The amount of information students must memorize, internalize, and then apply in a class over the course of a semester can be daunting for the student. Having to keep track of information taught up to fifteen weeks prior is not an easy task, yet students are required to do it for several different classes in a single semester. Class Extractor attempts to remedy this situation. Class Extractor is a powerful desktop application, written in Objective-C and Python, that uses students’ notes, audio recordings of professors, and PowerPoints of lectures (if available) to extract the most pertinent information, rank topics in terms of importance, and give the student an overview of the lecture. For students who might have difficulty identifying the most salient topics of a class, this application can be especially useful as it prioritizes and organizes information in an easy-to-understand manner. Importantly, Class Extractor does neither the studying for the student nor allows the student to skip class. Student-typed notes are necessary for analysis, and it is up to the student to review and apply the extracted information. This information will ease the studying process and discretely show the student a break down of the class in an organized form through the use of graphs and descriptors that point to certain topics.

The intellectual merit of Class Extractor is a new application of natural language processing (NLP), using both text and audio to glean meaning. Through this new application, the program will attempt to increase the accuracy of topic identification by combining both forms of input. Natural language processing is a rapidly growing field, so there are potentially myriad ways that the results of this program can influence how NLP is reasoned about. Class Extractor’s backend utilizes a three prong approach. The first prong handles the audio; the program records the professor’s voice during the lecture, and at the end of the lecture, sends the raw audio recording to IBM’s Watson. Watson converts the audio to text and returns a JSON file to Class Extractor with every word, the time stamp of the word, Watson’s confidence that the word is correct, and alternatives. Using summarization and topic identification algorithms, Class Extractor will create a JSON file with various topics and associated weights of importance. A second prong handles textual notes. At the end of the lecture, after the student has completed typing his or her notes, Class Extractor will use the same natural language processing algorithms as before - summarization and topic identification - while also taking other factors into account, such as titles and indentation. The third prong processes the PowerPoint file. This prong uses the same algorithms as the above two, while also including picture identification if there are pictures in the PowerPoint file. At this point, Class Extractor will calculate averages and frequencies of the various topics and their weights to produce graphs with the most important topics, their weights, and where in the three forms of notes each topic can be found for future reference. Combining these different input sources to create a larger idea is a unique application of natural language processing.

Class Extractor allows students to gain much more from their classes. It allows them to grasp the material better through more efficient and effective studying. The main benefit of this outcome is more successful students who turn into more successful members of the workforce, but there are some side benefits as well. Students who better understand their classes and perform better may be happier with lower stress levels, allowing the students to benefit more from the university experience overall (including engaging in more extracurricular activities). Furthermore, students who have issues prioritizing information can benefit from using this application as well, as Class Extractor provides the importance ranking for them. Class Extractor has the ability to make learning easier and more accessible through the use of natural language processing and careful calculations of topic weightings.