MyMeMe

ESE Senior Design 2018-2019: Team 3

May 6, 2019

Anish Jain (SSE) - jainani@seas.upenn.edu
Saku Rimali (SSE) - srimali@seas.upenn.edu
Darius Rodgers (SSE) - darodg@seas.upenn.edu
Devan Tierney (SSE) - dtierney@seas.upenn.edu

Advisor - Dr. Rakesh Vohra - rvohra@seas.upenn.edu
Team Email - eseseniordesign201819team3@gmail.com

Table of Contents

II. Executive Summary

Memes are becoming premier sources of information, especially for Generation Z. A meme is a form of comedic expression through digital content, typically seen as a picture, gif, or video, accompanied by text; they are fun, quick, and meant to be shared. According to Forbes (2019), "[m]emes, have become one of the hottest communication forms for Millenials". People find enjoyment both when looking at funny memes themselves and sharing these memes with their friends. These memes are transmitted through channels such as Facebook Messenger, email, social media, text, meme apps, etc. Across these channels, memes congregate themselves in groups such as focused Instagram accounts. But with so many accounts out there, and not much linkage between similar accounts, a person is only viewing a small subset of those memes that are ideally meant for them. Thus, it often takes a person much "meme hunting" before finding one that satiates his/her sense of humor.

Memes have taken social media platforms, which have essentially become glorified meme hubs, by storm over the last few years, with users spending more and more time on popular destinations, such as Facebook, Instagram, Twitter, and Snapchat, just looking at memes; much of the value of these platforms is captured in the "stickiness" of their users. There is a natural fragmentation in the channels that users utilize to receive this content, as there is no centralized meme source but instead "niche" humor groups scattered across social media. Enter MyMeMe, the "Tinder for Memes". Through the use of machine learning, particularly clustering, we decided to create a system that identified different "senses of humor". By continually learning which senses of humor each user prefers, the system can deliver memes that are better tailored to each individual user.

MyMeMe is a meme curation platform application where users have a personalized meme feed and input either a "like" or "dislike" for every meme they see. Based on the attributes of the memes that the user likes and dislikes, the platform learns what sense of humor the user has and then better curates for and presents memes to him / her. Each meme and user can be defined by different levels of these attributes; specific attributes used to identify memes include political correctness, corniness, recency, and intended age, among others. These attributes create a new kernel space where senses of humor can be identified when clustering. Clusters of memes are calculated based on their attributes. A user sees and either "likes" or "dislikes" a first "training" set of 20 random memes from the meme feed on the user interface, and the user's attributes are subsequently updated upon every action. When this happens, the distance between the user's attribute vector and the centroids of each of the meme clusters is calculated. The closer a user's vector is to a particular meme cluster, the more that memes from that cluster are shown to the user in the user's next set. Subsequent sets have 15 memes. Over time through this iterative process, MyMeMe learns each one of its users, providing them with an optimized meme consumption experience as the memes become better-curated with each presented set. As user attributes converge, the users' decisions will be used to assign attributes to the newly entering memes in the meme ecosystem, creating a self-ranking system so that external rating is not necessary (this feature has not yet been implemented). The iOS user interface is shown in **Appendix I.**

III. Overview of Project

The objective of this project was to create a smartphone application that implements a meme database, utilizes clustering and suggestion algorithms to provide better-tailored content to users over time, and uses a "like-dislike mechanism" with a user preferences algorithm to assign attributes to users and then update these attributes; this has been accomplished. Specifically, we wanted to achieve fast delivery of memes and dynamic user profile updating, create an extensive meme database, and

achieve a 60% long-term “like-swipe ratio” ratio; we achieved all of these except for testing the “like-swipe ratio”.

MyMeMe solves five main problems with meme consumption: classification of memes, the presence of multiple suboptimal platforms, overload of content, short attention span of users, and lack of structured sharing. Since a meme is essentially just a picture with some text on it, it is difficult to classify. Thus, it often takes a person much “meme hunting” before finding one that satiates his/her sense of humor, and these “senses of humor” are extremely difficult to classify in simple terms. This presents a problem of inefficient searching for enjoyable memes and content. Users find information on multiple platforms, including Facebook, Twitter, Instagram, and Snapchat, all of which use feed systems that are overloaded with content posted and shared by other accounts; memes are decentralized. With an increasing amount of this stickiness of major social media platforms residing in memes, we saw a tremendous opportunity in creating a platform to optimize the meme viewing experience. People spend millions of hours per day swiping, scrolling, liking, and commenting on memes every day. However, people generally have short attention spans when browsing content and are prone to get bored, distracted, and / or frustrated while searching for memes. With advanced technology and Internet browsing available at their fingertips, people may just give up searching for memes and engage in another activity if they are unable to find ones that appeal to their unique senses of humor. There is also no structured way of sharing memes, with most being screenshotted and then sent in group chats. In summary, MyMeMe uses the cutting edge of data science to help inch closer to perfect matching between users and memes, providing a centralized meme feed with curated meme delivery, and it hopes to include a structured service for sharing memes in the future.

With its extensive meme database and dynamic user and (eventual) meme profiles, MyMeMe increases user entertainment from meme consumption while simultaneously decreasing time spent searching for memes, overall providing more concentrated enjoyment. We believe a web service or smartphone application with a “like” / “dislike” swiping mechanism and constant feed of memes, providing better-curated memes to individuals over time as they provide feedback, is the most feasible solution to improve meme consumption. Users have previously shown interest in this swiping activity, as shown by the popularity of applications like Tinder, Bumble, and Hinge, and memes are more prevalent in society than ever before.

MyMeMe targets young people (we have defined the primary market as those under age 25, but it is not just limited to these individuals) who regularly use smartphones and social media. Users will thus have a meme feed where they can “like” and “dislike” memes one at a time. Through the use of machine learning based around identifying distinct “senses of humor”, MyMeMe will learn about users over time and provide more and more tailored content, removing the extra, unnecessary channels users currently have to scour to find content that appeals to their taste. This not only saves the user time, but also increases utility gained per unit time the user spends on our app, as he / she is receiving more personalized content, and what is not relevant to them is filtered out. In addition, the application will eventually include social aspects, hopefully allowing users to post their own memes, send memes to and message friends, and search for memes. MyMeMe will become the ultimate social media destination for meme consumption and distribution.

It is important to note that there are no applications on the market that offer self-improving meme distribution, as most people have resorted to more-tailored Facebook groups and forums like Reddit. The value-add of the consumer-facing side of MyMeMe is primarily gained from individuals who may like many types of memes, and an algorithm based on meme attributes has been designed to hit these many “senses of humor” well and deliver content that is appealing to the specific users.

IV. Technical Description

Our project consists of a meme database (500) with ranked attributes, a clustering algorithm, a meme suggestion algorithm, a user preferences updating algorithm, and a user interface iOS application. All of these components have been completed, and the technical specifications of each are discussed in more detail below. For a final, commercially-viable system, we would also need to add the adjustment of meme attributes based on user interactions, but this is left for future development. **Appendices II and III** display the block diagram and solution implementation progression. We are able to suggest memes and update user preferences in under 100ms, and the user interface is instantly responsive. Our main constraint is the fixed meme database and the current basic, less aesthetically-appealing user interface design. Google Drive is used to store the memes, Microsoft Excel is used to assign ranks to the meme attributes, iCloud and CloudKit are used to implement the meme database and store meme and user data, Python and the Pandas library are used to implement the clustering and suggestion algorithms, and the iOS application user interface and backend are coded with XCode and Swift (**Appendix IV).**

At the heart of the system are meme-attributes that we predict to be good differentiators of different “senses of humor”. Each meme is defined by different levels of these attributes. These defined memes are shown to users in small sets on our user interface, and they are able to either “like” (“treasure”) or “dislike” (“trash”) the meme. When users make decisions, their preferred attributes are updated based on their reactions (“like” or “dislike”) to the memes and their attributes. A like makes a small positive change and a dislike a small negative change on matching attributes. This is essentially done by computing the updated average of attributes of all the memes a user has liked / disliked. A user’s preferences / own attributes are used to find the distance between him / her and various clusters of memes. The clusters closest to the user will have a proportionally higher share of memes shown to the user in the next set of memes. The first set of memes are defined as the “training” set of memes, and they are randomly selected from the database and used to define the initial user attributes to suggest better-curated memes in the future. Over time, as user attributes converge, the users’ decisions will eventually be used to assign attributes to the newly entering memes in the meme ecosystem, creating a self-ranking system so that external rating is not necessary. The system design is shown visually in **Appendix V**.

