Varderes Barsegyan
016163470

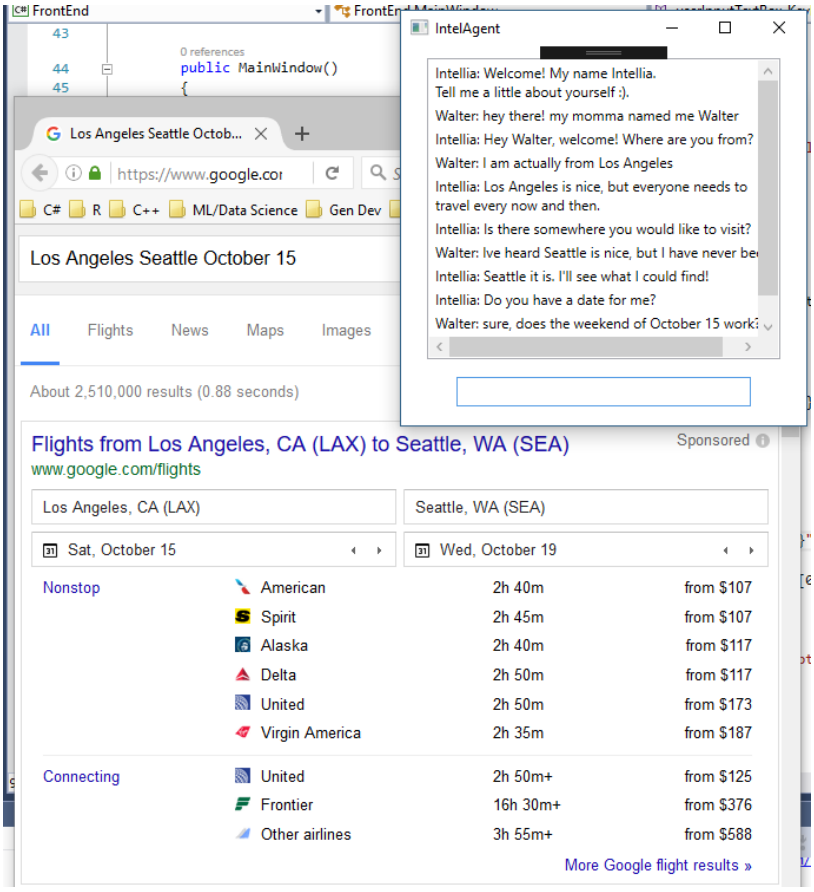
Task 1: IntelAgent

The source code for my conversational agent can be found here: https://github.com/Vardominator/CSULBProjects/tree/master/CECS451_ArtificialIntelligence/ConversationalAgent

I used C# on the frontend to make a simple chat system and R on the backend to run natural language processing functions from the OpenNLP library.

For this project, I took the approach of building a smart travel agent called *Intellia*. Intellia figures out your name, where you currently live and where you would like to travel on what date. I initially thought it would nice to use a travel website API, such as Expedia's, but for this first iteration it simply opens up a webpage and does a google search with the information that it has gathered.

Eventually I would like to make the chatting more advanced and fault tolerant. For example, it recognizes common names such as Mark, Bob, and Jane, but it does not recognize foreign names such as Varderes, Tankelevich, and Armen. I want to also make the system more versatile so that it can be an overall travel platform: the chat bot can buy tickets for you, it can give you suggestions, and it can also help you find a place to stay.



As you can see, I am having a casual conversation with Intellia. She is able to extract my name regardless of how I say it. I could have just as well said, "asldkfjoweiursdf Walter apsoidflkjas;df". She then finds out where I am from and where I want to travel. Finally, she asks for a date. Once that is provided, it runs a Google search and opens a browser with the results.

Varderes Barsegyan
016163470

Task 2: The role of LISP in conversational agents

Although I have used R to run natural language algorithms, it is important to discuss the role of LISP in the development of advanced conversational agents. However, before going into LISP, it is even more important to discuss conversational agents in general, the motivations for the technology, and their current state in the world.

Conversational agents (CA) were developed to blur the line between human and computer communication. Since humans mainly communicate verbally (vocal or textual), a truly intelligent machine should be able to speak to humans seamlessly. Manifestations of this are CAs, which provide a way for humans to speak to computers, particularly using written text. The underlying mechanisms involved are algorithms developed in the realm of *natural language understanding*, which attempts to extract entities, the connections between those entities, and the overall context of language. The best examples of this are Microsoft's Cortana, Google Now, and Apple's Siri. These technologies are nowhere near the level required for generally intelligent machines, but they do show the importance and need of high-end human-computer interaction.

Due to the need for natural language understanding in cutting-edge technology, many developers have created open-source libraries available to anyone interested in tinkering with sophisticated language processing algorithms. Some of these include Natural Language Toolkit, GATE, and OpenNLP (which I have used in my project). On the proprietary end you have platforms such as Microsoft Bot Framework and the Google Cloud Natural Language API. These tools are used in enterprise software all over the world. It is important to consider this dynamic; you have free tools available to everyone and proprietary software available to enterprise developers. Clearly, artificial intelligence in general and natural language processing in particular are still undergoing an evolution. It will be interesting to see how this all plays out.

LISP is the godfather programming languages for artificial intelligence, and it set the foundation for AI and its connection to computer science. Since LISP is a symbolic programming language and problems in AI have often been approached symbolically, it became a staple prototyping language. Due to the nature of symbolism in language, LISP became a powerful tool for solving difficult AI problems related to language and translation between languages. Furthermore, LISP being a functional language, works well with the highly recursive nature of language processing: words make up phrases, phrases make up sentences, and sentences make up concepts. Finally, LISP is extremely extensible (part of the reason by the R language became so popular in the last few years); it has been extended to many applications in various industries for advanced symbolic processing and extremely complex algorithms.