Web Search Project

Gonçalo Pereira da Silva
Departamento de Informática
Faculdade de Ciências e Tecnologia
Universidade Nova de Lisboa
Nº 45213
gcs.silva@campus.fct.unl.pt

Gonçalo Banha
Departamento de Informática
Faculdade de Ciências e Tecnologia
Universidade Nova de Lisboa
Nº 45429
g.banha@campus.fct.unl.pt

Miguel Almeida
Departamento de Informática
Faculdade de Ciências e Tecnologia
Universidade Nova de Lisboa
Nº 45526
mh.almeida@campus.fct.unl.pt

ABSTRACT

Este relatório aborda a temática dos motores de busca, em particular, no contexto de um problema relacionado com a redes sociais e o tipo de conteúdos que são partilhados nas mesmas. O problema consiste na distribuição de informação (tweets) com base nos interesses de cada utilizador. Desta forma, qualquer utilizador está a par das mais recentes novidades em qualquer campo acerca do qual demonstre interesse. Assim, o projeto consiste no desenvolvimento de métodos análise dos conteúdos publicados num certo dia e posterior distribuição destes mesmos conteúdos, com base nos perfis de interesse de determinado utilizador.

ACM Reference format:
Gonçalo Pereira da Silva, Gonçalo Banha, and Miguel Almeida. 2018. Web Search Project. In *Proceedings of ACM Conference, Washington, DC, USA, July 2017 (Conference'17),* 7 pages.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUÇÃO

Os sistemas de monitorização automática das redes sociais têm vindo a aumentar a sua popularidade devido ao facto destes conseguirem sumarizar a informação que é divulgada nas mesmas, possibilitando assim um acompanhamento mais conveniente das novidades por parte de um utilizador. Deste modo, e no âmbito da unidade curricular de Pesquisa na Web, o projeto a que este relatório faz referência consiste no desenvolvimento e implementação de um sistema que monitoriza as publicações diárias de uma rede social, que no caso concreto deste projeto é o Twitter, e que com base nos interesses definidos pelos utilizadores, a que chamamos de perfis de interesse, é capaz de calcular as publicações mais relevantes para cada perfil de interesse. Calculadas as publicações mais relevantes para cada utilizador, estas são depois distribuídas através de email.

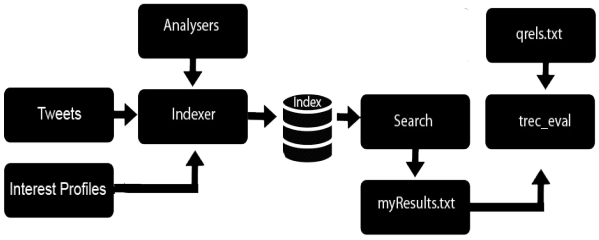
A implementação do projeto contou com o total de três fases, dois checkpoints e uma fase final de projeto. No primeiro checkpoint foi implementada uma versão simples de um motor de pesquisa (suportado pelo Lucene), com o propósito de implementar diferentes tipos de analysers e para os quais foram realizados vários testes de forma a perceber de que forma a variação do tipo de filtros utilizado num analyzer tem ou não influencia no resultado final.

Numa segunda fase, correspondente ao segundo checkpoint do projeto, foi abordado o tema da recuperação de informação tendo em conta certos conceitos como o query expansion e diferentes tipos de similatiry.

Por ultimo, na fase final do projeto a informação e conclusões retiradas das fases anteriores foram utilizadas de forma a conseguir implementar um mecanismo que consiga analisar as publicações efetuadas e reunir as informações mais relevantes e diversificadas para cada tipo de interesse.

2 MÉTODOS E ALGORITMOS

A principal componente nos motores de pesquisa é a informação. Dito isto, passamos a apresentar a arquitetura base:



.

Observando o diagrama apresentado, conseguimos ver que a primeira etapa na criação de um index de informação viável é a partir da utilização de um indexer para o processamento de informação, neste caso particular, consiste em tweets e perfis de interesse. Parte do processo de indexação da informação consiste na utilização de analyzers, que representa a política de extração do index a partir de diferentes termos.

Neste projeto trabalhamos com diferentes tipos de filtros, filtros estes que definem a política do analyzer.

Definição dos diferentes tipos de filtros: (Definição em inglês uma vez que certas palavras não têm tradução directa.)

