Project:
Absurdly Intelligent – A Simple Commentator Bot Using Machine Learning and Natural Language Processing.

ECS170, Spring 2018

Group Members

Saksham Bhalla
Aakaash Kapoor
Yash Bhartia
Rohit Nair
Hung Mai

Problem and Contribution Statement

Currently in StarCraft II, players can watch the replay or look at the post-game stats to see how they matched up against their opponent. By watching the replay, a player can see what actions their opponent take throughout different phrases of the game. But, it may be difficult to determine which actions or phrases in the game that is a key turning point. Our tool can act as a live commentator on their own gameplay and make watching replays more fun; at the same time, assist the player to identify key points in the game. Based on our preliminary research, we believe that no significant work has been done on this problem and the only tool that the new users have right now is watching professional tournaments or streamers on youtube or twitch and learn from there. However, we believe that live commentary on the player own's replay would significantly help them learn to play better.

One of the focuses of artificial intelligence in general is to try and model the human mind and philosophy, and this could be related to the problem we're trying to solve with this bot. Commentating is a job that great Esports casters take years to master, and that too the problem is they would never cast a game that happens between two casual players. A solution would be a bot that is able to provide useful information on impactful events happening in the game. We wonder how accurate we can make a bot that learns how to commentate and what to commentate on games depending on how important an action is. We know that this will prove to be a fun and challenging problem to solve.

We will implement Support Vector Machines that first look at the list of actions that happens throughout the game and figure which actions are more important to provide

commentary on. Support Vector Machines are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. Since we will have a lot of data to analyze given our replay dataset, a support vector machine would be able to predict whether a task is essential to the game. While training we will classify each task of the winner as a 1 and classify each task of the loser as a 0. This will allow us to predict the important task given more than one task.

Possible alternatives to determine how well a player did are the post-game stats, or an actual StarCraft II caster, however hiring a caster for a casual game is not realistic. Amateur players may find the post game stats to be overwhelming to analyze and pinpoint what aspect of the game to improve on. It also requires that spectators wait until the game is over to see how the players played relative to each other. Our solution, hopefully, will provide action-by-action (both important micro and macro actions) analysis to the user in real time.

Design and Technical Approach

For this project we are using NLP techniques such as normal word processing into natural language and SVM for training purposes. We will have more than one type of Support Vector Machines and we have to choose from within those. To do so, our training data will be split into training and testing , and we will find the best accuracies within each model and choose the best model. Additionally we will not be implementing a neural network because our input layer would have a range of data namly but not limited to the health of each building and health of each monster. Although neural networks would work with a big input layer, it would only work if we have a lot hidden layers and the computation time would thus be slower. Since we want real time commentary on the replays, a slow approach is not ideal.

Hopefully we have access to decent amounts of data to make our model more accurate. We also use natural language generation models to contextualize our results in a way that the users or spectators can understand. The general problem is that human work is necessary in commentating on sports, and that games that don't have commentary may be hard to learn from or understand. Making a bot that can at least recognize important actions in a way that it emulates casting in StarCraft II is a beginning in trying to solve this problem.

Our output layers would be spitting out 1 hot vectors that would trigger different functions. We would have a natural language sentence generated function which would generate sentences around selected sentence blanks associated with word types. For example: "_____ won the game!" The blank is suppose to be a proper noun. The rest of the words are generated to indicate the blank won!

The reason why we feel this bot is relevant is because the bot is converting raw data in language humans will understand. The raw data will be converted by Natural Language

Processing for players playing the game to understand in terms of normal english. We are going to train on previous game data and decide which

plays were the best based on PPM or other achievements the player is getting.

The technology stack for this project will use Python modules such as NLTK and TensorFlow for the machine learning and the natural language processing side of the development.

Our technology stack is the best choice for our development in that we will use large amounts of data to mold the perfect/best output to the users' game. NLTK and TensorFlow are developed by Google which uses large amounts of data for training. This will be beneficial for our project with not as much data, but the training functions provided will work best for our goal.

