Elliot Liskin September 2015

Proposal Summary Page

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 ago is not an easy task, yet students are required to do it for several different classes in a single semester, and for several semesters over the course of a student’s time in college. If students learn and internalize less information, then they may be less successful students in future classes and possibly less successful members of the workforce after that. Class Extractor attempts to remedy this situation. Class Extractor is a powerful desktop application, written in Objective-C++, that uses notes typed by a student, 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 future studying. Class Extractor does neither the studying for the student nor allows the student to skip class; thorough notes taken by the student are necessary for analysis, and it is then 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 show where in the course notes and lectures certain topics can be found.

The intellectual merit of Class Extractor is a new application of natural language processing, using both text and audio to glean meaning. Class Extractor’s backend utilizes a three prong approach. The first prong handles the audio; the program records the professor’s voice over the course of the lecture, and at the end of the lecture, sends the raw audio recording to IBM’s Watson, which 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. Another JSON file will be created with the weights of the topics calculated from this second prong. 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. This analysis will produce a third JSON file. At this point, Class Extractor will calculate averages and frequencies of the various topics and their weights from the JSON files to produce graphs with the most important topics, their weights, and where in the three forms of notes (audio, textual, and PowerPoint) 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 benefits of this outcome, as noted above, include 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 or taking more, different, or varied classes). 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.