Some societal concerns arise with young and content-sensitive individuals using the application. There should be an age limit to use the application, or at least some content should be restricted from very young and self-identified sensitive users, just like in other forms of media. If the application became widely adopted, the number of users and memes would rise exponentially, and we would also have to focus on computational limitations and processing efficiency. However, at this stage of development these are not primary concerns. There were no major environmental or economic concerns that influenced the design.

The memes are currently stored in a Google Drive folder holding 500 memes. Paired up with the folder is an Excel file that has a row for each meme and a rating assigned to each attribute. Some of these attributes are scalar on a scale of 1-5 or 1-3, and some are binary variables. They group memes based on common factors. Sample entries for the database can be found in **Appendix VI**. The memes and user attributes are also stored in iCloud to integrate the data with the application.

The current suggestion algorithm utilizes K-Means clustering to group all memes in our database. We use *KMeans* function included in the *sklearn* Python library for our clustering. The full code for data cleaning and clustering can be found in **Appendix VII**. These clusters are supposed to mimic “senses of humor” different people possess. The algorithm samples memes from multiple clusters that are close matches to the user measured by the Euclidean distance (L2 Norm) of the centroid of the cluster to the user’s own preferences. Reference **Appendix VIII** for our suggestion algorithm.

The user will swipe on 20 random memes from the database to define his / her individual preferences. At this point the user will be given their first recommended set of 15 memes. This set is calculated by first measuring the Euclidean distances between each cluster center and the user's preferences. The proportion of memes from each cluster in the set is inversely proportional to the respective distance to the cluster center. This allows the set to contain the most memes from the closest cluster, the second most from the next closest cluster, and so on, with least number of memes coming from the furthest cluster. Proportions are calculated by dividing the sum of cluster-preference distances by each individual distance. These proportions are turned into percentages by dividing each proportion by the sum of all proportions. They are then scaled by a factor of 1.5 and rounded up to integers. This gives us the number of memes from each individual cluster to be placed in the recommendation set totaling to 15 memes. X number of memes are randomly selected from each cluster, where X is the number of memes from that cluster, previously defined. If the user's recommended set contains a previously seen meme, instead of showing that meme, they are shown a random meme (not previously seen) from the database. A new recommendation set is generated every time the previous recommendation set is empty with the user's updated preferences.

The software also includes a dynamic framework for adjusting user preferences based on user interactions. When users make decisions, their preferred attributes are updated based on their reactions to the memes and their attributes. A like makes a small positive change and a dislike a small negative change on matching attributes. This is essentially done by computing the updated average of attributes of all the memes a user has liked/disliked. Our user preferences updating algorithm is included in **Appendix IV.**

The only way to test how well our clustering algorithm captures the "humor preferences" is letting real users test the product. For this purpose, we have created an iOS application connected to back-end processing. The interface allows new users to create accounts in order to view memes. The app is connected with Apple's iCloud to save meme and user data. This allows access to both private and public databases for user protection and for accessing the like ratios. We have also implemented an auto-login feature if the user closes the application without signing out. There are two main viewing windows. The first shows user information and allows the user to see his / her preferences, number of memes liked, and current like ratio. This is mainly a placeholder for a window in which the user can manage his / her account and make customizations. Once the social aspect is integrated, here users would be able to add friends, join groups, send messages, etc. The second viewing window is the main screen for interacting with content. Currently we just have like and dislike buttons (represented with "treasure" and "trash" icons, respectively). Eventually swiping will be implemented as originally planned. Using these buttons will change the meme in the viewing window, save data to iCloud, update user preferences, and create the next recommendation set if necessary. Additional features will be implemented in this window to enhance user experience. Users will be able to save memes, read/write comments, and share memes through various methods.

The effectiveness of the most important component, the suggestion algorithm, can be measured quantitatively by comparing the "like-swipe ratio" obtained from a set of 15 random, uncorrelated memes to that obtained from a different set of 15 memes that have been generated by the algorithm and suggested to the user. In theory, the user will "like-swipe" these suggested memes at higher frequencies than memes that were randomly generated; from this user data, the ratio should improve over time as user preferences are further specified. Our long-term target ratio is 60%, which is much higher than the typical "like-ratio" of content provided by existing platforms such as Facebook or Twitter.

Due to none of us having any previous mobile application development experience, the task of building the backend of our beta application took us longer than expected. We completed the MyMeMe beta just before final demo day. We were not able to post it to the Apple App Store at this time because

of two reasons. The first is the app was not working correctly for every iPhone device. We had to create some buttons programmatically, instead of utilizing XCode's Storyboard. Because of this, the constraints for these buttons worked differently on each device. We could not find a quick solution after the completion of the app. The second reason involves the App Store's requirements for posting applications. We found that gathering the information needed and going through the process of validating our software would take too long. We would need more time to solve these two issues before having the ability to allow for meaningful testing.

In total we had four design iterations. In our first design iteration, we used R to create and analyze the clusters. Using these results, we refined our attributes and built our second iteration that included a clustering and a suggestion algorithm in Python, as well as a simple user interface. For our third design iteration, we built a fully-functioning iOS application implementing a meme feed, "like" and "dislike" interactions, and login features. We used this application to test the functionality of the concept and decided on necessary features to add for the next iteration. For our fourth and final iteration, we added features to show the users their current preferences in the application, how many memes they have viewed, and their like-swipe ratios.

We considered a number of alternatives for our system. First, we needed to classify the memes in order to be sorted for recommendations. We looked at similar projects that used text and picture classification. We did not feel that something like PCA could effectively capture senses of humor, so we decided on numerical attributes and iterative clustering for our classification method. Our next alternative involved defining user preferences based on interactions. We proposed using a scaling system to define preferences based on attributes. This was not an ideal method because we wanted to implement the swiping feature for fast user interaction. We believe binary interaction was more user-friendly and captured preferences better. Our last design alternative was in the consideration of a platform for the beta application. At first, we considered a web application, but this did not seem useful, as most people look at memes on their smartphones. We then had to decide if we wanted to make it available for Android and iOS devices. Ideally, our application would be available on all platforms. To save time, we decided just to make the beta available for iOS. This also allowed us to use iCloud for our backend, as it is integrated in XCode already.

In conclusion, we use meme attributes saved in an Excel file to cluster memes with our Python clustering algorithm. We then use the clustering results to suggest a set of memes from different clusters with the amount per cluster inversely proportional to the distance to the specific user's preferences. We show these memes in our iOS application and give the user the option to either "like" or "dislike" the meme. Based on the user's "likes" and "dislikes", the user's preferences are updated, and the next set of memes suggested will be even closer to the user's true preferences. Over time, the user preferences converge to the user's true humor preferences and the user receives highly tailored content in our application, increasing the "like-swipe ratio".

V. Self-Learning

The skills required for this project mostly extend to areas in which group members have acquired expertise through studies and internships. One of the core aspects of the project involves the machine learning framework, which applies primarily to the meme suggestion and rating processes. The basis of our machine learning knowledge was acquired through the course *ESE 305 - Foundations of Data Science*, which both Anish and Saku completed as juniors. Anish and Saku also completed internships last summer closely related to data science, which further honed their knowledge. Using our understanding of different unsupervised learning methods, we decided that clustering would be the best way to approach our meme classification problem. This previously obtained knowledge and skill set also enabled us to test and compare different clustering algorithms and methods.

In our first design iteration we used R for the clustering, about which Saku had gained knowledge through internship experience and *STAT 520 - Applied Econometrics I*, an econometrics class he took last semester. For our second design iteration we moved completely to Python, with which Darius and Anish are also familiar through coursework and personal projects. Our clustering algorithm still runs on Python with our front-end moved to an iOS application.

