Documentos

ISSN 1677-9274
Setembro, 2017

152

Analisadores e mapeamento de um Sistema de Recuperação de Informação(SRI) de publicações técnico-científicas agropecuárias

Embrapa

ISSN 1677-9274
Setembro, 2017

Empresa Brasileira de Pesquisa Agropecuária
Embrapa Informática Agropecuária
Ministério da Agricultura, Pecuária e Abastecimento

Documentos 152

Analisadores e mapeamento de um Sistema de Recuperação de Informação(SRI) de publicações técnico-científicas agropecuárias

Glauber José Vaz

Embrapa Informática Agropecuária
Campinas, SP
2017

Embrapa Informática Agropecuária
Av. Dr. André Tosello, 209 - Cidade Universitária, Campinas - SP
Fone: (19) 3211-5700
https://www.embrapa.br/informatica-agropecuaria

Comitê de Publicações da Unidade
Presidente: Giampaolo Queiroz Pellegrino
Secretário-Executivo: Carla Cristiane Osawa
Membros: Adriana Farah Gonzales, Carla Geovana do Nascimento Macário, Flávia Bussaglia Fiorini, Ivo Pierozzi Júnior, Kleber X. Sampaio de Souza, Luiz Antonio Falaguasta Barbosa, Maria Goretti G. Praxedes, Paula Regina K. Falcão, Ricardo Augusto Dante, Sônia Ternes
Supervisão editorial: Kleber X. Sampaio de Souza
Revisão de texto: Adriana Farah Gonzales
Normalização bibliográfica: Maria Goretti G. Praxedes
Editoração eletrônica: Tuíra Santana Favarin, sob supervisão de Flávia Bussaglia Fiorini.
Imagem da capa: Tuíra Santana Favarin

1ª edição publicação digital - 2017

Dados Internacionais de Catalogação na Publicação (CIP)
Embrapa Informática Agropecuária

Vaz, Glauber José.
Analisadores e mapeamento de um Sistema de Recuperação de Informação(SRI) de publicações técnico-científicas agropecuárias / Glauber José Vaz. - Campinas : Embrapa Informática Agropecuária, 2017.
42 p. il.: color. ; 16 cm x 21 cm.- (Documentos / Embrapa Informática Agropecuária, ISSN 1677-9274 ; 152).

1. Sistemas. 2. Recuperação de Informação. 3. Elasticsearch. 3. Produção científica I. Vaz, Glauber José. II. Embrapa Informática Agropecuária. III. Título. IV. Série.

CDD 025.04 (21.ed.)

Autor

Glauber José Vaz
Bacharel e mestre em Ciência da Computação
Analista da Embrapa Informática Agropecuária,
Campinas, SP

Apresentação

Construir Sistemas de Recuperação de Informação (SRI) que atendam da melhor maneira possível a seus usuários é bastante complexo. Inúmeros detalhes devem ser considerados em seu projeto e os diferentes elementos que o compõem são altamente dependentes entre si, de maneira que pequenas alterações podem afetar o sistema como um todo.

Este trabalho trata de um SRI para a produção bibliográfica agropecuária, explicando em detalhes o processamento de texto tanto para a fase de indexação quanto para a de busca. Além disso, expõe o mapeamento de um banco de dados real para os índices dos documentos.

O SRI apresentado foi desenvolvido com a tecnologia Elasticsearch, que é baseada na Apache Lucene. As ferramentas utilizadas para a construção de SRIs continuam a evoluir e novas versões são lançadas com grande frequência, mas as soluções apresentadas neste trabalho são associadas a explicações conceituais. Portanto, mesmo que haja atualizações significativas nas ferramentas utilizadas, as ideias podem continuar a ser aplicadas em sistemas futuros.

Não é comum a disponibilização de obras na literatura que detalhem as decisões que nortearam a construção de SRIs específicos. Este trabalho visa a reduzir esta lacuna e contribui para o enriquecimento de projetos de sistemas como esse, especialmente aqueles voltados para a recuperação de documentos técnico-científicos.

Sílvia Maria Fonseca Silveira Massruhá
Chefe da Embrapa Informática Agropecuária

Sumário

Analisadores e mapeamento de um Sistema de Recuperação de Informação(SRI) de publicações técnico-científicas agropecuárias

Glauber José Vaz

Introdução

Sistemas de recuperação de informação (SRI) são fundamentais para se obter maior impacto a partir do conhecimento produzido sob forma de publicações técnico-científicas. O principal objetivo de um SRI é recuperar todos os documentos que são relevantes para uma consulta de usuário, enquanto recupera o mínimo possível de documentos irrelevantes. Na tentativa de satisfazer às necessidades de informação do usuário, o SRI deve, de alguma maneira, "interpretar" o conteúdo dos documentos de uma coleção e ordená-los em um *ranking* de acordo com um grau de relevância para a consulta do usuário. Esta "interpretação" dos documentos envolve a extração de informações sintáticas e semânticas do texto dos documentos e o uso destas informações para casar com as necessidades de informação do usuário (BAEZA-YATES; RIBEIRO-NETO, 2010).

A arquitetura de um SRI deve conter pelo menos dois componentes: um que faz a indexação dos documentos de uma coleção e outro que realiza a recuperação e o ranqueamento de documentos. O índice criado pelo primeiro elemento possibilita que a recuperação e o ranqueamento sejam realizados rapidamente. Neste trabalho, consideramos publicações técnico-científicas relacionadas à agropecuária. Em princípio, a coleção de

documentos é composta pela produção bibliográfica da (Empresa Brasileira de Pesquisa Agropecuária (Embrapa), mas outros documentos podem ser acrescentados. O SRI construído no âmbito deste trabalho também pode ser utilizado em outros contextos que envolvam publicações técnico-científicas.

Como não é comum a disponibilização de documentos que tratem com profundidade da implementação de SRIs, nosso objetivo é detalhar a maneira como as publicações técnico-científicas são "interpretadas", ou analisadas, nas duas etapas do SRI proposto, tanto na fase de indexação quanto na de busca. Além disso, mostramos o mapeamento entre os dados referentes às publicações da Embrapa e os campos criados para a indexação desses dados.

A próxima seção aborda as tecnologias empregadas em nosso sistema e explica o funcionamento geral da análise de textos e do mapeamento dos documentos para o índice. Depois, os analisadores e seus componentes utilizados na solução são detalhados. Em seguida, o mapeamento é explicado e finalmente as conclusões do trabalho são apresentadas.

Material e Métodos

A estrutura de índice mais comum é o índice invertido, em que cada termo distinto da coleção é mapeado para os documentos que o contém. O processo de geração deste índice é realizado antes que qualquer consulta seja feita pelos usuários. Portanto, pelo menos duas fases podem ser consideradas em SRIs: a de indexação e a de busca.

Marinho et al. (2012b) propuseram analisadores para as fases de indexação e de busca de um SRI que permite a realização de consultas à produção bibliográfica da Embrapa. A solução aqui proposta representa uma evolução desse trabalho, incorporando os resultados de Marinho et al. (2012a), Vaz (2016) e de Zanardo e Vaz (2013). Os documentos que utilizamos são lidos a partir de um serviço web do sistema Ainfo construído para gerar informações que alimentam o portal da Embrapa (2017).

