SOL

Senior Design Final Report

May 7, 2021

Richard Ma

Ervin Wong

Aaron Varghese

Faculty Advisor: Wendy Tang

Table of Contents

Background

Text mining is one of the many techniques used in Machine Learning (ML). It is the process of analyzing large amounts of unstructured text data (for example, in a document) using software to find patterns, topics or keywords within the data [3]. Since a lot of data is organized through text, text mining is a technique correlated to information retrieval (also known as data mining), which is then used to sort through the data sets. Text mining uses analytical strategies and models to sort through unstructured data sets and use word relationships, or the relationships between data entities, to create statistical summaries. Aided by Natural Language Processing (NLP), which is an Artificial Intelligence technique, it parses and goes through the data to find certain patterns, keywords, and behaviors to understand how users feel [2][4]. Companies utilize text mining to obtain feedback on things that need to be improved such as product reviews and customer satisfaction, or to inquire about new business proposals/marketing campaigns. An example of text mining that is famous to every kid or teenager is the genie fortune teller game Akinator[1]. Akinator prompts users by saying that it can read people's mind based on their inputs. In actuality, the game parses any keywords that you enter and searches through databases to find things/people related to those keywords. With text mining, it clarifies all data provided and allows it to handle clusters of unorganized data.

Societal Needs and Impacts

Text analysis of survey data is a useful tool that can be used to help individuals or companies/organizations (no matter how big or small) improve their product or service. To save time and money in creating or answering surveys, people would usually make surveys that are a list of questions and have the answers to those questions as numbers in a range (usually from 1 to 10), and the person taking the survey would have to choose from those numbers. Although this is convenient, it is not very helpful, since these types of surveys do not really show engineers the area the product or service needs improvement in. This can be detrimental since the engineers would then be playing a guessing game and may remove or add functionality to the product even though it would not be beneficial in the long run. Time and money would be wasted, and there is a chance that the company may lose customers to competition in the long run.

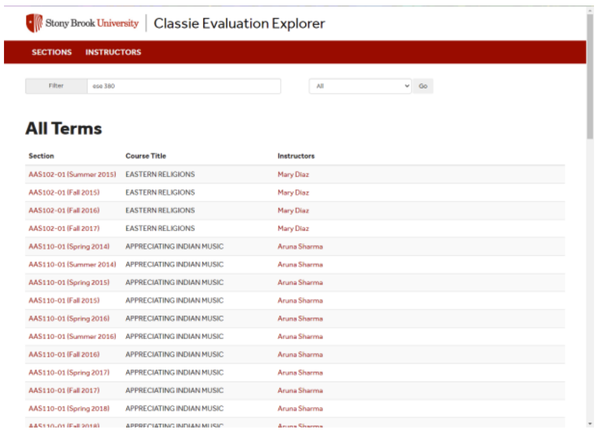
To fix this problem, there are also surveys given by companies that are open-ended and try to get to the heart of the matter by giving a customer enough space to vent or speak highly of the product. However, reading this information takes time, and the people who are reading the surveys are only human. There is a danger of skipping over lines or not taking something seriously due to bias for the product. Also, the way people interpret language is different due to the different experiences people have. This is where text analysis of survey data shines.

Some people believe that computers are very smart and capable machines, but what they often overlook is that computers always require some sort of commands. Though not completely autonomous, computers are very good at keeping track of objects, whether they be letters, numbers, or words. Depending on the instructions you give the computer, the computer can do a certain task, and this is where text analysis of survey data gets comes to play.

Due to the reasons given above as well as some others, natural language processing (or NLP) is a hot topic in the technology industry [4]. This is because NLP tries to reduce the ambiguity that occurs in human languages. People have different dialects and ways of speaking, and NLP provides many ways of interpreting human language. In other words, NLP basically breaks down a language into smaller basic building blocks, and it gives insight into how those blocks come together to have meaning. Text mining of survey data is just a single example of NLP.

Statement of Work and Test Bed for Verification

First off, to get an idea on what the project is supposed to accomplish, we will give a brief introduction to the problem we are trying to solve. If you are not familiar with Classie-Evals, it is a website that Stony Brook uses that allows students to give course evaluations for a course or courses that they were taking in a particular semester. Usually, there are course evaluations given at the end of every semester for classes that students are taking, and those course evaluations are placed onto Classie-Evals.



Stony Brook University | Classie Evaluation Explorer

SECTIONS INSTRUCTORS

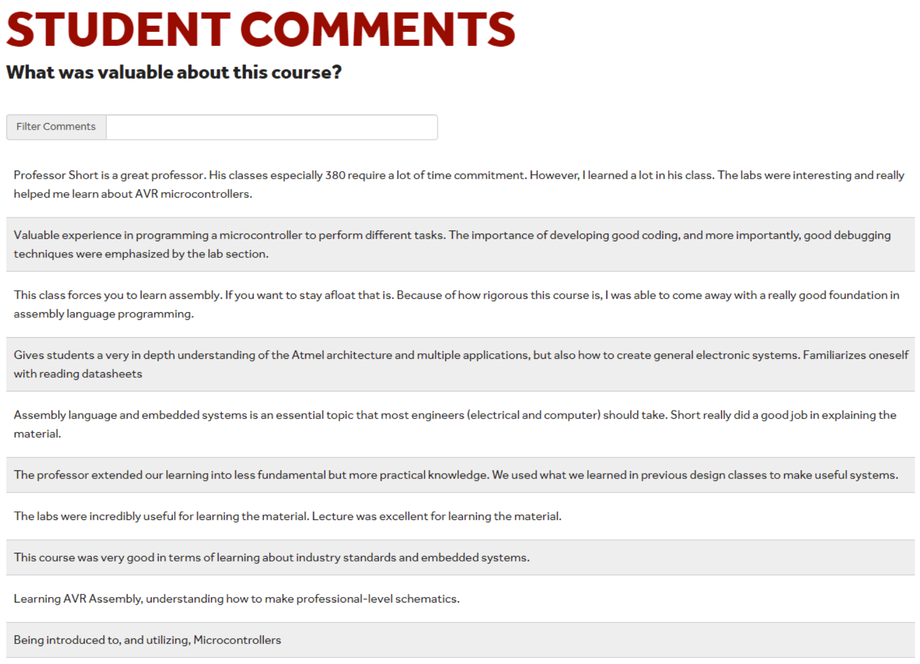
Filter: ese 380 | All | Go

All Terms

Section	Course Title	Instructors
AAS102-01 (Summer 2015)	EASTERN RELIGIONS	Mary Diaz
AAS102-01 (Fall 2015)	EASTERN RELIGIONS	Mary Diaz
AAS102-01 (Fall 2016)	EASTERN RELIGIONS	Mary Diaz
AAS102-01 (Fall 2017)	EASTERN RELIGIONS	Mary Diaz
AAS110-01 (Spring 2014)	APPRECIATING INDIAN MUSIC	Aruna Sharma
AAS110-01 (Summer 2014)	APPRECIATING INDIAN MUSIC	Aruna Sharma
AAS110-01 (Spring 2015)	APPRECIATING INDIAN MUSIC	Aruna Sharma
AAS110-01 (Fall 2015)	APPRECIATING INDIAN MUSIC	Aruna Sharma
AAS110-01 (Spring 2016)	APPRECIATING INDIAN MUSIC	Aruna Sharma
AAS110-01 (Summer 2016)	APPRECIATING INDIAN MUSIC	Aruna Sharma
AAS110-01 (Fall 2016)	APPRECIATING INDIAN MUSIC	Aruna Sharma
AAS110-01 (Spring 2017)	APPRECIATING INDIAN MUSIC	Aruna Sharma
AAS110-01 (Fall 2017)	APPRECIATING INDIAN MUSIC	Aruna Sharma
AAS110-01 (Spring 2018)	APPRECIATING INDIAN MUSIC	Aruna Sharma
AAS110-01 (Fall 2018)	APPRECIATING INDIAN MUSIC	Aruna Sharma

Figure 1: This figure shows the interface of Classie-Evals and how course data is arranged based on the professor's name and the timing of the course

This would also allow students to be able to rant about a course, praise a course, etc. Classie-Evals is a neat website, and it allows professors to better their course (or it is supposed to), and it also allows students to decide on whether they would want to steer the course based on feedback that the students from previous semesters have given.



STUDENT COMMENTS

What was valuable about this course?

Filter Comments

Professor Short is a great professor. His classes especially 380 require a lot of time commitment. However, I learned a lot in his class. The labs were interesting and really helped me learn about AVR microcontrollers.

Valuable experience in programming a microcontroller to perform different tasks. The importance of developing good coding, and more importantly, good debugging techniques were emphasized by the lab section.

This class forces you to learn assembly. If you want to stay afloat that is. Because of how rigorous this course is, I was able to come away with a really good foundation in assembly language programming.

Gives students a very in depth understanding of the Atmel architecture and multiple applications, but also how to create general electronic systems. Familiarizes oneself with reading datasheets

Assembly language and embedded systems is an essential topic that most engineers (electrical and computer) should take. Short really did a good job in explaining the material.

The professor extended our learning into less fundamental but more practical knowledge. We used what we learned in previous design classes to make useful systems.

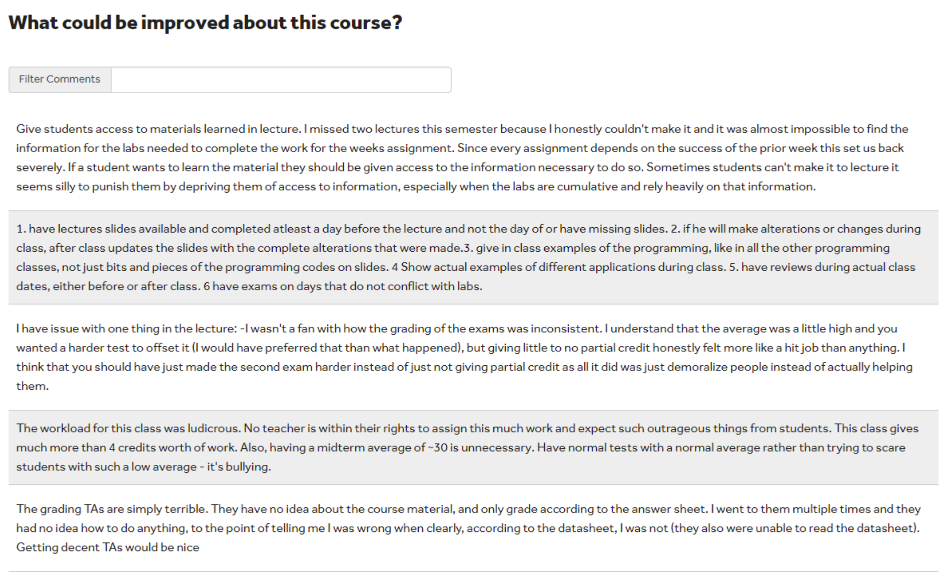
The labs were incredibly useful for learning the material. Lecture was excellent for learning the material.

