THESIS PROPOSAL: TASK CLASSIFICATION AND RECOMMENDATION FOR CROWDWORKERS

by

Riley Miller

A thesis submitted to the Faculty and the Board of Trustees of the Colorado School of Mines in partial fulfillment of the requirements for the degree of Master of Science (Computer Science).

Golden, Colorado

Date _____

Signed: _____
Riley Miller

Signed: _____
Dr. Chuan Yue
Thesis Advisor

Golden, Colorado

Date _____

Signed: _____
Dr. Tracy Camp
Department Head
Department of Computer Science

ABSTRACT

TODO: Adding after draft of proposal

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

The following contains my proposal for the research I plan to conduct under the guidance of Dr. Chuan Yue in pursuit of my Masters of Computer Science degree from Colorado School of Mines. My research will be centered around the development of a machine learning approach to improve the efficiency of crowdworkers in crowdsourcing platforms.

In this section I will outline the problem that we are trying to solve, provide a high level overview of crowdsourcing platforms, ecosystems, and tools, as well as highlight traditional approaches to developing recommendation systems.

1.1 Background

Crowdsourcing platforms have become increasingly popular over the course of the last several years, introducing the concept of utilizing "crowd"' intelligence to complete work. [1] These platforms are historically architected to facilitate the relationship between requesters, who create work on the platform in the form of items commonly referred to as tasks, and crowdworkers who are real people that complete tasks that are curated by the requesters. There are a variety of different crowdsourcing platforms that have entered the crowdsourcing market such as Figure Eight, ClickWorker, Spare5, Respondent, Swagbucks, and the most popular platform: Amazon Mechanical Turk (MTURK). These platforms are used for a variety of different use cases ranging from collecting data via surveys, image/video annotation, translations, spreadsheet modifications and analysis, writing, to really anything imagineable. The typical workflow in crowdsourcing platforms starts with requesters who break work that they need to get completed down into tasks. This process usually involves writing a description of what needs to be done, how it needs to be completed, providing the input the crowdworker needs to complete the work, setting a qualification so that only certain crowdworkers may have access to the task based off their work history, setting a lifetime

on the task for when the work needs to be completed, and setting a reward for successful completion of the task. All of these specifications vary by platform since a universal specification standard between different crowdsourcing platforms fails to exist. After the requesters finish a task and publish it to the crowdsourcing platform, the task becomes available to crowdworkers (depending on the platform it may only be made available to crowdworkers who match the qualification set by the requesters) who can then attempt to complete the task. After the crowdworkers are satisfied with their work on a given task they can then submit their work on a task to the requesters for approval. After this work is received by the requesters, the requesters have the ability to review the work that was completed by the crowdworker and determine whether they will accept the work, reject the work, or send the task back to the crowdworker because the quality of the work was insufficient. This workflow in crowdsourcing platforms presents several inherent issues that need to be improved in order for crowdsourcing platforms to continue to grow.

1.2 Requester Role in Crowdsourcing Platforms

Requesters often utilize crowdsourcing platforms to outsource large pieces of work to a distributed workforce to leverage the skills, time, and experience of crowdworkers. This allows requesters to offload tasks to ultimately save themselves time and allow them to focus on higher priority tasks. Some of the practical use cases that requesters offload to crowdsourcing sites are bulk tasks that require human input but take a considerable amount of time like data annotation of machine learning and computer vision data sets which requires a large amount of accurately labeled data to fuel deep learning algorithms. One of the primary benefits that crowdsourcing platforms present requesters is that requesters are able to curate a massive amount of tasks and distribute them amongst crowdworkers for a reasonable cost considering the average wage for crowdworkers lingers around $2-$5 and hour [2, 3]. Many of the tasks that requesters publish on crowdsourcing sites have small rewards ranging on average from a couple cents to several dollars. The low average cost of tasks in crowdsourcing platforms inherently allows researchers to publish a large number of tasks

which also simultaneously benefits crowdworkers and crowdsourcing platforms by increasing the volume of work available. Some of the issues that exist for requesters in crowdsourcing platforms are the amount of time it takes to curate tasks, the amount of time it takes to review and approve tasks completed by crowdworkers, and the diversity in the quality of the work that crowdworkers perform.

