Uso de Ferramentas Livres para Recuperação de Conteúdos Textuais em Ambientes Big Data

Alternative Title: Open Source Tools Applied to Text Data Recovery in Big Data Environments.

Brunno F. M. Attorre
Universidade Presbiteriana Mackenzie
Rua da Consoloção, 930
São Paulo, SP
brattorre@gmail.com

Leandro A. Silva
Universidade Presbiteriana Mackenzie
Rua da Consoloção, 930
São Paulo, SP
prof.leandro.augusto@mackenzie.br

RESUMO

Com o aumento do volume de dados na Web, a tarefa de construir um mecanismo de busca com alta precisão se torna cada vez mais difícil. Como uma alternativa para melhorar esses resultados, o desenvolvimento de uma ferramenta de recomendação baseada no conteúdo dos documentos a serem buscados pode se tornar bastante útil. Nesse contexto, objetivo desse trabalho é analisar como algoritmos de indexação, Machine Learning e análise textual podem melhorar os resultados de busca e, através do conteúdo buscado em cada documento, construir uma aplicação de busca e recomendação usando as tecnologias Open Source disponíveis no mercado.

Palavras-Chave

Aprendizado de máquina, big data, ferramenta de indexação

ABSTRACT

As the volume of data on the web continue to increase, it is getting more challenging for the search mechanism to find with a high precision rate what the users want to find. As a solution to improve these results, the development of a recommender engine, based on the content of the documents, would prove itself very useful. In this context, this research has the objective to show how the current search and indexing tools could be improved with recommendation, Machine Learning and textual analysis algorithms. The idea behind these project would be to, based on the content of the documents recovered in the search, find similar documents using most of the Open Source technology we have available right now.

SBSI 2015, May 26th-29th, 2015, Goiânia, Goiás, Brazil

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Machine Learning

General Terms

Machine Learning Tools

Keywords

Machine Learning, Index tools, big data

1. INTRODUÇÃO

O desenvolvimento de aplicativos para a indexação e busca de documentos na internet vem se tornando fundamental para satisfação dos usuários, principalmente em aplicações que dependem da utilização desse tipo de recurso como sites de Buscas e de Web Commerce[2].

Essa essencialidade vem, principalmente, do surgimento do conceito de Big Data que emerge a partir do aumento de informações disponíveis publicamente e que exige das aplicações tratar terabytes de dados, na maioria dos casos sem estruturas bem definidas (dados não-estruturados), provocando com isso o aparecimento de novas tecnologias e o avanço em pesquisa sobre mineração de dados[3] .

Também temos, como influência para essa mudança no cenário de recuperação de dados, a necessidade que as pessoas de hoje em dia tem com a Internet. Nos tempos atuais .Não é mais necessário termos um desktop ou uma máquina para acessar a Internet, podemos fazer nossas buscas de qualquer lugar a qualquer momento com smartphone ou tablets. Ou seja, a portabilidade para o acesso provoca aumento na frequência de uso por buscadores e, consequentemente, na geração de dados.

Devido à estes dois efeitos, acesso e volume, o tempo de consulta resultante de um banco relacional não é mais suficiente para ferramentas de busca online, fazendo com que, recentemente, diversos conceitos e algoritmos de indexação como, por exemplo, o uso Índice Invertido, emergissem como propostas para diminuir a latência na busca de informações [4].

Junto com esses problemas de velocidade e do grande volume de dados, o gap semântico dos resultados de busca, ou seja, a diferença entre o que o usuário busca e a resposta do sistema, se torna cada vez mais difícil de ser tratado pelos

meios tradicionais. Diversos trabalhos no meio acadêmico já buscaram diminuir essa gap usando técnicas como clusters de dados [7] e combinação dos mais variados algoritmos [11].

Nessa pesquisa, iremos estudar a utilização de ferramentas de Big Data e de Aprendizado de Máquina para conseguir uma maior precisão de resultados e para minimizar esse gap.