- **Standard Filter:** Turns the input text into non-punctuated text.
- **NGramTokenFilter:** Tokenizes the given text into unbounded ngrams of the given size(s).
- **Stop Filter:** Which removes some of the words that are used in the english language as connectors but do not provide much information about the content itself.
- **Lower Case Filter:** Changes all the text into lowercase, so that words that are the same but change the capitalization would be indexed to the same token.
- **Snowball Filter:** Stems words according to the specified language. By doing that, we lose information about the original text, which can be useful because sometimes we would like words with the same root to be indexed to the same token.

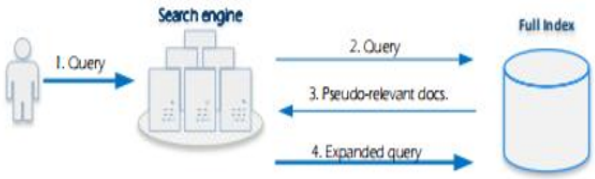
Conference'17, July 2017, Washington, DC, USA
2018. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

Concluída a construção do index de informação, passa a ser possível fazer pesquisas sobre essa mesma informação. Para que isto seja possível, todas as querys tem de ser processadas utilizando o mesmo analyzer que originalmente foi utilizado para a construção do index, e podem depois ser utilizadas para pesquisar a informação indexada, de acordo com a mesma política de similarity utilizada no index. Depois da pesquisa estar completa, o numero de resultados é devolvido. Estes resultados são assim escritos para um ficheiro (myResults.txt).

A última etapa é juntar todas estas métricas para cada um dos filtros do analyzer, que pode assim ser utilizado para construir representações visuais dos resultados, dando assim uma noção melhor de como diferentes tipos de filtros influenciam a precision e o recall. Por forma a confirmar as diferentes métricas utilizadas usamos o trec_eval, que utiliza os resultados corretos (qrels.txt) e os resultados obtidos de forma a poder pontuar os resultados e compará-los. Assim, estes resultados são apresentados posteriormente, na secção 4.3 deste relatório.

Suportado pelas conclusões retiradas da primeira fase do projeto, no segundo checkpoint implementámos um mecanismo PRF ("Pseudo Relevance Feedback") através da expansão de querys.

O mecanismo utilizado para fazer a expansão da query inicial consiste na utilização dos resultados mais relevantes da pesquisa inicial para que, criando uma combinação destes conjuntos de termos (do query inicial e dos resultados obtidos), seja possível melhorar o query introduzido pelo utilizador. Utilizando os documentos devolvidos pelo servidor após a pesquisa, são utilizados os termos mais frequentes ocorridos nos documentos melhor classificados (declarados pelo sistema como mais relevantes) e com esses mesmos termos é expandida a query inicial. Uma vez expandida a query com os termos mais frequentes esta é enviada para o motor de busca para novo processamento, de maneira a que agora possam ser obtidos resultados melhores.



.

Dentro do panorama de feedback por pseudo-relevância foi utilizado o algoritmo de Rocchio, algoritmo este que implementa o feedback de relevância num modelo de espaço vetorial. O algoritmo de Rocchio combina o vetor da query inicial com os vetores de feedback positivos e negativos, vetores estes criados utilizando os documentos relevantes e não relevantes, respetivamente, sendo que o feedback positivo é mais valioso que o feedback negativo, uma vez que o nosso objetivo é expandir a query de pesquisa. Por forma a que seja feita a distinção entre termos relevantes e termos não relevantes é calculada a distribuição de ocorrências de um termo entre os documentos de topo, com melhor ranking, e os documentos, mas ranking mais baixo.

Desta forma, atribuindo diferentes pesos aos diferentes vetores conseguimos construir uma nova query, com mais termos, com o objetivo de melhorar a precision dos documentos obtidos. Por forma a alterar a metodologia de atribuição de peso dos termos, relevantes e não relevantes, foram utilizados diferentes tipos de classes Similarity, nomeadamente BM25, existindo assim um impacto direto no ranking.

