Experimentação para o artigo Algorithms for Active Classifier Selection: Maximizing Recall with Precision Constraints[1]

Lucca Martins Felixr[116058800]

Universidade Federal do Rio de Janeiro
lucca_martins30@yahoo.com.br

Sumário. A proposta do trabalho é a de avaliar a eficiência do algoritmo proposto pelo artigo[1], que promete realizar a escolha de um classificador cuja revocação seja a maior, impondo restrições na precisão e limitando o número de itens a serem classificados pelo trabalho humano.

Palavras-chave: maximizar revocação, restrições na precisão, campanha de avaliação

1 Introdução

Na área de recuperação da informação avaliar os resultados de um modelo proposto é uma tarefa tão importante quanto a de desenvolver o próprio, haja visto a relação de dependência entre essas duas atividades. O problema, no entanto, reside no que espera-se como resultado de um modelo de recuperação da informação. Como já foi mostrado que usuários são mais sensíveis à falhas de precisão do que à falhas na revocação, muitas campanhas para seleção de modelos valorizam mais a precisão do modelo. No entanto, ainda que a experiência com o usuário aponte essa alta sensibilidade, um outro estudo[2] demonstra que a insatisfação (seguida da troca de ferramenta) entre os usuários com as search engines é 3.25x maior em relação à falhas na cobertura dessa ferramenta (o resultado não contém uma quantidade boa de elementos esperados) se comparada com a precisão em si.

Por conta desses dados, esse estudo propõe um campanha de seleção de modelo que, a partir de restrições no valor da precisão, execute um algoritmo de seleção que escolha o modelo que maximize o valor da revocação.

2 Definição do Problema

2.1 Peculiaridade do Cenário

Um fato interessante que dista as campanhas de avaliação propostas em aula com a modelada pelo artigo é sobre como a disponibilidade dos dados está condicionada. Ou seja, enquanto que nas campanhas em aula o resultado das consultas era acessível a todo instante pelo algoritmo de avaliação, no cenário do artigo assume-se que os

dados ainda não possuem um classificação (de relevante ou não-relevante), tendo apenas como recurso um oráculo humano capaz de classificar para qualquer que seja o item. Levando em consideração que muitas das campanhas podem dispor de um dado com poucas ocorrências de exemplos positivos, fazer uma escolha de classificador e garantir alto grau de confiança tornam essa tarefa custosa na quantidade de consultas ao oráculo. Dessa forma, um dos desafios embutido por conta do cenário é o de minimizar o número de consultas a esse oráculo humano.

Para o trabalho da disciplina assume-se que o oráculo humano é a própria classificação do dado (que já está disponível) e um dos valores a ser exibido na seção de resultados é a relação entre a quantidade de consultas que será realizada a esse oráculo e a qualidade do resultado, assim podendo confirmar se, de fato, o algoritmo precisa consultar poucas vezes o recurso humano. Visto que no artigo[1] demonstra-se uma redução significativa nesses custos.

2.2 **Modelando o Cenário**

A primeira grande ideia dos autores para resolver esse problema foi a de reformular o desafio de escolher entre todos modelos para um conjunto menor, no qual engloba apenas os melhores classificadores. O conjunto dos melhores classificadores contém apenas os classificadores que atendem as condições mínimas de precisão. Dessa forma, o escopo do problema foi reduzido ao desafio de maximizar o alcance apenas entre esses classificadores.

Observe que maximizar o alcance é equivalente a maximizar a revocação mas não requer estimar o denominador dela (o conjunto de todos os itens positivos, que é constante). Sendo assim, um problema que antes era o de estimar todos os itens positivos de um conjunto resultado agora pode ser quebrado em estimar apenas o número de itens positivos no conjunto predito como positivo pelo modelo (ou seja, os verdadeiros positivos).

2.3 **Extrapolando a Área de Recuperação de Informação**

Um adendo simples, porém interessante, capaz de conectar os ramos da recuperação da informação e aprendizado de máquina é o fato de que um modelo em R.I. nada mais é do que um classificador binário em A.M.. Dessa forma, aos que desejam escolher um modelo de A.M. onde seja desejável alto índice de revocação é possível, sem nenhuma mudança, aplicar a mesma campanha de avaliação visto que o material provindo dos classificadores ou modelos é apenas seu resultado produzido, como será definido formalmente a seguir.

