

Recuperação de Informação - Solr

Guilherme T. Bazzo and Nicolau P. Alff

Instituto de Informática - UFRGS
{gtbazzo, npalff}@inf.ufrgs.br

1 Introdução

Este trabalho tem o objetivo de experienciar os conceitos vistos em aula através da utilização de um sistema de Recuperação de Informações. Serão apresentadas as principais funcionalidades da ferramenta, assim como mostrar um rápido guia para instalação, indexação e coleta de dados.

Estes conceitos serão abordados utilizando a ferramenta Apache Solr¹. Esta é uma plataforma desenvolvida pela empresa Apache, juntamente com o projeto Apache Lucene², é um software de pesquisa empresarial escrito na linguagem Java e foi inicialmente lançado em 2004, estando em sua versão 8.6.1 atualmente.

2 Funcionalidades

Solr é um servidor de pesquisa autônomo que permite a indexação e coleta de documentos do tipo JSON, XML, CSV e binários por requisições HTTP.[5] Empresas como Netflix, Instagram, NASA e IBM utilizam o Solr com propósitos de busca e coleta de documentos.

A ferramenta é otimizada para grande volume de tráfego de requisições, escalável para suportar bilhões de documentos com grandes volumes de consultas, apresenta clusterização dinâmica e indexação em tempo quase real. Outras funcionalidades da ferramenta são:

- Busca avançada de texto completo: recurso como correspondências, incluindo frases, *wildcards*, associações e agrupamentos;
- Componentes de análise de texto: separação de palavras, regex e stemming;
- Interfaces moderna e responsiva: permite fácil controle sobre todas as funcionalidades do sistema, como *cores*, coleções, *logs* e *queries*, assim como janelas para análise, debug e monitoramento dos dados;
- Suporte a uma grande variedade de línguas, como Inglês, Português, Chinês e Alemão;
- Componentes de correção de texto, sugestões (*auto-complete*) e coleta de dados a partir de comandos SQL;
- MoreLikeThis: busca de documentos similares a um documento na lista de resultados;

¹ <https://lucene.apache.org/solr/>

² <https://lucene.apache.org/>

- Plugins e Extensões: plugins para detecção de língua, clustering, e diversos softwares Apache, e extensões para indexação, tratamento de solicitações, parsing de queries, entre outros.

3 Instalação

Para realização deste trabalho utilizamos a sua versão estável mais recente, 8.6.1.³ O programa pode ser executado nos sistemas Windows, Linux e macOS, dos quais apenas os dois primeiros serão testados ao decorrer deste trabalho. Os únicos requisitos para a instalação do Solr é Java 8 ou superior (sem flags para opções experimentais: -xx JVM) e que a JVM esteja atualizada.^[4]

Para instalar o programa basta baixar os arquivos binários específicos para seu sistema - *.zip* para Windows e *.tgz* para Linux - e extrair os arquivos para o local desejado.

Para testar a instalação, abrimos um terminal e, dentro da pasta dos arquivos extraídos, executamos o seguinte comando:

```
$ bin/solr start // Linux
$ bin/solr.cmd start // Windows
```

Será iniciado um servidor local do Solr em seu navegador na porta 8983,⁴ onde deverá aparecer uma tela como indicada na Figura 1.

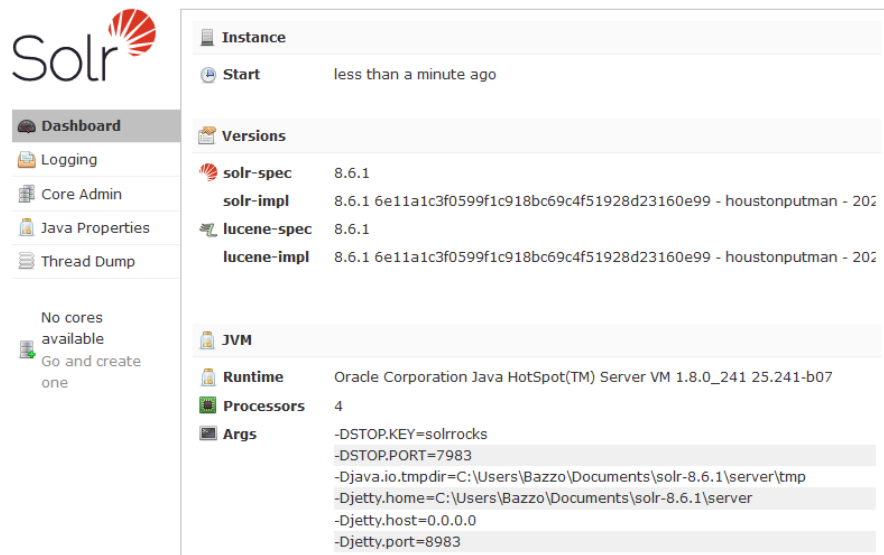


Fig. 1. Tela inicial do Solr.