Em relação às tecnologias utilizadas para a construção de mecanismos de buscas, há duas plataformas que são amplamente utilizadas e possuem código aberto: Apache Solr (THE APACHE SOFTWARE FOUNDATION, 2017) e Elasticsearch (ELASTIC, 2017). Em nossos primeiros trabalhos, adotamos a primeira. Posteriormente, o uso de Elasticsearch cresceu muito, inclusive na busca por conteúdo de sites importantes como GitHub, Stack Overflow e Wikipedia (GORMLEY; TONG, 2015). Então, também passamos a adotar esta tecnologia por motivos já apontados em Vaz (2016): a abordagem inovadora de possibilitar a construção de mecanismos de busca em textos, de busca estruturada e de análises; o oferecimento de uma interface simples via RESTful API; e o fato de apresentar uma curva de aprendizado que possibilita que novos usuários possam facilmente começar a utilizá-lo e, em pouco tempo, tornarem-se produtivos.

É importante salientar que o SRI foi construído com Elasticsearch na versão 1.7.1. Dada a constante evolução das tecnologias computacionais, os códigos podem não funcionar em versões futuras das ferramentas, mas como os conceitos atrelados a eles também são explicados detalhadamente, é possível ajustar de maneira mais fácil os códigos aqui apresentados para novas versões das tecnologias empregadas.

Tanto Solr quanto Elasticsearch são baseados na biblioteca de código aberto Apache Lucene, a qual oferece recursos de indexação e busca de textos. O índice invertido é criado por esta ferramenta.

Lucene, Solr e Elasticsearch utilizam analisadores tanto na fase de indexação quanto na de busca. A Figura 1 mostra a estrutura desses analisadores, que recebem uma cadeia de caracteres e retornam uma lista de *tokens*. Os analisadores de indexação determinam os *tokens* que correspondem aos termos do índice invertido, que são mapeados para os documentos que os contêm. Os analisadores de busca determinam os *tokens* que serão procurados no índice.

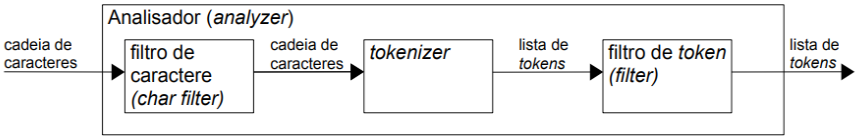

Figura 1. Estrutura de analisador de um SRI.

Um analisador pode conter filtros de caracteres que substituem uma cadeia de caracteres por outra. Por exemplo, um filtro de caractere pode ser usado para substituir a palavra "NULL", normalmente associada a uma informação faltante em um banco de dados, pela cadeia vazia. Enquanto filtros de caracteres são opcionais, é obrigatório o uso de exatamente um *tokenizer*, responsável por quebrar a cadeia de caracteres em uma lista de *tokens*. Por exemplo, *tokens* podem ser delimitados por caracteres de espaço em branco. Finalmente, essa lista de *tokens* pode ser modificada por uma sequência de diferentes filtros de *token*, que também são opcionais. Portanto, o analisador recebe uma cadeia de caracteres e retorna uma lista de *tokens*.

Os analisadores precisam ser construídos de acordo com os dados processados. Por exemplo, a análise dos nomes dos autores das obras deve ser realizada de maneira diferente da análise do resumo das obras. O que determina a maneira com que os documentos e seus campos são armazenados e indexados é o mapeamento. Basicamente, ele estabelece para cada campo dos documentos um tipo de dado e analisadores de indexação e de busca. Ainda pode especificar, por exemplo, se os valores dos campos são copiados para campos complementares ou se informações adicionais devem ser armazenadas no índice.

Na próxima seção são explicados os analisadores utilizados no SRI proposto, detalhando-se seus filtros de caracteres, *tokenizers* e filtros de *tokens*. A seção seguinte explica o mapeamento.

Analisadores

No Elasticsearch, os componentes de analisadores podem ser definidos conforme código da Figura 2. Em primeiro lugar, definimos uma lista de filtros de caracteres, depois uma lista de *tokenizers* e, ainda, uma lista de filtros de *token*. Finalmente, uma lista de analisadores que fazem uso dos elementos previamente estabelecidos. É importante destacar que vários componentes já são disponibilizados pela própria tecnologia e podem ser usados diretamente, sem que seja necessário declará-los. Somente são declarados nesse trecho aqueles componentes que foram customizados para a aplicação de interesse.

```
"analysis": {
      "char_filter": {},
      "tokenizer" : {},
      "filter": {},
      "analyzer":{}
}
```

Figura 2. Definição dos componentes de analisadores em Elasticsearch.

Os filtros de caracteres utilizados são exibidos na Figura 3. Todos eles mapeiam uma cadeia de caracteres ou um caractere para um outro caractere. O filtro 'null_to_empty' substitui cadeias de caracteres "NULL" por caractere vazio. Este filtro é necessário porque em alguns casos, na leitura das informações de produção bibliográfica, algum campo pode ter valor "NULL" indicando que não há informação disponível. Então é necessário fazer esse ajuste antes de efetivamente indexar o documento. Os filtros 'dot_to_blank' e 'comma_to_blank' substituem, respectivamente, ponto final e vírgula por espaço em branco, representado por "\\u0020". Para ilustrar o uso destes filtros, quando aplicados à cadeia "VAZ,G.J.", geram a saída "VAZ G J ". Este exemplo mostra porque o caractere deve ser espaço em branco e não cadeia vazia. Caso contrário, o resultado da aplicação destes filtros seria "VAZGJ", o que não é desejado por juntar sobrenome de autor com iniciais do nome. Já o filtro 'null_to_empty', quando aplicado à cadeia "NULL", gera "".

```
"char_filter": {
      "null_to_empty": {
            "type":        "mapping",
            "mappings": [ "NULL => "]
      },
      "dot_to_blank": {
            "type":        "mapping",
            "mappings": [ ". => \\u0020"]
      },
      "comma_to_blank": {
            "type":        "mapping",
            "mappings": [ ", => \\u0020"]
```

Figura 3. Definição de filtros de caracteres.

Em relação aos *tokenizers*, utilizamos principalmente aqueles já disponibilizados pelo Elasticsearch, 'uax_url_email' e 'letter'. O primeiro funciona como o *tokenizer* 'standard' (GORMLEY; TONG, 2015), que usa o algoritmo de segmentação de texto Unicode (DAVIS; IANCU, 2016) e obtém sucesso com textos em diferentes línguas. No entanto, o 'uax_url_email' também reconhece endereços de e-mail e URLs como *tokens*, o que o 'standard' não faz. Já o 'letter' quebra termos em cada ocorrência de caractere que não corresponde a uma letra. Ele geraria com a entrada "VAZ,G.J.", por exemplo, os *tokens* "VAZ", "G" e "J".

Além destes dois *tokenizers*, ainda precisamos customizar um outro, que é exibido na Figura 4. Basicamente, este componente separa termos de acordo com a ocorrência de ponto-e-vírgula ';'. Por exemplo, para a entrada "BAMBINI, M. D.; VAZ, G. J.; BARBEDO, J. G. A.", gera os *tokens* "BAMBINI, M. D.", "VAZ, G. J." e "BARBEDO, J. G. A.".

```
"tokenizer" : {
      "semicolon_tokenizer" : {
            "type" : "pattern",
            "pattern": "[*;]+"
      }
}
```

Figura 4. Definição de *tokenizer*.