2.4 **Definição Formal do Cenário**

Em um ponto de vista mais formal, assume-se um cenário onde um “usuário” deseja um sistema que possa selecionar o melhor classificador a partir de um conjunto

de **m** candidatos $h_1, ..., h_m$**.** Além disso, espera-se que o usuário possua conhecimento do domínio e saiba exatamente qual a tolerância de precisão ideal **PT** ele deseja (visto que ele quem irá escolher esse valor por meio da passagem de parâmetros).

Um candidato será classificado como **bom** se tiver pelo menos precisão maior que **PT**. Outrossim, o usuário também tem a opção de escolher um valor de folga **γ**, onde **PT**≥**γ**≥0, tal que permita dizer que um classificador é **aceitável** se sua precisão for maior que **PT-γ**. Com isso, o desafio está em procurar o classificador que maximize a revocação dentro do conjunto de classificadores aceitáveis.

Como havia sido mecionado na sessão *2.1* o corpus possui a peculiaridade de um alto número de testes sem classificação mas é possível consultar um oráculo capaz de classificar qualquer que seja o item de teste. Para o cenário, serão feitas no máximo até **T** consultas. Se soubéssemos os itens positivos do corpus, seríamos capazes de particionar em (1) n_{1i} os verdadeiros positivos, (2) n_{2i} os falsos negativos, (3) n_{3i} os falsos positivos e (4) n_{4i} os verdadeiros negativos. A revocação de h_i é definida como $\frac{n_{1i}}{n_{2i}+n_{1i}}$, a precisão de h_i é definida como $\frac{n_{1i}}{n_{3i}+n_{1i}}$ e o alcance é definido como n_{1i}.

A dificuldade prática em estimar o valor de revocação reside em estimar o denominador $n_{2i} + n_{1i}$ mas, como havia sido mencionado anteriormente, esse denominador é constante em todos os classificadores. Logo, a estimativa do valor de alcance de cada candidato nos garante um mesmo ranking entre os classificadores e a busca por itens positivos é reduzida apenas ao grupo de itens preditos como positivo pelos classificadores.

Por conveniência também será definido um classificador trivialmente aceitável **α** que possui precisão 1 e alcance de 0. Caso o algoritmo não encontre nenhum classificador que satisfaça as condições exigidas pelo usuário, então o classificador retornado será alpha. Além disso, definiremos como ótimo o classificador que maximiza o alcance em relação a todos os bons classificadores e denotaremos como **R*** o alcance desse classificador. Analogamente a precisão também definiremos uma variável Ɛ de folga para o alcance, permitindo assim que o classificador escolhido tenha um leve decréscimo de alcance em relação ao ótimo, tal que o alcance aceitável seja no mínimo (1 - Ɛ)R*, Ɛ é tal que 1≥Ɛ>0.

Para terminar, uma viável ***δ*** que garantirá uma confiança de 1-***δ*** para a escolha final do algoritmo. Em outras palavras, o algoritmo garante com uma probabilidade de 1-***δ*** que o classificador escolhido tenha precisão aceitável e tenha um alcance de (1 - Ɛ)R*. Mais formalmente, denotando com p_i e r_i a precisão e alcance do classificador i, então o algoritmo garante que as 2 condições a seguir ocorram com uma chance de (1 - Ɛ)R* para o classificador c retornado pelo algoritmo.

$$p_c \geq PT - \gamma \quad (1)$$

$$R_c \geq (1-\epsilon) \max_{c' \in M \cup \{\alpha\} \text{ s.t. } p_{c'} \geq PT} R_{c'} \quad (2)$$

3 Implementação do Algoritmo

3.1 Material de Entrada

Um ponto interessante no material de entrada é o de que não é necessário possuir a implementação dos classificadores candidatos, no lugar deles somente é necessário seu conjunto resultado. Este conjunto resultado nada mais é que um vetor contendo um identificador para cada item classificado como positivo. Dessa maneira, o algoritmo requer uma matriz de preditos positivos PP onde PP_i é o array de preditos como positivo pelo classificador i. Note que não é necessário informar o peso de cada item no conjunto resultado, assim como a sua ordem de relevância (mais para frente veremos como é possível contonar esse detalhe e trabalhar com essa relevância).

