Projeto: Search Engine

Cristiana Nogueira, Hanna Rodrigues e Marcos Antonio

18 de junho de 2020

Pré-processamento do Corpus

- Textos para a árvore e textos para o usuário (136 + "títulos.txt")
- Aproximadamente 250M palavras.
- Remoção de caracteres especiais para inserir na árvore
- Ordenação em ordem alfabética de títulos para indexar em tempo constante
- Remoção de palavras repetidas em um mesmo texto otimizando espaço e tempo de construção da árvore

pré-proc

```
def classifica():
    global lista
    t0 = time.time()
    c2 = 0

    for i in range(len(n)): ## chamo todos os textos para serem lidos
        titulos('../../../Desktop/MARCAO/araw.en/'+n[i]) ## adiciono na lista os Ids na ordem que são lidas
        print(i)
    for i in range(len(lista)):
        lista[i].append(i) ## adiciono à cada elemento o número correspondente à ordem em que ele foi lido

    lista.sort()   ## ordeno a lista com base nos títulos
    print(time.time()-t0)
    for i in range(len(lista)):
        lista[i].append(i)     ##  adiciono agora outro componente para cada elemento da lista, correspondente à sua
                               ## posição na ordem alfabética, ou seja; o ID

    lista.sort(key = lambda x: x[1]) ## retorno a lista para sua configuração original
    for i in range(len(n)):
        c2 = escreve('../../../Desktop/MARCAO/araw.en/'+n[i] ,c2)  ## agora já temos para cada texto seu ID e
                                                                   ## podemos escreve-los em ordem
        print(i)
    print(time.time()-t0)
    return lista


## ''''''''''''''
```

Figure: Código py

Estrutura de dados : Trie

- alfabeto = [0,1,2,...,9,a,b,c,d,...,z];
- cada nó é um array de ponteiros, elementos do alfabeto;
- id = index de cada texto do Corpus em ordem alfabética de títulos;
- cada nó possui um vetor de inteiros, ids;
- Se o caminho feito até certo nó representa uma palavra, no vetor desse nó se encontram os ids em ordem crescente dos textos onde essa palavra aparece;

Busca

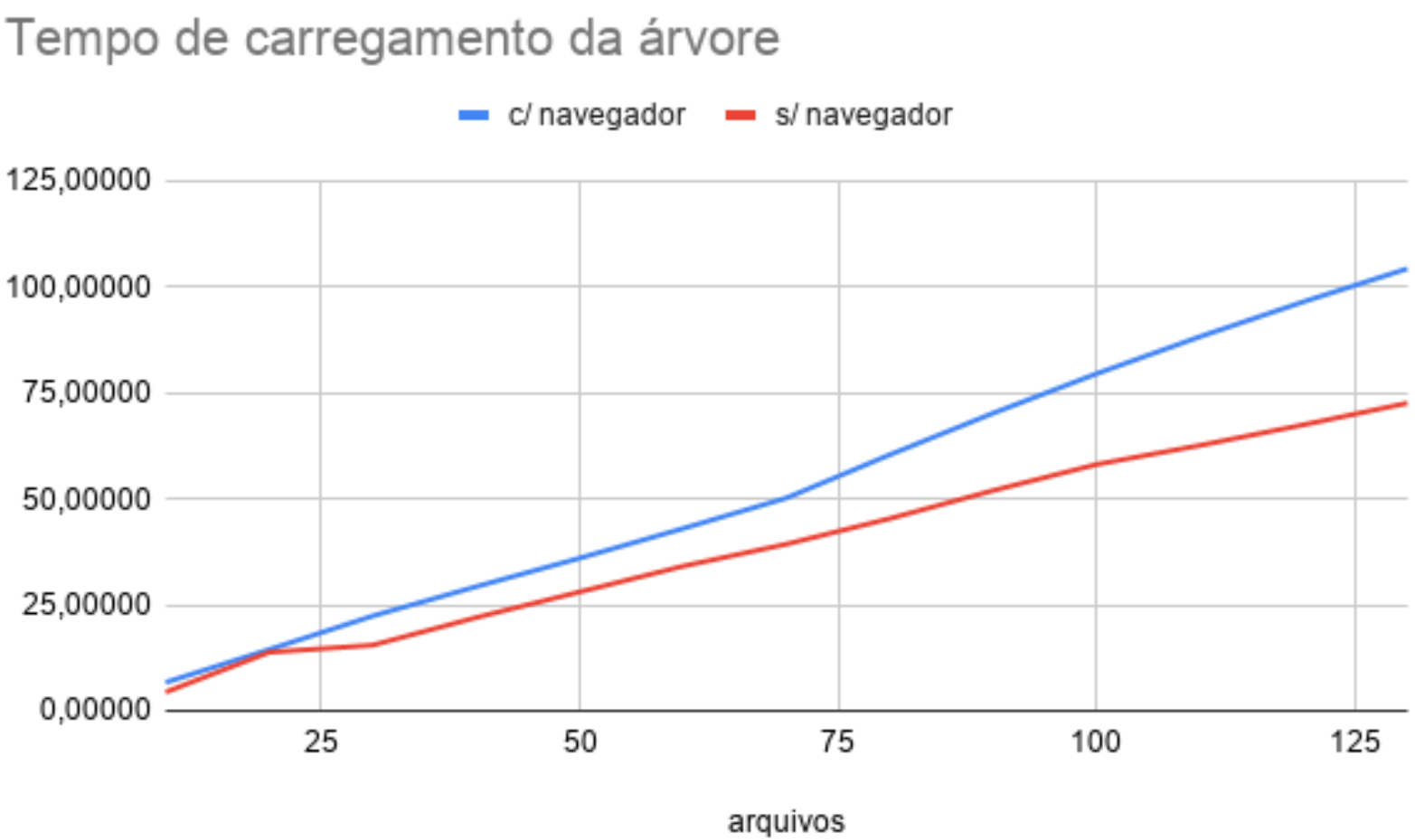
- Limpa entrada do usuário para o alfabeto utilizado;
- Sugestão de palavras,caso a busca não esteja na árvore;
- para frases retorna-se a intersecção das buscas de cada palavra;
- Busca a 1º ocorrência da palavra no texto;
- Resultado da busca ordenado por ordem alfabética de títulos;
- Busca realizada no console ou localmente através de interface web;

