1. Pré-processamento do Texto (PipeList):

**ArrayList<Pipe> pipeList = new ArrayList<>();**

**pipeList.add(new CharSequenceLowercase());**

**pipeList.add(new CharSequence2TokenSequence(Pattern.compile("\\p{L}[\\p{L}\\p{P}]+\\p{L}")));**

**pipeList.add(new TokenSequenceRemoveStopwords(new File("stoplists/pt.txt"), "UTF-8", false, false, false));**

**pipeList.add(new TokenSequence2FeatureSequence());**

* + Complexidade: O pré-processamento do texto envolve a leitura de cada palavra no documento.
    - Melhor caso, caso médio e pior caso: O(n), onde 'n' é o número total de palavras nos documentos.

1. Leitura e Processamento dos Arquivos:

**File directory = new File("exemplos");**

**File[] files = directory.listFiles((dir, name) -> name.endsWith(".txt"));**

* + Complexidade: A leitura e processamento dos arquivos ocorrem para cada arquivo na pasta "exemplos".
    - Melhor caso, caso médio e pior caso: O(k \* n), onde 'k' é o número de arquivos e 'n' é o número total de palavras nos documentos.

1. Treinamento do Modelo (ParallelTopicModel):

**ParallelTopicModel model = new ParallelTopicModel(numTopics); model.addInstances(instances);**

**model.setNumThreads(2); model.setNumIterations(1000); model.estimate();**

* + Complexidade: O treinamento do modelo de tópicos é geralmente dependente do número de iterações e do número de tópicos.
    - Melhor caso, caso médio e pior caso: O(t \* n \* k \* iterations), onde 't' é o número de tópicos, 'n' é o número total de palavras nos documentos e 'iterations' é o número de iterações.

1. Construção do Grafo (JGraphT):

**Graph<String, DefaultEdge> graph = new SimpleGraph<>(DefaultEdge.class);**

* + Complexidade: A construção do grafo envolve adicionar vértices e arestas para cada arquivo e autor.
    - Melhor caso, caso médio e pior caso: O(k \* (m + a)), onde 'k' é o número de arquivos, 'm' é o número médio de autores por arquivo e 'a' é o número total de autores.

1. Impressão do Grafo:

**System.out.println("Grafo de Tópicos e Coautores:"); System.out.println(graph.toString());**

* + Complexidade: A impressão do grafo não afeta significativamente a complexidade assintótica.

1. Extração de Tópicos e Autores de Arquivo (extractTopicAndAuthorsFromFile):

**private static TopicAndAuthors extractTopicAndAuthorsFromFile(File file)**

* + Complexidade: A extração de tópicos e autores de um arquivo é linear em relação ao número de linhas e caracteres no arquivo.
    - Melhor caso, caso médio e pior caso: O(c), onde 'c' é o número total de caracteres no arquivo.
* **Melhor caso: O(k \* n \* t \* iterations)**
* **Caso médio: O(k \* n \* t \* iterations)**
* **Pior caso: O(k \* n \* t \* iterations)**