This course was very good in terms of learning about industry standards and embedded systems.

Learning AVR Assembly, understanding how to make professional-level schematics.

Being introduced to, and utilizing, Microcontrollers

Figure 2: This figure shows an example of the section of unstructured data labelled "What was valuable about this course"? for a semester of ESE380



What could be improved about this course?

Filter Comments

Give students access to materials learned in lecture. I missed two lectures this semester because I honestly couldn't make it and it was almost impossible to find the information for the labs needed to complete the work for the weeks assignment. Since every assignment depends on the success of the prior week this set us back severely. If a student wants to learn the material they should be given access to the information necessary to do so. Sometimes students can't make it to lecture it seems silly to punish them by depriving them of access to information, especially when the labs are cumulative and rely heavily on that information.

1. have lectures slides available and completed atleast a day before the lecture and not the day of or have missing slides. 2. if he will make alterations or changes during class, after class updates the slides with the complete alterations that were made.3. give in class examples of the programming, like in all the other programming classes, not just bits and pieces of the programming codes on slides. 4 Show actual examples of different applications during class. 5. have reviews during actual class dates, either before or after class. 6 have exams on days that do not conflict with labs.

I have issue with one thing in the lecture: -I wasn't a fan with how the grading of the exams was inconsistent. I understand that the average was a little high and you wanted a harder test to offset it (I would have preferred that than what happened), but giving little to no partial credit honestly felt more like a hit job than anything. I think that you should have just made the second exam harder instead of just not giving partial credit as all it did was just demoralize people instead of actually helping them.

The workload for this class was ludicrous. No teacher is within their rights to assign this much work and expect such outrageous things from students. This class gives much more than 4 credits worth of work. Also, having a midterm average of ~30 is unnecessary. Have normal tests with a normal average rather than trying to scare students with such a low average - it's bullying.

The grading TAs are simply terrible. They have no idea about the course material, and only grade according to the answer sheet. I went to them multiple times and they had no idea how to do anything, to the point of telling me I was wrong when clearly, according to the datasheet, I was not (they also were unable to read the datasheet). Getting decent TAs would be nice

Figure 3: This figure shows an example of the section of unstructured data labelled "What could be improved about this course"? for a semester of ESE380.

The only problem is, the way Classie-Evals is set up is based on semesters for a particular class (you could also change it so that you view professors within Stony Brook as well), and with the amount of information that Classie-Evals has for a class per semester, it could end up being a lot of information that professors need to read through. The time of the professor is limited because they also must devote time for research, their personal life and whatever else it is that professors do.

It is convenient to have information about a class given in links for a specific semester, but it would also save a lot of time if professors can see how their class is evolving over a period of time that they choose. Classie-Evals does indeed have some graphs that give students an idea of the grading distribution for the class in that semester, and Classie-Evals recently started giving new information about classes after COVID-19 using more graphs. Below is an example of a bunch of these graphs for a random course within Classie-Evals.

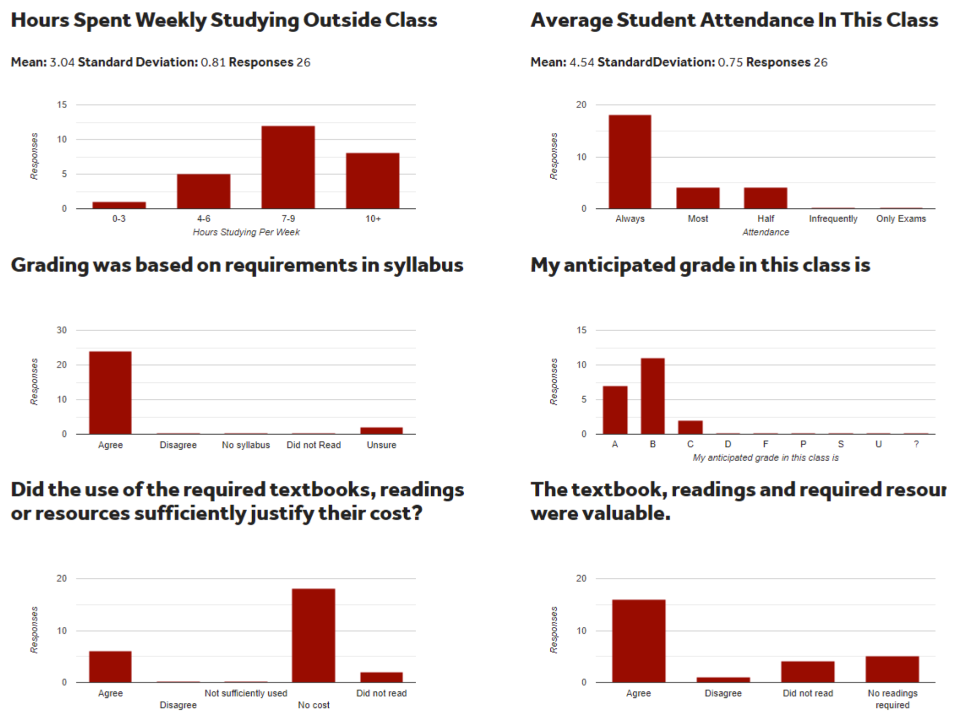


Figure 4: Example of the new graphs Classie-Evals started using after COVID-19

However, these graphs may not give the entire perspective of what the student is saying about the course. To understand that perspective, you would need to read through all the comments of students, and from the previous pictures, it should be clear that this is a lot of reading for professors to do on such a limited time frame. Also, there is a chance that professors may gloss over certain aspects of what students say because of internal biases or differences in their mastery of English (it is a complicated language that is constantly evolving).

To address these problems, we decided to give professors information that would allow them to see how their course evolves over time. We decided to **output graphs** that would **allow us to perform data analysis on certain keywords** (the important information pertaining to the course) and **how students feel about those keywords**, and then we also have **graphs** that would **allow us to show professors how their course evolves over time compared to the student's sentiment of the course** and **how certain aspects of the course have evolved over time compared to student's sentiment**. All this information will be explained more clearly in our explanation of the best case and worst-case design of the project.

First off, we would need to use an **API** (or an Application Programming Interface, which allows users to access services from another person or company using pre-written software) that already has code written that would allow us to use the functions within the API to do the analysis of data. We could have written our own code, but that would have taken too much time and money on our part, and we probably would not have been able to progress much on that due to the constraints that were set in place on our project. In our case, we decided to go for IBM Watson for the following reasons given in the table below. IBM Watson had its own disadvantages that we will talk about later in this paper, but for what we needed, we decided that this API would be the best for us to use.

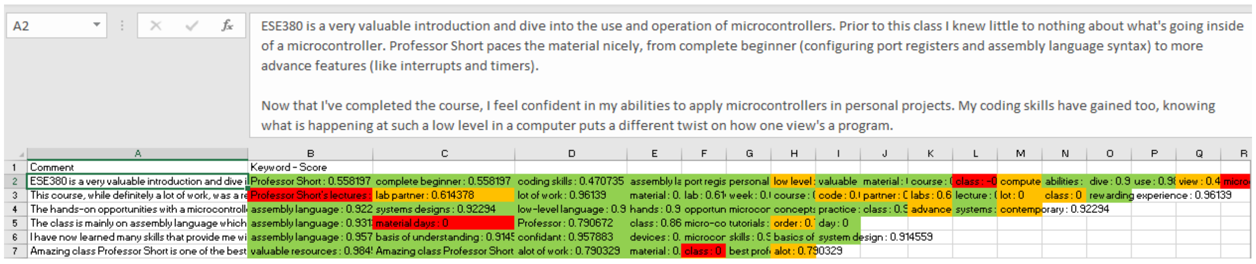
IBM Watson

Pros	Cons
✓ You can extract relationships, concepts, entities, semantic roles, keywords, etc. (versatile)	x The UI is not very intuitive (settings and requirements aren't very clear for users
✓ User-friendly	x The pricing can be steep
✓ Quick and high-resolution visuals	x The documentation may seem to be too much for a beginner, but it will become easier with more exposure
✓ Price model is flexible	x More suitable for medium to large sized businesses with lots of data
✓ Quick analysis	
✓ Community seems very helpful for helping beginners	
✓ The API seems to give a lot of power to the user through different APIs inside of IBM Watson that deal with different things.	

Table 1: Table showing the advantages and disadvantages of using IBM Watson

To verify the data that we received from IBM Watson, we decided to investigate the default sentiment model that IBM Watson uses. This is because a huge part of the project is based off the sentiment model that IBM Watson uses. If we deem the model to be insufficient in meeting our needs, we would then need to create our own sentiment model to create the program, which would mean more time spent to create a model that may not even work properly in the end due to limitations on our side (for example, the different intricacies of the English language).This was also a concern that our project advisor had brought to our attention, as she was worried that the default sentiment model that IBM Watson provided may not have been accurate enough for us to use, and that we might need to create our own.

To check this out, we decided to investigate the accuracy of the default sentiment model by using ESE380 from Fall 2018 as our test input. We decided to test this by reading each of the comments for the course, seeing what keywords IBM Watson returned to us, reading the comment again to find the keyword and understand the context, get an idea of the sentiment score that IBM Watson should have returned for the keyword and then comparing that to the actual sentiment score that IBM Watson returned to us. **If we agreed with the sentiment score**, we marked it with **green**. **If we didn't agree with the score**, we marked it with **red**, and **if we weren't sure about how to interpret the context the word appeared in**, we marked it with **yellow**. All of this was done with an excel file to make it easier for us to do analysis with the data later.



A2

ESE380 is a very valuable introduction and dive into the use and operation of microcontrollers. Prior to this class I knew little to nothing about what's going inside of a microcontroller. Professor Short paces the material nicely, from complete beginner (configuring port registers and assembly language syntax) to more advance features (like interrupts and timers).

