response including a quick method to obtain the value for the other *term* value. As evident in the algorithm of figure 3-1, once the user states an academic term, the system stores this value in case the user does not provide it in a future request.

When writing the input templates, the key tradeoff is between over-fitting and not generalizing well, thus increasing missed inputs, or under-fitting and causing false positives. In this work, the precision of the template is inversely proportional to the rate of occurrence of the template. That is, the responses that users most seek have a lower accuracy and higher coverage. In contrast, the precision of the template is proportional to the intricacy of the response, i.e., very specific answers have templates with higher accuracy. Within each topic, the templates are organized from complex to general, similar to an inverted decision tree. This approach minimizes false positives for questions with many specifics, a design constraint to prevent misleading the students, while causing most false positives errors in responses with broad information.

When an input statement does not match with any template, the system will respond with an estimated match or request a new entry. The last three topics of the NLU script respond to these statements.

The antepenultimate topic contains several on-topic templates with either, partial answers because the full statement was not recognized or with suggestions to help the user obtain the anticipated response. This process is alike typical communications where if the receiver is doubtful of the input, it notifies the sender of the issue or requests a confirmation of the received message. This topic also contains several templates for pronoun resolution, where the system will either redirect the user to a specific topic for the response or alert the user that the statement was not recognized as written, to encourage the user to not repeat such statements. In general, this topic is designed to respond to any input statement that contains a significant keyword, such as a topic keyword, and return the most likely match based on a score of the significant

keywords. This topic also contains templates that call external spell checking procedures, which details are covered in the next section.

The penultimate topic contains a small collection of less significant keywords, such as verbs and common nouns, and some off-topic templates. Most of the templates in this topic are produced after analyzing the user log files and finding various users who did not understand how to use the system. For example, many users would expect the system to have knowledge about popular culture or to successfully respond to one-word statements such as *course*, *enroll* and *hold*. Ideally, this topic should not respond to users, however, it proved useful in the preliminary tests.

The last topic responds to statements where the system cannot immediately help the user, either because it does not know to which topic to redirect the statement or because nothing was recognized. The templates are mostly for catching single keywords and responding with answers that explain how to obtain information related to the keywords. Some templates contain POS tags to generate responses to *wh*-questions with a proper grammar. The topic ends with a collection of responses, which acknowledge that no part of the input was recognized. For evaluation purposes, the system classifies responses generated from these last two topics as indeterminate.

While these features enhance the template creating process, all template-based systems are limited by the amount of patterns for input matching. The next section describes a method to involve the users and experts in scaling-up the advising KB. Chapter 5 includes the numerical specifications of the DM scripts and the results of the experiments.

Task Manager

The task manager represents all the functions Albert executes to complement the dialogue system. These functions include user account management, database management, the course plan analyzer, input validation, spell checking, statistical data collection, automatic updates, a scaling-up routine and all communication for CS. The task manager is built with Python. This section closes the chapter with the description of the major functions, whereas the next chapter covers the course plan analyzer.

Spell-Checking

Albert is a text-based system, thus spelling and typographical errors are very common, yet users expect the system to handle them effectively. CS includes a list of common spelling errors and the correct spelling that the system substitutes automatically on inputs. This list, plus a corresponding list for the technical and local terms common to the Albert users, was continuously updated throughout the experimental phases. CS also has a built-in spell checking method, although during the period of this work it was still a work-in-progress. The CS spell checking methodology was improved for this work by adding the capability to handle the required technical terms and unique expressions.

The CS spell checking procedure was also enhanced such to allow the system to return the results of the spell checking process, to alert the users that a correction was made and encourage them to verify the new input. While spell checking in general corrects typographical errors, sometimes the correction can change the meaning of the question and cause the system to return an unexpected response. Since users are mostly expecting the answer to their problem, it is very important to let the user know if the response is the answer to the question they asked. This procedure is akin to the

objective described for the non-linguistic ambiguity resolution, where when in doubt, always present the user with a solution and allow them to choose a different option if the solution was not correct.

As mentioned in Chapter 2, to manage the unique technical terms and proper nouns, a supplemental spell checker was constructed in Python using methods surveyed from the literature [18], [38], [49], including the measures:

- Deletion – item contains characters in excess
- Transposition – item contains characters in misplaced locations
- Alteration – item contains incorrect characters
- Insertion – item is missing characters

As described in the previous section, the antepenultimate topic in CS contains the templates that call the external spell-checking procedure. These templates are very similar to the counterpart templates meant to respond to the input, except for a placeholder where a missing term is expected, e.g. a noun term similar to a course name. The term in the location of the placeholder is evaluated to determine its lexical distance to the missing term. If the lexical distance is within a predetermined value, this candidate term is sent to the Python spell checker.

This spell-checking routine uses a similar lexical distance measure to determine the likelihood of the possible corrections. If a correction candidate has a significant likelihood versus the other candidates, this value is replaced in the original input statement and this new statement is sent to CS for evaluation. If there are candidates, but none of them is relatively significant, the system returns all the candidates to the user and prompts the user to choose a selection or rephrase the statement. If no candidates are found, the system terminates execution and responds to the user that statement was not recognized.

Figure 3-2 summarizes this algorithm with an example similar to that in figure 3-1, where the incorrect entry is the name of the course from which the user is requesting information. In this example the user is requesting, “*Who teaches EEL3101*”, however that course does not exist. The spell-checking algorithm will return to the user “*EEL3101 is not an EE course I recognize. If I assume you meant EEL3105, then my answer is: For Spring 2014 we have EEL 3105 by Latchman, Haniph A*”.

| |
|--|
| <p>Algorithm: Respond to a user request with the spelling error – Who teaches a course</p> <p>Input string S: <i>Who teaches EEL3101?</i></p> <p>Desired output: <i>EEL3101 is taught by Name-of-lecturer.</i> As EEL3101 does not exist, the desired output is to infer the intention of the user and respond with the expected response.</p> <p>If S matches with the Who-Teaches pattern with a missing term Determine if S contains a term X similar to any missing term candidate. If X is within a predefined distance Send X to the Python spell checking process Else End execution of this template. CS continues iteration input matching.</p> |
| <p>Algorithm: Python spell checking process</p> <p>Input string X: EEL3101, an incorrect value for a course name</p> <p>Desired output: Most likely course name</p> <p>Find all terms Y similar to X, within a lexical distance of 2. Assign a score K to each term. If a value Y_K is statistically significant versus all other Y_K Return Y_K to the task manager, to re-form the input and send to CS. Else if any value Y_K exists Send all Y_K to the task manager, to return these options to the user. Else End execution. The task manager responds to the user that the input statement was not recognized.</p> |

Figure 3-2. Example of a procedure for an input requesting who teaches a specified course, where the name of the course is incorrect

Automatic Updates

Albert has two main procedures for automatic updates; one procedure updates the schedule of courses database, the other updates the CS scripts. Both procedures are written using Python and Linux OS scripts. The course schedule information is available to the students in the UF Registrar's Office webpage, which is updated daily at <http://www.registrar.ufl.edu/soc/index.html>. The system includes the Python library Beautiful Soup version 4.0, to extract the information from the webpage [54]. The script then parses the data and saves it in a format that is readable by CS. A Linux cron job runs the complete procedure every business day, i.e., every day the Registrar's information is updated.

A separate cron job runs a daily procedure of shell scripts that updates all the CS content. This scheduled task has multiple purposes. First, to copy and store all the data on a remote drive for backup purposes. The second purpose is to update the CS knowledge base with the course schedule information. The third purpose is for programmer ease, as this procedure allows the developer to work upgrades to the system at any time and know that the changes will be available on the next day. As detailed below, the system contains methods for automatic scaling up, which does not involve programmer involvement. Therefore, this procedure will also automatically update the system with the changes made by the user community and thus achieve the automatic scaling-up.

Scaling-Up the Natural Language Scripts

As discussed throughout this work, the productivity of NLU systems depends on the amount of data available for the design. Ideally, the NLP system will include methods to allow the system to scale-up, with minimal involvement from the developers.

For this reason, the design of Albert has measures in place to allow such improvements to the system. This work includes the design of one scaling-up method, specifically, a procedure to allow users to suggest unofficial names for courses.

Collecting the academic advising data includes gathering the official names for all the courses and a list of nicknames, abbreviations and acronyms that the ECE-UF community commonly uses for these courses. Reasonably, it is not possible to obtain every possible moniker for each course; therefore, a viable solution is to have the users submit new course names to the system.

Users have two methods to submit course names to Albert. The first method is through a direct request, i.e., they implicitly state that they want to submit a course name. The second method occurs when the system recognizes a request for course information, e.g. who teaches the course, when is the course or where is the course taught, but the name of the course is not recognized and no spelling correction was obtained. Figure 3-3 shows the algorithm for submitting a course name.

Algorithm: Allow users to submit course names

Input: System recognizes user intent to submit a course name OR course information was requested and the system did not recognize the name

Desired output: Acknowledgment of successful name submission

System requests the official title of the course or the course code

If course is recognized

Request from user the new course name.

Send course code and new name to the ECE-UF advisor for approval.

Else

Inform user that name was not recognized and to try again if preferred.

Figure 3-3. Algorithm for course name submission

The algorithm obtains from the user the official name of the implied course and the recommended course moniker. This 2-tuple is automatically sent to the ECE-UF advisor database, which the advisors can access online through an independent webpage developed in this work for the task.

The second phase of the scaling-up procedure involves the ECE-UF advisors using the online system to accept or reject the user proposed course name. If the name is rejected, the process ends. If the name is accepted, the 2-tuple is added to a table in CS, which was designed in this work for this purpose. With the automatic update described in the previous section, this data pair is available for users by the next calendar day. This section of the advisor online system was not published due to an interruption in the BSEE advising service and low user turnout.

Statistical Data Collection

As explained previously, the DM classifies input statements according to the information returned to the user. Hence, the system produces a computation of the indeterminate responses. However, the system does not provide an automatic estimate of false positives outcomes, that is, input statements incorrectly matched to a determinate response, or false negatives outcomes, i.e., input that should have matched. Therefore, the analysis of the false positives and false negatives outcomes is done through an estimation of the data. The results from the estimation allow computing the standard statistical evaluation metrics from information retrieval, POS tagging, parsing systems, named entity recognition systems and WSD systems, namely, field such as precision, recall, accuracy and the F_1 measure [18], [56].

Additional scripts developed for the system evaluation include procedures to count the number of logins per user, the number of input statements per user and how many suggestions each user made through the scaling-up process.

To submit the results from the academic planning process, users must answer a questionnaire of three Likert items, as shown in figure 3-4. The figure also shows an optional write-in text area where students can leave feedback. The results of the questionnaire and comments section are not sent to the ECE-UF advisors.

- Is this process helpful for your academic planning?
Very helpful 5 4 3 2 1 Not at all helpful
- Is this system easy to use?
Very easy 5 4 3 2 1 Not at all easy
- What is your opinion on the natural language chat application?
I like it a lot 5 4 3 2 1 I do not like it at all

Any comments, complaints, suggestions?

Figure 3-4. Questionnaire required to submit the academic planning process

CHAPTER 4 THE ACADEMIC PLANNING SYSTEM

The fundamental task of academic advisors is helping students complete the requisites for graduation. For the typical student, the academic advisor is someone who helps them choose which courses they should take and provides solutions for any issue concerning academic advancement. Students expect personal attention with precise answers for their unique questions.

The theme throughout this work is that Albert provides widespread solutions for FAQs, to allow students and advisors to focus their limited face-to-face time on personal discussions, developmental and educative advising. Notwithstanding, an account with Albert allows students to manage data stored during their conversations, as detailed in Chapter 3, and from their academic records. Using this data, Albert offers individualized recommendations on course enrollment and a service for students to send to their advisor, their choice of courses for enrollment. This chapter describes the academic planning system.

The University of Florida Registrar's Office has an online Integrated Student Information System (ISIS), with a degree-auditing component that is available for all of its students. ISIS allows students to study their academic record, complete course enrollment and to process official UF documentation. Experiential data from the ECE-UF advisors indicates students do not use ISIS regularly, due mostly to the system's presentation and student reluctance to dealing with it. Therefore, most students repeatedly show up for their advising appointments unprepared to discuss their academic course plan. The main objective of Albert's academic planning system is to resolve this issue.

The Course Plan Analyzer

The Course Plan Analyzer (CoPA) is a curriculum management and degree-auditing service, which contains an expert system that analyzes the student's academic record, to provide students their options for selecting a course plan for the ensuing academic term.

Obtaining the Academic Information

To answer questions related to a student's academic record, the effective and efficient step is to link Albert to ISIS to obtain the necessary information. Ideally, this would be through a secured private connection that would allow students to download their information from ISIS and upload the data to Albert, without revealing the user's identity. However, federal and state laws limit access to this information only to authorized personnel. Therefore, the practical options for this work were to use synthetic data or to develop a method to allow students to enter the minimal information necessary to evaluate their academic progress, without compromising their privacy.

The most useful option is to request students to manually enter their data. The decision is based on student benefit, the value for this research work, design cost and time constraints. Ultimately, having students examine and enter their grades is a favorable self-assessment exercise for planning their course plan.