Vários filtros de *token* foram utilizados no SRI. A Figura 5 exibe os filtros customizados. Além destes, outros filtros já disponibilizados pela tecnologia e que não necessitam de configuração também foram utilizados: 'asciifolding', 'lowercase' e 'trim'. O primeiro elimina diacríticos, o segundo transforma todas as letras para minúsculas e o último elimina espaços nas extremidades dos *tokens*.

O filtro 'synonym_filter' é utilizado para obter sinônimos dos *tokens* de acordo com a lista de relações de sinonímias fornecida pelo arquivo indicado por 'synonyms_path'. Este arquivo contém linhas com sinônimos separados por vírgula, como por exemplo, "radiacao solar,luz solar" e é analisado com o *tokenizer* 'keyword', pois considera as cadeias de caracteres separadas por vírgula como *tokens* únicos. Então, neste exemplo, "radiacao solar" e "luz solar" são considerados tokens pelo filtro, enquanto termos como "radiacao", "luz" ou "solar" não são. Também é ignorado se letras são maiúsculas ou minúsculas ("ignore_case": true). Portanto, para cada ocorrência de "radiacao solar", são gerados os *tokens* "radiacao solar" e "luz solar". O mesmo ocorre para as ocorrências de "luz solar".

O filtro 'author_synonym_filter' funciona exatamente da mesma maneira. O que o diferencia do 'synonym_filter' é a lista de relações de sinonímias. No lugar de termos relacionados ao domínio do problema, neste filtro são enumerados nomes de autores de publicações que se referem à mesma pessoa. Por exemplo, "alves e,alves e r a" contém duas formas com que um mesmo autor pode ser citado em suas obras. Além de casos como este que envolve o uso de diferentes partes do nome, há casos em que o autor tem o nome modificado devido ao matrimônio ou à catalogação equivocada, por exemplo.

O filtro 'author_stop_filter' também é aplicado às informações de autor. Basicamente, é utilizado na indexação dos sobrenomes dos autores para eliminar as iniciais de seus nomes. Uma lista de *tokens* com as cadeias ["vaz", "g", "j"], por exemplo, gera a lista ["vaz"], justamente para se extrair apenas o sobrenome. Considera-se que este filtro é utilizado depois que as palavras passaram por um filtro que gera apenas letras minúsculas.

Por isso, a lista de stopwords é composta por todas as letras minúsculas do alfabeto.

```
"filter": {
      "synonym_filter":{
            "type": "synonym",
            "synonyms_path":"<PATH>/synonyms.txt",
            "ignore_case":true,
            "tokenizer":"keyword"
      },
      "author_synonym_filter":{
            "type": "synonym",
            "synonyms_path":"<PATH>/author_synonyms.txt",
            "ignore_case":true,
            "tokenizer": "keyword"
      },
      "author_stop_filter": {
            "type":       "stop",
            "stopwords": ["a","b","c","d","e","f","g","h","i",
                          "j","k","l","m","n","o","p","q","r",
                          "s","t","u","v","w","x","y","z"],
            "ignore_case":true
      },
      "compound" : {
            "type" : "compound_terms",
            "shingle" : "<PATH>/vocabulary.txt, <PATH>/synonyms.txt"
      },
      "correct_syn":{
            "type":"correct_synonyms"
      },
      "words_on_file" : {
            "type" : "keep",
            "keep_words_path" : "<PATH>/vocabulary.txt"
      },
      "one_whitespace": {
            "type": "pattern_replace",
            "pattern": "\\s{2,}",
            "replacement": " "
      }
},
```

Figura 5. Definição de filtros de *token.*

Os filtros 'compound' e 'correct_syn' são explicados detalhadamente em Vaz (2016). O primeiro trata de termos compostos por duas ou mais palavras e o segundo apenas corrige um erro introduzido pelo uso combinado dos filtros do tipo 'synonym' com o 'compound', garantindo maior qualidade das consultas, especialmente aquelas que envolvem buscas por frases. Os tipos declarados 'compound_terms' e 'correct_ synonyms' podem ser usados desde que seus plugins sejam instalados previamente. O primeiro ainda exige um parâmetro 'shingle' que indica um ou mais arquivos com termos compostos. Este parâmetro é necessário porque nem todos os possíveis termos compostos são considerados, mas apenas aqueles que estão relacionados nestes arquivos. Em nosso sistema, utilizamos os arquivos 'synonyms.txt' e 'vocabulary. txt'. O primeiro é o mesmo utilizado pelo filtro de sinônimos. Assim, termos como "radiacao solar" e "luz solar", presentes no arquivo, são considerados termos compostos. Outros termos do domínio que devem ser considerados como termos compostos e que não têm relações de sinonímia devem estar no arquivo 'vocabulary.txt'.

O filtro 'words_on_file' mantém apenas termos enumerados no arquivo especificado por 'keep_words_path'. É importante para obter os termos presentes no documento analisado e que fazem parte de um glossário no domínio do problema. Por exemplo, "balanço hídrico" é um termo presente neste arquivo para o qual há uma definição formal: "Balanço do fluxo de água que entra e sai de um sistema (solo, rios, lagos, vegetação úmida e oceanos) em um determinado período de tempo." Mais adiante, o glossário será tratado em detalhes.

Finalmente, o filtro 'one_whitespace' substitui sequências de dois ou mais espaços em branco por exatamente um. Por exemplo, a entrada com vários espaços em branco "VAZ G J" gera a saída "VAZ G J", que contém apenas um espaço em branco entre os diferentes componentes do termo "VAZ", "G" e "J".

Definidos os filtros de caracteres, *tokenizers* e filtros de *tokens*, analisadores podem ser customizados fazendo uso destes elementos. Embora o Elasticsearch já forneça vários analisadores, outros podem ser

criados para atender melhor a demandas específicas. A Figura 6 exibe todos os analisadores que foram customizados e é justamente por isso que são declarados do tipo 'custom'.

Os analisadores mais usados no nosso SRI são 'basic_analyzer', 'index_content' e 'search_content'. O primeiro é usado tanto para indexação quanto para busca e compõe a base dos dois outros. É utilizado em campos que não estão relacionados ao tema do domínio do problema, como, por exemplo, os que fornecem o tipo da publicação ou os nomes dos autores. Os outros dois estão mais associados ao conteúdo das publicações, como título, resumo e palavras-chaves. Enquanto 'index_content' é usado para a indexação, o 'search_content' é usado na busca.