1.3 Crowdworker Role in Crowdsourcing Platforms

While requesters supply crowdsourcing platforms with the volume of work, crowdworkers supply the labor. The primary motivation of crowdworkers discovered empirically as apart of a study in 2018 by a team of crowdsourcing researchers was to earn money [2]. The distributed workforce of crowdworkers complete tasks all over the world with a wide range of skillsets and experience while providing human intelligence to complete work published by requesters. The researchers who performed the survey on crowdsourcing workers in Amazon MTURK discovered that over half (61.7%) of crowdworkers were employed fulltime and 50.2% of workers had recieved a four year education.[2] This data shows that many of the crowdworkers don't rely on their crowdsourcing work as a primary source of income and that many crowdworkers are formally educated. However, since many crowdworkers aren't reliant on crowdsourcing platforms as their primary source of income this suggests that quality will be a lower priority for crowdworkers in exchange for efficiency as the repercussions for low quality work have much less severe implications than they would for low quality work with their full-time employers. With the primary motivation of crowdworkers regarding the wage they receive from crowdsourcing platforms, efficiency is imperative and many of the crowdworkers' interactions with crowdsourcing platforms will be guided by the overarching goal of making as much money as possible as efficiently as possible. To help optimize their time during crowdsourcing working sessions, crowdworkers use a variety of different tools to help improve their efficiency while completing tasks.

1.4 Requester vs. Crowdworker Imbalance

The two differing motives of both requesters and workers develops inherent tension between the primary relationship in crowdsourcing platforms. Requesters desire high quality work whereas crowdworkers desire efficient work resulting in two states in the crowdsourcing platform that are relatively mutally exclusive. However, the power in this relationship leans heavily towards requesters who are able to dominate the dynamic using several features that are common amongst crowdsourcing platforms such as: the ability to specify certain qualifications that workers must meet to access certain tasks and having ultimate authority over accepting or rejecting the work that crowdworkers complete based on their own personal, subjective view on quality. [2, 4] This imbalance places workers at a severe disadvantage in crowdsourcing platforms in addition to flaws in the platforms themselves. Some of the disadvantages that crowdsourcing platforms inflict on workers include: naive search functionality for surfacing tasks, lack of metrics for how long it will take to complete a task, stated feasibility of tasks, and a lack of estimated wage value for tasks [2]. Crowdworkers are at an inherent disadvantage in crowdsourcing platforms which harms worker participation, a fundamental requirement for crowdsourcing platforms to be successful.

1.5 Crowdworker Difficulties

More specifically, a survey on crowdworkers showed that the biggest painpoints for crowdworkers in the Amazon MTURK crowdsourcing platform are loss of compensation on rejected or returned tasks, data on whether or not a given task is even completeable (often times a task may not even be completeable because requesters may not have provided enough information for workers to successfully complete the task), and the amount of time it takes to find a task or switch context between different types of tasks [2]. The two pain points that I plan to explore as apart of my research are decreasing the number of returned or rejected tasks by providing data upfront to workers regarding the feasibility of a task as well as decreasing the amount of time spent searching for tasks. Another interesting data

point the researchers (Kaplan et al) collected from their survey on crowdworkers was that 30% of respondents said that finding a task was reported “4 - Very“ or “5 - Extremely“ difficult, but probably even more intriguing was the data that the most pertinent reason for ending a session was that they were unable to find a task worth doing (48% said this ranked as an “5 - Extremely Important“ reason in terminating a crowdsourcing session). [2] This dissatisfaction of crowdworkers shows that the current working model for crowdsourcing platforms needs to be improved to improve the user experience of crowdworkers. Improving the user experience of crowdworkers is imperative to the continued adoption of crowdsourcing platforms by new crowdworkers and has a lot of opportunity for research to improve the experience of crowdworkers in these platforms including the use of machine learning to aid crowdworkers in making intelligent decisions regarding task selection and to intelligently surface relevant tasks that crowdworkers believe are worth doing.

1.6 Goal

I plan to create an algorithm that determines the skills required to complete a crowdsource task to improve crowdworker efficiency through intelligent batching and content based recommendations.