Sugestão de palavras

- Realiza edições pontuais na palavra para produzir variações e testar se estão na árvore;
- Edições : deletar, transpor, permutar, inserir e susbtituir;
- Palavra única : efetua-se até duas vezes a produção de variações, retorna-se 5 sugestões;
- Caso frases: pelo menos uma palavra não está na árvore ou a intersecção é nula entre elas;
- Frase : realizam-se edições nas palavras , retorna-se 1 sugestão;

Interface web

- O código do servidor foi disponibilizado pelo monitor.
- Foi implementada interação entre o Java Script, C++ e Html.
- A função query no JavaScript recebe o texto submetido no quadrado de busca e chama a função de limpeza do input e pesquisa na árvore. Feito isso retorna uma lista com os primeiros 20 títulos.
- Cada título está associado a um hiperlink, o qual quando clicado abre o texto correspondente ao título.
- Há também um hiperlink com a palavra next que chama a função que retorna os títulos para os próximos 20 títulos da lista e assim sucessivamente.



Intel Core i5-10210U CPU @ 1.60GHz x 8

entrada	tempo de busca	resultados
future	0,001082	57043
health patrol	0,003129	450
scrape gutter affect	0,000235	1
crutch endure arrest mobile	0,000731	0
clique finger marine hunter arrest	0,000893	0

Palavras geradas em https://randomwordgenerator.com/

Intel Core i5-10210U CPU @ 1.60GHz x 8

Limitações

- Não diferencia caracteres maiúsculos de minúsculos e alguns caracteres são ignorados;
- Serialização e desserialização inacabada;
- Apenas uma sugestão para frases;
- Interface web apenas localmente e em formato básico;
- Pesquisas com caracteres especiais são simplificadas
- Estrutura da frase não é levada em conta

Trabalhos futuros

- Finalizar serialização
- Resultados ordenados por peso, ou seja; número de ocorrências
- melhorar o design da interface, apresentar gráficos etc..
- considerar caracteres especiais e a estrutura da frase.

Referências

- https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-851-advanced-data-structures-spring-2012/lecture-videos/session-16-strings/
- https://www.ime.usp.br/ pf/estruturas-de-dados/aulas/tries.html
- https://en.wikipedia.org/wiki/Serialization
- `https://en.wikipedia.org/wiki/Radix_tree`
- http://people.cs.ksu.edu/ rhowell/DataStructures/trees/tries/intro.html
- https://randomwordgenerator.com/

Obrigada!
Para dúvidas, sugestões:

Cristiana: cristiana.couto@fgv.edu.br

Hanna: hanna.rodrigues.ismart@gmail.com

Marcos: marcosantonioalves358@gmail.com

Repositório do projeto:
`https://github.com/Cristiananc/Search_Engine`