A diferença entre o 'search_content' e o 'index_content' está apenas nos dois últimos filtros. Não é necessário usar filtros de sinônimos na consulta porque estes são considerados na fase de indexação e isso já garante o retorno de documentos que contenham os termos sinônimos. O 'correct_syn' tampouco é necessário, uma vez que é utilizado apenas em combinação com o filtro de sinônimos. Já o analisador 'basic_analyzer' equivale ao 'search_content' sem o uso do filtro 'compound'. A Figura 7 ilustra o processamento realizado por estes analisadores do seguinte título de uma obra: "Água e agricultura: incertezas e desafios para a sustentabilidade frente às mudanças do clima e do uso da terra: anais.". Na figura, o último quadro exibe a saída do analisador 'index_content', o penúltimo exibe a saída do 'search_content' e o antepenúltimo a do 'basic_analyzer'. São colocados em destaque na figura os elementos que sofrem alterações. Por exemplo, o ponto final do primeiro quadro não aparece no segundo. Duas ocorrências de ":" deste quadro são excluídas pelo *tokenizer* e não aparecem no terceiro quadro. A cor vermelha indica elementos que são alterados no quadro seguinte. A cor verde destaca estes elementos com as modificações efetuadas.

```
"analyzer":{
      "basic_analyzer": {
            "type":           "custom",
            "char_filter":  [ "null_to_empty", "dot_to_blank" ],
            "tokenizer":      "uax_url_email",
            "filter":       [ "asciifolding", "lowercase", "trim"]
      },
      "index_content": {
            "type":           "custom",
            "char_filter":  [ "null_to_empty", "dot_to_blank" ],
            "tokenizer":      "uax_url_email",
            "filter":       [ "asciifolding", "lowercase", "trim",
                              "compound", "synonym_filter",
                              "correct_syn" ]
      },
      "search_content":{
            "type":           "custom",
            "char_filter":  [ "null_to_empty", "dot_to_blank" ],
            "tokenizer":      "uax_url_email",
            "filter":       [ "asciifolding", "lowercase", "trim",
                              "compound" ]
      },
      "index_full_author":{
            "type":           "custom",
            "char_filter":  [ "null_to_empty", "dot_to_blank",
                              "comma_to_blank" ],
            "tokenizer":      "semicolon_tokenizer",
            "filter":       [ "asciifolding", "lowercase", "trim",
                              "one_whitespace", "author_synonym_filter" ]
      },
      "search_full_author":{
            "type":           "custom",
            "char_filter":  [ "null_to_empty", "dot_to_blank",
                              "comma_to_blank" ],
            "tokenizer":      "semicolon_tokenizer",
            "filter":       [ "asciifolding", "lowercase", "trim",
                              "one_whitespace"]
      },
```

Continua

Continuação

```
      "surname_author":{
            "type":           "custom",
            "char_filter":  [ "null_to_empty" ],
            "tokenizer":      "letter",
            "filter":       [ "asciifolding", "lowercase", "trim",
                              "author_stop_filter" ]
      },
      "index_keywords":{
            "type":           "custom",
            "char_filter":  [ "null_to_empty", "dot_to_blank",
                              "comma_to_blank" ],
            "tokenizer":      "semicolon_tokenizer",
            "filter":       [ "asciifolding", "lowercase", "trim",
                              "one_whitespace", "synonym_filter" ]
      },
      "search_keywords":{
            "type":           "custom",
            "char_filter":  [ "null_to_empty", "dot_to_blank",
                              "comma_to_blank" ],
            "tokenizer":      "semicolon_tokenizer",
            "filter":       [ "asciifolding", "lowercase", "trim",
                              "one_whitespace" ]
      },
      "index_glossary":{
            "type":           "custom",
            "char_filter":  [ "null_to_empty", "dot_to_blank" ],
            "tokenizer":      "uax_url_email",
            "filter":       [ "asciifolding", "lowercase", "trim",
                              "compound", "synonym_filter", "correct_syn",
                              "words_on_file" ]
      },
      "exact_text": {
            "type":           "custom",
            "char_filter":  [ "null_to_empty", "dot_to_blank" ],
            "tokenizer":      "uax_url_email",
            "filter":       [ "lowercase" ]
      }
}
```

Figura 6. Definição de analisadores.

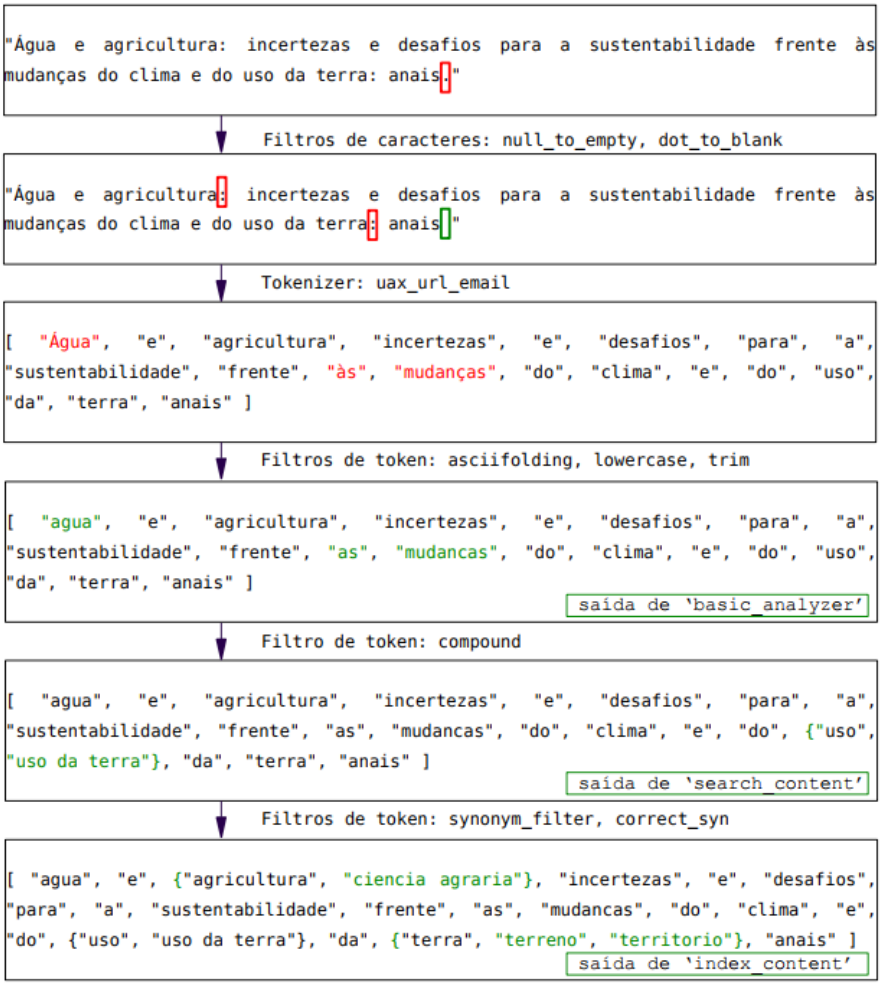

Figura 7. Análise feita com 'basic _ analyzer', 'search _ content' e 'index _ content'.

Primeiramente, os filtros de caracteres 'null_to_empty' e 'dot_to_blank' são aplicados em sequência. Como não há "NULL" neste título, a cadeia não é alterada pelo primeiro filtro, mas o segundo troca o ponto final por espaço, e a cadeia de entrada passa a ser finalizada com um espaço em branco no lugar do ponto final depois da palavra "anais". Em sequência, o tokenizer 'uax_url_email' quebra a cadeia de entrada e gera a lista

de *tokens* representada no terceiro quadro da Figura 7, eliminando as ocorrências de ':'. O filtro 'asciifolding' substitui os *tokens* "Água", "às" e "mudanças" por "Agua", "as" e "mudancas", respectivamente. Com a aplicação de 'lowercase', "Agua" é substituído por "agua". Após o filtro 'trim', que não produz efeitos neste caso, a lista de *tokens* se apresenta conforme o quarto quadro da Figura 7, o que já equivale à saída produzida pelo 'basic_analyzer'. No quadro seguinte, que representa a saída do 'search_content', o *token* "uso da terra" é incluído à lista devido ao uso do filtro 'compound'. Este termo é relacionado no arquivo 'vocabulary.txt', que contém os termos relevantes para o domínio da aplicação. Os *tokens* "uso" e "uso da terra" ocupam a mesma posição na lista, o que é representado pelos caracteres '{' e '}'. A aplicação de 'synonym_filter' gera o *token* "ciencia agraria", sinônimo de "agricultura", além de "terreno" e "territorio", sinônimos de "terra". Com o uso do filtro de sinônimos neste analisador, os documentos são indexados como se os termos sinônimos também estivessem presentes neles. Então, se alguém busca por "ciência agrária", por exemplo, o documento do exemplo seria retornado. O filtro 'correct_syn' não altera os termos correspondentes aos *tokens*, mas apenas suas posições na lista de saída. Ele garante que termos sinônimos sempre ocupem as mesmas posições na lista produzida pelo analisador 'index_content' e representada no último quadro da figura.

