Matheus de Araújo Nogueira

# Buscador aLine

Vitória, ES

2018

Matheus de Araújo Nogueira

# Buscador aLine

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Informática da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do Grau de Mestre em Informática.

Universidade Federal do Espírito Santo – UFES

Centro Tecnológico

Programa de Pós-Graduação em Informática

Orientador: Prof. Dr. Elias Silva de Oliveira

Vitória, ES

2018

Ficha catalográfica disponibilizada pelo Sistema Integrado de Bibliotecas - SIBI/UFES e elaborada pelo autor

---

de Araújo Nogueira, Matheus, 1994-
D278b Buscador aLine / Matheus de Araújo Nogueira. - 2018.
74 f. : il.

Orientador: Elias Silva de Oliveira.
Dissertação (Mestrado em Informática) - Universidade Federal do Espírito Santo, Centro Tecnológico.

1. Arquitetura de clusters. 2. Buscadores. 3. Jornal. 4. Inteligência Artificial. 5. Mineração de Dados. I. Silva de Oliveira, Elias. II. Universidade Federal do Espírito Santo. Centro Tecnológico. III. Título.

CDU: 004

---

# *Buscador aLine*

***Matheus de Araujo Nogueira***

Dissertação submetida ao Programa de Pós-Graduação em Informática da Universidade Federal do Espírito Santo como requisito parcial para a obtenção do grau de Mestre em Informática.

Aprovada em 30 de outubro de 2018:

Prof. Dr. Elias Silva de Oliveira
Orientador

Profª. Drª. Claudine Santos Badue Gonçalves
Membro Interno

Prof. Dr. Patrick Marques Ciarelli
Membro Externo

UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
Vitória-ES, 30 de outubro de 2018.

# Agradecimentos

Agradeço aos meus pais, Sergio e Marcia, e aos meus avós Almiro e Iedda, por terem valorizado a educação e o conhecimento para mim. Obrigado por serem um exemplo de bondade e honestidade. Obrigado ao meu pai por me incentivar nos estudos e nas escolhas que fiz. Obrigado à minha mãe por cuidar do meu bem estar, principalmente nas noites em que passava trabalhando. Vocês me acompanharam em cada estágio deste projeto e me deram todas as condições para que eu me transformasse em quem eu sou hoje. Obrigado pela atenção,educação e criação.

Agradeço o meu orientador Prof. Dr. Elias Silva de Oliveira por ter me guiado nesta jornada na pesquisa, pelos puxões de orelha quando não cumpria as tarefas e por me ensinar a ter um olhar mais crítico. Obrigado pelos conselhos e críticas.

Agradeço também aos meus amigos do Laboratório de Computação de Alto Desempenho(LCAD), pela camaradagem e companheirismo. Obrigado pelas conversas e pelos cafés.

Agradecimento especial para os meus amigos Leonardo Muniz de Lima, Marcos Alécio Spalenza e Rodrigo Ferreira Berriel, por compartilharem os seus conhecimentos, conversas no café, conselhos e críticas.

*"Mr. Helpmann, I'm keen to get into Information Retrieval."*
*(Sam Lowry, Brazil)*

# Publicações

Como parte deste trabalho, foram desenvolvidos e publicados os seguintes trabalhos, no período do mestrado, que apresentam, em maior ou menor grau, relação com o tema proposto.

Publicação de trabalhos em anais de congressos:

- Cavaca, A. G.; Antunes, M. N.; Nogueira, M. Comunicação, Informação e Saúde: Estratégia Interdisciplinar para Observar a Saúde em Jornais Digitais. In: *Memorias del XIII Congreso Latinoamericano de Investigadores de la Comunicación | Comunicación y Salud*. Cidade do México, México: ALAIC, 2016. p. 13–20. ISSN 2179-7617.

- Cavaca, A. G. et al. Sistema aLine: Experiência Interdisciplinar para Observar a Saúde em Jornais Digitais. In: *7º Congresso Brasileiro de Ciências Sociais e Humanas em Saúde*. Universidade Federal do Mato Grosso, Cuiabá, Mato Grosso, Brasil: ABRASCO, 2016. p. 147.

- Nogueira, M. de A.; Oliveira, E. de. Estratégias de Correção de Erros de Extratores de Palavras em Português. In: *Proceedings 5nd Symposium on knowledge Discovery, mining and learning(KDMile)*. Uberlândia, Minas Gerais: Faculdade de Computação – FACOM-UFU, 2017. p. 145–152.

- Pirovani, J.; Nogueira, M.; Oliveira, E. de. Indexing Names of Persons in a Newspaper Large Dataset. *Computational Processing of the Portuguese Language*, Springer, 2018.

# Resumo

O uso de buscadores é essencial nos dias de hoje, pois o imenso "mar de informação" que temos e geramos todos os dias impossibilita a realização de buscas rápidas sem essas ferramentas. Atualmente, existem vários tipos de buscadores, com especializações e características próprias, como buscadores para vídeos (*YouTube*), música (*qwant*), imagens (*Google Imagens*), áudios e textos. Mesmo que eles sejam comumente associados apenas à páginas *web*, independentemente dos tipos de acervos indexados, todos seguem a mesma arquitetura.

Muitos dos buscadores disponíveis trabalham com documentos publicados na internet. Um grande volume desses documentos, no entanto, não está indexado pelos buscadores, por não seguir os padrões que permitem a indexação – por exemplo, quando não estão vinculados a um *hyperlink* ou requisitam alguma interação com o usuário. Por isso, tais documentos ficam fora do alcance dos buscadores. Este é o caso dos jornais *A Tribuna* (jornal de abrangência regional no Espírito Santo) e *Metro* (jornal de abrangência nacional), os quais possuem em seus *sites* versões digitais das edições impressas.

Este trabalho descreve a arquitetura do aLine, buscador que foi criado para indexar as bases desses dois jornais, bem como a forma como seus módulos foram idealizados para trabalhar com esses acervos, as decisões tomadas para resolver problemas na extração da informação, experimentos para avaliar os documentos recuperados e o tempo médio de busca. O aLine, além de possibilitar a busca nos dois acervos, jornais *A Tribuna* e *Metro*, ajudou a equipe do Observatório Saúde na Mídia - Regional ES a realizar buscas mais eficientes e eficazes no acervo do jornal *A Tribuna*.

**Palavras-chave**: Arquitetura de *clusters*, buscadores, jornal, inteligência artificial e mineração de dados.

# Abstract

The use of search engines is essential nowadays, because the immense sea of information that we have and generate every day makes it impossible to carry out quick searches without these tools.Currently, there are several types of search engines, with specializations and their own characteristics such as search engines for videos( *YouTube*), music(*qwant*), imagens(*Google Images*), audio and texts. Even though they are only associated with web pages, regardless of the types of index collections, they all generally follow the same architecture.

Many of the available search engines work with documents published on the internet. A large volume of these documents, however, is not indexed by search engines, for not following the standards that allow indexing - for example, when they are not linked to a hyperlink or require some interaction with the user. Therefore, such documents are beyond the reach of search engines. This is the case of the newspapers (A Tribuna) (local newspaper of Espírito Santo) and *Metro* (newspaper of national scope), which they have in their sites digital versions of the printed editions.

This work describes the architecture of aLine, a search engine that was created to index the bases of these two newspapers, as well as the way the modules were designed to work with these collections, the decisions taken to solve problems in information extraction, experiments to evaluate the retrieved documents and the average search time. The aLine, in addition to making it possible to search the two collections, newspapers and publications, helped the team of the Observatório Saúde em Mídia - Regional ES to carry out more efficient and effective searches in the collection of the newspaper *A Tribuna*.

**Keywords**: Clusters Architecture, search engines, newspaper, artificial intelligence and mining data.

# Lista de ilustrações

# Lista de tabelas

# Sumário

# 1 Introdução

Com o avanço da tecnologia, cada vez mais nos vemos na dependência do computador para se realizar tarefas na vida cotidiana, principalmente quando elas estão relacionadas à informação. A internet possibilitou criar, ler e compartilhar qualquer tipo de informação, nos formatos textual, visual ou em áudio. É necessário, porém, organizar, classificar e indexar esses dados, a fim de possibilitar a recuperação dessas informações. Assim, foram criadas ferramentas para a recuperação de tais informações como os sistemas gerenciadores de documentos. Nestes sistemas gerenciais a indexação de um documento é realizada manualmente nos quais alguns termos são selecionados para indexação de um documento. Esse tipo de abordagem limita o usuário a utilizar um vocabulário restrito como chave de busca, assim reduzindo o escopo de termos que deseja buscar. Isso ocorre pois há limitações na estratégia de indexação, por exemplo, indexar metadados ou termos pré-determinados. Esta abordagem de indexação, em muitos casos, não retorna os documentos mais relevantes ao usuário e limita as buscas aos conteúdos manualmente registrados. Por exemplo, se assumirmos um acervo de jornal para realizarmos buscas, no qual os campos de indexação são: título da reportagem, autor, dia de publicação e página, as buscas estão limitadas a esses campos. Ou seja, se buscarmos por um termo que ocorre no corpo da reportagem, o sistema de busca não encontrará os documentos que contém o termo buscado.

A fim de resolver essas limitações, foram criados os buscadores. Os buscadores são ferramentas da área de Recuperação de Informação para coleções de textos em larga escala. Os buscadores trouxeram grandes avanços nas tarefas de busca e recuperação da informação, sendo praticamente indispensáveis para as atividades do dia a dia, sem eles seria impossível encontrar algum documento no montante de informações que criamos diariamente. Existem vários tipos de buscadores, baseados no domínio com que trabalham, tais como documentos, vídeos, áudio e imagens.

Geralmente, os buscadores são associados apenas à documentos web, como é o caso de alguns dos mais conhecidos (*Google*[1], *Bing*[2], *Yahoo!*[3], *Hao123*[4], *Baidu*[5]). Sendo o *Google* o buscador mais famoso e com maior parcela de usuários global, seguido pelo do *Bing* e do *Yahoo!*. Os buscadores *Hao123* e *Baidu*, são buscadores com maior presença na China em especial o *Baidu*, pois seu *site* e conteúdo está disponível apenas em chinês, entretanto o *Hao123*, possui uma interface para várias línguas porém utiliza o *Baidu* como motor de busca.

[1] <https://www.google.com>
[2] <https://www.bing.com/>
[3] <https://yahoo.com>
[4] <https://www.hao123.com/>
[5] <https://www.baidu.com/>

Em buscadores de vídeos como o *Youtube*[6], os dados de indexação utilizados são os metadados que o usuário insere nos vídeos, como título e descrição. Nos buscadores de áudio como o *Qwant Music*[7], um buscador especializado em músicas, os dados indexados são os títulos da música, letras, artistas, álbuns e ano de publicação. Por fim, os buscadores de imagens como o *Google Imagens*[8], utiliza termos presentes nos documentos em que as imagens estão contidas, aplicando modelos de classificação e detecção de objetos nas próprias imagens. Sendo possível que em alguns casos do *Google Imagens*, ao ser solicitado uma busca a qual a entrada da busca é uma outra imagem, o buscador retorna a classe da imagens e *links* para imagens semelhantes a buscada.

Sem os buscadores, indexar e realizar buscas nos modelos dos outros sistemas de recuperação de informação, seria impossível, pois, no caso de documentos web, todas páginas web deveriam ser indexadas manualmente. Entretanto, nos buscadores a indexação dos termos é automática, assim que um novo documento ou objeto é disponibilizado para os buscadores, ele é automaticamente indexado. Essa indexação pode abranger todo o conteúdo textual existente nos documentos, entretanto há buscadores que filtram alguns trechos dos conteúdo, pois não são de interesse para as buscas. Esse modo de operação é valido para qualquer tipo de objeto de indexação, com algumas diferenças entres os buscadores. Além disso, os buscadores possuem ferramentas que podem estender as chaves de busca com outras chaves relacionadas, com a intenção de ampliar os resultados e procurar entender a real intenção do usuário em relação à busca.

Contudo, mesmo havendo muitos buscadores disponíveis, em função da forma de disponibilização, uma grande parcela do conteúdo que criamos não é indexada por eles, o que impede que tais objetos sejam encontrados. Isso acontece, pois o método com qual esses buscadores utilizam para indexar os documentos não abrange alguns modelos de disponibilização dos mesmos. Dentro desses modelos de disponibilização estão: o uso de interações humana nas páginas web, por exemplo, a autenticação de um usuário ou definição dos parâmetros de acesso aos documentos, como os *sites* dos jornais *A Gazeta* e *Metro*. Onde ambos os jornais possuem uma versão digital do jornal impresso, no qual o primeiro requer a autenticação dos assinantes do jornal para a exibição da edição digital e o segundo requer a definição de parâmetros como local e dia da publicação. Com isso, esses jornais não são indexados pelos grandes buscadores, dentro deste mesmo escopo, documentos como relatórios gerados por uma empresa, também não são indexados por esses buscadores, pois em sua maioria estão isolados em uma rede interna(intranet) ou requerem um acesso autenticado.

Esses tipos de documentos possuem uma grande riqueza de dados, no geral se apenas a busca por esses documentos estivesse disponível já seria um grande avanço,

[6] <https://www.youtube.com>
[7] <https://www.qwant.com/music/home/overview>
[8] <https://www.google.com/imghp>

pois em muitas tarefas do dia a dia, o simples encontrar os documentos que possuem os termos de interesse, diminuiria o esforço do usuário. Contudo, se também for realizado processamentos nesses documentos, é possível encontrar relações e respostas para algumas questões. Por exemplo, se há alguma relação da ocorrência de um ou mais termos ao longo do tempo. Essa questão pode ser convertida para uma mais prática como : "a palavra dengue ocorrem com qual frequência ao longo do ano?". Ou, quais termos estão relacionados a um termo alvo.

Outro tipo de informação relevante que pode ser extraída destes documentos, é o reconhecimento de entidades nomeadas, como nomes de pessoas, instituições ou locais. O problema de reconhecimento de entidades nomeadas(*Named Entity Recognition* - NER) é muito interessante pois não é uma tarefa simples e possui um grande valor de recuperação, sendo fundamental no processo de processamento em Recuperação da Informação (AGGARWAL; ZHAI, 2012). Essa classe de problema, nos permite relacionar pessoas à eventos ou locais, e como foi citado anteriormente, também é possível saber a distribuição da citação de uma ou várias pessoas ao longo do tempo.

## 1.1 Motivação

Coletar, extrair e indexar esses tipos de documentos é um desafio, pois como dito anteriormente, uma grande parcela desses documentos não estão disponíveis nos modelos de indexação dos grandes buscadores. Portanto, sendo necessário uma aplicação personalizada para realizar a coleta, extração e por fim a indexação. No domínio de jornais, podemos encontrar até problemas mais complexos, como determinar qual página é mais relevante ou até mesmo como segmentar as notícias em uma página. Este trabalho se motiva em permitir buscas em acervos de jornais, os quais possuem uma demanda constante de busca e uma grande riqueza de conteúdo, como identificação de nomes, análise temporal de termos e imagens. Para realização destas buscas, criamos o buscador aLine.

O aLine é um buscador focado no domínio de jornais, no qual é o primeiro buscador desenvolvido no Programa de Pós-graduação em Informática na Universidade Federal do Espírito Santo. Atualmente indexa duas bases de jornais, o jornal local do Espírito Santo *A Tribuna*, e o jornal *Metro*, com edição local também do Espírito Santo. Esses jornais possuem uma versão digital do jornal físico e não é indexado pelos grandes buscadores. No acervo do jornal *A Tribuna*, o aLine ajudou a equipe do Observatório Saúde na Mídia - Regional ES a realizar buscas mais eficientes e rápidas. Sendo utilizado nos estudos de Cavaca, Antunes e Nogueira (2016) e Cavaca et al. (2016), auxiliando no trabalho de buscas do acervo por eles utilizado. Os tais trabalhos, inicialmente, utilizaram a estratégia de busca das palavras-chave, lendo cada página do acervo do jornal *A Tribuna*. Essa Estratégia previa um gasto de 5799 horas de trabalho no acervo do jornal *A Tribuna*.

Entretanto, com a ajuda o aLine, esse tempo foi reduzido para 32 horas de trabalho.

Além disso, o aLine foi desenvolvido para servir de base para outros tipos de processamentos que podemos realizar sobre documentos, como classificação e clusterização. A criação do buscador aLine, envolveu implementar todos os módulos básicos que compõe um buscador (Crof; Metzler; Strohman, 2015), com algumas modificações devido as restrições. Tais restrições se deram ao tipo de documento, em que partiu desde a coleta dos documentos, pois como dito anteriormente, tais documentos não são disponibilizados de uma maneira fácil para a coleta automática. A extração do conteúdo, também teve que ser adaptada, pois muitos ruídos foram encontrados na extração dos textos, no qual resultou no trabalho de correção desses textos (Nogueira; Oliveira, 2017). Por fim, houve adaptações nos processamentos dos textos e indexação.

## 1.2 Objetivos

O objetivo deste trabalho é desenvolver um buscador capaz de trabalhar com documentos que não são indexados pelos grandes buscadores, que no caso são os jornais *A Tribuna* e *Metro* que possuem uma versão digital da versão impressa. Além disso, propor uma plataforma de processamento para tarefas de classificação e clusterização, de forma que se apliquem diretamente com as tarefas do buscador.

Por fim, servir de referências para trabalhos que abrangem busca e análise de documentos, visando a redução do esforço dos usuários e permitindo um acesso mais fácil a documentos com grande valor de recuperação.

## 1.3 Método de pesquisa

Em busca por trabalhos que abordassem a arquitetura de buscadores, foi realizada a revisão da literatura buscando descrição de arquiteturas, algoritmos e abordagens para a construção de um buscador.

Ao definir a arquitetura do buscador e quais funções ele terá, foi realizada uma análise no acervo e no recurso computacional a fim de realizar a melhor implementação para a construção do buscador. Com a análise do acervo, foi necessário adaptar alguns módulos com o objetivo de permitir a busca nos acervos dos jornais *A Tribuna* e *Metro*. Para validar o buscador, realizou-se experimentos medindo o tempo de processamento das buscas e a qualidade dos documentos recuperados.