1.7 Approach

I will be focusing my research on improving the efficiency of crowdworkers in crowdsourcing platforms by using text classification of crowdsource tasks to determine from the raw contents and metadata of the task, what the necessary skills to complete the task would be. This categorization will lend itself to be leveraged in an approach I have defined as Intelligent Batching. The concept of intelligent batching regards the extraction of necessary skills from crowdsource tasks to cluster other similar tasks that require similar skills to complete the respective task in an effort to reduce the time it takes to find relevant tasks by clustering similar tasks together and to also reduce the number of rejected task by sufacing the necessary skills to complete a given task to crowdworkers so they’re aware of whether or not

they actually posses the skills necessary to successfully complete a task. I believe performing research on improving crowdworker efficiency is imperative to the health of crowdsourcing platforms as a whole because by improving the efficiency of crowdworkers the crowd will be able to generate higher throughput of tasks which helps requesters by reducing the amount of time it takes for the tasks that they publish to get completed as well as helping crowdworkers make more money through their newfound efficiency. Specifically regarding my research, by using intelligent batching, workers will reduce the amount of time it takes them to find and complete tasks and will also improve the quality of work that is done through the surfacing of the required skills it takes to complete a task. With explicit information on the required skills that are needed to complete a task, workers can make intelligent and efficient decisions on whether they engage with a task or not depending on if they possess the necessary skillset.

CHAPTER 2

RELATED WORK

2.1 Existing Tools

Crowdworkers have created a variety of user plugins and browser extensions to help out the community of crowdworkers to try and improve crowdworker efficiency. Some of the more prominent plugins focus on batching similar tasks to reduce the time users spend switching context between dissimilar tasks and plugins for workers to rate requesters based off their interactions directly and indirectly with how the tasks are structured. I've curated a list of tools used by crowdworkers from my own research and from the results of a survey of crowdworkers [2].

- **HIT Scraper:** A web scraper that helps provide additional search filters not offered as apart of the native offering for Amazon MTurk.
- **MTurk Suite:** A browser extension used to combine a plethora of other crowdworker tooling.
- **Turkopticon:** A web tool that allows crowdworkers to rate requesters and tasks.
- **Greasemonkey/Tampermonkey:** A browser extension that allows crowdworkers to run custom scripts to help boost efficiency in crowdsourcing platforms.
- **Panda Crazy:** A tool used by crowdworkers to batch similar tasks together.
- **Turkmaster:** A script that monitors search pages, requesters, and can automatically accepts tasks on Amazon MTurk.
- **Block Requesters:** Allows users to block and ignore requesters from search results, useful if crowdworkers have a bad experience with a requester and wish to avoid future interactions.

- **Pending Earning:** Allows crowdworkers to view pending earnings for tasks that have been completed and submitted but not approved.
- **MTurk HIT DataBase:** Improved interface for searching tasks that you have worked on previously, Amazon MTurk.
- **MTurk Worst Case Scenario Calculator:** Tool to calculate approval rate and how many rejections it would take to drop your approval percentage to a certain threshold.
- **MTurk Dashboard HIT Status links:** A tool which provides quick access to rejected and pending tasks, Amazon MTurk.
- **MTurk Engine:** A browser extension that combines additional search filters with batching, as well as automated task watching for Amazon MTurk. This tool also includes a dashboard to track earnings.

These tools show the desire for improvement in the crowdworker user experience from the native crowdsourcing platform and a high level of community involvement and support for crowdworkers. Although there is existing tooling for batching similar tasks using keywords and search filters there still lacks tooling for common painpoints highlighted in the survey results collected by (Kaplan et al). Some of the main areas of the crowdworker user experience that still need to be addressed are: surfacing useful recommendations of tasks that are curated based off worker history, expertise, and preferences, content-based analysis of the feasibility of a task, and intelligent automation based on manual user search for tasks.

2.2 Existing Recommendation Systems

There are several different approaches to developing recommendation systems, the two most common are collaborative filtering and content based approaches. Collaborative filtering is generally more accurate than content based approaches however, collaborative filtering struggles with recommending new items, a characteristic of tasks in crowdsourcing platforms.

Content based approaches are based on determining the similarity between items and how they associate with users in the platform, crowdworkers in our use case.