At the implementation time of CoPA, the ECE-UF BSCEE degree was undergoing a renovation. Therefore, the published version of CoPA contains the BSEE program only. The rules and information of this academic degree were obtained by surveying the ECE-UF academic advisors, the UF website and the official UF catalog. CoPA contains all the rules and requisites for taking each course in the BSEE curriculum, plus every course from the ECE department.

User Interface

The main expert system and the database for CoPA are built with Python. The user interface of CoPA is built on PHP, a popular server-side scripting language that is well suited for web development. These independent modules of Albert, in Python and PHP, communicate via sockets. The user interface of CoPA runs inside the Albert webpage as an independent frame, allowing users to simultaneously use Albert's messaging system and CoPA. With this feature, students can ask Albert about academic records, academic rules, the official course schedule, details about the courses and related information while they enter the forms in the CoPA webpages.

Figure 4-1 shows a screenshot of Albert, where the web frame on the right shows the initial webpage for CoPA. This frame shows the upper contents of the CoPA webpage, with the additional information accessible by scrolling down on the frame. Appendix B shows additional screenshots of Albert and the CoPA webpages.

The CoPA System

Students can initialize CoPA by requesting it, with expressions similar to *I want to update my degree audit* and *I want to enter my grades*. To prevent an unintended initialization, Albert will ask users to confirm the request for CoPA before presenting the initial CoPA webpage. An example of this process is evident in figure 4-1. Upon initialization, CoPA presents a template with the student's academic curriculum, similar to how it appears in the UF catalog. The student can enter the following academic information in this template.

- Catalog year, the year of admission into the academic program
- Grades in completed courses
- Courses currently enrolled in
- Courses dropped

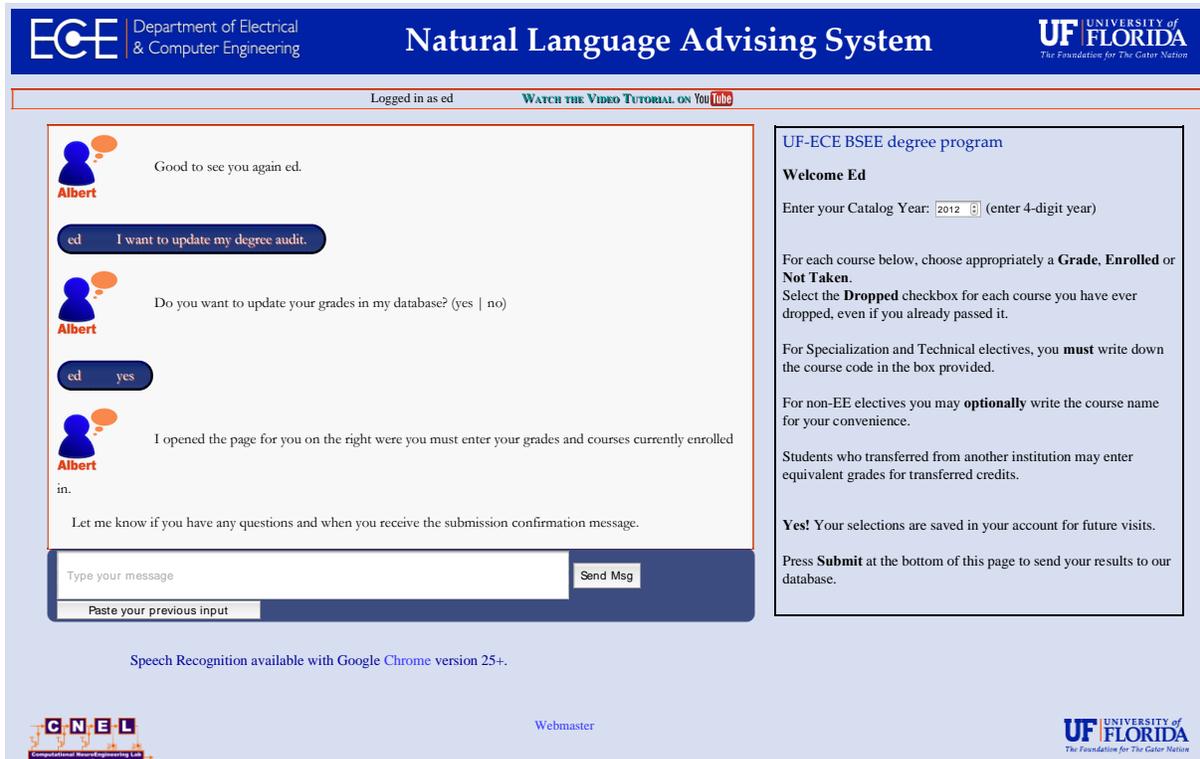


Figure 4-1. A screenshot of Albert with the right side showing the first page of CoPA opened inside the frame. The left side, the main window, shows the conversational exchange that took place to initiate the CoPA system.

Figure 4-2 shows a segment of the template for entering an academic record in CoPA. Students choose their status in each course using the drop-down menu and the checkbox. The default value for each course grade is *not taken*, so students need only to edit what they have taken. Students who have special cases, such as courses taken in other institutions, will have their course equivalencies issued by the Registrar’s Office.

In the template, elective courses have a text box where students can enter the name or code of the course. The BSEE program has a strict list of courses from which students must choose their technical and specialization electives. The template requires students to enter the correct code, for every course in this list that they have taken. Other elective courses do not include validation for the manual entry box.

Semester 1

Chemistry 1 (GE-P)

Dropped?

Chemistry 1 Laboratory (GE-P)

Dropped?

Humanities (GE-H) CourseName

Dropped?

Calculus 1 (GE-M)

Dropped?

Social and Behavioral Sciences (GE-S) CourseName

Dropped?

Semester 2

Chemistry 2 (GE-P) or Biological Science (GE-B)

Dropped?

Professional Communication for Engineers (GE-C)

Dropped?

Calculus 2 (GE-M)

Dropped?

Physics 1 (GE-P)

Dropped?

Physics 1 Laboratory (GE-P)

Dropped?

Semester 3

EEL 2000 Introduction to Electrical and Computer Engineering

Dropped?

Calculus 3 (GE-M)

Dropped?

Physics 2 (GE-P)

Dropped?

Physics 2 Laboratory

Dropped?

Computer programming elective CourseName

Dropped?

Social and Behavioral Sciences (GE-S) CourseName

Dropped?

Semester 4

Elementary Differential Equations

Dropped?

EEL 3105 Analytical Methods in Electrical Engineering

Dropped?

EEL 3111C Circuits 1

Dropped?

EEL 3135 Discrete-Time Signals and Systems

Dropped?

Summer

General Elective CourseName

Dropped?

Humanities (GE-H) CourseName

Dropped?

Semester 5

EEE 3396 Solid-State Electronic Devices

Dropped?

EEL 3112 Circuits, Systems and Signals

Dropped?

-
-
-

Semester 8

College breadth elective CourseName

Dropped?

EE Specialization elective

Course code (e.g. EEE4310C)

Dropped?

EE Technical elective

Course code (e.g. EEE4310C)

Dropped?

Semester 9

EEL 4924C Electrical and Computer Engineering Design 2

Dropped?

College breadth elective CourseName

Dropped?

EE Technical elective

Course code (e.g. EEE4310C)

Dropped?

Confirm your submission with Albert.

Figure 4-2. A segment of the template for entering the academic record in CoPA.

After students submit their academic record template, CoPA will determine:

1. The courses remaining in the plan of study to complete the degree requirements.
2. The courses the student can take based on the prerequisites and availability for the semester, along with a separate list for the technical and specialization electives.
3. The courses the student can repeat, if desired.

Once CoPA completes this analysis, Albert will invite students to prepare a course plan for the next term. When students accept to complete this process, Albert opens the second CoPA webpage, which contains the course information listed above. Figure 4-3 shows a screenshot of Albert with the message exchange that took place for Albert to open the CoPA webpage, after the student submitted the academic record. The frame on the right side of the screenshot, shows the uppermost section of the CoPA webpage.

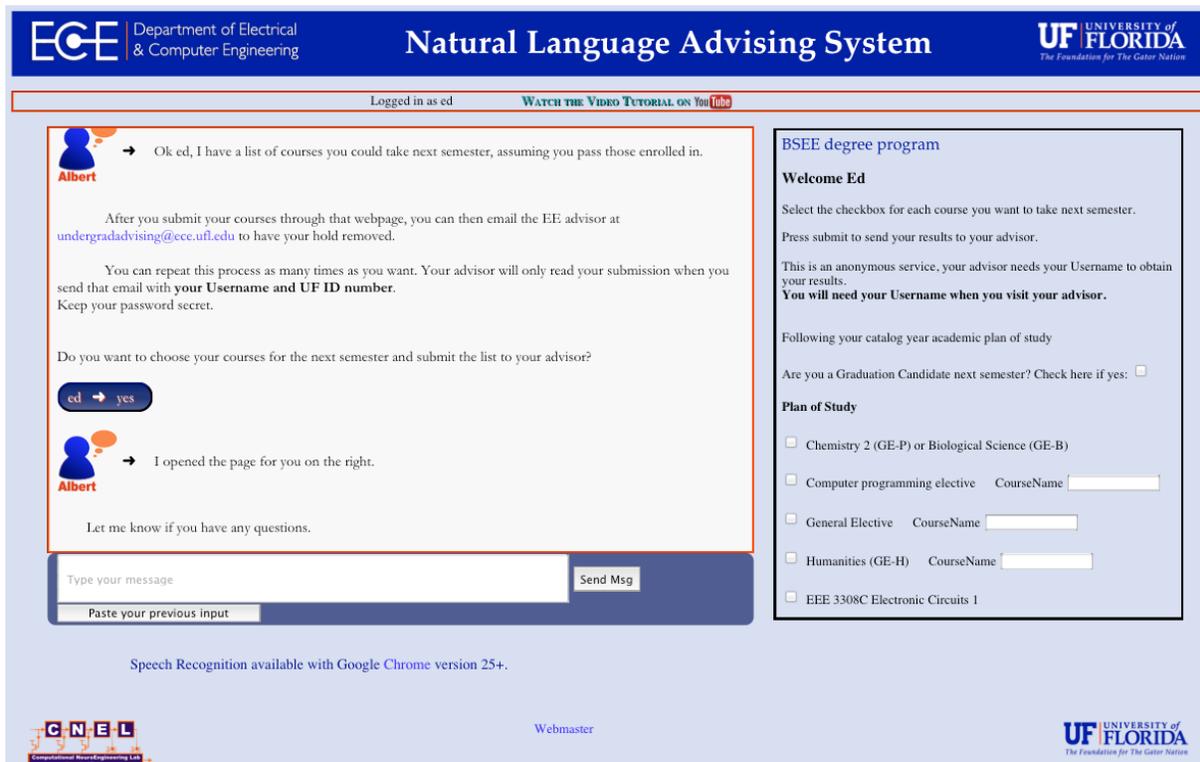


Figure 4-3. A screenshot of Albert with the right side showing the second part of CoPA

As seen in figure 4-4, each course listed in this CoPA webpage has a checkbox to the left, and for elective courses, a text box to the right. Students will select each course they wish to take during the next term and optionally enter the code or names of any elective course. The technical and specialization EE electives do not have an accompanying text box, given these are listed separately with their titles. In the most recent release, the special topic courses, which use the same numeration for multiple titles, also have a text box for entering the name. This separate list ends with a text box for students to enter special case courses that are not on the pre-approved list. The courses listed in the Plan of Study section are shown in the same order as in the template in figure 4-2, which is the order suggested in the BSEE curriculum.

Students who are graduation candidates can select that option at the top part of the CoPA webpage; see the upper-left part of figure 4-4. Students can submit additional courses that are not part of their major, by using the text-area form at the end of the last list of courses. The last portion of figure 4-4 shows the three Likert feedback questions that students must complete in order to submit their courses. The write-in text area at the end, where students can leave additional feedback, is not required for submission.

When students complete the course selection process, they submit the academic information to the advisor's online database. Advisors are not automatically notified of a submission, thus students can repeat the process as many times as they prefer. The advisor's database will only save the most recent submission. The feedback data is not saved in the advisor's database. Students can verify their submission with Albert.

When a student is prepared to discuss the course plan with the advisor, the student will provide his username, with which the advisor can obtain the student's

results from the online database. Students do not need to share their passwords with the advisor, thus keeping all their other data anonymous.

Albert's Advisor Website

Albert's advisor website provides access to the online database. The advisor website was built using PHP. Figure 4-5 shows the front page of the website, which contains a simple login form. To register, advisors will obtain the credentials from the developer. Once logged-in, advisors will see a form to search for the usernames, and to the right, the text pad area that they can use to take notes during their advising sessions. Figure 4-6 shows this webpage, after a successful search of the user *ed*.

ECE Login

Username:
Password:

Figure 4-5. Front page of the advisor website.