Os *tokens* ainda contêm outras informações, além dos termos em si e de sua posição na lista de *tokens*, como, por exemplo, a posição de seu caractere inicial na cadeia de entrada, seu tamanho em quantidade de caracteres e seu tipo. Porém, estas informações não são relevantes no contexto deste trabalho e não são exibidas nas figuras.

Devido às suas particularidades, as informações de autores requerem analisadores específicos. Conforme definição na Figura 6, utilizamos três para tratar este tipo de informação: 'index_full_author', 'search_full_author' e 'surname_author'. A relação de autores é especificada conforme este exemplo: "MARINHO, I. J. P.; CARDONE, H. T. M.; VAZ, G. J.". Os nomes dos autores são separados por ponto e vírgula e cada um contém o sobrenome seguido por suas iniciais.

A Figura 8, que exibe uma análise feita com 'index_full_author', mostra que os pontos depois das iniciais e as vírgulas após os sobrenomes são substituídos por espaços em branco pelos filtros 'dot_to_blank' e 'comma_to_blank'. Se a lista de autores não é fornecida, o campo de autores pode conter "NULL", que é substituído por caractere vazio devido ao filtro 'null_to_empty'. É importante notar que há dois espaços em branco entre os elementos de cada termo, um original e outro obtido pela substituição de pontos e vírgulas por espaço. Os retângulos vermelhos indicam elementos que são modificados no quadro seguinte e os verdes destacam as alterações efetuadas.

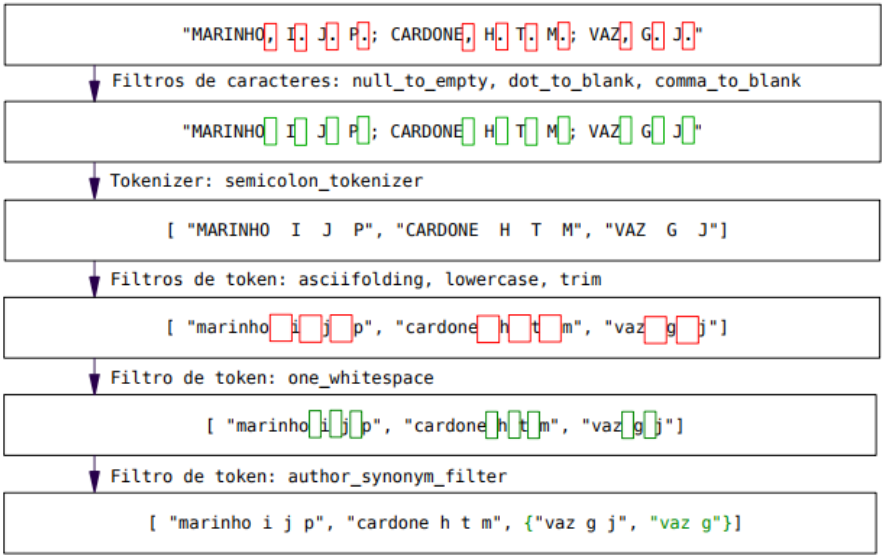

Figura 8. Análise feita com 'index _ full _ author'.

O *tokenizer* 'semicolon_tokenizer' delimita termos separados por ponto e vírgula, gerando assim um *token* por autor, conforme terceiro quadro da Figura 8. Assim como nos analisadores explicados anteriormente, os filtros 'asciifolding', 'lowercase' e 'trim' também são usados no 'index_full_author'.

Sequências de dois ou mais espaços em branco podem estar presentes nos *tokens*. Os quadros quarto e quinto da figura mostram a substituição dessas sequências por um único caractere devido ao filtro 'one_whitespace', cuja importância pode ser verificada pelo seguinte cenário: se o *token* fosse mantido como "vaz g j", com dois caracteres em branco precendendo as iniciais, buscas por "vaz g j", com apenas um caractere em branco precedendo as iniciais, não corresponderiam ao termo indexado.

O último quadro exibe a inclusão de um novo termo à lista de saída, o que só é possível se o arquivo 'author_synonyms.txt', utilizado por 'author_synonym_filter', possuir uma linha como "vaz g j,vaz g", indicando que ambas as formas "vaz g j" e "vaz g" referem-se a um mesmo autor. Cada linha desse arquivo deve ter nomes em letras minúsculas, separados por vírgula e declarados sem pontos ou vírgulas e com apenas um espaço em branco entre seus componentes. A adoção desta notação possibilita o tipo de análise desejado para o SRI e explica a importância dos filtros 'dot_to_blank' e 'comma_to_blank', bem como o motivo porque o filtro 'one_whitespace' deve preceder o 'author_synonym_filter'.

Não há necessidade de se utilizar o 'correct_syn' no 'index_full_author' porque não há o uso combinado de 'compound' com o filtro de sinônimos.

Assim como o 'search_content', o analisador 'search_full_author', que também é utilizado na fase de buscas, não utiliza filtros de sinônimos. E esta é a única diferença para o 'index_full_author', utilizado na indexação de autores.

Estes analisadores não consideram partes dos nomes dos autores, pois os tomam por completo, envolvendo sobrenome e iniciais conjuntamente. Em algumas situações, no entanto, pode ser interessante utilizar os sobrenomes dos autores isoladamente, já que é comum um usuário realizar buscas por sobrenomes de autores sem conhecer as demais partes de seu nome. O analisador 'surname_author', representado na Figura 9, possibilita isso.

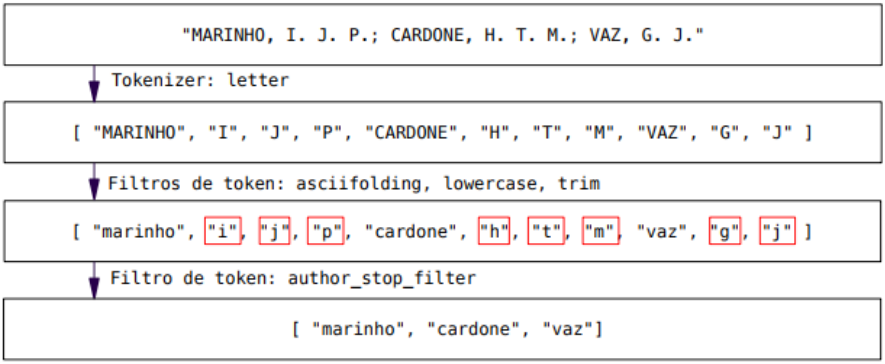

Figura 9. Análise feita com 'surname _ author'.

