SafetyRank: conceitos de recuperação de informação aplicados a documentos de alertas de segurança industriais

Wander Fernandes Júnior[1]

[1]Instituto Federal do Espírito Santo (Ifes) – Campus Serra
Rodovia ES-010 Km-6,5 – Manguinhos – CEP: 91.501-970 - Serra – ES – Brasil

wanderfj@gmail.com

Abstract. *Proposal of a computational tool that performs search and rank of public industrial safety alerts using the history of accidents in the oil and gas industry. An algorithm was developed to read the safety alerts records, carry out natural language processing (for tokenization, data cleansing and content lemmatization), perform indexing, database storage, search and rank of results. Search results are ordered based on the similarity with the input data using Okapi BM25 method. Application Programming Interface (API) functionality has been integrated into the algorithm, making it possible to perform requests by other applications. The use of this tool facilitated a way to seek, in a practical and fast way, reports of previous accidents that were relevant to a given particular activity.*

Resumo. *Proposta de uma ferramenta computacional que realiza busca e classificação de alertas públicos de segurança industrial utilizando o histórico de acidentes na indústria de petróleo e gás. Um algoritmo foi desenvolvido para leitura dos registros, processamento de linguagem natural (para tokenização, limpeza de dados e lematização dos conteúdos), indexação e armazenamento em banco de dados, e busca e ranqueamento de resultados. Os resultados da busca são ordenados pela similaridade com os dados de entrada utilizando o método Okapi BM25. Foram integradas funcionalidades de Application Programming Interface (API) ao algoritmo, sendo possível a realização de requisições por outras aplicações. A utilização desta ferramenta facilitou a busca, de forma prática e rápida, de alertas de acidentes anteriores que sejam relevantes a uma determinada atividade.*

1. Introdução

A quantidade de dados gerados diariamente tem crescido de forma significativa, sendo em sua maioria composta por informações não estruturadas que não são facilmente compreendidas e processadas por computadores [Allahyari et al. 2017]. A digitalização do mundo real gera uma grande quantidade de dados que necessitam ser tratados de forma apropriada para que sejam transformados em informações úteis e relevantes [Serrano 2019].

O processamento de linguagem natural (ou *natural language processing* - NLP) é o uso de linguagens humanas, tais como inglês ou francês, por computadores. Como as linguagens naturais normalmente são ambíguas e não obedecem a regras rígidas, existem desafios associados à extração de informações relevantes e classificação de textos

[Goodfellow et al. 2016]. Existe um vasto campo de aplicações de processamento de linguagem natural, tais como: análise de sentimentos, modelagem de tópicos, classificação de documentos, sistemas de recuperação de informações, *chatbots*, detecção e tradução de idiomas, sumarização, geração e previsão de textos [Kulkarni and Shivananda 2019].

Conforme [Zhai and Massung 2016], um sistema de informações textuais é uma aplicação que ajuda pessoas a acessar e analisar informações relevantes com rapidez e precisão, e que pode consistir de diversos módulos, como ilustrado na Figura 1. Inicialmente, é necessário um módulo de análise de conteúdo baseado em técnicas de processamento de linguagem natural. Este módulo permite ao sistema de informação transformar o texto bruto em representações mais significativas e que possam ser utilizadas de forma mais eficaz por um mecanismo de busca ou por qualquer sistema de análise de textos. A partir deste tratamento de conteúdo, diversas aplicações podem ser realizadas, tanto no âmbito de recuperação de informação (para acesso à informação) quanto no âmbito de mineração de dados (*data mining*) (para aquisição de conhecimento).

