SI400 – Programação Orientada a Objetos II

Prof. André F. de Angelis
Oferecimento 2S2020

Projeto I: Pré analisador de textos

Objetivo: desenvolver um programa em **Java** para realizar a etapa de pré-processamento de um texto a ser analisado como dígrafo pela ferramenta de software livre Gephi. O programa deverá aceitar uma lista de nomes de arquivos do tipo texto simples (.txt) e, para cada um deles, gravar uma saída no formato .csv contendo listas de adjacências indicando sucessão de palavras para posterior construção dos dígrafos, como descrito a serguir:

- cada palavra é um nó único do dígrafo (início de uma lista de adjacência);
- o nó A tem um arco para o nó B se e somente se a palavra representada por B segue aquela representada por A no texto;
- cada linha do arquivo de saída inicia-se com a palavra que representa o nó de origem, seguida das palavras que são os destinos dos arcos que partem dela, separadas por espaços ou vírgulas (formato Gephi).

Perceba que toda a pontuação, aspas, travessões e tais devem ser eliminados, enquanto que todo o texto deve ser convertido para minúsculas. Cada palavra do texto aparece uma e uma só vez como a palavra inicial das linhas, em ordem alfabética. A lista dos destinos pode ter uma ou mais palavras, mas sem repetição na linha.

Veja o exemplo:

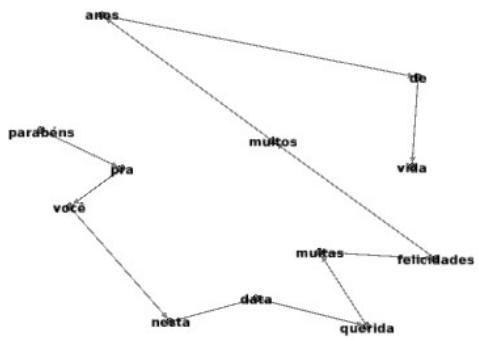
Entrada:

Parabéns pra você
Nesta data querida
Muitas felicidades
Muitos anos de vida

Saída:

anos, de
data, querida
de, vida
felicidades, muitos
muitas, felicidades
muitos, anos
nesta, data
parabéns, pra
pra, você
querida, muitas
você, nesta

Dígrafo:



Os seus arquivo de saída serão submetidos ao Gephi para revelar o dígrafo correspondente as sucessões de palavras. Veja o exemplo a seguir, com trecho da canção “Luar do Sertão”.

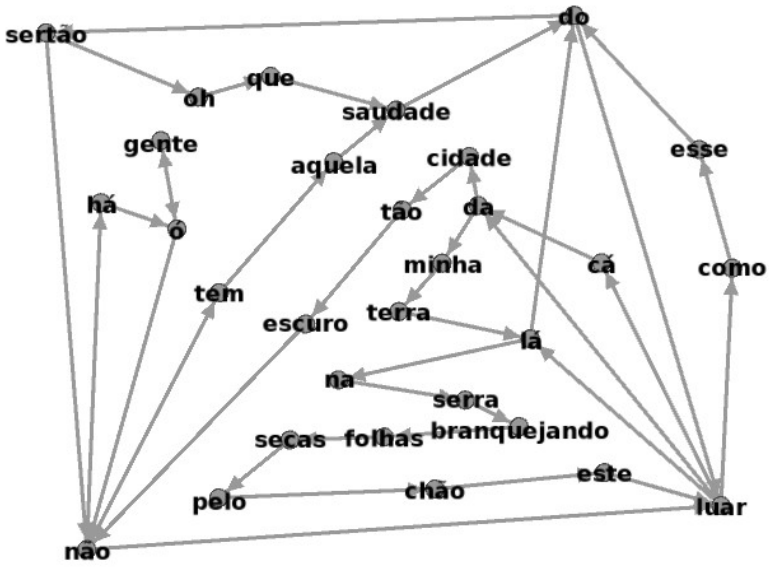
Entrada:

Não há, ó gente, ó não
Luar como esse do sertão
Não há, ó gente, ó não
Luar como esse do sertão
Oh que saudade do luar da minha terra
Lá na serra branquejando folhas secas pelo chão
Este luar cá da cidade tão escuro
Não tem aquela saudade do luar lá do sertão
Não há, ó gente, ó não
Luar como esse do sertão
Não há, ó gente, ó não
Luar como esse do sertão

