Um Classificador de Seriedade de Notícias Baseado em Análise de Texto

Gustavo F. Camilo e Leonardo G. C. e Silva

25 de Novembro de 2019

1 Introdução

O paradigma de programação orientada a objetos é inserida na linguagem C, sendo esta versão chamada de C++. A possibilidade do uso e criação de Classes elevam o grau de modularidade dos códigos e facilita o entendimento do programa como um todo. Dessa forma, aproveitando dessas qualidades, este relatório descreve a **parte 3** do trabalho que consiste na implementação em C++ de uma interface com um classificador de seriedade de notícias em Perl. Além disso, também propõem-se 4 funcionalidades extras escritas em C++ para o gerenciamento dos arquivos e criação de uma interface amigável com o usuário. Toda implementação e os arquivos auxiliares estão disponíveis no Github pelo endereço *https://github.com/JoltLeo/News_analyser*.

2 Implementação do Programa

O trabalho desenvolvido usou a linguagem C++ para a implementação de um gerenciador de um sistema que classifica a seriedade das notícias. Essa etapa do trabalho implementa cinco funções de gerenciamento de notícias e do sistema para classificar o tópico e a seriedade de uma notícia. Além disso, essa etapa integra o gerenciador feito em linguagem C++ com o analisador de texto implementado em Perl da etapa anterior e implementa uma interface amigável com o usuário. As cinco funções são descritas abaixo:

- **Exibição de uma notícia**: recebe como entrada uma *string* com o nome do arquivo de texto em da notícia e não possui retorno. Caso a notícia seja encontrada, seu conteúdo é impresso na tela, se não uma mensagem de erro é impressa na tela;
- **Listar os títulos das notícias classificadas como menos sérias**: não recebe argumentos do usuário e não possui retorno. A função imprime os títulos de todas as notícias classificadas como menos séria na tela;
- **Classificar a seriedade de uma ou todas as notícias**: pode receber ou não argumentos do usuário. Essa função chama o analisador de texto implementado em Perl para classificar a notícia. A função possui argumento opcional do nome do arquivo texto com a notícia a ser classificada. Caso não receba esse argumento, o modulo Perl classifica todas as notícias do diretório padrão de notícias. A função imprime na tela o número de notícias sérias e não sérias;
- **Adicionar ou remover autor/site da lista de autores/sites menos confiáveis**: recebe como argumento a operação, i.e, adicionar ou remover, além de receber o nome do autor ou endereço do site a ser adicionado ou removido e imprime na tela uma mensagem de sucesso ou erro;
- **Listar o número total de notícias e a relação das classificações**: não recebe argumentos do usuário e não possui retorno. Essa função imprime na tela uma tabela contendo o número total de notícias classificadas, o número de notícias classificadas como sérias e o número de notícias classificadas como menos sérias.

O trabalho foi separado em oito arquivos escritos em C++, cinco implementando cada uma das cinco funções, um menu para interação com o usuário, um programa principal que implementa a proposta e outro que implementa a classe de erros. Os oito são listados e resumidos abaixo:

- **show_news.cpp**: Implementa a função que exibe uma notícia ao usuário dado o caminho do arquivo.
- **classification_statistics.cpp**: Implementação da função classification_statistics que recebe o nome do arquivo com as classficações das notícias para mostrar ao usuário informações sobre a classificação.
- **change_blacklist.cpp**: Implementação de uma função que altera a lista contendo o nome de fontes menos confiáveis. Essa função permite que usuários possam modificar a lista, gerando uma experiência mais personalizada as suas necessidades.
- **error_class.cpp**: Implementação da *classe Error_class* que identifica se o retorno de uma função é de fato um erro. Caso positivo, imprime na tela a mensagem de erro referente ao código de retorno da função;
- **list_less_serious.cpp**:Implementação da função list_less_serious que lista as notíciaas classificadas como menos sérias por título.
- **perl_wrapper.cpp**: Implementação da classe da interface, *Perl_wrapper*, entre o C++ e o Perl. Essa classe faz o gerenciamento dos parâmetros passados e os retornos da interface do Perl;
- **menu.cpp**: Implementação da classe *Menu* que gerencia a impressão do menu principal do programa, a entrada de dados pelo usuário, as chamadas dos outros módulos que realizam as funcionalidades propostas para a parte do programa em C++ e a análise e tratamento dos erros;
- **main.cpp**: Programa principal que utiliza um objeto da classe *Menu* para executar métodos que realizam as funcionalidades propostas de acordo com a opção do menu escolhida pelo usuário.