Often times in platforms that are trying to develop a recommendation system where new data is constantly entering the platform and old data is constantly becoming outdated, the platform will apply a content based approach instead of collaborative filtering to address the cold start problem and the sparseness of interaction interaction of similar users on similar tasks. One example of this use case to a similar problem is the use of a content based approach to surface relevant news articles to content reviewers for media corporations like Buzzfeed. The researchers (Wang et al.) leverage a character level neual network langauge model (a CNN) to perform low-level textual feature learning [5]. This is a very applicable approach to my research since tasks in crowdsourcing platforms are constantly getting outdated and after a worker completes a task, the specific task is resolved and will not be reused, making any type of collaborative filtering approach forseeably ineffective.

Researchers (Yuen et al.) developed a matrix factorization method of recommending tasks in Amazon MTurk. Their approach was predicated on using matrix factorization on crowdworker performance history as well as their task searching history to surface relevant reccomendations to crowdworkers. Another set of researchers applied two different techniques based on implicit modeling of user history leveraging a Bag-of-words Approach and a classification approach [6]. One of the downfalls of their research was that they only used 24 users to evaluate their approach.

From my initial investigation, I haven't found any research that is directly related to my approach of extracting necessary skills from tasks using text classification. The primary targeted application of my research is to eventually create a content based recommendation system that will match a user to a task in a crowdsourcing platform based on implicit skills the user has obtained (based on previously completed tasks and user provided characteristics) and the required skills that are needed to complete a task.

2.3 Text Classification of Crowdsourcing Tasks

Interesting related work in the field of text classification of tasks in crowdsourcing platforms can be found in an analysis of the dynamics of crowdsourcing performed by (Difallah et al) in which they used supervised learning to classify types of tasks in Amazon MTurk [7]. This research is extremely useful for my approach and will allow me to use the common categories that they defined as apart of their research to gather data from crowdworkers as apart of my research on the **skills** that are required to complete crowdsource tasks.

CHAPTER 3

PROPOSED WORK

3.1 Potential Approaches

Recommendation systems have been around since the early days of the internet and several of the techniques used early on are still heavily relied on today and in some cases their general principles are utilized to develop intelligent approaches in conjunction with traditional methods. The two primary methods that the majority of recommendation systems are based off of are collaborative filtering and content based approaches. [8]

3.1.1 Collaborative Filtering

Collaborative filtering is the most popular recommendation system in use historically and today. This approach pertains to considering user data when processing information for recommendations, or in simpler terms, making predictions based off of other users. This approach has been in use since the earlier days of the internet, getting traction around the advent of e-commerce websites. An example of how this approach is applied is that for an e-commerce site, after items on the site are purchased by customers, many times the customer who purchased the item will be prompted to rate the product or to leave a review. You probably recognize this pattern from any time you buy something off Amazon and you get an email a couple of days after your package arrives asking you how you liked the product or if you'd be willing to leave a review. Anyways, once customers start interacting with items in the e-commerce store, collaborative filtering will use this data to recommend items to users who are similar to the people who originally bought the item. One of the primary stumbling blocks with this approach is the cold-start problem where the recommendation system isn't able to surface relevant recommendations at the inception of the system since there are a limited number of user interactions with items and thus the quality and relevancy of the recommendations will be poor. [5] Collaborative filtering is also only effective on data that

has already existed on the service and is unable to make recommendations for new items until the new items are interacted with. One thing that is interesting to note is that this idea of collaborative filtering can be used in conjunction with deep learning techniques to create intelligent hybrid recommendation systems based off of basic collaborative filtering principles. One example of this is in a study in which researchers created a model using a technique that they called collaborative deep learning (CDL) which leveraged a hierarchical Bayesian model which combined collaborative filtering and the content based approach using deep learning to create better recommendations . [9]

3.1.2 Content Based

The content based approach is less commonly used than collaborative filtering but it is still a prevalent approach for building recommendation systems. Content based recommender systems are based on similarity of items in the recommender system. These systems leverage the matching of similar items to a user profile. [10]. Content based approaches inherently don't have to deal with the cold-start problem that is common in collaborative filtering recommendation systems since they don't bother with the behavior of similar users but of similar items within the system. These recommendation systems are also commonly used in systems where content has frequent turnover, or where data is sparse [11].