Now that I've completed the course, I feel confident in my abilities to apply microcontrollers in personal projects. My coding skills have gained too, knowing what is happening at such a low level in a computer puts a different twist on how one view's a program.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1	Comment	Keyword - Score																
2	ESE380 is a very valuable introduction and dive i	Professor Short : 0.558197	complete beginner : 0.558197	coding skills : 0.470735	assembly la	port regis	personal	low level :	valuable	material : (	course : (	class : -0	compute	abilities :	dive : 0.9	use : 0.9	view : 0.4	micro
3	This course, while definitely a lot of work, was a re	Professor Short's lectures :	lab partner : 0.614378	lot of work : 0.96139	material : 0.	lab : 0.61	week : 0.	course : (	code : 0.	partner : (	labs : 0.6	lecture : (	lot : 0	class : 0	rewarding	experience : 0.96139		
4	The hands-on opportunities with a microcontroll	assembly language : 0.922	systems designs : 0.92294	low-level language : 0.9	hands : 0.9	opportun	microcor	concepts	practice :	class : 0.9	advance	systems :	contemporary : 0.92294					
5	The class is mainly on assembly language which	assembly language : 0.931	material days : 0	Professor : 0.790672	class : 0.86	micro-co	tutorials :	order : 0.	day : 0									
6	I have now learned many skills that provide me wi	assembly language : 0.957	basis of understanding : 0.9145	confidant : 0.957883	devices : 0.	microcor	skills : 0.9	basics of	system design : 0.914559									
7	Amazing class Professor Short is one of the best	valuable resources : 0.984	Amazing class Professor Short	alot of work : 0.790329	material : 0.	class : 0	best prof	alot : 0.790329										

Figure 5: Picture showing an example of the analysis for the default sentiment model using ESE 380

We did this for all the comments, and we ended up with the pie chart shown in the figure below. The area of the pie chart marked in **green** corresponds **to the percentage of the keywords whose sentiment scores we agreed with**, the area of the pie chart marked in **blue** corresponds to **the percentage of the keywords whose sentiment scores we disagreed with**, and then the area of the pie chart marked in **red** corresponds to **the percentage of keywords whose sentiment scores we were not sure how to interpret the context for**.

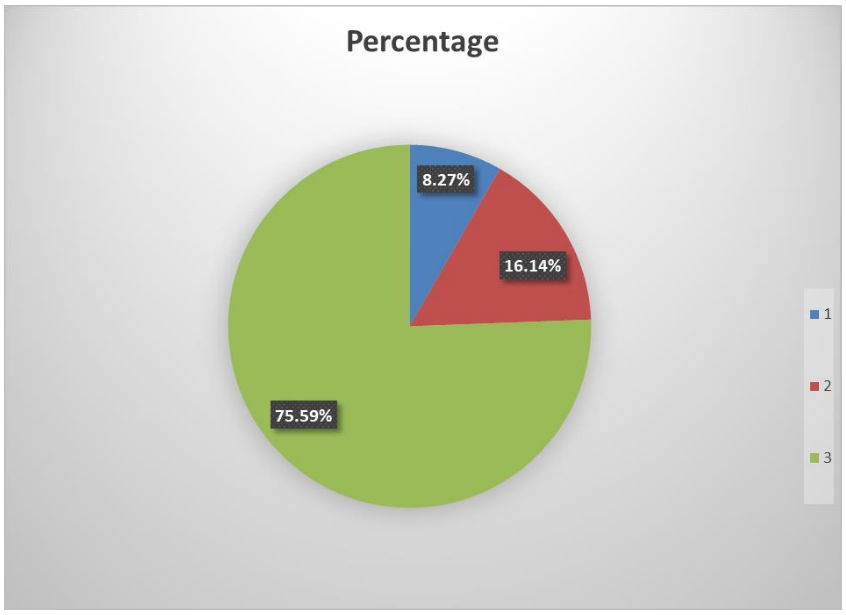
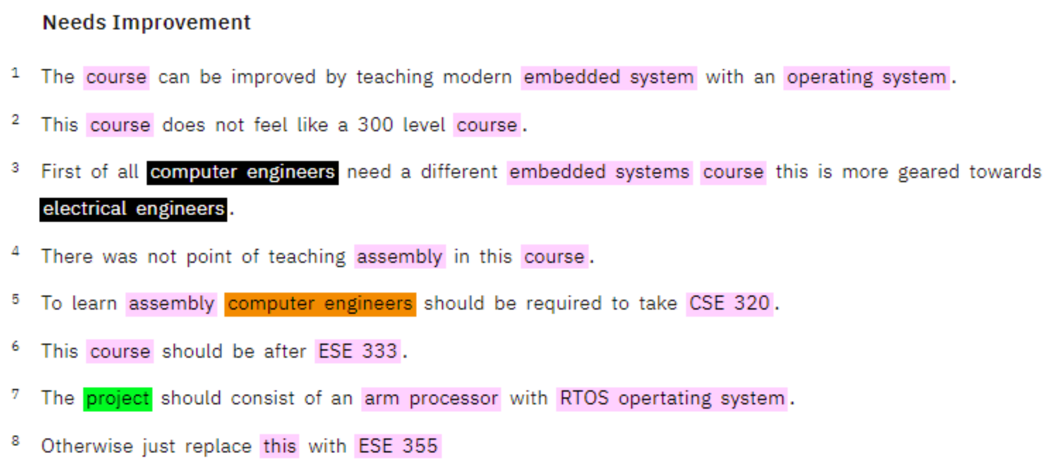


Figure 6: Picture showing the results received from the analysis of the default sentiment model

From this, we saw that the default sentiment model that IBM Watson gives us **a base accuracy of around 75%**. The default model gives us around **92% accuracy at its best**. We would approximate and say **that the accuracy of the model would be around the low to mid 80's**, which is actually a very decent percentage. We scrutinized the process for the creation of a sentiment model as well, and from what we saw, it was a very complicated process and was not worth the effort making. In that case, we decided to use the default sentiment model to proceed with the project.

Another important thing that we needed to do included the training and evaluation of the custom entity model that we were making. We had to make sure that the entity model we were making performed well to expectations, otherwise this would mean a lot of time wasted. IBM Watson usually expects at least 7 people on a team, but 10 is usually a good number of people on the team. People on the team annotate documents, which is the most time-consuming aspect of the project, and the part that we struggled with. Annotation allows us to basically let the model we are creating know how what entity types we are interested in, how to interpret certain words in certain contexts, etc.

You would have different sets that you would use (the **training set**, **test set** and **blind set**) and the sets are used for either **training** or **evaluation** of the model.



Needs Improvement

1 The course can be improved by teaching modern embedded system with an operating system.

2 This course does not feel like a 300 level course.

3 First of all computer engineers need a different embedded systems course this is more geared towards electrical engineers.

4 There was not point of teaching assembly in this course.

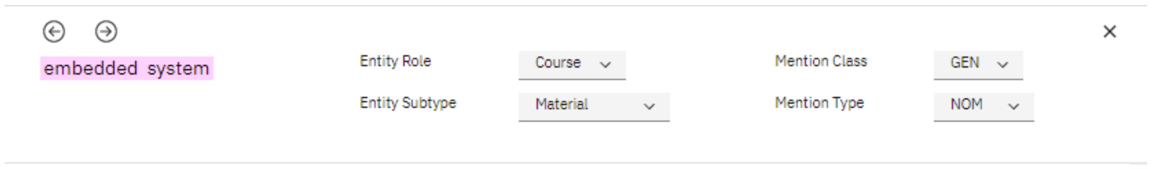
5 To learn assembly computer engineers should be required to take CSE 320.

6 This course should be after ESE 333.

7 The project should consist of an arm processor with RTOS opertating system.

8 Otherwise just replace this with ESE 355

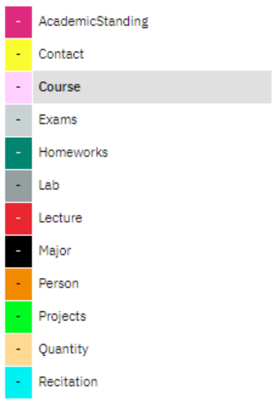
Figure 7: Picture showing an example of the annotations performed for ESE 380



embedded system

Entity Role	Course	Mention Class	GEN
Entity Subtype	Material	Mention Type	NOM

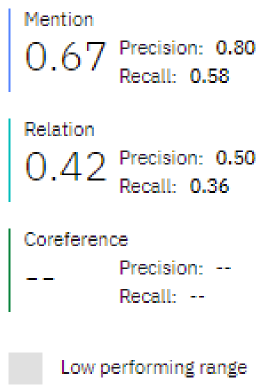
Figure 8: Picture showing how a certain word was annotated with a category of Course and a subcategory of Material



AcademicStanding
Contact
Course
Exams
Homeworks
Lab
Lecture
Major
Person
Projects
Quantity
Recitation

Figure 9: Picture showing the entities that we were using for annotation.

Once you do the annotation for the model, and start training and evaluating the model, you would get a performance score for the model.



Mention
0.67 Precision: 0.80
Recall: 0.58

Relation
0.42 Precision: 0.50
Recall: 0.36

Coreference
-- Precision: --
Recall: --

▨ Low performing range

Figure 10: Picture showing the results received from the training and evaluation of the default entities model

The picture above contains the most recent scores of our entities model. The scores that we are interested in are the mention score, the precision score for the mention and the recall score for the mention. The mention score is the overall performance score for our custom entities model. It is the bread and butter for the program, the end all be all. As you can see, **the score for the entities model is a 67%**. According to IBM Watson, a good score is anything above a 50%, so we aren't doing bad by any means, especially looking at the constraints placed on us.

A **precision score**, defined in the NLU API documentation for creation of a custom entity model, **is defined by the number of correctly labeled annotations divided by the total number of annotations added by the machine learning model**. In other words, **this score basically lets us know about how good the model is at classifying a keyword** (entity we are interested in) **as the entity types that we created in our type system and specified in the annotations**. It will calculate this score based on using our annotations (ground truth) as a reference point, and then check to see if it labels that entity the same way the human annotators label it. **A low precision score lets us know that the machine learning model needs improvement, since the model is generating incorrect annotations** (using the human annotator's annotations as a reference).
There are many different reasons for this, some of which are given in the documentation as:

- Domain
- Type System Complexity

- Appropriateness of Training Documents
- Human Annotator skills (This will play a big factor for us)
- Clarity of the annotation guidelines

A **recall score**, which is also defined in the NLU API documentation for the creation of a custom entity model, **is defined as a measurement which specifies how many mentions that should have been annotated by a given label were annotated with that label**. To calculate that, you would take the number of correctly labeled annotations divided by the total number of annotations that should have been made. **This score gives us an idea of how much the model misses marking annotations with a given label**. **A low recall score allows us to identify places where the model failed to create annotations it should have made (aka it misses a lot of annotations for a given label)**.

