**EECS 581 Team 12 – Initial Project Proposal**

**Matt Bailey, David Cuellar, Jacob Long, Manan Shah**

**10/23/2017**

**Team Name**: DeepHate.py

**Team Members and email addresses**:

David Cuellar david.cuellar@ku.edu,

Matt Bailey m229b487@ku.edu,

Manan Shah m406s889@ku.edu,

Jacob Long [j930l636@ku.edu](mailto:j930l636@ku.edu)

**Contact**: David Cuellar -- david.cuellar@ku.edu

**Project Description** (150-250 words)

Predictive Behavioral Modelling software that will integrate deep learning neural networks (MIT’s DeepMoji and IIT-Hyderabad’s Deep Learning for Hate speech identification network) with analysis of given user’s relationship networks from twitter and liked-page object network from Facebook to predict likelihood that user may engage in discriminatory behaviors and risk adverse outcomes to their employer.

The end result is a web-based application intended for use by HR departments and hiring managers (herein referred to as the customer) to screen the social media presences of prospective employees for indicators of hate speech/latent hateful ideology that may arise in unbecoming circumstances both within the company or in public view. By providing an automated, quantitative means for analyzing the social media of prospective hires, the customer would insulate themselves from legal risks of discovering protected information via social media screening, as would occur if such screenings were performed manually by the customer during pre-employment background checks.

**Project Milestones**

· 3-5 specific and measurable objectives per semester for first & second semester

Fall Semester.

1. Complete Dynamic Data-set generation scripts -- November 2nd
2. Finalize mathematical model used to make prediction from raw input data (Sets of tweets, user’s relationship graphs, facebook object graph) -- November 15th.
3. Initial web page diagramming -- December 3rd
4. Code review of DeepMoji and Deep Learning for Hate Speech ID LSTM implementations -- December 10th
5. Initial Backend diagram/design completion -- ~December 15th

Spring Semester:

1. Complete initial deployment of back-end -- January 8th
2. Complete front-end design -- January 8th
3. Connect Front and Back ends -- January 27th
4. Initial testing and model validation via front-end -- February 13th.
5. Complete final form front end GUI.

· Both implementation and documentation milestones

The Gantt Chart is shown in Figure 1 (1.0 and 1.1), with the complete chart available in its entirety at the following URL:

<https://www.tomsplanner.com/public/eecs581-team12>

The Gantt Chart is password protected – see proposal submission for password.

