Sistema de recomendação de edição de artigos do Wikipédia

Rennan de Lucena Gaio[1][114146746]

[1] Universidade Federal do Rio de Janeiro
rennan.gaio12@gmail.com

Abstract. Criar um sistema de recomendação de criação de conteúdo para o wikipédia especificamente para o português. Ranqueando os arquivos que seriam mais relevantes para a nossa língua, mas que ainda não existe tradução para nosso idioma. Como não teremos como testar diferentes editores, levaremos em consideração apenas os artigos que deveriam ter edição em português.

Keywords: Wikipédia, sistema de recomendação, machine learning

1 Introdução

Tendo em vista a grande quantidade de dados que são gerados hoje em dia, teve-se a necessidade de listá-los e armazená-los em grandes bibliotecas virtuais. Uma dessas grandes plataformas de arquivamento de informação é a famosa wikipédia, que possui mais de 4653746 páginas (20/06/2018) armazenando o conhecimento sobre diversos assuntos. Porém existe um grande problema intrínseco devido a globalização dos dados: existe uma grande barreira linguística entre pessoas de diferentes países, e diversas informações neste site não possuem tradução para muitas línguas, criando um desfalque de informação para diversas pessoas.

O artigo escolhido como base foi o "Growing Wikipedia Across Languages via Recommendation", que propõe um sistema de recomendação para a edição de páginas não existentes dado que existe uma página já criada com informações em outro idioma. Ele acredita que recomendar páginas para os editores cria uma maior motivação para o editor realmente criar aquele conteúdo, e acredita que a informação registrada será de maior qualidade, pois o tópico seria de maior interesse para o editor. Esses sistema de recomendação também leva em conta a relevância que um certo assunto tem em relação a aquele idioma, por exemplo, incentiva a criação de artigos que tenham relação com a cultura dos países que falam aquele idioma.

2 Proposta de trabalho

Criar um sistema de recomendação de criação de conteúdo para o wikipédia especificamente para o português. Ranqueando os arquivos que seriam mais

relevantes para a nossa língua, mas que ainda não existe tradução para nosso idioma. Como não teremos como testar diferentes editores, levaremos em consideração apenas os artigos que deveriam ter edição em português baseado em sua relevância de forma geral.

3 Modelagem do problema

Assim como no artigo, o sistema será separado em módulos. O primeiro cuidará de mapear os arquivos que existem em uma língua, porém não existem na nosso idioma alvo, o português. Para tal, o autor utiliza a plataforma do wikidata, conseguindo extrair esse tipo de informação entre wikis de diferentes idiomas.

O segundo cuidará de ranquear esses documentos achados no primeiro módulo, de forma a recomendar os melhores assuntos possíveis para se criar conteúdo em nosso idioma. Ele se utiliza de modelos de aprendizado de máquina para a classificação, e utiliza estatísticas coletadas no wikidata, wikimídia e outros.

No artigo ele cria um terceiro módulo, que recomenda artigos baseados no gosto de cada editor que ainda não tinham sido escolhidos, mas nesse trabalho esse módulo não será abordado.

4 Wikidata

4.1 Sobre o wikidata

"O Wikidata é um banco de dados secundário, livre, colaborativo e multilíngue que coleta dados estruturados que servem de suporte à Wikipédia, ao Wikimedia Commons, aos outros projetos do movimento Wikimedia e a qualquer um no mundo." - www.Wikidata.org.

Nessa plataforma é possível ter acesso a vários metadados sobre diversas páginas sobre o wikipédia, e que pode ser editado de forma colaborativa para melhorar ainda mais a eficiência de suas informações, que podem ser utilizadas para fazer diversas pesquisas. Esse grande repositório consiste em armazenamento de diversos itens rotulados, em que podem ser associados pelos mesmos, criando links entre diversos elementos registrados.

4.2 Estrutura de arquivos no site

Os rótulos podem ser divididos principalmente entre os do tipo Qxxx, que representam itens/entidades com material disponível em alguma wiki de algum país. E os do tipo Pxxx, que representam uma propriedade em que um objeto tem, ou um link que 2 objetos podem ter (formato de triplas), como por exemplo:

Table 1. Formato de triplas do wikidata.