The reasons for low recall are the same as for the reason given for low precision scores, which are:

- Domain
- Type System Complexity
- Appropriateness of Training Documents
- Human Annotator skills (This will play a big factor for us)

We have a low recall score, so to improve the model, we would most likely need to find some way of improving the recall score, for example by annotating more documents and checking that all our previous annotations were consistent in the way they were annotated. The precision score seemed fine to us since we were originally aiming for a precision score of around the same magnitude as the default sentiment model (low to mid 80's, since going beyond that means that our default sentiment model will be the bottleneck in the performance of the program).

Innovation

What we decided to do was to focus on learning how professors can improve their course by showing them how their course evolves in different areas over time. Within a course, you would have many different aspects of the course like the lecture, recitation, projects, labs and so forth. Each different aspect of the course also has other different aspects too, like how students feel about how the teacher teaches that part of the course, how difficult that part of the course is, and other things as well. This is important to keep in mind for later, as there are many ways of breaking down a course. It can get complicated very fast, and you wouldn't want your biases to have a say in how you structure a course.

The group who was working on the same project in the previous year had decided to focus on the lecture sentiment (how students feel about the lecture aspect of the course), the exam sentiment (how students feel about the exams aspect of the course), as well as the professor sentiment (how students feel about the professor). However, they were able to generalize their program so that they would be able to look at the whole ESE department with these criteria in mind. From this, they were able to do a simplistic version of text mining, where they were able to derive some information about the course using some pre-built models that IBM Watson has. They had a **GUI** (**Graphical User Interface**) so that users would be able to interact directly with their program using icons and buttons, and they displayed their data using Matplotlib, which is a library in Python that produces static images. They were also able to show students comments given in a semester that corresponded to the keyword that a user put in as well.

We decided to take the machine learning aspect of the project a couple of steps further through analyzing other areas of the course by breaking the course down into different categories, like projects, recitations, exams, etc. We then further divided those categories into subcategories, like how students feel about how the teacher teaches that part of the course, how difficult that part of the course is, and other things as well, as mentioned previously. Below shows the reader how we decided to break a course down into different categories and subcategories. We tried to make it so that each category and subcategory label would be obvious as to what it was related to (for example, words relating to the course would go to the course entity, and words relating to the grading of the course would be labelled as a course entity with a subcategory of grading). It wasn't always obvious what to label a word as, but we tried our best by understanding how the word was used in the context it appeared in.

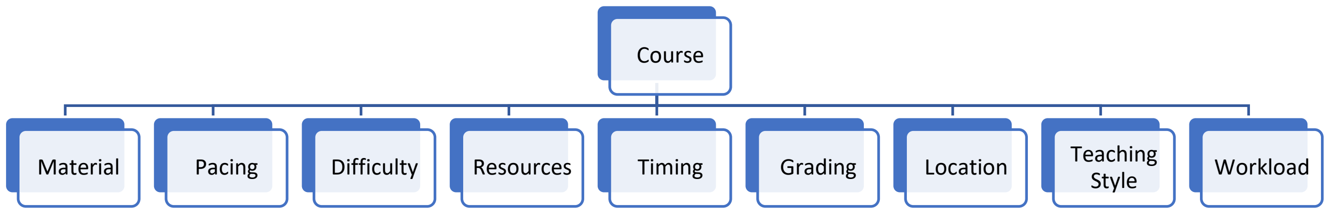


Figure 11: Graph showing how the "Course" entity in our type-system worked

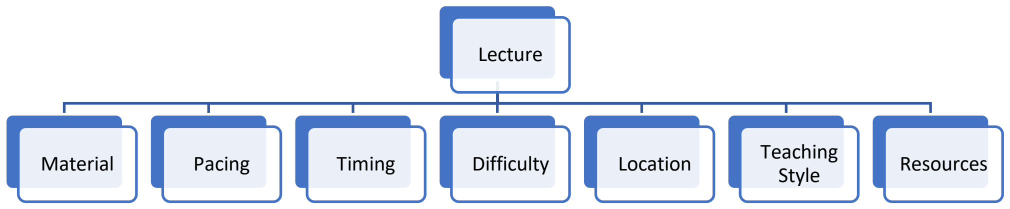


Figure 12: Graph showing how the "Lecture" entity in our type-system worked

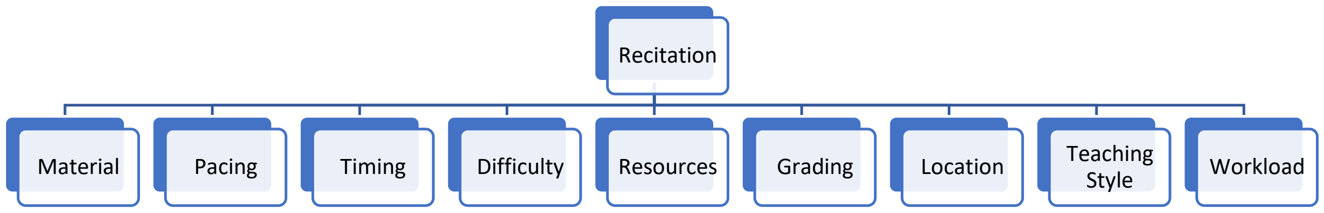


Figure 13: Graph showing how the "Recitation" entity in our type-system worked

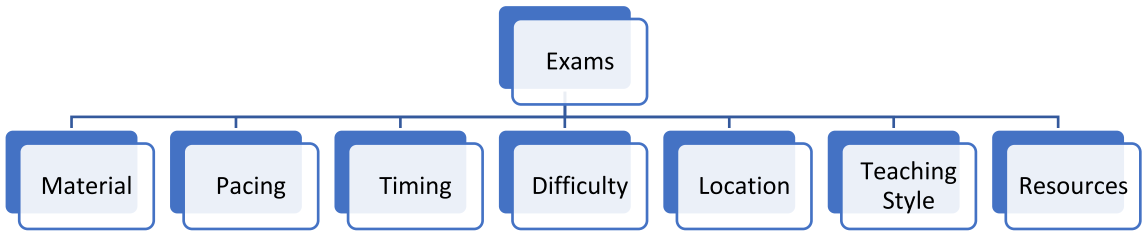


Figure 14: Graph showing how the "Exams" entity in our type-system worked

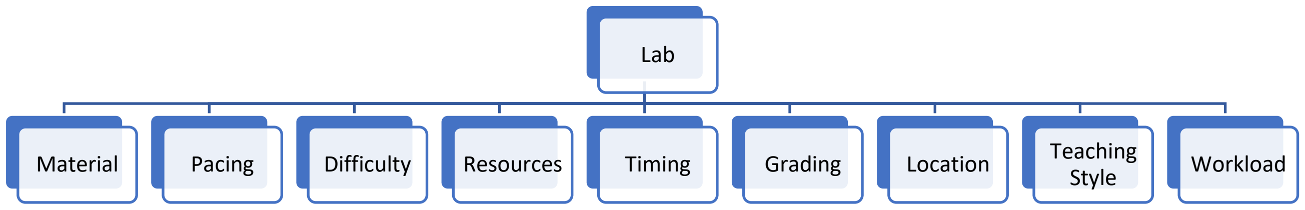


Figure 15: Graph showing how the "Lab" entity in our type-system worked

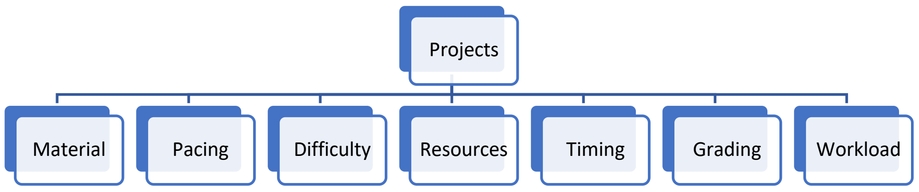


Figure 16: Graph showing how the "Projects" entity in our type-system worked

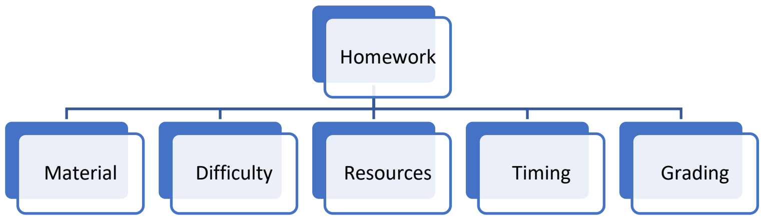


Figure 17: Graph showing how the "Homework" entity in our type-system worked

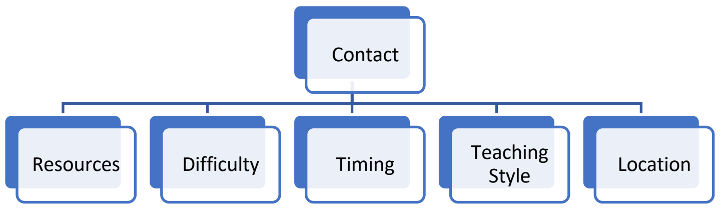


Figure 18: Graph showing how the "Contact" entity in our type-system worked

Our program allows us to divide a course down into different aspects of the course that professors could look at and think to themselves, "Hmmm, it seems as though a lot of people dislike this part of the course, maybe I should change it up a bit". The program does not give the professors what specifically is wrong with the material in and of itself, which is what we would have liked to do if we had more time, but our program would allow professors the chance to see how their course evolves over time, check to see what exactly they did in a specific semester and see how students react to that. This is where the actual text mining of the program comes to play, and compared to last year's program, this program is more involved in that aspect. This aspect of the program is shown in the graph below.