For this project we are going to be using multiple developing assistant softwares which will make the workflow convenient. To write code we will use IDEs where coding guidelines can be set in stone. And for Git interface alongside a work assignment board we will use Gitkraken and Gitkraken Glow.

For this project we will use github as our main code base. This will ensure that each developer can have their code made available to the team all the time and also have a secure backup on a cloud server. We will maintain two main branches: Master and Developer. Our team will push to Master only during team meetings and at other times we will push to the developer branch. Apart from these two each team member will have their own branch which they will use to test their code and then update the developer branch.

We will also use a pull first and then push format which will ensure that we do not have conflicts and those that we do are checked before submission into Developer branch, by the developer himself.

We will follow a strict coding guideline sheet which we have written up. This guideline sheet will allow our code to be readable, reusable and documented at all times. We are trying to figure out if we can make these rules strict on an IDE like atom or sublime text our work would become a lot more reliable and better.

Scope and Timeline

Although this task can be a bit overwhelming to complete in the span of one month, we believe that with a perfect plan and timeline, we will be able to complete our project and evaluate the feasibility of this product. In the timeline given below, we have set every week's goal. However, we also plan to set smaller goals to ensure completion.

Week 1 (Monday, April 30 - Friday, May 4)	● Getting familiar with the gameplay and the data given by the api. ● Figure out how to perform a API request in Python to get desired data.
Week 2 (Friday, May 4 - Friday, May 11)	● Understand game mechanics and be able to figure out manually what important data is. Discuss and come up with strategies for machine learning algorithms. ● Figuring out battle phase and build phase of the game are and how they can be distinguished using what the API gives us. ● Figuring out how to tackle the case when both battle phase and building phase is going on simultaneously.
Week 3 (Friday, May 11 - Friday, May 18)	● Run over multiple replays to train the data. ● Find what we think is the time when he stopped building.. then we can comment on what happened in the building phase and summarize rather than just bluntly flushing data. ● Maybe we can find out what happens different in different games to figure out what leads to a win or a loss.
Week 4 (Friday, May 18 - Friday, May 25)	● While this building phase is going on, we can give a pre analysis of different strategies. ● We can evaluate the tradeoff between resource loss and power gain to figure out if it was a good strategy. ● Implement a Natural Language Generator to summarize the results in a more readable way. ● Prepare deliverables and perform rigorous testing.

Documentation and Access

We are planning to post our project on to github with a detailed readme. This readme should briefly talk about the problems we're trying to fix and why we believe that our solution fixes the problem. Then we will explain how we structured the project and how to effectively use the product. We haven't yet decided if we're planning to make a website or not since our end product is not something that can be explained better using a website.

However, we have yet to come up with an interesting way of presenting our product.

Evaluation

On severe research we found out that there are almost no bots that do the task that we have taken upon ourselves. The creators of the bots that exists haven't spoken in their documentation about evaluation and just claim to be “good” at commentating. In order to evaluate the success of this project, we will be looking at the outputs of our program over a small number of simple replays. Our evaluation will be based on the grammatical correctness of English sentences, that are the output of our program. At the minimum, we want our program to generate understandable sentences. We will create a scale between 0 and 100 where we will know 0 is below satisfactory and 100 is brilliant. The score the scale generates depends on a variety of things. The list includes but is not limited to things like, number of times same sentence was used, appropriate use of sentences at appropriate times. Hopefully are scale is more generalized and then we can create a far and evenly distributed.

Plan for Deliverables

As this project is not a StarCraft Bots, we won’t be submitting to a tournament, but we are planning to post this project on Blizzard StarCraft II API forum, which can be found here: https://us.battle.net/forums/en/sc2/22814033/

Separation of Tasks for Team

Implementing the interface code: Rohit and Hung
Strategizing effective ML algorithms to train data: Saksham and Yash
Implementation and optimization of ML algorithms: Yash and Rohit
Implementing a contextualizer: Aakaash and Hung
Implement a Natural Language Generator: Saksham and Aakaash
Testing and Code Review: Yash and Hung
Evaluating Github Workflow: Saksham and Aakaash
Inline Documentation: Aakaash and Hung
Deliverable Reports: Rohit and Saksham