Item	Property	Value
Q42	P69	Q691283
Douglas Adams	educated at	St John's College

Com isso podemos mapear os diversos conteúdos encontrados nesse grande repositório.

4.3 Mapeamento de arquivos

Essa foi uma parte muito problemática para o início do projeto. No artigo de referência eles utilizam os dados do wikidata, porém sua documentação não é clara, nem fácil de achar, além de grandes limitações em sua API. Grande parte dos materiais procurados também eram muito incompletos, que gerou um grande problema para conseguir extrair todos os arquivos dessa base de dados. Como não era possível extrair/consultar todos os possíveis conteúdos de todos os IDs ao mesmo tempo, eu usei um grupo de propriedades para separar uma fatia desses IDs para trabalhar em cima. Essa estratégia de analisar apenas uma parte dos dados também foi utilizada pois a quantidade de dados era muito grande, e seria necessário muito mais tempo/processamento do que o possível.

Para isso foi utilizado os métodos do SPARQL, que estão disponíveis para a consulta no site: https://query.wikidata.org. Um exemplo de consulta para se obter todos os IDs que possuem como propriedade “Unified Astronomy Thesaurus ID” ou P4466 no SPARQL:

```
SELECT ?item ?itemLabel ?id WHERE {
        ?item wdt:P4466 ?id
        SERVICE wikibase:label {
                bd:serviceParam wikibase:language "en". }
}
```

Utilizando então o dump que podemos fazer deste site, foram feitas consultas com 2 propriedades: o P4466 (“Unified Astronomy Thesaurus ID”) e o P118 (“Divisão Esportiva”), extraindo assim mais de 35804 IDs do tipo Qxxx, que referenciam a páginas sobre um determinado assunto em algum idioma.

4.4 Encontrando arquivos relevantes

Depois de Mapearmos nossas opções de busca, foi criado um bot que retirava informações do Wikidata a partir de sua API. Esses dados poderiam ser retirados em formato JSON utilizando o método GET passando em sua URL dados relevantes para a sua busca, por exemplo:

URL:
https://www.wikidata.org/w/api.php?action=wbgetentities&ids=Q76&sitefilter=enwiki&format=json

Nesse request estamos utilizando o método “wbgetentities” que retorna informações sobre uma entidade, passando o id dessa entidade. Como parâmetro adicional estamos filtrando os resultados apenas para aqueles que estão na wikipédia em inglês, e formatando a resposta no formato JSON. Neste exemplo seria um exemplo de consulta para as páginas Wikipédia com o tópico “Barack Obama”. Mais informações sobre como se usa esse método podem ser encontradas através desse link: https://www.wikidata.org/w/api.php?action=help&modules=wbgetentities

Dado essa estrutura de utilização da API, coletamos todas as páginas e dividimos elas em 2 grupos: As que tinham páginas criadas com o conteúdo em português e as que ainda não tinham. Vale lembrar que eu só considerei as páginas que possuíam conteúdo na Wikipédia dos Estados Unidos, pois precisamos fixar uma língua para depois criarmos o ranqueamento em cima da comparação dessas 2. Se fossemos levar em consideração todas as línguas aumentaria muito a ordem de grandeza do processamento, e obteríamos uma tabela de dados muito mais esparsa, com dados faltantes.

5 Ranqueamento dos artigos faltantes

5.1 Coleta das estatísticas/features

Para conseguirmos criar um modelo de ranqueamento dos arquivos, precisávamos nos basear em quais estatísticas(features) seriam utilizadas para conseguirmos criar modelos de predição a nível de interesse que poderia ter para as pessoas que falam o idioma português. Como a quantidade de dados era enorme, e a disponibilidade dos

mesmos era muito restrita, foi utilizado como métrica para avaliar se uma página era relevante ou não a quantidade de acessos (page views) que aquela página obteve.

Esses dados puderam ser encontrados no site: https://dumps.wikimedia.org/ separados em anos, meses e dias. Cada um dos arquivos de dados que poderia ser baixado era dividido numa estrutura que poderia ser lida no formato "csv", em que os separadores seriam um espaço em branco. Os dados eram organizados da seguinte forma: A primeira coluna era a origem do dado (dependendo se o site era ou não propriamente dito da wikipédia, ele vinha com uma extensão, caso contrário ele vinha sem extensão, tendo seu valor igual a apenas o idioma de origem do artigo), a segunda coluna possuía o nome do artigo, a terceira a quantidade de acessos e a quarta o tamanho do artigo do wikipédia.