³ <https://lucene.apache.org/solr/downloads.html>

⁴ <http://localhost:8983/solr/#/>

4 Indexação

4.1 Criação de Core

Para indexar os documentos é preciso criar um Core. Um Core é usado para referenciar um índice único e associado a logs de transações e arquivos de configurações. Solr pode utilizar múltiplos cores caso necessários, permitindo indexar dados com diferentes estruturas no mesmo servidor.[3]

No terminal, executamos o seguinte comando para criar um Core:

```
$ bin/solr create -c CMP269 // Linux
$ bin/solr.cmd create -c CMP269 // Windows
```

O comando cria o Core e adiciona os arquivos de configuração e esquema base necessários, assim como outros arquivos pré-definidos como stopwords e contrações em vários idiomas.

4.2 Processamento dos Arquivos

Antes de indexar os documentos precisamos alterar o formato dos arquivos SGML para um formato compatível com o Solr, do qual optamos pelo formato JSON. Para facilitar esse processamento criamos um *script* em Python.

Para instalar os pacotes necessários para o *script* e processar os arquivos executamos os seguintes comandos:

```
$ pip install -r requirements.txt
$ python sgml2json.py efe95/efe95/
```

4.3 Criar Campo de Indexação

No Solr, documentos são compostos de campos (*fields*), que são pedaços de informação. Ao indexar um documento os campos de informação são coletados e adicionados à um índice. Quando realizada uma pesquisa, o Solr consegue rapidamente consultar o índice e retornar os documentos encontrados. [2, 1]

Neste passo, realizamos a criação de um campo que vai englobar todos os campos dos documentos sendo indexados, para facilitar as consultas. Em um terminal, executamos o seguinte comando:

```
$ python create_field.py CMP269
```

Este passo é opcional, e que pode deixar as consultas mais lentas, já que todos os campos dos documentos estarão armazenados em um mesmo índice. Entretanto, ao realizarmos uma busca estaremos olhando para mais informações dos documentos ao mesmo tempo, como sua categoria, título e texto, o que esperamos aumentar a acurácia nas consultas.

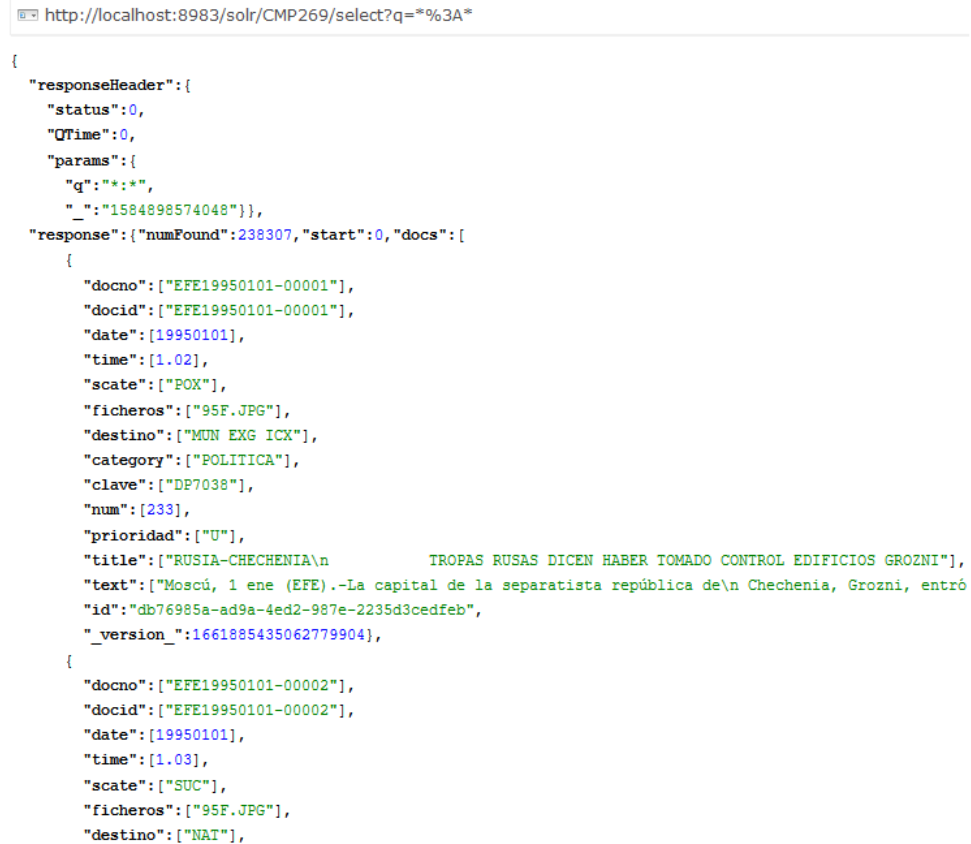
Outro ponto importante é a definição do novo campo baseado no tipo ‘text_es’. Como o texto sendo indexado e recuperado está em espanhol, o campo ‘text_es’ permite a utilização de *stopwords* e *stemmers* específicos para a língua espanhola.