## 1.4 Estrutura da Dissertação

Este trabalho está dividido na seguinte ordem, no Capítulo 2 apresentaremos a arquitetura dos buscadores em trabalhos apresentados na literatura, os quais apresentam os módulos que compõem os buscadores e apontam problemas existentes na arquitetura e como foram solucionados.

No Capítulo 3 apresentaremos a arquitetura do aLine, como os seus módulo funcionam, quais modificações nos módulos foram necessárias para viabilizar a busca nos acervos de jornais digitais, problemas foram encontrados durante o seu desenvolvimento e seus serviços de análise temporal e busca por nomes. Continuando, no Capítulo 4 serão abordados dois experimentos que validam o tempo e da qualidade das buscas do aLine, no primeiro uma análise no tempo de processamento de buscas no aLine e no segundo, a qualidade dos documentos recuperados nas buscas.

No Capítulo 5 serão tecidos comentários em relação ao trabalho e conclusões. Por fim, no Capítulo 6, serão apresentadas sugestões de melhorias no buscador aLine e novos horizontes de tipos de acervos e serviços.

# 2 Arquitetura dos buscadores

Crof, Metzler e Strohman (2015) apontam que há quatro tipos de buscadores: *web search*, *vertical search*, *enterprise search* e *peer-to-peer search*. O *web search* é especializado em documentos *web*, sendo, por isso, o buscador mais comum.

O *vertical search*, por sua vez, é uma especialização do *web search*, com um domínio restrito ou particular, por exemplo, os buscadores *Tumba* (Gomes; Silva, 2001), especializado em *sites* em português de Portugal, e *TodoBR*, especializado em *sites* do Brasil.

O terceiro tipo, *enterprise search*, busca a informação em uma variedade de computadores conectados em uma rede privada, como uma *intranet* empresarial, em busca de informações em arquivos ou e-mails, sendo o *Google Enterprise Search* (Google, 2018) um exemplo deste buscador.

O *desktop search* é uma versão pessoal do *enterprise search*, para buscas em que a informação desejada está em arquivos, *e-mails* ou páginas *web* armazenados no computador local, como o *Windows Search* da *Microsoft* (Microsoft, 2018).

Por fim, o *peer-to-peer search* é um buscador que procura informações em uma rede de *nós* ou computadores sem um controle central, como o buscador YaCy (YaCy, 2018).

Independentemente do tipo, a arquitetura dos buscadores é modelada com o objetivo de se resolver dois objetivos/requerimentos, os quais são: a eficácia e eficiência. A eficácia de um buscador representa a qualidade das buscas, no qual é desejado que o buscador seja capaz de recuperar o conjunto mais relevante de documentos possíveis para uma busca. E a eficiência de um buscador representa a velocidade da busca, a qual se deseja processar as buscas dos usuários o mais rápido possível (Crof; Metzler; Strohman, 2015). Portanto, com esses dois requisitos, no geral, os buscadores são compostos por dois módulos básicos: processamento de índice e processamento de *query*. O primeiro é responsável por possibilitar a busca em um acervo, enquanto o processamento de *query* possibilita ao usuário o acesso ao buscador.

## 2.1 Processamento de Índice

O processamento de índice possui três módulos para possibilitar a busca em um acervo, são eles: a aquisição de texto, transformação de texto e criação do índice. Essas funções estão interligadas, formando um fluxo do processo de indexação, como mostra a Figura 1.

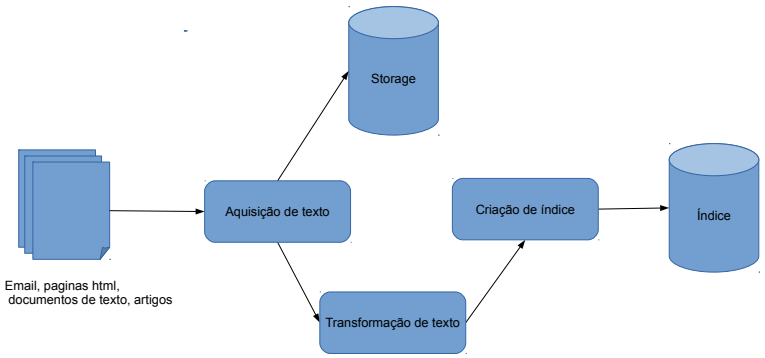



Figura 1 – Funções do *indexing process*.

### 2.1.1 Aquisição de texto

Um dos maiores desafios encontrados pelos buscadores é manter o índice atualizado, procurando nos domínios indexados atualizações ou novas páginas. Porém, esse trabalho é muito custoso, tanto para o buscador quanto para quem provê as páginas.

Para resolver essas tarefas, foram criados os coletores(*crawlers*). A principal função exercida por essas ferramentas é descobrir e coletar documentos, assim mantendo atualizado o índice do buscador. Existem vários tipos de coletores, entretanto, o mais comum é para *web*. Os coletores para *web* são projetados para encontrar *hyperlinks* nas páginas *web*, a fim de descobrir e coletar novas páginas. Além disso, os coletores são capazes de trabalhar com grandes volumes de páginas e, em casos mais avançados, criar regras para acessar alguns domínios que não são atualizados com frequência ou possuem baixa disponibilidade de banda para serem acessados por eles. Dependendo do tipo e domínio do buscador, o coletor pode ser especializado em apenas um domínio ou tipo de documentos.

As regras criada para os coletores retornar a esses domínios, implicam em diminuir o consumo de boa parte do tráfego de internet e encontrar os padrões de frequência com que esses domínios são atualizados. Essa foi a estratégia adotada pelo *Google* na criação

de seu coletor, conforme Brin e Page (1998), os quais tiveram problemas para indexar alguns domínios que eram protegidos ou *sites*, como os de jogos, em que o conteúdo era considerado como lixo para o buscador.

Outro problema também recorrente relacionado a coletores é a detecção de duplicatas na base. Tal situação foi abordada no trabalho de Manku, Jain e Das Sarma (2007), a partir de casos em que era muito pequena a diferença entre o documento já indexado com um novo documento, causada por contadores de páginas, propagandas e diferença de fusos horários.

Tal problema ocupava o coletor com páginas já indexadas e, por consequência, espaço em disco para fazer o processamento de *sites* já indexados. A solução proposta segmentava as páginas em blocos menores e gerava uma "impressão digital" para eles, diminuindo, assim, o retrabalho sobre tal conteúdo.

Outros trabalhos desenvolveram coletores específicos para um domínio, como o Tarântula (Gomes; Silva, 2001), especializado em buscar páginas que estão no domínio .PT, ou seja, páginas em português de Portugal.

Assim que um documento é encontrado, seu conteúdo é extraído para ser indexado. Essa extração depende diretamente do domínio em que o buscador atua: para documentos de texto, pode ser feita a leitura de metadados ou o reconhecimento de caracteres para, então, se extrair seu conteúdo. Em outros tipos de domínios de acervo, há outras informações de interesse que serão extraídas. Por exemplo, em uma partitura musical, podem ser extraídas as notas que a compõem.

### 2.1.2 Transformação de texto

Dando continuidade ao fluxo de processamento do índice, a transformação do texto realiza a conversão do documento em *tokens*, reconhecendo as classes ou contando a frequência dos termos encontrados. O reconhecimento das classes, no caso de documento em *HyperText Markup Language* (HTML), é capaz de encontrar *hyperlinks* para outros documentos, um dos elementos da linguagem de marcação, o qual impacta no coletor e alguns modelos de ranqueamento. Para outros casos, como documentos de texto puro, podem ser encontradas informações como data, hora, número de matrícula e nome de pessoas.

Realizada a classificação, os termos já podem ser indexados, mas, para alguns tipos de buscadores, há técnicas que refinam esse processo. Entre elas, estão as técnicas de radicalização (*stemming*), lematização (*lemmatization*) e remoção de palavras que não possuem valor semântico (*stopwords*). A radicalização dos termos consiste em normalizar as flexões dos termos encontrados em um acervo para as suas versões radical, através da remoção do afixo (Baeza-Yates; Ribeiro-Neto, 2011), por exemplo, os termos "conectado"

e “conectando” são convertidos para o seu radical “conect”. Portanto, esta técnica melhora a performance da recuperação da informação por causa da redução das variantes com a mesma raiz para o mesmo conceito, porém unifica termos que possuem semântica levemente diferentes em um único termo. Além disso, a técnica de *stemming* requer abordagens diferentes para cada língua, como é mostrados nos trabalhos de Soares, Prati e Monard (2009) e Braschler e Ripplinger (2004). No qual o primeiro trabalho focou em desenvolver uma ferramenta para a língua portuguesa e o segundo trabalho focou para a língua alemã.

A técnica de lematização é semelhante ao *stemming*, porém ao invés de fazer a normalização dos termos removendo o afixo, a lematização promove a normalização através da forma canônica(lema), por exemplo, os verbos são convertidos para a forma infinitiva, e os adjetivos e substantivos, para masculino singular, caso exista (Guimarães; MEIROSE; MORAES, 2015). Por exemplo, “conectado” e “conectando” seriam convertidos para “conectar”. Da mesma forma que o *stemming*, a lematização unifica os termos com semântica semelhante para apenas um termo, assim reduzindo a variabilidade de termos e descaracterizando os documentos.

A remoção de *stopwords* consiste em remover os termos que não possuem valor semântico, como artigos, preposições e conjunções (Baeza-Yates; Ribeiro-Neto, 2011). Essa eliminação agrega benefícios à redução de elementos indexados, assim, aplicando uma compressão no índice de palavras. Entretanto, a remoção desses termos implica na redução dos documentos retornados, em vista a buscadores. Pois se considerarmos a seguinte frase em inglês “to be or not to be”. A eliminação das *stopwords* pode reduzir a frase apenas ao termo “be”, assim sendo impossível reconhecer o documento que contenha a célebre frase de *Willian Shakespaere*.

Em razão dos produtos gerados pelas técnicas de *stemming*, lematização e remoção de *stopwords*, a maioria dos buscadores adotam a indexação de todo o conteúdo de um documento, para não descaracterizá-los.

### 2.1.3 Criação de índice

A criação do índice é o último módulo do processamento de índice, com a função de criar o índice de termos encontrados no acervo.

Para se criar um índice é necessário realizar alguns passo, como coleta de dados dos documentos, ponderação e por fim a indexação. A coleta de dados dos documentos realiza uma contagem dos termos presentes, as posições desses termos presentes nos documentos, a contagem das frequências que estes termos aparecem no documento e o comprimento dos documentos em relação ao número de termos encontrados. Esses dados são armazenados para serem utilizados nos próximos passos.

A ponderação nos documentos reflete a importância dos termos e é utilizada no

sistema de ranqueamento. O modelo de ponderação depende diretamente do modelo de recuperação adotado. Entretanto, esses modelos se baseiam na frequência com que os termos ocorrem nos documentos (*Term Frequency - tf*) (Salton; Wong; Yang, 1975) e na frequência que os termos ocorrem no acervo (*Inverse Document Frequency - idf*) (Robertson, 2004). O modelo de *idf* aplica valores maiores a termos que ocorrem em poucos documentos, sendo demonstrada pela Equação 2.1. Sendo $N$ o número total de documentos indexados e $n$ o número de documentos que contém um termo em particular.

$$idf_i = \log \frac{|N|}{|n_i|} \tag{2.1}$$

Por fim seguimos para o último passo, a indexação dos termos, o qual é o centro do processamento de índice. Em buscadores é utilizado o índice invertido para organizar os termos encontrados no acervo. O índice invertido é uma estrutura de dados que associa a um termo, uma lista de documentos que contém este termo. Com isso, o buscador pode realizar buscas mais rápida e eficientes, com o uso de tabela *hash* ou árvore binária(*B-Tree*) (Ziviani et al., 2004).

O *Google*, em seu início, utilizou a tabela *hash* em memória para identificar as palavras indexadas. Esse modelo conseguiu responder uma *query* não buscada anteriormente em aproximadamente 10 segundos em um acervo de 100 milhões de páginas, como é descrito em Brin e Page (1998).

Em seu trabalho, Botelho e Ziviani (2008) desenvolveram uma função mínima de *hash* perfeita, em que conseguem uma escalabilidade linear e melhor utilização do espaço de memória quando as *hashs* são estáticas, ou seja, não há inserção de novas *hashs* na tabela, em uma base com 1.024 bilhões de URLs(*Uniform Resource Locator*).

O trabalho de Kraska et al. (2017) introduz uma nova abordagem de busca. Utilizando redes neurais profundas, eles denominam este modelo de índices aprendidos. É proposto que um modelo neural pode aprender as *hashs* de pesquisa e prever sua existência na tabela. Tal modelo é comparado com o algoritmo *B-Tree* no quesito otimização para *cache*, conseguindo superar em 70% a velocidade de busca e diminuindo o uso da memória principal.

Entretanto, outro ponto importante para a criação do índice é a sua distribuição entre vários computadores, no qual melhora o desempenho no tempo de resposta. Atualmente, os buscadores são organizados em *clusters*, nos quais um computador é denominado como *broker* e controla outros computadores(*nós*), os quais têm a função de manter o índice (Brin; Page, 1998), (Ribeiro-Neto; Barbosa, 1998), (Badue et al., 2005), (Badue et al., 2010),(Baeza-Yates; Ribeiro-Neto, 2011), (Crof; Metzler; Strohman, 2015) e (Tolosa, 2016).

Vários trabalhos foram desenvolvidos para otimizar o tempo de resposta do *broker*

e os algoritmos de busca. Alguns deles analisaram que pontos desta arquitetura influenciam no tempo de resposta do buscador. O trabalho de Ribeiro-Neto e Barbosa (1998) identifica dois gargalos na arquitetura: o *broker* e a rede local do *cluster*, os quais não mais figuram como impacto em novos trabalhos, pois os avanços nos algoritmos e na tecnologia de rede diminuíram esses gargalos.

O trabalho de Badue et al. (2005), por sua vez, analisou que o *broker* não influencia no tempo de resposta do buscador. Nos testes, o componente que degrada o tempo é o acesso ao disco no qual são armazenados os índices. Estes autores analisaram a mesma arquitetura de Ribeiro-Neto e Barbosa (1998), porém, identificando desequilíbrio entre os *nós* de busca, isto é, alguns *nós* levavam mais tempo de resposta, pois recorriam ao disco principal para recuperar os documentos. Os modelos de balanceamento de índice funcional saíram-se bem em testes artificiais, mas a aplicação desses modelos no mundo real pode não resultar no mesmo efeito, conforme asseveram os autores.

Entretanto, mesmo com o avanço da tecnologia de rede, outros pontos dos buscadores evoluíram devido a busca da máxima eficácia. Entre esses avanços estão os sistemas de *cache*, os quais diminuem o esforço do buscador em encontrar as respostas para buscas realizadas anteriormente, mantendo em uma memória mais rápida resultados de buscas ou os termos buscados (Saraiva et al., 2001), (Baeza-Yates et al., 2007) e (Tolosa, 2016). O modelo de *cache* para buscas, armazena em uma memória mais rápida, as buscas realizadas anteriormente, este modelo evita que os *nós* de processamento recorram aos seus índices para responder as buscas. No modelo de *cache* para termos, o modo de trabalho é semelhante ao modelo anterior, sendo que o objeto para *caching* são os termos buscados. Para ambos os modelos de *cache*, as estratégias de quais objetos serão armazenados e o tamanho de cada são muito importantes, pois elas definirão se o sistema de *cache* é eficiente.

No trabalho de Saraiva et al. (2001) é proposto um modelo de *cache* no qual utiliza os dois modelos citados anteriormente, *cache* em dois níveis. Neste trabalho, o buscador assim que recebe uma *query*, ele a procura no *cache* de buscas realizadas, caso encontre, o resultado é enviado imediatamente. Caso não encontre, é seguido para o *cache* de termos. Se nesta parte do processo, os termos da *query* não forem encontrados, o buscador recorre ao índice de todos os *nós* para responder a busca. Essa estratégia de utilizar o *cache* em dois níveis, diminui o acesso aos índices dos *nós*, reduzindo as operações de acesso ao índice principal e diminuindo o tempo de resposta das buscas.

Em Baeza-Yates et al. (2007) é proposto um algoritmo estático de *cache* para o índice de palavras. Este algoritmo utiliza a frequências dos termos presentes nas *queries* que o buscador registrou sobre a frequência dos termos indexados, denominado como $Q_{TF}D_F$. Essa abordagem é mais simples que o modelo dinâmico, no qual conforme o buscador recebe as requisições de busca, o *cache* é construído. Os autores comparam, os dois modelo

com outros dois modelos de cache o LRU(*Least recently used*) para *cache* dinâmico, o qual descarta os elementos menos utilizados recentemente e LFU(*Least-frequently used*) para *cache* estático, no qual, descarta os elementos menos frequentes. Com o algoritmo $Q_{TF}D_F$ foi obtido um ganho de 12% em comparação ao LRU e LFU, e houve um melhor aproveitamento da banda de rede e melhora no tempo de resposta das buscas.

## 2.2 Processamento de *query*

O processamento de *query* é uma função que permite ao usuário o acesso ao buscador por meio de uma interface. A *query* pode ser tratada de várias formas, como um documento ou como várias outras *querys*.

No geral, esse tipo de processamento trabalha para responder a duas questões: i) "Essa *query* ou as palavras que estou buscando existem na minha base de dados?" e ii) "Quais documentos são mais relevantes para minha busca?".

O processamento de *query* é composto por três módulos: a interface do usuário, o módulo de ranqueamento e o módulo de avaliação, como mostrado na Figura 2.

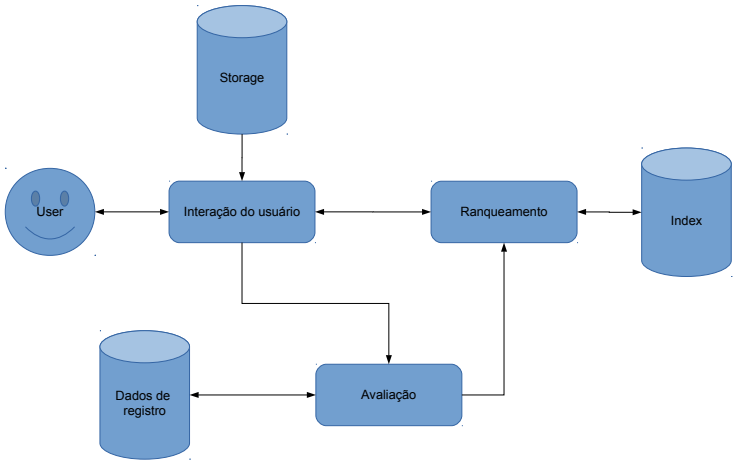



Figura 2 – Funções do *query process*.

