Métodos Probabilísticos para Engenharia Informática – Projeto
Criação de uma Aplicação de Bloom Filter, Minhash e LSH – Spell Checker

Filipe Pires 85122
João Alegria 85048

Introdução

Este relatório abarca não só a descrição da tarefa a nós proposta e da solução por nós implementada, como também uma breve listagem dos maiores problemas de implementação que encontramos e uma conclusão sobre o trabalho no geral.

O primeiro passo para a produção de uma implementação funcional e eficaz foi a construção dos módulos necessários: Bloom Filter, Minhash e LSH. Estes componentes foram implementados em Java e estão organizados por duas classes: BloomFilter e Palavra. Esta última classe incorpora a função *hash* Murmur, a função LSH e a função Minhash. Assim que concluídos, prosseguimos para testes individuais de cada módulo bem sucedidos.

Diariamente todos nós utilizamos aplicações de escrita de texto que permitem ao utilizador corrigir facilmente erros ortográficos. Tendo isto em conta, numa segunda fase do projeto, o nosso objetivo foi desenvolver uma aplicação SpellChecker em Java que funcionasse como corretor ortográfico, utilizando o Bloom Filter para verificar se as palavras existiam e aproveitando as capacidades do Minhash e do LSH para encontrar as alternativas de correção dos erros.

O resultado final tem uma construção muito lógica, sendo que a sua utilização é bastante intuitiva.

Implementação

Testes Unitários:

Os ficheiros *teste_bf* e *teste_min* têm como objetivo testar individualmente cada módulo do projeto. São apenas exemplos simples de utilizações dos módulos de forma a demonstrar o funcionamento destes.

Aplicação:

O SpellChecker está dividido em pequenas tarefas que interagem com o utilizador de forma a apresentar os resultados por ele pretendidos.

Assim que o programa se inicia, o utilizador define a língua em que vai escrever (Português ou Inglês) e decide se ativa ou não a correção automática (*autocorrect*). De seguida o programa entra num ciclo que termina quando o utilizador assim quiser. É lhe apresentada a opção de escrita e, após o fazer, o corretor verifica cada palavra à procura das que não pertençam ao dicionário.

Para cada palavra que não pertença ao dicionário, é criado um objeto Palavra que se responsabilizará por encontrar as palavras mais semelhantes à escrita pelo utilizador. Após se encontrar as palavras semelhantes, substitui-se a palavra inicial pela melhor candidata (caso a correção automática esteja ativada) ou permite-se ao utilizador escolher uma entre as melhores candidatas a substituir a palavra inicial (caso esteja desativada).

Quando finalmente todas as palavras da frase escrita pelo utilizador estão corretas, é impresso o resultado final no terminal e retorna-se ao início do ciclo.

Módulo 1:

A classe BloomFilter tem como objetivo, como o nome indica, estruturar um Bloom Filter que permita guardar num *array* de bytes a informação da existência ou não de elementos numa dada estrutura de dados. Tem como atributos o *array* de bytes, o respetivo tamanho e o número de funções *hash* a serem utilizadas. Tem como métodos o seu construtor, os *getters* dos atributos, um conversos para *String*, a função *hash* Murmur(), o insertMember() responsável pela inserção de um membro no *array* e o isMember() responsável pela verificação da existência de um dado elemento no *array*.

Módulo 2:

A classe Palavra tem como objetivo estruturar o módulo que permite a comparação entre dois elementos e a procura de semelhanças entre entidades (no caso, palavras). Tem como atributos o número de caracteres de cada *shingle* e o *Set* que armazenará as palavras em *hashes*. Tem como métodos o seu construtor, o *getter* do Set, a função *hash* Murmur(), o compare() que faz a comparação entre entidades em concreto, com o auxílio o lsh() que permite selecionar apenas os possíveis candidatos de interesse para esta comparação. Tem ainda os métodos auxiliares primo(), indiceHash() e signature().

Problemas / Dificuldades

Os desafios mais complicados que enfrentamos no desenvolvimento do projeto terão sido a implementação correta do LSH na classe Palavra e a procura de um tema para a aplicação que englobasse os módulos todos.

Para este primeiro problema a solução tornou-se clara quando compreendemos a 100% a ideia do LSH. Apercebemo-nos também de que a incorporação desta função na aplicação desenvolvida trouxe uma rapidez de processamento muito importante.

Quanto ao tema da aplicação, foi solucionado após alguma pesquisa sobre utilizações práticas de funções Minhash e Bloom Filters. Esta pesquisa levou-nos a pensar em funcionalidades dos nossos computadores e telemóveis que utilizamos no nosso dia-a-dia e que pudessem de alguma forma ser implementadas através dos módulos por nós criados. Assim, a ideia do corretor ortográfico surgiu naturalmente e a decisão de a por em prática foi unânime.

Conclusão

As conclusões que podemos retirar relativamente a este projeto são direcionadas ao nosso funcionamento como par de trabalho e ao que correu bem no desenvolvimento de uma solução para o problema.

Quanto ao trabalho dos dois elementos, concluímos que mantivemos uma comunicação constante de forma a estarmos sempre a par do ponto de situação do trabalho e que funcionamos bem neste tipo de tarefas.

Quanto à solução por nós implementada, podemos concluir que é 100% eficaz segundo os nossos testes e que tem uma velocidade de execução relativamente rápida dependendo do tamanho do dicionário escolhido.

Numa visão mais geral acerca do trabalho, terminamos dizendo que foi interessante e cativante trabalhar em Java e que o resultado final foi satisfatório.

Fontes de Pesquisa:

https://stackoverflow.com/
https://docs.oracle.com/javase/8/docs/api/allclasses-noframe.html
https://www.tutorialspoint.com/java/index.htm
https://www.tutorialspoint.com/matlab/index.htm
https://elearning.ua.pt/course/view.php?id=3440