Figure 1.0 -- Gantt Chart

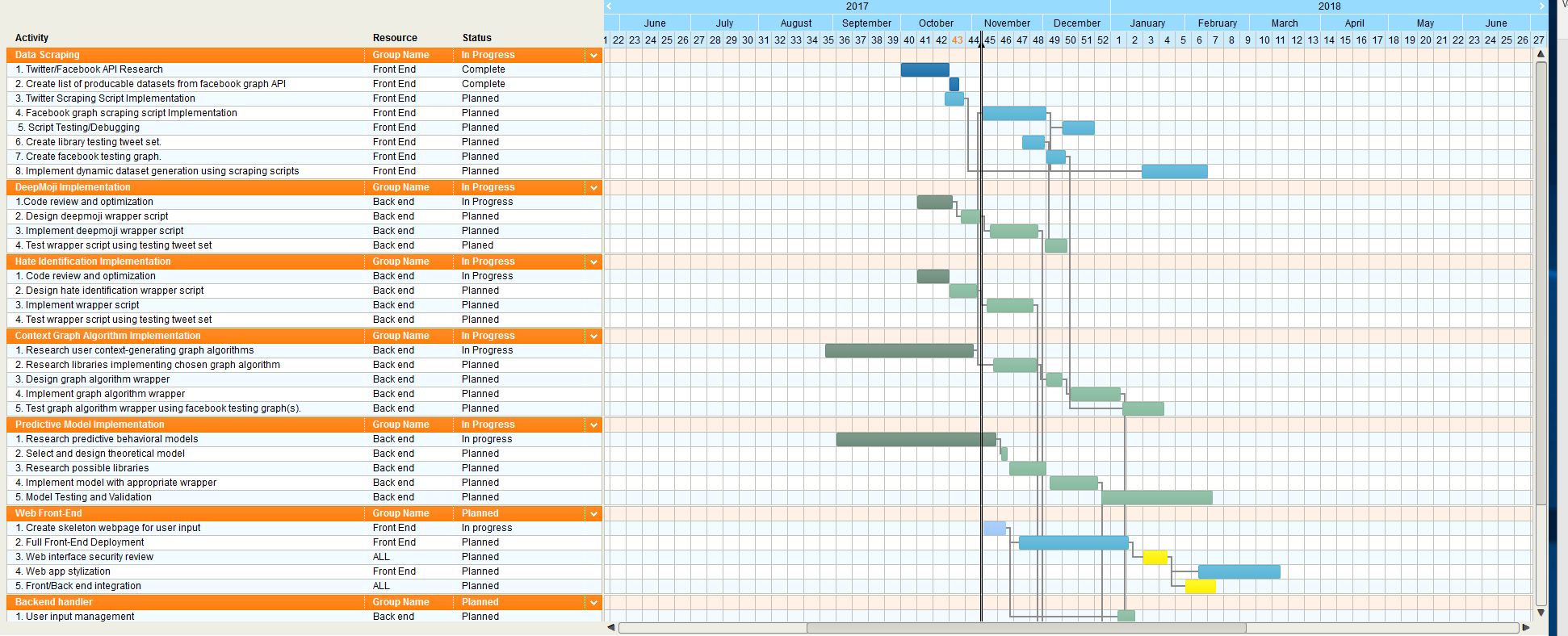
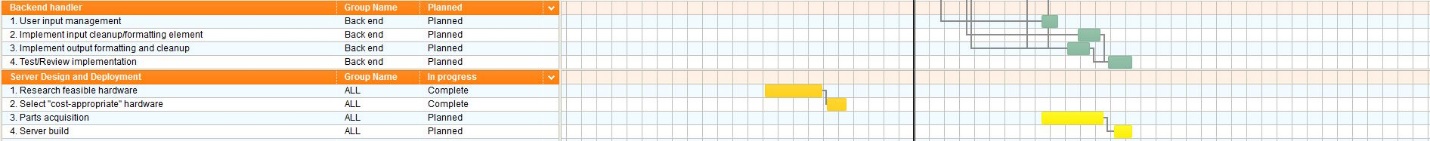


Figure 1.1 – Gant Chart Continued

****

**Preliminary Project Design**

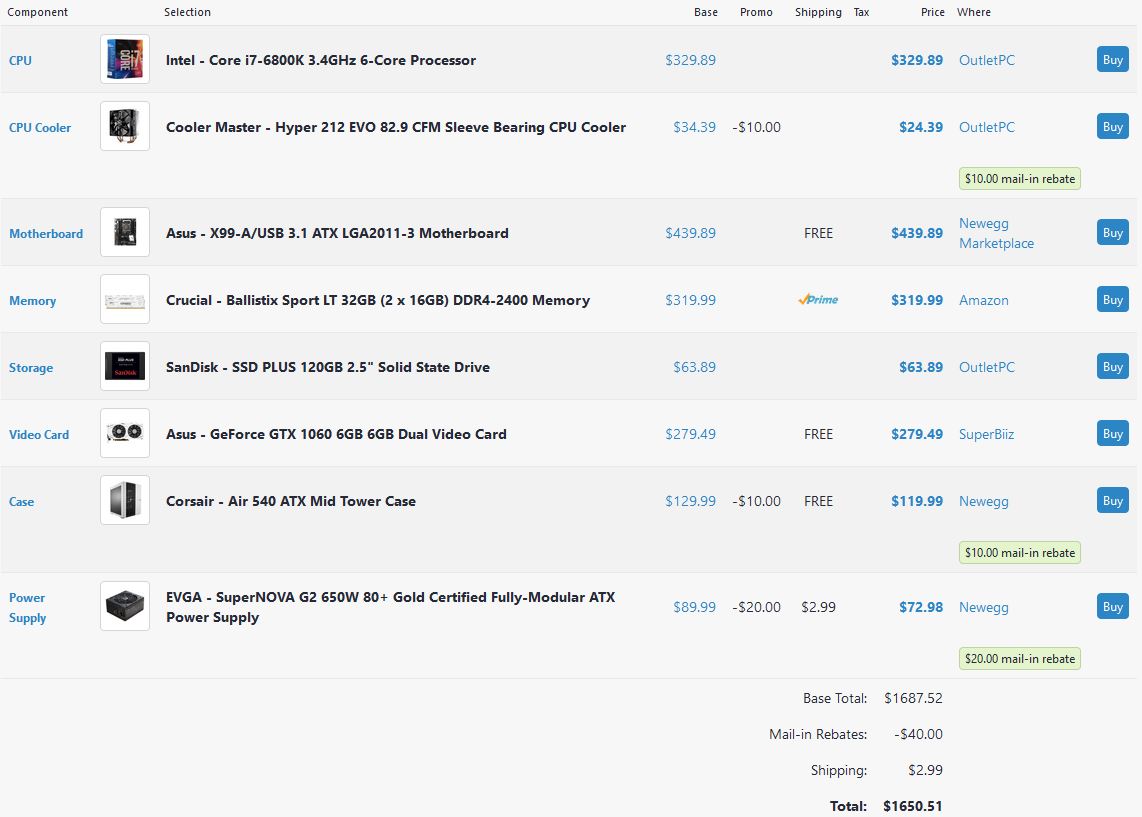
DeepHate behavioral analysis tool is best understood when broken down in terms of hardware and software. The complexity of human behavior and natural language necessitate the use of resource intensive techniques from artificial intelligence and data mining. This complexity stems from two sources – the very large number of variables that could potentially affect human decision-making both in general and in the interpersonal domain, and the difficulty disambiguating sentiment from language due to things like sarcasm, slang, or other forms of obscured semantic embedding that eludes accurate description and classification in many natural language processing approaches. To deal with this complexity, DeepHate aims to create a “behavioral profile” of a prospective job candidate by gathering all publicly available tweets by the candidate and analyzing them using MIT’s DeepMoji pre-trained neural network, IIT Hyderabad’s Hate Speech Identification neural network, and a graph analysis algorithm on one or more of the candidate’s Facebook graphs; the output of each of these algorithms will be fed into the predictive model to establish a sense of how the user views or responds to racially inflammatory events or other flashpoints of social division.

