Universidade Federal de Ouro Preto
Departamento de Computação e Sistemas – DECSI
Redes de Computadores I
Prof. Filipe Ribeiro

Atividade Prática

Objetivo

Praticar a programação para redes de computadores com a utilização de *Sockets* e *Threads*. A atividade deve ser feita em grupos de até 3 alunos.

1 – Introdução

Algumas publicações da mídia tradicional de cerca de 10 anos atrás, mostraram que um computador passou pelo teste de Turing. Em linhas gerais, o referido teste verifica se um humano confundiria a resposta oferecida por um computador como se fosse uma resposta fornecida por um humano. De acordo com o teste, se 30% dos humanos consultados acreditaram que se trata de outro humano, a máquina passa no teste de Turin. À época da publicação acima, o Chat-GPT e outras ferramentas de IA generativa não estavam disponíveis, e conforme publicações recentes, o próprio GPT-4 também teria passado no famoso teste proposto décadas atrás.

A proposta geral deste trabalho é que cada grupo planeje uma aplicação para fazermos o teste de Turing com alguma das IA's generativas disponíveis no mercado (Chat GPT, Claude, Copilot, etc).

2 - Orientações

1 – Acesso à IA generativa - em primeiro lugar, será necessário garantir o acesso a alguma IA generativa por meio de uma API (*Application Programming Interface*), isto é usando código-fonte em Python. O acesso com API permite que um programa Python faça uma requisição à ferramenta de Inteligência Artificial de maneira automatizada, simulando a consulta feita por meio de uma página web, por exemplo.

Cada equipe pode escolher utilizar aquela de sua preferência ou que já tenha tido contato anterior. Caso ainda não tenha tido nenhum contato, sugere-se a utilização do site rapidapi.com, que oferece acesso a uma vasta gama de serviços incluindo o GPT-4 por meio de API's em diversas linguagens, incluindo Python.

O site oferece acesso gratuito ao GPT-4 por exemplo com duração experimental de 14 dias, e 50 requisições. Neste caso, sugere-se que os membros do grupo gerenciem entre si a criação das contas na plataforma, de forma que após expirar o tempo de uso de uma conta, outra conta possa ser criada para dar continuidade aos testes. Da mesma forma, espera-se que uma chave funcional seja entregue na versão submetida no Moodle para que o professor possa testar.

2 – Aplicação desenvolvida -

A arquitetura da aplicação será como na figura abaixo, tendo uma aplicação cliente e uma aplicação servidora, ambas desenvolvida em Python, conforme exemplo apresentado em sala de aula.

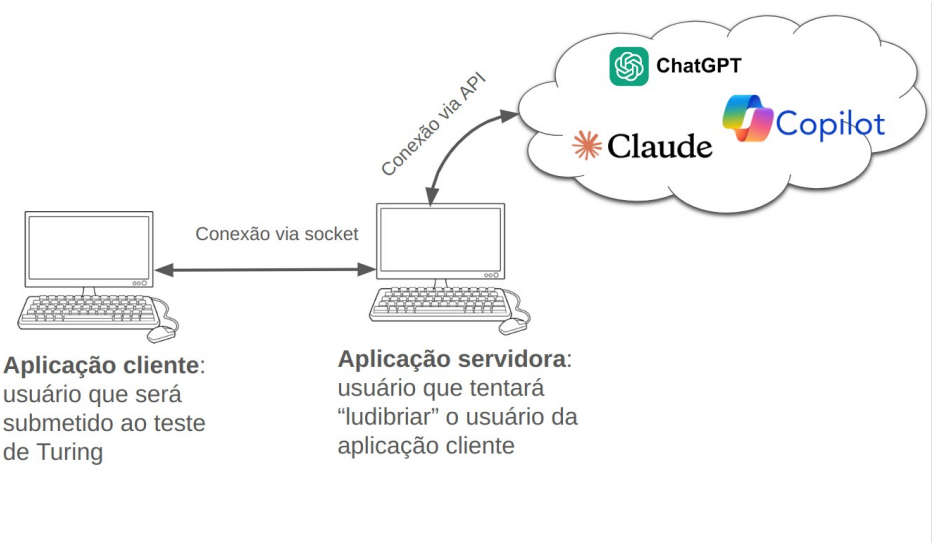

Figura 1: Arquitetura do sistema a ser desenvolvido.

Servidor: o servidor deverá receber conexões de aplicações clientes que irão passar pelo teste de Turing. De maneira geral, o servidor irá receber uma pergunta da aplicação cliente e deverá respondê-la. O servidor possuirá dois modos de funcionamento: 1 – automático, em que a resposta encaminhada ao cliente será sempre proveniente da IA generativa; e 2 – controlado, em que haverá um usuário indicando se a resposta enviada ao cliente será da IA generativa ou uma elaborada por ele mesmo. Caso a opção escolhida seja a resposta elaborada pelo usuário, ele mesmo irá digitar a resposta que será na sequência encaminhada como resposta.

