Limpeza_e_transformacao

May 29, 2024

1 Busca de Índice e Consulta de documento

Autor: Davi J. Leite Santos
Versão: 0.0.3
Data: 25 de Abril de 2024
Localização: Ribeirão das Neves, Minas Gerais - Brasil

1.1 Contato

- **Endereço:** Ribeirão das Neves, Minas Gerais - Brasil
- **Email:** davi.jls@outlook.com
- **LinkedIn:** davi-j-leite-santos
- **Website:** davijls.com.br

1.2 Principais Competências

- **Cibersegurança**
- **Segurança da Informação**
- **Operações de TI**

[ ]:
```
import json
import re
```

2 Limpeza dos dados

Nessa etapa serão excluidas as stopworlds e transformadas os dados em UTF8 e retiradas os caracteres especias

[ ]:
```
# Lista expandida de stopwords em português
stopwords = set([
    "a", "à", "adeus", "agora", "ainda", "além", "algmas", "alguns", "ali",␣
 ↪"ambas", "ambos",
    "ante", "antes", "ao", "aos", "apenas", "apoio", "após", "aquela",␣
 ↪"aquelas", "aquele",
    "aqueles", "aqui", "aquilo", "as", "até", "através", "cada", "cá",␣
 ↪"catorze", "cedo",
    "cento", "certamente", "certeza", "cima", "cinqüenta", "cinco", "com",␣
 ↪"como", "conselho",
```

```
  "contra", "contudo", "da", "daquela", "daquelas", "daquele", "daqueles",␣
↪"dar", "das",
  "de", "dela", "delas", "dele", "deles", "demais", "dentro", "depois",␣
↪"desde", "dessa",
  "dessas", "desse", "desses", "desta", "destas", "deste", "destes", "deve",␣
↪"devem", "deverá",
  "dez", "dezanove", "dezasseis", "dezassete", "dezoito", "dia", "diante",␣
↪"disse", "disso",
  "disto", "dito", "diz", "dizem", "do", "dois", "dos", "doze", "duas",␣
↪"dúvida", "e", "é",
  "ela", "elas", "ele", "eles", "em", "embora", "enquanto", "então", "entre",␣
↪"era", "eram",
  "éramos", "és", "essa", "essas", "esse", "esses", "esta", "está",␣
↪"estamos", "estão",
  "estar", "estas", "estás", "estava", "estavam", "estávamos", "este",␣
↪"estes", "esteve",
  "estive", "estivemos", "estiveram", "estiveste", "estivestes", "estou",␣
↪"eu", "exemplo",
  "faço", "fará", "favor", "faz", "fazeis", "fazem", "fazemos", "fazer",␣
↪"fazes", "feita",
  "feitas", "feito", "feitos", "fez", "fim", "final", "foi", "fomos", "for",␣
↪"fora", "foram",
  "forma", "foste", "fostes", "fui", "geral", "grande", "grandes", "grupo",␣
↪"há", "haja",
  "hajam", "hão", "havemos", "havia", "hei", "hoje", "hora", "horas",␣
↪"houve", "houvemos",
  "houveram", "houverei", "houveremos", "houveria", "houveriam", "houvermos",␣
↪"houvesse",
  "houvessem", "isso", "isto", "já", "la", "lá", "lhe", "lhes", "lo", "logo",␣
↪"longe",
  "lugar", "maior", "maioria", "mais", "mal", "mas", "máximo", "me",␣
↪"melhor", "mesma",
  "mesmas", "mesmo", "mesmos", "meu", "meus", "minha", "minhas", "momento",␣
↪"muita", "muitas",
  "muito", "muitos", "na", "nada", "não", "naquela", "naquelas", "naquele",␣
↪"naqueles", "nas",
  "nem", "nenhum", "nessa", "nessas", "nesse", "nesses", "nesta", "nestas",␣
↪"neste", "nestes",
  "ninguém", "nível", "no", "noite", "nós", "nome", "nos", "nossa", "nossas",␣
↪"nosso",
  "nossos", "nova", "novas", "nove", "novo", "novos", "num", "numa",␣
↪"número", "nunca", "o",
  "obra", "obrigada", "obrigado", "oitava", "oitavo", "oito", "onde",␣
↪"ontem", "onze", "os",
```

```
    "ou", "outra", "outras", "outro", "outros", "para", "parece", "parte",␣
 ↪"partir", "paucas",
    "pela", "pelas", "pelo", "pelos", "perante", "perto", "pode", "pude",␣
 ↪"podem", "poder",
    "poderá", "podia", "pois", "põe", "põem", "ponto", "pontos", "por",␣
 ↪"porque", "porquê",
    "pouca", "poucas", "pouco", "poucos", "primeira", "primeiras", "primeiro",␣
 ↪"primeiros",
    "promeiro", "própria", "próprias", "próprio", "próprios", "próxima",␣
 ↪"próximas", "próximo",
    "próximos", "pude", "pôde", "quais", "quáis", "qual", "qualquer", "quando",␣
 ↪"quanto",
    "quarta", "quarto", "quatro", "que", "quê", "quem", "quer", "quereis",␣
 ↪"querem", "queremas",
    "queres", "quero", "questão", "quieto", "quinze", "quén", "quén",␣
 ↪"relação", "sabe", "sabem",
    "são", "se", "segunda", "segundo", "sei", "seis", "seja", "sejam", "sem",␣
 ↪"sempre", "sendo",
    "ser", "será", "serão", "seria", "seriam", "sete", "sétima", "sétimo",␣
 ↪"seu", "seus", "seus",
    "só", "sob", "sobre", "sois", "somente", "somos", "sou", "sua", "suas",␣
 ↪"tal", "talvez",
    "também", "tampouco", "tanta", "tantas", "tanto", "tão", "tarde", "te",␣
 ↪"tem", "têm", "temos",
    "tendes", "tendo", "tenha", "tenham", "tenho", "tens", "tentar",␣
 ↪"tentaram", "tente", "tentei",
    "ter", "terá", "terão", "terei", "teremos", "teria", "teriam", "termos",␣
 ↪"teu", "teus",
    "teve", "tive", "tivemos", "tiveram", "tivera", "tiveram", "tiveste",␣
 ↪"tivestes", "toda",
    "todas", "todavia", "todo", "todos", "trabalho", "três", "treze", "tu",␣
 ↪"tua", "tuas",
    "tudo", "última", "últimas", "último", "últimos", "um", "uma", "umas",␣
 ↪"uns", "usa", "usar",
    "valor", "veja", "vem", "vens", "ver", "vez", "vezes", "vindo", "vinte",␣
 ↪"você", "vocês",
    "vos", "vossa", "vossas", "vosso", "vossos", "zero"
])
```