From the top-level description of DeepHate, it quickly becomes apparent that these algorithms are very resource intensive; custom hardware is necessary to deploy such a product to perform the analysis at a rate that is acceptable to our intended end-users at a reasonable cost. The custom server for DeepHate would leverage the computational power of multiple GPUs to reduce the time needed to perform the described social media analysis on a given candidate. Using GPU’s for deep learning and graph analysis is standard practice in the artificial intelligence and network analysis fields; however, it is important to note that our desire for multi-GPU custom hardware comes from the marginality of performance gains via parallelization of such deep learning approaches. Our research into designing deep learning software showed that the performance gains from parallelizing a deep learning neural network is negligible and not worth the effort; parallelizing requires deep knowledge of GPU chipset architectures as well as the performance/design of the deep learning libraries used to implement the algorithms. The small gains in performance result from communicative bottlenecks between the GPUs; if they implement a deep learning algorithm in parallel, our research source found that they spend a substantial amount of clock cycles waiting rather than computing, as the communication channel between GPU’s becomes a bottleneck on computation due to the GPU’s FLOPS rate being much higher than the bandwidth of the GPU communication channel (usually a network card in parallelized GPU clusters). This bottleneck contributes to the infeasibility of Amazon’s AWS compute instances as a possible deployment solution, though that will be discussed in further detail later. Moreover, since the Twitter API restricts the number of tweets that can be pulled for a given user during a given time interval, it is doubtful that the size of our tweet input sets will greatly benefit from any parallelization of deep learning algorithms. Instead, our need for multiple GPU’s comes from the diversity of social media analysis approaches used by DeepHate. The performance gains from multiple GPUs arises from the ability to perform multiple resource intensive computational tasks simultaneously by running the algorithms in parallel, rather than parallelizing single resource intensive algorithms across multiple GPU’s and repeating the process for each algorithm.

At this juncture, it may appear as if AWS GPU instances would be an adequate backend platform for DeepHate, though the cost structure of AWS GPU instances combined with minimal benefits of parallelization and risk of GPU communications bottlenecks preclude its use. First, the AWS pricing structure is extremely inefficient for our purposes at the time of writing; GPU instances can be purchased in either single GPU, 8 GPU, or 16 GPU varieties with 61 GB of RAM per GPU. The relatively small input sizes of the two deep learning tweet analysis elements of DeepHate, make the most affordable instance (single GPU instance) adequate for using DeepHate once it is complete due to its price of $0.9 per hour of usage, if the graph analysis algorithm consumes resources on a comparable order of magnitude. However, the currently undetermined nature of the graph algorithm module makes it difficult to make a more thorough prediction of feasibility of this specific AWS instance. If DeepHate were implemented on an 8 GPU AWS instance, such implementation would cost $7.2 per hour, making testing, model validation, and model training extremely expensive, as tweaking predictive behavioral modeling parameters is generally considered a computationally long and laboriously tedious process. Furthermore, this GPU instance would provide much more resources than needed while adding the implementation and development constraints associated with AWS instances, including the availability of such instances in Amazon’s data center(s). The costs of such instances could be lowered by scheduling them in advance, but that would create a substantial amount of pressure to stick to our development schedule, which would require assuming that nearly everything will go according to plan. This assumption is untenable in software development in any instance.