4.4 Indexação

Com os passos anteriores devidamente realizados, podemos executar o comando abaixo para indexar os documentos.

```
$ bin/post -c CMP269 ../efe95/efe95/*.json // Linux
$ java -jar -Dc=CMP269 -Dauto example\exampledocs\post.jar
..\efe95\efe95/*.json // Windows
```

Como medida de verificação da indexação, podemos verificar o número de documentos na aba *Overview* do Core ou realizar uma pesquisa com a query “*:*”, que solicita todos os documentos. É esperado algo como mostrado na Figura 2.



The screenshot shows a web browser window with the address bar displaying `http://localhost:8983/solr/CMP269/select?q=%3A*`. The main content area displays a JSON response from the Solr API. The response includes a `responseHeader` with `status:0`, `QTime:0`, and `params` containing the query `q:*:*` and a unique identifier. The `response` object shows `numFound:238307` and `start:0`. It lists two documents with their respective fields: `docno`, `docid`, `date`, `time`, `scate`, `ficheros`, `destino`, `category`, `clave`, `num`, `prioridad`, `title`, `text`, `id`, and `_version_`.

```
{
  "responseHeader":{
    "status":0,
    "QTime":0,
    "params":{
      "q":":*",
      "_:":1584898574048"}},
  "response":{"numFound":238307,"start":0,"docs":[
    {
      "docno":["EFE19950101-00001"],
      "docid":["EFE19950101-00001"],
      "date":["19950101"],
      "time":["1.02"],
      "scate":["POX"],
      "ficheros":["95F.JPG"],
      "destino":["MUN EXG ICX"],
      "category":["POLITICA"],
      "clave":["DP7038"],
      "num":["233"],
      "prioridad":["U"],
      "title":["RUSIA-CHECHENIA\n          TROPAS RUSAS DICEN HABER TOMADO CONTROL EDIFICIOS GROZNI"],
      "text":["Moscó, 1 ene (EFE).-La capital de la separatista república de\n Chechenia, Grozni, entró"],
      "id":["db76985a-ad9a-4ed2-987e-2235d3cedfeb"],
      "_version_":1661885435062779904},
    {
      "docno":["EFE19950101-00002"],
      "docid":["EFE19950101-00002"],
      "date":["19950101"],
      "time":["1.03"],
      "scate":["SUC"],
      "ficheros":["95F.JPG"],
      "destino":["NAI"]}]}
```

Fig. 2. Query Solr

5 Consultas

As consultas para os 60 tópicos será feita utilizando a API do Solr. Em um *script* em Python realizamos todas as consultas e salvamos em um arquivo chamado ‘consultas.topicos.txt’.

As consultas são baseadas no título do tópico, sua descrição e narrativa. Consideramos todos os campos pois há informações relevantes em todos eles para a consulta. O título e a descrição resumizam o tópico, mas a narrativa trás detalhes que podem contribuir na recuperação de documentos mais relevantes.

A forma com que o Solr trabalha com dados textuais é definida a partir de analisadores, tokenizadores e filtros. Nenhum destes detalhes foram alterados de seus padrões durante o processo de indexação ou consulta neste trabalho. Porém, como indexamos os documentos sobre um campo de tipo ‘text_es’, como descrito na seção 4, alguns destes filtros já estão definidos para a língua espanhola. O tokenizador sendo utilizado é o *Standart Tokenizer*, que separa tokens por espaço em branco ou pontuações de separação. Os filtros sendo utilizados são *Lower Case Filter*, *Stop Filter* (*stopwords* em espanhol) e *Spanish Light Stem* para realizar o *stemming* do idioma espanhol.

Para realizar as consultas, executamos o *script* na seguinte forma:

```
$ python queries.py CMP269 Topicos_UTF8.txt
```

5.1 Resultados

Os arquivos resultados foram gerados no formato [‘Número Consulta’, ‘Q0’, ‘Número Documento (DOCID)’, ‘Ranking’, ‘score’, ‘Alunos’], como solicitado para avaliação.

Foram gerados dois arquivos de resultados, alterando entre eles a função do cálculo de ranking dos documentos. Um dos resultados é com a utilização da função BM25 (*Best Match*), que é padrão para esta versão do Solr, e para comparação geramos outro arquivo utilizando a função DFR (*Divergence-from-randomness*).

References

1. Copying Fields. https://lucene.apache.org/solr/guide/8_6/copying-fields.html, [Online; accessed 29-Agosto-2020]
2. Overview of Documents, Fields, and Schema Design. https://lucene.apache.org/solr/guide/8_6/overview-of-documents-fields-and-schema-design.html, [Online; accessed 29-Agosto-2020]
3. Solr Cores. https://lucene.apache.org/solr/guide/8_6/solr-cores-and-solr-xml.html, [Online; accessed 29-Agosto-2020]
4. Solr Downloads. <https://lucene.apache.org/solr/downloads.html>, [Online; accessed 29-Agosto-2020]
5. Solr Funcionalidades. <https://lucene.apache.org/solr/features.html#top>, [Online; accessed 29-Agosto-2020]