[ ]:
```
# Função para limpar texto, removendo caracteres especiais, stopwords e␣
 ↪convertendo para UTF-8
def clean_text(text):
    # Remover caracteres especiais e substituir por espaços
    text = re.sub(r'[^\w\s]', ' ', text)
    # Converter para minúsculas
    text = text.lower()
```

```
    # Remover stopwords
    text = ' '.join(word for word in text.split() if word not in stopwords)
    # Garantir que o texto esteja em UTF-8
    return text.encode('utf-8').decode('utf-8')
```

[ ]:
```
# Função para limpar vocabulário e índice
def clean_vocab_and_index(vocab, index):
    cleaned_vocab = {}
    cleaned_index = {}
    for word, word_id in vocab.items():
        cleaned_word = clean_text(word)
        if cleaned_word not in cleaned_vocab:
            cleaned_vocab[cleaned_word] = word_id
        cleaned_word_id = cleaned_vocab[cleaned_word]

        if cleaned_word_id not in cleaned_index:
            cleaned_index[cleaned_word_id] = index[str(word_id)]
        else:
            for doc_id, positions in index[str(word_id)].items():
                if doc_id not in cleaned_index[cleaned_word_id]:
                    cleaned_index[cleaned_word_id][doc_id] = positions
                else:
                    cleaned_index[cleaned_word_id][doc_id].extend(positions)

    return cleaned_vocab, cleaned_index
```

[ ]:
```
# Função para limpar o formato de tupla e lista
def clean_tuple_and_list_formats(tuple_format, list_format):
    cleaned_tuple_format = [(clean_text(str(word_id)), doc_id, positions) for␣
 ↪word_id, doc_id, positions in tuple_format]
    cleaned_list_format = [{'word_id': clean_text(str(item['word_id'])),␣
 ↪'doc_id': item['doc_id'], 'positions': item['positions']} for item in␣
 ↪list_format]

    return cleaned_tuple_format, cleaned_list_format
```