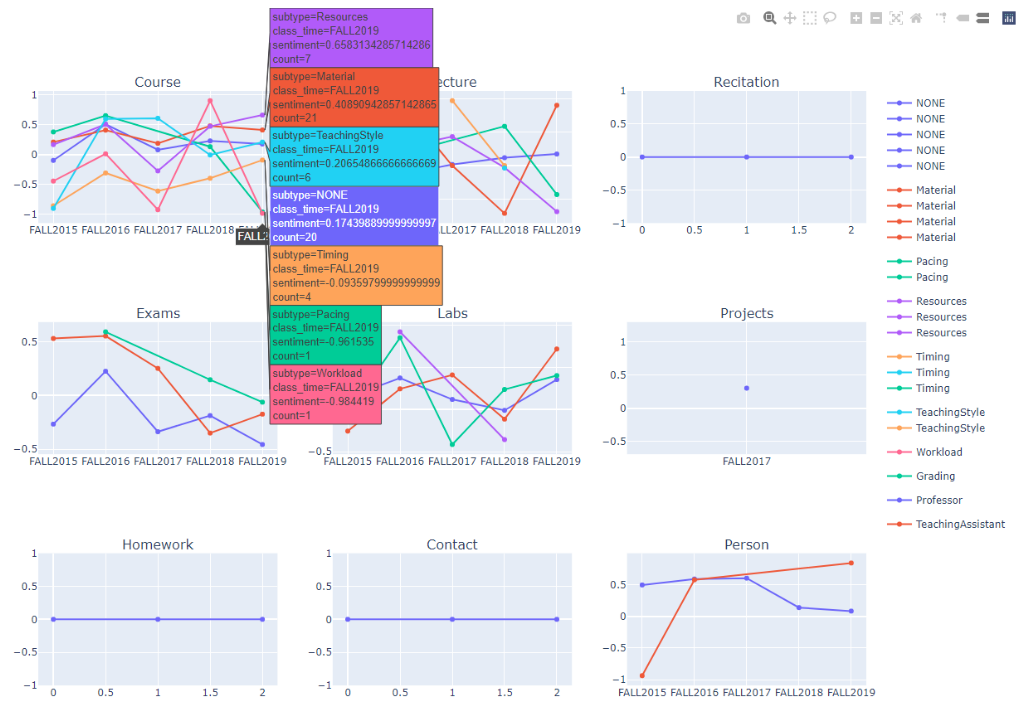


Figure 19: Picture of a graph with a bunch of different graphs that contain information about the important entity types in our type-system

We also decided to show how the course evolves over time, and this is shown in another graph below. This is like what the previous group from last year did, where we would use a default model (in this case, IBM Watson's default sentiment model) to check for the sentiment each comment, and then take the average of all those scores to create a sentiment score for the entire course. This will ultimately allow professors to understand how students perceive their course.

The difference between the graph of the "Course" category and this graph is that this graph gives a good idea of the overall sentiment of the entire course (including all the different subcomponents). The previous graph only takes entities (subjects) into account, while this graph takes other keywords into account, like actions. In other words, it gives a more "complete idea" of how students see the course in its entirety. The previous graph with the "Course" label will give professors a good idea of how different aspects of the course were liked or disliked by students.

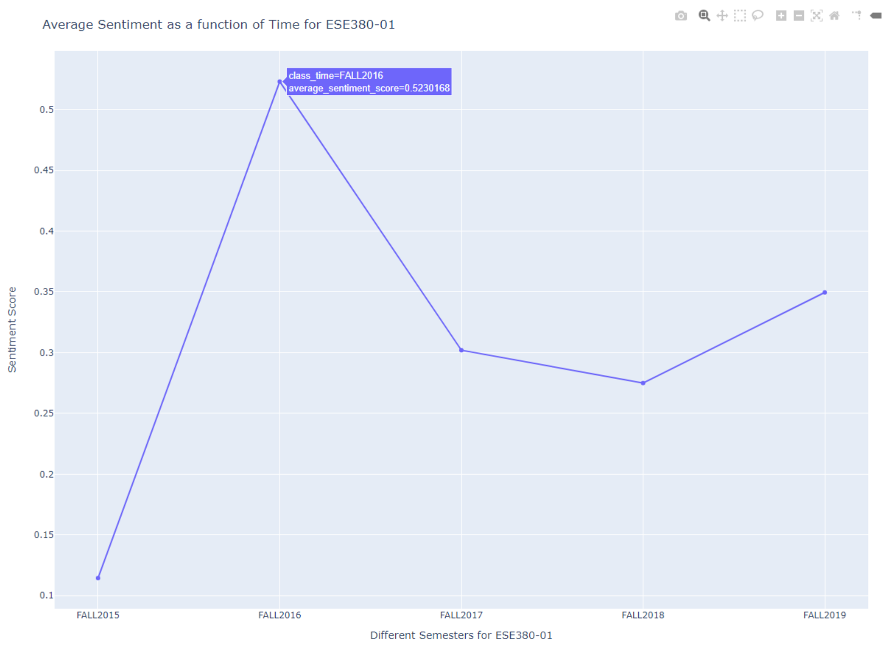


Figure 20: Example of the output for the graph of the overall sentiment score of the course

We also decided to show professors the types of words students use when talking about the course. This is so that professors can get an idea of how students feel about certain parts of the course, whether it be specific aspects of the course like certain topics, or just general aspects of the course like the labs, for instance.

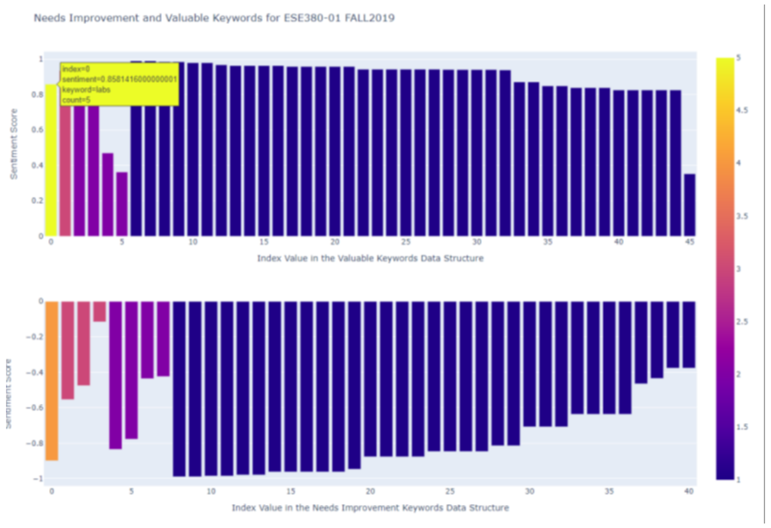


Figure 21: Example of a graph showing the important keywords in the course displayed as a bar graph

In our case, our program would run inside of a terminal, so that users would not have to be bombarded with clunky icons (the graphical user interfacing capabilities that Python provides is not great from what we see). The user would need to pick from a folder of .csv files that we created previously, and then put in the names of the files they want to use as input into IBM Watson, and then wait a couple of minutes for IBM Watson to return the data that is used for the output. For the output, we decided to display our images using Plotly, which outputs user interactive graphs that the professors can use to view the output of the program and offers a lot more interactive features compared to Matplotlib. The one downside with our program is that currently, compared to last year's ability to create their output for any class in the ESE department, we were limited to ESE 380, 381 and 382, because our model was trained with data from those classes. However, with the way we structured our program, given more time, we would be able to use our program with any class in the ESE department if we have trained our custom model with all the classes in the ESE department.

Constraints and Applicable Standards

This year, we had a lot of constraints which hindered the progress of our project. Some of these constraints were like the constraints that other teams taking ESE 440 and ESE 441 were facing, but due to the nature of the project, there were others that were applicable only to this project in terms of hindering progress. Currently, there is no applicable industrial standard for Machine Learning. Since our project is mainly software-based on Machine Learning, it does not propose any imminent danger or harm to the users of this application.

One big thing that came against us was the COVID-19 pandemic. We needed to find a way to get bring all the group members together at certain times during the semester to do research into the topic and learn Python (we were new to Python and were not too savvy in terms of the software that we designed, so this was all new to us), so there was a lot of meetings towards the beginning of the semester. We did not like Zoom that much, because only one person would be able to share their screen at a time. It would be annoying to switch screens constantly and would serve to just hinder our progress in the long run, so we decided to use Discord. Discord was a platform that is used for gamers to communicate with friends and hang out, but we decided to create a messaging channel and use it to talk about senior design stuff. In case we would need to talk about the project, we would message each other and then get on

Discord to talk about what we needed to talk about. The hard part, however, was getting everyone’s schedules to sync up. Some group members were busy when one was free, and vice versa, so our progress could have been better if all the group members were not as busy.

Since the program needs to access data from Classie-Evals, we needed to find some way of accessing Classie-Evals through a log-in with our program. In this case, the program was limited by an authentication method (DUO) that Stony Brook University had recently implemented for staff members and people working on campus. We were able to work around that, since two people of the three people in the group were still able to access Classie-Evals without DUO. However, we knew that this was not a good workaround for the program. The purpose of the program was to help professors understand how their course evolves over time, and from that, use the information to better their course. Since our main target audience was the professors of the ESE department at Stony Brook, we needed to tailor the program so that it would be simple for professors to use. Since professors had DUO set up for their account, we would not be able to have professors use the program. In other words, we needed to find another workaround.

The pricing of IBM Watson’s API and the size of our team was another constraint as well, since we would need to pay a certain amount of money depending on how many comments we decided we wanted to analyze, and IBM Watson assumes that you would have at least 7 people on a team to do the annotations. To make this program usable for other professors in the ESE department, Stony Brook would need to pay money to IBM to use their NLU API, and the amount of money can be quite costly. We did not need to pay anything, luckily, but if we had to pay actual money to use the NLU API, we would have been spent our entire budget in around half a month since one of IBM Watson’s plans has us pay $800 per month. There are other plans that IBM Watson provides, but this number is given to give the reader an idea of how expensive using an API can be.

We were also limited to the quality of the feedback that students give in the course evaluations as well. We noticed that for some of the course evaluation comments, we were able to see a lot of information about what the class was doing right and what it was doing wrong, which helped us out a lot for this project. However, there were other comments that were not very helpful, like people saying the words, “Nothing” or “Everything”. There were a lot of comments on Classie-Evals that had comments like these, and they hinder the effectiveness of the model because the students aren't isolating specific aspects of the course that they like or dislike. If they did, in fact, like or dislike everything about the course, they could say that using full sentences. Our project was hindered so much from comments like these on Classie-Evals.

Since this is considered mainly a software project, we decided that to prevent any of the members of the group from having conflicting versions of code (certain parts of the code may have different variables, different ways of doing things, etc.), we decided to use GitHub to help us keep the code more universal between all the group members, to prevent the conflicting code problems that we have mentioned before. It will also help us with version control, because in the case that we accidentally delete a file from our project folders, we can just re-download the file from GitHub and do whatever we need to with the file again.