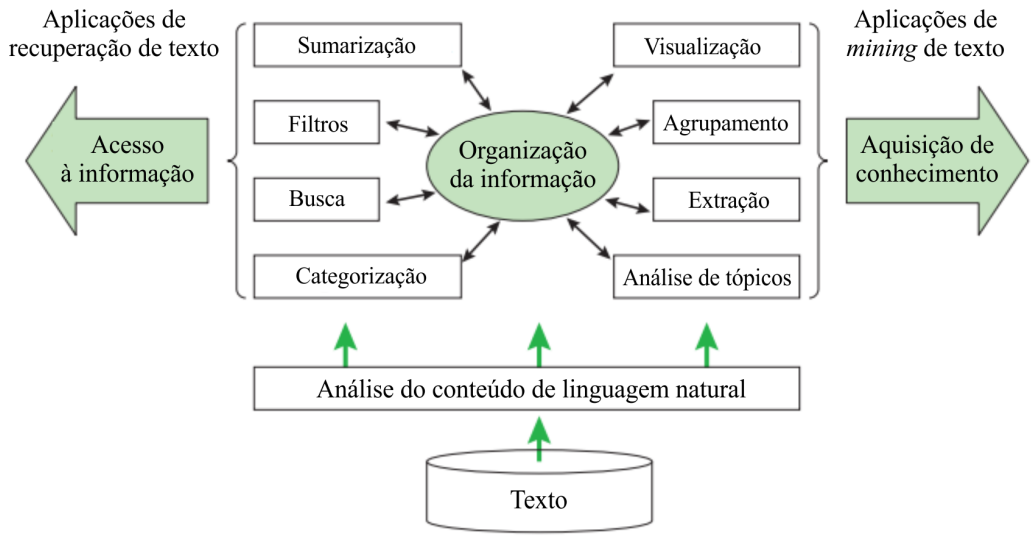

Figure 1. Framework conceitual de sistema de informação de textos traduzido de [Zhai and Massung 2016].

Conforme [Russell and Norvig 2016], recuperação de informação é a tarefa de encontrar documentos relevantes e aplicáveis às necessidades dos usuários. Os melhores exemplos de uso são os buscadores da internet. Um sistema de recuperação de informação é caracterizado por quatro componentes:

- **Documentos**: também chamado de *corpus*, é o conjunto de dados a ser analisado.
- **Consulta**: também chamada de *query*, é a informação fornecida pelo usuário sobre o que necessita ser buscado no conjunto de documentos.
- **Resultados**: subconjunto dos documentos que o sistema de recuperação de informações identificou como relevantes para a consulta de entrada.
- **Apresentação**: é a exibição visual dos resultados ao usuário.

Nas áreas industriais muitos acidentes ocorrem porque os envolvidos não sabem como evitá-los, embora outras pessoas na mesma organização possuam tal conhecimento.

Acidentes podem ocorrer com menor frequência se as lições aprendidas forem divulgadas, discutidas e registradas, para que as ações tomadas no passado estejam acessíveis às pessoas em situações similares [Kletz 1993]. É comum que sejam emitidos alertas de segurança após a ocorrência de acidentes, conforme exemplos ilustrado na Figura 2. Estes documentos de textos contém relatos da ocorrência e lições aprendidas que visam prevenir danos às pessoas e fortalecer a cultura de segurança por meio do compartilhamento de experiências.

Figure 2. Exemplos de textos de alertas de segurança obtidos de [ANP 2019] [CCPS 2019], [US-Coast-Guard 2019], [IADC 2019].

Este trabalho propõe um programa computacional para a realização de busca de alertas públicos de segurança industrial relativos a acidentes na indústria naval e de petróleo e gás. O algoritmo desenvolvido realiza a leitura dos registros de acidentes, o processamento da linguagem natural (para tokenização, limpeza de dados e lematização dos conteúdos), a indexação e armazenamento em banco de dados, além da busca por aler-

tas e ranqueamento de resultados. Os resultados da busca são ordenados pela similaridade com os dados de entrada utilizando a função Okapi BM25.

2. Desenvolvimento

A elaboração deste trabalho seguiu as seguintes etapas: coleta de dados, extração de textos e tabulação em base de dados, aplicação das técnicas de pré-processamento tokenização, remoção de *stop words* e lematização dos documentos, elaboração de consulta para busca e ordenação nos respectivos documentos aplicando a técnica Okapi BM25 e elaboração de uma interface para exibição dos documentos recuperados.