Este filtro não precisa de 'dot_to_blank' ou de 'comma_to_blank' porque pontos e vírgulas já são eliminados pelo *tokenizer*. Neste exemplo, o filtro 'null_to_empty' não surte efeito porque não há "NULL" na entrada. Por isso, a Figura 9 já exibe em seu segundo quadro a saída para o *tokenizer* 'letter', que separa *tokens* delimitando-os por ocorrências de caracteres que não correspondam a letras. Após os filtros 'asciifolding', 'lowercase' e 'trim', explicados anteriormente, o 'author_stop_filter' elimina *tokens* formados por apenas uma letra.

As palavras-chave associadas aos documentos também são separadas por ';', como em "Portlet; Recuperação da informação; Arquitetura orientada a serviço", de maneira parecida com a especificação de autores. O analisador 'index_keywords', portanto, é essencialmente igual ao 'index_full_author', com a diferença de que o filtro de sinônimos utilizado referencia um arquivo com o vocabulário do domínio em vez de um arquivo com nomes de autores. O analisador 'search_keywords', por sua vez, é igual ao 'index_keywords', exceto pela ausência do filtro de sinônimos, uma vez que é utilizado para buscas.

O analisador 'index_glossary' é utilizado para identificar nos documentos termos que fazem parte do glossário do domínio. Funciona como o 'index_content', conforme mostra a Figura 10, mas finaliza com o filtro 'words_on_file', que só não descarta os termos da lista de tokens que

são enumerados no arquivo indicado. Esses termos são aqueles que possuem uma definição no glossário utilizado pelo sistema. No exemplo da Figura 10, os termos “agricultura”, “sustentabilidade” e “uso da terra” fazem parte do glossário. Por isso, estão na saída do analisador. É importante que o ‘words_on_file’ seja aplicado após os filtros ‘compound’ e ‘synonym_filter’ para conseguir capturar termos compostos e sinônimos presentes no glossário. Uma aplicação interessante para o resultado desse analisador é a exibição das definições dos termos presentes nos documentos recuperados por uma consulta.

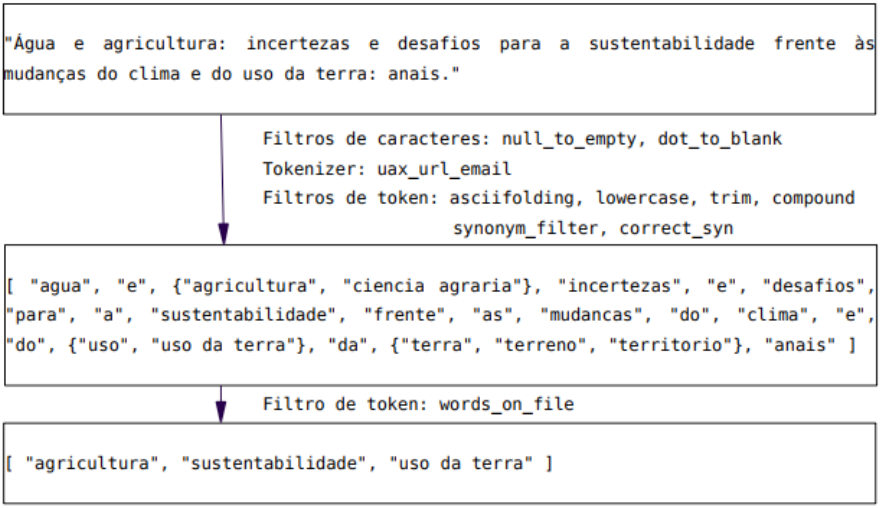

Figura 10. Análise feita com ‘index _ glossary’.

Finalmente, o analisador ‘exact_text’ é responsável por melhorar a qualidade das buscas, principalmente as que envolvem sinônimos e palavras diferenciadas por acentos. Marinho et al. (2012a) ilustram um exemplo envolvendo o estado do Pará para mostrar a importância de um analisador como este. Devido ao filtro ‘asciifolding’, documentos que apresentam, por exemplo, a palavra “Pará” ou “para” apresentam a mesma relevância se considerarmos apenas os analisadores apresentados até aqui. Com o uso combinado de ‘exact_text’ com os demais analisadores, é possível atribuir maior peso a termos conforme são fornecidos pelo usuário. A ideia, portanto, é criar um campo adicional

em que menos filtros são aplicados para manter os termos em uma forma mais próxima da que está no documento ou na consulta. Optamos também por utilizar o 'lowercase' para ignorar diferenças promovidas pelo uso de letras maiúsculas, embora esta distinção seja importante em alguns casos. Este analisador é utilizado tanto em indexações quanto em buscas.

Mapeamento

Os analisadores explicados na seção anterior foram construídos especificamente para tratar as informações associadas a publicações técnico-científicas da Embrapa, embora as ideias expostas aqui possam ser utilizadas para construir analisadores para outras aplicações. Nesta seção, explicamos como esses analisadores são utilizados na indexação e na busca dessas publicações através do mapeamento, que é o processo de definir como um documento e os campos que ele contém são armazenados e indexados (GORMLEY; TONG, 2017).

Antes disso, porém, é necessário mostrar o exemplo de uma publicação, conforme ela pode ser indexada no sistema. A Figura 11 mostra a inclusão no índice 'bdpa' do documento do tipo 'documento', cujo identificador é 10. Ele possui os campos 'TITULO', 'AUTORIA', 'FONTE', 'ANO', 'PALAVRAS_CHAVES', 'RESUMO' e 'TIPO'. Um mapeamento deve especificar como esses campos devem ser analisados. A Figura 12 mostra o mapeamento concebido para o SRI.

```
PUT /bdpa/documento/10
{
 "TITULO" : "Uma arquitetura orientada a serviços para um sistema de
recuperação de informação para a Rede Agrohidro.",
  "AUTORIA" : "VAZ, G. J.",
  "FONTE" : "In: SEMINARIO DA REDE AGROHIDRO, 4., 2016, Brasília, DF. Agua
e agricultura: incertezas e desafios para a sustentabilidade frente às
mudanças do clima e do uso da terra: anais. Brasília, DF: Embrapa, 2016.",
  "ANO" : 2016,
 "PALAVRAS_CHAVES" : "Mecanismo de busca; Portlet; Recuperação da
informação; Search engine; Service Oriented Architecture (SOA)",
  "RESUMO" : "Sistemas de recuperação de informação são fundamentais para
o acesso a informações nas organizações. Este recurso pode ser acessível,
interna ou externamente, por outras aplicações e estar disponível em
diferentes páginas web, o que motiva tratá-lo como um serviço. Neste
artigo, apresentamos uma arquitetura de sistema de recuperação de
informação orientado a serviço e um caso de uso para a Rede Agrohidro,
grupo de pesquisa que trabalha com recursos hídricos na agricultura. O
sistema é formado por dois grandes componentes, um, acessível via API
RESTful, que faz indexação e busca, e outro, baseado em portlets, que
oferece interface de usuário. A abordagem proposta promove escalabilidade,
simplicidade e interoperabilidade de sistemas. Também favorece a economia
de recursos, a unificação de métodos de indexação e de busca, maior
alcance das informações e a melhor experiência dos usuários.",
  "TIPO" : "Artigo em Anais de Congresso"
}
```