Nesse contexto, o objetivo do trabalho será apresentar uma alternativa para melhorar o resultado de busca através do uso da experiência do usuário durante o uso dos sistema a partir de ferramentas livres para indexação associadas as de recomendação.

Para a validação dos resultados, definiu-se como base de dados o conteúdo indexado diariamente por um feed RSS onde, a cada dia, as notícias de 5 temas diferentes eram indexadas. Com o conteúdo dessa base, foi construído um módulo de busca textual e outro de aprendizado de máquina para que, com a aplicação de técnicas de recomendação e classificação, fosse possível apresentar ao usuário conteúdos semelhantes, analisando os resultados finais com diferentes números de documentos.

Na parte técnica, foram utilizados os pacotes Apache Hadoop juntamente com o Apache Mahout para a criação da ferramenta de recomendação e armazenamento da base de aprendizado e, para as notícias, foi utilizado o Apache Solr de forma a indexar os documentos.

Além da introdução, este trabalho está dividido em um tópico de trabalhos relacionados, que apresenta as principais técnicas e ferramentas de indexação e recomendação que foram usadas na pesquisa, uma parte de metodologia experimental, explicando sobre o sistema criada para avaliar o experimento e, por fim, uma análise dos resultados com uma conclusão final sobre a pesquisa.

2. TRABALHOS RELACIONADOS

Na literatura, alguns autores discutem o uso de técnicas inteligentes para o tratamento de conteúdos textuais como estratégia para aprimorar os resultados de sistemas de buscas com uso de mineração de dados e inteligência artificial.

BÜTTCHER, CLARKE e CORMACK [4] estudaram técnicas de busca aplicadas a conteúdos textuais, apresentando e avaliando cada uma das ferramentas existente. O estudo se torna bastante interessante, principalmente relacionado a técnicas de utilização do algoritmo de Índice Invertido e de tratamento textual, apresentando variações de implementação dessa técnica e demonstrando maneiras de verificar sua eficácia.

ANTONELLIS e GALLOPOULOS [1] , por outro lado, apresentam uma análise elaborada em cima do algoritmo de Matriz de Termos, mostrando uma visão extremamente útil para seu uso dentro da avaliação dos textos e sua utilização em algoritmos de Inteligência Artificial e Aprendizado de Máquina.

Seus experimentos se baseiam no uso de uma combinação de algoritmos de Indexação Semântica Latente e o uso de um modelo vetor espacial para representar seus dados, com a aplicação de uma Matriz de Termos em memória para melhorar o desempenho dos algoritmos. Os resultados se apresentam extremamente positivos, onde para uma coleção de 10330 documentos, divididos em 5526 termos, se obteve uma precisão média de 30%.

MARLIN [6] analisa a utilização de algoritmos de mineração de dados para a criação de filtros colaborativos e como aplicar o conceito de Aprendizado de Máquina pode ser utilizado para a criação de sistemas adaptativos.

Entre essas análises, uma das mais relevantes à nossa pesquisa é referente a precisão do algoritmo de Naive Bayes tradicional aplicado a análise de dados de uma rede social para filmes. Analisando um conjunto de pontuações de 30000 usuários, os autores chegaram a atingir uma precisão média de 50% com o uso do classificador, uma porcentagem bastante positiva.

METEREN e SOMEREN [10] apresenta em seu estudo uma análise da utilização de filtros de conteúdo para a criação de ferramentas de recomendação, se tornando extremamente útil principalmente para conteúdos textuais.

Em sua pesquisa, apresentam a criação de um Framework para recomendação baseado na criação de um filtro de conteúdo, que foi criado pelos próprios autores. Utilizando técnicas como a frequência de termos dos documentos e a utilização do feedback do usuário para ajustar o sistema de ML, atingiram resultados interessantes com uma média de 60 a 70% de precisão na recuperação dos documentos.