Como a quantidade de dados era muito grande, para testes do programa utilizamos apenas 1 dia de coleta de dados (11-05-2015), que já totalizavam 24 arquivos compactados (.gz), cada um com aproximadamente 400 MB de tamanho, totalizando mais de 10GB de dados. E mesmo utilizando apenas 1 dia de acesso, pela quantidade de dados, isso já se tornou um problema para se executar em um computador comum.

Para a análise dos dados foi utilizada a biblioteca "pandas" do python, para que fosse possível indexar os dados e conseguir relacionar essa base de dados com a base que tinha sido extraída na etapa de encontrar arquivos relevantes, que seriam futuramente classificados. Após fazer esse pareamento, os dados foram divididos na seguinte estrutura:

id_arquivo_wiki	page_views_eng	page_views_pt

De tal forma em que os conteúdos que não possuíam página em português tinham a coluna de page_views_pt sem nem um valor, e que teriam seu valor predito por um algoritmo de machine learning, para depois serem ranqueadas.

5.2 **Classificação e ranqueamento**

Antes de falarmos sobre os métodos de classificação propriamente dito, existe também uma lista com os tópicos mais importantes que todas as wikipédias de todos os idiomas deveriam ter: https://meta.wikimedia.org/wiki/List_of_articles_every_Wikipedia_should_have. Nós não estaremos cobrindo ela, consideramos somente os arquivos coletados e suas features para o nosso modelo de predição.

Ao fim da separação de dados da etapa anterior teremos nossas features salvas em arquivos com extensão ".csv", um completo com o valor de "page_views_pt" que seria usado para treinar o nosso classificador, e o outro com esse valor faltando, que terá seu valor predito pelo algoritmo. Com essa estrutura, foi criada uma sequência de testes para definir a escolha de um classificador mais apropriado. De acordo com o artigo original, eles utilizaram a técnica de regressão "random forest" como o que

obteve melhor performance. Para nosso teste, foi criada uma lista de classificadores para serem avaliados, dentre eles: Regrassão Logistica, KNN, Random Forest, Naive Bayes e XGBoost. Cada um desses classificadores possui sua própria característica, porém somente será utilizado para classificar o arquivo com dados faltando aquele que possuir o maior índice de AUC dentre eles. Essa métrica foi escolhida pois ela tende a ser pouco enviesada, podendo gerar melhores resultados na classificação. Essa métrica também é utilizada frequentemente em competições de aprendizado de máquina, e como no artigo ele não especifica qual métrica ele utilizou, essa pareceu bem adequada.

Depois da predição dos possíveis valores de page views que as páginas em português poderiam ter, basta ordenar em ordem decrescente a lista de resultados obtidos. Dessa forma, seria mais relevante a criação de uma página aquela que teria o seu valor de visualizações previsto mais alto.

5.3 **Implementação do modelo de classificação**

Para a implementação do sistema de classificação foi utilizada a biblioteca python do sklearn com os módulos para cada um desses classificadores. Para a separação dos dados em conjunto de treino e conjunto de teste, foi utilizado o método de KFold, separando os dados em 10 conjuntos distintos, e utilizando-os em 10 iterações distintas, com 9 dos 10 conjuntos para treino e o conjunto restante para o teste, gerando assim a métrica AUC. O funcionamento dessa técnica pode ser ilustrado como abaixo (Fig. 1):