The screenshot shows the 'ECE BSEE Advising' website interface. At the top left, there is a search box labeled 'Search by Username:' with a 'Search' button. Below the search box, the results for user 'ed' are displayed. The text reads: 'ed is a graduation candidate for the next semester.' This is followed by a section titled 'Courses for ed' which lists courses chosen for summer (EEE 3308C Electronic Circuits 1, EEL4930 -- NLP) and for fall (EEL 3105 Analytical Methods in Elec Eng or MAS 3114 Comp Linear Algebra, EEL 4924C Electrical and Computer Engineering Design 2). There is also a section for 'EE Electives' listing EEL3834 Programming for Electrical and Computer Engineers, EEL4948 Practical Work in Electrical and Computer Engineering, and EEL4949 Co-op Work Experience Adv NLP. At the bottom of the results, it says 'Student comments: These are my notes'. To the right of the search results is a large text area containing the word 'test'. Below the text area is a 'Save text' button. At the bottom left of the page is a 'Logout' button. A note at the bottom right of the text area says: 'Save the notes for this user with the button below. The webpage is refreshed when saved. Change the size of text-area by dragging the bottom-right corner of it.'

Figure 4-6. The advisor website, with an example student report.

Upon a successful username search, the system will return the information submitted by the student divided in up to 4 lists, i.e., the courses chosen from the Plan of Study section, the courses chosen from the EE electives section, any course the student chose to repeat and the manually written entries. The results will also show if the student selected the graduation candidate option.

Evaluating CoPA

Albert, with the addition of the course plan system, was officially published for students of the BSEE program during enrollment period in the Fall 2013 semester. Students of the BSEE program were invited to use Albert during this period and the results obtained were used to evaluate the effectiveness of the system.

As a direct measurement of CoPA, the system records the number of users who submitted an academic record, the number of users who submitted a course plan to the advisor database and the catalog year of each user. The ECE advisor provided the number of users who requested an evaluation of their course plan submission in Albert. Indirect evaluation is done through the questionnaire at the end of the CoPA process, which contains one question that directly relates to CoPA, the user comments and surveys from the advisors.

CHAPTER 5 EXPERIMENTS AND ASSESSMENT

Albert was designed for rapid deployment to maximize the data collected. User data is crucial for the success of the system and the best available method to create the advising corpus. User data analysis was a major factor in Albert's design, as it triggered approximately seventy percent of the input templates produced.

Albert went through three incremental phases of data collection and testing. While data collected during the first two phases was not enough for significant statistical analysis, it did provide user behavioral analytics, and significant design decisions such as the need for a course planning system and a broad FAQ list on the webpage. This chapter describes the experimental phases and the major results obtained in each phase. The chapter closes with an overall assessment of the system and the current state of operation.

Experiments

All the experiments were designed for the undergraduate students of the ECE-UF programs BSEE and BSCEE. The ECE-UF department has approximately 950 students in these programs, with approximately 500 in the BSEE program. The experiments took place between October 2012 and December 2013. Each testing phase began when the ECE-UF advisors would inform the students of the availability of Albert and provided the web address for the system. The web address was not available in any other medium, including search engines.

Students visit their advisors in the Electrical and Computer Engineering Student Services Office (ECE-SSO). To encourage students waiting to meet the advisors inside the ECE-SSO, a desktop computer was deployed for the exclusive usage of Albert. This

computer was available during normal operation hours of the ECE-SSO, i.e., business days from 8:00AM to 12:00PM and from 1:00PM to 5:00PM. The computer and Albert's website were available 24/7 during the entire fifteen month period, however, student use was minimal outside each delimited phase and thus not included in the results.

Undergraduate students at UF must visit their advisors before enrollment period, to discuss their projected course plan and obtain authorization to enroll for courses using the university's online system. The official period for these visits starts following the publication of the course schedules by the UF Registrar's office. Students are required to bring a draft of the course plan form they must turn in during the meeting; however, most students arrive unprepared for this task. In addition, this period is the busiest for advisors each semester, as many students arrive at the same time, usually shortly before their scheduled date for enrollment. Each experimental phase for Albert refers to this advising period during the Fall 2012, Spring 2013 and Fall 2013 semesters respectively.

Since the objective with Albert is to provide an advising experience that is as close as possible to traditional human interaction, advisors did not provide students any tutorials on how to use the system or how to communicate with it. Albert's website would provide the login instructions, example questions and for phase three, a video tutorial.

Results refer to users as unique accounts created. For fairness, identifiable log files were removed, including those from faculty and advisors. Users who made two or fewer statements were also removed. The remaining user files were included in the results, even when the user never meant to converse about academic advising.

Phase One

The first experimental phase took place during a six-week period between October 2012 and November 2012. For this implementation, the system did not require a password and CoPA was not available. The webpage showed some instructions and a handful of example FAQ. The main NLU script had approximately 160 input templates, for roughly 100 unique responses.

Log files from 52 user accounts revealed the average number of logins per user account at 1.5 and the average number of input statements per user at 16. The input count histogram in figure 5-1 shows that 50% of users made 10 or fewer entries.

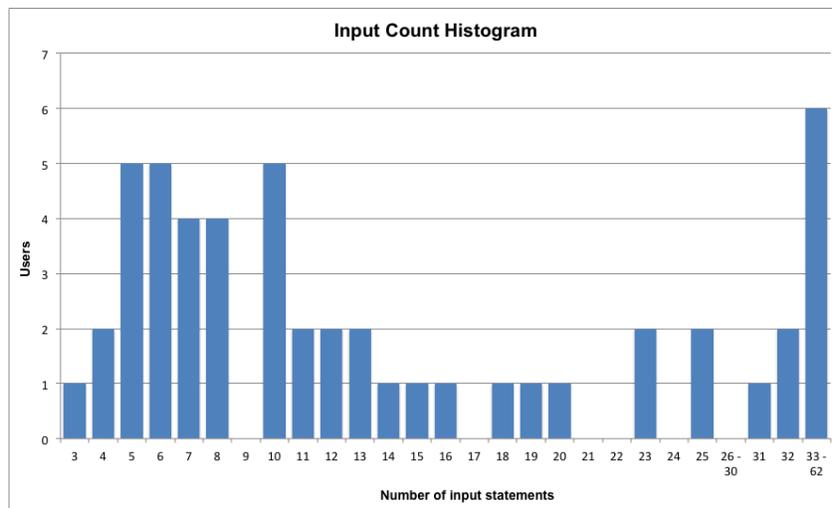


Figure 5-1. Input statement histogram for phase one.

The data shows that most users initiated the dialog with a predetermined question. In general, most users experimented with random statements, with many using the example FAQs and the help command. Therefore, to teach students about the information in Albert, the FAQ in the webpage display was increased in the subsequent

implementation. Users with over fifteen statements showed enthusiasm for the chat system by asking from a wide range of topics. The FAQ list would also influence students to focus on these topics.

The statements that were not correctly recognized had three significant reasons. As expected, the first reason was that the information was not included in the design. The data collected provided information for the subsequent designs. The second reason was that users would approach Albert as if it were a standard search engine, that is, they used mostly one-word entries and fragmented statements that did not follow natural language. This result prompted adding to Albert the ability to recognize keywords of interest to inform the user about the information available on the topic. This allowed users to navigate towards the solution they seek through natural conversation.

The third most common reason for errors was that students requested answers that required knowledge of their academic records. While the system stated that accounts were anonymous and student registration only required choosing a random username to login, students still requested answers that required knowledge of their academic records. Some students explicitly stated that the system needed access to their academic records to provide a service they would appreciate. The log files also showed that the system needed a robust spell checking process.

Phase Two

The second phase of experimentation took place during a four-week period between March and April 2013. For this implementation, the system did not require a password and CoPA was not available. The website had 14 questions on the right-side help frame, as seen in figure 5-2. The main NLU script had close to 300 input templates, for roughly 160 unique responses.

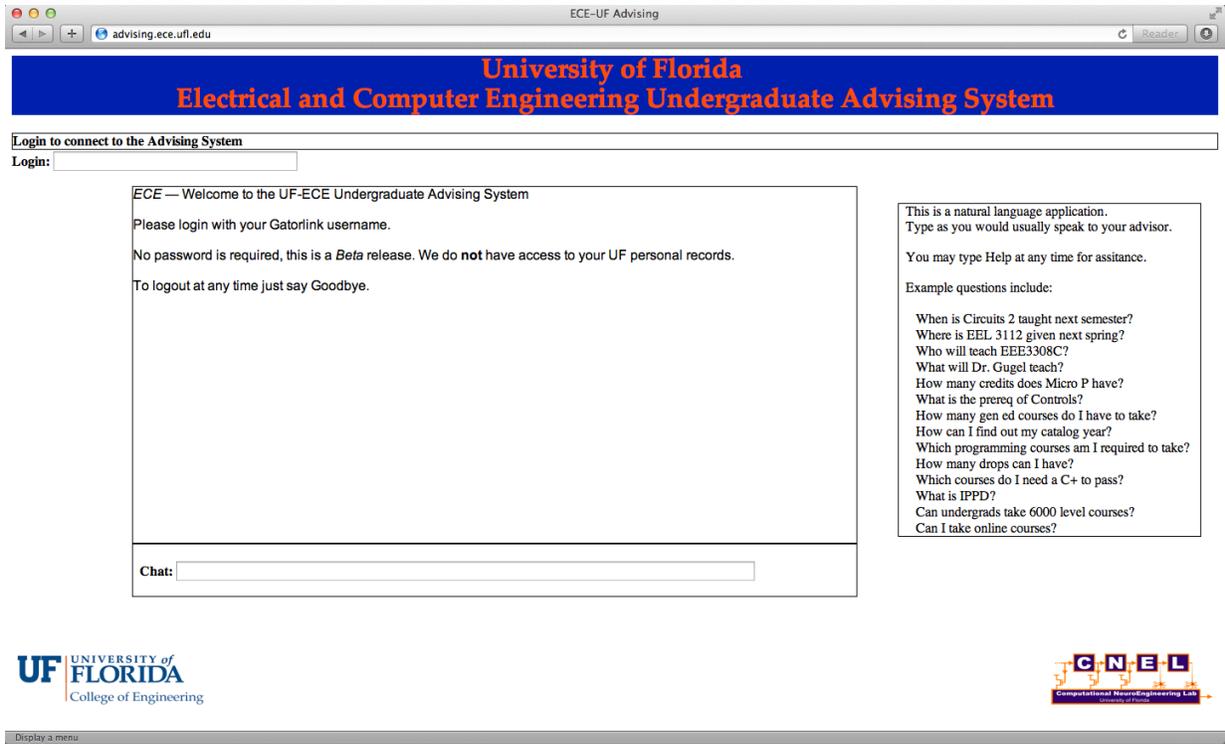


Figure 5-2. A screenshot of Albert, during phase two, showing the FAQ list on the right.

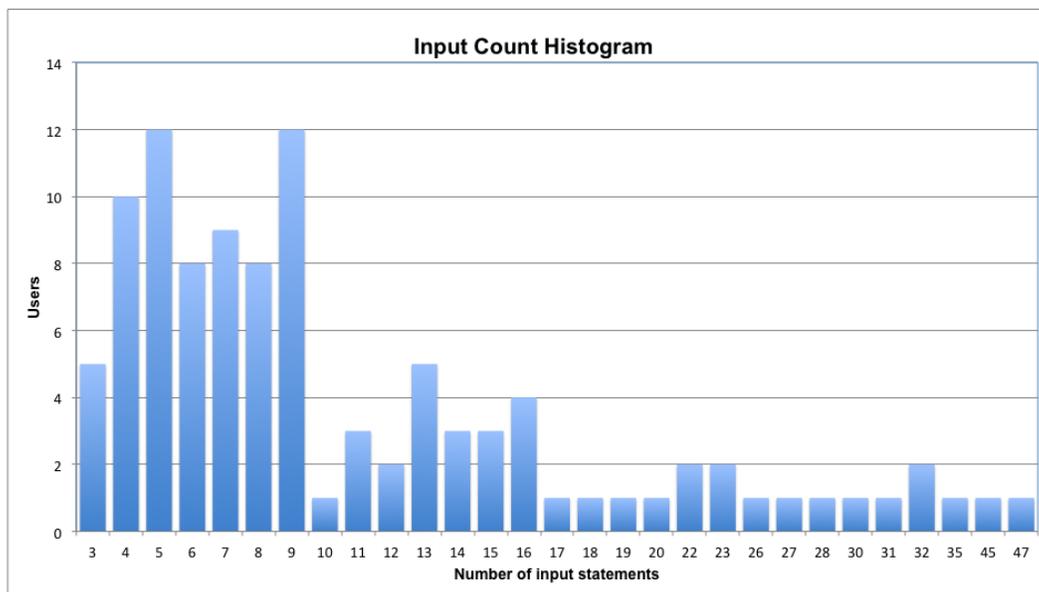


Figure 5-3. Input statement histogram for phase two.

Log files from 103 user accounts indicated the average number of logins per user account was 1.9 and the average number of input entries per user was 11.4. The input count histogram in figure 5-3 shows that 62% of the users made less than ten statements. For clarity in figure 5-3, the values for input statements are discontinuous after twenty.

The log files showed that the most requested feature was personalized recommendations for course enrollment. Other highly requested features were also related to course enrollment planning, e.g., the ability to track their progress through the degree and obtain recommendations for course sequence planning. These requests prompted the development of CoPA. With respect to recognition errors, the data was inline with the results obtained in phase one. Correspondingly, the data provided advising topics for the ensuing phase.