Após a conclusão da implementações propostas na primeira e segunda fase do projeto foi iniciado o desenvolvimento da fase final do projeto, para a qual os objetivos são:

(1) Modificação do sistema de indexação inicial para indexação incremental;
(2) Expansão dos querys (construídos através dos perfis de interesse);
(3) Remoção de resultados duplicados através da utilização dos algoritmos MinHash e Kmeans;
(4) Utilização de diversos modelos de ranking dos resultados e comparação dos resultados obtidos;

Em suma, temos como objetivo principal focar a implementação final na relevância e diversidade dos resultados.

3 IMPLEMENTAÇÃO

No seguimento do que foi descrito na secção anterior deste relatório, podemos afirmar que parte da solução para os problemas (1) e (2) anteriormente mencionados foi desenvolvida nos primeiros dois checkpoints deste projeto. Com base nesta implementação o processo de indexação foi modificado por forma a que seja criado um index para cada dia de atividade. Desta forma, é possível fazer as pesquisas apenas sobre um dia em particular, respeitando as restrições impostas.

Por forma a resolver o problema da diversidade de informação, recorremos aos algoritmos K-Means, de forma a tornar possível o agrupamento dos tweets como items semelhantes, gerando clusters dos mesmos, mas também ao algoritmo MinHashing.

3.1 Algoritmo Kmeans

No que toca ao algoritmo de K-Means este é um dos métodos mais comuns e eficazes para classificar os dados por causa de sua simplicidade e capacidade de lidar com volumosos conjuntos de dados. Primeiramente sentimos a necessidade de introduzir a noção de cluster. Um cluster é um conjunto de elementos semelhantes entre si, sendo que o processo de clustering tem como objetivo, com base num conjunto de dados, juntar em grupos os dados de acordo com as suas características, sendo que dentro de um grupo (cluster) o objetivo é maximizar as semelhanças entre elementos, sendo que se tenta maximizar as diferenças entre clusters, de forma a que cada grupo seja semelhante entre si mas diferentes uns dos outros. O algoritmo K-Means aceita o número de clusters e o conjunto inicial de centróides como parâmetros, este é o número total de grupos que serão formados durante a execução do processo de clustering. O objetivo principal do algoritmo é obter uma diferença quadrática mínima entre o centróide do cluster e o item no conjunto de dados. A principal desvantagem do agrupamento de K-means é a determinação dos centróides de cluster iniciais, uma vez que é determinado aleatoriamente. O algoritmos de K-Means utiliza a distancia de cada ponto ao centro de um cluster como motivo para adicionar um certo ponto a um cluster, pois parte do princípio que pontos próximos uns dos outros partilham diversas semelhanças entre si, sendo que pontos separados por longas distâncias no espaço possuem muitas diferenças. O processamento do algoritmo apresenta diversas fases:

- São gerados aleatoriamente K centróides distribuídos no espaço. A distância de cada item no conjunto de dados é calculada a cada um dos centróides do respetivo cluster. Um documento é então atribuído ao cluster com o qual a distância ao documento for menor. Após todos os pontos terem sido atribuídos, o centróide do cluster ao qual os items foram atribuídos são recalculados;
- Após recalcular o centróide para o ponto cujas coordenadas correspondem à média das coordenadas de todos os pontos do cluster. O algoritmos termina quando chegar a um ponto em que as coordenadas do centróide já não são atualizadas, o que significa que os clusters estão formados e todos os items estão atribuídos ao cluster cujo centroide mais se aproxima dele.

3.2 Método de MinHashing

O MinHashing é um método de Hashing sensível à localização, este mapeia documentos semelhantes ao mesmo código hash. Desta forma torna-se possível detetar casos de documentos duplicados, ou muito semelhantes. Para encontrar documentos semelhantes a um documento particular X, numa diretoria de 10000 documentos, é necessário fazer uma comparação de cada par par individualmente. Ainda assim, esta implementação é bastante custosa, não sendo escalável. Como alternativa, e por forma a reduzir o custo do algoritmo, a comparação é feita através de conjuntos, shingles selecionados aleatoriamente de dois documentos. Desta forma um problema não escalável passa a ser computável em tempo útil.

4 AVALIAÇÃO

No que toca à avaliação o método utilizado para obtenção de valores foi atravez do Normalized Discount Cumulative Gain (nDCG), aplicando a formula abaixo:

$$nDCG = \frac{1}{best_{DCG}} \sum_{i=1}^{n} \frac{2^{r(i)} - 1}{\log_2(i+1)}$$

. Onde bestDCG representa o máximo ganho possível, em que os tweets não relevantes têm um ganho de 0, os tweets relevantes têm um ganho de 0.5 e os tweets muito relevantes têm um ganho total de 1 valor.