A aplicação de algoritmos de classificação também é estudada no trabalho de PENNACCHIOTTI e POPESCU [8], realizado pelo Yahoo Labs. Nesse experimento, a aplicação de algoritmos de classificação é feita em cima de uma base de usuários do Twitter, onde se tenta classificar os usuários dentro de categorias relacionadas ao seu direcionamento político.

Apesar de propor uma solução para um problema diferente do que iremos analisar no artigo, diversas ideias interessantes surgem dessa pesquisa, principalmente relacionadas ao modo de como tratar a informação capturada da rede social e as formas de traçar o perfil de cada usuário, que se provou extremamente útil na formulação do módulo de tratamento de texto desse artigo.

Como algoritmo de Machine Learning, a escolha do autor foi o Gradient Boosted Decision Trees, ou GBDT, e, como resultados, foi possível atingir uma precisão média de 88% na classificação de 10338 usuários.

Finalmente, RENNIE e colaboradores [9] trabalharam na otimização do algoritmo de classificação Naive Bayes. Com este estudo foi possível a criação de um algoritmo baseado no classificador original de Naive Bayes, porém sem os pontos negativos do mesmo, otimizando-o ainda mais para classificação de textos tradicionais.

Esse algoritmo modificado se mostra extremamente eficaz para a classificação de conteúdos textuais, sendo que nos experimentos realizados durante a pesquisa de RENNIE e colaboradores, foram alcançados entre 69.4% a 88.7% na classificação de uma base de notícias do site Reuters com 21578 artigos.

3. METODOLOGIA EXPERIMENTAL

Nesse experimento científico foi proposto a criação de um framework de aprendizado de máquina capaz de, através de uma base de notícias previamente alimentada ao algoritmo de classificação, encontrar documentos semelhantes através da análise textual de conteúdos de notícias.

Essa estratégia foi adotada, pois, através da recomendação de conteúdos similares aos exibidos para o usuário após sua busca, nos baseando nos conteúdos clicados, ou seja, visualizados pelo usuário, conseguimos aprimorar a experiência de navegação do mesmo, ao mostrar o conteúdo que se deseja encontrar logo na primeira página, e aumentar a probabilidade de encontrar outra notícia relacionada a que estava

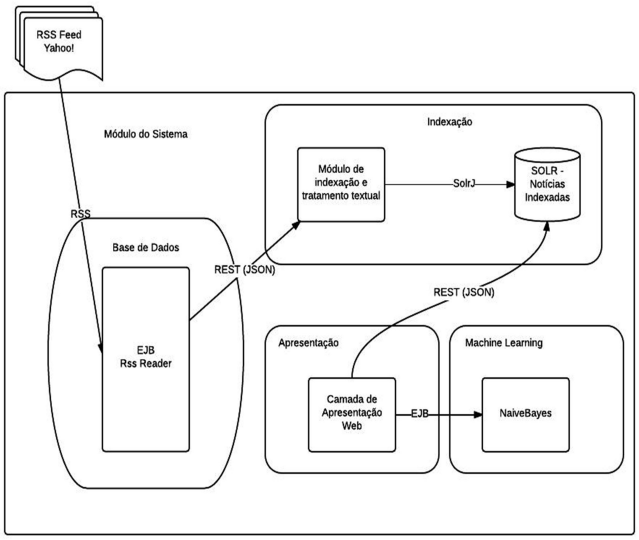

Figura 1: Módulos do sistema usado no experimento.

pesquisando em primeiro lugar, consequentemente minimizando o gap semântico de sua busca.

Também, dentro da elaboração do framework para o desenvolvimento experimental, houve a preocupação na criação de módulos de baixo acoplamento, podendo ser facilmente substituídos por outras implementações ou funcionalidades.

Com o objetivo de demonstrar o experimento, foi criado um sistema dividido em 4 módulos principais, conforme apresentado na Figure 1. Nessa figura podemos também notar as entradas e saídas de cada um dos módulos, bem como a tecnologia utilizada para suas comunicações.