While Albert contained close to 160 unique answers to academic FAQs, most students did not seek many of these answers, probably due to not having the need for them at the time of the conversation or because they lacked the academic experience to necessitate finding out the information. Many students opted to start the conversation with one of the FAQs listed on the website and then ask similar and follow-up questions as they thought of them. For many of these students, this exercise proved to be a useful academic informative session. To instigate further dialog and display the system capabilities, the FAQ list was extended for the subsequent experimental phase.

Phase Three

The third experimental phase took place during an eight-week period between October 4, 2013 and November 27, 2013. The system for this implementation contained all the features described in Chapter 3, Chapter 4 and the screenshots in Appendix B.

The main NLU script had approximately 415 input templates, with over 200 unique responses. For this phase, only the BSEE advisor sent out an announcement to the students. Therefore, the amount of BSCEE students who communicated with Albert was minimal. The system compiled data from 387 users during this phase.

The plan for CoPA was to complement the advisors during the pre-enrollment task. However, shortly before this task period began the BSEE advising position became vacant. As a response, Albert was assigned the full task of advising BSEE students during the first half of the period. Albert was initially announced to students who had completed at least two years of the degree and to all students afterwards. In order to receive authorization to enroll for courses, students would complete a course plan using CoPA and then e-mail the request along with their Albert username to the ECE-UF advisor. The name CoPA was not revealed to the students. The instructions specified that students should submit their course plan using Albert.

While the interruption in the advising services prevented testing the second phase of the scaling-up process, the first half of the process was operating. The system recorded submissions for course names from six users. Three of these users made useful contributions.

As an official ECE-UF service, students were offered four sources for help with using Albert and completing the CoPA process. The first was an email address exclusive for this service, which nobody used. As the second source, the developer was available for tutoring in the ECE-UF student computer laboratory during the first three-week period. About twenty students attended these sessions. The third source was the ECE-UF Student Branch of the Institute of Electrical and Electronics Engineers (IEEE).

Two weeks before phase three began, the developer presented a workshop to approximately twenty chair members of the ECE-IEEE. The ECE-IEEE estimates that ten students visited them for help with Albert during this period.

As the fourth source for help with completing the CoPA process with minimal effort, Albert featured a link for an online video tutorial, hosted in the YouTube website. The tutorial video length is 11 minutes, with about five minutes describing an example of using CoPA. The rest of the video describes additional features and the usage of Albert. Figure 5-4 shows a graph with viewer data provided by YouTube.

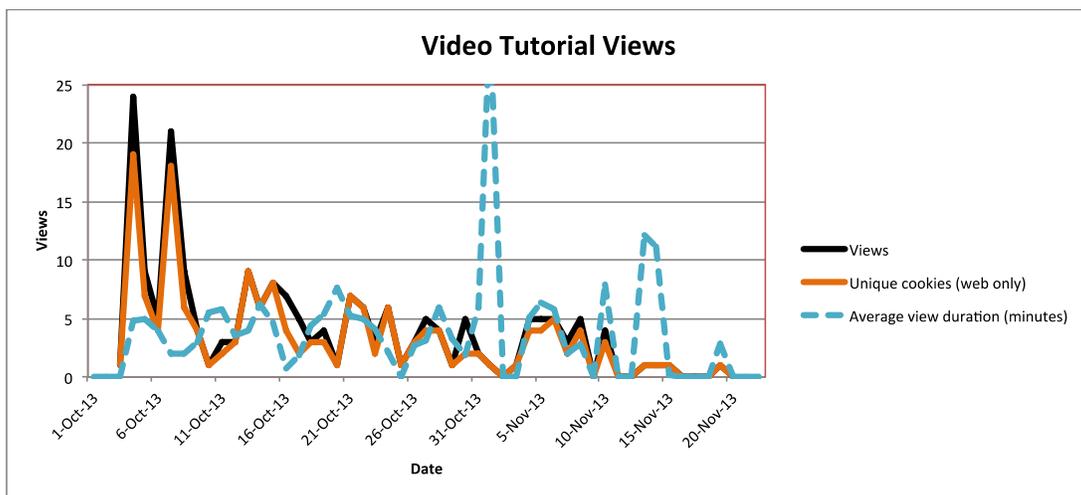


Figure 5-4. Statistics for the video tutorial. Data obtained through YouTube Analytics.

Though video views dropped to zero by the end of November, the YouTube results do not necessarily represent unique ECE students using Albert. Nonetheless, the plot shows that 19 computers with unique cookies viewed the video on October 4, the day the advisor made the announcement of Albert. These 19 views had an average duration of approximately five minutes. On the next business day, Monday October 7,

the video had 18 unique cookie viewers with an average of five minutes per view. With these numbers reflecting peak views, clearly the tutorial is too long. The plot also shows students tend to deal with advising topics mostly at the beginning of the week. Views picked up slightly after November 4, the day that enrollment began, though by then, the BSEE advisor was available for service as well.

Figure 5-5 shows the input count histogram for the 387 users during this phase. The histogram shows that 53% of users made less than ten entries. Many users used the system explicitly for the CoPA process. Registering and completing the CoPA process required a minimum of four statements. The average login per user was two, with 78% of users having two or less.

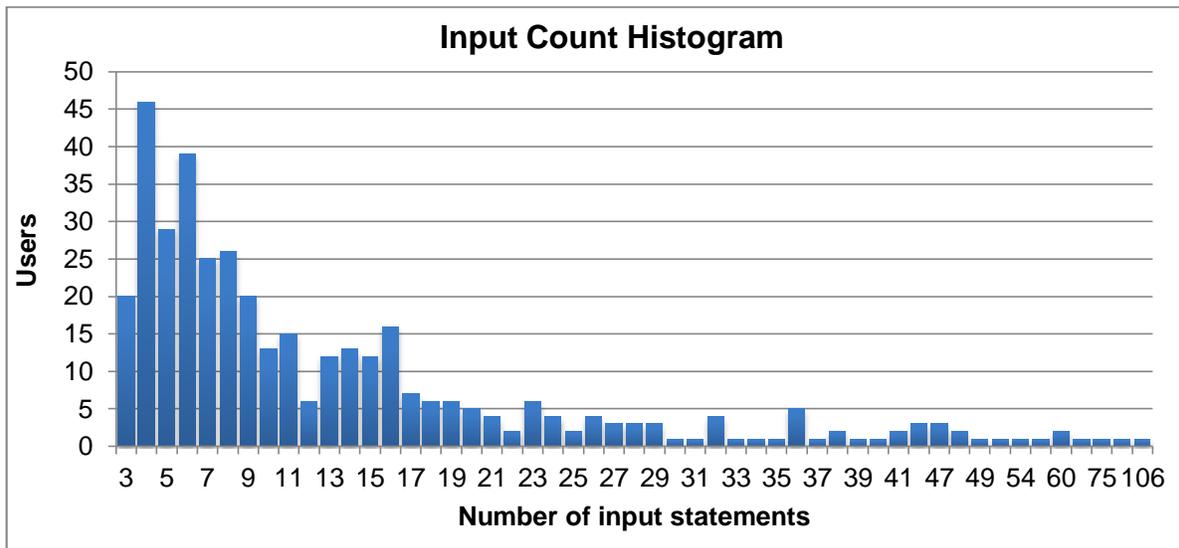


Figure 5-5. Input statement histogram for phase three.

From the user total, 292 entered their grades. Figure 5-6 shows the catalog year, that is, the year of admission into the BSEE program, of these users. Clearly, many users did not write the correct response, as 2014 or more is not possible. Fifty-five

users asked Albert to explain the phrase catalog year. It is likely that many students entered either the current year or the expected graduation year, judging by those who entered a future date. The results for the years 2009 through 2012 should be accurate and the overall shape of their distribution is as expected, i.e., with most students in their second and third year of academic progress. This data will contribute to the assessment of the ECE-SSO services.

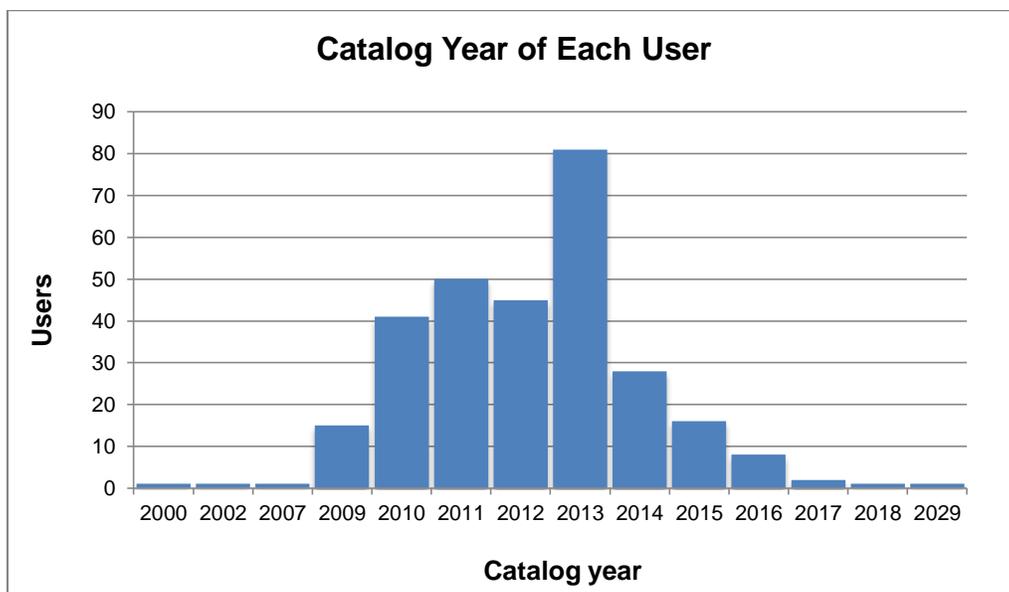


Figure 5-6. Catalog year of each user who filled an academic record with CoPA.

The ECE-SSO estimates that 240 students requested and completed the course plan analysis using Albert, which is about 50% of the BSEE students and 65% of the user accounts. Albert collected survey results from 224 students. The discrepancy between the number of surveys and the ECE-SSO estimate is mostly due to students who sent their course plan via email, while claiming to have completed CoPA. This

outcome is expected when people working under time pressure integrate a new process into a known system. Table 5-1 and figure 5-7 show the results from the survey.

Table 5-1. Results from the survey at the end of the CoPA process

| Question | Is this process helpful for your academic planning? | Is this system easy to use? | What is your opinion on the natural language chat application? |
|-----------------------|---|-----------------------------|--|
| Average Value (1 - 5) | 2.92 | 3.29 | 3.14 |

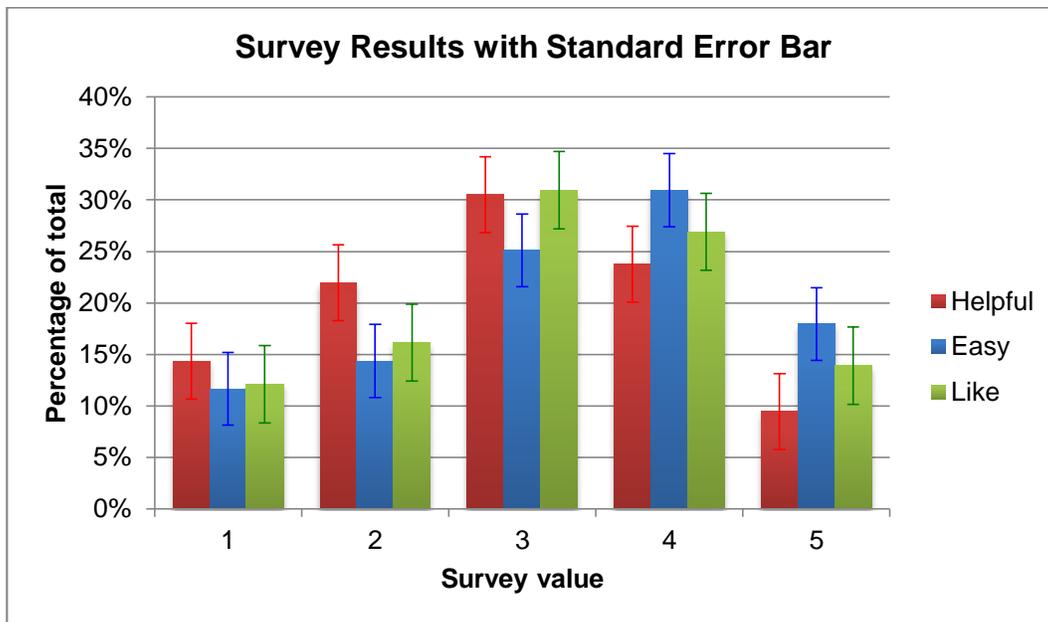


Figure 5-7. Survey results with a standard error bar for each question.

The survey results reflect mostly positive reviews, as a majority of students gave values of three and four for how much they liked it and how easy it was. The rating of helpfulness is smeared over two, three and four. Conversational systems with similar surveys obtained averages within the low threes, to four and a half [46], [47]. However,

these systems have a reduced input domain, as the system is who drives the dialog, and users know beforehand what to expect from the system.