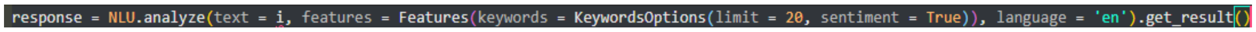
Best/Worst Case Design

Initially, while we were brainstorming our ideas, we realized that we would need to develop both the worst-case design (the base functionality that our project needed to implement) and a best-case design (our true goal for the project). To lead up to our worst-case design, we processed all the information extracted from the Stony Brook Course Evaluations webpage and stored them into CSV files. The Natural Language Understanding (NLU) API provided many important features that were vital to our program in assisting professors with evaluation of their class evaluations. The three main features were the:

- Entities Feature
- Keywords Feature
- Sentiment Feature

Each feature played a key role in allowing us to extract necessary information and provide us feedback to show how accurate the NLU API was by itself. After discussing the different features and what was necessary to our program, we then started to test how each feature would work. The Keywords Feature was the main feature that we used for our worst-case design. Extracting words from the comments that the API dictated as keywords allowed us to measure the accuracy of the API by looking at the sentiment of the keyword and understanding the context of the keyword. Using a function called the **analyze** function, we were able to extract data from an input and output the feature associated within the line of code. The JSON library in

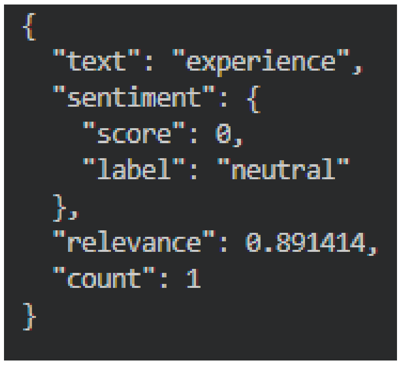
Python also allowed us to store the information that the API extracted as a JSON output and output that information cleanly onto the terminal. Using that JSON output, we were able to see what information was extracted. For example, the keywords feature allowed us to extract keywords using the input of unstructured text. As explained before, we used the **analyze** function that the API provided and specified the specific parameters that were enclosed within the parentheses. Provided below is a screenshot of the analyze function.



```
response = NLU.analyze(text = i, features = Features(keywords = KeywordsOptions(limit = 20, sentiment = True)), language = 'en').get_result()
```

Figure 22: Figure showing the function name and arguments used within the NLU API

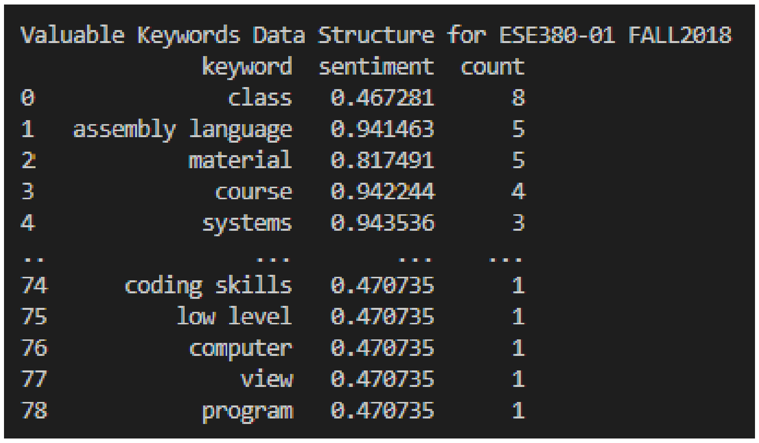
Specifying the feature is important because it allows the API to choose which of the features that it needs to utilize. As such, the output of this analyze function would be words that the NLU API would deem as “keywords”. Shown below is the output. It provided the text that the API extracted, the relevance of the word within the text, the number of times the word appeared in the input text and the sentimental score of it (meaning if it provided a positive reaction or a negative reaction).



```
{
  "text": "experience",
  "sentiment": {
    "score": 0,
    "label": "neutral"
  },
  "relevance": 0.891414,
  "count": 1
}
```

Figure 23: Example of using the JSON library to get a clean output out to the terminal for debugging

After we retrieved all the information provided by the JSON files, we would then output the keywords onto a bar graph. We utilized Plotly, a Python library which would allow us to plot these graphs. We also used the Pandas library as well to store the JSON output into a table (known as a frame) that would separate based on our columns. Attached below is how the Pandas table would look like.



```
Valuable Keywords Data Structure for ESE380-01 FALL2018
              keyword  sentiment  count
0               class   0.467281      8
1   assembly language   0.941463      5
2            material   0.817491      5
3              course   0.942244      4
4             systems   0.943536      3
..                ...        ...    ...
74       coding skills  0.470735      1
75           low level  0.470735      1
76            computer  0.470735      1
77                view  0.470735      1
78             program  0.470735      1
```

Figure 24: Example of a Pandas frame structure outputted onto the terminal

The first column would specify the keyword, the second column would specify the sentiment of the keyword and the last column would specify the frequency of how often the word appeared. Plotly would then use this Pandas data frame to graph the information. Plotly was the best choice as it allowed for a user interactive interface with the graphs. Although the graphs are outputted within a web browser, it did allow features that users could play around with such as a zoom functionality and a hover feature. Shown below would eventually be our worst-case design.

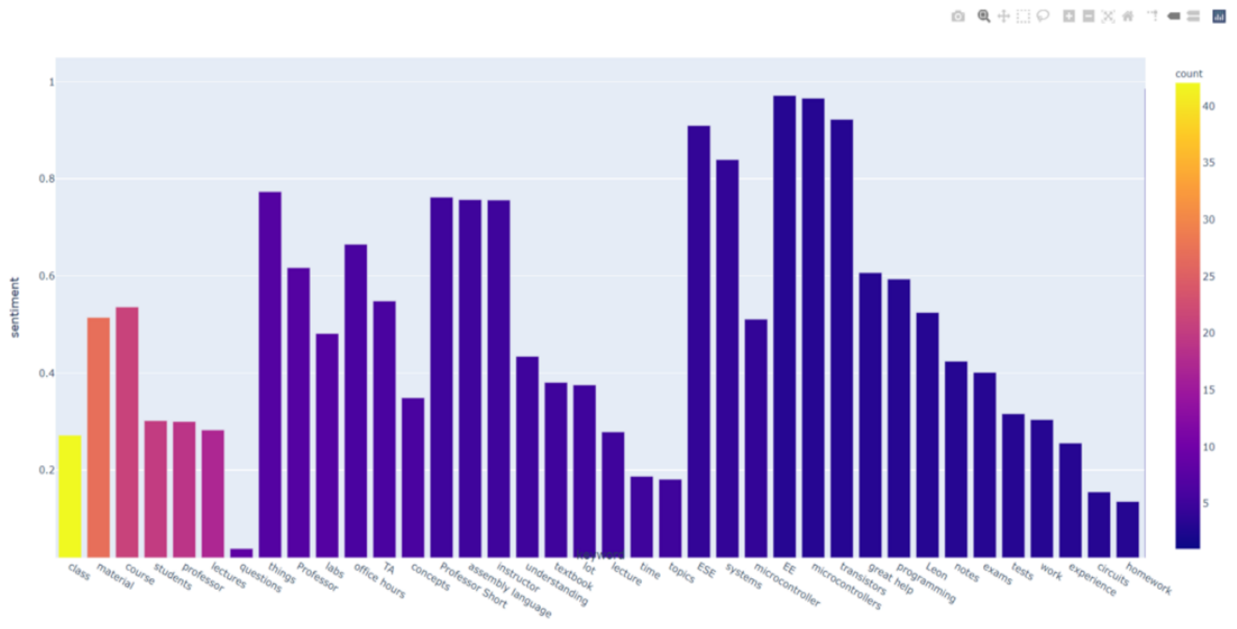


Figure 25: Example of a graph showing the important keywords in a course displayed as a bar graph

One disadvantage of this graph was that although it gave information of what students were saying about specific topics about the course and how they felt about those topics, we still did not believe that it would provide sufficient information for professors to look to see what they could change about the course. In addition, we also had an idea of using word maps to help us illustrate the idea of what keywords were discussed most about. Shown below is an example of how a word map would look like:

Figure 26: Picture of a Word Cloud

However, we did not go along with including the word map because we believed that it would provide redundant information already given by the bar graph, and the word map seemed more cluttered and harder to read.

This eventually led us to brainstorm our best-case design. With all the information considered, our team decided that we could show how a course has evolved over time. Utilizing the sentiment data that was provided in our worst-case scenario to show a more general scale of how people thought about the class, we took the average sentiment of all the sentiment values that was outputted by the sentiment feature and the keywords feature. In this way, we would then show the sentiment value of the course itself for a particular semester. To show how it evolved, we then took the overall sentiment value of a course over a course of different continuous years to show how the course was improved or decreased in quality. Below would show us our line graph output of our best-case design.

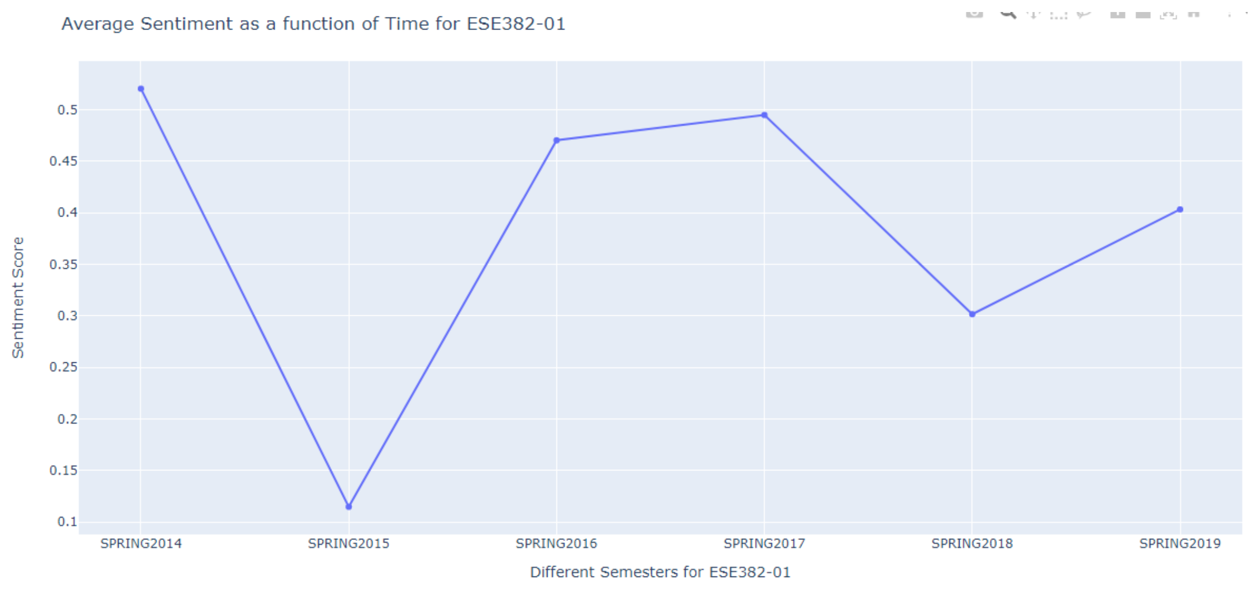


Figure 27: Line graph showing the average sentiment in a course over a period of time

In addition to showing the average sentiment score being changed over time, we also considered the machine learning aspect of the NLU API spoken about previously. Based on the Entities feature that we looked at previously, we were able to output subplots based on what our machine learning custom entity model learned from the annotations. These subplots would show us how a specific topic of the course has changed over time. Based on the type system previously discussed before, each subplot would correspond to an entity type. Shown below is the overview of all the entity types described.