O principal objetivo da aplicação servidora é ludibriar o cliente, de forma que ele sempre pense que a resposta é proveniente de um humano normal. Neste sentido, alguns artifícios podem ser úteis como por exemplo o teste *a priori* de prompts submetidos a IA que obtenham respostas mais parecidas com uma resposta rápida de um ser humano e não uma resposta tão elaborada. Note que uma resposta elaborada humana demandaria muito tempo para ser bem escrita. Neste sentido, uma boa alternativa seria esperar um tempo mínimo antes de retornar a resposta ao cliente nos casos em que ele virá da IA generativa.

No início da execução da aplicação servidora, deverão ser exibida duas configurações para que o usuário defina: o tipo de execução (se automático ou controlado) e o tempo de espera padrão para envio da resposta da IA para o cliente (dada em segundos).

Cliente: a aplicação cliente será executada pelo usuário que passará pelo teste de Turing e terá fluxo mais simples do que a aplicação servidora. Antes de iniciar o teste, o usuário deverá informar seu nome para fins de registro no servidor. Na sequência, o usuário poderá fazer uma pergunta qualquer ao servidor, e após o retorno da resposta, deverá ser questionado se pensa que a resposta é proveniente de um humano ou de alguma inteligência artificial. Por fim, ele será informado se sua escolha quanto a origem da resposta está correta. Após o término do fluxo mencionado acima, a aplicação cliente deverá questionar ao usuário se ele deseja fazer uma nova pergunta repetir o ciclo caso a resposta seja positiva ou encerrar a aplicação em caso negativo. Ao final da execução, o usuário deverá ser informado sobre a quantidade de respostas

provenientes de IA x quantidade provenientes de humanos bem como a quantidade de vezes que o usuário acertou a fonte.

Controle das respostas: O servidor deverá guardar um histórico das perguntas que foram feitas pelas aplicações clientes, bem como as respostas enviadas, e se o usuário da aplicação cliente acertou o autor da resposta. Além disso, o servidor deverá manter o ranking dos usuários que utilizaram o serviço e o percentual de acertos do mesmo. Sugere-se a utilização de arquivos para salvar os dados.

Desafio (Até 3 pts extras):

- Utilização de interface gráfica que facilite a utilização das aplicações cliente e servidora. O simples fato de ter uma interface gráfica não garante pontos extras, caso a interface não seja adequada ou seja ineficiente.

Obs: Para a obtenção de pontos extras é necessário concluir o jogo básico antes e encaminhar um e-mail à parte informando quais funcionalidades extras foram implementadas.

3 – Linguagem de Implementação

A linguagem utilizada deverá ser **Python** e o tipo de saída/entrada ficará à sua escolha. Pode-se utilizar janelas com interface gráfica de botões, caixas de texto, ou o próprio prompt.

OBS: Não pode ser desenvolvimento Web.

4 – Nota e detalhes da submissão

Valor da atividade: 20 pontos.

Data da entrega: 15/09/2024

Formato de submissão:

1 - arquivo ".zip" contendo os arquivos do projeto. Além do código-fonte, o arquivo compactado também deverá incluir dois arquivos, README.txt e o relatório.pdf. O arquivo README.txt deverá detalhar como executar o jogo (por exemplo: "Execute o arquivo tal que será o servidor; Em seguida execute tal arquivo...".

2 – O relatório em formato PDF deve descrever como foi feito o trabalho e quais foram os resultados obtidos. Use desenhos, diagramas, figuras, todos os recursos que permitam ao professor compreender como o grupo estruturou o trabalho e quais resultados obteve. O objetivo é o professor entender como o grupo fez o trabalho e como o trabalho funciona.

3 – o arquivo zip deverá conter o nome e ultimo sobrenome de cada aluno do grupo, por exemplo "filipe_ribeiro_maria_silva.zip".

Apresentações:

Caso seja necessário, o professor poderá convocar o grupo para apresentar o trabalho desenvolvido.

Observações finais e plágio:

1 – Um único aluno do grupo deverá fazer uma submissão via Moodle de um arquivo .zip.

2 – Entrega de arquivos em formatos diferentes do especificado ocasionará desconto na pontuação.

3 – Não encaminhe sua atividade para outros grupos. Os plágios serão punidos com rigor. Inclusive quem compartilhou.

4 – Sobre o plágio:

- Atividades plagiadas serão zeradas para ambos (quem fez e quem copiou)

- Os softwares para detecção de plágio identificam a cópia mesmo que se façam alterações.