For our last design iterations, we produced an iOS application, for which we had to develop skills in the two areas of the project in which we had no experience: iOS application development and user interface design. Darius taught himself how to use CloudKit, XCode, and Swift to implement the iOS application. He also learned how to integrate iCloud, which stores the meme database and user information, and the suggestion and clustering algorithms within the application.

VI. Ethical and Professional Responsibilities

This application will help people find memes more tailored to their senses of humor at faster rates than before. MyMeMe would potentially increase meme enjoyment while simultaneously decreasing time spent searching and consuming memes, since the meme distribution would be more targeted. This would further perpetuate memes as primary sources of comedy and possibly information sharing in society. As smartphone and Internet usage around the world becomes more widespread, more people could be exposed to, enjoy, and potentially learn from memes. MyMeMe could revolutionize the advertising industry, which could implement meme usage and distribution into their budgets, and knowledge dissemination in general. These advertisers and big brands do not currently utilize this medium of communication with potential customers. Exposure to memes may bring previously unexperienced entertainment to many different people, from the young to the elderly and from students to professionals. As will be discussed in the ethics component of this section, this content may also be disturbing to some individuals, some it would be optimal to have some level of filtering for users.

There are multiple ethical issues that pertain to this project. The main one involves data privacy, since the business model revolves around displaying ads to users. Personal user data (possibly including Facebook and Twitter data) will eventually hopefully be collected when one creates an account, and this data should not be manipulated by those who have access to it or mistreated when being sold / accessed by third parties. This will be addressed by not selling data that can directly identify individuals. Moreover, in the current iteration of the project, user data is not gathered from external sources, and the only data being collected is defined user preferences based on swiping the memes, not even on what memes the user swiped. Also, the question of whether or not memes that are in the database are appropriate for all users is something that needs to be considered. While some individuals may prefer more “savage” or “insensitive” memes, others will not and may be offended by the text and images; our team has to respect the preferences of both sides of the spectrum. This presents the biggest issues during the trialing phase, when users are exposed to memes of various different kinds, and when new memes are being added to the system and thus need to be assigned attribute rankings. The trialing aspect can be avoided by not showing memes that are “extremely savage” or potentially disturbing during this time, and these will only be shown in the future if the user liked the “savage” memes in the beginning set. Once memes are defined, an age and / or “savage” filter can be used to avoid showing these memes to young people and those who do not find them funny, respectively.

As users attributes converge from increased usage, they will be exposed to more- and more-similar style content. One may argue that this effectively creates an “echo chamber” for and consistently reinforces the individual’s sense of humor (similar to those for political beliefs), but we

assert that the purpose of this application is to repeatedly appeal to specific senses of humor, not to expose to / educate about many types of humor.

VII. Meetings

Our sole advisor is Dr. Rakesh Vohra (rvohra@seas.upenn.edu). We email Dr. Vohra on a weekly basis and meet if necessary to discuss our progress and ideas. We send him most of the material that we develop for the class, from the Demo Day Poster to our Storytelling Presentation to our MVP Demo Slides. Dr. Vohra helped us initially decide which one of our three projects to pursue ("Tinder for Memes" vs. "Uber for Public Transport" vs. "Fantasy Football Predictor"), which initially ended up being the fantasy football predictor, but upon reviewing existing prediction services, we believed that we could not design a superior product and pivoted back to the meme curator, which he supported. Dr. Vohra also mentioned gathering user data, but we assured him that many students would be interested in testing out implementation because of memes' widespread popularity among college students and the addicting "like / dislike" aspect of the project. Dr. Vohra is still advising our team.

We did not meet with subject experts or consultants, but we do often discuss our ideas with other college and high school students, who generally are very intrigued with our application because of the widespread acceptance and appeal of memes among the youth.

VIII. Proposed Schedule with Milestones

The original projected spring semester schedules from the Fall 10.24.18 2-3 Minute Presentation can be found in **(Appendix X)**, the first updated spring semester schedule from the Fall 11.30.18 4-Minute Presentation can be found in **(Appendix XI)**, and the finalized schedule from the 2.8.19 MVP Demo can be found in **(Appendix XII)**. Green boxes indicate that the assignment was completed on time, blue boxes indicate that the assignment was in-progress, and the red boxes indicate that that assignment was not completed. We decided to revise and simplify our schedule at the 2.8.19 MVP Demo to just include tasks directly related to our project implementation and major project milestones (Demo Day and Final Report / Video), discarding the executive summary entries (since these were not necessary anymore) and updating Dr. Vohra entries. We mostly followed the plan according to the 2.18.19 MVP Demo, turning in all class assignments (presentations, videos, etc.) on-time, emailing Professor Vohra on a weekly basis, and staying on-track with refining the algorithms and developing the iOS application, which was completed for the 3.30.19 2-Minute Demo Video. That being said, we did not gather a large sample of user preference data for ranking each of the memes. This could have been accomplished by sending surveys to many different individuals and having them rank the memes themselves, instead of having just the four members of our team rate the memes. We decided not to do this and instead focus on developing the iOS application robustly this semester for proof of concept, since we did not want to completely update all of our attributes for all of the memes, which would be an extremely time-consuming process. **Appendix XIII** shows our final task completion.

The block diagram **(Appendix II)** and implementation flowchart **(Appendix (III)** demonstrate that we have completed the project that we initially intended to complete. That being said, we decided to utilize buttons for "like" and "dislike" functionality as a simple proof of concept. Over this semester, we were able to find and manually rate an additional 250 memes according to defined attributes (up to a total of 500 in the meme database), code and release the iOS application for beta testing, and implement the clustering, user preferences, and suggestion algorithms and meme database with the iOS application.

We hoped to rank 1,000 memes, integrate self-rating for new memes recently introduced into the database, and conduct extensive testing with users to find like-swipe ratios, but these were not

accomplished (they were not in the schedule for progress and were just additional features that we hoped to implement if we had time). We chose to rank only 500 memes and not 1,000 because ranking an additional 500 would be an extremely time-consuming process, and if we decided to change attributes over the semester, it would add much more work for incremental benefit. Moreover, we have not yet figured out the best way to implement new memes into the database, so we chose to not focus on implementing the self-ranking for new memes, even though we have devised the algorithm. The main idea is that user attributes will converge after being exposed to and swiping on hundreds of memes, and once these preferences have stabilized, they can be used to rate entering memes so manual entry is not necessary. User testing was not accomplished because we faced difficulties and limitations distributing the application after it was constructed, as mentioned before. To test the effectiveness of this application, we could show 15 memes randomly from the set and then suggest 15 memes based on attributes updating over time; the latter should have an increasingly improving like-swipe ratio over time as the user swipes more, and this ratio should be higher than the ratio achieved for the random memes.

IX. Discussion of Teamwork

Work was coordinated among the members primarily via text message, and tasks were subdivided and distributed based on knowledge and strengths; class assignments, code, memes, etc. are shared on a Google Drive. Our skillset is diverse and complementary: Anish, Saku, and Darius all possess extensive programming and data science knowledge and experience, both from classes at Penn (such as ESE 305) and internships. Anish has skills in data science, system design, and Python, Saku has skills in data analytics, programming, and Android development, and Darius has skills in data analytics, Python, business development, and market analysis. Devan, an M&T student, has strong organizational skills, business analysis knowledge, vision, attention-to-detail, PowerPoint skills, and report-writing capabilities.

With regards to specific accomplishments this semester, everyone contributed roughly equally to gathering and ranking the additional 250 memes, creating slides for the TED-S and Demo Day presentations, and writing the Spring Final Report. Anish was instrumental in developing the M&T Business Analysis and creating the Storytelling Presentation. Saku helped create the 1- and 2-Minute Demo Videos and assisted Darius with some elements of integrating the algorithms with the iOS application. Darius designed the iOS application, including the user interface, statistics, meme feed, etc., implemented the meme database, user preference updating algorithm, and suggestion algorithm with the app, and helped create the 1- and 2-Minute Demo Videos. Devan emailed Dr. Vohra on a weekly basis, created the two Devpost abstracts, helped create the M&T Business Analysis and Storytelling Presentation, designed the outline for the TED-S Presentation, created the Demo Day Poster, and oversaw and outlined the Spring Final Report, TED-S and Demo Day presentations, and Final Kickstarter Video. He made sure all submissions were ready on time, constantly checking in with the team on progress made, work distribution, and next steps.