4.1 Descrição dos dados

Assim como descrito previamente, a informação é responsável pelos alicerces sobre os quais um motor de pesquisa pode funcionar. Assim, nesta secção descrevemos o tipo de informação utilizada para este problema, assim como, a estrutura desta informação após o processo de indexação através do Lucene. Para a criação daquilo a que é chamado um daily email digest, lidamos essencialmente com dois "blocos"de dados, nomeadamente:

- Tweets [id, userId, text, userFollowers, isVerified, userTweets, hashtags, PostDate];
- Perfis de interesse dos utilizadores [topid, title, description, narrative];

Com isto, os campos indexados são utilizados para diferentes fases da pesquisa e ranking. Por forma a construir os querys iniciais são utilizados o título de cada perfil de interesse.

4.2 Baselines

4.2.1 LMD Retrieval Model.

Na abordagem a modelos de linguagem para recuperação de informações, documentos e consultas estão representados como modelos probabilísticos. Tipicamente, os documentos são classificados pela sua probabilidade de gerar a consulta. Os modelos de documentos são alvo de smoothing antes de medir a probabilidade de gerar a consulta (query).

$$p(t|M_d^{MAP}) = \frac{f_{t,d} + \mu \cdot M_c(t)}{|d| + \mu}$$

O smoothing é uma característica importante dos modelos de linguagem, pois equilibra as probabilidades do termo descontando a probabilidade dos termos que ocorrem no documento e ajusta probabilidades baixas ou nulas para cima, para outros termos que não ocorrem no documento. Isso contorna o problema da frequência zero, onde os termos da consulta que não ocorrem num documento, levariam a uma probabilidade de consulta geral de zero. Normalmente, abordagens de suavização combinam frequências de um termo num documento com a frequências do termo em toda a coleção. A biblioteca utilizada do lucene, LMDirichletSimilarity, necessita de um parâmetro Mu, como é possível verificar na formula abaixo:

Este parâmetro permite regular a similarity utilizada nos diferentes testes efetuados.

4.2.2 BM25 Retrieval Model.

*BM25 melhora sobre TF * IDF. BM25 significa "Best Match 25". Lançado em 1994, é a 25ª iteração de ajustes da computação de relevância. BM25 tem suas raízes na recuperação de informações probabilísticas. Uma pontuação de relevância (relevance score), de acordo com a recuperação probabilística da informação, deve refletir a probabilidade de um utilizador considerar o resultado relevante.*

O IDF do BM25 parece muito parecido com o Lucene IDF padrão (clássico). A única razão para a diferença aqui é a derivação da recuperação de informações probabilísticas. O Lucene faz uma mudança para o IDF regular da BM25. O IDF do BM25 tem potencial para dar pontuações negativas para termos com frequência de documentos muito alta. Assim, o IDF no BM25 do Lucene faz uma computação diferente para resolver esse problema. É adicionado 1 ao valor, antes de tomar o valor, o que torna impossível calcular um valor negativo.

O BM25 aceita dois parâmetros como argumentos, float k1 e float b, onde k1 controla a normalização da frequência de termos não-lineares e b controla em que medida o comprimento do documento normaliza os valores de tf. Por defeito os valores originais para estes parâmetros são k1 = 1.2 e b = 0.75, contudo, para assegurarmos que tínhamos os melhores resultados possíveis decidimos fazer diferentes testes, alterando os valores de k1 e b. Começamos os teste fixando o

valor de b em 0.75 e fizemos variar o valor de k1 desde 1.2 a 2, com um passo de 0.1, desta forma conseguimos analisar o comportamento dos dados à medida que o valor de k1 aumentava. Após iterar para todos os valores de k1, decidimos fixar o valor de b em 0.60, 0.65, 0.70, 0.80 e 0.85, sendo que para cada um destes valores foi feita a combinação com todos os valores de k1 de 1.2 a 2, com passo de 0.1. Assim, cobrindo toda esta gama de valores pudemos concluir que os melhores valores a utilizar como parâmetros para o BM25 foram de k1 = 2 e b = 0.8.