O programa principal deve ser criado pelo *src/C++/Makefile* ao executar o comando '*make install*' no diretório *src/C++/*, criando o executável *news_analyser*. Para desinstalar o program, execute o comando *make clean* nesse mesmo diretório.

2.1 Estrutura dos Diretórios e Arquivos

Por decisão de projeto e organização, todos os códigos em C++ e arquivos de texto estão no diretório *src/C++/*, e os códigos usados para os testes estão em *src/C++/test_codes*. O arquivo *constantes.h* contem a declaração de todas as constantes utilizadas no programa, enquanto o arquivo *error_messages.h* mapeá todos os códigos de error do C++ e Perl e suas respectivas mensagens de retorno. O programa principal está implementado no arquivo main.cpp e a interface com o classificador de notícias em Perl é feita pela classe Perl_wrapper.

2.2 Implementação das Classes em C++

A implementação da interface contou com a criação de 3 classes escritas em C++, denominadas *Error_class*, *Perl_wrapper* e *Menu*.

2.2.1 Classe *Error_class*

A classe *Error_class* tem a finalidade de analisar os códigos de retorna das funções, identificar a respectiva mensagem de erro e realizar sua impressão na tela. Para a visualização das mensagens de erro, utilizou-se uma sobrecarga no operador ¡¡ da classe *ostream*. Os atributos e métodos da *Error_class* são apresentados no diagrama UML da Figura 1.

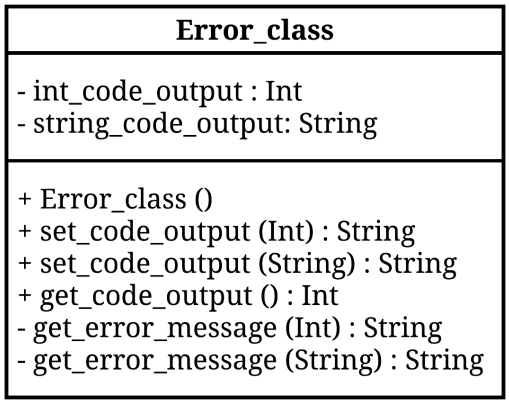


Figura 1: Diagrama UML da classe **Error_class**.

Vale ressaltar que a sobrecarga dos métodos *get_code_output* e *get_error_message* são realizadas para possibilitar o tratamento tando de códigos de retorno do tipo *Int* quando do tipo *String*.

2.2.2 Classe *Perl_wrapper*

A classe *Perl_wrapper* possibilita a criação da interface com o classificador de notícias em Perl. Essa funcionalidade é realizada pelo método *classify_news (String)* que recebe como argumento uma *string* contendo o arquivo da notícia para análise. A interface acontece ao utilizar o modulo *classify_news.pm* escrito em Perl que utiliza os demais módulos apresentados na Parte 2 deste trabalho e realiza todos os procedimentos necessários para classificação da notícia. O método retorna um valor *Int*, sendo o retorno 0 uma notícia séria, 1 uma notícia NÃO seria e retornos negativos significam erro. Os atributos e demais métodos da *Perl_wrapper* são apresentados no diagrama UML da Figura 2.

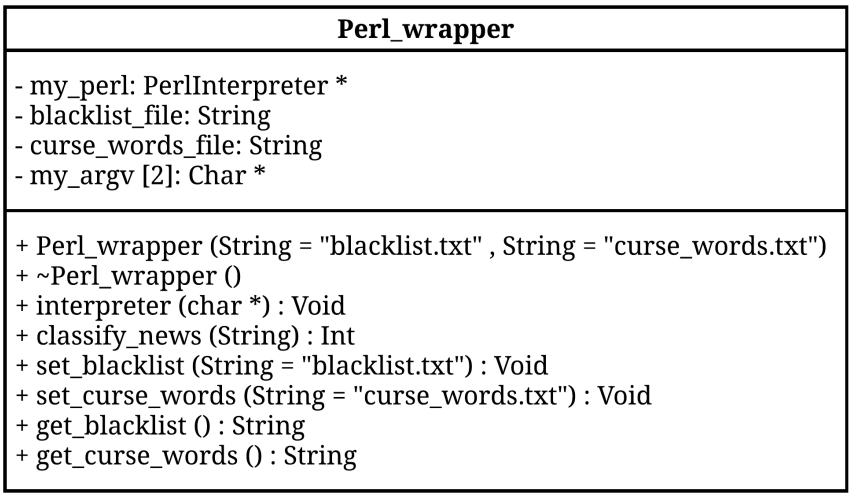


Figura 2: Diagrama UML da classe **Perl_wrapper**.