2.1. Coleta de dados

A coleta da dados é a fase inicial, na qual é realizado o levantamento dos dados de texto a serem utilizados. Os conjuntos de dados em geral contém um sequência de textos em documentos $D = \{D_1, D_2, ..., D_n\}$ onde D_i refere-se a um documento com s sentenças, w_s palavras [Kowsari et al. 2019]. Foram utilizados 1.500 documentos de alertas de segurança industriais em inglês obtidos em formato PDF (*Portable Document Format*) das seguintes instituições: *Center for Chemical Process Safety* (CCPS), *United States Coast Guard*, *Bureau of Safety and Environmental Enforcement* (BSEE), *National Offshore Petroleum Safety and Environmental Management Authority* (NOPSEMA), *International Association of Drilling Contractors* (IADC) e Agência Nacional do Petróleo, Gás Natural e Biocombustíveis (ANP).

2.2. Extração de textos e tabulação em base de dados

A partir dos documentos coletados, foi realizada a extração dos conteúdos de texto bem como sua tabulação em base dados. Foi utilizado o módulo FTS5 (*Full Text Search*, que é uma tabela virtual do banco de dados SQLite que permite a realização de buscas *full-text* nas bases. Este tipo de mecanismo permite a busca em um grande conjunto de documentos.

2.3. Pré-processamento

Paralelamente à etapa de extração de textos e tabulação em base de dados, foi realizada um pré-processamento nos textos utilizando a linguagem de programação Python e a biblioteca NLTK (*Natural Language Toolkit*). Esta etapa refere-se a uma preparação inicial para facilitar as buscas na ferramenta. Nestas etapas incluíram-se a tokenização, remoções de *stop words* e lematização/*stemming* [Allahyari et al. 2017].

Tokenização é tarefa de dividir o texto (palavras ou frases) em partes chamadas *tokens*. A remoção de *stop words* é um tipo de filtro no qual são removidas as palavras que aparecem frequentemente no texto mas em geral não tem muita informação sobre conteúdo (por exemplo, preposições e conjunções). A lematização e *stemming* são técnicas semelhantes nas quais as palavras são reduzidas aos seus lemas (lematização) ou às suas raízes (*stemming*).

Na ferramenta desenvolvida foram aplicadas as etapas de tokenização, remoção de *stop words* e *stemming*.

2.4. Busca e ordenação utilizando Okapi BM25

A maioria dos sistemas de recuperação de informação são baseados em estatísticas de contagem de palavras (*bag of words*). Uma função de ranqueamento recebe um documento e uma consulta e retorna um indicador numérico, sendo que os documentos mais relevantes recebem indicadores com valores mais altos. Dentre as funções de ranqueamento, existe a BM25, que foi desenvolvida no projeto Okapi na *London's City College* e tem sido usado por mecanismos de busca, tais como o projeto *open-source* Lucene. Na função BM25, este indicador é uma combinação linear dos indicadores de cada palavra que compõe a consulta e leva em consideração três fatores: o primeiro é a frequência com a qual uma palavra aparece em determinado documento (TF ou *Term Frequency*); o segundo é a frequência invertida dos documentos (IDF ou *Inverse Documento Frequency*), que representa a contagem relativa de documentos que contém determinada palavra; e o terceiro é o tamanho total de cada documento [Russell and Norvig 2016].

A implementação utilizada na ferramenta proposta utilizou a abordagem de [SQLite 2019], na qual a função BM25, representada na Equação 1, que retorna um valor real indicando a similaridade entre os resultados dos documentos D e a *query* de entrada Q. Para calcular este valor, a função divide a *query* em componentes e, quanto melhor a similaridade, menor o valor retornado pela função.

$$BM25(D,Q) = \sum_{i=1}^{Npalavras} IDF(q_i).\frac{f(q_i,D).(k+1)}{f(q_i,D)+k.(1-b+b.\frac{|D|}{mediadl})} \tag{1}$$

Nesta Equação , $Npalavras$ é o número de palavras na *query*, $|D|$ é o número de palavras no documento, e $mediadl$ é a média do número de palavras em todos os documentos da tabela FTS5. O valores k e b são constantes estabelecidas em 1,2 e 0,75 respectivamente.