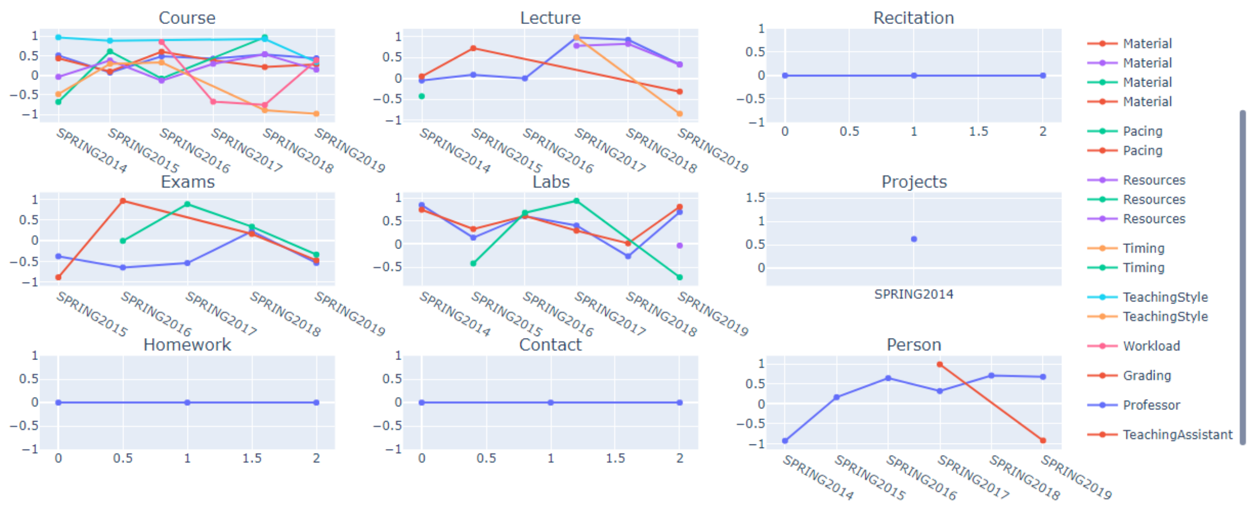


Figure 28: Picture of a graph with a bunch of different graphs that contain information about the important entity types in our type-system for ESE 382

Within each entity subplot, each line plotted corresponded to a subtype. Shown below is a zoomed-in view of how the subplot looks.

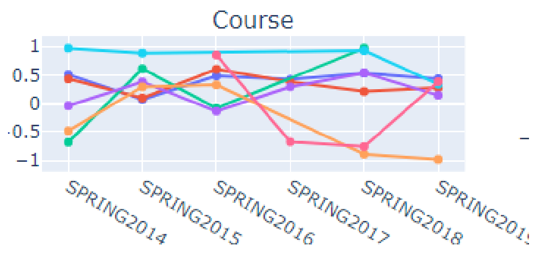
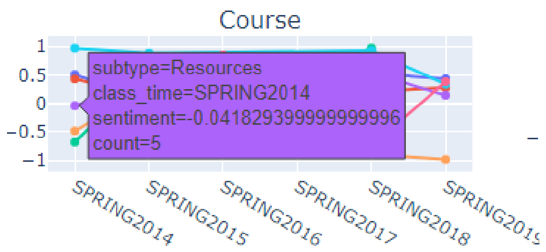


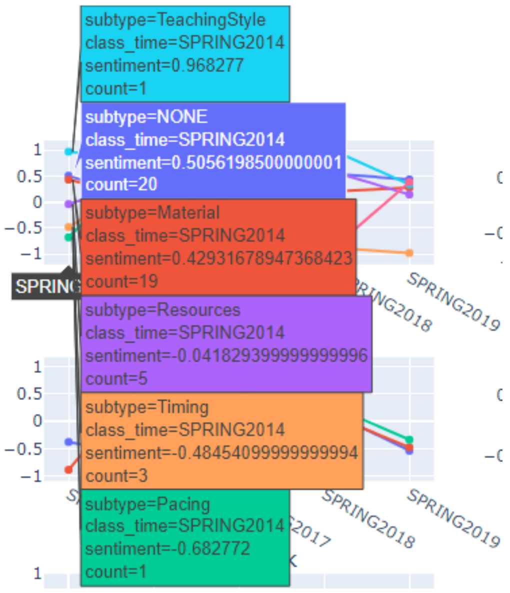
Figure 29: Zoomed in picture of one of the subplots



subtype=Resources
class_time=SPRING2014
sentiment=-0.041829399999999996
count=5

Figure 30: Zoomed in picture of one of the subplots showing off one of the features of the user-interactive graphs

As mentioned, Plotly allows us to show the data of a point on the line graph if you hover over it. Plotly also has a feature that allows you to view all the hover data of a specific year. Shown below is the hover data for all the subtypes for the semester of Spring 2014.



subtype=TeachingStyle
class_time=SPRING2014
sentiment=0.968277
count=1

subtype=NONE
class_time=SPRING2014
sentiment=0.5056198500000001
count=20

subtype=Material
class_time=SPRING2014
sentiment=0.42931678947368423
count=19

subtype=Resources
class_time=SPRING2014
sentiment=-0.041829399999999996
count=5

subtype=Timing
class_time=SPRING2014
sentiment=-0.48454099999999994
count=3

subtype=Pacing
class_time=SPRING2014
sentiment=-0.682772
count=1

Figure 31: Zoomed in picture of one of the subplots showing off some more features of the user-interactive graphs

As presented, these output graphs provide the users a more detailed view of how the over class has been doing over the years and the specific topics such as if the course material's quality has improved or decreased over time.

Methods

To breakdown the project into a series of steps, we would have these different stages shown below.

- **Gathering of the Input data and Preprocessing**: This step involves us gathering the data from Classie-Evals and putting it into a form that we can use to put into Classie-Evals.
    1. To gather the data, we decided to make another script that would allow us to web scrape Classie-Evals and gather the information we need (the comments in the "What was valuable about the course?", comments from the "What needs improvement in the course", and other things like the class name, time of offering and professor's names). However, we realized that it would be impractical to use this program for the actual program, since the web-scraping portion only works when you wouldn't have DUO set up for your Stony Brook account. Professors have DUO set up, so we need another way of having the information ready so that professors can use it at their will.
    2. In this case, we decided to use the current web-scraping program that we must gather the information and store them inside of .csv files within a folder that the professor also has. This would allow us to have a local database that professors can freely access through our program, and they can also choose the file or files that they want to look at. The way that we decided to store the information inside of the .csv files were structured so that we would be able to access the information in the file easily to give to the API, and it was also useful for training with the model later.
- **Choosing of the Input**: Professors can then run the program and use the terminal to choose what files or files they decide they were interested in. We decided to keep this part of the program nice and simple by having the professors do it in the terminal so they wouldn't be confused as to what they needed to do to run the program.

- **Use the NLU API to perform the Analysis of the Data**: From the files that we have the user choose, we then call the NLU API using the analyze function supplied by IBM Watson, along with some authentication keys that would allow us to use the API. This step takes around 2-3 minutes depending on how fast your internet connection is, how much data you are supplying to IBM Watson, etc. It would take more time if you were analyzing a lot of classes as well.
- **Receive the results from the NLU API and perform process the results to prepare for outputting of data in a clean form**: We are receiving the data returned from the features that we used (sentiment, keywords, and entities). After the data is received, we would then process that data, and then store all that information into a single data structure so that it would be easier for us to use when outputting data.
    1. For the keywords feature, this means we would combine all the duplicated keywords (many comments contain the same keywords), add together all of the count and sentiment scores, and then at the end take an average of all the sentiment scores so you have an idea of how many times the keyword is mentioned and the sentiment behind those keywords.
    2. For the sentiment feature, we would take the sentiment scores for all the comments in the course (both the comments from the valuable section and the comments from the needs improvement section), add them together, and then take the average of those scores to get a definitive idea of how people feel about the course.
    3. For the entities feature, we would see what the entity's category and subcategory relates to, and then put all those words together, add together all the sentiment for each category and subcategory and then take the average. For example, a bunch of entities may be returned for the category **Course** and the subcategory **Material**. We would see what keywords corresponded to that category and subcategory, add up all the sentiment scores there and then take the average of all of that.
- **Output the Information using Plotly:** In this case, we are just putting all the information from the processing and outputting that into Plotly.