X. Budget and Justification

The original budget was constructed for the Fall 10.24.18 2-3 Minute Presentation; this initial expected cost to design and implement our project was $0. This budget was decided based on the ideas that all software that we thought would be necessary is free for Penn students (Google Drive, Python, Microsoft Excel, HTML), memes are free to download, survey data can be collected and compiled for free (Google Forms - potentially useful to ranking via attributes), and that websites / web interfaces can be created free of charge (Webnode, WordPress). That being said, we did note that

additional funding may be required if necessary if these free resources could not meet our goals, since we were not completely sure about which resources would be most useful.

We decided to revise the budget for the Fall 11.30.18 4-Minute Presentation. It is important to note that we decided the expected cost to be at least $99. Again, we assumed that all necessary software is free for Penn students (Google Drive, Python, Microsoft Excel, RStudio, Swift / Objective C), memes are free to download, and survey data can be collected and compiled for free. The iOS Apple Developer Program, which is necessary for creating and then placing applications on the App Store for distribution, costs $99 annually.

This semester, we note that all of our software needs were currently met with free programs, specifically Google Drive, Microsoft Excel, iCloud, CloudKit, Python, XCode, and Swift / Objective C. Memes are free to download. That being said, we considered the iOS Apple Developer Program at $99 as our only cost. However, we expect to pay for Amazon Web Services for databases and computing if we decide to grow the business and meme database. Because this is a pay-as-you-use service, the cost would be completely dependent on the number of users and memes; that being said, it would become incrementally cheaper as new memes and users are added. If MyMeMe developed as a business, we would allocate between 45-55% of revenue as costs for research and development and sales, general, and administrative; we expect AWS costs to drop from 30% of revenue to 17.50% by the end of the 5-year projection period (see Business Analysis Excel Model).

XI. Standards and Compliance

With regards to engineering standards, Standards IEEE P7002, IEEE P7003, IEEE STD 23026 from the IEEE Standards Association and the EU General Data Protection Regulation (GDPR) are relevant to our project. It is important to note that we cannot access the specific details of the IEEE Standards without a subscription to the IEEE. That being said, IEEE P7002 relates to the data privacy process and provides users with specific procedures, diagrams, and checklists with regards to systems / software engineering design process for privacy considerations involving user personal data. This is vital to ensure trust with users that their data will be protected; this is relevant both when users are inputting their personal data when creating accounts (which will be implemented eventually) and when swiping, indicating their likes and dislikes. IEEE P7003 discusses algorithmic bias considerations and notes that algorithms cannot have negative bias, meaning that they cannot discriminate against people based on race, gender, sexuality, etc. While we do identify users based on their attributes determined by their swiping choices, no other considerations are considered when suggesting memes. IEEE STD 23026 talks about engineering and management of websites for systems, software, and services information, stating requirements and guidelines for software development and operations. Even though MyMeMe is a mobile application, it may also become a website in a future iteration, so this standard could be applicable. It is interesting to note that this standard asserts that information should be usable for the intended audience, meaning that the memes should be appropriate for their viewers.

With regards to regulatory bodies in the US, there is no single principal data protection legislation, unlike in Europe, which has the GDPR for data protection and privacy. The GDPR defines regulations for collecting and handling user data, and this has similar implications as those proposed by IEEE P7002. That being said, the FTC (Federal Trade Commission) has broad powers to protect consumers against unfair or deceptive practices to protect data, but there are no specific regulatory bodies that are especially pertinent. In summary, we have to be careful with user data, especially when distributing the data to third parties, to preserve privacy and prevent users from being identifiable, although that should not be too problematic.

XII. Work Done Since Last Semester

Over this semester, we were able to find and manually rate an additional 250 memes according to the previously defined attributes (up to a total of 500 in the meme database), code and release the iOS application for beta testing, and implement the clustering, user attribute updating, and suggestion algorithms and meme database within the iOS application. The iOS application has a user interface with a meme feed and a page that tracks statistics unique to each unique user login.

XIII. Discussion and Conclusion

Our team created a mobile meme curation application, MyMeMe. MyMeMe displays a feed of memes and learns a particular user's humor preferences over time by observing what memes he / she "likes" or "dislikes". This is done through binary inputs by the users based on whether or not they particular memes. Over time, the memes a user sees become more and more tailored to his / her humor preferences. Each meme and user have an attribute vector with features we found to create a kernel where clusters of different senses of humor could be identified. These features were created through an iterative process throughout most of the first semester. In summary, the MyMeMe project has five major components, which include the meme database with defined user attributes, the clustering algorithm, the recommendation algorithm, the user preferences updating algorithm, and the iOS application. All that remains to be implemented is the self-rating meme algorithm.

We used Swift and XCode to create the MyMeMe iPhone application. Behind this application is very intricate system design. Traditional machine learning applications such as clustering, which is at the heart of MyMeMe, are quite computationally expensive, so we had to design a system to reduce the complexity. We found iterative clustering, where the meme clusters are recalculated only as new memes enter the system and the user-centroid distances are only calculated after the user evaluates each set of 20 memes, the most scalable when considering memes entering and leaving the system.

Possible additional features that could be included in the future include implementing Facebook and Twitter API (if possible) to gather initial user data to complement the curating process. Linking the applications will allow for easy transfer of useful data to the application and user profiles. Eventually, we also want to add a social aspect to our platform. We hope to allow users to post their own memes and track their respective likes / dislikes, send memes to and message friends, and search for memes. Memes could also be automatically updated to the database using a Python web scraper of top meme pages on Facebook, Instagram, Twitter, etc. Users will have a database in which they are able to access all the memes that they have previously "liked". As mentioned before, it would be ideal for new memes to be self-ranked when they enter the system based on users, whose attributes have clearly converged, liking and disliking them. The existing memes would also ideally be rated by a group more representative of the entire population, rather than just our team.

Over the past two semesters, we learned a lot about system design principles, using mobile application development to bring those to life, and working together as a group. We learned to apply data science methods and algorithms to a specific rather than a general application. One of the major lessons we learned the first semester was to define the set of features before rating all the memes. We had to re-rate hundreds of memes every time we redefined our attribute set, which took a lot of time. Another key lesson was system design for computational complexity. We originally began with a simple continuous clustering model but this did not scale well at all and slowed down the application tremendously in use.

XIV. Business Analysis

Memes are becoming premier sources of information, especially for Generation Z. A meme is a form of comedic expression through digital content, typically seen as a picture, gif, or video, accompanied by text; they are fun, quick, and meant to be shared. According to Forbes (2019), "[m]emes, have become one of the hottest communication forms for Millenials". People find enjoyment both when looking at funny memes themselves and sharing these memes with their friends (and thus hopefully making their friends laugh). These memes are transmitted through channels such as Facebook Messenger, email, social media, text, meme apps, etc. Across these channels, memes congregate themselves in groups such as focused Instagram accounts. But with so many accounts out there, and not much linkage between similar accounts, a person is only viewing a small subset of those memes that are ideally meant for them. Thus, it often takes a person much "meme hunting" before finding one that satiates his/her sense of humor.

These memes are transmitted through social media channels that have essentially become glorified meme hubs. Memes have taken social media platforms by storm over the last few years with users spending more and more time on major platforms such as Facebook, Instagram, Twitter, and Snapchat just looking at memes; much of the value of these is captured in their "stickiness" of users. There is a natural fragmentation in the channels that users utilize to receive this content, as there is no centralized meme source but instead "niche" humor groups scattered across social media. Enter MyMeMe, the "Tinder for Memes".