Figura 11. Exemplo de documento indexado no SRI.

```
PUT /bdpa/_mapping/documento
{
    "dynamic": "strict",
    "properties": {
        "TITULO": {
            "type":            "string",
            "index_analyzer":  "index_content",
            "search_analyzer": "search_content",
            "copy_to":         ["original_text", "glossario"]
        },
        "RESUMO":{
            "type":            "string",
            "index_analyzer":  "index_content",
            "search_analyzer": "search_content",
            "copy_to":         ["original_text", "glossario"]
        },
        "FONTE":{
            "type":            "string",
            "index_analyzer":  "index_content",
            "search_analyzer": "search_content",
            "copy_to":         ["original_text", "glossario"]
        },
        "PALAVRAS_CHAVES":{
            "type":            "string",
            "index_analyzer":  "index_content",
            "search_analyzer": "search_content",
            "copy_to":         ["original_text", "glossario"],
            "fields": {
              "full": {
                "type":            "string",
                "index_analyzer":  "index_keywords",
                "search_analyzer": "search_keywords"
                }
            }
        },
        "TIPO":{
            "type":     "string",
            "analyzer": "basic_analyzer",
            "copy_to":  "original_text"
        },
```

Continua

Continuação

```
        "AUTORIA":{
            "type":     "string",
            "analyzer": "basic_analyzer",
            "copy_to":  "original_text",
            "fields": {
              "full": {
                "type":             "string",
                "index_analyzer":  "index_full_author",
                "search_analyzer": "search_full_author",
              },
              "surname": {
                "type":     "string",
                "analyzer": "surname_author"
              }
            }
        },
        "ANO":{
            "type":    "integer"
        },
        "LOCAL_NAME":{
            "type": "string"
        },
        "LOCAL_COORDS": {
            "type": "geo_shape"
        },
        "original_text":{
            "type":     "string",
            "analyzer": "exact_text"
        },
        "glossario":{
            "type":             "string",
            "index_analyzer":  "index_glossary",
            "search_analyzer": "search_content",
            "term_vector":     "yes"
        }
    }
}
```

Figura 12. Mapeamento de 'documento'.

Os campos associados ao conteúdo do documento normalmente são indexados de uma mesma maneira. 'TITULO', 'PALAVRAS_CHAVES' e 'RESUMO' fornecem, respectivamente, o título da obra, as palavras-chave associadas a ela e seu resumo. Além destes, 'FONTE' fornece a fonte ou imprenta da obra, que pode conter elementos relacionados ao seu conteúdo, como, por exemplo, no caso do documento da Figura 11, que apresenta neste campo termos como "água", "agricultura", "sustentabilidade", "mudanças do clima" e "uso da terra". Declarados do tipo 'string', todos estes campos usam o analisador 'index_content' na indexação e o 'search_content' na busca. Além disso, têm seu conteúdo copiado para os campos complementares 'original_text' e 'glossario', de maneira que os mesmos dados sejam analisados de diferentes maneiras.

O campo 'original_text' é usado para melhorar a qualidade dos resultados de busca, conforme já explicado no contexto do analisador 'exact_text'. Este analisador foi construído justamente para ser usado na indexação e na busca do 'original_text', como pode ser observado na Figura 12. Novos campos complementares podem ser construídos a fim de melhorar os resultados das buscas. Como o SRI em questão possui uma demanda por consultas muito maior do que por indexações, esta é uma estratégia interessante para se alcançar maior qualidade do sistema.

O campo 'glossario', por sua vez, possibilita a identificação nos documentos de termos que fazem parte do glossário do domínio previamente estabelecido. É processado pelo analisador 'index_glossary' durante a indexação justamente para que apenas termos presentes no glossário sejam indexados. Já a busca é feita com o 'search_content', uma vez que o 'index_glossary' é baseado no 'index_content'. O campo ainda é mapeado com o valor "yes" para 'term_vector'. Os *term vectors* armazenam informações adicionais sobre os documentos, como, por exemplo, a lista de termos e as posições na cadeia de entrada destes termos e de seus caracteres de início e de fim. No entanto, no exemplo, apenas a lista de termos é armazenada para cada documento. Outras informações são armazenadas quando outros valores de 'term_vector' são usados, como "with_positions", "with_offsets" ou "with_positions_offsets".

A Figura 13 mostra na primeira linha um comando para consultar os term vectors do campo 'glossario'. O resultado obtido e exibido abaixo considera que apenas o documento representado na Figura 11 é indexado em 'bdpa' e que os termos "agricultura", "sustentabilidade" e "uso da terra" fazem parte do glossário do sistema. Portanto, usar *term vectors* no campo 'glossario' de acordo com o mapeamento exibido na Figura12 possibilita a recuperação dos termos que estão presentes em um determinado documento e no glossário do sistema. Isso permite agregar novas funcionalidades ao SRI, como, por exemplo, a exibição das definições de termos do glossário quando estes estiverem presentes nos documentos listados como resultado para uma consulta.

As palavras-chaves associadas às publicações ainda têm um tratamento adicional para que também sejam analisadas como termos únicos. Por exemplo, o texto "Mecanismo de busca; Portlet; Recuperação da informação" é indexado em 'PALAVRAS_CHAVES' com os termos "busca", "da", "de", "informacao", "mecanismo", "portlet" e "recuperacao". Já no campo 'full' de 'PALAVRAS_CHAVES', os termos indexados são "mecanismo de busca", "portlet" e "recuperacao da informacao", devido aos analisadores 'index_keywords' e 'search_ keywords'.

O campo 'TIPO' identifica a natureza da publicação, como, por exemplo, artigo em periódico, artigo em anais de congresso ou documento da série Embrapa. Como não está associado ao domínio do problema, não é necessário copiar o conteúdo deste campo para 'glossario'. É por este mesmo motivo que os filtros 'compound' e 'synonym_filter' não precisam fazer parte da análise deste campo e o 'basic_analyzer' é suficiente.

Da mesma forma, os nomes dos autores não têm palavras que pertencem ao domínio do problema. Portanto, também são processados por 'basic_ analyzer' e não são copiados para 'glossario'. Porém, os nomes podem ser analisados de diferentes maneiras, conforme discutido anteriormente e ilustrado nas Figuras 8 e 9. Os campos 'full' e 'surname' de 'AUTORIA' utilizam os analisadores 'index_full_author', 'search_full_author' e 'surname_author'.

```
GET /bdpa/documento/10/_termvector?fields=glossario

{
   "_index": "bdpa",
   "_type": "documento",
   "_id": "10",
   "_version": 1,
   "found": true,
   "took": 1,
   "term_vectors": {
      "glossario": {
         "field_statistics": {
            "sum_doc_freq": 3,
            "doc_count": 1,
            "sum_ttf": 4
         },
         "terms": {
            "agricultura": {
               "term_freq": 2
            },
            "sustentabilidade": {
               "term_freq": 1
            },
            "uso da terra": {
               "term_freq": 1
            }
         }
      }
   }
}
```

Figura 13. Uso de *term _ vectors* no campo 'glossario'.

Para o campo 'ANO', basta declará-lo do tipo "integer". A Figura 11 mostra que este campo é identificado por um número e não por uma cadeia de caracteres delimitada por aspas. Não há tratamento especial dedicado a este campo.

