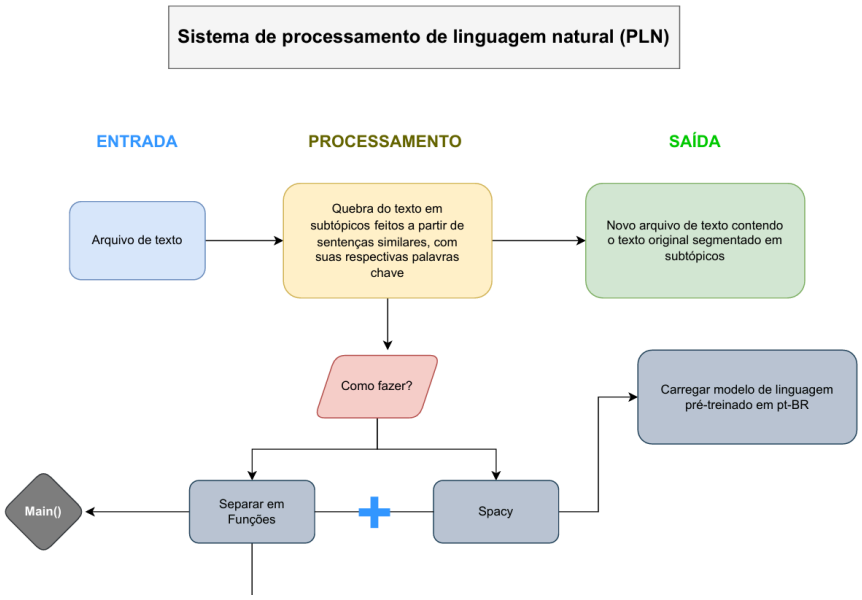
Faculdade de Computação
Projeto Prático de Programação 1 Período: 2024/4
Alunos: Andrey Silva da Silva, Hiago da Silva dos Anjos, Maria Eduarda Borges Pinheiro

- **Projeto**

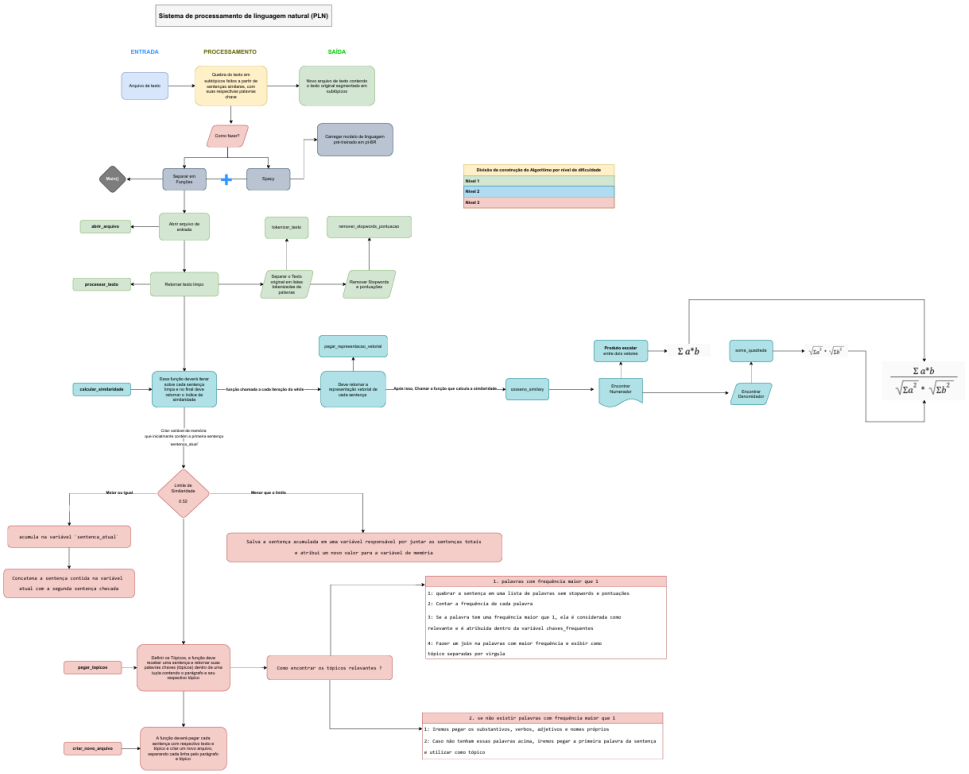
O objetivo deste projeto é a implementação de um sistema de Processamento de Linguagem Natural (PLN) utilizando os recursos abordados em sala de aula. O projeto consiste na identificação de tópicos a partir de um texto, buscando mapear e segmentar o conteúdo em subtópicos. Cada subtópico será formado por um conjunto de sentenças com alta similaridade, preservando a ordem original do documento. Para cada subtópico, será atribuído um rótulo que represente seu tema principal. Esse rótulo será composto por uma lista de, no máximo, cinco palavras pertencentes às classes de substantivos ou verbos, e todas as palavras selecionadas deverão estar presentes no próprio subtópico. Foi utilizado as bibliotecas SpaCy, Collections e a Math.