3.1.3 Text Classification

Text classification is the process of determining similarity from free text. This is a common area of natural language processing research and pertains to a wide array of different applications. Text classification can leverage either supervised or unsupervised techniques, providing a lot of flexibility for differnet approaches. Supervised text classification techniques use labeled training data to develop models that can then be used to categorize unlabeled free text. The most important piece of the supervised approach to text classification is possesing a large, annotated corpus to train accurate models. Alternative to the supervised approach, unsupervised learning can also be applied to text classification problems. Unsupervised

learning relies on discovering similarities within a dataset and grouping together similar data, this approach is typically referred to as clustering. One of the beautiful things about unsupervised learning is that it doesn't rely on annotated training data, it will just discover similarities from the data provided. These two approaches can be applied to improving efficiency of crowdworkers.

3.2 Research Focus

The area of research that I plan to explore is the extraction of skills that are needed to perform tasks in crowdsourcing platforms using text classification natural language processing techniques. This idea is formed on the basis of the data that we will be able to procure from crowdsourcing platforms.

As apart of a subsequent faction of the grant that this research is founded on, Dr. Yue has formed a team to garner data collection from crowdsourcing platforms using a client side browser extension. Some of the predictions of the dataset we believe that we are going to be able to collect using the browser extension are: that we will be able to generate a sizeable corpus of raw text and metadata of HITs on Amazon MTurk but that we aren't going to be able to procure enough data on user sessions, user profiles, and user interactions under the time constraints of the grant to generate a collaborative filtering based recommendation system due to the forseeable initial adoption of the plugin.

However, once we are able to generate the corpus of tasks, we will be able to apply text classification to extract skills that are needed to perform a task and once we are able generate enough user data from the browser plugin, we will be able to develop a content-based recommendation system for surfacing relevant tasks in crowdsourcing platforms.

This approach will also give us the ability to improve crowdworker efficiency through the intelligent batching of tasks. This will be the more immediate application of my research. Using text classification to determine the skills needed to perform a task, we will be able to use our algorithm to cluster similar tasks and eventually queue and accept tasks with the same skill requirements which will allow crowdworkers to be able to complete tasks quicker

and more efficiently. Ultimately, I predict that intelligent batching will reduce the amount of time that crowdworkers waste sifting through crowdsourcing platforms to find tasks that would be worth doing since they would a) possess the skills necessary to complete the task and b) the tasks would be similar to the other enqueued tasks which would save users time from context-switching.

3.3 Potential Algorithms

Listed below are some of the algorithms that I plan to investigate to perform the skill extraction of tasks in crowdsourcing platforms. These algorithms will be explored due to their common usage to solve text classification problems as outline in a survey of text classification techniques [12].

- **Decision Trees**
- **Rule-based Classifiers**
- **SVM Classifiers**
- **Bayesian Classifiers**
- **Neural Network Classifiers:** Specifically LSTM RNNs and CNNs
- **Nearest Neighbor Classifiers**

I will explore the application of the algorithms above to the problem of extracting required skills from tasks and compare my results accordingly.

3.4 Dataset

As explained in the "Research Focus" section above, Dr. Yue has created a team of researchers who will be handling the development of a client side browser extension to web scrape crowdsourcing platforms for task data as well as to track user profile information and user interactions with the crowdsourcing sites. We will be able to quickly web scrape

crowdsourcing sites to gather enough data for the text classification I plan to perform on crowdsource tasks. But we are envisioning that it will take considerably more time to gather enough user data to develop a machine learning recommendation system which utilizes content based recommendations.

3.5 Evaluation

In this section I will outline some of the experiments I plan to conduct to develop an effective algorithm which will be able to accurately extract skills that are required to complete tasks.

3.5.1 Metrics

To determine the success of different approaches I will allocate the dataset that we collect from scraping the crowdsourcing site into an 80/20 split where 80% of my data set will be used for training the machine learning algorithms while reserving the remaining 20% of the data set to test the accuracy of the models and to evaluate which approach is most effective for determining which skills are necessary to perform tasks. Due to the timeline of the grant and time constraint of my graduation date, I will be leveraging offline testing to perform my analysis. Ideally we would be able integrate the models into the browser extension and perform A/B testing in live environments to test the performance of the different techniques but that isn't a viable option given the timeline of the project.