As notícias indexadas em nosso experimento são provenientes do site de notícias www.yahoo.com.br, sendo capturadas pelo sistema através do módulo RSS , descrito na figura como Base de Dados, durante um período de 1 mês.

Esse módulo serve como ponto de entrada de documentos para o sistema. Através de uma base de dados previamente alimentada com o endereço de fontes de RSS, é feito diariamente a leitura de notícias dessa base em 5 tópicos arbitrariamente escolhidos: Economia, Internacional, Tecnologia, Ciências e Entretenimento. Lidos esses documentos, cria-se uma entidade do sistema denominada RSSFeed e envia-se todas as entidades não duplicadas para o módulo de tratamento textual e indexação do sistema. O envio para esse módulo é feito através do uso da tecnologia de webservices REST e os dados são serializados no formato JSON.

No módulo indexação, por sua vez, composto pela ferramenta Open Source Apache Solr e que utiliza, para indexar seu conteúdo, o algoritmo de Índice Invertido, ou seja, os documentos são indexados de acordo com seus tokens (palavras do texto). No módulo de Indexação é feito também um tratamento textual para preparar as notícias para serem processadas posteriormente pelo algoritmo de IA. Esse tratamento é feito antes da indexação e armazenamento das notícias no sistema e é composto pelos seguintes passos:

- Eliminação das tags do código HTML: Todas as tags e códigos Java Scripts são eliminados do conteúdo da página. Além disso, é feito uma análise para verificar qual conteúdo realmente faz parte da notícia e não de links ou propagandas de outros documentos. Todo esse tipo de texto é eliminado, sobrando somente o conteúdo textual da notícia, concatenado em um único parágrafo.

- Eliminação das stopwords: É feita a eliminação das denominadas stopwords do texto, ou seja, eliminação de conjunções, artigos, verbos e outras classes gramaticais que não são características para descrever o contexto da notícia a ser analisada. Como critério para a eliminação dessas palavras, foi utilizado a biblioteca da língua portuguesa brasileira do Apache Lucene, permitindo a eliminação das palavras a partir de um dicionário de stopwords previamente criado. Após essa eliminação, também é feito um tratamento para eliminar as palavras que aparecem em mais de 95 por cento dos textos indexados, garantindo ainda mais que as palavras presentes no documento indexado sejam somente as que realmente são referentes ao tema em questão.

- Realização do stemming das palavras: Cada palavra do conteúdo a ser tratado é manipulada de forma a realizar o stemming, ou seja, reduzir todas as palavras para a sua raiz. Isso também foi feito com o uso da biblioteca de stemming para a língua portuguesa presente dentro do Apache Lucene.

Após a realização desses passos, esse conteúdo é armazenado na instância do Solr através da biblioteca de integração SolrJ, que fornece APIs para integração direta do Java com a aplicação Solr.

Agora, com o tratamento textual do conteúdo e o seu armazenamento numa base indexada, cada um dos documentos está mais apropriado para ser tratado pelo algoritmo de Machine Learning, minimizando o gap semântico dos resultados.

No módulo de ML (de Machine Learning) é onde encontra-se a parte principal do projeto, a criação do classificador bayesiano de forma que, a partir de um documento previamente indexado, a base histórica de artigos é percorrida com o objetivo de encontrar o artigo mais semelhante, criando um recomendador com base em um filtro por conteúdo (content-based filtering). A implementação do classificador foi feita com base no Apache Mahout, uma ferramenta Open Source de Machine Learning que implementa os principais algoritmos de aprendizado de máquina existente.

Conforme o trabalho realizado por RENNIE e colaboradores [9], a escolha do algoritmo modificado de Naive Bayes, o Complementary Naive Bayes, para realizar a recomendação foi graças a dois fatores principais. Alta performance devido ao fato do algoritmo, por ser bastante simplificado não exigir grande poder computacional para ser aplicado e, portanto, o tempo de resposta, que no caso de uma busca é bastante crítico, é bastante rápido. E ainda, o algoritmo apresenta facilidade de desenvolvimento sem a necessidade de estabelecer relações entre os elementos.