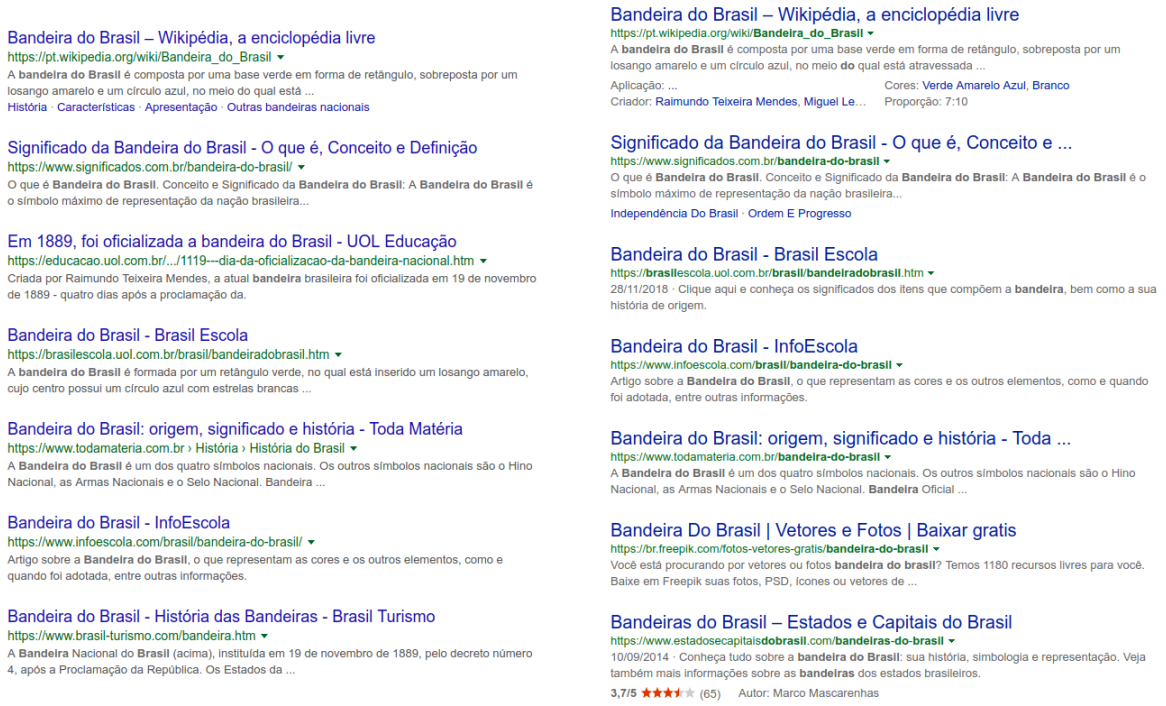
### 2.2.1 *Interface*

A *interface* é um componente muito importante, pois tem a tarefa de receber a *query* do usuário, transformando-a em um índice de termos para que a busca possa ser realizada, usando-se as mesmas técnicas de indexação dos documentos. Além disso, a interface também tem a função de exibir os resultados do buscador, utilizando a lista dos documentos ordenados conforme sua relevância.

Durante a interação, a interface pode coletar dados referentes aos elementos com os quais o usuário está interagindo. Esse tipo de informação pode indicar quais documentos foram mais relevantes, dada uma *query*.

Partindo da inserção da *query* na interface, o processamento analisa sua sintaxe. Em alguns buscadores, é possível utilizar comandos e operadores para refinar a busca, como limitar o domínio de busca, período ou buscar uma frase específica. Essas sintaxes são bem semelhantes entres os buscadores, facilitando, assim, que sejam operadas pelo usuário. Em alguns buscadores, é realizada a correção da *query* ou sugestão de *querys*, tendo-se como base os termos que o usuário informa, incluindo sugestões de expansão, oferecidas a partir dos registros do buscador.

Entretanto, várias técnicas de visualização foram propostas para exibir os resultados de uma busca (Hearst, 2009), mas no geral, os buscadores adotaram o modelo que consiste em uma lista ranqueada com os sumários de cada documento, associados ao *hyperlink* para o documento original. Esse modelo é exemplificado na Figura 3.




(a) Exemplo da *Google*. (b) Exemplo do *Bing*.

Figura 3 – Resultado da busca "bandeira do Brasil" no *Google* e *Bing*.

Como é exibido nas Figura 3a e 3b, ambos os buscadores(*Google* e *Bing*) utilizam o modelo descrito anteriormente, porém, apresentam no resumo do conteúdo do documento, os termos utilizados na busca em negrito. Essa forma facilita o usuário a localizar no texto do resumo os termos buscados.

### 2.2.2 Ranqueamento

O módulo de ranqueamento é considerado o centro do buscador, pois é ele que define se este é ou não bom. O ranqueamento depende das estratégias adotadas e dos módulos que compõem o buscador, como o modelo de recuperação adotada, os *links* que referenciam outros documentos e, ainda, a estrutura que estes possuem. A eficiência de processar muitas *querys* em pouco tempo e a eficácia do modelo de recuperação para encontrar informações relevantes são aspectos que permitem avaliar o ranqueador.

Além disso, buscadores como o *Google* e o *Bing* recomendam o uso de boas práticas na estrutura do *site*, tais como as técnicas de *Search Engine Optimization* (SEO). Essas técnicas incluem acessibilidade do conteúdo e adaptabilidade de *layout* para dispositivos móveis, permitindo melhor avaliação do *site*.

Contudo, as características e os pesos extraídos dos documentos indexados e das *querys* utilizadas nos algoritmos de ranqueamento devem ter similaridade com os tópicos buscados e relevância para os usuários, caso contrário, o buscador não cumpre o papel que dele se espera.

O trabalho de Brin e Page (1998) introduziu o algoritmo *PageRank* para melhor ranquear as páginas *web* indexadas pelo *Google*, utilizando um modelo probabilístico para determinar o nível de relevância entre as páginas indexadas. Nesse algoritmo, os *hyperlinks* que as páginas contêm são usados para determinar seu nível de relevância, montando, assim, um grafo dos domínios indexados. O grafo é gerado a partir da mensuração de quais *links* são mais citados por outros, criando o ranqueamento de todas as páginas indexadas, sem levar em conta as *querys*.

Para buscadores que não indexam documentos *web*, pode ser utilizada técnicas de ranqueamento mais simples, como as métricas de distâncias e similaridades, as quais são amplamente utilizadas em problemas de classificação e clusterização (Salton; Wong; Yang, 1975), (De Souza; Ciarelli; Oliveira, 2014), (Oliveira et al., 2016),(Oliveira; Branquinho Filho, 2017). Comparando a *query* com os documentos indexados e ordenando esses documentos conforme a maior similaridiade ou com a menor distância em relação a *query*.

Como este é considerado o principal módulo em um buscador, deve ser o mais otimizado para trabalhar inúmeras *querys* ao mesmo tempo e para processar todos os documentos recuperados, em um tempo não superior a 1 segundo. Este tempo é considerado como o tempo máximo para a percepção humana para um atraso da resposta, mas sem que o fluxo de pensamento seja interrompido (Nielsen, 1993). Além disso, o ranqueador, pode levar em conta as características do usuário, tendo como base buscas passadas e localização geográfica.

### 2.2.3 Avaliação

O módulo de avaliação é responsável por medir a eficiência e eficácia do ranqueador, armazenando essas informações em *logs*. Esses dados podem ser utilizados para ajustar o ranqueador e até mesmo o índice.

Este módulo não é executado em tempo real, pois trabalha em uma parte muito crítica de qualquer buscador, demandando muitas análises e experimentos.

Dentro dos *logs*, podemos saber se o ranqueamento está bom, com base nas interações dos usuários com os resultados, ou se eles estão realizando outras buscas ao perceberem que a anterior não foi capaz de retornar os documentos desejados.

Entretanto, a avaliação desses *logs* também é um problema complexo, pois depende de vários fatores, como período das buscas, contexto, análise subjetiva dos usuários, forma com que os usuários fazem as buscas e como os documentos são avaliados.

Muitos trabalhos foram desenvolvidos partindo da premissa de que é preciso analisar os *links* clicados pelos usuários conforme o buscador era utilizado (Joachims, Thorsten, 2002), (Radlinski; Kurup; Joachims, 2008). Esse tipo de análise é de baixo custo. Além disso, poucos usuários estão dispostos a informar dados sobre a escolha dos *links* que acessaram entre os retornos de uma busca.

Grimes, Tang e Russell (2007) descreveram três métodos para coletar dados dos usuários: o estudo em laboratório, painéis de instrumentos e registros de buscas.

O método de estudo em laboratório é realizado em um ambiente controlado, no qual os pesquisadores observam um grupo pequeno de usuários nas atividades pré-determinadas pelo experimento, analisando quais *links* os usuários acessaram e que decisões eles tomaram para a escolha desses *links*. Também é possível o uso de equipamentos mais específicos, como *eye-tracker*, que permite coletar mais dados. Porém, os usuários não se comportam de forma "natural", pois estão sendo observados em meio às atividades dos pesquisadores. Outra desvantagem deste método é o número de amostras e sua variabilidade.

Os painéis de instrumentos são programas instalados nos computadores ou no navegador dos usuários, objetivando monitorar seu comportamento ao utilizar o buscador. O número de usuários é maior que a abordagem em laboratório, tendo-se, desse modo, uma variabilidade também maior. Essas ferramentas podem coletar quais as aplicações estão em execução nos computadores, quais *links* eles acessaram, podendo, ainda, perguntar aos usuários os motivos de suas escolhas. Esta abordagem necessita ser desenvolvida e distribuída para todos os participantes e, com isso, os usuários se comportam com mais naturalidade.

Por último, os registros de buscas permitem coletar os *links* acessados pelos usuários, buscas anteriores por eles realizadas, localização geográfica, entre outros dados. Esta abor-

dagem permite trabalhar com muitos ou todos os usuários, conferindo maior variabilidade à amostra, embora não seja possível saber quais as reais intenções de busca dos usuários.

Entretanto, existem algumas situações que introduzem ruídos nos dados coletados, como o compartilhamento do computador com outras pessoas, remoção periódica dos *cookies* e ocultação do rastreamento das buscas. Esses ruídos podem ser insignificantes em relação à massa de usuários do buscador, mas devem ser descartados.

A análise de registro de buscas é o método mais utilizado, em função do custo e da escalabilidade. Porém, como foi dito, só é possível predizer a intenção das buscas por meio das ações dos usuários e, ainda assim, com dificuldade. Há casos em que, ao longo das interações com o buscador, os usuários vão refinando as *querys* para encontrar os resultados esperados, partindo de termos genéricos e seguindo para termos de interesse, para, assim, explicitar seus objetivos.

Um problema a ser trabalhado nos dados é a desambiguação das buscas. Por exemplo, relações de termos que normalmente não são amplamente buscados, como no exemplo citado por Grimes, Tang e Russell (2007), as buscas com o termo "galo" podem referenciar com maior probabilidade o animal ou produtos a ele relacionados. No entanto, a intenção de busca dos usuários era encontrar dados sobre o animal galo do zodíaco chinês.

Outro ponto vital da análise e ranqueamento dos resultados é o quão atualizado está um buscador, pois é frequente uma mudança na relação de algumas *querys-chave*, influenciada por aspectos como o falecimento de algum famoso, desastre natural ou evento de larga escala. Além da análise do acervo e dos registros de buscas, outra maneira de melhorar os resultados é expandir a *query* original em várias outras, para gerar *querys* que mais se aproximem do contexto da busca.

O trabalho de Mendes, de Moura e Ziviani (2002) propõe uma técnica de expansão de *query* utilizando a indexação semântica latente ( *Latent Semantic Indexing* – LSI) (Furnas et al., 1988). Tal técnica mostrou-se 14% melhor em relação ao trabalho de Qiu e Frei (1993). Porém, o LSI é bastante custoso, sendo necessário muito poder computacional para a execução em tempo real.

# 3 Proposta do Trabalho

O aLine foi desenvolvido para indexar bases e documentos que estão fora do escopo dos buscadores mais usados. No momento, temos os jornais *A Tribuna* e *Metro* como acervos indexados ao buscador desenvolvido.

O buscador foi utilizado nos estudos de Cavaca, Antunes e Nogueira (2016) e Cavaca et al. (2016), auxiliando no trabalho de buscas do acervo por eles utilizado. Em Cavaca, Antunes e Nogueira (2016), foi realizada uma análise no acervo do jornal *A Tribuna* no período de 17 de março de 2004 a 30 de junho de 2009. Nesta análise, foram coletadas 214 reportagens contendo os termos "odontologia", "saúde bucal" e "dentista". O uso do aLine foi crucial, pois, conforme estimativa, o tempo necessário para a conclusão dessa tarefa sem o buscador seria de 5.799 horas (724 dias, trabalhando-se oito horas por dia). Com o aLine, foram necessárias apenas 32 horas (quatro dias, trabalhando-se oito horas por dia). A economia foi de 720 dias, conforme a estimativa.

Na primeira seção deste capítulo, está a descrição da arquitetura do aLine, os componentes nele presentes e como estes foram adaptados ao tipo de base que o buscador indexa. A segunda seção contém a descrição dos serviços que o aLine disponibiliza a análise temporal de um termo e a busca por nomes.

## 3.1 Arquitetura do aLine

O aLine realiza todas as funções de um buscador citadas na Seção 2, porém, com algumas modificações, em função do tipo de documento que ele processa. Sua arquitetura é composta pelos seguintes módulos: interface, coletor, indexador e motor de busca, conforme exibido na Figura 4.

### 3.1.1 Base de dados

A base de dados utilizada no aLine é formada pelos acervos dos jornais *A Tribuna* e *Metro*, nos quais é disponibilizada, em seus respectivos *sites*, a versão digital da edição impressa desses veículos.

Todos os documentos de *A Tribuna* estão no formato PDF e abarcam edições publicadas de 2003 a 2018. Estão separados por páginas do jornal, ou seja, há um único arquivo para cada página publicada, totalizando mais de 300 mil páginas. Além disso, há marcadores definindo à qual caderno do jornal a página pertence, sendo, no total, 12 classes, como mostra a Tabela 1.

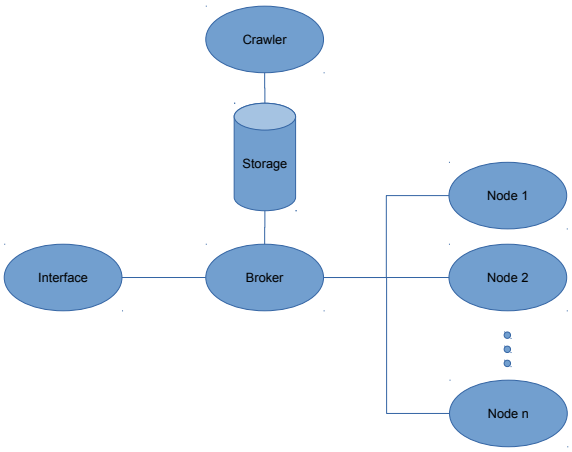



Figura 4 – Diagrama da arquitetura do aLine.

| Cadernos do jornal A Tribuna | | | |
|---|---|---|---|
| At2 | Noticiário | Sobre Rodas | Jornal da Família |
| Tv Tudo | Mulher | Minha Casa | Informática |
| Especial | Imóveis | Classificados | Tudo a Ver |

Tabela 1 – Cadernos do jornal *A Tribuna*.

O acervo do *Metro* indexado pelo aLine é composto por 410 documentos da edição publicada no estado do Espírito Santo entre 02/01/2017 e 11/09/2018. Esses 410 documentos totalizam mais de 7000 páginas indexadas. Diferente do jornal *A Tribuna*, o jornal *Metro* não classifica seus jornais por caderno.

### 3.1.2 Coletor

Para coletar todos esses documentos do jornal *A Tribuna* e *Metro*, foi criado um coletor especial para essas bases, pois a forma como eles estão disponibilizados não é igual aos padrões de indexação dos grandes buscadores, como explicado na Subseção 2.1.1.

No caso de *A Tribuna*, os documentos estão organizados em pastas pelo ano, mês e dia, sendo que cada página do jornal, conforme dito, é um documento em PDF. Esses documentos possuem um nome único, diferenciando-se por uma sequência de caracteres baseada no dia, mês, ano, sigla do caderno e página. Como não há *hyperlinks* que referenciam esses documentos, é necessário montar os *links* com essas combinações para coletar os documentos.

As publicações do jornal *A Tribuna* são diárias, mas, no *site* da empresa, somente são disponibilizadas as edições de 15 dias antes da data da busca, não sendo possível acessar a edição mais atual no buscador.

A base do jornal *Metro* possui uma estrutura muito parecida. Todavia, na versão digital, é liberado um único arquivo, contendo a edição completa do jornal, publicada apenas em dias úteis, aos quais se restringe a circulação do jornal. Ao contrário do jornal *A Tribuna*, o *Metro* disponibiliza a versão digital no mesmo dia da publicação da edição impressa.

Dadas as regras de publicação dos jornais, o *crawler* é acionado uma vez por dia, mantendo nos registros quais *links* deve criar para coletar os arquivos necessários. Há casos em que o *link* existe, mas o documento não está disponível. Assim, o *crawler* adiciona esse *link* a um registro e, ao longo do tempo, tenta coletá-lo.

### 3.1.3 Extrator

Depois que os *links* são montados e validados, os documentos coletados são enviados à um servidor para serem armazenados e, na sequência, os textos serem extraídos.

A extração é feita a partir da leitura dos metadados encontrados nos arquivos *PDF*, coletados e armazenados em arquivos em formato de texto simples para serem indexados. A extração de texto do buscador aLine, utiliza o Tika[1], pois é uma ferramenta de detecção e extração de metadados em texto armazenado em vários tipos de arquivos (como arquivos de apresentação, planilhas e PDF) (Apache, 2017).