Outro detalhe a ser destacado é o de que esse algoritmo provê uma campanha de avaliação capaz de avaliar resultados de apenas uma consulta. Por um bom tempo na leitura acreditei que esse detalhe, não muito bem explicado, seria bem definido e uma solução para o problema seria oferecida. No entanto, não consegui desenvolver nenhuma maneira de contornar o problema pois, diferente do que foi possível fazer em *3.3*, os itens precisam ser de uma mesma consulta.

Ademais, o usuário também informará $\boldsymbol{\delta}$, $\boldsymbol{\gamma}$, ε , T e uma forma de comunicação com o oráculo. Note que caso o usuário não queira trabalhar com a possibilidade de ser selecionado um classificador com precisão aceitável, então basta que ele defina $\boldsymbol{\gamma}$=0. Além disso, como já possuímos todo o dado classificado o oráculo que utilizaremos nada mais é que uma consulta ao dado já classificado.

3.2 Variabilidade do Algoritmo

Por se tratar de um algoritmo extenso, sua formulação permite que alguns pedaços possam ser substituídos. Mais especificamente, a função *SampleAndUpdateStatistics* é deixada livre pra qualquer implementação que realize o desejado. No artigo[1] os autores sugerem 2 algoritmos possíveis para essa função: *round-robin* e *pooled-sampling-shared-label*, sendo apenas o segundo implementado de fato pois este apresentou os melhores resultados segundo os autores. Para a proposta do trabalho o escopo também foi limitado apenas a experimentação dessa versão.

A função em questão é responsável por escolher um item dentre os que ainda não foram classificados e então realizar uma consulta ao oráculo para descobrir se este é positivo ou não. A partir daí é estimada uma precisão pro alcance desse classificador, com esse valor é possível atualizar o intervalo de onde reside a real precisão. Este intervalo é o único dado que será de fato utilizado no algoritmo principal.

Enquanto que o *round-robin* possui uma abordagem mais ingênua, onde escolhe um dos classificadores e seleciona aleatoriamente um item ainda não classificado, o *pooled-sampling-shared-label* tira proveito do fato de que muitos itens serão preditos como positivos por mais de um classificador podendo atualizar mais do que apenas um classficador por consulta ao oráculo. Dessa forma, a escolha do item a ser classificado pelo oráculo é baseado na quantidade de classificadores que possuem

esse item em seu conjunto PP_i, além da quantidade de itens já classificados por cada classificador (para garantir distribuição uniforme).

3.3 **Trabalhando com o Grau de Relevância de um Classificador**

Enquanto realizava os testes me incomodei com o fato de que o grau de relevância dos resultados era simplesmente ignorado por essa campanha de avaliação. Por isso propus um meio de contornar essa limitação sem alterar em nada a implementação do algoritmo. Basicamente, o que queremos é saber qual a tolerância para o grau de relevância que, dada as restrições na precisão, maximize o valor de revocação.

Com o problema agora definido, a ideia baseia-se em transformar o que antes era apenas um classificador em vários. Como havia sido definido em *3.1*, apenas precisamos de um array que contenha quais foram os itens preditos como positvo pelo classificador. Já que temos um vetor ordenado pela relevância onde cada elemento possui o identificador do item t_i e seu respectivo valor de relevância rel_i, podemos definir um corte nesse vetor da seguinte forma: dado um valor de tolerância tol para a relevância dos itens, parta da última posição onde há o item mais relevante e itere até a posição k tal que $rel_k < tol$, o corte será do intervalo [k:-1] que pode ser interpretado como todos os itens que possuem relevância maior que tol.