MyMeMe solves five main problems with meme consumption: classification of memes, the presence of multiple suboptimal platforms, overload of content, short attention span of users, and lack of structured sharing. Since a meme is essentially just a picture with some text on it, it is difficult to classify. Thus, it often takes a person much "meme hunting" before finding one that satiates his/her sense of humor, and these "senses of humor" are extremely difficult to classify in simple terms. This presents a problem of inefficient searching for enjoyable memes and content. Users find information on multiple platforms, including Facebook, Twitter, Instagram, and Snapchat, all of which use feed systems that are overloaded with content posted and shared by other accounts; memes are decentralized. With an increasing amount of this stickiness of major social media platforms residing in memes, we saw a tremendous opportunity in creating a platform to optimize the meme viewing experience. People spend millions of hours per day swiping, scrolling, liking, and commenting on memes every day. However, people generally have short attention spans when browsing content and are prone to get bored, distracted, and / or frustrated while searching for memes. With advanced technology and Internet browsing available at their fingertips, people may just give up searching for memes and engage in another activity if they are unable to find ones that appeal to their unique senses of humor. There is also no structured way of sharing memes, with most being screenshotted and then sent in group chats. In summary, MyMeMe uses the cutting edge of data science to help inch closer to perfect matching between users and memes, providing a centralized meme feed with curated meme delivery, and it hopes to include a structured service for sharing memes in the future.

With its extensive meme database and dynamic user and meme profiles, MyMeMe increases user entertainment from meme consumption while simultaneously decreasing time spent searching for memes, overall providing more concentrated enjoyment. We believe a web service or smartphone application with a "dislike" / "like" swiping mechanism and constant feed of memes, providing better-curated memes to individuals over time as they provide feedback, is the most feasible solution to improve meme consumption. Users have previously shown interest in this swiping activity, as shown by the popularity of applications like Tinder, Bumble, and Hinge, and memes are more prevalent in society than ever before.

Users will thus have a meme feed where they can “like” and “dislike” memes one at a time. These users and memes are defined with attribute vectors, and these are updated based on the users’ preferences, thus making the feed more personalized with increased utilization. Exposing individuals to memes they like on a consistent basis is a value-additive process. In addition, the application will eventually include social aspects, including allowing users to post their own memes, send memes to and message friends, and search for memes. MyMeMe will become the ultimate social media destination for meme consumption and distribution.

It is important to note that there are no applications on the market that offer self-improving meme distribution, as most people have resorted to more-tailored Facebook groups and forums like Reddit. That being said, all platforms that provide outlets for meme expression, such as Reddit, Facebook, and Twitter, are potential indirect competitors with MyMeMe. More direct competitors would include those such as Basement, which is a Y Combinator-backed startup that allows users to share and post content on a feed-based system with only 20 friends. The application also features a meme feed showing users the top trending memes and gives them an option to share this content. Facebook was developing its own meme feed “LOL” to target teens, but it recently ceased work on it. There is an existing service that utilizes Amazon Web Services to analyze memes, but we do not believe that image processing is the best method to provide meme content. There is often a disconnect between the text and image being presented, and we do not believe that senses of humor can be accurately captured with techniques such as principal component analysis and other forms of image processing. The value-add of the consumer-facing side of MyMeMe is primarily gained from individuals who may like many types of memes, and an algorithm based on meme attributes has been designed to hit these many “senses of humor” well and deliver content that is appealing to the specific users. There will be no intellectual property (IP) associated with this project and application.

MyMeMe targets young people (we have defined the primary market as those under age 25, but it is not just limited to these individuals) who use regularly use smartphones and social media. The stakeholders are extensive and include but are not limited to all individuals who view and are affected by memes, the MyMeMe team, the database supplier (Apple, but will be Amazon Web Services eventually), Apple with its App Store and iOS software for application distribution, competing social media platforms such as Reddit, Facebook, and Instagram, etc. Through the use of machine learning based around identifying distinct “senses of humor”, MyMeMe will learn about users over time and provide more and more tailored content, removing the extra, unnecessary channels users currently have to scour to find content that appeals to their taste. This not only saves the user time, but also increases utility gained per unit time the user spends on our app, as he/she is receiving more personalized content and what is not relevant to them is filtered out.

MyMeMe will also feature a B2B arm in addition to its consumer-facing app. This targets the mobile advertising market. We look to monetize the data collected on users and the subsequent network effects, such as users sharing memes in groups. Social media marketing budgets are ballooning for B2C companies, but few are capitalizing on virality. Mobile social media advertising spend is expected to grow by almost 5x from $31 billion in 2017 to $158 billion in 2023; moreover, mobile application revenue is expected to grow almost 3x from $70 billion in 2015 to $189 billion in 2020. Traditionally, advertising is centered around number of engagements or views. If something goes viral, like a Wendy’s tweet, the value realized from it has asymmetric upside. The probability of this happening is relatively low, given the amount of content on the Internet, but its risk-reward profile justifies that savvy businesses at least invest a small percent of their social media marketing budget in this, similar to speculative investments without the downside risk. Historically, virality has been extremely tough to predict, leading to few companies successfully harnessing it, but the data MyMeMe collects on users, humor profiles, and network effects will allow much better prediction. MyMeMe will

capitalize on this data to help businesses create viral content, which it will subsequently promote on the app. This value proposition for businesses can be thought of in terms of Blue Ocean Strategy. With this current plan, MyMeMe becomes the leader in a currently uncontested market, virality advertising, that creates new value for advertisers while decreasing costs to obtain the same level of views and thus can garner asymmetric reach upside for fixed marketing investments.

The business-facing arm of MyMeMe solves the current inability of marketers to capitalize on the virality of social networks. Advertising is currently structured such that companies pay for a certain "expected" reach. MyMeMe will allow for viral marketing to be accessible through insights on the data that is collected. We define viral advertising as advertising based on developing content which companies leave to network effects, or virality, to self-propagate to users across the internet. While the expected distribution of reach has high variance, it is extremely right-tailed. This justifies businesses at least investing a small percentage of their social media advertising portfolio in viral marketing. This can be compared to high-risk-high-return stocks.

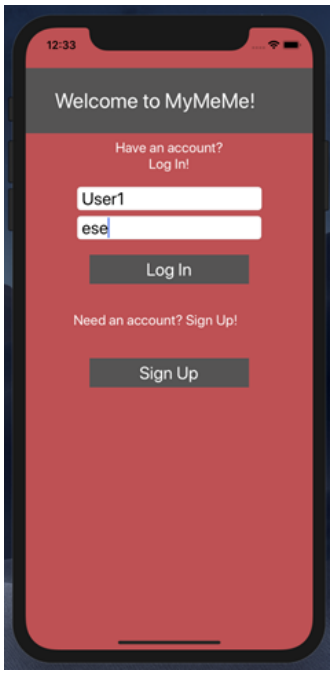
MyMeMe will generate revenue through a combination of in-app advertisements (non-paying users), a premium subscription offering that removes advertisements, and the consulting B2B arms that helps marketers create and display tailored content.

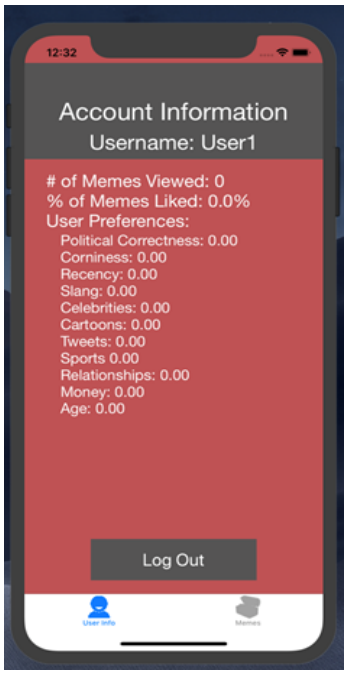
A simple, top-down profit-and-loss model has been created, forecasting revenues and expenses over the next 5 years to calculate potential operating profit. For now, we are just assuming that we can capture a certain percentage of the mobile application and social media advertising markets, but we hope to break this down more granularly and precisely based on our two market segments, especially with regards to the first segment (revenue coming from individuals age 25-and-under who regularly use their smartphones and social media). Preliminary segment sizes and growth rates have been obtained from Statista. The cost structure allows us to maintain high operating leverage. Variable costs primarily come from storing user and meme data in the cloud servers, and this can be accomplished with Amazon Web Services. Fixed costs will exist in the form of sales, general, and administrative expenses (which we will define as all overhead not related to software development, including rent and executive salaries), as well as research and development expenses (which we will define as software developer salaries and related expenses). Generally, as the business grows, it will achieve more economies of scale and organically expand margins, especially as storage costs become smaller per incremental data usage.