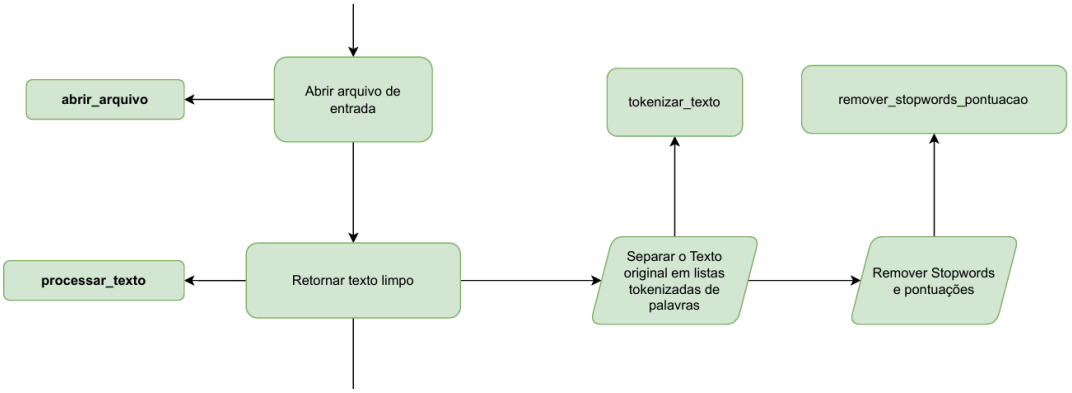
- **Início**

A equipe composta por Andrey, Hiago e Maria Eduarda iniciou a organização do projeto final no dia 25/02/2025, com a criação do grupo no aplicativo de mensagens, por ser uma atividade importante e ser um programa extenso, o código foi divido em três níveis, cada nível para um membro da equipe, o Hiago, por ter uma certa experiência com trabalhos em grupo, fez um esquema para os membros

entenderem a linha que o código deveria seguir para maior eficiência, as cores representam respectivamente os níveis, com exceção do encontro de limite da similaridade, pois foi uma tarefa do nível dois e está com a coloração da nível três. A divisão ficou a seguinte, cor verde, nível um, Hiago, cor azul, nível dois, Andrey, cor vermelha, nível três, Maria Eduarda.