Agora que temos um método para transformar o vetor com itens e grau de relevância em material de entrada do algoritmo, podemos definir um algoritmo que, a partir do vetor, de um intervalo de tolerância $[tol_{min}, tol_{max}]$ e uma quantidade de passos p faça p diferentes cortes com valores de tolerância dentro do intervalo e utilize esses cortes como p candidatos na campanha de avaliação.

Extrapolando ainda mais a ideia, também é possível combinar vários cortes para mais de um classificador em uma mesma campanha, dessa forma teremos como o resultado o melhor classificador com a melhor configuração para o seu valor de tolerância.

3.4 **Dificuldades com a Implementação**

Por mais que o algoritmo trate de tópicos já discutido em sala, a sessão *2* deixa bem claro que o artigo[1] possui uma forte base estatística e matemática por trás por conta das peculiaridades discutidas na sessão *2.1*. Sendo assim, vale ressaltar que, por mais simples que seja a explicação e ideia do algoritmo, torná-lo código executável já não foi uma tarefa simples. A implementação foi realizada em notebook jupyter com o Kernel de Python 3. A escolha da linguagem foi favorável pois me parece que muito da descrição do algoritmo parecia estar escrita de forma a facilitar quem usa essa linguagem.

No entanto, como durante toda a execução do algoritmo não é definido um valor para a estimativa de alcance dos classificadores, sofri algumas dificuldades em definir o que seria a configuração do melhor classificador. Definir essa configuração é uma

tarefa necessária visto que em um dos trechos (figura a seguir) o autor pede pra escolher o melhor classificador.

> **if** $|RQ| > 1$ **then** return best classifier in RQ;
> **elseif** $|KA| > 0$ **then** return best classifier in KA;
> **else** return α signifying the trivial acceptable classifier
> $(PP_0 = \emptyset)$.;

Teoricamente é fácil saber que o melhor classificador é aquele que, ao mesmo tempo que tem precisão aceitável, também possui a maior revocação comparado aos outros aceitáveis. Como não há esse valor definido no algoritmo, tomei como escolha que a estimativa de alcance seria o teto de alcance desse classificador.

4 Experimentação

4.1 Coleta de Dados

A escolha desse artigo para o projeto teve uma forte influência pela base de dados que eu poderia usar neste trabalho. Como no trabalho 1 nossos resultados haviam sido analisados em uma campanha de avaliação, aproveitei a oportunidade para então realizar uma campanha semelhante, mas reduzindo o número de candidatos por não conseguir baixar todos os dados e alguns não serem parseáveis ao meu script. Além dos dados[3] que utilizei dos colegas de sala, também baixei parseei o conjunto dos relevantes[4].

Como havia sido mencionado na seção *3.3*, transformarei um mesmo candidato em vários nessa campanha por meio de cortes, variando apenas a tolerância para o valor de relevância retornado. A partir disso será possível avaliar não só o melhor candidato, mas também seu melhor corte.

Pelo fato de termos algumas consultas com um número relativamente baixo de casos positivos, isso influenciou bastante os resultados e trouxeram alguns dados inconsistentes (muitos vezes o algoritmo retornou alpha, mesmo que alguns atendessem as exigências de precisão) e por conta disso decidi remover as consultas de número 9,8,6,5,3.

4.2 Problemas com a Parametrização

Além de óbvios problemas de implementação que ocorrem em todo algoritmo nas suas primeiras execuções, sofri também de problemas em escolher os valores de parâmetros (definidos na seção *2.4*). O material utilizado pelo artigo original retornava resultados com mais de 12800 itens preditos como positivos, por conta dessa disparidade de ordem de grandeza entre as bases de dados, tive que realizar o estudo de precisão e revocação sem o uso do algoritmo dos autores para então que, com esse conhecimento no corpus e modelos, pudesse definir parâmetros viáveis.