O termo $IDF(qi)$ representa a frequência de documento invertida da palavra i da *query*. É calculado conforme Equação 2, onde N é o total de documentos da tabela e $n(q_i)$ é o número de linhas que contém pelo menos umas instância da palavra i.

$$IDF(q_i) = log\frac{N-n(q_i)+0.5}{n(q_i)+0.5} \tag{2}$$

O termo $f(q_i, D)$, representado na Equação 3, é a frequência de ocorrência da palavra i, ou seja, o número de ocorrências de determinada palavra em determinado registro. Entretanto, para permitir a possibilidade de inserir pesos diferentes em diferentes colunas dos registros, a função implementada por [SQLite 2019] permite a definição de pesos para as colunas, onde w_c é o peso associado à coluna c e $n(q_i, c)$ é o número de ocorrências da palavra i no respectivo campo.

$$f(q_i,D) = \sum_{c=1}^{Ncoluna} w_c.n(q_i,c) \tag{3}$$

2.5. Elaboração de interface para exibição dos resultados

Para a elaboração da interface da ferramenta foi utilizado o *framework Flask*, que é uma das mais populares implementações de aplicação *web* na linguagem de programação Python [Flask 2019]. Este *framework* permite a construção de interfaces de forma rápida e fácil, bem como suporta o desenvolvimento de aplicações complexas [Grinberg 2018]. Além da interface *web*, uma funcionalidade de *Application Programming Interface* (API) foi integrada à ferramenta, sendo possível a realização de requisições HTTP (*Hypertext Transfer Protocol*) por outras aplicações, a qual as informações são trocadas em formato JSON (*JavaScript Object Notation*).

3. Resultados

A ferramenta desenvolvida foi disponibilizada em uma conta gratuita da plataforma Heroku, que é um serviço de computação em nuvem do tipo PaaS (*Platform as a Service*) no qual é possível realizar o gerenciamento de aplicações *web* [Heroku 2019]. Conforme ilustrado na Figura 3, a ferramenta foi desenvolvida para receber do usuário apenas uma entrada: uma *query* com as palavras a serem buscadas.