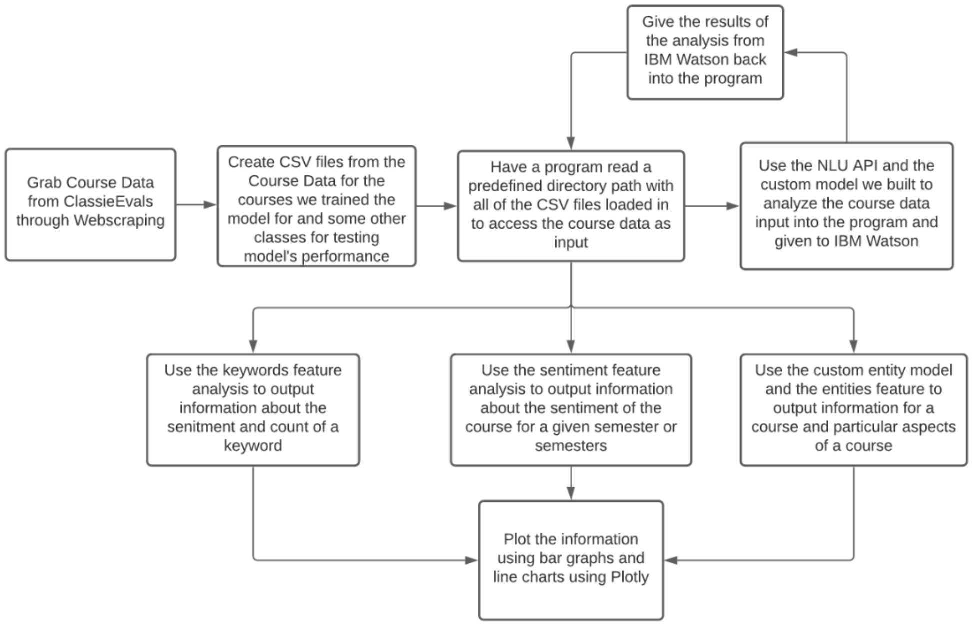

Give the results of the analysis from IBM Watson back into the program
Grab Course Data from ClassieEvals through Webscraping
Create CSV files from the Course Data for the courses we trained the model for and some other classes for testing model's performance
Have a program read a predefined directory path with all of the CSV files loaded in to access the course data as input
Use the NLU API and the custom model we built to analyze the course data input into the program and given to IBM Watson
Use the keywords feature analysis to output information about the senitment and count of a keyword
Use the sentiment feature analysis to output information about the sentiment of the course for a given semester or semesters
Use the custom entity model and the entities feature to output information for a course and particular aspects of a course
Plot the information using bar graphs and line charts using Plotly

Figure 32: Process of how the program is supposed to run

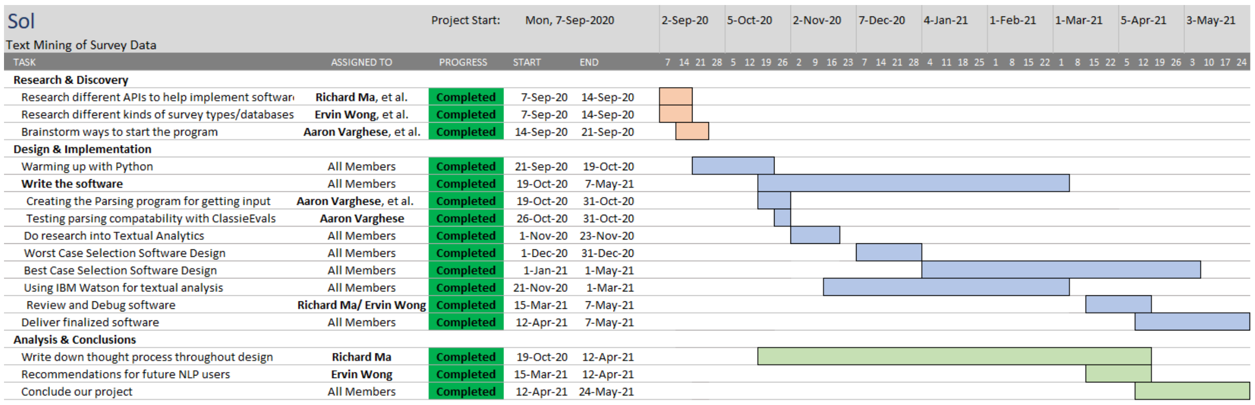
Results

With everything put together, we were able to successfully accomplish our end goal. First, we finished our worst-case design, which would simply organize sentiment data of different keywords from IBM Watson. This would allow us to get an idea of the type of words students use to describe the course, which is something we felt professors would have wanted. A bar graph would be created from this data using the Plotly API which would provide a visual for professors to see. As stated before, the worst-case design would show keyword count and sentiment for specific course reviews on a particular class. For the best-case design, our program will gather numerous sentiment averages of a course, displaying a line graph showing the sentiment average over the years, giving professors an idea of how the course is evolving over time. This data can prove helpful for professors who want to visualize how their course is going, and how they could steer their course accordingly. We also created our own entity type model, with about an 80% accuracy rating. Using our entity type model, the program can display a graph which shows certain sentiment scores of specific entity types identified for a course. This allows the professor to distinguish which part of their class should be improved, and their characteristics over the years. Overall, we believe that our program offers a lot of detailed information for professors to analyze when reviewing their courses.

Discussion on Ethical Concerns

One ethical concern which our group encountered while creating this project involved the privacy of the reviews which are stored in the class evaluation database. Since the reviews are supposed to be private and only kept within students within Stony Brook University, it could be a problem if someone not in the college breached the reviews. Additionally, while attempting to scrape off the webpage, we encountered some issues involving the DUO authentication service. We were able to bypass the DUO authentication service so that we can use the information in the surveys, but the topic of web-scraping is still a very gray area. Although we had the rights to access the reviews since we were Stony Brook students, the program we were using for web-scraping may not have. Because of these concerns and some other concerns listed above, we decided not to include the web-scraping program as part of the actual program.

Appendix A (Complete Gantt Chart)



Sol

Text Mining of Survey Data

Project Start: Mon, 7-Sep-2020

TASK	ASSIGNED TO	PROGRESS	START	END
Research & Discovery				
Research different APIs to help implement softwar	**Richard Ma**, et al.	Completed	7-Sep-20	14-Sep-20
Research different kinds of survey types/databases	**Ervin Wong**, et al.	Completed	7-Sep-20	14-Sep-20
Brainstorm ways to start the program	**Aaron Varghese**, et al.	Completed	14-Sep-20	21-Sep-20
Design & Implementation				
Warming up with Python	All Members	Completed	21-Sep-20	19-Oct-20
Write the software	All Members	Completed	19-Oct-20	7-May-21
Creating the Parsing program for getting input	**Aaron Varghese**, et al.	Completed	19-Oct-20	31-Oct-20
Testing parsing compatability with ClassieEvals	**Aaron Varghese**	Completed	26-Oct-20	31-Oct-20
Do research into Textual Analytics	All Members	Completed	1-Nov-20	23-Nov-20
Worst Case Selection Software Design	All Members	Completed	1-Dec-20	31-Dec-20
Best Case Selection Software Design	All Members	Completed	1-Jan-21	1-May-21
Using IBM Watson for textual analysis	All Members	Completed	21-Nov-20	1-Mar-21
Review and Debug software	**Richard Ma/ Ervin Wong**	Completed	15-Mar-21	7-May-21
Deliver finalized software	All Members	Completed	12-Apr-21	7-May-21
Analysis & Conclusions				
Write down thought process throughout design	**Richard Ma**	Completed	19-Oct-20	12-Apr-21
Recommendations for future NLP users	**Ervin Wong**	Completed	15-Mar-21	12-Apr-21
Conclude our project	All Members	Completed	12-Apr-21	24-May-21

Appendix B (List of Team Meetings)

Meeting Number (and Day)	Date	How long the meeting lasted
Meeting #1 (Sunday)	9/13/2020	30 min
Meeting #2 (Saturday)	9/19/2020	1 hour
Meeting #3 (Sunday)	9/27/2020	1 hour
Meeting #4 (Sunday)	10/18/2020	10 min

Meeting #5 (Sunday)	11/7/2020	45 min
Meeting #6 (Sunday)	11/22/2020	45 min
Meeting #7 (Tuesday)	11/24/2020	45 min
Meeting #8 (Monday)	11/30/2020	10 min
Meeting #9 (Sunday)	12/06/2020	2 hours
Meeting #10 (Monday)	12/28/2020	1 hour
Meeting #11 (Sunday)	1/03/2021	2 hours
Meeting #12 (Tuesday)	1/11/2021	2 hours
Meeting #13 (Sunday)	1/16/2021	3 hours
Meeting #14 (Thursday)	1/20/2021	2 hours
Meeting #15 (Saturday)	1/23/2021	3 hours
Meeting #16 (Friday)	2/05/2021	1 hour
Meeting #17 (Sunday)	2/07/2021	3 hours
Meeting #18 (Tuesday)	2/16/2021	2 hours
Meeting #19 (Saturday)	2/27/2021	2 hours
Meeting #20 (Saturday)	3/27/2021	2 hours
Meeting #21 (Saturday)	4/17/2021	1 hour
Meeting #22 (Sunday)	4/18/2021	1 hour
Meeting #23 (Monday)	4/19/2021	4 hours
Meeting #24 (Tuesday)	4/20/2021	4 hours
Meeting #25 (Sunday)	4/25/2021	2 hours
Meeting #26 (Monday)	4/26/2021	3 hours
Meeting #27 (Tuesday)	4/27/2021	2 hours
Meeting #28 (Monday)	5/01/2021	20 minutes

We also meet up every Friday for each week with our advisor as well for the first semester. For the second semester, we kept in contact through emails in case we had any questions about the project, since we needed to focus on how we were going to implement the project.

Appendix C (Recommendations for Future Prospects)

From this project, we learned many key aspects such as learning to program with Python, had a chance to learn about machine learning and utilized APIs and libraries to assist us in doing the manual labor. However, there were some things we could have done differently that would have helped us.

- **API of choice**: Be careful with the API you are choosing (if you choose one), as that will make or break the project. IBM Watson was a good choice for us, but that was because of the vast number of tools it provided us. If we decided to go the paid route, we would have been done for. Looking back, we most likely would have gone for

Google's API, but IBM Watson also released new features while we were working on the project that seemed worthwhile.

- **Take advantage of free time**: Don't waste the time you have during winter break. You are free for a month, so use your time wisely.
- **Divide the tasks evenly**: Make sure that everyone has an equal share of work to do among yourselves (we didn't have this problem, but for a project like this, it's important to keep in mind).
- **Be sure to do your research thoroughly**: Don't jump in immediately to finish something. Relax, you'll have time to finish it off.
- **Be sure to allot enough time for the software aspect of the project**: In the end, we had a working program, but it was a bare bones program. We spent most of our time in the creation, training, and evaluation of the entity model that we were making for the best-case design of the program and didn't leave enough time for the software portion of the project. Although we were able to complete the goals of the project, it would have been nice to extend the project to include more functionality to help professors improve their course.

Bibliography

1) Elokence, en.akinator.com. [Online]. Available: https://en.akinator.com/. [Accessed: 08-Sep-2020].
2) K. D. Foote, "A Brief History of Natural Language Processing (NLP)," DATAVERSITY, 17-Jun-2019. [Online]. Available: https://www.dataversity.net/a-brief-history-of-natural-language-processing-nlp/. [Accessed: 06-Sep-2020].
3) M. Rouse, "What is text mining (text analytics)? - Definition from WhatIs.com," SearchBusinessAnalytics, 31-May-2018. [Online]. Available: https://searchbusinessanalytics.techtarget.com/definition/text-mining. [Accessed: 06-Sep-2020].
4) SAS, "What is Natural Language Processing?" [Online]. Available: https://www.sas.com/en_us/insights/analytics/what-is-natural-language-processing-nlp.html. [Accessed: 07-Sep-2020].