Sua facilidade de desenvolvimento e uso e a não necessidade de estabelecer relações entre os elementos se mostrou extremamente útil para tratarmos da busca de conteúdos sem ter o conhecimento prévio do que iríamos tratar. Por esse motivo, escolhemos inicialmente somente o uso desse algoritmo deixando para trabalhos futuros a aplicação de outros classificadores de ML dentro do framework.

O carregamento da informação nesse algoritmo, por sua vez, foi feito com base no conteúdo tratado previamente no módulo de indexação, separando seu texto em tokens e agrupando-os em um vetor de frequência de termos (ma-

Tabela 1: Frequência de termos

Frequencia de termos				
Id documento	compr	Avia	econom	escol
1	3	2	0	0
2	2	0	10	0
3	0	0	2	5

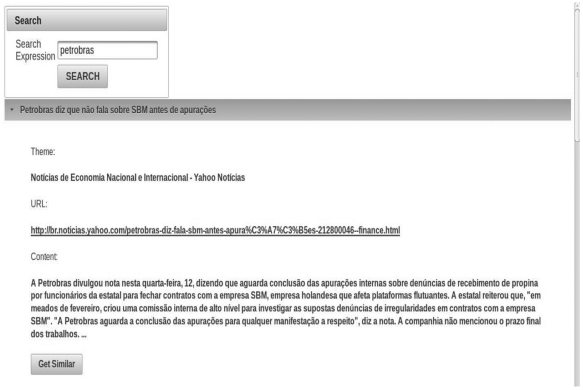

Figura 2: Resultado de buscas com o botão "Get Similar"para buscar recomendações

triz de termos). Esse vetor, por sua vez, era alimentado ao Mahout que, posteriormente, utiliza seu conteúdo para realizar a classificação. Cada vetor também era associado à um identificado (id) respectivo do documento, permitindo assim uma identificação de qual documento cada vetor se refere. Na figura 2, está demonstrada um exemplo de um conjunto de 3 documentos, os quais foram transformados em vetores de termos com seus tokens (palavras), previamente tratados pelo módulo de indexação.

Como podemos ver na Tabela 1, o vetor de frequência de termos possui, para cada documento, o número de aparições que cada palavra (token) tem no documento. Através desses vetores, o algoritmo de Naive Bayes é capaz de classificar o texto e encontrar, dentre a base de treinamento, quais documentos melhor se classificam nesse perfil, atribuindo um peso a cada documento, de acordo com o algoritmo modificado de Naive Bayes.[9]

Dentro da execução da aplicação, uma vez o usuário visualiza uma notícia, tratamos o texto dessa notícia no algoritmo e devolvemos a lista de documentos com os respectivos pesos calculados pelo algoritmo.

Uma vez esses pesos são retornados, a aplicação é responsável por ordenar a lista de resultados pelo valor dos mesmos e mostrar ao usuário os documentos que possuem o maior peso, ou seja, um conteúdo mais semelhante ao conteúdo buscado inicialmente.

Portanto, na camada de apresentação, apresentamos ao usuário tanto os resultados da busca como os artigos mais semelhantes ao conteúdo do resultado em questão. Dessa forma, permitimos que, logo nas primeiros resultados, o usuário seja capaz de ver a notícia que mais se assemelhou ao texto pesquisado e, também, outras notícias de conteúdos similares, tentando dessa forma diminuir o gap semântico do resultado de busca.

O exemplo do módulo Web de apresentação pode ser visto na Figura 2 e 3. Na primeira, apresentamos apenas as notícias provenientes do resultado de busca pela palavra "petrobras". No canto inferior esquerdo, o botão "Get Similar"permite que seja expandido as recomendações de notícias classificadas como semelhantes a notícia sendo visualizada.