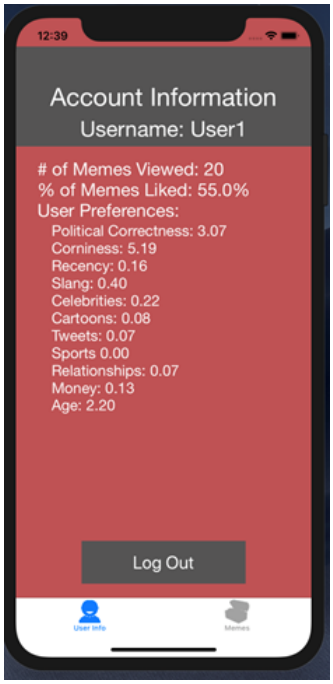
Sources:
Mobile Application Revenue: https://www.statista.com/statistics/269025/worldwide-mobile-app-revenue-forecast/
Social Media Advertising Spend: https://www.statista.com/outlook/220/100/social-media-advertising/worldwide

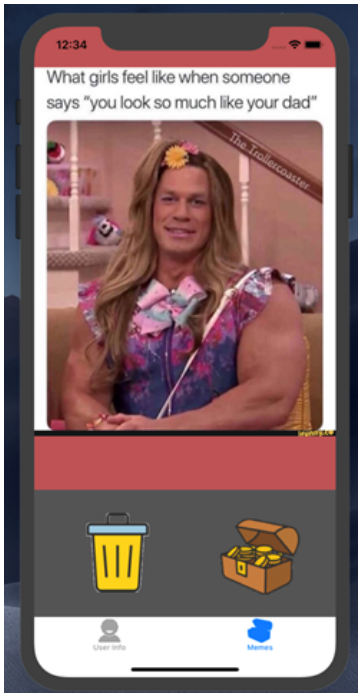
XIV. Appendices

Appendix I: iOS User Interface

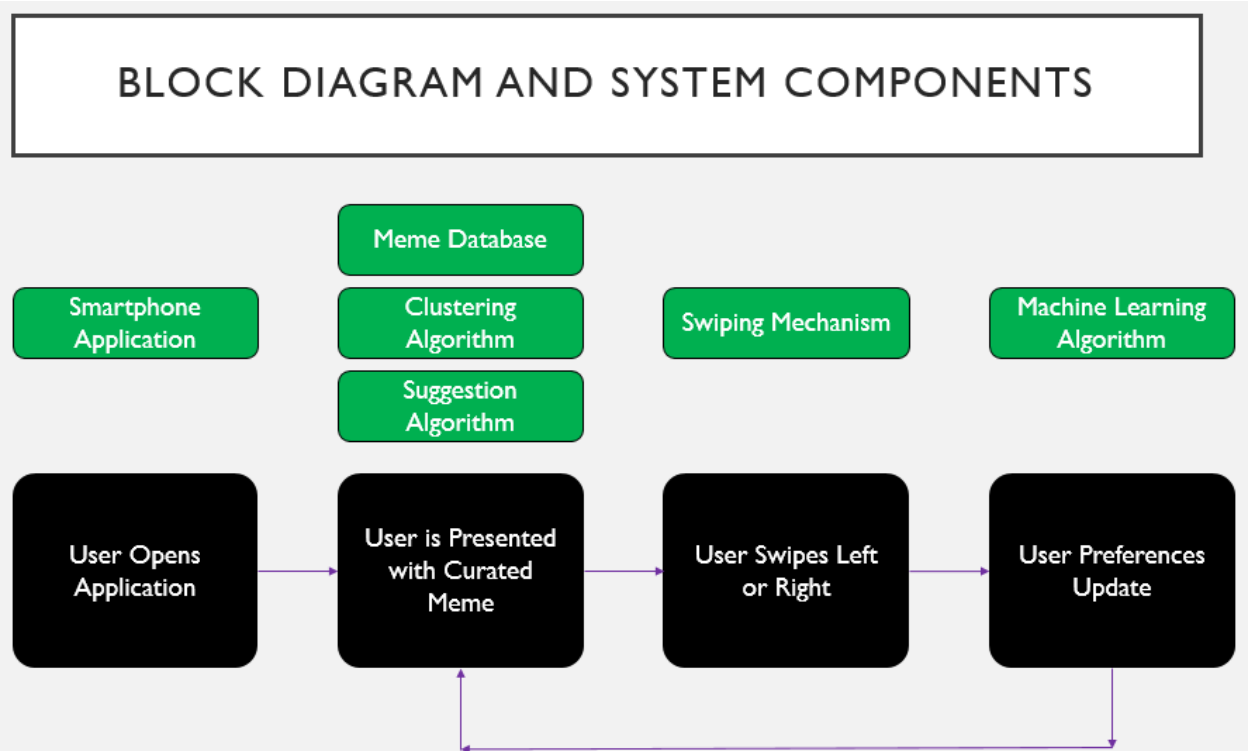




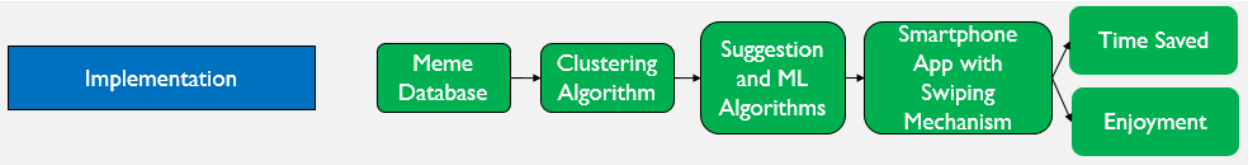




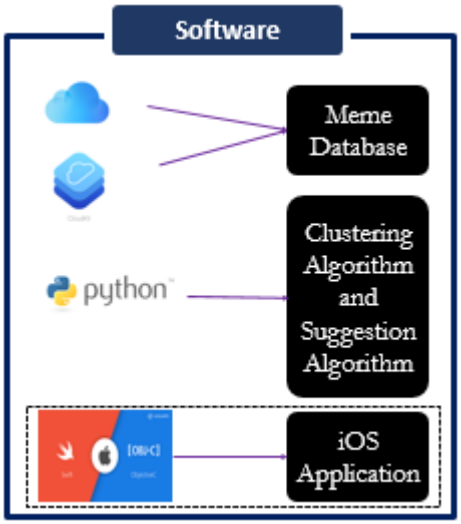
Appendix II: Block Diagram



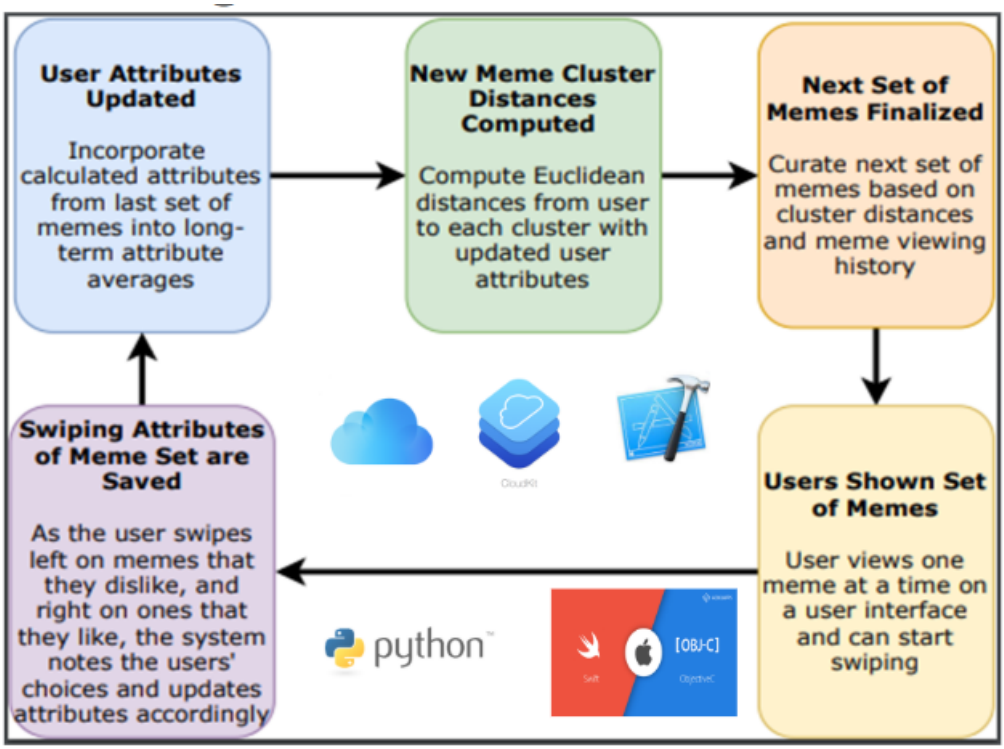
Appendix III: Solution Implementation



Appendix IV: Software Implementation



Appendix V: System Design



Appendix VI: Sample from Meme Database

Meme	Darkness	Corniness	RecencyInteraction	Slang	Celebrity	Cartoon	Tweet	Sports	Relationship	Money	Age	ClusterID
1	1	2	0.0	1	False	True	False	False	False	False	2	3
2	1	4	0.0	1	False	False	False	False	True	True	2	3
3	1	3	9.0	1	True	False	True	True	False	False	2	4
4	2	5	4.0	1	False	False	False	False	False	False	3	1
5	2	3	0.0	1	False	False	False	False	False	False	2	3

Appendix VII: Clustering Code

In [11]:
```
import pandas as pd
```

In [4]:
```
from sklearn.cluster import KMeans

kmeans = KMeans(n_clusters=5).fit(df)
kmeans.cluster_centers_
```

Out[4]:
```
array([[  1.44444444e+00,   4.59259259e+00,   1.48148148e-01,
          1.22222222e+00,   1.85185185e-01,   1.48148148e-01,
          1.11111111e-01,  -1.38777878e-17,   1.48148148e-01,
          3.70370370e-02,   1.74074074e+00],
       [  2.50000000e+00,   3.00000000e+00,   9.00000000e+00,
          2.12500000e+00,   8.75000000e-01,   1.25000000e-01,
          7.50000000e-01,   3.75000000e-01,   0.00000000e+00,
          2.50000000e-01,   1.62500000e+00],
       [  2.20000000e+00,   2.26666667e+00,   1.33333333e-01,
          1.13333333e+00,   3.33333333e-02,   3.00000000e-01,
          1.33333333e-01,   6.66666667e-02,   2.00000000e-01,
          1.00000000e-01,   1.90000000e+00],
       [  3.96969697e+00,   3.30303030e+00,   2.42424242e-01,
          1.87878788e+00,   1.21212121e-01,   1.81818182e-01,
          2.12121212e-01,  -2.08166817e-17,   9.09090909e-02,
          6.06060606e-02,   1.60606061e+00],
       [  2.50000000e+00,   3.50000000e+00,   1.60000000e+01,
          2.00000000e+00,   0.00000000e+00,   5.00000000e-01,
          1.00000000e+00,   0.00000000e+00,   0.00000000e+00,
          5.00000000e-01,   2.00000000e+00]])
```

In [5]:
```
df['ClusterID'] = kmeans.labels_
df.head()
```

Out[5]:

Meme	Darkness	Corniness	RecencyInteraction	Slang	Celebrity	Cartoon	Tweet	Sports	Relationship	Money	Age	ClusterID
1	1	2	0.0	1	False	True	False	False	False	False	2	2
2	1	4	0.0	1	False	False	False	False	True	True	2	0
3	1	3	9.0	1	True	False	True	True	False	False	2	1
4	2	5	4.0	1	False	False	False	False	False	False	3	0
5	2	3	0.0	1	False	False	False	False	False	False	2	2

Appendix VIII: Suggestion Algorithm

ESE Senior Design 2018-2019

Recommender System

Team 3

User Sees:

MEME makes decision ↑ (like) or ↓ (dislike)

User Does Not See:

Meme Attributes: $N * 1 : \vec{M}_i$ for i memes

$w_i \geq 0$: Meme Views (counter) for i memes

User Attributes: $N * 1 : \vec{V}_j$ User Preferences for j users

$s_j \geq 0$: Memes Seen (counter) for j users

$i * 1 : \vec{H}_j$ Meme History for i memes

(binary vector, $\vec{H}_{ji} = 1$ if seen and $\vec{H}_{ji} = 0$ if unseen)

Cluster Attributes: $N * 1 : \vec{C}_k$ for k clusters

$n_k \geq 0$: Memes in Cluster (counter) for k clusters

Updating User Attributes:

IF ↑ : $f = \vec{V}_j$ x s_j

$s_j \mathrel{+}= s_j$

$\vec{V}_j = \dfrac{f + \vec{M}_i}{s_j}$

$\vec{H}_{ji} = 1$

IF ↓ : $f = \vec{V}_j$ x s_j

$s_j \mathrel{+}= s_j$

$\vec{V}_j = \dfrac{f - \vec{M}_i}{s_j}$

$\vec{H}_{ji} = 1$

Attribute values are centered at 0.

Updating Meme Attributes:

Once User Preferences start to converge, as $s_j \to \infty$ for user j, new memes can be defined by user j. For an undefined meme, M_i, users with

$s_j \geq s^*$ (a pre-determined value which is sufficiently large) can view the meme.

IF $\uparrow$: if $w_i = 0$:

$$w_i \mathrel{+}= w_i$$
$$\vec{M}_i = \vec{V}_j$$
$$\vec{H}_{ji} = 1$$

else :

$$f = \vec{M}_i \text{ x } w_i$$
$$w_i \mathrel{+}= w_i$$
$$\vec{M}_i = \frac{f + \vec{V}_j}{s_j}$$
$$\vec{H}_{ji} = 1$$

IF $\downarrow$: if $w_i = 0$:

$$w_i \mathrel{+}= w_i$$
$$\vec{M}_i = -\vec{V}_j$$
$$\vec{H}_{ji} = 1$$

else :

$$f = \vec{M}_i \text{ x } w_i$$
$$w_i \mathrel{+}= w_i$$
$$\vec{M}_i = \frac{f - \vec{V}_j}{s_j}$$
$$\vec{H}_{ji} = 1$$

After the meme is rated a sufficient number of times, $w_i \geq w^*$ (a pre-determined value), it's attributes, $\vec{M}_i$, remain constant. This meme is now defined, and can be clustered with the entire set of defined memes into cluster C_k.

Recommendation Algorithm:

Assuming user V_j and all memes $M_i \in C_k$ are defined, the distance between user attribute vector, $\vec{V}_j$, and each average cluster attribute vector, $\vec{C}_k$, is calculated:

$$d_k = d(\vec{V}_j, \vec{C}_k)$$

for all k clusters.

$$D = \sum_{n=1}^{k} d_k$$

The number of memes, p, to show user V_j from cluster k is:

$$p_k = a * \frac{D}{d_k}$$

where a is a scalar constant. p_k random memes are selected from cluster C_k under the condition $\vec{H}_{ji} = 0$ for those memes, meaning the user, V_j, has not previously seen the p_k memes. These memes are placed into a new viewing set Z. The total number of memes to be viewed before a new Z is calculate is:

$$|Z| = \sum_{n=1}^{k} p_k$$

Appendix IV: User Preferences Updating Algorithm

```
@objc func disliked(_ button: UIButton){
    //memes are viewded only when rated. first meme is added to array before rating.
    //append
    if self.CurrentUser!.viewedMemes.count != self.CurrentUser!.ratings.count{
        self.CurrentUser!.ratings.append(0)

    }
    else {
        self.CurrentUser!.ratings.append(0)
        self.CurrentUser!.viewedMemes.append(self.currMemeNum!)
    }
    let arrA = self.CurrentUser!.attributes
    let arrB = attrDiff().map {$0 * (-1)}
    self.CurrentUser!.attributes = zip(arrA,arrB).map(+)
    for (i,pref) in self.CurrentUser!.attributes.enumerated() {
        self.CurrentUser!.attributes[i] = max(pref,0.00)
    }
    ChangeMeme()
}

func attrDiff() -> [Double] {
    var currMemeAttr : [Int] = []
    if self.currMemeNum! > 250 {
        let ind = self.currMemeNum! - 251
        currMemeAttr = self.memeAttr2[ind]
    }
    else {
        let ind = self.currMemeNum! - 1
        currMemeAttr = self.memeAttr1[ind]
    }
    let divisor: Double = Double(self.CurrentUser!.viewedMemes.count) - 1.0
    var diff: [Double] = []
    for (i, attr) in currMemeAttr.enumerated() {
        let dub=(Double(attr)-self.CurrentUser!.attributes[i])/divisor
        diff.append(Double(String(format: "%.2f", dub))!)
    }
    return diff
}
```

Here we see the preferences updating when a user “dislikes” a meme, and the attrDiff() function calculates the incremental values to subtract (add for like) from the user’s preferences.