4.2.3 Pseudo-Relevance Feedback.

O feedback de relevância é uma característica de alguns sistemas de recuperação de informações. A ideia por detrás do feedback de relevância é levar os resultados inicialmente devolvidos de uma determinada consulta, reunir opções dos usuários e usar informações sobre se esses resultados são ou não relevantes para realizar uma nova consulta. O tipo de feedback de relevância utilizado no projeto foi o feedback por pseudo-relevância.

Tabela 1: Valores de nDCG antes e depois da aplicação de PRF

	nDCG1	nDCG0
Antes	0.2047	0.0199
Depois	0.2072	0.0227

O feedback de pseudo-relevância, fornece um método para análise na qual nao é necessario um utilizador. É automatizado a parte manual do feedback de relevância, para que o user obtenha melhor desempenho de recuperação sem uma interação intermédia. O método passa por fazer uma recuperação normal para encontrar um conjunto inicial de documentos mais relevantes, para então assumir que os primeiros K documentos classificados são relevantes e, construir uma nova query com informações destes documentos de forma a que seja possível consultar o sistema com esta nova query estendida e assim melhorar o resultado de resposta dos nosso sistema.

Na tabela de valores é possível ver que, com a utilização do método de feedback por pseudo-relevância o valor obtido através do nDCG apresenta um valor um pouco superior àquele que era o valor antes da implementação do PRF. Isto demonstra claramente que com a utilização deste tipo de estratégias passa a ser possivel a um motor de pesquisa melhorar os seus resultados apenas com simples passos internos, que melhoram significativamente o tipo de resposta que o utilizador recebe.

4.2.4 K-Means.

O algoritmo de clustering K-Means é um exemplo de cluster baseado em protótipos. No cluster baseado em protótipos cada exemplo é atribuído ao cluster representado pelo protótipo mais próximo, geralmente o resultado torna-se uma partição Voronoy do espaço de recursos porque a distância mais utilizada medida é a distância euclidiana. O algoritmo K-Means clustering consiste em dividir o conjunto de dados em k clusters, sendo o protótipo de cada cluster o vetor médio dos membros do cluster. Cada exemplo é atribuído ao cluster representado pelo protótipo mais próximo. Contudo, a maneira de como executar o algoritmo de K-Means já foi referida na secção 2 (Methods and Algorithms).

A forma como tentamos determinar o melhor numero possivel de cluster passou por iniciar um ciclo em em que k = 2 e percorrer o mesmo até 100. A linha de pensamento por detrás destes numero deve-se ao facto de que, formando apenas um super cluster com todos os dados seria o mesmo que não fazer absolutamente nada, deste modo, não faz sentido utilizar um algoritmo de cluster com menos do que 2 clusters diferentes. Depois, de forma a termos a certeza que tínhamos um grupo suficientemente grande, mas não grande demais para que os tweets não ficassem apenas agrupados individualmente ou simplesmente aos pares, decidimos limitar o numero máximo de cluster ao número 100.

Recorremos à ajuda de alguns indexes como o recall, precision, F1-Score, Rand Index, Adjusted Rand Index e valores de silhueta de forma a conseguirmos perceber qual o melhor numero de clusters para o problema. O F1-Score representa a média harmônica da precisão e do recall, onde uma pontuação F1 atinge seu melhor valor em 1 (precisão perfeita e recall) e pior em 0.

$$2 * \frac{Precision * Recall}{Precision + Recall}$$

O Rand Index calcula uma medida de similaridade entre dois clusters considerando todos os pares de amostras que são atribuídos nos mesmos ou diferentes clusters nas previsões de true clusterings. A pontuação do Rand Index é então ajustada para chance no ARIscore usando a seguinte fórmula (Onde N é o número total de exemplos) .

$$RandIndex : \frac{TruePositives + TrueNegatives}{N(N-1)/2}$$

$$ARI : \frac{RI - ExpectedRI}{max(RI) - ExpectedRI}$$

Tabela 2: Valores de nDCG para diferentes numeros de clusters

	nDCG1	nDCG0
14 Clusters	0.2037	0.0202
15 Clusters	0.2074	0.0251
16 Clusters	0.2034	0.0215

A tabela mostra os resultados de nDCG obtidos para diferentes valores de k, onde k representa o numero de clusters formados. Como é possível verificar, o algoritmo apresenta resultados melhores para k=15, o que significa que quando reunimos os nossos tweets em 15 conjuntos para depois estes serem processados conseguimos ter melhores resultados do que com 14 ou 16, assim sendo, do ponto de vista da otimização do nosso sistema o valor a escolher, face ao problema que estamos a abordar e sem duvida o valor de 15 clusters.