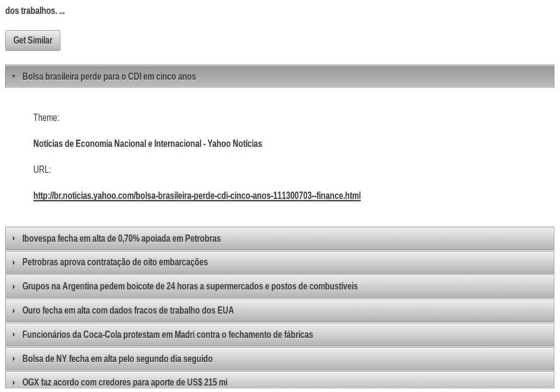

Figura 3: Recomendações de notícias expandidas

Uma vez pressionado o botão, o sistema irá expandir as recomendações conforme pode ser visto na Figura 3, exibindo as mesmas para o usuário.

4. RESULTADOS EXPERIMENTAIS

Como metodologia de análise dos resultados, utilizou-se o método K-Fold (ou K-Pastas). Esse método consiste na divisão da base de dados em K número de subconjuntos de dados de forma randômica, sendo 1 desses subconjuntos utilizado para o teste e os outros K-1 utilizados como base de treinamento para o algoritmo. Em seguida, o procedimento é repetido para todos os outros K subconjuntos, terminando quando todos os subconjuntos tiverem sido utilizados como teste.

O uso do K-Fold apresenta a vantagem de garantir um conjunto dinâmico de dados para o algoritmo de ML, diminuindo problemas ou estatísticas incorretas que podem acontecer ao utilizar um conjunto fixo de dados e permitindo que distribuições irregulares da informação sejam contornadas ao analisar-se mais de um conjunto.

Para avaliar o resultado do algoritmo Bayesiano e o resultado de nossa consulta por notícias semelhantes, foi utilizado o conceito de precisão x revocação. Esse conceito indica, dentro do domínio de amostragem em que foi trabalhado, a porcentagem de artigos de notícias que retornaram dentro do esperado (precisão) a partir de um determinado número de artigos em meu conjunto total (revocação) [5].

Na análise do resultado do algoritmo Naive Bayes implementado foi utilizado a metodologia K-Fold, com K=4, ou seja, 75 por cento da base total era utilizada como treinamento e 25 por cento como conteúdo a ser testado.

Para a formulação dos gráficos de recuperação, foi definido que a precisão da recuperação da informação seria a igualdade entre o tema da notícia original e o tema da notícia recomendada. Em relação a revocação, essa foi indicada pelo número de artigos recomendados, variando de 1 até 10.

4.1 Análise para 605 documentos

Como base de análise, o experimento foi realizado em dois momentos distintos e com uma base crescente de artigos, iniciando pela base de 605 documentos conforme a Figura 4,