Contudo, durante o desenvolvimento do buscador, foi detectado que algumas palavras estavam incorretas quando extraídas. Ao indexar a base do jornal *A Tribuna*, encontramos várias inconsistências no índice, guiando-nos, a partir disso, para encontrar três tipos de ruídos: a translineação (quebra da palavra por hífen), a fragmentação da palavra e a união dos dois tipos de ruído. Esse processo é descrito no trabalho de Nogueira e Oliveira (2017).

Para corrigir os ruídos encontrados na base de dados, tivemos que criar um dicionário com as palavras da língua portuguesa, identificar os ruídos e seus tipos.

Esse dicionário utiliza as palavras do português do Brasil do arquivo *Delaf* (Núcleo Interinstitucional de Linguística Computacional, 2004) presentes na ferramenta UNITEX (Unitex, 2016). Escolhemos as palavras deste arquivo, pois precisávamos da maior quantidade de palavras e suas flexões. Além disso, removemos todas as letras do alfabeto que ocorrem sozinhas, para não confundir o programa nos casos de ruído de fragmentação. Além disso, adicionamos nesse dicionário, uma lista com todos os nomes dos municípios brasileiros.

Outros trabalhos existente para a correção de texto extraído de documentos físicos como o (Bassil; Alwani, 2012), utilizaram consultas no Google para determinar qual a correção para o termo. No nosso caso, o segundo e o terceiro tipo de ruído nos trouxe

[1] https://tika.apache.org/

alguns problemas para essa estratégia, pois como não sabemos exatamente qual o tamanho do termo e em alguns testes o Google nos fez sugestões fora do Português. Então optamos por utilizar um dicionário local.

A detecção e a solução de tipo de ruído é descrita a seguir separadamente, de acordo com suas particularidades.

#### 3.1.3.1 Primeiro Caso: Quebra por Hífen

O primeiro tipo de ruído encontrado, a quebra por hífen, ocorre quando uma palavra está no final de uma linha e ela necessita de mais espaço para ser inserida. Se ela não possuir hífen, a quebra deve ser feita por sílabas. Caso contrário, a quebra deve ser feita no hífen. A sua detecção é a mais simples, basta apenas verificar em cada quebra de linha, se o último caractere é um hífen. Como mostra a Figura 5.

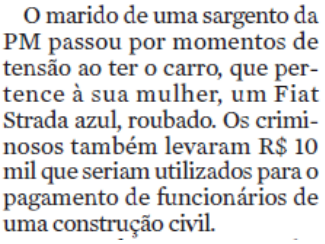

O marido de uma sargento da PM passou por momentos de tensão ao ter o carro, que pertence à sua mulher, um Fiat Strada azul, roubado. Os criminosos também levaram R$ 10 mil que seriam utilizados para o pagamento de funcionários de uma construção civil.

Figura 5 – Trecho de uma notícia extraída do jornal.

Como podemos ver Figura 5, na terceira e quinta linhas, as letras das palavras “pertence” e “criminosos” não foram totalmente acomodadas na mesma linha, assim sendo necessária as suas quebras.

Um procedimento constituído de 4 etapas foi criado para correção deste problema. A primeira etapa consiste em encontrar as linhas que possuem o hífen como último caractere no último termo. A segunda etapa, consiste em concatenar a linha que possui o hífen com a próxima linha, mantendo o hífen.

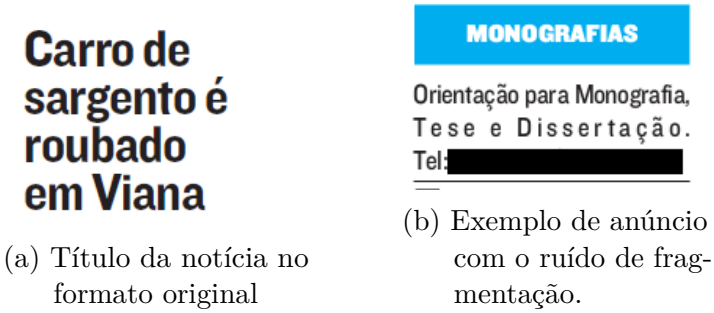
Terminado esta etapa, seguimos para terceira etapa, a busca no dicionário. Essa nova palavra unida é buscada no dicionário. Se ela existir, mantemos a forma com que está, caso contrário passamos para o próxima etapa.

Nesta etapa, contamos a quantidade de hífen que ela tem e criamos as variações dela, com e sem hífen. Se encontramos uma correção para ela, realizaremos a correção, caso contrário, classificamos como terceiro tipo ruído.

#### 3.1.3.2 Segundo Caso: Fragmentação

O segundo caso ocorre em duas situações, a primeira quando extraímos o texto e a segunda quando o texto está formatado com essa fragmentação, para enfatizar ou ocupar

melhor o espaço. Um exemplo da primeira situação está na Figura 6a. O seu texto está perfeitamente formatado, mas quando extraímos o texto, o ruído aparece.




(a) Título da notícia no formato original

(b) Exemplo de anúncio com o ruído de fragmentação.

Figura 6 – Exemplos do segundo tipo de ruído.

O resultado de extração do Tika para esse título ficou o seguinte: “Carro de sargento é ro u b a d o em Viana”. A palavra “roubado” foi fragmentada na primeira sílaba e nas letras, mas também encontramos outros casos onde a fragmentação ocorre por pares, ou mais, de letras ou em trechos da palavra.

Para a segunda situação do ruído de fragmentação, a Figura 6b é um excelente exemplo, a formatação do texto causa esse ruído. A parte do texto “Tese e Dissertação”, foi fragmentada justamente para utilizar todo o espaço do anúncio.

Portanto, para ambas as situações, o ruído não possui um tamanho fixo de fragmentação ou locais fixo na estrutura do texto, sendo de causa aleatória.

Esse tipo de ruído, a fragmentação das palavras, é um pouco mais complexo de detectar e resolver. Para detectar as palavras com erro, substituímos os sinais de pontuação, com exceção do hífen, por espaços em branco, separamos as palavras pelo espaçamento e as inserimos em um vetor. Depois calculamos a média de letras por palavra em cada linha, se essa média for igual ou inferior a 3 marcamos esta linha como suspeita. Escolhemos a média como 3 pois, as palavras com tamanho a partir de três letras começam a ter mais importância na nossa língua em relação a palavras com duas letras. Utilizaremos a Figura 6a para exemplificar o método.

| Linha | Conteúdo | Vetor | Média |
|---|---|---|---|
| 1 | “Carro de” | [“Carro”;“de”] | 3.5 |
| 2 | “sargento é” | [“sargento”;“é”] | 4.5 |
| 3 | “ro u b a d o” | [“ro”;“u”;“b”;“a”;“d”;“o”] | 0.85 |
| 4 | “em Viana” | [“em”;“Viana”] | 3.5 |

Tabela 2 – Vetorização e médias de cada linha.

Como podemos ver na Tabela 2 o processo de detecção do ruído de fragmentação. A terceira linha da Figura 6a possui a média inferior a 3, assim sendo marcada como suspeita.

Seguindo para o segundo passo, verificamos para cada palavra deste vetor, se elas existem no dicionário. O resultados dessa verificação é armazenado em um outro vetor do tamanho igual a quantidade de palavras que ocorre na linha. Caso a verificação no dicionário retorne verdadeira, adicionamos no vetor o valor 1, caso contrário, esse valor será 0.

Assim podemos dizer em qual região da linha possui o ruído de fragmentação. Esse método não é preciso, mas nesse passo já conseguimos descartar as linhas que estão corretas ao consultar o dicionário.

| | |
|---|---|
| Vetor | [“ro”;“u”;“b”;“a”;“d”;“o”] |
| Mapa | [0;0;0;0;0;0] |

Tabela 3 – Mapa da linha suspeita.

Como é exibindo na Tabela 3, todos os termos da linha selecionada não existem no dicionário. Assim, para cada termo é associado a ele o valor 0 no mapa.

Com esse mapa, o nosso algoritmo monta e verifica todas as combinações, dessas letras, no dicionário, a partir do primeiro elemento que possui o valor 0 e atribuindo pontos para cada combinação existente. Esses pontos são baseados na quantidade de termos utilizados para formar uma palavra válida sobre a quantidade total de termos na linha. Eles vão de 0 a 1 e quanto mais próximo de 1, mais provável de ser a correta.

Nesse passo, vamos para o primeiro termo de valor 0 no mapa da Tabela 3 e verificamos se ele é um artigo ou advérbio. Se for, criamos mais uma possibilidade para esse termo, presumindo que ele não faça parte do termo analisado. Se ele não passar nessa verificação, o concatenamos com o próximo termo. Assim formando uma ou mais possibilidades para essa região.

Ao gerar todas a variações possíveis de termos válidos no dicionário, pontuamos cada possibilidade gerada de acordo com o número total de letras utilizadas para gerar o termo sobre o total de termos na linha (descartando os sinais de pontuação). Esse sistema de pontuação nos indica um *ranking* das possibilidades de correção, assim facilitando a escolha da correção mais provável para a linha.

Se existir uma possibilidade de pontuação 1, aplicamos a correção. Caso não exista, mas haja possibilidades com pontuações menores, nós combinamos todas as possibilidades entre elas, tentando gerar um conjunto de termos combinados de acordo com a soma das pontuações chegando o mais próximo de 1. Mas, também levamos em conta se há conflitos entre as possíveis correções utilizadas no conjunto, assim em cada conjunto só há termos que não possuem esse conflito.

Mesmo se não houver um conjunto que chegue a 1, nós pegamos o conjunto de maior pontuação e aplicamos a correção. Mas essa linha é separada para posteriormente

ser anotada por um humano. E no caso de não existir um conjunto ou possibilidade de correção, essas linhas também serão separadas para a anotação manual.

Continuando o exemplo, geramos todas as possibilidades de palavras válidas a partir do primeiro elemento com valor 0 no mapa de erros, como é mostrado na Tabela 4.

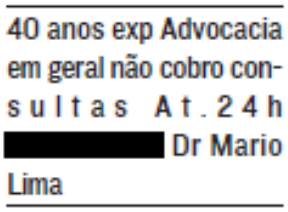
| Termo criado | Índices dos termos utilizamos | Pontuação |
|---|---|---|
| "rouba" | [0;1;2;3] | 0.666 |
| "roubado" | [0;1;2;3,4,5] | 1 |
| "do" | [4;5] | 0.333 |

Tabela 4 – Possibilidades geradas a partir do primeiro elemento do mapa com valor 1.

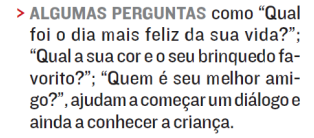
Como mostra a Tabela 4, o nosso exemplo gerou uma possibilidade com pontuação 1, então essa possibilidade será escolhida para corrigir a linha que está fragmentada.

#### 3.1.3.3 Terceiro Caso: Primeiro Caso + Segundo Caso

Já o terceiro tipo de ruído, é a união do primeiro, a translineação, com o segundo, a fragmentação. A detecção dele não é difícil, mas a sua solução sim. Igual ao ruído de fragmentação, o terceiro também ocorre tanto na formatação quanto na extração do texto.




(a) Exemplo de anúncio com o terceiro tipo de ruído.




(b) Trecho de uma notícia sem erro na formatação mas, com o terceiro tipo de ruído.

Figura 7 – Exemplos do terceiro tipo de ruído.

A Figura 7a é um anúncio do caderno de classificados que contém o terceiro tipo de ruído em sua formatação. A palavra "favorito" na terceira linha da Figura 7b, ficou dessa forma "fa- vor i to" quando extrairmos o texto dessa notícia. Em nosso desenvolvimento, não chegamos em uma estratégia que o corrigisse automaticamente, então a sua correção é realizada manualmente.

Essas estratégias foram avaliadas em um experimento, no qual corrigiram 79% dos termos com problema da base de experimentação com 100 páginas do jornal *A Tribuna*, no tempo de 30 segundos, e em comparação a uma outra estratégias que conseguiu 71% de correção em 130 minutos (Nogueira; Oliveira, 2017). Ao finalizar as correções dos textos extraídos, os documentos são enviados ao *broker* para serem indexados.

### 3.1.4 Indexador

O indexador do aLine é distribuído dentro de um *cluster*, no qual o *broker* controla os *nós* de processamento do buscador. Essa forma de organizar os computadores permite trabalhar com bases de dados grandes com um custo menor. Como é descritor em (Brin; Page, 1998), (Ribeiro-Neto; Barbosa, 1998), (Badue et al., 2005), (Badue et al., 2010),(Baeza-Yates; Ribeiro-Neto, 2011), (Crof; Metzler; Strohman, 2015) e (Tolosa, 2016).

Assim que um documento ou mais documentos prontos para serem indexados, ou seja, o conteúdo foi extraído e eventualmente corrigido, esses documentos são enviados ao *broker*. O *broker* assim que recebe esses documentos, automaticamente os distribui por entre os *nós*. Esta distribuição procura dividir igualmente os documentos, mas caso não seja possível, o *broker* mapeia os *nós* com menos documentos para que em uma nova inserção. Esses *nós* recebem mais documentos a fim de igualar a distribuição entre os *nós*.

Ao finalizar a distribuição dos documentos, o *broker* envia uma mensagem para os *nós* para que iniciem a indexação. Os *nós* iniciam a leitura dos documentos atribuídos a cada um em lotes, criando um índice de termos para cada documento e contabilizam a frequência desses termos presentes nos documentos. Este primeiro processo, agiliza a indexação pois não há dependência direta entre os documentos.

Após isso, cada documento recebe um número de identificação, esta identificação é referente a ordem de inserção do documento no acervo. O aLine considera como termos apenas conjuntos de caracteres separados por espaço em branco, descartando caracteres não alfanuméricos.

Realizado este passo, os *nós* iniciam a indexação dos termos encontrados nos documentos. Para prover a agilidade na busca do aLine, foi utilizado a tabela *hash* para mapear os termos existentes no acervo, e para cada termo é associado uma lista com os documentos os quais ocorrem. A *hash* utilizado no aLine é uma versão existente na linguem *C++*, a qual possui implementações otimizadas para tratar colisões de *hash* (Galowicz, 2017). Para cada termo existente no acervo é associado a ele um número referente a sua ordem de indexação no nó. Dada essas estruturas, o processo de indexação parte da leitura do índice de termos de cada documento, buscando no índice do nó, se os termos de um documento existem no índice. Caso exista, o termo no índice associa o número de identificação deste documento a sua lista de ocorrências. Caso não exista, o nó adiciona o novo termo no índice global, atribuindo um identificador e associando o documento que ocorre este novo termo.

Finalizado este processo de indexação local, os documentos indexados são representados pelo modelo vetorial proposto por Salton, Wong e Yang (1975) (*Term Frequency - tf*). Neste modelo, cada documento é representado por um vetor $d_j$, onde $j$ varia de 1 a $m$

e $m$ é a quantidade de documentos na base. Este vetor possui $n$ dimensões, essa dimensão é quantidade de termos existentes na base; cada termo no documento é associado por peso $w_{i,j}$, este peso, no atual momento, é associado a frequência do termo. E para cada termos existente na base é associada a ele um valor de $i$, no qual varia de 1 a $n$. Dessa maneira, o documento é representado pela Equação 3.1.

$$d_j = (w_{1,j}, w_{2,j}, w_{3,j}, w_{4,j}, ..., w_{i,j}, ..., w_{n,j}) \tag{3.1}$$

Com este modelo de representação dos documentos indexados, cada nó armazena os dados dos documentos em um arquivo para que possa ser utilizado futuramente. Dentre esses dados estão o nome do documento, o modelo vetorial e se houver, o *link* que aponta para o documento.

Assim que todos os *nós* finalizem a indexação dos documentos, o *broker* requisita a todos os *nós* os termos indexados e a frequência de cada um, para que essa informação seja sumarizada. Assim que este processo é concluído, o *broker* envia para cada nó o resultado desta sumarização. Este processo é importante, pois a partir dele o aLine é capaz de ponderar os pesos de cada documento. Para isso, utilizamos a combinação da frequência do termos junto com o ponderador *Inverse Document Frequency* (*idf*) (Robertson, 2004), citando na Seção 2.1.3. O *idf* pode diminuir a importância dos termos que ocorrem na maioria dos documentos, assim valorizando termos com baixa ocorrência no acervo.

O peso final de cada termo no documento será obtido através da multiplicação do *tf* pelo *idf* correspondente. Por fim, será regida pela seguinte Equação 3.2.

$$w_{i,j} = tf_{i,j} * idf_i \tag{3.2}$$

Concluída a ponderação dos vetores nos documentos, o processo de indexação é finalizado e o buscador está apto a realizar buscas. Posto que o tipo de documento trabalhado no aLine e modo com que os textos foram extraídos, não foi utilizado a técnica de *LSI* no processo de indexação, pois em uma página do jornal, por exemplo do jornal *A Tribuna*, é comum existir mais de uma reportagem. Assim gerando correlações equivocadas entre os termos, como é exemplificado na Figura 8, a qual apresenta seis reportagens distintas entre elas.

Para que o *LSI* possa ser aplicado no aLine será necessário uma segmentação das reportagens das páginas dos jornais. O uso de um segmentador nas reportagens melhoraria os resultados nas buscas, pois as buscas não seriam por documentos(página inteira do jornal) e sim por cada reportagens no interior da página do jornal.

VITÓRIA, ES, SEGUNDA-FEIRA, 27 DE AGOSTO DE 2012 A TRIBUNA 7

Informática

GAMER NEWS

1

Zumbis à solta

Um jogo de ação em mundo aberto, onde os gamers lutam pela sobrevivência em um cenário pós-apocalíptico repleto de zumbis, bem ao estilo da série "The Walking Dead" e do popular "DayZ".
Assim será "State of Decay", novo projeto de Jeff Strain, co-criador de "World of Warcraft" e "Guild Wars".
O jogo conta com o suporte da Microsoft Game Studios e será distribuído por download para PC e Xbox Live Arcade. O título ainda não tem data de lançamento definida.

2

Em setembro

Chega no mês que vem ao Brasil o Nintendo 3DS XL, a nova versão do portátil da empresa que apresenta imagens tridimensionais sem o uso de óculos especiais usados no cinema. A data oficial ainda não foi divulgada, mas já se sabe que ele custará R$ 1.200.
A Nintendo lançou o videogame nos Estados Unidos no último dia 19. O aparelho tem como principal destaque telas 90% maiores do que as do modelo anterior do 3DS, que chegou ao mercado no ano passado.
Detalhe: nos EUA, o aparelho custa US$ 200.