4.2.5 MinHash Algorithm.

O algoritmo de MinHash é uma técnica para estimar rapidamente se são dois conjuntos são semelhantes. O coeficiente de semelhança Jaccard é um indicador tipicamente utilizado na análise de similaridade entre dois conjuntos. Para os documentos (tweets) A e B, é definida como a proporção do número de elementos da sua interseção e o número de elementos da sua união. Este mede a semelhança entre os dois documentos. O valor está entre 0 e 1. Onde 0 mostra

que os documentos são diferentes e 1 mostram que esses documentos são idênticos entre si. O valor entre 0 e 1 mostra a probabilidade de semelhança entre os documentos. A formula de Jaccard é dada pela expressão abaixo:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

.

Tabela 3: Valores de nDCG antes e depois da aplicação de MinHash

	nDCG1	nDCG0
Antes	0.1998	0.0158
Depois	0.2007	0.0159

Para todos os documentos que sejam considerados como duplicados podemos então considerar que visto ser informação repetida não vai interessar ao utilizador ver a mesma no seu daily email de novidades, sendo assim, procedemos à remoção dos tweets considerados repetidos.

De acordo com os valores em tabela é visível ver que a aplicação do algoritmo de MinHash com a formula de Jaccard trouxe alguma melhoria ao programa, pois, através da leitura do valor de nDCG conseguimos perceber que ocorreu uma ligeira melhoria, isto deve-se ao facto de que, removendo os tweets repetidos, conseguimos aumentar o ganho geral do programa.

4.3 Discussão dos resultados

Primeiramente, e relativamente aos resultados obtidos na primeira fase do projeto:

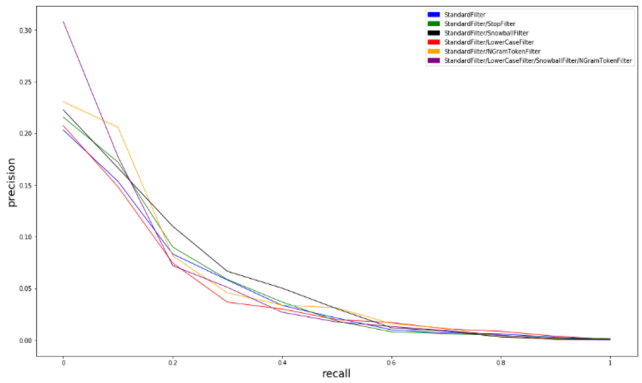


Figura 1: Precision and recall graph.

Filters	MAP	P@10
StandardFilter	0.0421	0.0690
StopFilter	0.0467	0.0793
SnowballFilter	0.0530	0.0967
LowercaseFilter	0.0413	0.0733
NGramTokenFilter	0.0475	0.0828
FinalAnalyser*	0.0516	0.0931

Figura 2: MAP and P@10 values table.

.

Todos os testes acima foram executados contendo o StandardFilter.

**O FinalAnalyser é resultante de uma combinação de diferentes filtros, nomeadamente: LowercaseFilter/SnowballFilter/NGramTokenFilte.*

De notar que MAP é a abreviação de Mean Average Precision e a segunda coluna representa a precisão após dez documentos devolvidos. (P@10).

O gráfico acima mostra como os valores de precisão e recall estão relacionados em diferentes combinações de filtros utilizados para construir o analisador. Olhando para os resultados, podemos ver que há uma relação de proporção inversa entre o recall e a precisão, embora isso se aplique somente a esse caso particular. Podemos ver claramente que, à medida que o valor da reviração aumenta, a tendência dos valores de precisão é zero.

Um sistema ideal com alta precisão e alto recall retorna muitos resultados, sendo que uma alta percentagem deles é relevante. Em primeiro lugar, ao analisar os resultados do StandardFilter e da combinação de StandardFilter + LowerCaseFilter, seus valores MAP (que nos dão uma noção de como a precisão é mantida em todo o espectro de valores de recall) são bastante semelhantes. Esse pode ser um indicador de quão pouco eles variam um do outro (a única diferença é que cada letra está em minúscula).]Um sistema com recall elevado, mas baixa precisão retorna muitos resultados, mas a maioria dos seus resultados previstos são incorretos quando comparados aos resultdos de obtidos. Um sistema com alta precisão, mas baixo recall é exatamente o oposto, retornando poucos resultados, mas a maioria está prevista.

