Tópicos Especiais em Recuperação da Informação
Alunos: Matheus Balonecker, Maria Luísa Braga e Rodrigo Leão

1.Introdução

O objetivo deste trabalho é implementar uma máquina de busca e testá-la nos documentos da coleção CFC (Cystic Fibrosis Collection), devemos também avaliar os resultados retornados pela nossa máquina de busca em comparação com os resultados ideais sugeridos para a coleção em questão.

2. Implementação

A primeira coisa que fizemos foi a criação de uma lista invertida com as palavras da nossa coleção. Criamos um hash (hashWords), cujas chaves são as palavras da coleção e seu conteúdo é uma lista de tuplas, onde cada tupla tem o número do documento no qual a palavra aparece e sua frequência neste documento.

Para testar as consultas sugeridas pela coleção, fizemos um processo semelhante, mas o hashQueries guarda como chave a consulta e seu conteúdo é uma lista com os resultados ideais segundo a coleção.

As principais funções utilizadas no trabalho foram:

- TermoaTermo(consulta): Essa função recebe a consulta como parâmetro, a qual é uma lista de strings e retorna uma lista de tuplas, nas quais o primeiro elemento é a similaridade do documento retornado com a consulta, e o segundo elemento é o documento que foi julgado como relevante para a consulta feita.
- filterHash(consulta): Tem como parâmetro uma lista com strings que representam a consulta e retorna um hash com derivado do hashWords, que é composto apenas pelas palavras que existem na consulta.
- norma(doc): Calcula a norma.
- weight(doc,word): Calcula o peso da palavra no documento.
- idf(word): Retorna o IDF de uma palavra pesente no hashIdf, que é um hash com os valores dos IDFs das palavras presentes no hashWords.
- td(doc,word): Calcula o TF da palavra no documento
- precisao(resultadoIdeal, meuResultado): Retorna uma lista com as precisões consecutivas levando em consideração o resultado ideal proposto na coleção e o resultado obtido pela função TermoaTermo.
- MAPi(resultadoIdeal, meuResultado): Retorna a média das precisões para uma determinada consulta, levando em consideração o resultado ideal proposto na coleção e o resultado obtido pela função TermoaTermo.
- calculaNDCG(meu,ideal): Calcula e retorna o NDCG para cada consulta (para essa função, utilizamos como relevância de um documento a moda das relevâncias desse documento).

Desconsideramos as stopwords para criar o hashWords e quando fazemos a consulta. As palavras só podem ser adicionadas nesse hash se fizerem parte de uma das categorias de interesse (Author(s), Title, Major Subject, Minor Subject, Abstract ou Extract, que são definidas na coleção).

3. Execução

Como nosso objetivo era avaliar os resultados da nossa máquina de busca, fizemos com que o nosso programa executasse a busca apenas para as consultas avaliadas que estão no arquivo cfquery.

Execução no terminal: python3 TrabRIMelhoria.py

4. Resultados

Resultados sem aplicação de melhoria:

MAP: 0.26652266012222536
NDCG: 0.646592350739715

Na tentativa de executar uma melhoria, optamos por remover as stopwords (palavra que não dão significado aos documentos) do índice e observamos que não há melhora quanto à precisão dos resultados, no entanto houve um ganho de aproximadamente 50% no desempenho.

Tempo de execução com stopwords indexadas: 6.592s
Tempo de execução sem stopwords indexadas: 3.468s

Ainda pensando em melhorar os resultados, utilizamos a técnica de stemming (com auxílio da biblioteca nltk do python), essa técnica consiste em não utilizar as palavras como índices da lista invertida, mas sim os radicais dessas palavras, para obter um resultado mais abrangente. Usando essa técnica, o MAP aumentou seu valor em aproximadamente 12%, e o NDCG em 1%, obtendo os resultados abaixo:

MAP = 0.2987630145749851
NDCG = 0.6545696783535484

5. Conclusão

Após a análise dos resultados obtidos, concluímos que o modelo vetorial consegue produzir um resultado bastante razoável, porém é possível melhorá-los com diversas técnicas, as que

usamos neste trabalho foram a remoção de stopwords do índice da máquina de busca e stemming. A primeira tentativa resultou em melhoria apenas de desempenho, já a segunda melhorou os resultados obtidos pela máquina de busca.