3

Inteligência artificial e dança

"Yakuza 5", que será lançado no Japão em dezembro para Playstation 3, receberá um novo sistema de combate, inteligência artificial melhorada e até um sistema de dança.
Os principais diferenciais são a nova inteligência artificial, nos locais de combate, que foram redesenhados, e na animação dos inimigos, que foi refeita.
Desta vez, os inimigos poderão aparecer em qualquer lugar do game e perseguirão o herói pelo cenário do jogo.

Disponível [illegible] novidades

"Counter-Str[illegible]bal Offensive", que marca a volta [illegible] que conquistou jogadores po[illegible]do [illegible]ndo há 12 anos, já está disp[illegible]el no [illegible]m e Xbox Live.
O gar[illegible]azmais [illegible]5 armas, novos mapas, n[illegible] visuais adaptados pa[illegible]a atualiza[illegible]ruído isto acompanhado pela jogabilid[illegible] que tornou o título famoso por muitos [illegible]rs.

Enem americano

Quatro [illegible]asileiros criaram o "Estuda[illegible], um jogo virtual que ajuda os [illegible]didatos a vagas em universidades a[illegible]icanas a estudarem para o Teste de [illegible]lar, semelhante ao Exame Nacional do[illegible]o Médio (Enem), realizada nos Estados[illegible]idos como parte do processo seletivo [illegible]o ensino superior.
De a[illegible]erson Ferminiano, um dos criadores do jogo, mais de mil pessoas já instalaram as versões do aplicativo para o Facebook e para o iPhone e o iPad desde que elas foram lançadas.

Lançamento adiado de pacote

O pacote que adiciona suporte para multiplayer on-line em "Scott Pilgrim vs. the World", que estava previsto para ser lançado na semana passada, só vai chegar no final do ano. A Ubisoft não explicou o motivo da alteração da data. Um representante da empresa afirmou que serão fornecidas outras informações quando o lançamento tiver mais próximo.
Além da adição do multiplayer on-line, o extra custará 400 MS Points no Xbox 360 (possivelmente US$ 5 no Playstation 3.

6

Figura 8 – Exemplo de página do jornal *A Tribuna* com mais de uma reportagem.

### 3.1.5 Interface

O aLine possui três páginas *web*: a primeira é a página inicial do buscador e contém apenas a barra de busca, como mostra a Figura 9.

A segunda página é utilizada para exibir os resultados das buscas, o que é feito em páginas de dez documentos, como exibido na Figura 10, caso a busca tenha mais que dez documentos, um sistema de paginação é exibido para que o usuário acesse os outros documentos das busca.

A terceira página apenas exibe um gráfico da análise temporal de um ou vários termos, como mostra a Figura 11. Essa funcionalidade será melhor descrita na seção 3.3.

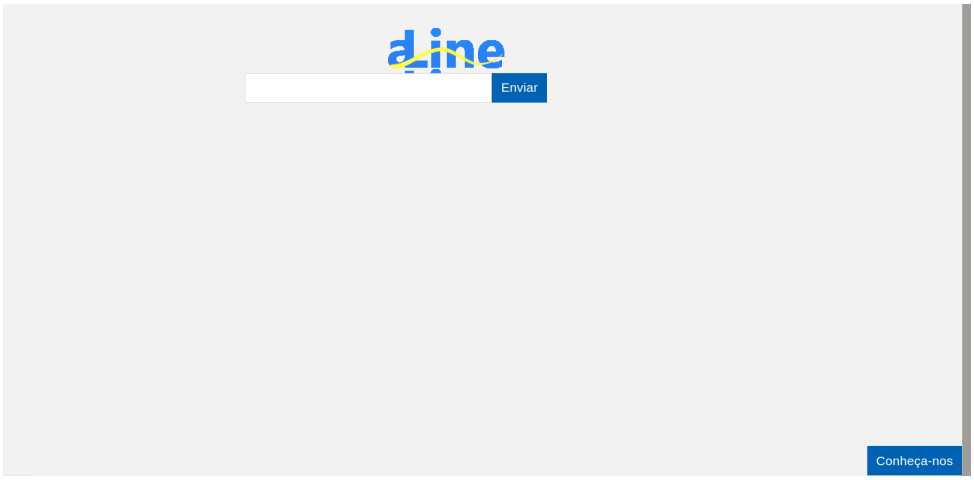



Figura 9 – Página inicial do buscador aLine.

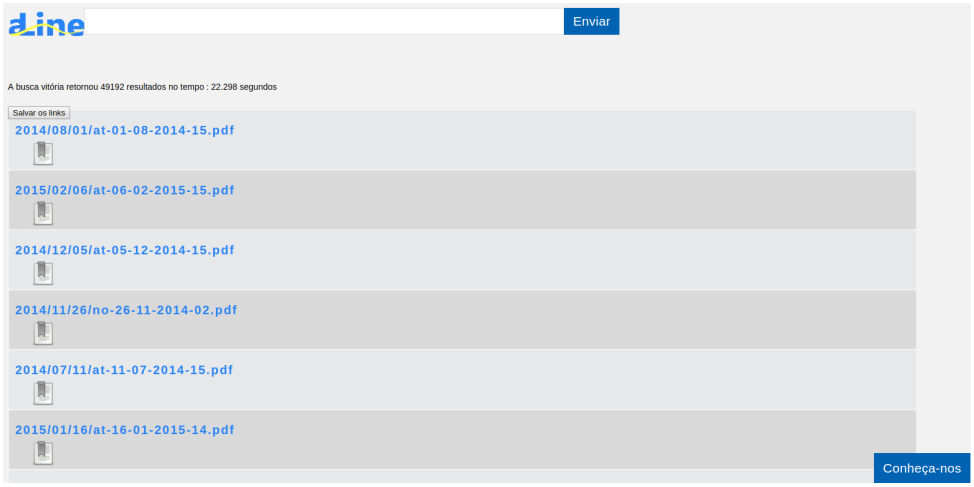



Figura 10 – Página de resultados das buscas do aLine.


Figura 11 – Página com o gráfico da análise temporal de uma busca.

## 3.2 Processo de Busca

Todas as buscas seguem uma ordem de execução. Inicialmente, independentemente do serviço desejado, seja de buscas simples ou análise de um termo desejado, é realizado um

processamento na *query* do usuário, identificando alguns elementos, tais como os termos que ele deseja buscar, a base que deve ser buscada, se há restrições de datas e o tipo de serviço. Feito isso, a saída deste processamento é encaminhada ao *broker*, para que a busca seja iniciada.

Assim que a *query* chega no *broker*, ele inicia o processo de busca, enviando a *query* para todos os *nós* que possuem o acervo desejado. Logo que os *nós* recebem a *query*, eles procuram nos índices próprios, os termos presentes na *query* e armazenam as listas de documentos associados aos termos encontrados. Durante este processo de busca, os *nós* também realizam a vetorização da *query* para que possa ser utilizada no processo de ranqueamento.

Logo que o processamento da *query* e a busca dos termos presentes nela são finalizados, os *nós* recuperam os vetores dos documentos que possuem os mesmos termos que a *query*. Concluído a recuperação dos vetores, os *nós* iniciam o ranqueamento dos documentos. A estratégia de ranqueamento adotada no aLine, baseia-se no uso da similaridade de Cosseno dos documentos recuperados com a *query* (Baeza-Yates; Ribeiro-Neto, 2011). A similaridade de Cosseno é regida pela Equação 3.3.

$$cos(d_i, d_j) = \frac{d_i\, d_j}{\|\, d_i\|\,\|\, d_j\|} = \frac{\sum_{k=1}^{n} w_{k,i}\, w_{k,j}}{\sqrt{\sum_{k=1}^{n}(w_{k,i})^2}\sqrt{\sum_{k=1}^{n}(w_{k,j})^2}} \tag{3.3}$$

Por fim, os documentos recuperados são ordenados pela maior similaridade com a *query* e, na interface do buscador, são entregues os 100 primeiros resultados. Utilizamos este valor para diminuir o tamanho da mensagem enviada à interface, mas caso o usuário requisite os outros resultados, o buscador fará a entrega quando o usuário chegar na décima página de resultado.

## 3.3 Outras aplicações

O aLine possui outras aplicações além da busca de termos genéricos nas bases de dados, tais como busca por nomes e em domínios e, ainda, análise temporal. Essas aplicações são executadas a partir de palavras-chave, como exibido na Tabela 5.

| Comando | Descrição |
|---|---|
| action:<ação> | Define qual ação o buscador irá executar |
| datefrom:<data> | Define o início do período temporal dos documentos buscados |
| dateto:<data> | Define o fim do período temporal dos documentos buscados |
| database:<base de dados> | Comando que define qual base de dados será buscada |

Tabela 5 – Comandos do aLine e suas descrições

A busca convencional não necessita informar o tipo de ação que o buscador deve executar, pois é definida como a ação padrão. Porém, nas outras aplicações, isso precisa

ser feito. A análise temporal de um ou mais termos é executada assim que a palavra-chave *action:report* é explicitada na *query*. Por exemplo, se inserirmos na barra de busca do aLine uma das seguintes *querys*: "facebook orkut action:report" ou "action:report facebook orkut". Será exibido um gráfico da distribuição dos termos "facebook" e "orkut" como o gráfico a Figura 12.

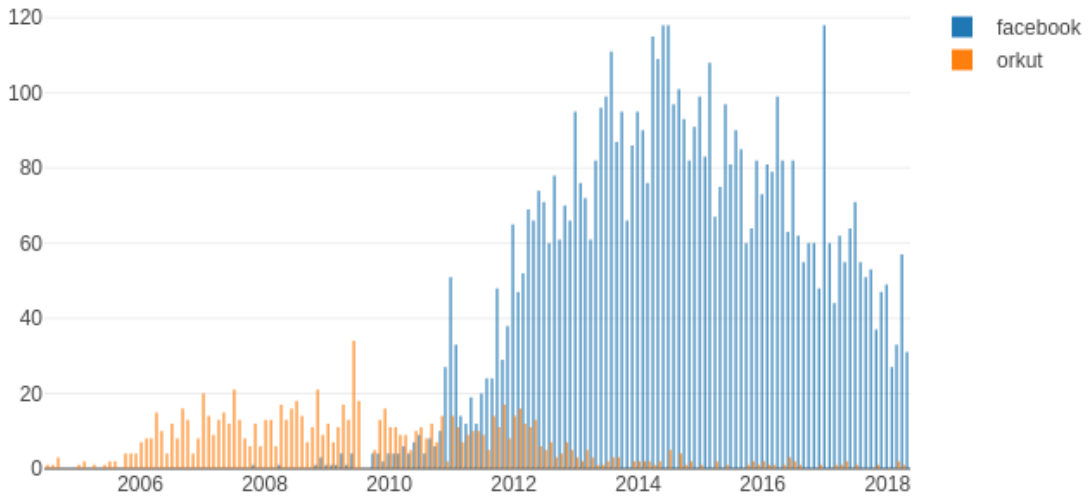



Figura 12 – Análise temporal da busca "facebook orkut" na base do jornal *A Tribuna*.

A busca por nomes, é realizada a partir da definição do comando *database* com a palavra chave "ner-" (ex.: "Alberto Ferreira database:ner-atribuna"). Este tipo de busca só foi possível com o uso das ferramentas desenvolvidas por Pirovani, Nogueira e Oliveira (2018) para a tarefa de identificar entidades nomeadas. Além de realizar buscas por nomes, o aLine, retorna o documento encontrado com marcações no nome, como é exemplificado na Figura 13.

14 ATRIBUNA VITÓRIA, ES, DOMINGO, 20 DE NOVEMBRO DE 2016

Cidades

QUE FIM LEVOU?

# Carro que anda sozinho faz 1ª viagem em janeiro

Trajeto do veículo, que anda sem precisar de motorista, será entre Vitória e Guarapari. Projeto é desenvolvido pela Ufes desde 2011

Carlos Mobutto

O carro inteligente – que anda sem precisar de motorista – vem sendo desenvolvido desde 2011 pela Universidade Federal do Espírito Santo (Ufes) e fará sua primeira viagem em janeiro de 2017. O trajeto será entre Vitória e Guarapari.

O protótipo, batizado como Iara (Intelligent Autonomous Robotic Automobile), funciona com combustível e eletricidade e possui um sistema que dispara 32 lasers ao redor do carro para criar os dados que alimentam o sistema.

Segundo o coordenador do projeto do carro autônomo e professor da Ufes, Alberto Ferreira De Souza, o veículo também conta com uma câmera para fazer o reconhecimento de placas e dos semáforos.

"Tinhamos duas grandes metas: a primeira é dar a volta completa na Ufes, o que a gente já cumpriu várias vezes. A outra meta é a ida de Vitória até Guarapari, que já envolve a presença de outros carros, pessoas, semáforos, placas de trânsito" disse o professor.

Segundo ele, outro quesito para a ida ao balneário é fazer com que o carro possa andar a 50 km/h.

Em 2013, durante uma demonstração em um programa de televisão, houve um acidente no qual o carro atropelou a apresentadora Ana Maria Braga.

Sobre o incidente ocorrido na época, o professor Alberto disse que serviu de aprendizado e não comprometeu a credibilidade do projeto.

"O carro funcionou perfeitamente, já tinha chegado ao seu destino, a Ana tinha saído do carro tranquila. Aí ela quis ver os computadores atrás do carro, e nessa hora eu mudei o modo automático para o modo humano. Estávamos num plano inclinado e então mesmo colocando o carro no modo de estacionar, ele não parou", explicou o professor.

OS NÚMEROS

**32 lasers** são disparados ao redor do carro para que sejam criados os dados que alimentam o sistema

**50 km/h** é a meta de velocidade para que seja feito o trajeto do veículo inteligente até Guarapari



PROFESSOR Alberto Ferreira, coordenador do projeto, diz que veículo já deu volta completa na Ufes várias vezes

## Como funciona

Oito câmeras monitoram espaço ao redor do carro

**1 "OLHOS"**
Para o carro se orientar no espaço, identificar obstáculos e ficar centralizado na pista, ele terá "olho", ou seja um escâner a laser e oito câmeras de vídeo, capazes de fazer a "leitura" em tempo real de tudo o que acontece em volta do veículo.

**2 "CÉREBRO"**
O "cérebro" é formado por um supercomputador que fica no porta-malas, ligado a um aparelho ao lado do banco. Ele faz as leituras das imagens e envia os comandos, como frear e acelerar. O "cérebro" equivale à configuração de cerca de 30 notebooks.

O carro conta com um laser frontal de longa distância para conseguir perceber obstáculos a até 150 metros de distância

**3 TOMADA DE DECISÕES**
O supercomputador será responsável por "pensar" e tomar decisões pelo carro. Ele vai identificar placas de trânsito, obstáculos, objetos fixos e móveis, pessoas e determinar o melhor caminho, com ajuda de GPS.

**4 APRENDIZADO GRADATIVO**
O objetivo dos cientistas da Ufes é desenvolver um "cérebro" capaz de aprender coisas novas gradativamente, assim como o cérebro humano e "pensar" assim como o motorista faria ao dirigir, porém mais rápido e mais preciso.

**R$ 500 MIL REAIS**
Somando o veículo, as adaptações e as tecnologias empregadas, o carro desenvolvido pela Ufes está custando mais de meio milhão de reais.

Fontes: Alberto Ferreira De Souza, professor da Ufes e coordenador do projeto do carro autônomo e alunos envolvidos no projeto; Fernando Osório, professor de Robótica da USP e um dos coordenadores do projeto Carina; Augusto Silva Pereira, professor da UFMG e coordenador do projeto do carro autônomo Cadu.

## "Desafio é lidar com os fatores imprevisíveis", diz professor

O projeto da Ufes do carro autônomo tem um custo estimado de R$ 500 mil e deve estar disponível no mercado na próxima década, segundo o professor Alberto Ferreira De Souza, coordenador do projeto.

"Os mais otimistas dizem que o carro vai estar no mercado para venda em cinco anos, mas eu acredito que isso só acontecerá daqui uns 10 ou 15 anos, pois tem alguns detalhes que precisamos melhorar", explicou Alberto Ferreira.

Segundo ele, o grande desafio, atualmente, é adaptar as reações do carro autônomo aos fatores imprevisíveis que ocorrem no trânsito urbano e nas rodovias, para que ele possa andar mais rápido e atingir pelo menos 50 km/h.

"Na Ufes costumamos fazê-lo andar a uma velocidade aproximada de 30 km/h. O percurso que fazemos usualmente não apresenta as mesmas dificuldades do trânsito. Temos dirigido com ele no modo humano, para que o sistema possa captar mais informações."

De acordo com o coordenador do projeto, o carro ainda precisa de aprimoramento para integração de todos os sistemas, como o de reconhecimento dos semáforos ou pedestres, por exemplo, para ela andar junto com os outros veículos.

"Ainda não teve essa experiência. Precisamos dar capacidade para ele. Ainda há necessidade de alguns testes", disse.

Alberto Ferreira disse que o projeto capixaba é referência tecnológica no setor, mesmo com uma equipe pequena de pesquisadores.

"São quatro pesquisadores à frente do projeto, além dos alunos que têm ajudado muito no desenvolvimento da tecnologia. Outras universidades e empresas têm equipes e recursos maiores."

TECNOLOGIA

A tecnologia da Ufes vem sendo apresentada em vários lugares do mundo, mas outros projetos parecidos estão sendo desenvolvidos na Universidade de São Paulo (USP), na Universidade Federal de Minas Gerais (UFMG), além das universidades de Stanford nos Estados Unidos, Oxford, na Inglaterra e na Universidade de Berlim, na Alemanha. O Google e a montadora Nissan também desenvolveram veículos inteligentes.



PROTÓTIPOS do Google e da Nissan

Figura 13 – Exemplo da busca por nome pelo aLine com a *query* "Alberto Ferreira database:ner-atribuna".

# 4 Avaliação do Trabalho

Para analisar a capacidade de execução do buscador e a relevância de suas respostas, foram realizados dois experimentos, descritos neste capítulo. Para tanto, o ambiente utilizado foi um *cluster* com quatro nós de processamento e um *broker*. Para simular buscas reais, foram utilizados 100 documentos com reportagens da base do jornal *A Tribuna* (De Souza; Ciarelli; Oliveira, 2014); (Oliveira; Branquinho Filho, 2017).