To help clarify the reasoning behind the survey results, a comment section was added to the survey on October 12, 2013. Of the 191 users who completed CoPA after this date, 61 wrote comments, of which eleven were positive feedback, 45 were negatives remarks and five were technical suggestions with neutral sentiment. The positive remarks were general compliments and appreciating the course plan system. Of the negative comments, 47% criticized the system as not appropriate for substituting a human advisor, 42% criticized having to manually enter their academic record, while the remaining 11% reported technical difficulties and unique situations. Therefore, a minimum of 89% of the negative comments was outside the scope of the implementation.

The disapproval of the method for entering the grades was completely understandable and expected. As described in Chapter 4, this was the appropriate resolution for the privacy concern in this testing phase, as the academic department would manage the final product and thus have the resources to automatically transfer the information upon student request.

To address the comments about substituting a human advisor, the main solution is to inform the students about the objectives for Albert and allow the advisor to integrate the system into the daily tasks. A basis throughout the design of Albert is that to appreciate the value of this service, students must understand that the objective is to assist and not replace. The fact that almost half of the negative comments concern this statement validates its significance. Unfortunately, the disruption in the advising

services hampered this guideline. Altogether, 89% of the negative remarks require realistic provisions from the academic institution and thus easy to implement.

Appendix C shows example dialog extracted from the user log files. Responses by the system are curtailed for reader ease, while user grammatical and typographical errors are preserved for authenticity.

The ECE advisor, who processed the student submissions from Albert, completed a review of the system and the course submission process. The review is available in Appendix D. In general, the advisor was very encouraging of the system and service provided, with recommendations in line with student reviews.

Phase three introduced the measure to quantize Albert's input response as described in Chapter 3. This measure classifies each input in Albert as the following.

- Literal match (LM) are input statements that match a FAQ listed on the webpage.
- Partial match (PM) are statements that have partially matching templates.
- Outcome negative (ON) includes false negatives (FN), i.e., statements not recognized by the system and true negatives (TN), i.e., statements outside the design scope.
- Outcome positive (OP) includes correct responses or true positives (TP), and false positives (FP).

The LM statements are 71 FAQs listed in the webpage. These include the examples for initiating CoPA, the help command and an example from each topic in Albert. The LM statements do not have any uncertainty for recognition; therefore, these are subtracted from the total input to evaluate the system error. As CoPA was a main feature of phase three, eliminating these commands increases the estimated error.

Most of the PM statements the system responds with are advice for the user to obtain the information of interest or incomplete answers. This includes some statements

that are outside the scope of the system, to which the system responds with statements related to the topic and lets the user know that more information on the topic is not available. Although these templates were successful in their design, given the user did not directly receive the desired response, these are not classified as outcome positive.

Results are available from 366 users between October 7 and November 27.

Table 5-2 shows the results of users who made an expression under each classification and the amount of input statements under each class.

Table 5-2. Results from Albert’s classification of the input statements

| Input classification | Total | LM | Total Original | PM | ON | OP |
|----------------------|-------|--------|----------------|--------|--------|--------|
| Users | 366 | 60.49% | - | 11.99% | 55.04% | 99.73% |
| Input Statements | 4952 | 12.52% | 4332 | 1.69% | 14.66% | 83.52% |

The results in Table 5-2 show that 60% of the students copied an instruction exactly as written, which for all purposes is akin to making a selection from a menu. While this result suggests that the interface could benefit from menu selections, the objective of this experiment is to encourage unrestricted expressions. Having approximately 80% of the students initiate the CoPA process and 60% using example input shows the student preference of using the quickest possible method to achieve a goal. Nevertheless, to serve as an educational tool, the previous data showed students benefit from a display of the FAQ. A solution is to include a display of topics with less example statements, while extending the NLP routines that allow users to access data by navigating through topics. While this approach is not within the scope of the vision for Albert, the data shows that including both approaches would increase the user-base

and allow users to explore the capabilities of the system through a more familiar experience.

Table 5-2 shows that less than 15% of all input was not recognized, however, these statements came from 55% of the users. This result is expected from a system with a restricted domain that accepts all type of input. Any user who decides to test the boundaries of the system will contribute to this result. Figure 5-7 shows the distribution of the ON classified statements for these users.

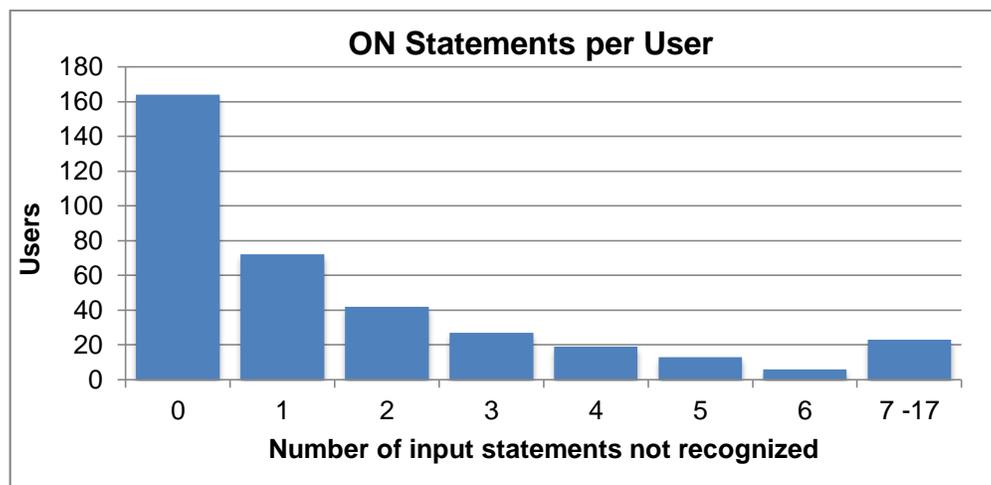


Figure 5-8. Distribution of ON input statements.

To reduce the ON responses, as in the previous phases during revision of the log files, false negatives were continuously identified and updated in the NLU system. As the data collection increased, the number of false negatives decreased. By the fifth week of the eight-week period, the amount of false negatives per user had dropped to almost zero. Approximately 25% of the users accessed the system during this

culminating period. During these last three weeks, no additional measures were taken to minimize ON outcomes from statements that were outside the scope of the design.

To determine the number of FP and FN, it is necessary to manually evaluate the OP and ON statements. For this evaluation, a FP outcome is a response that is not relevant to the input statement. This definition of FP error conditions the system's capabilities of responding to the statement with respect to the available information. For example, if a user asks, "*Who is my financial advisor*", the system response "*For financial information please see the following [website](#)*" is not classified FP as the answer is relevant and it is the best response available in the system. If for the same input the system replied "*Your BSEE advisor is Shannon M. Chillingworth*", the response is classified as FP, given it was not generated by the expected template by design.

A FN outcome occurs when the input statement has an available answer, yet the system did not respond correctly. Following the same example as above, if a user asks, "*Who is the person in charge of financial advising*", the system response "*I do not understand that question*" is classified FN, as this question should have been responded as stated previously. TN outcomes occur when the input statement is outside the design scope. The FN outcomes include cases caused by technical difficulties in other areas of the system; however, with respect to the overall system performance they are still incorrect responses.

It is possible to estimate the FP and FN results by selecting a random sample of the OP and ON statements. To select the samples, a uniform random number is assigned to each of the 3618 OP statements and to each of the 641 ON statements. Each statement is tested for the binary outcome FP or not FP, and FN or not FN

respectively. By treating each statement as an independent random variable, the outcomes of the tests will follow a Bernoulli distribution. In the Bernoulli distribution, the maximum variance $p(1 - p)$, occurs when the mean p is equal to one half, thus the variance is equal to one fourth. For a given sample size n , by the central limit theorem, the distribution is closest to the standard normal distribution when p is near one half. Under this scenario, a conservative estimate of the sample size n needed to estimate p with a confidence level $1 - \alpha$ and margin of error e is given in equation 5.1, where the $\lceil \cdot \rceil$ operator represents rounding to the next integer and Z_α is the estimated standard score for a given two-sided confidence level [57].

$$n = \left\lceil \frac{Z_\alpha^2}{4 \cdot e^2} \right\rceil \quad (5.1)$$

For a maximum margin of error equal to 10% and a confidence interval of 95%, equation 5.1 returns a minimum size of 97 samples. Table 5-3 summarizes the result for each test.

Table 5-3. Results from the outcomes error estimation

| Outcome | Total Statements | Confidence Interval | Error Margin | Sample Size | Error |
|----------|------------------|---------------------|--------------|-------------|--------|
| Positive | 3618 | 95% | 10% | 97 | 15.46% |
| Negative | 641 | 95% | 10% | 97 | 24.74% |

The tests returned 15 false positive statements, for 15.46% of the OP samples and 24 false negatives for 24.74% of the ON samples. Using these results, it is possible to determine the statistical measures precision, recall, accuracy and the F_1 measure [18]. Precision is defined as

$$P = \frac{TP}{TP+FP} \quad (5.2)$$

to measure the percentage of outcomes that were correct.

Recall is defined as

$$R = \frac{TP}{TP+FN} \quad (5.3)$$

to measure the percentage of positive outcomes given the statements with expected outcome positive. Accuracy is defined as

$$A = \frac{TP+TN}{OP+ON} \quad (5.4)$$

to measure the percentage of statements that obtained the expected outcome. The unbiased F–measure, or F_1 measure, is the harmonic mean of precision and recall.

$$F_1 = \frac{2PR}{P+R} \quad (5.5)$$

Table 5-4 summarizes the result for each test.

Table 5-4. System performance estimation results

| Precision | Recall | Accuracy | F_1 measure |
|----------------|----------------|----------------|---------------|
| $\geq 84.54\%$ | $\geq 77.36\%$ | $\geq 79.90\%$ | ≥ 0.81 |

The results show the system performance metrics are all close to 80%. These results are estimated minimums; given the error is an estimated maximum. Recent studies for comparable e-learning systems show Albert offers a high-level performance for the complexity of the task [58]–[59]. Regarding the main NLP tasks in Albert, specifically keyword extraction, question answering and natural language interfaces, recently published systems that require data corpora for training, obtain results that show the performance of Albert is highly competitive [53], [60]–[63]. Another possible performance measure is a study of human accuracy and response time, though such a study is not available. In any case, the results show Albert offers a competitive

performance versus the available technologies, and much promise for advancement as data collection continues.

The results for precision and recall are in line with the design goal of providing students with high precision answers and minimizing false positives. At the same time, the results show that over 75% of the queries to the system, obtained a reply that is effective versus an input-not-recognized type answer. This result is valuable considering users did not have any training on how to use the system.

Analysis Summary

The enhancements developed throughout the experiments proved successful in increasing system performance and user interest for the system. The results of the third experimentation phase showed Albert has developed into a practical and valuable application for academic advising. Overall, ECE-UF students and ECE-SSO were appreciative of the services provided and supportive for future developments.

The data shows that, as is customary in electronic text communications, students prefer to ask questions using the minimum amount of words possible, instead of through a Standard English sentence. Therefore, Albert should include methods to allow this user preference, while concurrently inspiring students to use complete expressions.

As a new service, students were initially reluctant to agree to use Albert for advising chores. Indeed, much effort is required in marketing the service as a useful option versus visiting a human advisor, who is usually a short walk away. However, during peak periods, when the waiting time is long, the predisposition to experiment with a new option is very high. Therefore, the implementation of CoPA, a tool for an essential process that all students must complete each academic semester, confirmed the viability of Albert as a real world solution.

Current State of Albert

Albert continues to serve the ECE-UF department during the Spring 2014 semester. During this fourth experimental phase, Albert will continue growing as an effective advising solution. As in previous phases, Albert has been updated to reflect the most recent results, including a revision of the FP and FN statements from the third phase. For the student enrollment period of the semester, Albert's fourth phase will feature a twofold process for CoPA, that is, it will allow students to simultaneously prepare a course plan for summer and fall.

CoPA will provide students with a richer course planning experience by also providing information to plan for courses ahead of the ensuing semester. Through requests similar to *when can I graduate* or *in how many semesters can I finish my degree*, students will obtain a suggested academic track for the courses remaining to complete the degree, divided by semesters.

The advisor's webpage was updated to allow the advisor to save the notes they write on the provided text pad area. Their notes are saved in the server and they retrieve them upon searching for the username. The ECE-UF BSEE advisor requested this feature, as the system becomes an integral database of the daily advising tasks.

The statistics measured from the data in phase three will provide the baseline for the performance evaluation of phase four. In addition, the ECE-SSO can compare the service provided between both semesters and provide additional evaluation of the convenience of this system with their tasks.

The main webpage design was updated to meet most of the accessibility compliances of the Americans with Disabilities Act and Section 508, 29 U.S.C. § 794d, of the Workforce Rehabilitation Act of 1973.

CHAPTER 6 CONCLUSION AND FUTURE WORK

Conclusion

The fundamental objective of this work is to provide students with an automated online advising experience that is as close as possible to traditional human interaction. This work presents the design, development, deployment and evaluation of an intelligent natural language conversational system for academic advising. The online system is available 24/7, it includes an advising dialog engine, an academic planning system that integrates the students with the advisors, and a method to allow the users to contribute to the system's knowledge base.