The proposed custom server build was designed with minimal resource use in mind; we leverage the known limitations in the size of two input data sets in order to use older GPU’s that cost less than the current generation of GPU chipsets. This choice of suboptimal hardware may potentially slow down training or model validation slightly as our preferred GPUs have only 6 GB of RAM, the 24-hour availability of the server upon its construction provides ample opportunities to perform model training and validation without concern for training time in the event that GPU memory is exceeded and additional code would be needed to manage the memory “overflow” if it is not swapped to CPU memory or the disks swap space. The rough parts list is given in Figure 2.

Figure 2 – Hardware Cost Table

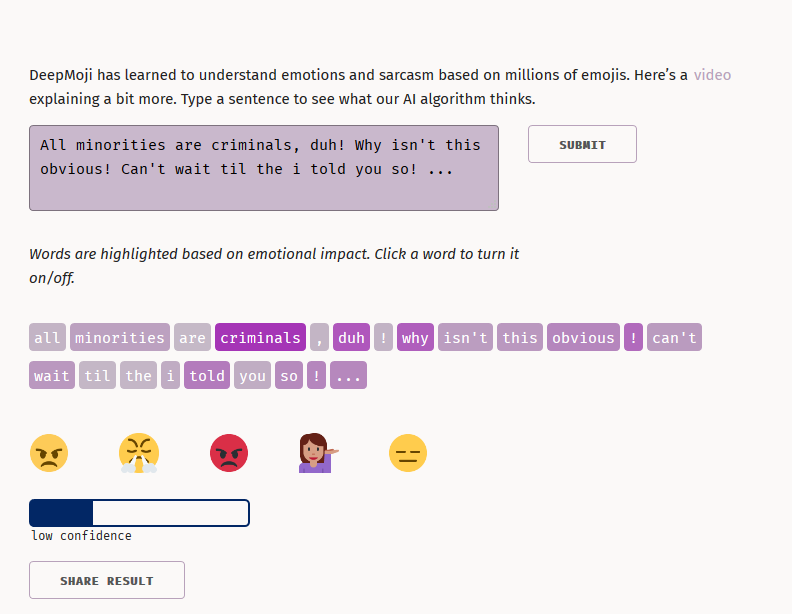
****

Note, the final cost of hardware would be on the order of magnitude of the shown total, with some fluctuation due to part sourcing, as some components can be purchased at a discount on eBay or other retailers that deal in used/refurbished hardware. Further, this total includes the price of a single GPU, though roughly four will be needed for project implementation.

In terms of software, DeepHate consists of seven primary components: web application interface, backend handler, data scraping module, DeepMoji neural network module, hate speech ID neural network module, graph algorithm module, and the predictive model module. The web application interface is the front-end through which the end user passes a job candidates information (Name, Location, Phone Number, e-mail address, twitter handle, and education) and receives the behavioral profile generated by DeepHate. The backend handler deals with the generation and pre-processing of candidate Twitter and Facebook graph datasets by passing the candidate’s information to the data scraping module, which identifies the candidate’s Facebook userID before scraping their Facebook page to create a liked-object graphs (other graphs are likely to be used in the graph analysis algorithm, but at this design stage, liked-objects seemed to be the most obviously usable graph) using the Facebook Graph API. With these datasets generated, the tweet vector of length n and the Facebook graphs are passed to DeepMoji, Hate Speech ID, and Graph Analysis modules, respectively. DeepMoji analyzes individual tweets and encodes the sentiment of the tweet in the form of a vector of 5 emojis using a long short-term memory (LSTM) neural network, a type of recurrent neural network architecture; the DeepMoji module would take the tweet vector as input and return an n x 5 matrix of integers, with each integer being mapped to a corresponding emoji. Hate Speech ID takes a tweet and classifies it as racist or sexist, thus the output of the Hate Speech ID module would be a 2 x n matrixof integers, with the first entry being 1 if the tweet is racist, 0 otherwise and the second entry being 1 if the tweet is sexist, 0 otherwise. The graph analysis algorithm would take one or more of a candidate’s Facebook graphs and generate a behavioral context of the candidate that will be used to disambiguate how they relate to the sentiment expressed in one of their tweets, given a particular classification. It remains uncertain which specific graph algorithm will be used for this purpose, but some examples include subgraph discovery or clique discovery as means of community identification. For instance, a person that does not espouse White Supremacist ideals is unlikely to like the Facebook pages of the KKK, David Duke, and Stormfront; if these elements appear as nodes in a subgraph of a given candidate, it provides insight that can be leveraged to predict how a user interprets the “affective space” carved out in a Tweet and its associated cultural context. The output of such community identification algorithms would depend on the implementation, but at this moment, I am certain it can at least lead to a proposition: whether or not a candidate has a hateful subgraph.