Outros dois campos exibidos na Figura 12 são 'LOCAL_NAME' e 'LOCAL_ COORDS'. Embora o documento da Figura 11 não trate de nenhuma

localidade geográfica além do local de publicação, é muito comum que trabalhos científicos apontem regiões geográficas como objeto de estudo ou escopo de pesquisa. Nestes casos, é interessante haver campos como os mencionados para registrar tanto o nome da localização quanto suas coordenadas geográficas, que possuem, respectivamente os tipos “string” e “geo_shape”. Este último tipo usa GeoJSON para representar estruturas de dados geográficas. A Figura 14 exibe um exemplo para estes campos.

```
"LOCAL_NAME" : "Brasilia, DF",
"LOCAL_COORDS" : {
      "type" : "polygon",
      "coordinates" : [[
         [-48.0897647,-15.8651711],
         [-47.784951,-15.8651711],
         [-47.784951,-15.5783532],
         [-48.0897647,-15.5783532],
         [-48.0897647,-15.8651711]
      ]]
  }
```

Figura. 14. Exemplo de valores para ‘LOCAL _ NAME’ e ‘LOCAL _ COORDS’.

Além do tipo ‘documento’, também mapeamos ‘glossario’, conforme Figura 15. É essa indexação que permite recuperar definições de termos do glossário utilizado pelo sistema. O campo ‘TERMO’ especifica o termo cuja definição é fornecida e, assim como o campo ‘full’ das palavras-chave dos documentos, é processado com os analisadores ‘index_keywords’ e ‘search_keywords’. ‘DEFINICAO’ já corresponde à própria definição do termo. Em princípio, este campo não é indexado (“index:no”), uma vez que é utilizado somente para retornar seu valor mediante a consulta pelo termo correspondente. Porém, se futuramente alguma funcionalidade exigir uma busca que verifique palavras presentes na definição, este campo também pode ser indexado. A Figura 16 mostra a inclusão em ‘glossario’ de “balanço hídrico” com o identificador ‘1’.

```
PUT /bdpa/_mapping/glossario
{
    "dynamic": "strict",
    "properties": {
      "TERMO":{
          "type": "string",
          "index_analyzer":  "index_keywords",
          "search_analyzer": "search_keywords"
      },
      "DEFINICAO":{
          "type": "string",
          "index": "no"
      }
    }
}
```

Figura. 15. Mapeamento de 'glossario'.

```
PUT /bdpa/glossario/1
{
  "TERMO": "balanço hídrico",
  "DEFINICAO": "Balanço do fluxo de água que entra e sai de um sistema
(solo, rios, lagos, vegetação úmida e oceanos) em um determinado período
de tempo."
}
```

Figura. 16. Exemplo de definição de termo do glossário.

Conclusões

Este trabalho apresenta os analisadores que compõem um SRI de publicações técnico-científicas agropecuárias construído com a tecnologia Elasticsearch. Além disso, mostra o mapeamento de campos presentes em um banco de dados real para os índices do sistema. Os detalhes aqui apresentados não são encontrados em outras publicações na literatura.

Embora tenha apresentado o SRI com Elasticsearch, os analisadores e os mapeamentos utilizados podem ser reproduzidos em ferramentas similares. As especificações fornecidas neste documento podem ser alteradas futuramente, mesmo para aplicações que usam Elasticsearch, uma vez que frequentemente atualizações de ferramentas como esta exigem uma reescrita de códigos. A expectativa, porém, é que alterações significativas não sejam necessárias.

Neste trabalho, não expomos todos os dados referentes ao banco de publicações utilizado, e sim os mais importantes. A cada campo acrescentado na indexação, é necessário estabelecer um mapeamento apropriado e talvez criar novos analisadores, filtros ou *tokenizers*. Tampouco exploramos outros elementos importantes de um SRI, como o ranqueamento dos resultados, *facets* ou o realce nos resultados dos termos utilizados nas consultas. Estas e outras questões serão tratadas em trabalhos futuros.

Referências

BAEZA-YATES, R.; RIBEIRO-NETO, B. **Modern information retrieval**: the concepts and technology behind search. 2nd. Harlow: Addison-Wesley, 2010. 913 p.

DAVIS, M.; IANCU, L. **Unicode text segmentation.** Unicode Standard Annex, v. 29, 2016. Disponível em: <http://www.unicode.org/reports/tr29/>. Acesso em: 12 abr. 2017.

GORMLEY, C.; TONG, Z. **Elasticsearch Reference 2.1.** Mapping. Disponível em: <https://www.elastic.co/guide/en/elasticsearch/reference/2.1/mapping.html>. Acesso em: 12 abr. 2017. 12 abr. 2017.

GORMLEY, C.; TONG, Z. **Elasticsearch:** the definitive guide [2.x]. Standard Tokenizer. 2015. Disponível em: <https://www.elastic.co/guide/en/elasticsearch/guide/2.x/standard-tokenizer.html>. Acesso em: 5 jan. 2017.

ELASTIC. Elasticsearch: RESTful, distributed search & analytics. Disponível em: <https://www.elastic.co/products/elasticsearch>. Acesso em: 1 jun. 2017.

EMBRAPA. Portal da Embrapa. Disponível em: <https://www.embrapa.br/>. Acesso em: 16 fev. 2017.

MARINHO, I. J. P.; CARDONE, H. T. M.; VAZ, G. J. Analisadores complementares para melhorar a qualidade das buscas em sistemas de recuperação de informação. In: MOSTRA DE ESTAGIÁRIOS E BOLSISTAS DA EMBRAPA INFORMÁTICA AGROPECUÁRIA, 8., 2012, Campinas. **Resumos...** Brasília, DF: Embrapa, 2012a. p. 23-26. Disponível em: <http://ainfo.cnptia.embrapa.br/digital/bitstream/item/80005/1/23.pdf>. Acesso em: 12 abr. 2017.

MARINHO, I. J. P.; CARDONE, H. T. M.; VAZ, G. J. Evolução do mecanismo de busca do Ainfo-Consulta com uso de thesaurus agropecuário. In: CONGRESSO INTERINSTITUCIONAL DE INICIAÇÃO CIENTÍFICA, 6., 2012, Jaguariúna. **Anais...** Jaguariúna: Embrapa; ITAL, 2012b. p. 1-9. CIIC 2012. No 12610. Disponível em: <http://ainfo.cnptia.embrapa.br/digital/bitstream/item/65705/1/RE12610.pdf>. Acesso em: 12 abr. 2017.

THE APACHE SOFTWARE FOUNDATION. **Apache Solr.** Disponível em: <http://lucene.apache.org/solr/>. Acesso em: 1º jun. 2017.

VAZ, G. J. **Plugins do Elasticsearch para tratamento de termos compostos e correção do filtro de sinônimos.** Campinas: Embrapa Informática Agropecuária, 2016. 28 p. (Embrapa Informática Agropecuária. Documentos, 144). Disponível em: <http://ainfo.cnptia.embrapa.br/digital/bitstream/item/155704/1/Doc144.pdf>. Acesso em: 1º jun. 2017.

ZANARDO, D. F.; VAZ, G. J. Tratamento eficiente de termos compostos na tecnologia Solr. In: MOSTRA DE ESTAGIÁRIOS E BOLSISTAS DA EMBRAPA INFORMÁTICA AGROPECUÁRIA, 9., 2013, Campinas. **Resumos...** Brasília, DF: Embrapa, 2013. p. 25-27. Disponível em: <http://ainfo.cnptia.embrapa.br/digital/bitstream/item/98960/1/tratamento.pdf>. Acesso em: 12 abr. 2017.

Embrapa

Informática Agropecuária

MINISTÉRIO DA
AGRICULTURA, PECUÁRIA
E ABASTECIMENTO

BRASIL
GOVERNO FEDERAL

CGPE 13527