The system offers academic departments an instrument to potentially increase student integration, retention, satisfaction and performance, by providing additional advising services without the need for additional personnel. This achievement is possible by allowing academic advisors to allocate resources to tasks typically classified as developmental and educative advising, instead of repetitive time-intensive tasks related to prescriptive advising.

This work contributes real-world solutions for the academic community, through a unique combination of software applications and advanced NLP techniques. Every resource used for this system is original or available open sourced. With the experimental data collected, this work has originated a KB of academic advising FAQs that will serve to build a corpus on the topic. Throughout extended use of the system, Albert can potentially exhaust the scope of academic advising FAQs to allow development of systems for multiple programs through mostly answer editing.

This system is currently in operation for students of the BSEE and BSCEE degree programs of the University of Florida's ECE Department. The online system Albert provides instant service as an academic advising assistant for students, professors and additional members of the academic community. This system allows users to obtain academic information easily through communication analogous to how two people interact, thus not requiring a user learning curve.

After three major experimental evaluation periods, the system has proved to be a practical and valuable application for academic advising, based on positive reviews from the users, the advising personnel and the statistical measures. Within a confidence interval of 95% and a margin of error of 10%, Albert showed an estimated precision of 85%, a recall of 77%, an accuracy of 80% and a F_1 measure of 0.81. Overall, users were supportive and excited about future developments.

Future Work

The immediate developments for this system include expanding CoPA for multiple semesters and adding a computerized method for submitting the academic record. Using a full academic record from the students will increase the services the system can implement for students and advisors.

Already in development, the advising corpus will provide the data to integrate statistical methods into the dialogue manager, such as those described in Chapter 2. Additional future updates include an escalation mechanism to forward selected student conversations to the advisors, adding course information from other academic departments and traditional advising tasks referred to in Chapter 2. For example, Albert could provide students the available paths towards graduation and suggestions for a balanced course plan.

APPENDIX A
TOPICS COVERED BY ALBERT

1. Topics related to managing CoPA
2. Determine the default academic term for student questions
3. Introductions
4. Information related to each curriculum
5. Academic rules from the university and academic department
6. Information related to classes in the students academic program
7. Course schedules and related information from the Registrar's webpage
8. Course prerequisites
9. EE technical electives and core courses
10. EE specialization courses
11. CEE technical electives
12. CEE major courses
13. Breadth electives
14. Tracking courses
15. Non-ECE courses
16. Internships
17. BSMS program
18. Minors
19. Financial assistance
20. Faculty and staff office hours
21. UF academic calendar
22. Student organizations
23. Academic questions not captured by any previous topic
24. UF campus
25. Teach information to Albert
26. General and off-topic questions and comments
27. Information about Albert
28. Keyword catching
29. No response matched; default answers

APPENDIX B SCREENSHOTS OF THE ALBERT WEBSITE

The screenshot shows the main interface of the ECE-UF Natural Language Advising System. At the top, there is a blue header with the ECE logo (Department of Electrical & Computer Engineering) on the left, the title "Natural Language Advising System" in the center, and the UF University of Florida logo on the right. Below the header, a navigation bar contains the text "Login to connect to the Advising System", "Chrome, Safari and Firefox recommended", and a "WATCH THE TUTORIAL ON YouTube" link.

The main content area is divided into several sections:

- Registration/Login:** A section for "New Users - to register an account check here" with a "register" link. Below it are input fields for "Username:" (containing "Anonymous Username") and "Password:", followed by a "Login" button.
- Welcome Message:** A large white box with a red border containing the following text:

Welcome to the ECE-UF Natural Language Advising System

Register or login by entering your preferred username & password. **Do not identify yourself!** Registered users have their data saved for ensuing sessions.

Your participation is anonymous. We do **not** have access to your UF personal records. Be careful when registering a password. To ensure anonymity, there is no password recovery method.

To logout at any time, just say Goodbye.
- Message Input:** A dark blue box at the bottom left of the main content area with a "Type your message" input field, a "Send Msg" button, and a "Paste your previous input" button.
- Example Questions:** A dark blue box on the right side titled "Ask Albert, our Natural Language Advisor, the same questions you would ask your advisor, including" contains several example questions in orange text:
 - I want to choose my courses for the next semester.
 - What courses do you recommend I take?
 - When is Circuits 2 taught next semester?
 - Where is EEL 3112 given next spring?
 - What will Dr. Guge] teach?
 - Who will teach EEE3308C?
- Course Selection:** Below the example questions, it says "Start your course selection process with" followed by two options:
 - I want to update my degree audit.
 - or
 - I want to enter my grades.
- Teaching Assistant:** Below that, it says "Teach Albert your preferred name for a course:" followed by an example:
 - I want to teach you a course name.
- Help:** At the bottom of the example questions box, it says "Type Help at any time for more assistance." and a button labeled "Press here for more example questions".

At the bottom of the page, there is a "Speech Recognition available with Google Chrome version 25+." notice, a "Webmaster" link, and logos for CNEEL (Computational Networking Engineering Lab) and UF University of Florida.

Figure B-1. Main webpage for Albert

Ask Albert, our Natural Language Advisor, the same questions you would ask your advisor, including

I want to choose my courses for the next semester.
What courses do you recommend I take?
When is Circuits 2 taught next semester?
Where is EEL 3112 given next spring?
What will Dr. Gugel teach?
Who will teach EEE3308C?

Start your course selection process with

I want to update my degree audit.
or
I want to enter my grades.

Teach Albert your preferred name for a course:

I want to teach you a course name.

Type **Help** at any time for more assistance.

Press here for more example questions

Figure B-2. Main FAQ webpage.

The screenshot shows the main webpage for the ECE-UF Natural Language Advising System. The header includes the ECE logo, the Department of Electrical & Computer Engineering, the system title 'Natural Language Advising System', and the University of Florida logo. Below the header, there is a navigation bar with links for 'Login to connect to the Advising System', 'Chrome, Safari and Firefox recommended', and 'WATCH THE TUTORIAL ON YouTube'. The main content area features a login form with fields for 'Username' (containing 'Anonymous Username') and 'Password', and a 'Login' button. To the right of the login form is a list of example questions, including 'I want to choose my courses for the next semester', 'What courses did I select for next semester?', 'What courses do you recommend I take?', 'I want to teach you a course name.', 'When is E Circuits taught next semester?', 'Where is EEL3135 given next spring?', 'Who will teach EEL 4201L?', 'Will Dr Harris teach anything next semester?', 'How many credits does Micro P have?', 'What is the prereq of Controls?', 'What is the course number of DIC?', 'Can I take Nanodevices without the prerequisite?', 'How many gen ed courses do I have to take?', 'How many critical tracking courses are required for my major?', 'How can I find out my catalog year?', 'Where can I see my curriculum?', 'Where can I find courses' syllabi?', 'How do I find my upper division GPA?', 'Is it ok if I take less than 12 credit hours?', and 'Which programming courses am I required to take?'. At the bottom of the page, there is a 'Send Msg' button and a footer with the CNEEL logo, the word 'Webmaster', and the University of Florida logo.

Figure B-3. Main webpage for Albert showing second FAQ page.

1. I want to choose my courses for the next semester.
2. What courses did I select for next semester?
3. What courses do you recommend I take?
4. I want to teach you a course name.
5. When is E Circuits taught next semester?
6. Where is EEL3135 given next spring?
7. Who will teach EEL 4201L?
8. Will Dr Harris teach anything next semester?
9. How many credits does Micro P have?
10. What is the prereq of Controls?
11. What is the course number of DIC?
12. Can I take Nanodevices without the prerequisite?
13. How many gen ed courses do I have to take?
14. How many critical tracking courses are required for my major?
15. How can I find out my catalog year?
16. Where can I see my curriculum?
17. Where can I find courses' syllabi?
18. How do I find my upper division GPA?
19. Is it ok if I take less than 12 credit hours?
20. Which programming courses am I required to take?
21. Which courses have a lab?
22. I was thinking about taking two lab classes. Is that a good idea?
23. How many times can I repeat a course?
24. How many drops can I have?
25. What is the college of engineering drop rule?
26. I've used 2 lower division drops and I'm failing Calc 3. Can I get a third drop?
27. My tracking GPA is below a 2.0. Am I kicked out of the major?
28. Which courses do I need a C+ to pass?
29. I made a C- in a major course. Do I have to retake it?
30. What will happen if i drop below 12 credits?
31. What is the max number of hours I can take in a semester?
32. What is a college breadth elective?
33. (For EEs) What are specialization courses?
34. (For CEEs) What major courses must I take?
35. What is an independent study?
36. What is IPPD?
37. How do I qualify for the Dean's List?
38. What GPA do I need to have to complete an honors thesis?
39. Can undergrads take 6000 level courses?
40. What do I need to do if I'm interested in grad school?
41. Can I take online courses?
42. What classes are taught through EDGE?
43. Can I substitute a physics class for electromagnetic fields?
44. How many technical elective credits do I have to take?
45. Can I take technical electives outside of the department?
46. Show me the titles of all EE courses
47. Show me the list of electives for EE.
48. Can you override any course quota?
49. Show me the UF map please.
50. What tutoring services are available on campus?
51. How can I get my holds removed?
52. How do I drop a course?
53. When is the deadline to drop a class?
54. What student organizations do we have in ECE?
55. Who is my advisor?
56. How do I apply for ECE Scholarships?
57. How do I apply for a minor?
58. How can I find an internship?
59. How do I apply to be a transient student?
60. How can I start the master's in ECE?
61. How can I leave you feedback?

Figure B-4. Full list of questions in the second FAQ webpage.

Department of Electrical & Computer Engineering

Natural Language Advising System

UNIVERSITY of FLORIDA
The Foundation for The Gator Nation

Logged in as ed WATCH THE VIDEO TUTORIAL ON YouTube

Albert Good to see you again ed.

ed I want to update my degree audit.

Albert Do you want to update your grades in my database? (yes | no)

ed yes

Albert I opened the page for you on the right were you must enter your grades and courses currently enrolled in.

Let me know if you have any questions and when you receive the submission confirmation message.

Type your message Send Msg

Paste your previous input

UF-ECE BSEE degree program

Welcome Ed

Enter your Catalog Year: (enter 4-digit year)

For each course below, choose appropriately a **Grade**, **Enrolled** or **Not Taken**. Select the **Dropped** checkbox for each course you have ever dropped, even if you already passed it.

For Specialization and Technical electives, you **must** write down the course code in the box provided.

For non-EE electives you may **optionally** write the course name for your convenience.

Students who transferred from another institution may enter equivalent grades for transferred credits.

Yes! Your selections are saved in your account for future visits.

Press **Submit** at the bottom of this page to send your results to our database.

Speech Recognition available with Google Chrome version 25+.

C-N-E-L Webmaster UF UNIVERSITY of FLORIDA

Figure B-5. Main webpage for Albert showing page one of CoPA.

Department of Electrical & Computer Engineering

Natural Language Advising System

UNIVERSITY of FLORIDA
The Foundation for The Gator Nation

Logged in as ed WATCH THE VIDEO TUTORIAL ON YouTube

Albert → Ok ed, I have a list of courses you could take next semester, assuming you pass those enrolled in.

After you submit your courses through that webpage, you can then email the EE advisor at undergradadvising@ece.ufl.edu to have your hold removed.

You can repeat this process as many times as you want. Your advisor will only read your submission when you send that email with your **Username and UF ID number**. Keep your password secret.

Do you want to choose your courses for the next semester and submit the list to your advisor?

ed → yes

Albert → I opened the page for you on the right.

Let me know if you have any questions.

Type your message Send Msg

Paste your previous input

Are you a Graduation Candidate next semester? Check here if yes:

Plan of Study

EEE 3308C Electronic Circuits 1

EEL 4924C Electrical and Computer Engineering Design 2

You need 1 Specialization elective

You need 3 Technical electives

EE electives (specialization courses noted with *)

EEL4201L Electric Energy Conversion Lab *

EEL4514 Communication Systems & Components *

EEL4657 Linear Control Systems *

EEL4657L Linear Control Systems Laboratory * --> **Needs coreq EEL4657**

EEE4331 Microelectronic Fabrication Technologies -- Not offered next semester

EEE4373 Radio-Freq Electron -- Not offered next semester

Speech Recognition available with Google Chrome version 25+.

C-N-E-L Webmaster UF UNIVERSITY of FLORIDA

Figure B-6. Main webpage for Albert showing an example page two of CoPA.

ECE Department of Electrical & Computer Engineering
UF UNIVERSITY of FLORIDA
The Foundation for The Greater Future

Natural Language Advising System

Login to connect to the Advising System
WATCH THE TUTORIAL ON YouTube

New Users - to **register** an account check here

Username:

Password:

Welcome to the ECE-UF Natural Language Advising System

Register or login by entering your preferred username & password. **Do not identify yourself!** Registered users have their data saved for ensuing sessions.

Your participation is anonymous. We do **not** have access to your UF personal records. Be careful when registering a password. To ensure anonymity, there is no password recovery method.

To logout at any time, just say Goodbye.