3.5.2 Experiments

The two different types of experiments that I plan to conduct can be broken down into the two high-level categories of supervised and unsupervised learning.

3.5.2.1 Supervised Learning

The majority of my time will be spent performing experiments on supervised learning techniques, comparing the performance of shallow learning vs. deep learning algorithms on my data set. For these sets of supervised learning algorithms a large part of my research

will be predicated on annotating the dataset that is collected by Dr. Yue's team. My plan for collecting this annotation data can be broked down into two phases:

Survey: I plan on administering a survey to crowdworkers on Amazon MTurk to gather data about common skills that are required to complete HITs. At a high level I plan on collecting worker demographic data to analyze the difference in reported skills across different types of workers with different backgrounds and differing qualifications and number of completed tasks. Another facet of the survey will include giving crowdworkers a small subset of common tasks from some of the most commonly occuring categories of HITs on MTURK [7]:

- **Information Finding:** Searching the web to find certain information
- **Verification and Validation:** Verifying certain information
- **Interpretation and Analysis:** Interpreting web content (i.e human generated sentiment analysis)
- **Content Creation:** Generating content from audio or video input
- **Surveys & Questionaires:** A set of questions which gather worker demographic info as well as answers on a subject matter
- **Content Access:** Interacting with web comment (i.e watching videos, visiting website)

and asking them to list common skills that are required to solve the sample task. There will also be a free response portion of the survey where I will prompt crowdworkers to list an n number of skills for each general category of tasks.

Annotation: Then after curating a list of skills from the survey, I plan on having crowdworkers annotate my dataset of tasks with n skills that are required or helpful to complete a given task using skills from the pool of skills generated from the survey.

After I have collected and annotated the dataset of tasks, I will then train the supervised learning techniques listed in the "Potential Algorithms" section above using either Jupyter Notebooks on a platform like Google Colab or I will leverage AWS GPU-enabled P-instances to train my models on my training set. After the models are trained, I plan on testing the supervised learning models on my test set and will generate a wide-variety of insightful tables and visualizations that will provide additional insight into the effectiveness of each of the different techniques. I also plan on leveraging GitHub for version control as well as creating a data pipeline using continuous integration and machine learning frameworks like PyTorch or TensorFlow to iterate quickly on the different supervised learning techniques.

3.5.2.2 Unsupervised Learning

In addition to the supervised learning techniques outlined above I also plan on experimenting with several unsupervised learning techniques for comparison's sake, specifically nearest neighbor and clustering approaches. I plan on using the same infrastructure, hardware, and frameworks listed above for my unsupervised learning experiments while forgoing the training and data collection steps listed in the "Supervised Learning" since these steps are not required for unsupervised learning algorithms.

CHAPTER 4

PROJECT PLAN

The project plan listed below contains weekly/biweekly objectives to keep me on track and hold myself accountable in order to successfully defend my thesis in April 2020.

4.1 Outline of Project Timeline

- **November 25:** Thesis Proposal
- **December 9:** Address Thesis Proposal feedback, receive data set from Dr. Yue, Create survey for surveying workers on common skills for common tasks
- **December 16:** Tasks created and published for skills survey on Amazon MTurk
- **December 23:** Approving skills surveys info, storing skills survey info in database, begin analysis on skills database
- **January 6:** Tasks created and published for annotation on crowdsourcing site (Amazon MTurk or Figure 8) Apply to Graduate
- **January 13:** Approve, store, and analayze completed annotation data
- **January 20:** Approve, store, and analayze completed annotation data. Begin applying shallow learning techniques to annotated dataset.
- **January 27:** Developed and tested shallow learning techniques (Decision Trees, Rule-based, SVM, Bayesian). Begin applying deep learning techniques to annotated dataset.
- **February 10:** Developed and test deep learning techniques (LSTM RNN and CNN approaches), begin writing paper for conference submission