Desses 100 documentos, foram geradas *querys* contendo uma quantidade de termos variando de um a dez havendo, para cada variação, dez exemplos. Além disso, as *querys* selecionadas são as mais representativas para o documento de origem, totalizando 10.000 *querys* para os experimentos realizados no âmbito deste trabalho.

Para ambos experimentos, foi utilizado cinco métricas no ranqueamento do aLine, sendo elas, similaridade de cosseno, soma dos pesos *tf-idf*, distância Euclidiana, distância de Tanimoto e distância de Manhattan.

O ambiente utilizado para esses experimentos foi o Laboratório de Computação de Alto Desempenho(LCAD), tanto o *cluster* do buscador quanto os computadores utilizados para dispararem as *querys* estão na mesma rede de computadores. Esse ambiente permite um controle mais fino sob as variáveis de rede que podem interferir nos experimentos.

## 4.1 Tempo de acesso

Neste experimento o objeto medido será o tempo de resposta do buscador, ou seja, dada uma *query* qual o tempo para o buscador entregar a resposta da busca. Para medir tempo de resposta do buscador, foi utilizado as 10.000 *querys* citadas anteriormente.

| Métrica | Média | Desvio padrão |
|---|---|---|
| Cosseno | 0.095 | 0.084 |
| Euclidiana | 0.101 | 0.096 |
| *TF-IDF* | 0.099 | 0.096 |
| Manhattan | 0.089 | 0.079 |
| Tanimoto | 0.090 | 0.080 |

Tabela 6 – Tempo médio das buscas em segundos nas cinco métricas.

Como é exibido na Tabela 6, a métrica de Tanimoto mostrou a menor média de tempo de busca para todas as *querys* geradas. Entretanto, todas as métricas não possuem diferenças significativas no tempo médio. Com exceção da métrica Euclidiana, as outras possuem um tempo médio de busca abaixo de 100 milissegundos.

Ao analisar as Figuras 15 a 19 do apêndice, os gráficos de cada métrica, vemos que o aumento da quantidade de termos nas *querys* não gera um grande impacto no tempo de resposta. Assim, podemos concluir que o tempo de resposta do aLine é o mesmo independente da métrica.

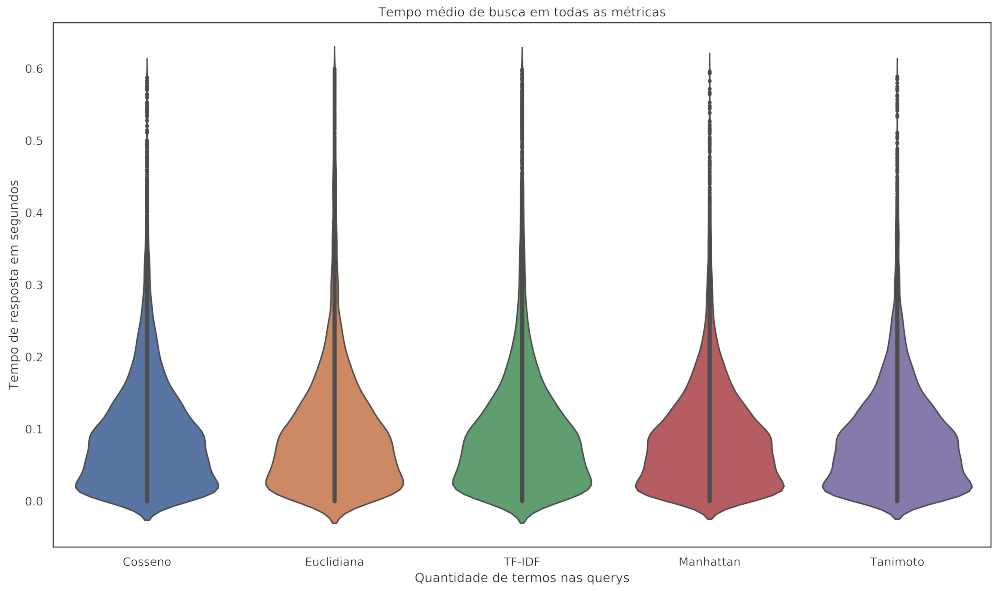



Figura 14 – Distribuição do tempo médio das métricas.

A Figura 14, mostra a distribuição de todos os valores de tempo para cada métrica. Durante os teste, houve buscas em que o tempo de resposta foi muito maior que a média calculada para cada métrica. Essas anomalias, se deram pela interferências na rede interna do laboratório, dado ao fato que, os computadores utilizados para dispararem as *querys* foram utilizados por integrantes do laboratório. Assim, para essas *querys* foram realizados novos disparos a fim de confirmar o tempo de resposta, que no fim se mostrou estar dentro da média encontrada.

Concluindo, o tempo de buscas para *querys* com mais de um termo não cresce tanto quanto de um para dois termos, situando-se como um bom tempo de resposta para os usuários, se levarmos em conta o tempo necessário para a busca sem a ajuda do aLine.

## 4.2 Relevância dos documentos retornados

Para avaliar se o buscador está retornando os documentos mais relevantes, executamos um experimento de avaliação dos resultados utilizando as cinco métricas do ranqueador. Foram usadas as mesmas *querys* do primeiro experimento. Inicialmente foi calculada a similaridade entre as *querys* e os documentos e por meio de votação, foi definida o nível de importância dos documentos da *querys*.

Como é exibido nas Figuras de 20 a 24 do apêndice, conforme a quantidade de termos nas *querys* aumentam, os documentos retornados são os mais similares aos documentos que originaram as *querys*. Este é um comportamento esperado, pois conforme se aumenta as características em comum entre dois objetos, maior é a sua similaridade entre esses objetos. Portanto, o uso de apenas um termo não é muito representativo para retornar o documento desejado, assim para se obter resultados mais similares à *query* é necessário inserir mais termos para as buscas.

| Métrica | Tamanho 1 | Tamanho 5 | Tamanho 10 |
|---|---|---|---|
| Cosseno | 10.89 | 43.99 | 82.16 |
| Euclidiana | 10.87 | 47.02 | 83.27 |
| *TF-IDF* | 10.62 | 45.99 | 83.08 |
| Manhattan | 10.63 | 47.32 | 83.21 |
| Tanimoto | 10.17 | 42.97 | 82.39 |

Tabela 7 – Valores de acerto, em porcentagem, de cada métrica para *querys* de um, cinco e dez termos para os dez primeiros documentos retornados.

Comparando os resultados entre as métricas para este experimento, não há uma diferença significativa entre elas. Conforme a Tabela 7, as *querys* de um termo, todas métricas possuem a mesma faixa de acerto, sendo 10% dos documentos retornado nas dez primeiras posições das respostas das buscas, realmente são os mais relevantes. Este mesmo comportamento se repete nas *querys* com cinco e dez termos. Assim, observamos que, semelhante ao experimento de tempo de resposta, as métricas apresentam os mesmos resultados.

# 5 Considerações Finais

Ao observarmos o universo digital, entendemos que a produção e disponibilidade de documentos em ambientes digitais tende a crescer cada vez mais. Isso se deve à praticidade na criação e o acesso dos documentos digitais. Assim, é necessário adicionar essas informações aos buscadores, para facilitar a recuperação desses dados.

Atualmente o aLine, buscador desenvolvido no âmbito deste trabalho, permite realizar buscas nos acervos dos jornais *A Tribuna* e *Metro* em um tempo aceitável, como mostrado nos experimentos. Isso se deve ao bom ranqueamento dos documentos apresentado. Ainda por conta da agilidade na busca nos acervos, o buscador colaborou nos trabalhos de *clipping* do jornal *A Tribuna* do Observatório Saúde na Mídia - Regional ES.

No aLine, buscamos implementar todos os requisitos descritos por Crof, Metzler e Strohman (2015) e que outros buscadores, tais como o *Google*, utilizam para funcionar. Em função das particularidades dos dois acervos e dos recursos computacionais, todos os módulos foram adaptados para viabilizar o buscador. O *crawler* recebeu uma estratégia de busca de documentos diferente da encontrada na literatura. Isso porque os acervos indexados não utilizam *hyperlinks* para apontar para outros documentos, gerando os *links* que levam a eles.

No módulo de extração, foi preciso adicionar um corretor para diminuir os ruídos encontrados ao se extrair os textos dos documentos indexados, gerados pela formatação e disposição das reportagens nos jornais. Isso permitiu diminuir inconsistências no índice do aLine e, por consequência, melhorar as buscas.

Mesmo com uma arquitetura complexa, foi possível construir um buscador de boa qualidade para o tipo de acervos indexados. Como mostram os resultados dos experimentos, o aLine consegue entregar os documentos mais relevantes com um tempo de resposta reduzido. Buscas com até dez termos possuem um tempo médio de resposta abaixo de um segundo. Ademais, conforme o usuário refina a *query* (aumentando a quantidade de termos), melhor é o ranqueamento dos documentos recuperados, pois essas *querys* são mais representativas em relação a *querys* com poucos termos.

Os serviços de análise temporal, busca por nomes e os acervos indexados são diferenciais em relação a outros buscadores. No aLine, é possível conseguir um gráfico com a ocorrência de um ou vários termos. Isso torna possível determinar se, no período analisado, os termos ocorriam de forma sazonal ou constante ou se o declínio na ocorrência de um termo influenciou na ascensão de outro e vice-versa. A busca por nomes é outro diferencial, pois ela desambigua nomes próprios simples, como Rosa e Margarida, os quais, em buscas no *Google*, aparecem relacionados com flores, e não com pessoas. Destacamos

ainda que, os acervos que o aLine indexa não estão disponíveis nos grandes buscadores, os quais, dessa forma, são exclusivos deste buscador.

# 6 Trabalhos Futuros

O aLine pode oferecer outras funcionalidades, mas, para isso, será necessário aprimorar alguns pontos, por exemplo, diminuir o tempo de resposta e melhorar o ranqueamento existente. Junto com o processo de integração de análises mais complexas poderão ser necessárias mais otimizações. A redução no tempo de resposta pode ser alcançada com uma melhor distribuição do índice entre os nós de busca, criando um sistema de cache para buscas já realizadas. A melhora no ranqueamento, por sua vez, pode se dar com uma análise do perfil de busca dos usuários do aLine, para predizer o significado das buscas e, a partir disso, melhor ranquear os documentos retornados.

Além disso, a inclusão de um processo de segmentação das notícias de jornais também diminui o ruído no ranqueamento, permitindo, ainda, que a busca por imagens seja adicionada. Assim, será possível encontrar imagens semelhantes no acervo ou buscar termos que estão relacionados às imagens que acompanham as notícias. Unindo-se a busca por imagens com o detector de entidades nomeadas, será possível realizar busca de imagens de pessoas.

Outra sugestão é inserir novos tipos de acervos, como base de partituras, arquivos de áudio, vídeos, códigos de programas, acrescentando mais classes de entidades nomeadas.

# Referências

Mining text data. In: AGGARWAL, C. C.; ZHAI, C. (Ed.). Boston, MA: Springer, US, 2012. cap. Information Extraction from Text, p. 11–41.

Apache. *Apache Tika - a Content Analysis Toolkit.* 2017. <https://tika.apache.org/>.

Badue, C. et al. Capacity planning for vertical search engines. *arXiv preprint arXiv:1006.5059*, 2010.

Badue, C. et al. Basic Issues on the Processing of Web Queries. In: *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. New York, NY, USA: ACM, 2005. (SIGIR '05), p. 577–578.

Baeza-Yates, R. et al. The impact of caching on search engines. In: ACM. *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*. Amsterdam, The Netherlands, 2007. p. 183–190.

Baeza-Yates, R.; Ribeiro-Neto, B. *Modern Information Retrieval: The Concepts and Technology behind Search (ACM Press Books)*. 2011.

Bassil, Y.; Alwani, M. OCR Post-Processing Error Correction Algorithm Using Google ' s Online Spelling Suggestion. *Journal of Emerging Trends in Computing and Information Sciences*, v. 3, n. 1, p. 90–99, 2012.

Botelho, F. C.; Ziviani, N. Near-optimal Space Perfect Hashing Algorithms. *The thesis of PhD. in Computer Science of the Federal University of Minas Gerais*, 2008.

Braschler, M.; RIPPLINGER, B. How effective is stemming and decompounding for german text retrieval? *Information Retrieval*, Springer, v. 7, n. 3-4, p. 291–316, 2004.

Brin, S.; Page, L. The Anatomy of a Large-scale Hypertextual Web Search Engine. *Comput. Netw. ISDN Syst.*, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, v. 30, n. 1-7, p. 107–117, abr. 1998. ISSN 0169-7552.

Cavaca, A. G. et al. Sistema aLine: Experiência Interdisciplinar para Observar a Saúde em Jornais Digitais. In: *7º Congresso Brasileiro de Ciências Sociais e Humanas em Saúde*. Universidade Federal do Mato Grosso, Cuiabá, Mato Grosso, Brasil: ABRASCO, 2016. p. 147.

Cavaca, A. G.; Antunes, M. N.; Nogueira, M. Comunicação, Informação e Saúde: Estratégia Interdisciplinar para Observar a Saúde em Jornais Digitais. In: *Memorias del XIII Congreso Latinoamericano de Investigadores de la Comunicación | Comunicación y Salud*. Cidade do México, México: ALAIC, 2016. p. 13–20. ISSN 2179-7617.

Crof, W. B.; Metzler, D.; Strohman, T. *Search Engines: Information Retrieval in Practice*. Indiana, United States: Addison-Wesley Reading, 2015.

De Souza, F. P.; Ciarelli, P. M.; Oliveira, E. de. Combinando Fatores de Ponderação para melhorar a classificação de Textos. *Anais do Computer on the Beach*, p. 32–41, 2014.

Furnas, G. W. et al. Information Retrieval Using a Singular Value Decomposition Model of Latent Semantic Structure. In: *Proceedings of the 11th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. New York, NY, USA: ACM, 1988. (SIGIR '88), p. 465–480.

Galowicz, J. *C++ 17 STL Cookbook*. Birmingham, United Kingdom: Packt Publishing Ltd, 2017.

Gomes, D.; Silva, M. J. Tarântula-sistema de Recolha de Documentos da Web. In: *CRC'01 - 4ª conferência de Redes de Computadores*. Covilhã, Portugal: CRC'01 - 4ª Conferência de Redes de Computadores, 2001.

Google. *Google Enterprise Search - Fast and Efficient*. 2018. <https://enterprise.google.com/search/>.

Grimes, C.; Tang, D.; Russell, D. M. Query Logs Alone are not Enough. In: CITESEER. *Workshop on query log analysis at WWW*. Banff, Alberta, Canada, 2007.

Guimarães, G. T.; MEIROSE, M. V.; MORAES, S. M. W. ngramas de caractere como técnica de normalização morfológica para língua portuguesa : um estudo em categorização de textos . In: *Anais do X Simpósio Brasileiro de Tecnologia da Informação e da Linguagem Humana*. Porto Alegre, RS, Brasil: SBC, 2015. p. 211–220. Disponível em: <https://portaldeconteudo.sbc.org.br/index.php/stil/article/view/3983>.

Hearst, M. *Search user interfaces*. Cambridge, United Kingdom: Cambridge University Press, 2009.

Joachims, Thorsten. Optimizing Search Engines Using Clickthrough Data. In: *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, NY, USA: ACM, 2002. (KDD '02), p. 133–142.

Kraska, T. et al. The Case for Learned Index Structures. *arXiv preprint arXiv:1712.01208*, 2017.

Manku, G. S.; Jain, A.; Das Sarma, A. Detecting Near-duplicates for Web Crawling. In: *Proceedings of the 16th International Conference on World Wide Web*. New York, NY, USA: ACM, 2007. (WWW '07), p. 141–150.

Mendes, C. A.; de Moura, E. S.; Ziviani, N. Expansão de Consultas Utilizando Indexação Semântica Latente. In: *SBBD*. Gramado, Rio Grande do Sul, Brazil: UFRGS, 2002. p. 166–180.

Microsoft. *Windows Search Overview (Windows)*. 2018. <https://msdn.microsoft.com/pt-br/library/windows/desktop/aa965362(v=vs.85).aspx>.

Nielsen, J. Response times: the three important limits. *Usability Engineering*, Academic Press, 1993.

Nogueira, M. de A.; Oliveira, E. de. Estratégias de Correção de Erros de Extratores de Palavras em Português. In: *Proceedings 5nd Symposium on knowledge Discovery, mining and learning(KDMile)*. Uberlândia, Minas Gerais: Faculdade de Computação – FACOM-UFU, 2017. p. 145–152.

Núcleo Interinstitucional de Linguística Computacional. *Recursos Lexicais*. 2004. <http://www.nilc.icmc.usp.br/nilc/projects/unitex-pb/web/recursos.html>.

Oliveira, E.; Branquinho Filho, D. Automatic Slassification of Journalistic Documents on the Internet. *Transinformação*, SciELO Brasil, v. 29, n. 3, p. 245–255, 2017.

Oliveira, E. et al. Recommending groups to users based both on their textual and image posts. In: SCITEPRESS-SCIENCE AND TECHNOLOGY PUBLICATIONS, LDA. *Proceedings of the International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management*. Porto, Portugal, 2016. p. 315–320.

Pirovani, J.; Nogueira, M.; Oliveira, E. de. Indexing Names of Persons in a Newspaper Large Dataset. *Computational Processing of the Portuguese Language*, Springer, 2018.

Qiu, Y.; Frei, H.-P. Concept Based Query Expansion. In: *Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. New York, NY, USA: ACM, 1993. (SIGIR '93), p. 160–169.

Radlinski, F.; Kurup, M.; Joachims, T. How does Clickthrough Data Reflect Retrieval Quality? In: *Proceedings of the 17th ACM conference on Information and knowledge management*. New York, NY, USA: ACM, 2008. (CIKM '08), p. 43–52.

Ribeiro-Neto, B. A.; Barbosa, R. A. Query Performance for Tightly Coupled Distributed Digital Libraries. In: *Proceedings of the third ACM conference on Digital libraries*. New York, NY, USA: ACM, 1998. (DL '98), p. 182–190.