Since the graph algorithm is currently unknown, a detailed explanation of the predictive behavioral model is not possible; however, the overarching idea can best be characterized by an analogy. Imagine you see a tweet as follows: “All minorities are criminals, duh! Why isn’t the danger obvious to y’all?! Cant wait to say I told you so! … ” From verbal inspection, this could be either a genuine racist sentiment, or a sarcastic statement from a minority, mocking how they are represented at a particular cultural moment or in a specific situation. The resulting emoji output vector from DeepMoji is shown in figure 4.

Figure 4 – Deep Moji Tweet Sample

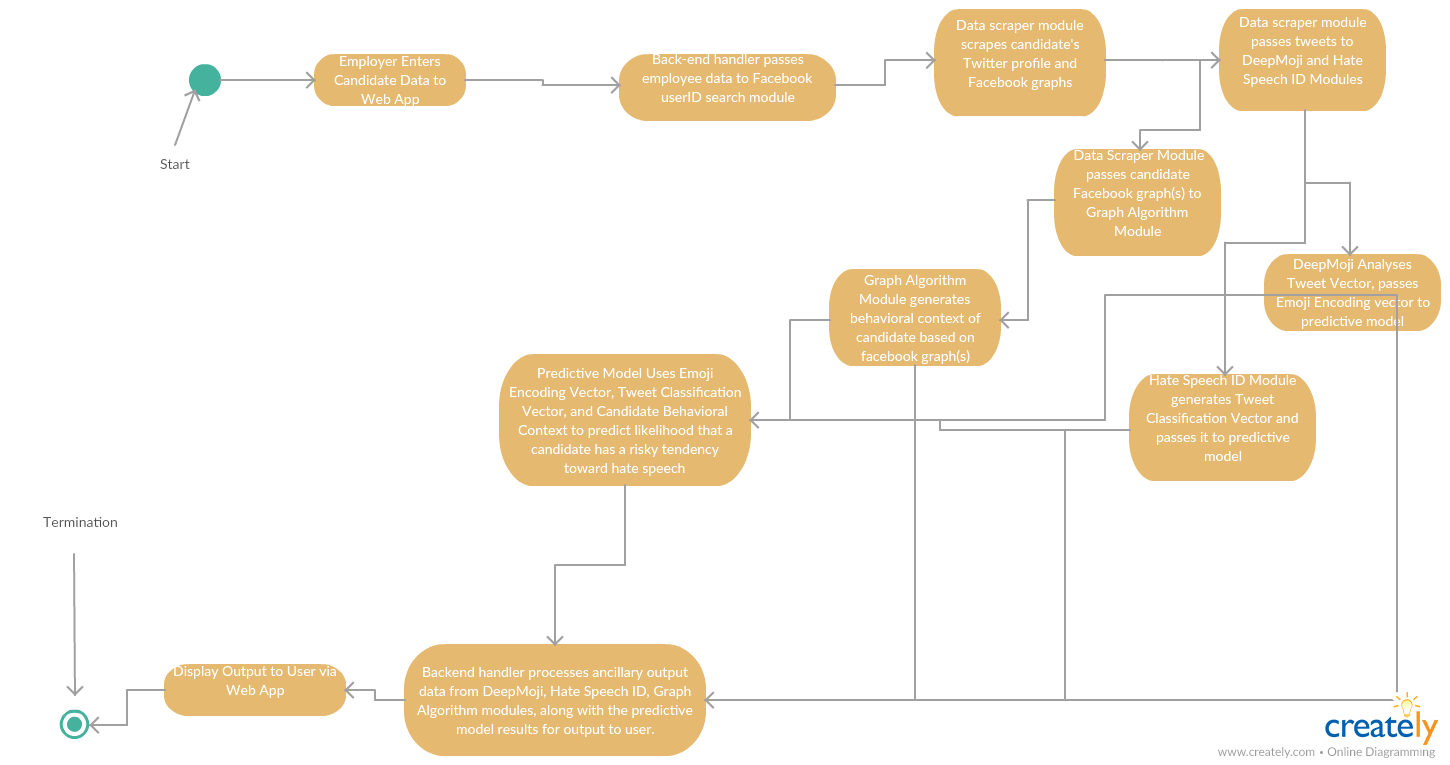


The emoji vector and confidence level given here do not accurately explain which of the two possible cases informs the tweet author’s sentiment. The first three emojis clearly correspond with the sort of frustration felt by either a person that feels dismissed due to their views on race (the racist sentiment) or a person that is demoralized by discriminatory social elements to the point of sarcasm (the sarcastic minority sentiment). The final two emojis correspond to the sarcastic subtext of both possible sentiments. This exemplifies an instance of noisy data that DeepMoji’s creators describe as difficult to accurately classify and use. These noisy cases are resolved by the graph algorithm. If we take the graph algorithm to be a simple subgraph discovery looking for edges between the candidate and Stormfront, KKK, David Duke, and a handful of other common White Supremacist social media outlets, then the discovery of such nodes and edges in the candidate’s graph would reveal that the tweet did indeed arise from the racist sentiment case. Such analysis would be performed in the predictive behavioral model. At this moment, the exact mathematical technique for relating them remains to be seen, though the naïve solution would be taking a dot product of the two output matrices multiplied by a weight determined by the output of the graph algorithm, though this approach may make too many assumptions about the data. Further research is necessary to determine a specific mathematical function to relate the three objects.

The output of the behavioral model would be combined, by the backend handler, with ancillary output data from DeepMoji and Hate Speech ID, such as the number of racist/sexist tweets, the number of racist/sexist tweets paired with a positive/approving emoji encoding vector, “red flag” nodes in the graph such as Stormfront or the KKK Facebook pages, and so on, to generate the behavioral profile to be given to the end user via the web portal.

The relationships between such components and their functions are outlined in the activity diagram in Figure 4.

Figure 4 – Activity Diagram

****

**Ethical and Intellectual Property Issues**

The intellectual property issues of DeepHate remain to be seen; as of writing, both DeepMoji and Hate Speech ID neural network libraries are licensed under licenses allowing for unrestricted public use in other products or capacities. This may change depending on what is used to implement both the graph analysis algorithm as well as the predictive behavioral model.

The ethical implications of such a software tool are a bit more complex. Some could argue that it persecutes thought crimes on behalf of prospective job candidates. However, the mathematical rigor of DeepHate and other behavioral models mitigates claims of persecution. Further, such claims ignore the fact that all material gathered and used by DeepHate results from an active, conscious decision by the job candidate to associate themselves with a particular type of web content; absent a compromised social media presence, an individual reacts to tweets, crafts tweets, and responds to Facebook objects in ways that align with how they wish to represent themselves. Thought crimes in the traditional sense are not associated with behaviors that consciously curated like the social media presence of modern professionals.

**Project Budget**

1. Monthly amazon web service -- in the event that server construction is too cost-prohibitive, AWS may be necessary, assuming that AWS instances exist that have compatible versions of tensorflow -- more research is needed to project monthly compute costs.
2. Costs to create server are listed in hardware table (Figure 2).

Meeting time with TA (Amir):

Monday Lab -- 11:00am- 12:50pm.

Work Plan

Front End/Web App

Jacob, Manan

Neural Nets & Graph Analysis design/implementation

David, Matt.

Github link:

https://github.com/dsca1729/581\_team12\_17-18

**Change log**

No substantial changes to the initial project description were made; however, the Gantt chart does not exactly match the dates specified in the initial project description. This is a conscious decision, as the initial project description was written with little knowledge of the groups availability and willingness to work on the project during winter break. The predicted start/end dates are somewhat flexible, since at least three of us will be working on the project over winter break. The dependencies and workflow are unchanged, however.