4.3 Resultados Esperados

Ainda que na seção *3.3* eu tenha proposto uma solução para trabalhar com o grau de relevância que os modelos retornam, sabe-se que um candidato x com subconjunto de itens positivos de um candidato y terá valor de revocação no máximo igual ao de y mas a tendência é diminuir. Como o algoritmo não utiliza a estimativa de precisão para nada além de definir se um modelo é ou não aceitável, então já pode-se esperar que a solução que propus não maximize o valor de precisão. No entanto, imagine um candidato k que não atenda as condições de precisão, por conta de que tenha retornado muitos falsos positivos. Ainda assim, esse candidato k possui um alto número de verdadeiros positivos entre os itens mais relevantes, o que lhe garante alta revocação e alta precisão somente com esses itens. Por conta disso, fazer os cortes da seção *3.3* permitem descobrir alguns modelos que estavam na mesma situação que o esse candidato hipotético.

Outro resultado esperado é o da economia no quesito de número de consultas ao oráculo, como pode-se ver na imagem a seguir, muitos são os resultados que colidem em mais de um trabalho logo muitos serão os dados reaproveitados visto que o algoritmo *pooled-sampling-shared-label* utiliza as colisões como critério de escolha.

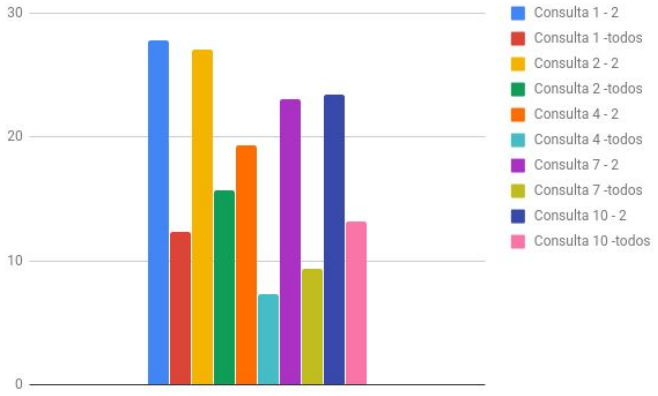

Fig1: Gráficos com % de colisões de resultados em cada consulta

4.4 Resultados Obtidos

Conforme foi mencionado, houveram 5 campanhas de avaliação 1, 2, 4, 7, 10 em todas estas campanhas eu realizei cortes com até 10 valores de tolerância diferentes, mas isso alternou para cada situação. Os registros que acompanham esse arquivo mostram os resultados para cada uma das 5 campanhas realizadas. O pode-se descrever aqui é que os resultados escolheram em 3 das 5 campanhas o algoritmo ideal. Para isso foi necessário deixar o parâmetro $\boldsymbol{\delta}$ em 0.5, ou seja, 50% de confiança que de fato ocorreram. Quando o valor de confiança era alto demais o algoritmo retornava apenas $\boldsymbol{\alpha}$.

5 Conclusões

Por se tratar de um time de research da Microsoft que possui uma grande ferramenta de recuperação de informação na web, o Bing, minhas expectativas eram altas sobre como eles elaborariam a campanha de avaliação. O que me supreendeu foi que a proposta, mesmo sendo muito simples, envolveu teoremas complexos e com literaturas extensas para que as exigências do cenário em questão fossem satisfeitas. Pela forma como os autores descrevem o problema, creio que esse cenário seja o real cenário em que as campanhas de avaliação do Bing ocorrem, visto que eles não dão destaque as peculiaridades nele existentes, apenas as descrevem como uma parte comum que eles já estão acostumados a lidar.

Para a nossa campanha de avaliação os resultados não impressionaram porque estamos comparando um algoritmo baseado em estimativas com uma campanha que realiza mensurações exatas. Ainda assim, foi impressionante a redução em ordem de grandeza ao número de itens que de fato precisam ser classificados por um humano para que a avaliação seja próxima da realizada no trabalho 1.

Referências

1. Artigo usado como tema do estudo - https://www.microsoft.com/en-us/research/wp-content/uploads/2017/01/wsdm-2017-bennett-et-al.pdf
2. Q. Guo, R. W. White, Y. Zhang, B. Anderson, and S. Dumais. Why searchers switch: Understanding and predicting engine switching rationales. In Proceedings of the 34th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2011), pages 334–344, 2011.
3. Base de Dados retirada do grupo no Facebook - https://www.facebook.com/groups/360725367745792/
4. Multi-armed bandit problem - https://www.cs.mcgill.ca/~vkules/bandits.pdf