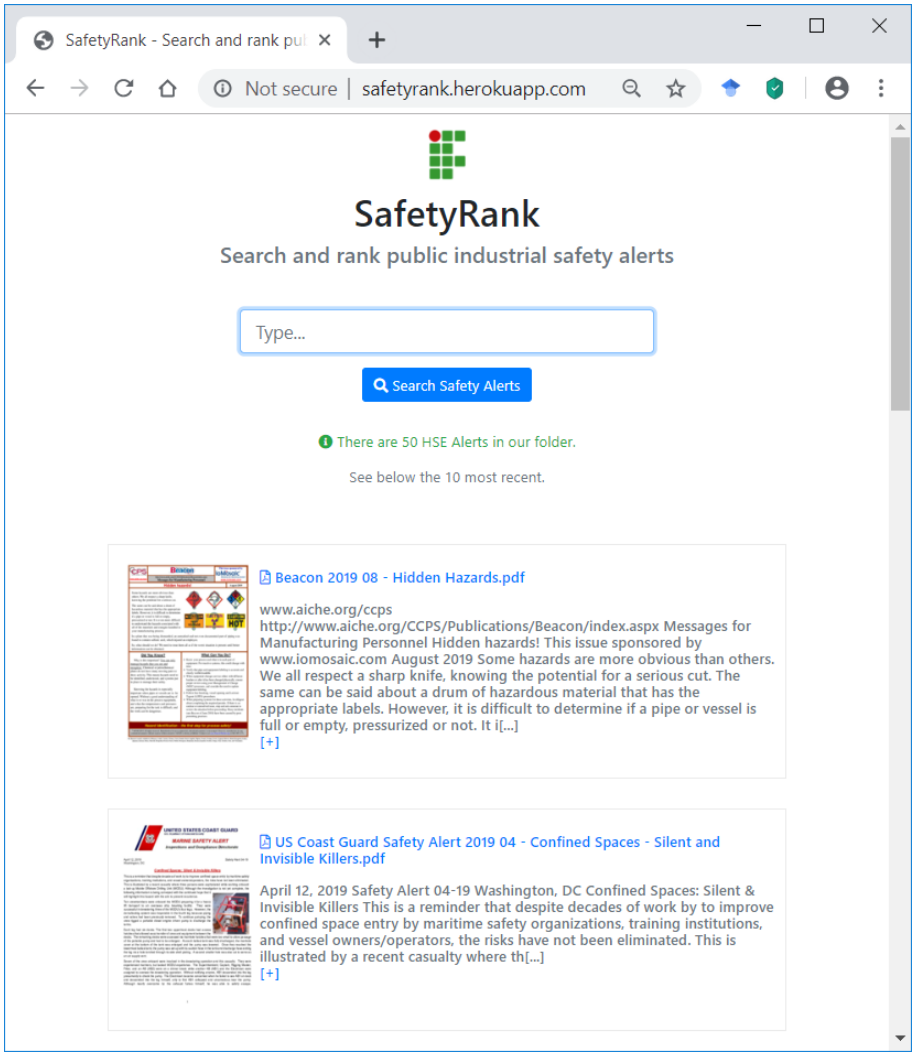

Figure 3. Interface de busca desenvolvida.

Os diversos documentos de alertas de segurança industriais ficam dispersos por diferentes sites de instituições mundiais, e a utilização desta ferramenta facilitou a busca centralizada, de forma prática e rápida, de alertas de acidentes anteriores que sejam relevantes a uma determinada atividade.

4. Conclusões

Neste trabalho foi desenvolvido um programa computacional que realiza busca de alertas públicos de segurança industrial relativos a acidentes na indústria naval e de petróleo e gás. O algoritmo desenvolvido realiza leitura dos registros, processamento de linguagem natural (para tokenização, limpeza de dados e lematização dos conteúdos), indexação e armazenamento em banco de dados, e busca e ranqueamento de resultados. Os resultados da busca são ordenados pela similaridade com os dados de entrada utilizando o método Okapi BM25.

Para implantação das etapas de extração de textos, tabulação e pré-processamento foi utilizada a linguagem de programação Python e a biblioteca NLTK (*Natural Language Toolkit*). O armazenamento foi realizado em banco de dados SQLite, no qual foi utilizada uma consulta ordenada conforme a função BM25. A exibição dos resultados foi realizada por meio de uma página *web* utilizando o framework *Flask*.

Foram integradas funcionalidades de *Application Programming Interface* (API) ao algoritmo, sendo possível a realização de requisições por outras aplicações. A utilização desta ferramenta facilitou a busca, de forma prática e rápida, de alertas de acidentes anteriores que sejam relevantes a uma determinada atividade.

References

Allahyari, M., Pouriyeh, S., Assefi, M., Safaei, S., Trippe, E. D., Gutierrez, J. B., and Kochut, K. (2017). A brief survey of text mining: Classification, clustering and extraction techniques. *arXiv preprint arXiv:1707.02919*.

ANP (2019). Alertas de segurança. [Web; acessado em 18-05-2019].

CCPS (2019). Process safety beacon. [Web; acessado em 18-05-2019].

Flask (2019). Flask. [Web; acessado em 21-08-2019].

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep learning*. MIT Press.

Grinberg, M. (2018). *Flask web development: developing web applications with python*. " O'Reilly Media, Inc.".

Heroku (2019). What is heroku? [Web; acessado em 17-08-2019].

IADC (2019). Safety alerts. [Web; acessado em 18-05-2019].

Kletz, T. A. (1993). *Lessons from disaster: how organizations have no memory and accidents recur*. IChemE.

Kowsari, K., Jafari Meimandi, K., Heidarysafa, M., Mendu, S., Barnes, L., and Brown, D. (2019). Text classification algorithms: A survey. *Information*, 10(4):150.

Kulkarni, A. and Shivananda, A. (2019). *Natural Language Processing Recipes*. Springer.

Russell, S. J. and Norvig, P. (2016). *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited,.

Serrano, W. (2019). Neural networks in big data and web search. *Data*, 4(1):7.

SQLite (2019). Sqlite fts5 extension. [Web; acessado em 17-08-2019].

US-Coast-Guard (2019). Safety alerts. [Web; acessado em 18-05-2019].

Zhai, C. and Massung, S. (2016). *Text data management and analysis: a practical introduction to information retrieval and text mining*. Morgan & Claypool.