Saída:

aquela, saudade
branquejando, folhas
chão, este
cidade, tão
como, esse
cá, da
da, cidade, minha
do, sertão, luar
escuro, não
esse, do
este, luar
folhas, secas
gente, ó
há, ó
luar, como, lá, da, cá
lá, na, do
minha, terra
na, serra
não, luar, há, tem
oh, que
pelo, chão
que, saudade
saudade, do
secas, pelo
serra, branquejando
sertão, oh, não
tem, aquela
terra, lá
tão, escuro
ó, gente, não

Dígrafo:



O programa deve ser constituído de, **ao menos,** quatro classes com as seguintes responsabilidades:

1. iniciar o programa, instanciar o objeto de controle e tratar as exceções;
2. controlar a execução do programa, coordenando as demais classes;
3. ler arquivos de texto, separar as palavras, convertê-las em minúsculas e remover pontuação;
4. gravar o arquivo de saída no formato adequado com as palavras ordenadas.

Use mais classes se desejar. Se achar conveniente, pesquise como em implementar um *multimap,* já que está classe não é disponível na biblioteca padrão da linguagem. Podem ser úteis as seguintes classes e interfaces da biblioteca do Java:

- ArrayList
- Map
- TreeMap

Implemente e depure o programa, gere a documentação com a ferramenta *javadoc*. O professor fornecerá exemplos de textos de entrada e de saída,para que você possa testar seu programa, e dois textos de validação. Submeta os arquivos de validação ao seu programa e, com as saídas produzidas, gere os correspondentes dígrafos no gephi, cuidando para que sejam legíveis.

A entrega do projeto será feita como segue, no Moodle:

- 1 arquivo compactado contendo todo ocódigo fonte (zip);
- 1 arquivo jar executável (.jar);
- 1 arquivo do relatório de contribuição (.pdf);
- 1 arquivo de figura com o diagrama de classes UML do projeto (.jpg);
- 2 arquivos de figuras dos grafos de validação (.jpg);
- 1 link para o vídeo de apresentação (direto no campo de texto).

O vídeo deve ter até 12 minutos com a apresentação do projeto pelo grupo, enumerando os principais desafios encontrados, as soluções adotadas, as funcionalidades das classes e as estruturas de dados usadas. Mostre também o programa rodando em seu vídeo.

Se o grupo utilizou uma metodologia de desenvolvimento padronizada (*XP, Scum, Rup*, …), descreva no vídeo a aplicação dela no projeto (não é para dizer o que é ou como funciona, mas indicar como ela foi usada).

Critérios de Avaliação

- Só serão considerados os programas que compilem sem erro e executem corretamente a(s) funcionalidade(s) requerida(s).

#	Item	Max. pontos
01	Entrega completa de todos os ítens solicitados	0,5
02	Arquitetura solicitada com 4 ou mais classes	0,5
03	Javadoc para todas as classes e todos os métodos	0,5
04	Bônus para código e documentação totalmente em inglês	0,3
05	Arquitetura de classes (estruturação, divisão de responsabilidades, diagrama UML)	1,0
06	Qualidade do código (estilo, indentação, organização, limpeza)	1,0
07	Consistência do código, incluindo nomeação de identificadores	0,5
08	Uso de estruturas de dados apropriadas	0,5
09	Bônus para implementação de estruturas de dados não disponíveis no Java como classes	0,5
10	Tratamento de exceções	0,3
11	Bônus para lançamento de exceções apropriadas	0,3
12	Adequação dos algoritmos	1,0
13	Adequação do vídeo de apresentação (objetivo, tempo, conteúdo)	0,3
14	Clareza na demonstração de desafios, soluções e funcionalidades	0,4
15	Clareza na apresentação das estruturas de dados e algorítmos	0,3
16	Demonstração do programa em execução	0,2
17	Correção dos dígrafos gerados pelo Gephi	0,2
18	Bônus para metodologia de desenvolvimento padronizada	0,5
19	Recursos extras significativos	0,2
20	Aspecto geral do trabalho	1,0
	Total	10,0