- **February 24:** Algorithm iteration and improvement, continue writing paper for conference submission, start drafting thesis
- **March 9:** Develop visualization and analysis of test results, further iteration and improvement of algorithms, continue writing paper for conference submission, begin writing thesis
- **March 23:** Continue algorithm analysis, continue writing paper for conference submission, continue writing thesis
- **April 6:** Defend Thesis
- **April 13:** Complete graduation checkout course, submit signed thesis defense form, upload content approved thesis to ProQuest
- **April 17:** Thesis formatting approval by 1:00pm
- **April 17 - May 7:** Continue drafting paper and submit to conferences
- **May 7:** Graduate

4.2 Support

In order to deliver high quality research I request the following meeting cadence with the thesis committee to gather feedback and correction on my approach as needed:

- **Dr. Chuan Yue (Thesis Advisor):** Weekly status meeting and review. This meeting should be leveraged to review weekly progress and to give feedback and code review to ensure satisfactory progress. **Suggested Duration: 1 hr**
- **Dr. Thomas Williams (Committee Member):** Bi-weekly status meeting on research progress. This meeting will be leveraged to gather feedback on specific research techniques, to update committee member on progress, and for guidance and additional considerations. **Suggested Duration: 15-30 min**

- **Dr. Hua Wang (Committee Memer):** Bi-weekly status meeting on research progress. This meeting will be leveraged to gather feedback on specific research techniques, to update committee member on progress, and for guidance and additional considerations. **Suggested Duration: 15-30 min**

REFERENCES CITED

[1] Siou Chew Kuek, Cecilia Paradi-Guilford, Toks Fayomi, Saori Imaizumi, Panos Ipeirotis, Patricia Pina, and Manpreet Singh. The global opportunity in online outsourcing. 2015.

[2] Toni Kaplan, Susumu Saito, Kotaro Hara, and Jeffrey P Bigham. Striving to earn more: a survey of work strategies and tool use among crowd workers. 2018.

[3] Kotaro Hara, Abigail Adams, Kristy Milland, Saiph Savage, Chris Callison-Burch, and Jeffrey P Bigham. A data-driven analysis of workers' earnings on amazon mechanical turk. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, page 449. ACM, 2018.

[4] Mohammad Allahbakhsh, Boualem Benatallah, Aleksandar Ignjatovic, Hamid Reza Motahari-Nezhad, Elisa Bertino, and Schahram Dustdar. Quality control in crowdsourcing systems: Issues and directions. *IEEE Internet Computing*, 17(2):76–81, 2013.

[5] Xuejian Wang, Lantao Yu, Kan Ren, Guanyu Tao, Weinan Zhang, Yong Yu, and Jun Wang. Dynamic attention deep model for article recommendation by learning human editors' demonstration. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 2051–2059. ACM, 2017.

[6] Vamsi Ambati, Stephan Vogel, and Jaime Carbonell. Towards task recommendation in micro-task markets. In *Workshops at the Twenty-Fifth AAAI Conference on Artificial Intelligence*, 2011.

[7] Djellel Eddine Difallah, Michele Catasta, Gianluca Demartini, Panagiotis G Ipeirotis, and Philippe Cudré-Mauroux. The dynamics of micro-task crowdsourcing: The case of amazon mturk. In *Proceedings of the 24th international conference on world wide web*, pages 238–247. International World Wide Web Conferences Steering Committee, 2015.

[8] Ivens Portugal, Paulo Alencar, and Donald Cowan. The use of machine learning algorithms in recommender systems: A systematic review. *Expert Systems with Applications*, 97:205–227, 2018.

[9] H. Wang N. Wang and D.-Y Yeung. Collaborative deep learning for recommender systems. *Proc. KDD*, pages 1235–1244, 2015.

[10] Michael J Pazzani and Daniel Billsus. Content-based recommendation systems. In *The adaptive web*, pages 325–341. Springer, 2007.

[11] Shumpei Okura, Yukihiro Tagami, Shingo Ono, and Akira Tajima. Embedding-based news recommendation for millions of users. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1933–1942. ACM, 2017.

[12] Charu C Aggarwal and ChengXiang Zhai. A survey of text classification algorithms. In *Mining text data*, pages 163–222. Springer, 2012.