Robertson, S. Understanding Inverse Document Frequency: on Theoretical Arguments for IDF. *Journal of documentation*, Emerald Group Publishing Limited, v. 60, n. 5, p. 503–520, 2004.

Salton, G.; Wong, A.; Yang, C.-S. A Vector Space Model for Automatic Indexing. *Communications of the ACM*, ACM, v. 18, n. 11, p. 613–620, 1975.

Saraiva, P. C. et al. Rank-preserving two-level caching for scalable search engines. In: ACM. *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*. New Orleans, Louisiana, USA, 2001. p. 51–58.

Soares, M. V. B.; PRATI, R. C.; MONARD, M. C. Improvement on the porter's stemming algorithm for portuguese. *IEEE Latin America Transactions*, IEEE, v. 7, n. 4, p. 472–477, 2009.

Tolosa, G. H. *Técnicas de caching de intersecciones en motores de búsqueda*. Tese (Doutorado), 2016.

Unitex. *Unitex/GramLab*. 2016. <http://unitexgramlab.org/pt>.

YaCy. *YaCy - The Peer to Peer Search Engine: Home*. 2018. <https://yacy.net/en/index.html>.

Ziviani, N. et al. *Projeto de algoritmos: com implementações em Pascal e C*. São Paulo, Brasil: Thomson, 2004. v. 2.

# Apêndices

| Número de termos | Média | Desvio padrão |
|---|---|---|
| 1 | 0.120 | 0.141 |
| 2 | 0.123 | 0.106 |
| 3 | 0.088 | 0.082 |
| 4 | 0.077 | 0.066 |
| 5 | 0.072 | 0.059 |
| 6 | 0.077 | 0.062 |
| 7 | 0.082 | 0.064 |
| 8 | 0.091 | 0.065 |
| 9 | 0.103 | 0.070 |
| 10 | 0.111 | 0.069 |
| Média | 0.095 | 0.084 |

Tabela 8 – Tempo médio das buscas em segundos em Cosseno.

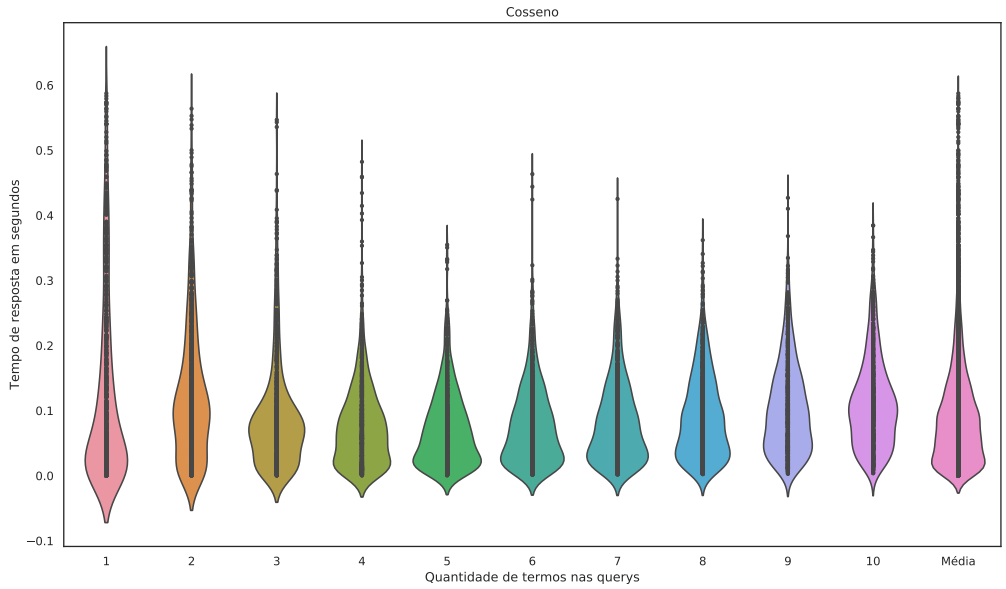



Figura 15 – Distribuição do tempo médio em Cosseno.

| Número de termos | Média | Desvio padrão |
|---|---|---|
| 1 | 0.155 | 0.160 |
| 2 | 0.174 | 0.151 |
| 3 | 0.100 | 0.101 |
| 4 | 0.077 | 0.068 |
| 5 | 0.070 | 0.061 |
| 6 | 0.075 | 0.060 |
| 7 | 0.078 | 0.059 |
| 8 | 0.088 | 0.061 |
| 9 | 0.097 | 0.062 |
| 10 | 0.106 | 0.064 |
| Média | 0.101 | 0.096 |

Tabela 9 – Tempo médio das buscas em segundos em Euclidiana.

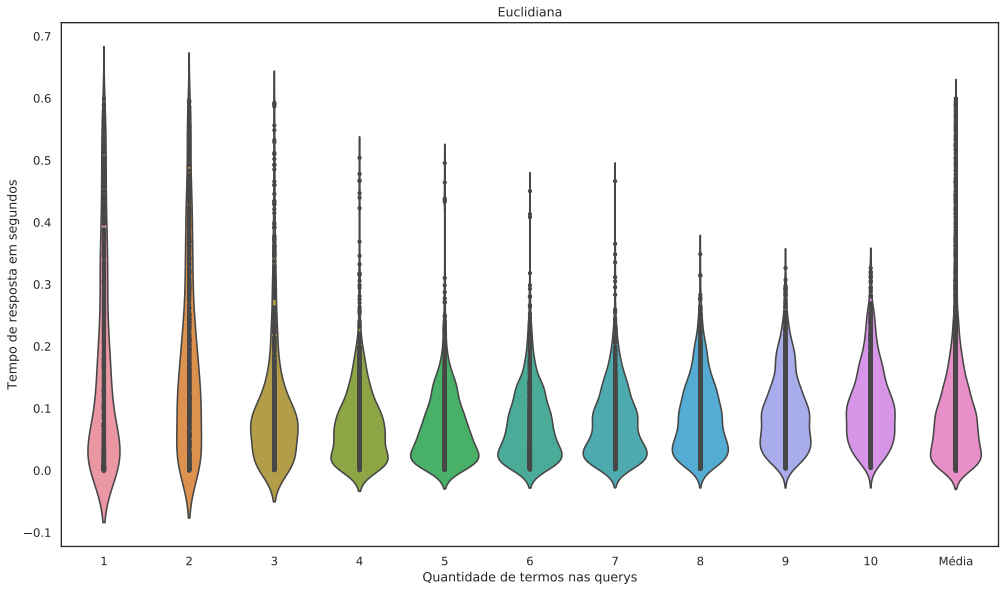



Figura 16 – Distribuição do tempo médio em Euclidiana.

| Número de termos | Média | Desvio padrão |
|---|---|---|
| 1 | 0.118 | 0.139 |
| 2 | 0.123 | 0.106 |
| 3 | 0.082 | 0.073 |
| 4 | 0.072 | 0.059 |
| 5 | 0.068 | 0.054 |
| 6 | 0.073 | 0.055 |
| 7 | 0.074 | 0.054 |
| 8 | 0.085 | 0.057 |
| 9 | 0.095 | 0.060 |
| 10 | 0.103 | 0.061 |
| Média | 0.089 | 0.079 |

Tabela 10 – Tempo médio das buscas em segundos em Manhattan.

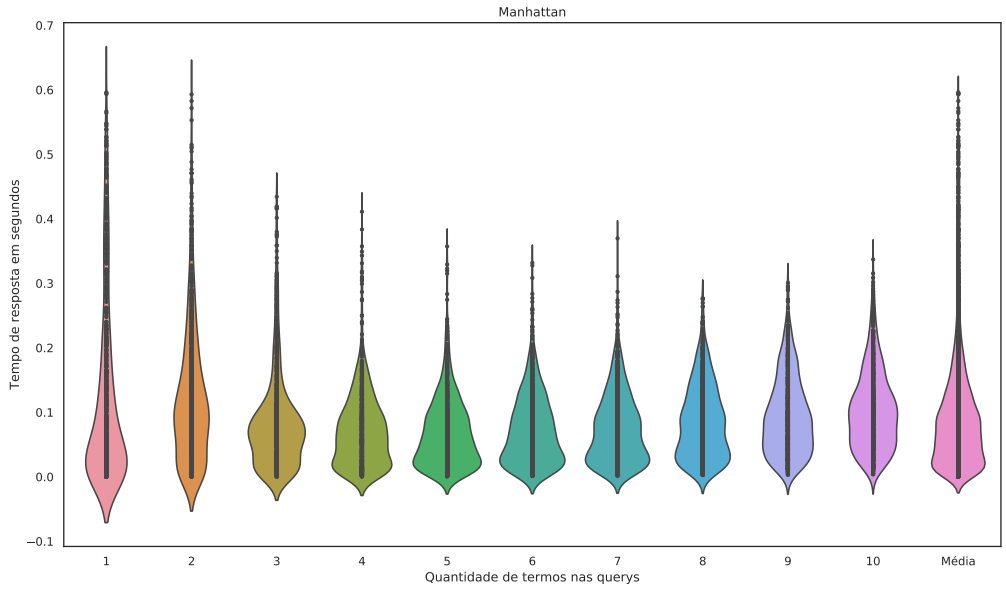



Figura 17 – Distribuição do tempo médio em Manhattan.

| Número de termos | Média | Desvio padrão |
|---|---|---|
| 1 | 0.120 | 0.142 |
| 2 | 0.119 | 0.101 |
| 3 | 0.082 | 0.074 |
| 4 | 0.071 | 0.059 |
| 5 | 0.069 | 0.056 |
| 6 | 0.073 | 0.056 |
| 7 | 0.076 | 0.056 |
| 8 | 0.086 | 0.060 |
| 9 | 0.095 | 0.061 |
| 10 | 0.105 | 0.064 |
| Média | 0.090 | 0.080 |

Tabela 11 – Tempo médio das buscas em segundos em Tanimoto.

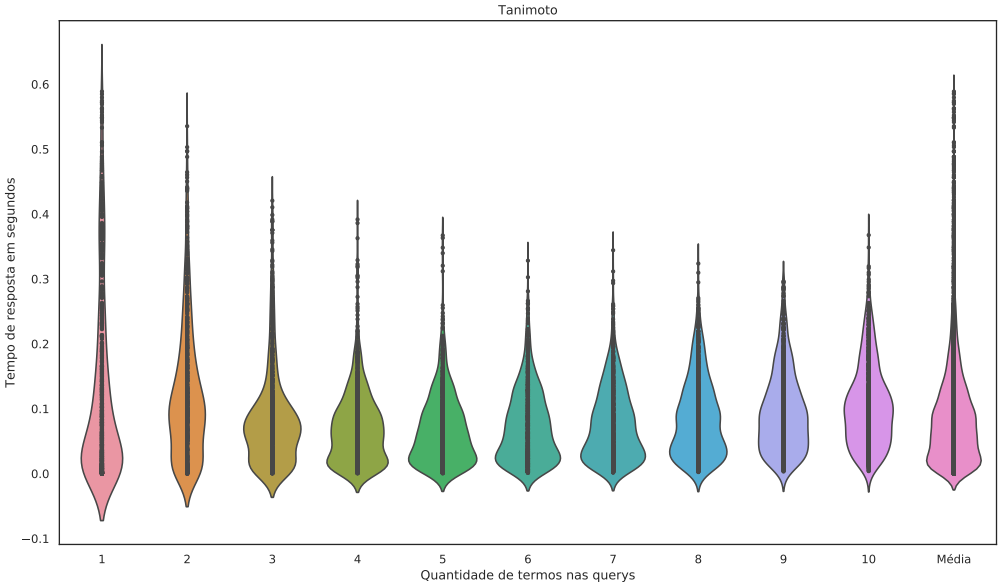



Figura 18 – Distribuição do tempo médio em Tanimoto.

| Número de termos | Média | Desvio padrão |
|---|---|---|
| 1 | 0.157 | 0.161 |
| 2 | 0.169 | 0.149 |
| 3 | 0.099 | 0.098 |
| 4 | 0.078 | 0.073 |
| 5 | 0.070 | 0.063 |
| 6 | 0.073 | 0.058 |
| 7 | 0.076 | 0.056 |
| 8 | 0.085 | 0.057 |
| 9 | 0.095 | 0.060 |
| 10 | 0.106 | 0.064 |
| Média | 0.099 | 0.096 |

Tabela 12 – Tempo médio das buscas em segundos em TF-IDF.

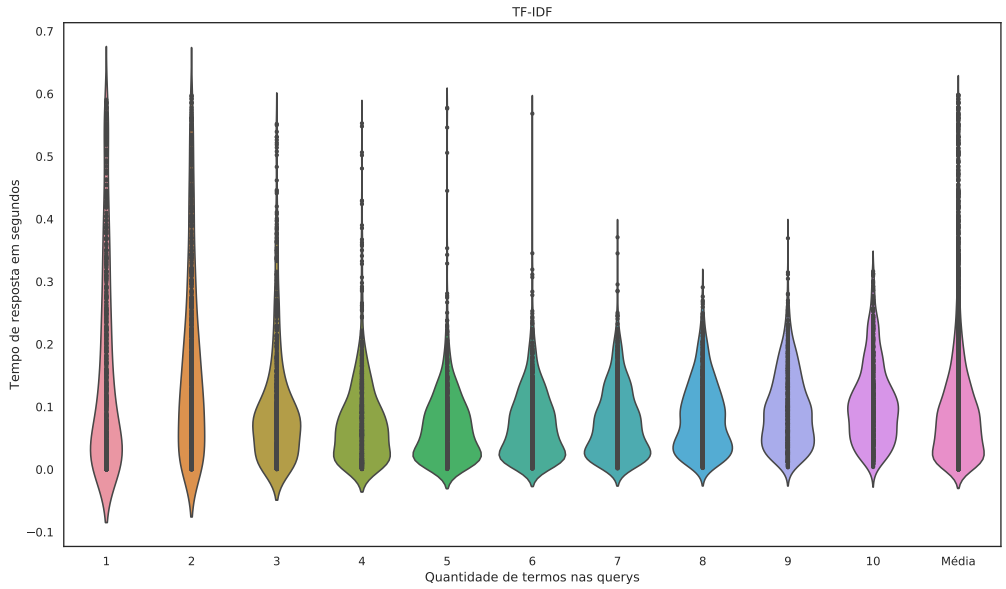



Figura 19 – Distribuição do tempo médio em TF-IDF.

| *Tamanho da Query* \ *Rank* | 10 | 20 | 30 | 40 | 50 | 60 | 70 | 80 | 90 | 100 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 10.89 | 10.53 | 10.48 | 10.25 | 10.21 | 9.38 | 9.62 | 9.93 | 9.5 | 9.22 |
| 2 | 12.78 | 10.6 | 9.54 | 9.9 | 10.03 | 9.46 | 9.21 | 9.63 | 9.27 | 9.57 |
| 3 | 19.54 | 12.24 | 10.23 | 9.59 | 9.05 | 8.49 | 8.31 | 8.12 | 7.24 | 7.2 |
| 4 | 30.42 | 14.17 | 10.53 | 9.19 | 7.61 | 6.16 | 6.1 | 5.55 | 4.94 | 5.33 |
| 5 | 43.99 | 16.19 | 9.09 | 6.98 | 5.35 | 4.16 | 4.04 | 3.67 | 3.09 | 3.45 |
| 6 | 56.41 | 17.0 | 7.48 | 5.15 | 4.04 | 2.7 | 1.99 | 1.81 | 1.83 | 1.59 |
| 7 | 66.32 | 15.49 | 5.54 | 3.48 | 2.6 | 1.75 | 1.71 | 1.1 | 1.22 | 0.79 |
| 8 | 72.2 | 14.5 | 5.36 | 2.52 | 1.7 | 1.23 | 1.03 | 0.52 | 0.49 | 0.45 |
| 9 | 77.88 | 13.79 | 4.0 | 1.39 | 0.86 | 0.67 | 0.32 | 0.3 | 0.39 | 0.39 |
| 10 | 82.16 | 10.99 | 3.76 | 1.14 | 0.79 | 0.42 | 0.24 | 0.24 | 0.16 | 0.11 |

Tabela 13 – Tabela de distribuição dos experimentos em porcentagem pelo tamanho da *query* em Cosseno.