[ ]:
```
# Função principal para carregar, limpar e salvar os dados
def clean_data(index_geral_file, vocab_geral_file, tuple_format_file,␣
 ↪list_geral_file):
    # Carregar os arquivos
    with open(index_geral_file, 'r', encoding='utf-8') as f:
        index_geral = json.load(f)
    with open(vocab_geral_file, 'r', encoding='utf-8') as f:
        vocab_geral = json.load(f)
    with open(tuple_format_file, 'r', encoding='utf-8') as f:
        tuple_format = json.load(f)
    with open(list_geral_file, 'r', encoding='utf-8') as f:
```

```
        list_geral = json.load(f)

    # Limpar vocabulário e índice
    cleaned_vocab, cleaned_index = clean_vocab_and_index(vocab_geral,␣
 ↪index_geral)

    # Limpar formatos de tupla e lista
    cleaned_tuple_format, cleaned_list_format =␣
 ↪clean_tuple_and_list_formats(tuple_format, list_geral)

    # Salvar os dados limpos
    with open(index_geral_file, 'w', encoding='utf-8') as f:
        json.dump(cleaned_index, f, ensure_ascii=False, indent=4)
    with open(vocab_geral_file, 'w', encoding='utf-8') as f:
        json.dump(cleaned_vocab, f, ensure_ascii=False, indent=4)
    with open(tuple_format_file, 'w', encoding='utf-8') as f:
        json.dump(cleaned_tuple_format, f, ensure_ascii=False, indent=4)
    with open(list_geral_file, 'w', encoding='utf-8') as f:
        json.dump(cleaned_list_format, f, ensure_ascii=False, indent=4)
```

[ ]:
```
# Defina os caminhos dos arquivos gerais
index_geral_file = 'index_geral.json'
vocab_geral_file = 'vocab_geral.json'
tuple_format_file = 'tuple_format.json'
list_geral_file = 'list_geral.json'
```

[ ]:
```
# Limpar os dados
clean_data(index_geral_file, vocab_geral_file, tuple_format_file,␣
 ↪list_geral_file)
```

3 Análise Léxica e stemming

Instalando a biblioteca para fazer a analise Léxica e stemming

[ ]:
```
# pip install nltk
```

[ ]:
```
import json
import re
import os
import time
import nltk
from nltk.stem import RSLPStemmer
```

[ ]:
```
# Baixar os recursos necessários do NLTK
nltk.download('rslp')
nltk.download('punkt')
```

```
# Inicializar o Stemmer para português
stemmer = RSLPStemmer()
```

```
[nltk_data] Downloading package rslp to
[nltk_data]     C:\Users\davim\AppData\Roaming\nltk_data…
[nltk_data]   Unzipping stemmers\rslp.zip.
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\davim\AppData\Roaming\nltk_data…
[nltk_data]   Package punkt is already up-to-date!
```

Criando as funções para a utilização das analises

[ ]:
```
# Função para análise léxica e stemming
def lexical_analysis_and_stemming(text):
    # Tokenização do texto em palavras
    tokens = nltk.word_tokenize(text, language='portuguese')
    # Aplicar stemming em cada token
    stemmed_tokens = [stemmer.stem(token) for token in tokens]
    return stemmed_tokens
```

[ ]:
```
# Função para aplicar análise léxica e stemming em vocabulário e índice
def apply_lexical_analysis_and_stemming(vocab, index):
    stemmed_vocab = {}
    stemmed_index = {}

    for word, word_id in vocab.items():
        # Análise léxica e stemming
        stemmed_word = ' '.join(lexical_analysis_and_stemming(word))
        if stemmed_word not in stemmed_vocab:
            stemmed_vocab[stemmed_word] = word_id
        stemmed_word_id = stemmed_vocab[stemmed_word]

        if str(word_id) in index:
            if stemmed_word_id not in stemmed_index:
                stemmed_index[stemmed_word_id] = index[str(word_id)]
            else:
                for doc_id, positions in index[str(word_id)].items():
                    if doc_id not in stemmed_index[stemmed_word_id]:
                        stemmed_index[stemmed_word_id][doc_id] = positions
                    else:
                        stemmed_index[stemmed_word_id][doc_id].extend(positions)

    return stemmed_vocab, stemmed_index
```

[ ]:
```
# Função para aplicar análise léxica e stemming no formato de tupla e lista
def apply_lexical_analysis_and_stemming_to_formats(tuple_format, list_format):
    stemmed_tuple_format = [(lexical_analysis_and_stemming(str(word_id))[0],␣
 ↪doc_id, positions) for word_id, doc_id, positions in tuple_format]
```

```
    stemmed_list_format = [{'word_id':␣
 ↪lexical_analysis_and_stemming(str(item['word_id']))[0], 'doc_id':␣
 ↪item['doc_id'], 'positions': item['positions']} for item in list_format]

    return stemmed_tuple_format, stemmed_list_format
```