2.2.3 Classe *Menu*

A classe *Menu* busca simplificar o entendimento do programa e modular a exibição e execução das funcionalidades da interface C++. Essa classe realiza a impressão do menu principal na tela, gerencia todas as entradas do usuário ao programa, chama as funções para atender uma determinada

funcionalidade das 5 propostas. As opções do menu principal e sua execução serão demostrados na Seção 3. Um detalhe importante dessa classe encontra-se no método *menu_classify_news ()* que cria um objeto *static* da classe *Perl_wrapper*. Essa abordagem foi necessária para evitar conflitos com o interpretador do Perl, visto que pode existir somente 1. Os atributos e demais métodos da *Menu* são apresentados no diagrama UML da Figura 3.

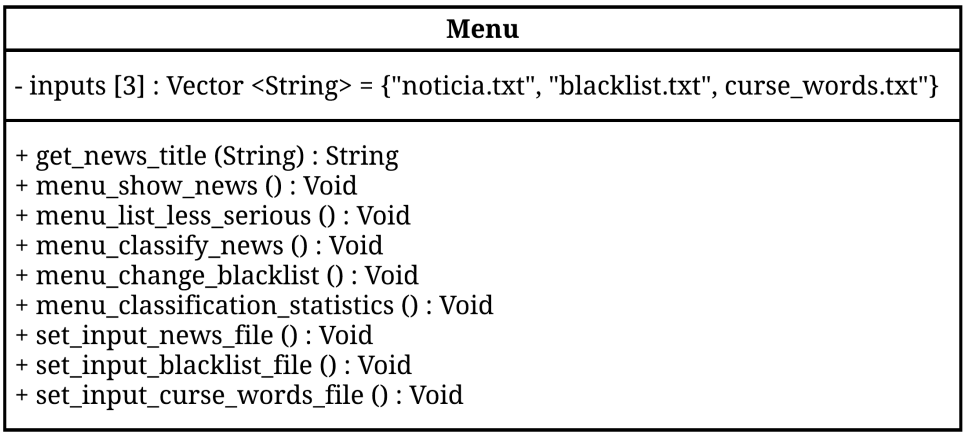


Figura 3: Diagrama UML da classe **Perl_wrapper**.

2.3 O Arquivo show_news.cpp

Esse arquivo implementa uma função que exibe a notícia ao usuário. Para isso, o programa recebe o caminho até o arquivo contendo a notícia e a exibe na tela. A Lista 1 apresenta o código dessa função.

```
#include <iostream>
#include <string>
#include <fstream>
#include "constants.h"
#include "show_news.h"

using namespace std;

int show_news (string news_filename){

    ifstream file;
    string line;

    if (news_filename.length() == 0){
        return BLANK_NEWS_FILENAME;
    }

    file.open(news_filename);
    if (file.is_open()){
        while (getline (file, line)) {
            cout << line << endl;
        }
    }else{
        //cout << "Could not open file " << news_filename << endl;
        return NEWS_FILE_ERROR;
  }
  return SUCCESS;

}
```

Lista 1: Implementação da função show_news.