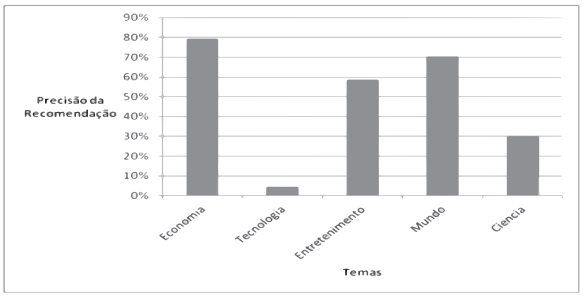

Figura 4: Gráfico de precisão x revocação para 605 artigos

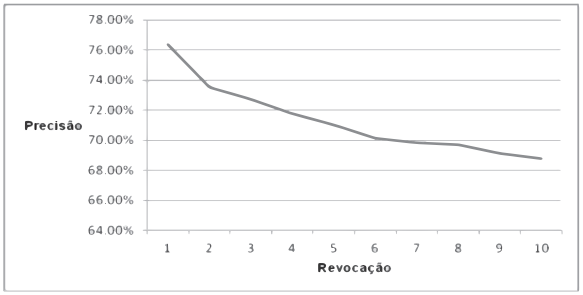

Figura 5: Gráfico de precisão x revocação para 605 artigos

onde apresentamos o gráfico de precisão x revocação. Esse gráfico apresenta, no eixo Y, a porcentagem de precisão, ou seja, o número de artigos recomendados do mesmo tema que o artigo e original (caso o tema seja diferente, consideramos como uma recomendação errada) e, no eixo X, a quantidade de recomendações requisitadas ao sistema.

Nessa figura, podemos observar que a curva de precisão do algoritmo se comporta numa decrescente quase que linear entre as recomendações de 1 a 5 artigos de notícias e, após isso, perde-se um pouco a acentuação da queda e se atinge uma menor diminuição de precisão com o aumento do número de artigos recomendados,aproximadamente aos 69%.

Na Figura 5, por sua vez, é apresentado, para o mesmo número de documentos, o número de recomendações corretas divididas por tema. É interessante observar que os temas que possuem menor número de documentos indexados, Tecnologia, por exemplo, tiveram uma performance pior que outros que possuem um número maior, indicando a importância de uma base de dados completa para treinar o algoritmo.

4.2 Análise para 903 documentos

Continuando a análise, para uma base de 903 documentos, já se evidencia uma melhora no algoritmo de recomendação, possuindo melhores resultados no geral. Aumenta-se a porcentagem de precisão para todas as faixas de recomendação, conforme observado na Figura 6.

Outro fator interessante de se observar é que, no geral, grande parte dos temas obtiveram um aumento de precisão de 5 a 10 por cento, com exceção de Economia que, basicamente, se manteve na mesma faixa de 80 por cento conforme Figura 7.

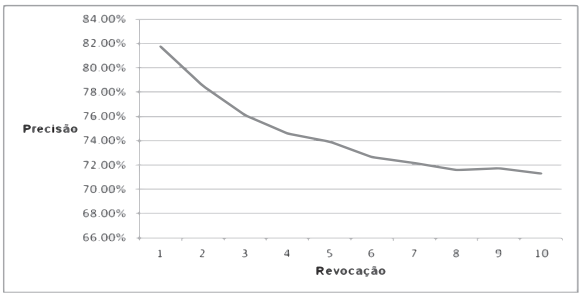

Figura 6: Porcentagem de recomendações corretas para 903 artigos

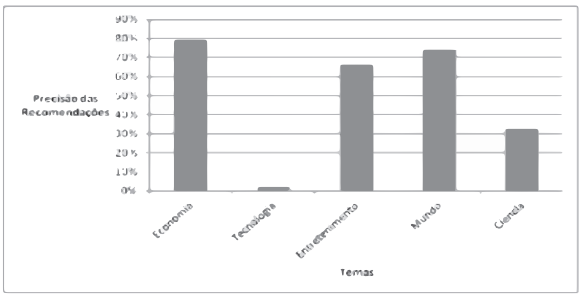

Figura 7: Gráfico de precisão x revocação para 903 artigos

A queda de precisão, semelhante a Figura 2, também se mostra bem acentuada entre 1 a 5 artigos e, após isso, diminui sua intensidade, se estabilizando em, aproximadamente, 70% de precisão, um aumento de 2% em relação a base testada anteriormente.

Também foi construído, com base nos itens recomendados erroneamente, uma matriz de confusão indicando, para cada recomendação errada, qual o tema da noticia recomendada pelo algoritmo, conforme a Tabela 2.

Através dessa matriz, Tabela 2, podemos observar que, dentre as recomendações erradas, o algoritmo se confunde, na maioria das vezes, por temas similares. Um exemplo claro disso está na recomendação de notícias de Tecnologia onde os erros ocorridos foram, na maior parte, em notícias de Ciência, tópicos com notícias correlatas.