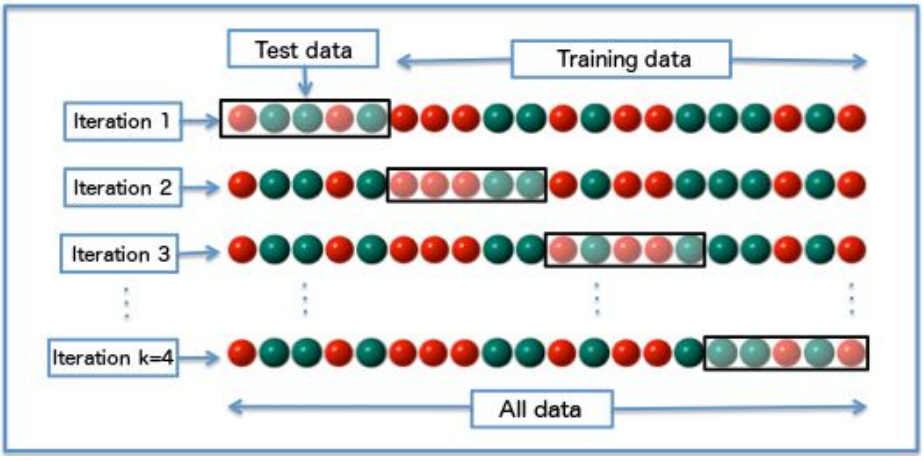

Fig. 1. A imagem ilustra o funcionamento do Kfold.

Em relação aos hyperparâmetros relacionados a cada classificador, foi utilizado o método de grid_search para ajustá-los de forma a maximizar a métrica AUC que aquele classificador obteria para cada iteração.

Ao fim de todas as iterações de todos os classificadores, selecionamos o com maior AUC resultante, e é utilizado o método predict do classificador para obter o valor da variável de pt_page_views.

Como nossa quantidade de features era muito restrita, só possuíamos 1 feature, nosso classificador funcionaria de forma linear, o que não seria tão eficiente. Porém o código foi feito para suportar uma quantidade n de features, bem generalizado. Logo se possuíssemos uma lista csv mais completa com os dados o algoritmo conseguiria executar da mesma maneira. Todo o código foi feito de maneira bem genérica para qualquer tipo de alteração dos dados.

6 Resultados

Infelizmente pela quantidade de dados, o programa não concluiu seu processamento, e o computador não suportou o programa. Ao tentar indexar a tabela de features de pageviews que as páginas possuíam, a quantidade de dados era muito extensa. Porém era necessário pegar todas as páginas, pois os dados não estavam ordenados.

Só foi possível verificar o funcionamento do programa em partes com pequenos testes de cada módulo separadamente, com pequenos dados gerados. Esse processamento, no entanto conseguiria ser realizado em uma máquina mais robusta, com mais memória.

7 Problemas encontrados

Um dos principais problemas encontrados foi a disponibilidade dos dados. No artigo eram enumerados vários datasets em que eram utilizados, mas sua disponibilidade não era tão trivial. Isso causou um grande atraso no desenvolvimento do projeto, e limitação das features que seriam utilizadas para os modelos de regressão.

Outro problema foi a quantidade de dados, todos de forma descentralizada. Mesmo pegando uma pequena parcela de dados para teste, como as bases de dados de arquivos do wikipédia era separada da base de dados de quantidade de acessos, como eles não possuíam também o mesmo padrão, para fazer o cruzamento de dados e associar um valor ao outro era necessário utilizar pelo menos toda a base de dados de um dos dois, e isso impossibilitou a execução em meu ambiente de trabalho.

8 Conclusão

A proposta de trabalho tem bastante relevância para a comunidade como um todo. Num mundo onde a informação possui cada vez mais valor, ampliar essa capacidade de obtenção de dados por meio do aumento de conteúdo linguístico sobre um tópico relevante para aquele idioma tem grande valor. Pelos desafios observados durante a execução do trabalho, vale ressaltar a necessidade de maior organização, padronização e centralização dos dados. De modo que fiquem mais acessíveis para que possam ocorrer mais pesquisas nessa área.

Infelizmente, devido a falta de processamento não foi possível obter resultados sobre o problema em si, e nem sequer comparar se o método de regressão escolhido pelo grupo que desenvolveu o artigo foi o mais interessante.

Referências:

1. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5092237/pdf/nihms823743.pdf
2. https://www.wikidata.org
3. https://www.mediawiki.org/wiki/API:Main_page
4. https://stackoverflow.com/questions/25100224/how-to-get-a-list-of-all-wikidata-properties
5. https://github.com/maxlath/wikidata-cli/blob/master/docs/read_operations.md
6. https://en.wikipedia.org/wiki/Wikipedia:Statistics
7. https://dumps.wikimedia.org
8. https://pandas.pydata.org/
9. http://scikit-learn.org/stable/index.html