2.4 O Arquivo classification_statistics.cpp

Este arquivo é responsável por entregar ao usuários as estatísticas de notícias classificadas. Para isso, o programa lê um arquivo texto que guarda estatísticas sobre as notícias classificadas. A saída da função é uma tabela mostrando ao usuário o número total de notícias classificadas, o número de notícias classificadas como séria e o número de notícias classificadas como menos séria. A Lista 2 apresenta o código que implementa essas funcionalidades.

```
#include <string>
#include <iostream>
#include <fstream>
#include <iomanip>
#include "constants.h"
#include "classification_statistics.h"

using namespace std;

int classification_statistics(string filename){
    unsigned counter_serious = 0;
    unsigned counter_less_serious = 0;
    unsigned ratio = 0;

    size_t serious_position;
    size_t less_serious_position;

    string line;
    ifstream classification_file;

    classification_file.open(filename);
    if (!classification_file.is_open()){
        return CLASSIFICATION_FILE_ERROR;
    }

    while (getline(classification_file,line)){
        serious_position = line.find (SERIOUS);
        less_serious_position = line.find(LESS_SERIOUS);

        if((!serious_position) && (!less_serious_position)){
            classification_file.close();
            return CORRUPTED_FILE;
        }
        if (serious_position != string::npos){
            counter_serious ++;
        }else{
            counter_less_serious ++;
        }
    }

    cout << "-------------------------------------------------------------------------------" <<
    endl;
    cout << "|    NUMBER OF NEWS    |    SERIOUS NEWS    | LESS SERIOUS NEWS |" <<
    endl;
    cout << "-------------------------------------------------------------------------------" <<
    endl;
    cout << setw(13) <<  counter_less_serious + counter_serious << setw(18) <<
    counter_serious << setw(20) << counter_less_serious << endl;
    cout << "-------------------------------------------------------------------------------" <<
    endl;
}
```

Lista 2: Implementação da função classification_statistics.

3 Casos de Uso

O usuário interage com o sistema através de um menu que fornece as opções de sair do programa além das funcionalidades apresentadas anteriormente. A Figura **??** apresenta uma imagem do menu que faz a interface entre o usuário e o programa.

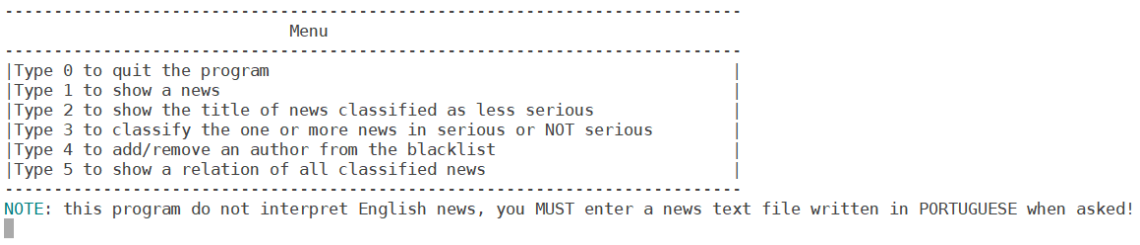


Figura 4: Menu do sistema implementado.

Ao selecionar uma opção, o programa pede ao usuário informações necessárias para executar a funcionalidade. Caso o usuário prefira, o programa é executado com as opções padrão. A Figura 5 mostra como o programa pede ao usuário as entradas necessárias quando a opção 3 é selecionada. Essa opção comunica com o analisador em Perl da etapa anterior do trabalho para classificar as notícias como sérias ou não sérias.

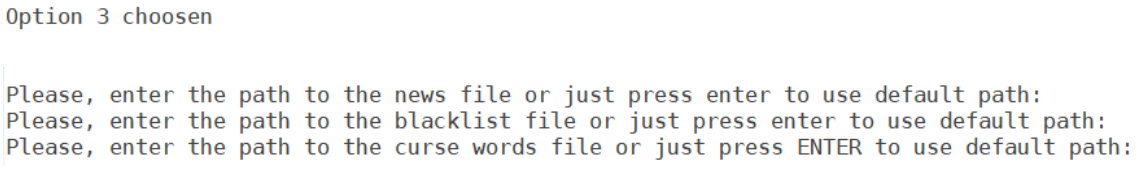


Figura 5: Programa pedindo que o usuário forneça as entradas necessárias.

Como apresentado na etapa anterior do trabalho, essa opção apresenta as estatísticas coletadas pelo programa que foram usadas para classificar a notícia. A Figura 6 apresenta a notícia usada para o teste que foi impressa na tela utilizando a opção 1 do menu.

