**COMPUTER SYSTEMS RESEARCH - YILMAZ**

**Legend (or feel free to use your own legend or color codes):**

❌ - Incomplete

❔ - Complete, requires further exploration

✔️ - Complete, successful

⭕ - Complete, unsuccessful

⚪ - Not started

Please add your milestones below. Make it clear who will do which task if you are grouping. Remember each group member should be busy at all times.

Make it clear what kind of dependencies your tasks may have (Task A is dependent on Task B if you cannot start A before finishing B.).

Final Report will be due in May 2025. Final presentation will be done 2 weeks before TJStar. Exact dates will be announced later.

I should be able to measure your performance after each task completion. Therefore, you should think of measurable goals while dividing your project into tasks. Keep the format as a succinct list, don’t write a paragraph for a task.

**MILESTONES**:

Project Presentation 1 October 7-21, 2024

(must have decided the project by now)

✔️ *Task 1: Look more into the GitHub repository of research article (on retrieval methods) and test their code with AI generated text (spend a week on this). See the performance and how it works. Familiarize yourself with the code that they wrote.*

✔️ *Task 2: Research 3 articles outlining retrieval encoding methods/vectorizing words approaches and how they work, then create a bulleted list summarizing each of the articles. Can take parts of their approach in solving your problem.*

✔️ Task 3: Do more research into contrastive learning and how I can use that for my database

✔️ *Task 4: Experiment with “mixing attacks”, look at different AI detectors and how they fair with mixed AI and human writing (a week or so on this). Test AI detectors against “mixing attacks” and record the results.*

❌ *Task 5: DEPENDENT ON TASKS 1, 2, 4. Write code to store AI generated text in a database, look into how to create these databases by looking at GitHub Repository for research article. Also can hook up the database to GPT 4.0 to generate AI text, find a way to make the database.*

*Dividing this into smaller tasks:*

✔️ *Mini Task 1: Use the flask and HTML skills learned in web to create a web server that takes in text as an input and stores it into a database in SQL*

✔️ *Mini Task 2: Process in the text and furnish up the website so that it looks presentable*

❌ *Mini Task 3: Make the Words2Vec model with the text given to encode the text*

*I have started this but haven’t finished yet*

❌ *Mini Task 4: Try to hook up the model to GPT 4.0 to make the processes of generating text easier for the user. Can have the user enter in a certain prompt, then that goes to GPT 4.0 to generate results instead of manually typing them in.*

Project Presentation/Demo 2 December 2-13, 2024

(20% of the project must be done)

⭕ Task 1: Work on researching the Word2Vec model in order to understand it better. Write summaries of the article for future reference. Look into at least 3 articles for reference. (I did this, but after further evaluation, I realized that this isn’t the best approach for encoding whole documents)

⭕ Task 2 DEPENDENT ON TASK 1: Using the research done, try and implement the Word2Vec model to get a vectorized version of each of the pieces of text. This will then be used to compare semantic similarity near the end. (I tried this, but it was unsuccessful, and I pivoted towards Doc2Vec instead, which will help achieve my goals better)

✔️ Task 3: Research into the Doc2Vec model, which is better than Word2Vec because it gives a vector representation of the document itself. Try to look at 2-3 articles summarizing the approach in Doc2Vec and how it works.

✔️ Task 4 DEPENDENT ON TASK 3: Implement Doc2Vec by creating vectors for each text that is inputted into the website. There needs to be vectors for the database as well as a vector for the input generation.

✔️ Mini Task 1: Make sure the database has vectors that represent it inside the SQL file

✔️ Mini Task 2: Make sure the input has vectors that represent it in the SQL file

✔️ Task 5 DEPENDENT ON PREVIOUS TASKS: Test thoroughly to make sure that the website works correctly after Doc2Vec is implemented, meaning the website doesn’t crash or anything else, it should still run fine after Doc2Vec is put inside of it.

✔️ Task 6: Start to initially change the parameters in Doc2Vec to make sure that it gives optimal results and does well in creating representative vectors where similar documents have similar vectors.

Project Presentation/Demo 3 February 3-14, 2025

(45% of the project must be done)

Task 1: Change the parameters around a little bit more to get better results. (**GOING TO DO THIS TASK LATER)**

✔️ Task 2: Add a prompt into the website for the website to pull responses on. Ex: “Write an essay on the influence of Thomas Jefferson on American Society”

✔️ Mini Task 1: Then display that prompt on the website itself, so that the user knows what prompt they have put in.