Type your message

Ask Albert, our Natural Language Advisor, the same questions you would ask your advisor, including

I want to choose my courses for the next semester.
 What courses do you recommend I take?
 When is Circuits 2 taught next semester?
 Where is EEL 3112 given next spring?
 What will Dr. Gugel teach?
 Who will teach EEE3308C?

Start your course selection process with

I want to update my degree audit.
 or
 I want to enter my grades.

Teach Albert your preferred name for a course:
 I want to teach you a course name.

Type **Help** at any time for more assistance.

Please leave feedback about this experimental feature. Is it accurate? Thanks!

Who is teaching circuits in fall

Webmaster

Figure B-7. Main webpage for Albert showing an example of the speech API.

APPENDIX C SAMPLES FROM USER LOG FILES

This appendix shows excerpts from the log files of six users, recorded during the third experimental phase. User statements are in bold font weight and their syntactic errors are preserved for authenticity. Italic type represents actions. Responses by the system are curtailed for clarity. Some html code is included to illustrate how the responses were presented, specifically `` for bold font and `
` for new lines.

User 1 Welcome to the University of Florida Electrical and Computer Engineering Department's Natural Language Advising System 1.02 Caiman For help at any time type in "Help" and read the example questions. Please be advised that all curriculum information is for ECE students. `
` I'll remember our conversation when you return. `
` `
` `
` Hello Freezing7, my name is Albert. To better serve you please tell me, are you an `EE` or a `CEE` major?

What courses do you recommend I take? ==> It seems I don't recognize your degree program. If you are in the BSEE program please enter `EE`. If you are in the BSCE "Computer Engineering - Hardware" program please enter `CEE`. If neither applies to you please answer `None`.

EE ==> Very well Freezing7, you are an Electrical Engineering student. What can I do for you?

What courses do you recommend I take? ==> I will tell you which courses you could take next semester after you enter your grades. Do you want to enter your grades into the database? (yes | no)

yes ==> *No response. The first frame of CoPA is launched at this time.*

confirm ==> Do you want to choose your courses for the next semester and submit the list to your advisor?

yees ==> To better understand you I assumed you meant: yes `
` Instead of: yees

The second frame of CoPA is launched at this time.

drop ==> Drop? I can't tell what you're saying with a one word sentence. Look at the example questions on the right of this webpage for ideas of dialog we can have. `

` All I can say about the word drop is - `
` Noun drop: the act of dropping something. Verb drop: the act of dropping something.

can i drop a class ==> So you want to drop a course? Are you in upper division or lower division? (You're in lower division if you haven't passed 60 credits yet.)

lower ==> Students are given TWO drops prior to completing 60 credit hours. Also

know about the college of engineering drop rule; The College of Engineering ...

User 2 OK F9788, you are an Electrical Engineering student. What can I do for you?

What courses do you recommend I take? ==> I will tell you which courses you could take next semester after you enter your grades. Do you want to enter your grades into the database? (yes | no)

y ==> No response. The first frame of CoPA is launched at this time.

confirm ==> Do you want to choose your courses for the next semester and submit the list to your advisor?

what courses do you recommend I take? ==> Assuming your info is up to date, these are the courses you could take next semester.

 EEL 3472
Electromagnetic Fields and Applications 1

 College breadth elective ...

User 3 Welcome back willwrdrn

I want to drop a course ==> So you want to drop a course? Are you in upper division or lower division? (You're in lower division if you haven't passed 60 credits yet.)

upper ==> Students are given TWO drops after completing 60 credit hours. Any unused lower division drop is lost forever. ...

what is the deadline for dropping a course ==> I'll tell you this, under extreme circumstances that are beyond your control, it is possible to drop a course after ...

am I able to drop a course via ISIS? ==> So you want to drop a course? Are you in upper division or lower division? (You're in lower division if you haven't passed 60 credits yet.)

upper ==> Students are given TWO drops after completing 60 credit hours. ...

okay ==> All right.

so how do I drop a course? ==> You drop classes via ISIS. If the add/drop period has ended you can still do it, but you need to see your advisor so he can complete the process.

alright! thank you! That's all the information I need today ==> Well, I am learning every day. What can I do for you?

logout ==> Before you go, please share any feedback or suggestions to help you better next time.

Perhaps it was just that my queries were better and fewer this time, but the system seems to have improved. Good job! I hope it gets to access to more information soon so that it can answer more questions instead of some variation of I don't know ==> All right, thanks!
 Later, Gator!

User 4 Hi Novael!

show my schedule ==> You want me to show you the courses you chose for next semester? OK.

 These are your courses:
 ...

programming classes ==> Electrical Engineering students must take one programming course in Java, C or C++ for their major. The following courses are recommended: COP3502, COP3504, EEL3834 or COP2271.

digital logic prerequisite ==> To better understand you I assumed you meant: Digital_Logic prerequisite
 Instead of: digital logic prerequisite

 I recognize Digital Logic is a course in our department, but could you be more specific about your request?
 Maybe the example questions on the right of this webpage can help.
 I certainly can tell you the number for Digital Logic, it's EEL 3701C.

prerequisite for eel 3701c ==> The prerequisite for EEL 3701C is a Programming Course.

is eel3834 being taught next semester? ==> The course EEL 3834 is not offered in Spring 2014 .

what classes can i take ==> Assuming your info is up to date, these are the courses you could take next semester.

 Professional Communication ...

circuits 2 ==> I recognize CIRCUITS 2 is a course in our department, but could you be more specific about your request?
 Maybe the example questions ...

is eel3701c being offered spring ==> For Spring 2014 we have EEL 3701C is ...

Who will teach EEE3308C? ==> For Spring 2014 we have
 EEE 3308C by Yoon,Yong Kyu

digital logic prereq ==> For Spring 2014 we have
 EEL 3701C by Gugel,Karl S

digital logic prerequisite ==> For Spring 2014 we have EEL 3701C by Gugel,Karl S

eel3701c any sections available ==> For Spring 2014 we have
 EEL 3701C by Gugel,Karl S

Is it ok if I take less than 12 credit hours? ==> UF policy states that the minimum full-time load for undergraduate students is 12 credits. The minimum full-time load ...

How many critical tracking courses are required for my major? ==> You need to pass 8 critical tracking courses. You need to pass all the listed tracking courses ...

What courses did I select for next semester? ==> These are your courses:
 ...

I was thinking about taking two lab classes. Is that a good idea? ==> These are the courses with labs included:
 EEL3111C
 EEL3135
 EEL3701C ...

Where can I see my curriculum? ==> Here is your curriculum. [EE Curriculum](#)

User 5
Log 1

Hello gil3167, my name is Albert. To better serve you please tell me, are you an EE or a CEE major?

EE ==> OK gil3167, you are an Electrical Engineering student. What can I do for you?

I need to have my hold lifted ==> I can't know if your hold was removed because I am not authorized to access ISIS. If your academic hold was removed you should have gotten a confirmation email by the ECE advisor.

I want to choose my courses for spring ==> You can do this if you have your grades up to date in my system. To update your degree audit, just let me know ...

yes ==> *No response. The first frame of CoPA is launched at this time.*

what is catalog year? ==> Students graduate under the catalog in effect ...

is ANT 2000 a humanities ==> Could you rephrase that please?

ANT 2000 ==> I'm not following.

are my grades analyzed? ==> Ok gil3167, I have a list of courses you could take next semester, assuming you pass those enrolled in. After you submit your courses through that webpage, you can then email the EE advisor at ...
 Do you want to choose your courses for the next semester and submit the list to your advisor?

yes ==> *No response. The second frame of CoPA is launched at this time.*

i'm interested in a minor ==> You can find the general information regarding ...

computer engineering minor ==> You can find the general information regarding ...

bye ==> Before you go, please share any feedback or suggestions to help you better next time.

User 5
Log 2

=> Hi again gil3167

how many credits is digital design ==> EEL 4712C has 4 credits.

how many credits is electromagnetic fields ==> EEL 3472 has 3 credits.

does eel 3472 have a lab? ==> These are the courses with labs included:
 ...

is engineering statistics hard ==> I only know information for courses of our department.

is digital design hard ==> A good course is where you will learn the important ...

who is teaching engineering statistics ==> Who teaches engineering? I can't tell...

what do you recommend that i take next ==> I recommend you study hard and ...

what courses should i take ==> Assuming your info is up to date, these are ...

what are the college breadth electives ==> A College Breadth Elective can be ...

User 5
Log 3 => Good to see you again gil3167 .

choose classes ==> I didn't quite understand your statement. Regarding EE ...

what courses do you recommend i take? ==> Assuming your info is up to date ...

i would like to register ==> You register for courses via ISIS.
 You can ask ...

removing my hold ==> As you may have read in your advisor's email, I will help ...
Do you want to enter your grades into the database? (yes | no)

yes ==> *No response. The first frame of CoPA is launched at this time.*

done ==> ... Do you want to choose your courses for the next semester and submit the list to your advisor?

yes ==> *No response. The second frame of CoPA is launched at this time.*

User 6 **enter my grades** ==> ... Do you want to enter your grades into the database?

yes ==> *No response. The first frame of CoPA is launched at this time.*

confirmed my grades ==> Excuse me aradulescu1, I'm not sure what you mean.

Student submits first CoPA page.

confirmed my grades ==> ... Do you want to choose your courses for the next semester and submit the list to your advisor?

no ==> No problem. What can I do for you?

I want to choose my courses for the next semester ==> *No response. The second frame of CoPA is launched at this time.*

What courses do you recommend. ==> Assuming your info is up to date, ...

I'm done. ==> Done with what?

Talking to you. ==> Is that so? I need to learn about that.

Log out. ==> Before you go, please share any feedback or suggestions to help you better next time.

I did. ==> All right, thanks!
 Later, Gator!

APPENDIX D ADVISOR SURVEY QUOTES

Conversational Advising System Assessment

11.26.13

Assessor: Shannon M. Chillingworth, ECE Student Services Supervisor

1. Overall, how would you rate the academic advising system?

I think ALBERT is a great tool to be used in conjunction with traditional in-person advising. ALBERT helped to reduce the time needed to lift semesterly advising holds and allowed me to focus more on one-on-one student interactions.

2. What positive features can you highlight?

ALBERT's ability to download the schedule of courses from ISIS was extremely helpful. This feature helped to narrow down the available course selection choices for the student body.

3. What negative features can you highlight?

ALBERT had a hard time handling course selections for non-traditional students (i.e., students with minors, transfer students). However, the addition of the write-in field that allowed students to communicate their non-conventional course selections to advising staff helped to alleviate a lot of these issues.

4. What was your perception of student attitudes toward the advising system?

Most students were extremely reluctant to use the system at first, but I do not think that this reluctance had anything to do with ALBERT. Once students actually used the system, I think most had a positive experience.

5. Comment on how effective you think the advising system is at helping students

In principle, it is very helpful to students. However, most students are reluctant to use the system because they are used to a one-on-one personal interaction with an academic advisor. Once student's got used to using this system, they would consider it helpful.

6. Comment on how effective you think the advising system is for your work.

It is extremely effective for the course selection process. It helped to reduce the time required for advisors to lift advising holds.

7. In what ways might this advising system be improved?

Incorporating more flexibility into the course selection process would be a good way to improve the system. Adding an upload feature that would allow

students to upload their transcript or degree audit would also be extremely helpful.

Final Comments/Suggestions/Recommendations:

Overall, I was very pleased with my interactions with ALBERT. I would like for it to be used in future semesters, but more would need to be done to convince students of its utility.

LIST OF REFERENCES

- [1] V. N. Gordon, W. R. Habley, T. J. Grites and National Academic Advising Association. *Academic Advising: A Comprehensive Handbook*, 2nd ed. San Francisco: Jossey-Bass, 2008.
- [2] S.H. Frost, "Historical and Philosophical Foundations for Academic Advising", in *Academic Advising: A Comprehensive Handbook*, V. Gordon, W. R. Habley, T. J. Grites and National Academic Advising Association, Eds. San Francisco: Jossey-Bass, pp. 3-16, 2008.
- [3] K. Soria, "Advising Satisfaction: Implications for First-Year Students' Sense of Belonging and Student Retention", in *The Mentor: An Academic Advising J.*, Oct 2012. [Online]. Available: <http://dus.psu.edu/mentor/2012/10/advising-satisfaction/> Retrieved Feb 2013.
- [4] M. J. Leonard, "Advising Delivery: Using Technology", in *Academic Advising: A Comprehensive Handbook*, V. Gordon, W. R. Habley, T. J. Grites and National Academic Advising Association, Eds. San Francisco: Jossey-Bass, pp. 292-306, 2008.
- [5] L. Waldner, D. McDaniel, T. Esteves and T Anderson, "The eQuad: A Next-Generation eAdvising Tool to Build Community and Retain Students", in *The Mentor: An Academic Advising J.*, Oct 2012. [Online]. Available: <http://dus.psu.edu/mentor/2012/10/equad-eadvising-tool-build-community-retain-students/> Retrieved Feb 2013.
- [6] National Academic Advising Association, Advising Technology Innovation Awards. [Online]. Available: <https://www.nacada.ksu.edu/programs/Awards/EPublication.htm> Retrieved Feb 2013.
- [7] H. S. Hart, R. B. Hussey, M. J. Leonard, J. Levin and S. M. Winck, "eLion: Penn State's Comprehensive Web-Based Academic Advising System", in *The Mentor: An Academic Advising Journal*. [Online]. Available: <http://dus.psu.edu/mentor/bookstore/elion-advising-system/> Retrieved Feb 2013.
- [8] University Advising Center, The University of Texas at Arlington, Arlington, TX. [Online]. Available: <http://www.uta.edu/universitycollege/current/academic-planning/uac/index.php> Retrieved Feb 2013.
- [9] Academic Advising and Career Center, eAdvising, University of Michigan-Flint, Flint, Michigan. [Online]. Available: <http://www.umflint.edu/advising/eadvising.htm> Retrieved Feb 2013.
- [10] E. R., White and M. J. Leonard, Faculty Advising and Technology, in *Faculty advising examined: Enhancing the potential of college faculty as advisors*, G. L. Kramer, Ed. Bolton, MA: Anker Publishing Company, 2003.