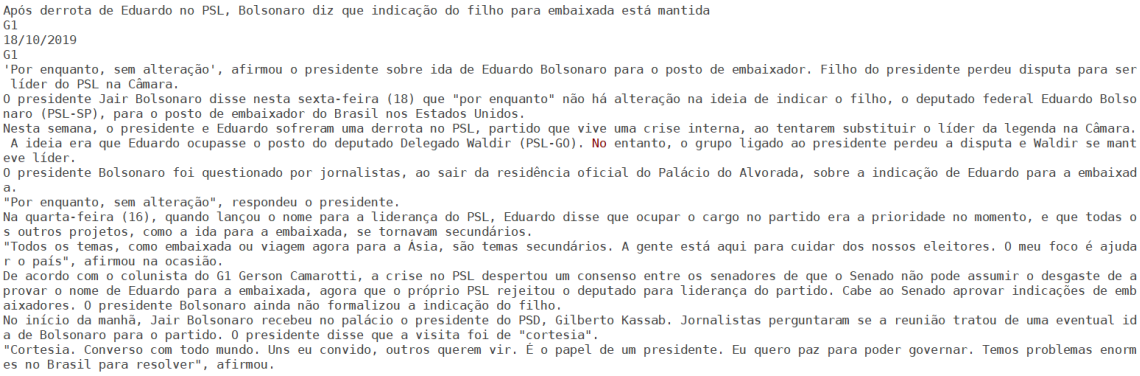


Figura 6: Notícia usada para o teste.

Os resultados obtidos com a notícia da Figura 6 são mostrados na Figura 7.

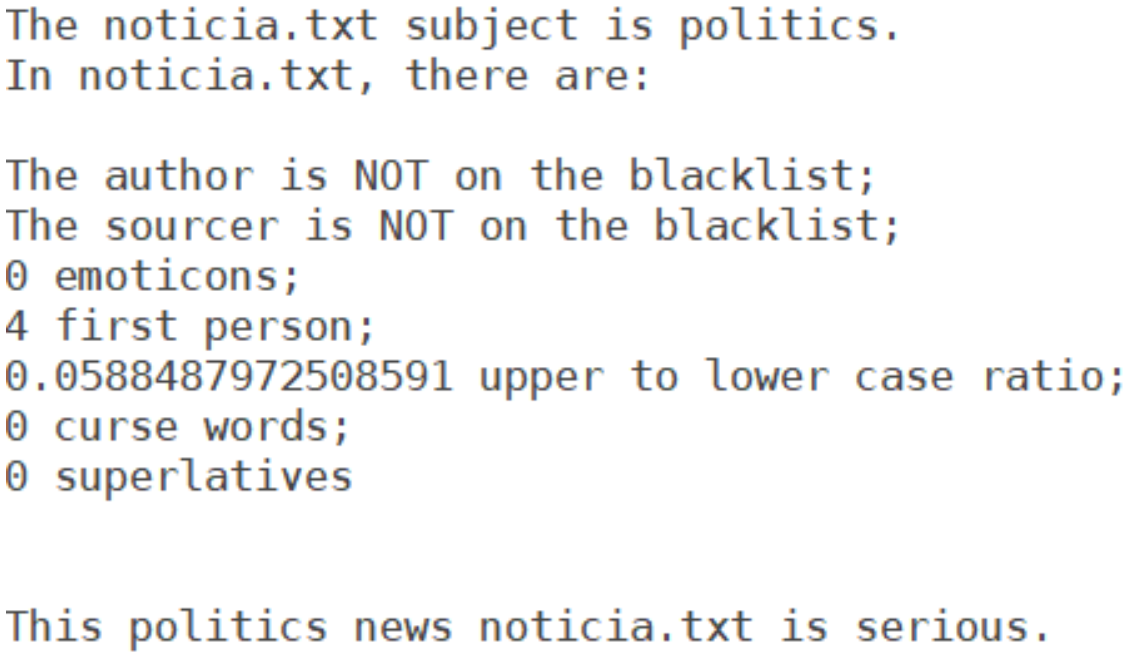
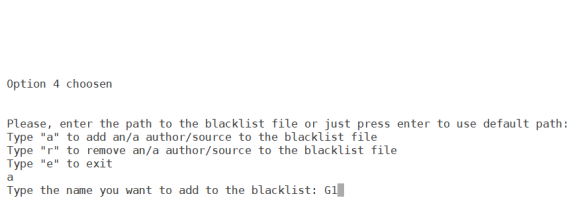
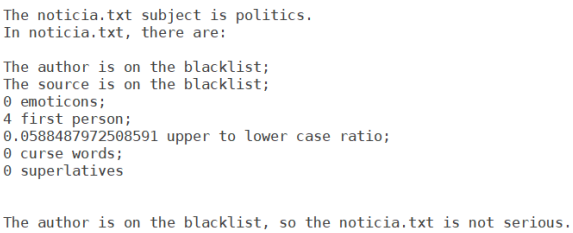


Figura 7: Resultado da notícia classificada.