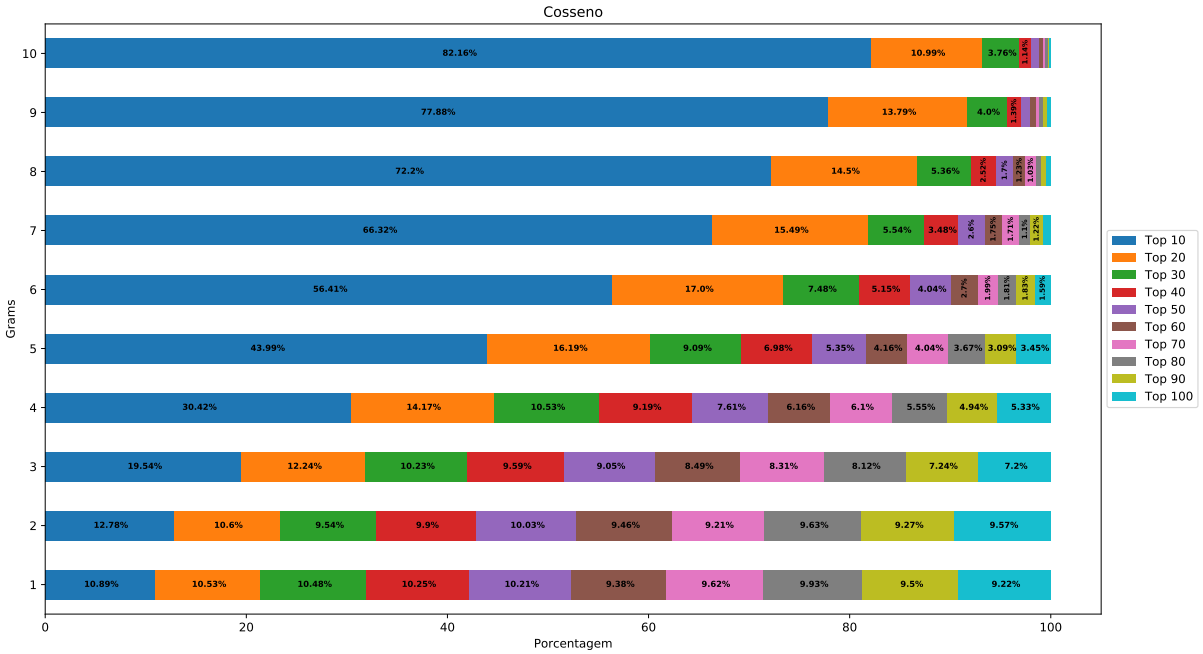



Figura 20 – Distribuição do experimento 2 em Cosseno.

| *Tamanho da Query* \ *Rank* | 10 | 20 | 30 | 40 | 50 | 60 | 70 | 80 | 90 | 100 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 10.87 | 10.13 | 10.02 | 10.34 | 9.78 | 9.33 | 9.61 | 10.1 | 9.87 | 9.95 |
| 2 | 13.03 | 11.4 | 10.16 | 9.97 | 8.75 | 9.44 | 9.26 | 9.27 | 9.28 | 9.44 |
| 3 | 20.61 | 12.58 | 9.96 | 9.89 | 9.05 | 8.27 | 7.65 | 8.37 | 6.8 | 6.83 |
| 4 | 32.38 | 15.21 | 10.94 | 8.79 | 6.54 | 5.68 | 5.23 | 5.31 | 4.87 | 5.06 |
| 5 | 47.02 | 16.98 | 8.84 | 6.66 | 5.5 | 4.04 | 3.14 | 2.75 | 2.52 | 2.54 |
| 6 | 58.31 | 16.9 | 7.22 | 5.02 | 3.8 | 2.36 | 1.88 | 1.6 | 1.27 | 1.64 |
| 7 | 67.7 | 15.5 | 5.89 | 3.44 | 2.39 | 1.34 | 1.01 | 1.09 | 0.81 | 0.81 |
| 8 | 73.57 | 14.31 | 5.32 | 2.54 | 1.53 | 0.86 | 0.49 | 0.54 | 0.43 | 0.41 |
| 9 | 78.66 | 13.26 | 3.84 | 1.74 | 0.74 | 0.46 | 0.3 | 0.23 | 0.42 | 0.35 |
| 10 | 83.27 | 10.3 | 3.47 | 1.32 | 0.61 | 0.24 | 0.29 | 0.21 | 0.21 | 0.08 |

Tabela 14 – Tabela de distribuição dos experimentos em porcentagem pelo tamanho da *query* em Euclidiana.

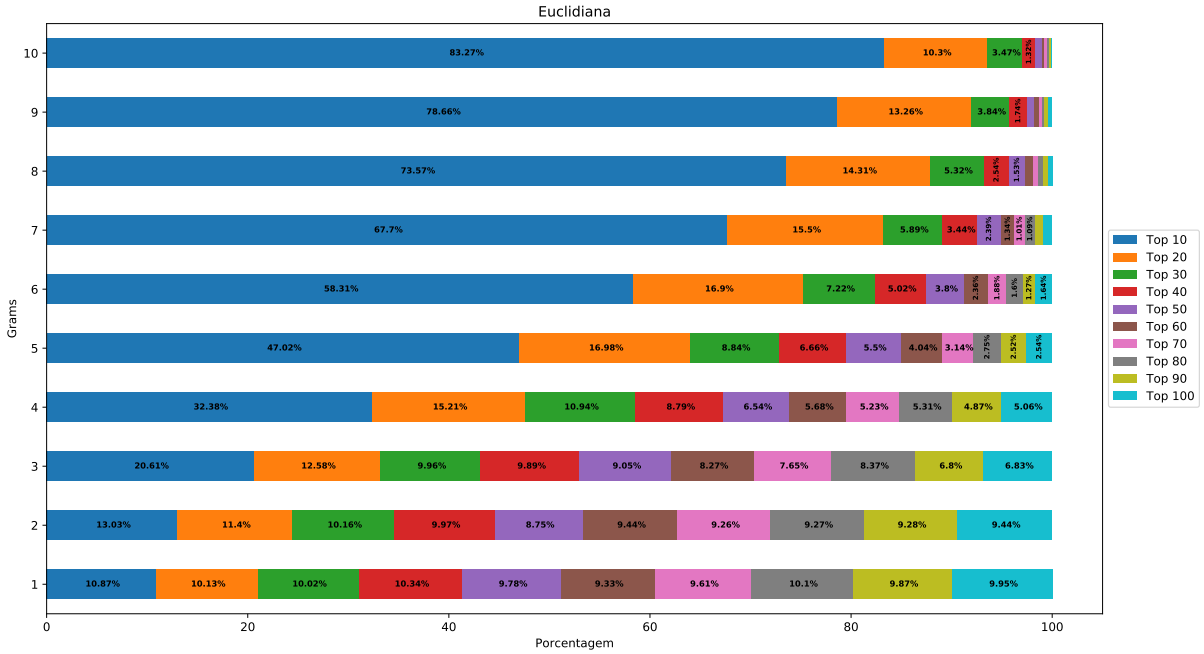



Figura 21 – Distribuição do experimento 2 em Euclidiana.

| *Tamanho da Query* \ *Rank* | 10 | 20 | 30 | 40 | 50 | 60 | 70 | 80 | 90 | 100 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 10.62 | 9.69 | 9.85 | 9.89 | 9.91 | 9.9 | 9.6 | 10.03 | 9.97 | 10.52 |
| 2 | 13.46 | 10.96 | 10.36 | 9.9 | 9.55 | 9.35 | 8.92 | 8.62 | 9.59 | 9.3 |
| 3 | 20.22 | 12.65 | 10.04 | 9.97 | 8.62 | 8.41 | 8.15 | 7.56 | 7.11 | 7.28 |
| 4 | 31.6 | 15.23 | 10.54 | 8.39 | 7.09 | 5.91 | 5.83 | 5.75 | 4.8 | 4.86 |
| 5 | 45.99 | 16.91 | 8.71 | 6.44 | 5.32 | 4.42 | 3.22 | 3.15 | 3.13 | 2.7 |
| 6 | 58.19 | 16.92 | 7.8 | 4.81 | 3.48 | 2.5 | 1.86 | 1.64 | 1.57 | 1.22 |
| 7 | 67.37 | 15.98 | 5.62 | 3.31 | 2.19 | 1.44 | 1.01 | 1.2 | 1.09 | 0.79 |
| 8 | 73.29 | 14.5 | 5.29 | 2.41 | 1.61 | 0.75 | 0.65 | 0.6 | 0.49 | 0.39 |
| 9 | 78.52 | 13.65 | 3.77 | 1.37 | 0.9 | 0.46 | 0.32 | 0.28 | 0.35 | 0.37 |
| 10 | 83.08 | 10.27 | 3.63 | 1.3 | 0.74 | 0.19 | 0.29 | 0.21 | 0.11 | 0.19 |

Tabela 15 – Tabela de distribuição dos experimentos em porcentagem pelo tamanho da *query* em Manhattan.

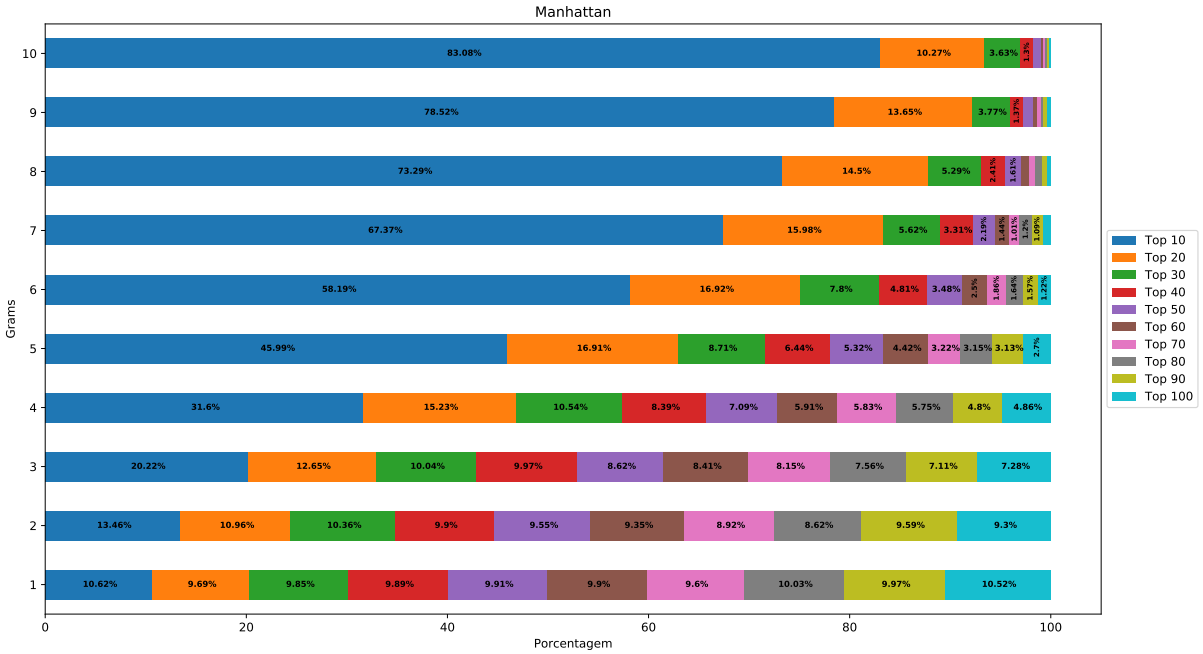



Figura 22 – Distribuição do experimento 2 em Manhattan.

| *Tamanho da Query* \ *Rank* | 10 | 20 | 30 | 40 | 50 | 60 | 70 | 80 | 90 | 100 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 10.63 | 10.35 | 9.92 | 9.74 | 9.77 | 10.08 | 9.57 | 9.92 | 9.95 | 10.06 |
| 2 | 12.16 | 10.25 | 9.85 | 9.39 | 9.91 | 9.51 | 9.6 | 9.47 | 9.65 | 10.19 |
| 3 | 20.23 | 13.16 | 10.39 | 9.44 | 8.65 | 8.49 | 7.93 | 7.15 | 7.3 | 7.26 |
| 4 | 32.27 | 15.87 | 9.73 | 8.09 | 6.45 | 6.17 | 6.06 | 5.22 | 5.23 | 4.92 |
| 5 | 47.32 | 17.23 | 9.05 | 5.92 | 5.24 | 3.84 | 3.45 | 2.72 | 2.72 | 2.5 |
| 6 | 57.76 | 17.49 | 7.99 | 4.51 | 3.0 | 2.65 | 2.02 | 1.73 | 1.32 | 1.52 |
| 7 | 67.32 | 16.7 | 5.47 | 3.07 | 1.93 | 1.47 | 1.22 | 1.2 | 0.87 | 0.75 |
| 8 | 73.98 | 14.63 | 5.01 | 2.11 | 1.42 | 0.77 | 0.69 | 0.41 | 0.32 | 0.65 |
| 9 | 78.36 | 14.14 | 3.89 | 1.32 | 1.0 | 0.37 | 0.3 | 0.21 | 0.16 | 0.25 |
| 10 | 83.21 | 10.86 | 2.97 | 1.3 | 0.85 | 0.48 | 0.24 | 0.05 | 0.03 | 0.03 |

Tabela 16 – Tabela de distribuição dos experimentos em porcentagem pelo tamanho da *query* em Tanimoto.

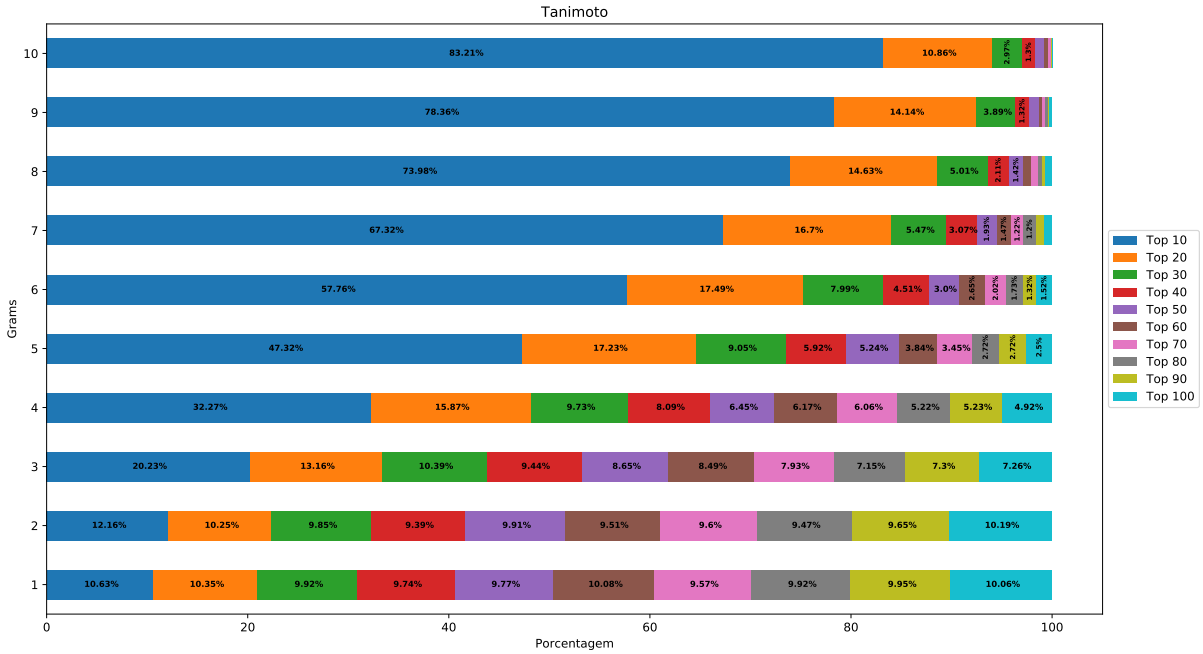



Figura 23 – Distribuição do experimento 2 em Tanimoto.

| *Tamanho da Query* \ *Rank* | 10 | 20 | 30 | 40 | 50 | 60 | 70 | 80 | 90 | 100 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 10.17 | 10.39 | 10.15 | 10.19 | 9.62 | 9.38 | 10.12 | 9.33 | 10.18 | 10.49 |
| 2 | 12.38 | 10.77 | 9.39 | 9.94 | 10.01 | 9.35 | 9.42 | 9.53 | 9.56 | 9.64 |
| 3 | 18.96 | 12.52 | 10.25 | 9.14 | 8.0 | 8.32 | 8.34 | 8.14 | 8.11 | 8.22 |
| 4 | 28.88 | 15.33 | 10.63 | 8.43 | 6.76 | 6.48 | 6.23 | 6.01 | 5.85 | 5.41 |
| 5 | 42.97 | 16.18 | 9.62 | 6.69 | 6.1 | 4.75 | 4.2 | 3.54 | 2.7 | 3.25 |
| 6 | 55.26 | 16.58 | 7.64 | 4.88 | 3.87 | 2.97 | 2.47 | 2.28 | 2.09 | 1.97 |
| 7 | 65.27 | 16.06 | 5.78 | 3.63 | 2.47 | 1.75 | 1.34 | 1.57 | 1.22 | 0.92 |
| 8 | 72.3 | 14.59 | 4.84 | 2.58 | 1.74 | 1.16 | 1.08 | 0.65 | 0.65 | 0.41 |
| 9 | 78.04 | 13.52 | 3.82 | 1.74 | 1.09 | 0.67 | 0.46 | 0.14 | 0.3 | 0.23 |
| 10 | 82.39 | 10.88 | 3.47 | 0.98 | 1.03 | 0.53 | 0.4 | 0.21 | 0.11 | 0.0 |

Tabela 17 – Tabela de distribuição dos experimentos em porcentagem pelo tamanho da *query* em TF-IDF.

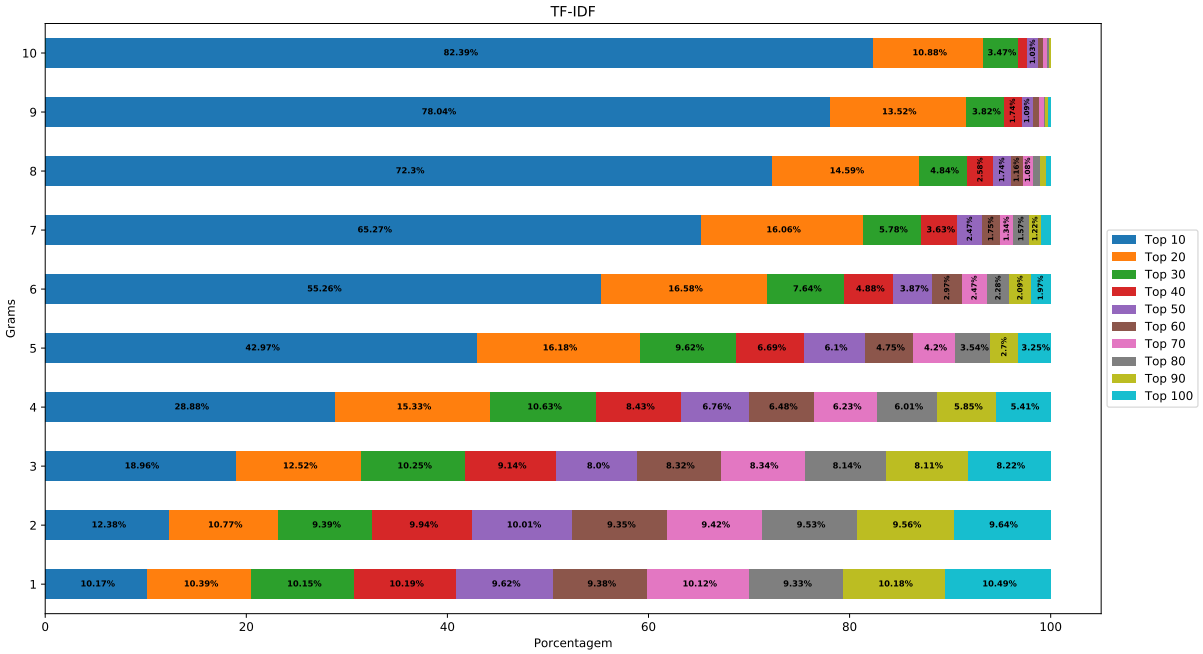



Figura 24 – Distribuição do experimento 2 em TF-IDF.