- **Nível 1**



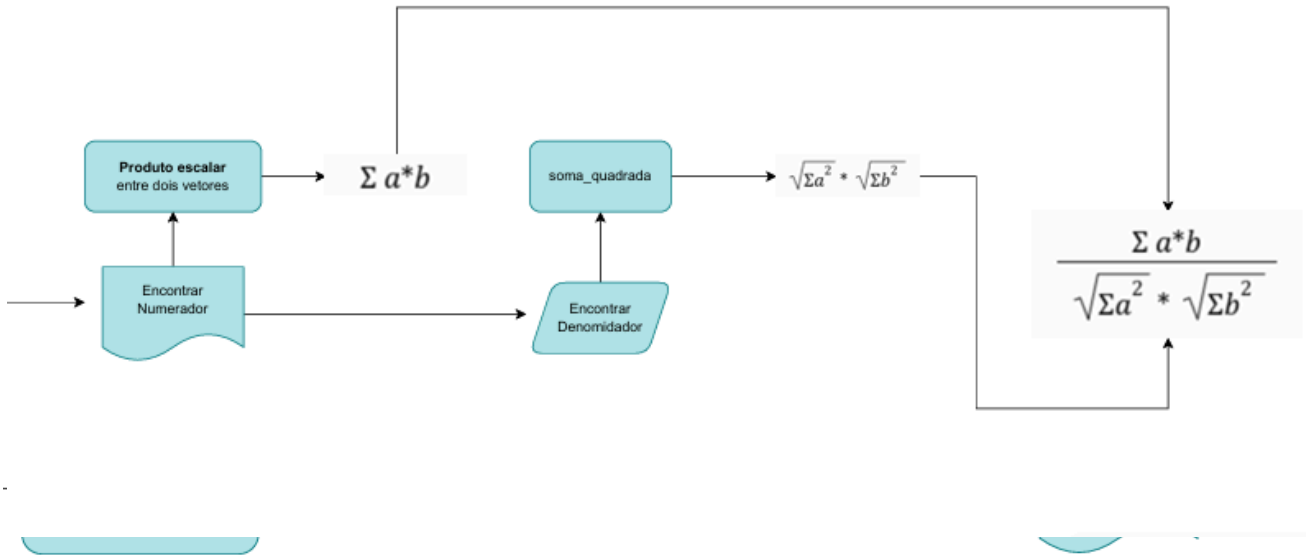
O primeiro nível do projeto ficou sob responsabilidade do Hiago. Essa etapa envolvia a abertura do arquivo de texto, a tokenização das sentenças e a remoção de stopwords e pontuações. Com esse objetivo, começou a desenvolver o código, implementando as funções necessárias para garantir o processamento correto do texto. Inicialmente, foi criada a função “abrir_arquivo”, responsável por carregar o texto do arquivo especificado e lidar com possíveis erros de arquivo não encontrado.

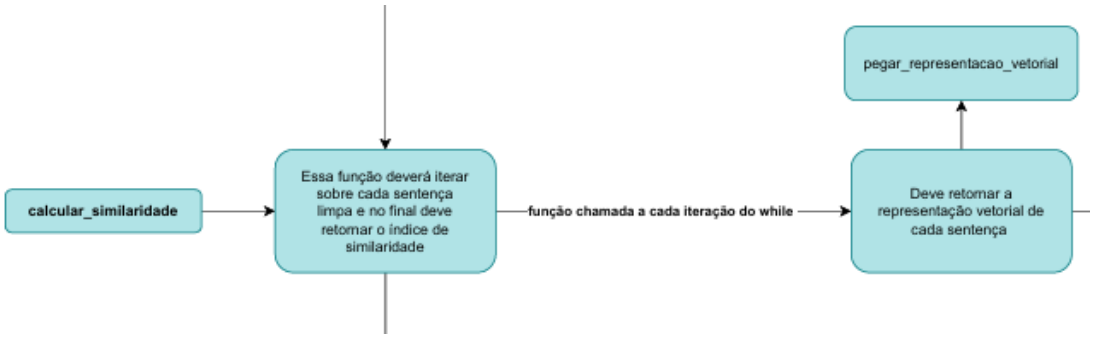
Em seguida, foi implementada a função ‘remover_stopwords_pontuacao’, utilizando a biblioteca SpaCy para processar as sentenças, removendo pontuações e palavras irrelevantes para a análise.

Depois, foi criada a função “processar_texto”, que recebe o texto extraído do arquivo e realiza a tokenização por meio da biblioteca SpaCy, separando o conteúdo em sentenças. Cada sentença foi limpa utilizando a função “remover_stopwords_pontuacao”, gerando uma versão filtrada do texto. No final do processamento, essa função retorna uma tupla contendo a sentença limpa e a sentença original.

Por fim, foi criada a função “start” para organizar o fluxo do programa. Essa função abre o arquivo de entrada, processa o texto e retorna as sentenças originais e suas versões limpas. Ao rodar o código, foi possível verificar que a extração e limpeza das sentenças ocorreram corretamente, garantindo que a próxima fase do projeto pudesse trabalhar com uma estrutura organizada em uma lista contendo as tuplas retornadas pela função “processar_texto”.

- **Nível 2**





O segundo nível do projeto ficou sob responsabilidade do Andrey. A parte do código apresentado tem como objetivo realizar a concatenação de sentenças com base na similaridade entre elas, utilizando o método da similaridade cosseno para verificar a proximidade semântica entre pares de sentenças consecutivas. A função recebe uma tupla e renomeia as variáveis para facilitar a compreensão:

- texto processado[0]: lista maior que contém outras listas com as sentenças processadas em tokens, renomeada para “sentencas_processadas”.
- texto processado[1]: lista maior que contém outras listas com as sentenças originais em tokens, renomeada para “sentencas_originais”.