- [11] R. M. Siegfried, A. M. Wittenstein and T. Sharma, "An automated advising system for course selection and scheduling", J. Comput. Sci. Coll., vol. 18, no. 3, pp.17-25, Feb 2003.
- [12] F. Albalooshi and S. Shatnawi, "HE-Advisor: A multidisciplinary web-based higher education advisory system", Global J. Computer Sci. and Tech., vol. 10, no. 7, pp. 37-49, Sept. 2010.
- [13] A. N. Nambiar and A. K. Dutta, "Expert System for Student Advising using JESS", Int. Conf. Educational and Inform. Tech., vol. 1, pp. V1-31 – V1-315, Sept. 2010.
- [14] T. Feghali, I. Zbib and S. Hallal, "A Web-based Decision Support Tool for Academic Advising", J. Educational Technology & Society, vol. 14, no. 1, pp. 82–94, 2011.
- [15] E. Onyeka, D. Olawande and A. Charles, "CAES: A model of an RBR-CBR course advisory expert system," Int. Conf. Inform. Soc., pp.37-42, June 2010.
- [16] B. C. McMahan and R. A. Bates, "An Automatic Dialog System for Student Advising", in J. Undergraduate Research, Minnesota State University, Mankato, 2010.
- [17] C. M. Leung, E. Y. M. Tsang, F. S. S. Lam and D. P. C. Wai, "Intelligent Counseling System: A 24 x 7 Academic Advisor," EDUCAUSE Quart. 33, no. 4, 2010. [Online]. Available: <http://www.educause.edu/edUCAUSE+Quarterly/edUCAUSEQuarterlymagazineVolume/intelligentCounselingSystema24/219101> Retrieved Feb 2013.
- [18] D. Jurasky and J. H. Martin, Speech and Language Processing An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition, 2nd Ed, New Jersey: Pearson Education, 2008.
- [19] K. Bach, "Ambiguity", in Routledge Encyclopedia of Philosophy Online. E. Craig General Ed., Taylor & Francis Group, 2012. [Online]. Available: <http://www.rep.routledge.com/Routledge> Retrieved Feb 2013.
- [20] Princeton University, "About WordNet", Princeton University, 2010. [Online]. Available: <http://wordnet.princeton.edu> Retrieved Feb 2013.
- [21] R. Navigli, "Word Sense Disambiguation: A Survey", ACM Comput. Surv., vol. 41 no. 2, pp. 1–69, 2009.
- [22] Quarteroni, S. and Manandhar, S. "Designing an interactive open-domain question answering system." J. Natural Language Eng, vol. 15, no. 1, 73-95, 2008.
- [23] L. Androutsopoulos, G.D. Ritchie and P. Thanisch, "Natural Language Interfaces to Databases - An Introduction", J. Natural Language Eng., vol. 1, pp. 29–81, 1995.

- [24] T. Harley, *The Psychology of Language: From Data to Theory*, 3rd Ed., Hove: Psychology Press, 2008.
- [25] E. Klepousniotou, "The Processing of Lexical Ambiguity: Homonymy and Polysemy in the Mental Lexicon", in *Brain and Language*, vol. 81, pp. 205–223, Elsevier Science, USA, 2002.
- [26] T. Klüwer, "From Chatbots to Dialog Systems", in *Conversational Agents and Natural Language Interaction: Techniques and Effective Practices*, D. Perez-Marin and I. Pascual-Nieto, Eds., USA: IGI Global, 2011.
- [27] E. Reiter and R. Dale, *Building Natural Language Generation Systems*, Cambridge University Press, 2000.
- [28] M.A. Walker, D.J. Litman, C.A. Kamm and A. Abella, "PARADISE: A framework for evaluating spoken dialogue agents," Proc. 8th Conf. European Chapter Assoc. Computational Linguistics, pp. 271–280, 1997.
- [29] V. Hung, M. Elvir, A. Gonzalez and R. Demara, "Towards a method for evaluating naturalness in conversational dialog systems," Syst. Manage. and Cybernetics IEEE Int. Conf., pp.1236–1241, 2009.
- [30] D. Oelke, Ming Hao, C. Rohrdantz, D.A. Keim, U. Dayal, L. Haug and H. Janetzko, "Visual opinion analysis of customer feedback data," *Visual Analytics Science and Technology. VAST 2009. IEEE Symposium on*, pp. 187-194, Oct. 2009.
- [31] S. Harnad and P. Scherzer, "First, scale up to the robotic Turing test, then worry about feeling", *Artif. Intell. Med.*, vol. 44, no. 2, pp. 83-89, Oct. 2008.
- [32] The Loebner Prize in Artificial Intelligence, [Online]. Available: <http://www.loebner.net/Prize/loebner-prize.html> Retrieved Feb 2013.
- [33] R.S. Wallace, Alice A.I. Foundation, [Online]. Available: <http://www.alicebot.org> Retrieved Feb 2013.
- [34] B. Wilcox, ChatScript project, [Online]. Available: <http://chatscript.sourceforge.net> Retrieved Feb 2013.
- [35] Pandorabots, the open-source software robot hosting service, [Online]. Available: <http://pandorabots.com/botmaster/en/home> Retrieved Feb 2013.
- [36] N. Bush and N. Dubrovsky, aitools.org, Open-Source Community, [Online]. Available: <http://aitools.org> Retrieved Feb 2013.
- [37] F.A. Mikic, J.C. Burguillo, M. Llamas, D.A. Rodriguez and E. Rodriguez, "CHARLIE: An AIML-based chatterbot which works as an interface among INES and humans," *EAAEIE Annual Conference*, pp. 1-6, 2009.

- [38] S. Bird, E. Klein and E. Loper, *Natural Language Processing with Python*, O'Reilly Media, 2009. [Online]. Available: <http://nltk.org/book/> Retrieved Feb 2013.
- [39] P. Ljunglöf, "trindikit.py: An open-source Python library for developing ISU-based dialogue systems", Proc. IWSDS, Kloster Irsee, Germany, 2009
- [40] K.E. Boyer, E.Y. Ha, R. Phillips, M.D. Wallis, M.A. Vouk and J.C. Lester, "Dialogue Act Modeling in a Complex Task-oriented Domain", *Proc 11th Annu. Meeting Special Interest Group Discourse and Dialogue, Association Computational Linguistics*, pp. 297-305, 2010.
- [41] S. Quarteroni and S. Manandhar, "Designing an interactive open-domain question answering system" *J. Natural Language Eng.*, vol. 15, no. 1, pp. 73-95, Jan 2008.
- [42] D. C. Appleby, "Advising as Teaching and Learning", in *Academic Advising: A Comprehensive Handbook*, V. Gordon, W. R. Habley, T. J. Grites and National Academic Advising Association, Eds. San Francisco: Jossey-Bass, pp. 85-102, 2008.
- [43] T. Brown, "Critical Concepts in Advisor Training and Development", in *Academic Advising: A Comprehensive Handbook*, V. Gordon, W. R. Habley, T. J. Grites and National Academic Advising Association, Eds. San Francisco: Jossey-Bass, pp. 309-322, 2008.
- [44] P. L. Hagen and P. Jordan, "Theoretical Foundations of Academic Advising", in *Academic Advising: A Comprehensive Handbook*, V. Gordon, W. R. Habley, T. J. Grites and National Academic Advising Association, Eds. San Francisco: Jossey-Bass, pp. 17-35, 2008.
- [45] A. Kerly, R. Ellis and S. Bull, "Conversational Agents in E-Learning", *Applicat. and Innovations in Intelligent Systems XVI*, pp. 169-182, 2009.
- [46] S. Gandhe, N. Whitman, D. Traum and R. Artstein, "An integrated authoring tool for tactical questioning dialogue systems", *Proc. 6th IJCAI Workshop on Knowledge and Reasoning in Practical Dialogue Systems*, Association for the Advancement of Artificial Intelligence, 2009.
- [47] D. Mori, R. Berta, A. De Gloria, V. Fiore and L. Magnani, "An Easy to Author Dialogue Management System for Serious Games", *J. Comput. Cult. Herit.*, vol. 6, no. 2, pp. 10:1-10:15, May 2013.
- [48] Contributors to the Web Speech API Specification, Web Speech API, Speech API Community Group, 2012. [Online]. Available: <https://dvcs.w3.org/hg/speech-api/raw-file/tip/speechapi.html> Retrieved Feb 2013.
- [49] P. Norvig. *How to Write a Spelling Corrector*. [Online]. Available: <http://norvig.com/spell-correct.html> Retrieved Feb 2013.

- [50] Council for the Advancement of Standards in Higher Education, "The Role of Academic Advising Programs", Washington DC, 2011.
- [51] A. Hulth, "Combining Machine Learning and Natural Language Processing for Automatic Keyword Extraction" (Ph.D. thesis), Dept. Comput. and Syst. Sci., Stockholm University, 2004.
- [52] F. Bellotti, R. Berta, A. De Gloria and E. Lavagnino, "Towards a conversational agent architecture to favor knowledge discovery in serious games", Proc 8th Int. Conf. Advances Comput. Entertainment Technology, T Romão *et al.*, Eds. ACM, New York, art. 17, ch. 2, 2011.
- [53] D. de Castro Reis, F. Goldstein and F. Quintao, "Extracting Unambiguous Keywords from Microposts Using Web and Query Logs Data", *2nd Workshop Making Sense of Microposts*, 21st Int. Conf. World Wide Web, 2012.
- [54] D. X. Wang, X. Gao, and P. Andreae, "Automatic keyword extraction from single-sentence natural language queries", Proc 12th Pacific Rim Int. Conf. on Trends Artif. Intell., P. Anthony, M. Ishizuka and D. Lukose, Eds. Springer-Verlag, Berlin, Heidelberg, pp. 637-648, 2012.
- [55] L. Richardson, Beautiful Soup [Online]. Available: <http://www.crummy.com/software/BeautifulSoup/> Retrieved Feb 2013.
- [56] R. A. Baeza-Yates and B. Ribeiro-Neto, *Modern Information Retrieval*, Addison-Wesley Longman Publishing, Boston, MA, 1999.
- [57] J. A. Gubner. Probability and Random Processes for Electrical and Computer Engineers, Cambridge University Press, Cambridge, United Kingdom, 2006.
- [58] A. Olney, M. Louwerse, E. Matthews, J. Marineau, H. Hite-Mitchell, and A. Graesser, "Utterance classification in AutoTutor", HLT-NAACL Building Educational Applicat. using NLP, *Assoc for Computational Linguistics*, vol. 2, 1-8, USA, 2003.
- [59] W. Ward, R. Cole, D. Bolaños, C. Buchenroth-Martin, E. Svirsky, S. Van Vuuren, T. Weston, J. Zheng and L. Becker, "My science tutor: A conversational multimedia virtual tutor for elementary school science", *ACM Trans Speech and Language Process.*, TSLP, vol. 7, no. 4, 2011.
- [60] E. Sneider, "Automated FAQ Answering with Question-Specific Knowledge Representation for Web Self-Service", *Proc 2nd Int Conf Human Syst Interaction*, IEEE, May 21-23, Catania, Italy, pp. 298-305, 2009
- [61] M. Minock, "C-Phrase: A system for building robust natural language interfaces to databases", *Data & Knowledge Engineering*, vol. 69 no. 3, pp. 290-302, 2010.
- [62] M. Olvera-Lobo and J. Gutiérrez-Artacho, "Open-vs. Restricted-Domain QA Systems in the Biomedical Field," *J. Inf. Sci.*, vol. 37, no. 2, pp. 152-162, 2011.

- [63] V. Lopez, C. Unger, P. Cimiano and E. Motta, "Evaluating question answering over linked data", *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 21, pp. 3-13, 2013.

BIOGRAPHICAL SKETCH

Edward is a Professional Engineer and a Doctor of Philosophy in electrical and computer engineering, whose research area comprises natural language processing, software application development, and computer science and engineering education.

Edward obtained a Bachelor of Science in electrical engineering and a Master of Science in electrical engineering, in the areas of digital electronics and digital signal processing, from the University of Puerto Rico – Mayagüez Campus. In spring 2014, he obtained the PhD and a Master of Science in electrical and computer engineering from the University of Florida. For his academic achievements and passion, he obtained a Google scholarship for the 2013 – 2014 academic year.