[ ]:
```
# Função principal para carregar, processar e salvar os dados com análise␣
 ↪léxica e stemming
def process_data_with_lexical_analysis_and_stemming(index_geral_file,␣
 ↪vocab_geral_file, tuple_format_file, list_geral_file):
    # Carregar os arquivos
    with open(index_geral_file, 'r', encoding='utf-8') as f:
        index_geral = json.load(f)
    with open(vocab_geral_file, 'r', encoding='utf-8') as f:
        vocab_geral = json.load(f)
    with open(tuple_format_file, 'r', encoding='utf-8') as f:
        tuple_format = json.load(f)
    with open(list_geral_file, 'r', encoding='utf-8') as f:
        list_geral = json.load(f)

    # Aplicar análise léxica e stemming em vocabulário e índice
    stemmed_vocab, stemmed_index =␣
 ↪apply_lexical_analysis_and_stemming(vocab_geral, index_geral)

    # Aplicar análise léxica e stemming nos formatos de tupla e lista
    stemmed_tuple_format, stemmed_list_format =␣
 ↪apply_lexical_analysis_and_stemming_to_formats(tuple_format, list_geral)

    # Salvar os dados processados
    with open(index_geral_file, 'w', encoding='utf-8') as f:
        json.dump(stemmed_index, f, ensure_ascii=False, indent=4)
    with open(vocab_geral_file, 'w', encoding='utf-8') as f:
        json.dump(stemmed_vocab, f, ensure_ascii=False, indent=4)
    with open(tuple_format_file, 'w', encoding='utf-8') as f:
        json.dump(stemmed_tuple_format, f, ensure_ascii=False, indent=4)
    with open(list_geral_file, 'w', encoding='utf-8') as f:
        json.dump(stemmed_list_format, f, ensure_ascii=False, indent=4)
```

[ ]:
```
# Defina os caminhos dos arquivos gerais
index_geral_file = 'index_geral.json'
vocab_geral_file = 'vocab_geral.json'
tuple_format_file = 'tuple_format.json'
list_geral_file = 'list_geral.json'
```

[ ]:
```
# Processar os dados com análise léxica e stemming
process_data_with_lexical_analysis_and_stemming(index_geral_file,␣
 ↪vocab_geral_file, tuple_format_file, list_geral_file)
```

4 Etapa para fornece uma medição básica de desempenho e espaço

[ ]:
```python
# Função para carregar o índice e vocabulário
def load_data(index_file, vocab_file):
    with open(index_file, 'r', encoding='utf-8') as f:
        index = json.load(f)
    with open(vocab_file, 'r', encoding='utf-8') as f:
        vocab = json.load(f)
    return index, vocab
```

[ ]:
```python
# Função para buscar termos no índice
def search_term(index, vocab, term):
    stemmed_term = stemmer.stem(term)
    if stemmed_term in vocab:
        word_id = vocab[stemmed_term]
        if str(word_id) in index:
            return index[str(word_id)]
    return {}
```

[ ]:
```python
# Função para medir o tempo de pesquisa e o espaço de indexação
def measure_performance(index_file, vocab_file, search_terms, output_file):
    index, vocab = load_data(index_file, vocab_file)

    # Medir tempo de pesquisa
    search_times = []
    for term in search_terms:
        start_time = time.time()
        search_term(index, vocab, term)
        end_time = time.time()
        search_times.append(end_time - start_time)

    avg_search_time = sum(search_times) / len(search_times)

    # Medir tamanho dos arquivos
    index_size = os.path.getsize(index_file)
    vocab_size = os.path.getsize(vocab_file)
    total_size = index_size + vocab_size

    # Escrever resultados em um arquivo de texto
    with open(output_file, 'w', encoding='utf-8') as f:
        f.write(f"Tempo médio de pesquisa: {avg_search_time:.6f} segundos\n")
        f.write(f"Tamanho do arquivo de índice: {index_size} bytes\n")
        f.write(f"Tamanho do arquivo de vocabulário: {vocab_size} bytes\n")
        f.write(f"Tamanho total dos arquivos: {total_size} bytes\n")
```

[ ]:
```python
# Definir os caminhos dos arquivos
index_geral_file = 'index_geral.json'
```

```
vocab_geral_file = 'vocab_geral.json'
output_file = 'performance_metrics.txt'

# Definir termos de pesquisa para teste
search_terms = ['educação', 'linguagem', 'noticia', 'ideia', 'politica']
```

[ ]:
```
# Medir desempenho e salvar resultados
measure_performance(index_geral_file, vocab_geral_file, search_terms,␣
↪output_file)
```

[ ]:
```

```