Primeiramente, ele percorre a lista de “sentencas_processadas”, comparando cada par de sentenças consecutivas. Para cada par de sentenças, ele:

- Chama a função juntar, que recebe o índice da lista “sentencas_processadas” e a própria lista. Com essas informações, a função adiciona todos os tokens do índice da lista informado até o próximo índice em uma variável, utilizando set() para garantir que os tokens não se repitam. No final, a função retorna essa variável como uma lista (palavras_unicas_lista) usando list().

Após isso, são geradas duas listas para cada sentença, “representacao_vetorial01” e “representacao_vetorial02”, que são listas de zeros com o mesmo tamanho de “palavras_unicas_lista”. Cada índice dessas representações corresponde a uma palavra na lista de palavras únicas.

Com a lista contendo as palavras únicas de ambas as sentenças (palavras_unicas_lista), é necessário realizar a contagem da presença ou ausência dessas palavras nas representações de cada sentença. Para isso, ele verifica em qual posição da lista “palavras_unicas_lista” a palavra aparece e incrementa o valor naquela posição na representação vetorial, utilizando o .index().

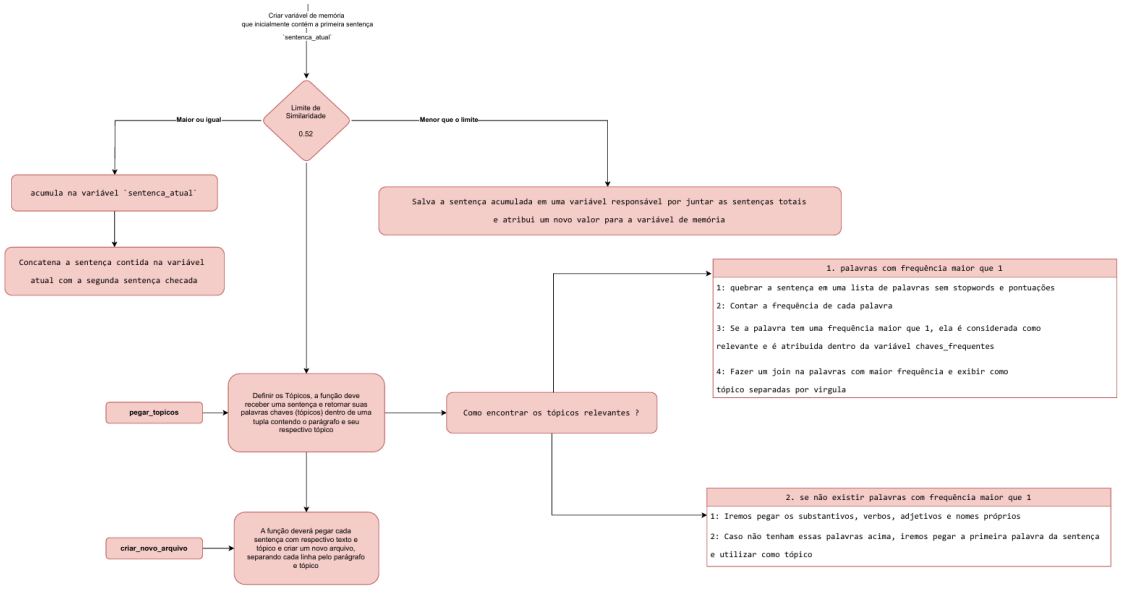
- O código usa a função “similaridade_cosseno” para calcular a similaridade entre os vetores de duas sentenças consecutivas. Essa função recebe dois vetores, que são a representação vetorial das sentenças, cada um representando A e B, de acordo com a fórmula. O cálculo é feito por meio de um laço, onde é realizada a multiplicação dos elementos dos vetores A e B (numerador) e a potenciação ao quadrado dos elementos dos vetores A e B separadamente (denominadores A e B). A cada iteração, o resultado da multiplicação é somado a uma variável que representa o somatório.
- Fora do laço, tira-se a raiz quadrada do somatório da potenciação utilizando a função sqrt da biblioteca math. Por último, é verificado se o denominador é igual a zero; se sim, é retornada uma similaridade igual a zero. Caso contrário, é retornada a divisão do numerador pelo denominador.

Se o resultado da similaridade for maior ou igual ao limite pré-definido, as sentenças são concatenadas. Se não, a sentença atual é adicionada à lista de resultados, e a próxima sentença se torna a nova sentença atual a ser analisada. Ao final do loop, o código retorna uma lista com as sentenças concatenadas.