Um sistema ideal com alta precisão e alto recall retorna muitos resultados, sendo que uma alta percentagem deles é relevante. Em primeiro lugar, ao analisar os resultados do StandardFilter e da combinação de StandardFilter + LowerCaseFilter, seus valores MAP (que nos dão uma noção de como a precisão é mantida em todo o espectro de valores de recall) são bastante semelhantes. Esse pode ser um indicador de quão pouco eles variam um do outro (a única diferença é que cada letra está em minúscula).

As médias do MAP são comparativamente semelhantes, mas são baixas se considerarmos a média dos outros filtros.

Olhando para as métricas StopFilter, podemos afirmar que há um aumento notável no seu valor MAP, que é o resultado do fato de que,

com este filtro, estamos removendo palavras que não são particularmente importantes para a precisão da busca, ou seja, interromper palavras.

O analisador Snowball é semelhante ao analisador Padrão, exceto no fato de que ele converte palavras em palavras de caule. Ignora colons, #,%, $, parênteses e barras. Inscreve os apóstrofes se eles estão no meio de uma palavra, mas os remove se eles estão no início ou no fim de uma palavra. A aplicação deste filtro mostra um grande aumento no valor MAP e, de fato, mostra o valor MAP mais alto, que é um reflexo do tipo de conteúdo que estamos usando. Este tipo de conteúdo usa muita pontuação, portanto, ao removê-lo no processo de indexação e pesquisa, podemos aumentar a precisão dos resultados. NGram Filter Tokenizers divide palavras em seqüências de token. Cada token tem um comprimento de grama mínimo especificado (min_gram) e o comprimento máximo do grama (max_gram). O NGramTokenFilter é aquele que obtém resultados mais próximos do ótimo, o que significa alta precisão e alta precisão. Este filtro atinge alta precisão sempre que há poucos documentos retornados, mas essa precisão cai drasticamente quando o recall aumenta, o que afeta o valor do MAP..

Na tentativa de maximizar o valor do MAP, tentamos combinar diversos tipos de filtros, nomeadamente o Filtro Padrão, o LowerCaseFilter, o SnowballFilter e o NGramTokenFilter. Embora essa combinação obtenha o valor de precisão mais alta, cerca de 33% quando o recall está fechado é de 0%, diminui drasticamente, indo até os valores de precisão mais baixos em vários pontos diferentes do gráfico.

Relativamente à segunda fase do trabalho, foram efetuados vários testes com diferentes tipos de similatiry, resultados estes expostos nas figuras 3, 4 e 5.