✔️ Mini Task 2: Add in a question for the user about how many words the essay has (minimum/maximum)

❌ Task 3: Send this prompt to Chat GPT 4.0 and get responses on the prompt. Meaning take the prompt and generate GPT 4.0 responses on it, up to 200-300 responses and store that in the code. (**DID THIS TASK LATER, TASK 8)**

<https://blog.hubspot.com/website/chatgpt-integration>

✔️ Task 4: Test Doc2Vec encoding strategy with AI generations that were gotten manually from the website itself and not from an API

✔️ Mini Task 1: Figure out a question that a teacher would input: How should technology be used to address the challenges of climate change?

✔️ Mini Task 2: generate responses from the newest AI models, this includes Microsoft Copilot, GPT, DeepSeek, Grok. (Generate 2 responses for each)

✔️ Mini Task 3: Copy and paste the AI outputs into the code so that they can be stored into the database and encode them.

✔️ Mini Task 4: Generate an input text to check against, ie: the text the teacher would put in for the student.

✔️ Mini Task 5: Encode the input text

✔️ Mini Task 6: Compare the database text to the input text using cosine similarity of vectors, and find the highest cosine similarity among all the vectors in the database.

✔️ Mini Task 7: Compare the highest cosine similarity value to the threshold, if its higher than the threshold then its AI generated, if not its human written.

✔️ Task 5: Test Doc2Vec one more time with 2 paragraphs instead of 1 and see how it does.

✔️ Mini Task 1: Do the same process as above for a different prompt and 2 paragraph responses

✔️ Mini Task 2: Test it and output the results

✔️ Task 6: Test one more time Doc2Vec and the model overall on 3 paragraphs of text with 10 pieces of text in the database

✔️ Mini Task 1: Do the same process as above for a different prompt and 3 paragraph responses

✔️ Mini Task 2: Test it and output the results

❌ Task 7: Let the user enter in AI generations for the database to use instead of having to manually put it in the background code every time. (**DID NOT NEED TO DO THIS BECAUSE I AM USING GPT’s API)**

✔️ Task 8: Send the prompt the user entered to Chat GPT 4o mini and get responses on the prompt. Meaning take the prompt and generate GPT 4o responses on it. Then take those responses and use their encoded versions as part of the database.

✔️ Mini Task 1: Create an API Key

✔️ Mini Task 2: Look on openAI’s website and compare the different models, find a model that suites this purpose the best

✔️ Mini Task 3: Add some money to the openAI account in order to use their API

✔️ Mini Task 4: Look at their instructions for how to use that model

✔️ Mini Task 5: Implement the instructions in my code, but change it up slightly to work within the context of what I want (so like have the API take in the question the teacher entered with a maximum and minimum word count that corresponds to what the teacher entered in and generate responses on that).

✔️ Mini Task 6: Test it and make sure it works, debug any errors as necessary.

Project Presentation/Demo 4 April 1st -, 2025

(70% of the project must be done)

✔️ Task 1: Take 10 GPT 4 generated texts and test them for AI generation

✔️ Mini Task 1: Generate GPT 4 text on 5 different prompts

✔️ Mini Task 2: Input the text from GPT 4 as well as the input question for each generation into the website

✔️ Mini Task 3: Note down what the similarity score is and whether its AI generated or not.

✔️ Task 2: Take 10 human written texts and test them for AI generation

✔️ Mini Task 1: Find my own human written text to use in order to test my model.

✔️ Mini Task 2: Input that text into my website as well as the input question

✔️ Mini Task 3: Note down what the similarity score is and whether its AI generated or not.

✔️ Task 3: Take 9 total (3 each) DeepSeek, Copilot, Grok output and test them for AI generation

✔️ Mini Task 1: Generate DeepSeek, Copilot, Grok output (3 different outputs for each AI model on one question)

✔️ Mini Task 2: Input the text from the AI’s as well as the input question for each generation into the website

✔️ Mini Task 3: Note down what the similarity score is and whether its AI generated or not.

✔️ Task 4: Take GPT 4, DeepSeek, Copilot, Grok output and paraphrase using 2-3 different paraphrasers and test that for AI generation.

(Take one output from all of these AI content generators, paraphrase it 2-3 times and for each of those paraphrases check it for AI generation).

✔️ Mini Task 1: Take the GPT 4, DeepSeek, Copilot, Grok outputs and use the <https://www.grammarly.com/paraphrasing-tool> and <https://www.zerogpt.com/paraphraser> paraphrasing tools in order to paraphrase the original text

✔️ Mini Task 2: Input that paraphrased text as well as the input question into the website

✔️ Mini Task 3: Note down what the similarity score is and whether its AI generated or not.

Project Presentation/Demo 6 May 1-, 2025

(95-100% of the project must be done)

(any videos/pictures for the presentation back it up somewhere cuz google drive will be slow, also back up the presentation too)

(copy everything to a USB so that you can use someone else's computer if something happens to your own)