Dificuldades e Erros Durante Este Nível:

- Dificuldades de usar o GitHub: Por falta de prática, foi difícil utilizar o GitHub para um projeto em grupo pela primeira vez. No entanto, as dificuldades foram superadas com comunicação e trabalho em equipe.
- Erro de lógica ao representar as sentenças em vetores: Na primeira versão desse nível, foi aplicada uma lógica errada onde a contagem de presença ou ausência das palavras nas sentenças era adicionada no mesmo lugar. Por exemplo, em um texto com 8 sentenças, era retornado 7 vetores binários, quando deveria ser 8. A similaridade era calculada entre esses vetores, o que resultava em similaridades altas e erros de laço na concatenação.

- **Nível 3**



O nível três do projeto ficou na responsabilidade da Maria Eduarda Borges Pinheiro, essa era onde os tópicos do texto seriam encontrados e um novo arquivo será criado. Com esse objetivo, Maria Eduarda começou o código, porém sentiu um pouco de dificuldade para encontrar os tópicos, chegou a iniciar uma função, mas não conseguiu avançar muito. Por esse motivo, chamou o Hiago da Silva que é mais experiente, juntos realizaram a fase três. O código se trata de uma função chamada “pegar_tópicos” que remove as stopwords do texto e recebe as sentenças, aquelas as quais foram separadas na fase anterior, e retorna as suas palavras chaves dentro de uma tupla contendo o parágrafo que estava e o seu tópico. Usando a função “Counter” da biblioteca collections que contabiliza a frequência de palavras, em uma repetição do tipo for, verifica-se caso a chave tem a frequência maior que o limite de similaridade adiciona-se essas em uma lista chamada “chaves_frequentes”. Usando um if, fez-se um print formatado e retornamos o texto e os tópicos.

- No else, foi criado a lista “palavras_frequentes”, essa lista vai armazenar as palavras que não possuem a frequência maior que esse limite, mas são o tópico principal da sentença, pegando a primeira palavra que é um verbo, um substantivo, um pronome ou um adjetivo usando a estrutura for, que funciona da seguinte maneira, pega-se os tokens e os concatena em uma única string, processada com um modelo NLP, dentro dessa estrutura de repetição, abrimos uma possibilidade, um if, que é para caso o token se encaixe ou em verbo, adjetivo, nome próprio ou substantivo, será adicionado à lista palavras relevantes.

Para gerar os tópicos a partir desses tokens já separados, usa-se a seguinte ideia, se existir pelo menos duas palavras relevantes, as duas primeiras são usadas para formar uma string de tópicos que serão imprimidas ao usuário na formatação seguinte “<Tópicos: assunto1, assunto 2>”, função retorna uma tupla "(texto, topicos_gerados)", associando o texto original aos tópicos extraídos.

- Caso houver apenas uma ou nenhuma palavra relevante, a primeira palavra do texto é usada como tópico, também retorna a tupla dos textos.

A segunda função do nível três é a criação do arquivo de texto, chamada de “criar_aquivos”, usando como parâmetro a "lista_sentencas", que é a tupla no formato (texto, tópicos), retornada na função anterior. Para criar um arquivo, se abre um arquivo para escrita ("w") com codificação UTF-8, a fim de evitar qualquer possível erro, então, usa-se mais uma estrutura for para escrever a tupla (texto, tópicos) no arquivo, seguidos de uma linha em branco.

- **Conclusão**

O projeto para rotulação de textos em subtópicos com base na similaridade semântica utilizando o Processamento de Linguagem Natural (PLN) foi implementado com sucesso pela equipe. Pela divisão do programa em três níveis, permitiu que os membros da equipe fizessem diferentes etapas do processo com maestria, desde a limpeza do texto até a extração de tópicos em si. Apesar das dificuldades durante o processo, como os erros do Github, os empecilhos de lógica e os reajustes que tiveram que ser feitos na lógica da vetorização, a equipe os superou por meio da colaboração mútua e o aprendizado com os erros. Ao final, o grupo conseguiu desenvolver o sistema funcional de identificação e organização de tópicos em textos, aplicando os conceitos estudados em sala de aula e reforçando a importância de projetos em equipe e da resolução de problemas na implementação de técnicas de Processamento de Linguagem Natural.