Appendix X: Initial Spring Milestones Fall 10.24.18 2-3 Minute Presentation

TENTATIVE SPRING SCHEDULE

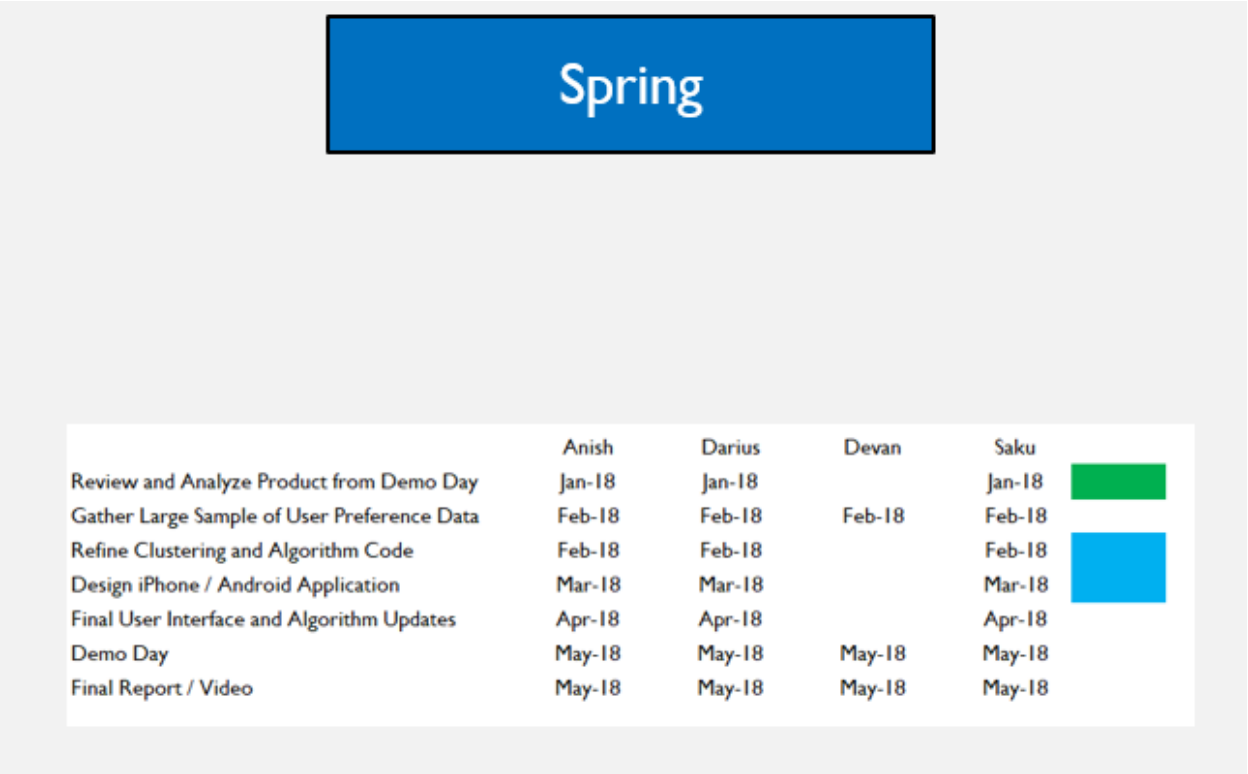
	Anish	Darius	Devan	Saku
Update Dr. Vohra (2x)	Jan-18	Jan-18	Jan-18	Jan-18
Review and Analyze Product from Demo Day	Jan-18	Jan-18		Jan-18
January Executive Summary			Jan-18	
Gather Large Sample of User Preference Data	Feb-18	Feb-18	Feb-18	Feb-18
Refine Clustering and Algorithm Code	Feb-18	Feb-18		Feb-18
Update Dr. Vohra (4x)	Feb-18	Feb-18	Feb-18	Feb-18
February Executive Summary			Feb-18	
Update Dr. Vohra (4x)	Mar-18	Mar-18	Mar-18	Mar-18
Design iPhone / Android Application	Mar-18	Mar-18		Mar-18
March Executive Summary			Mar-18	
Update Dr. Vohra (5x)	Apr-18	Apr-18	Apr-18	Apr-18
April Executive Summary			Apr-18	
Final User Interface and Algorithm Updates	Apr-18	Apr-18		Apr-18
Update Dr. Vohra (1x)	May-18	May-18	May-18	May-18
Demo Day	May-18	May-18	May-18	May-18
Final Report / Video	May-18	May-18	May-18	May-18

Appendix XI: Revised Spring Milestones Fall 11.30.18 4-Minute Presentation

Spring

	Anish	Darius	Devan	Saku
Update Dr. Vohra (2x)	Jan-18	Jan-18	Jan-18	Jan-18
Review and Analyze Product from Demo Day	Jan-18	Jan-18		Jan-18
January Executive Summary			Jan-18	
Gather Large Sample of User Preference Data	Feb-18	Feb-18	Feb-18	Feb-18
Refine Clustering and Algorithm Code	Feb-18	Feb-18		Feb-18
Update Dr. Vohra (4x)	Feb-18	Feb-18	Feb-18	Feb-18
February Executive Summary			Feb-18	
Update Dr. Vohra (4x)	Mar-18	Mar-18	Mar-18	Mar-18
Design iPhone / Android Application	Mar-18	Mar-18		Mar-18
March Executive Summary			Mar-18	
Update Dr. Vohra (5x)	Apr-18	Apr-18	Apr-18	Apr-18
April Executive Summary			Apr-18	
Final User Interface and Algorithm Updates	Apr-18	Apr-18		Apr-18
Update Dr. Vohra (1x)	May-18	May-18	May-18	May-18
Demo Day	May-18	May-18	May-18	May-18
Final Report / Video	May-18	May-18	May-18	May-18

Appendix XII: Revised Spring Milestones MVP Demo 2.8.19



	Anish	Darius	Devan	Saku
Review and Analyze Product from Demo Day	Jan-18	Jan-18		Jan-18
Gather Large Sample of User Preference Data	Feb-18	Feb-18	Feb-18	Feb-18
Refine Clustering and Algorithm Code	Feb-18	Feb-18		Feb-18
Design iPhone / Android Application	Mar-18	Mar-18		Mar-18
Final User Interface and Algorithm Updates	Apr-18	Apr-18		Apr-18
Demo Day	May-18	May-18	May-18	May-18
Final Report / Video	May-18	May-18	May-18	May-18

Appendix XIII: Finalized Spring Milestones

	Anish	Darius	Devan	Saku
Review and Analyze Product from Demo Day	Jan-18	Jan-18		Jan-18
Gather Large Sample of User Preference Data	Feb-18	Feb-18	Feb-18	Feb-18
Refine Clustering and Algorithm Code	Feb-18	Feb-18		Feb-18
Design iPhone / Android Application	Mar-18	Mar-18		Mar-18
Final User Interface and Algorithm Updates	Apr-18	Apr-18		Apr-18
Demo Day	May-18	May-18	May-18	May-18
Final Report / Video	May-18	May-18	May-18	May-18