Analisando em questão de tempo de resposta, o algoritmo também se mostrou bastante eficiente. Possui, para a base inicial de 605 documentos, um tempo de resposta para a primeira requisição de 1 segundo e, para as consultas subsequentes, 0.6 milissegundos. Isso se deve pelo fato de que, durante a requisição inicial, é carregado o modelo bayesiano previamente construído, o que acaba por aumentar o tempo de resposta.

Com a base de 903 documentos, por sua vez, vemos um aumento para 1.08 segundos para a requisição inicial, porém, para as requisições subsequentes, não vemos aumento considerável (menor que 0.01 segundos).

5. CONCLUSÃO

Através do uso do algoritmo de Naive Bayes, implementado na ferramenta Apache Mahout, conseguimos atingir o objetivo proposto de construir uma ferramenta para minimizar o gap semântico da busca do usuário, capaz de reco-

Tabela 2: Matriz de Confusão para os itens classificados erroneamente.

	Economia	Tecnologia	Entretenimento	Mundo	Ciência
Economia	0%	12%	15%	54%	20%
Tecnologia	9%	0%	17%	0%	74%
Entretenimento	18%	19%	0%	39%	24%
Mundo	40%	7%	24%	0%	30%
Ciencia	27%	21%	17%	36%	0%

mendar documentos semelhantes ao usuário, aprimorando a probabilidade do usuário de encontrar o conteúdo buscado.

Além disso, através da implementação otimizada do algoritmo de Machine Learning e da própria natureza do Apache Mahout, construído em cima da ferramenta Apache Hadoop, foi possível se construir uma ferramenta extremamente performática e escalável, com um tempo de resposta aceitável para a navegação do usuário.

Em relação aos resultados, foi possível observar que o algoritmo se comportou da maneira esperada, obtendo excelentes resultados em relação a precisão e tendo resultados melhores conforme a base de dados aumenta de volume.

Como próximos passos, a evolução desse framework implica a implementação de outros algoritmos diferentes do Naive Bayes e, também, a utilização de paralelização computacional de forma a minimizar o tempo de resposta da recomendação.

6. REFERÊNCIAS

[1] I. Antonellis, E. Gallopoulos, I. Antonellis, and E. Gallopoulos. Exploring term document matrices from matrix models in text mining, 2006.

[2] R. Berman and Z. Katona. The role of search engine optimization in search marketing. *Marketing Science*, 32(4):644–651, 2013.

[3] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. In *Seventh International World-Wide Web Conference (WWW 1998)*, 1998.

[4] S. Buettcher, C. L. A. Clarke, and G. V. Cormack. *Information Retrieval: Implementing and Evaluating Search Engines*. The MIT Press, 2010.

[5] O. N. P. Cardoso. Recuperação de informação. *INFOCOMP Journal of Computer Science*, 2(1):33–38, 2004.

[6] B. Marlin. Collaborative filtering: A machine learning perspective. 2004.

[7] C.-T. Nguyen. Bridging semantic gaps in information retrieval: Context-based approaches. *ACM VLDB*, 10, 2010.

[8] M. Pennacchiotti and A.-M. Popescu. A machine learning approach to twitter user classification, 2011.

[9] J. D. Rennie, L. Shih, J. Teevan, D. R. Karger, et al. Tackling the poor assumptions of naive bayes text classifiers. In *ICML*, volume 3, pages 616–623. Washington DC), 2003.

[10] R. Van Meteren and M. Van Someren. Using content-based filtering for recommendation. In *Proceedings of the Machine Learning in the New Information Age: MLnet/ECML2000 Workshop*, pages 47–56, 2000.

[11] R. Zhao and W. I. Grosky. Narrowing the semantic gap-improved text-based web document retrieval using visual features. *Multimedia, IEEE Transactions on*, 4(2):189–200, 2002.