Utilizando a opção 4 para modificar a lista com as fontes menos confiáveis, o programa fornece ao usuário as opções de remover ou adicionar uma fonte à lista, além da opção de sair do programa. Selecionada a opção de adicionar ou remover um nome da lista, o programa pede ao usuário o nome a ser removido ou adicionado. Adicionando a fonte da notícia da Figura 6 e executando o programa novamente, a saída muda. As Figuras 8a e 8b apresentam as opções quando a opção 4 é selecionada e a saída do prgrama quando ele é executado novamente com as novas mudanças, respectivamente.

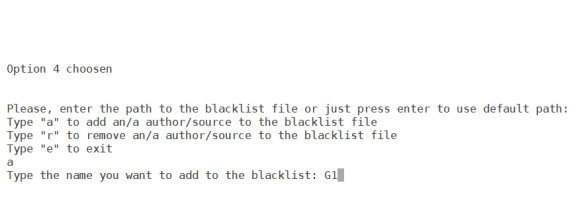


(a) Programa pedindo que o usuário entre com o nome a ser acrescentado à lista.

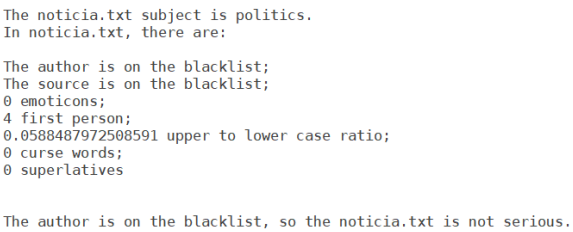


(b) Saída do programa mostrando a notícia como não séria, uma vez que a fonte foi acrescentada à lista de menos confiáveis.

As Figuras 9a e 9b apresentam a saída da opção 5 antes e depois da inclusão do autor à lista de menos confiáveis.



(a) Estatísticas de classificação mostram uma notícia classificada como séria antes da inclusão do autor à lista de menos confiáveis.



(b) Estatísticas de classificação mostram uma notícia classificada como não séria após a inclusão do autor à lista de menos confiáveis.

Utilizando a opção 2, o programa exibe as notícias classificadas como menos sérias na tela pelo título. A Figura 10 mostra a saída do programa após a inclusão do autor da notícia à lista de fontes menos confiáveis.

```
Option 2 choosen


------------------------------------------------------------------------
|                 NOT SO SERIOUS NEWS BY TITLE                         |
------------------------------------------------------------------------
"Após derrota de Eduardo no PSL, Bolsonaro diz que indicação do filho para embaixada está mantida"

------------------------------------------------------------------------
```

Figura 10: Notícias classificadas como menos sérias por títulos.

4 Conclusão

A proposta do trabalho é implementar um classificador de seriedade de notícias baseado em análise de texto. Este trabalho apresenta sua parte 3 que consiste na implementação da interface C++ com o classificador de notícia escrito em Perl. A classificação final da notícia é feita através de um conjunto de métricas calculadas pelo programa em Perl. A eficácia da interface foi demostrada e obteve sucesso na comunicação com o módulo Perl através do uso de um estrutura de dados do tipo pilha. Além da classificação correta do assunto e da seriedade da notícia previamente comprovado na parte 2 deste trabalho e da eficácia da interface do C++ com Perl, uma interface amigável e simples foi desenvolvida, assim facitando o uso e entendimento do sistema

A Implementação da Contagem de Palavras no Spark

```
#Execute code with "spark-submit --master spark://master:7077 <path to this python code>"

#Loading libs and Spark configurations
from pyspark import SparkContext, SparkConf
conf = SparkConf().setAppName("word_count")
sc = SparkContext(conf=conf)

#Input file to spark read
text_file = sc.textFile("hdfs://master:9000/user/app/politics_news.txt")
counts = text_file.flatMap(lambda line: line.split(" ")) \
             .map(lambda word: (word, 1)) \
             .reduceByKey(lambda a, b: a + b, 1) \
             .map(lambda (a, b): (b, a)) \
             .sortByKey(0, 1) \
             .map(lambda (a, b): (b, a))


#Output directory to save results
counts.saveAsTextFile("hdfs://master:9000/user/app/politics_count.txt")
```

Lista 3: Implementação da função de contagem de palavras no Spark.