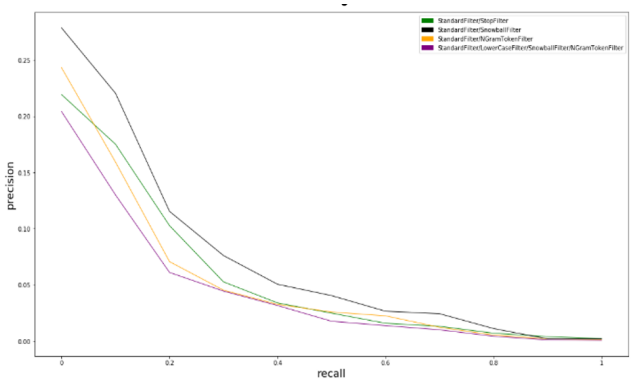


Figura 3: Classic Similarity, Precision and recall graph

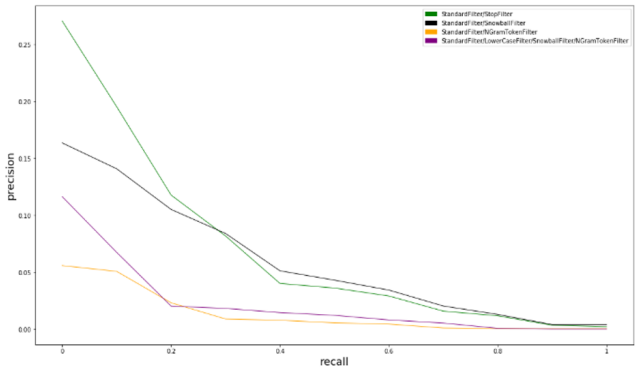


Figura 4: BM25 Similarity, Precision and recall graph

Filters	MAP	P@10	Map Exp	P@10 Exp
SnowballFilter	0.0530	0.0967	0.0670	0.0903
StopFilter	0.0467	0.0793	0.0487	0.0839
NGramTokenFilter	0.0475	0.0828	0.0468	0.0581
FinalAnalyser*	0.0516	0.0931	0.0384	0.0677

Figura 5: MAP and P@10 values table

.

Todos os testes acima foram executados contendo o StandardFilter.

**O FinalAnalyser é resultante de uma combinação de diferentes filtros, nomeadamente: LowercaseFilter/SnowballFilter/NGramTokenFilte.*

De notar que MAP é a abreviação de Mean Average Precision e a segunda coluna representa a precisão após dez documentos devolvidos. (P@10).

Na tabela, representada no ponto 3.2, estão registados os valores do MAP e do P@10 obtidos, em testes relativos a diferentes analysers, antes de termos implementado o algoritmo de expansão da Query, bem como os mesmos valores depois ter sido implementado o algoritmo.

Como é possível verificar nessa tabela, a implementação do algoritmo que expande a query, melhorou o valor MAP da pesquisa para alguns analysers, nomeadamente Snowball Filter e o Stop Filter. Para os outros analysers, este valor baixou ligeiramente. Em relação ao P@10, melhorou no Stop Filter, sendo que nos restantes analysers, este valor baixou.

Após implementar a expansão de queries, testámos o nosso programa com diferentes similatys para ver de que forma estas influenciavem os resultados e de tipo de similarity nos iria fornecer os melhores resultados. Foram feitos diversos testes no que toca à utilização do BM25 para determinador os melhores valores, sendo que no final conseguimos concluir que o valores que melhor se adequam ao problema são K= 2 e B=0.8.

É possível verificar no gráfico que houve uma ligeira melhoria nos valores obtidos, contudo esta melhoria não é transversal a todos os tipos de analysers, uma vez que no caso do analyser composto pelos filtros: StandardFilter, LowercaseFilter, SnowballFilter, NGramTokenFilter os valores inicias de precisão não ascendem a valores tão grandes como na versão sem expansão de queries.

5 MELHORAMENTOS POSSÍVEIS

No que toca a possíveis melhoramentos temos como sugestão a fusão de conhecimentos aprendidos noutras unidades curriculares, como é caso da cadeira de aprendizagem automática, uma vez que, utilizando os métodos aprendidos na mesma passa a ser possível construir um algoritmo que, dado um conjunto de dados consegue classificar em relevante ou não relevante cada um dos documentos inseridos, com base nos argumentos dados. É possível também, agora que os documentos estão classificados, aplicar técnicas de clustering sobre os dados de forma a ser possível analisar de que maneira os dados se relacionam. Desta maneira tornar-se-ia possível, com apenas um modelo de machine learning treinado, carregar, classificar, dividir e analizar os dados.

Um tipo de melhoramento que pode ser tido em conta é a alteração do tipo de clustering efetuado, seria interessante aplicar outro tipo de tecnicas de clustering como por exemplo uma mistura gaussiana ou um modelo de DBSCAN, que tem por base a densidade dos pontos para os classificar, no contexto do problema é certo que temos de ter em conta todos os tweets para que dessa forma se consiga avaliar quais os melhores dentro de cada area para que assim os passamos recomendar, contudo, seria interessante analisar de existem alguns pontos, neste caso tweets, que seria classificados como noise, e tentar perceber o que levou os classificadores a atribuir este tipo de label aos mesmos.

Um tipo de experiência que podia ser realizada era a utilização de mais tipos de similarity, nos enquanto desenvolvíamos o projeto tivemos a oportunidade de utilizara similarity clássica do lucene, a BM25 e LM Dirichlet Similarity, contudo, um possível melhoramente seria o teste e analise dos resultados utilizando outro tipo de similarity de forma a tentar com que os resultados melhorassem ainda mais.