PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
NÚCLEO DE EDUCAÇÃO A DISTÂNCIA
Pós-graduação Lato Sensu em Ciência de Dados e Big Data

Helder Geraldo Ribeiro

Um mecanismo de recomendação de textos semelhantes com base em tópicos ocultos

Belo Horizonte
2021

Contents

1 Introdução

1.1 Contextualização

Hoje em dia a geração de conteúdo acontece de forma muito rápida. A quantidade e complexidade dos dados faz com que o processamento manual de tudo que é gerado de seja impraticável, senão impossível, e é preciso automatizar determinadas tarefas para que tenhamos mais eficiência e agilidade para tirar valor e conhecimento dos dados. Uma das áreas onde a automatização pode ser feita é no processamento de textos e diversas informações podem ser extraídas dos textos que são escritos e compartilhados na internet, assim como em mídias privadas também.

Neste contexto se destaca o Processamento de Linguagem Natural (PLN), uma área que une Ciência da Computação, Linguística e Inteligência Artificial focada na interação entre computadores e linguagem humana e em especial na programação de máquinas para processar e analisar grandes volumes de textos em linguagem natural. Esta área contém um conjunto de técnicas que quando aplicadas a textos permitem que diversas tarefas sejam executadas sobre os mesmos.

Uma destas tarefas é processar este conteúdo escrito em linguagem humana e identificar contextos dentro dele, permitindo que possamos levantar algumas informações como tópicos que relacionam estes documentos. A identificação deste contexto pode não ser simples porque a mesma informação pode ser escrita de formas distintas utilizando linguagem natural. Ao mesmo tempo, extrair e analisar estas informações nos permite extrair conhecimento de nosso conjunto de textos como um todo.

1.2 O problema proposto

Alguns desafios da PLN envolvem reconhecimento de voz, compreensão de linguagem natural e até mesmo geração de conteúdo em linguagem natural. Neste trabalho o objetivo foi desenvolver um **mecanismo para fazer recomendação de documentos que tenham contextos semelhantes a outros documentos**, sendo estes de uma mesma base ou de bases diferentes. Para ilustrar vamos dar um exemplo de dois blogs, aqui chamados de A e B. Com o mecanismo de recomendações que propomos é possível extrair contextos ocultos do conjunto completo de ambos os blogs e com base neste conhecimento identificar posts no blog B que são parecidos com os posts do blog A.

Há várias utilidades para este tipo de recomendação. Eu, como escritor de um blog, posso escrever um texto e antes mesmo de publicar posso usar meu algoritmo de recomendação para procurar em um outro conjunto de textos conteúdos de alguma forma semelhantes ao que eu estou escrevendo. Isto permite que eu possa enriquecer e agregar valor ao meu conteúdo. Além disso é também uma excelente forma de aumentar o conhecimento próprio em áreas de meu interesse pessoal.

Mas o treinamento e recomendação não se limita apenas a comparar posts

de blogs. Qualquer conjunto de textos pode ser usado como base ou destino da comparação, desde que sejam processados de forma a servirem de insumos para algoritmos de extração de conteúdo oculto.

Neste trabalho eu usei duas fontes de dados diferentes, chamados aqui de origem e destino. A fonte de origem é a que eu usei seu conjunto completo de palavras (dicionário) para treinar o algoritmo de identificação de contextos ocultos. Para cada documento da fonte de origem eu usei o modelo de aprendizado gerado e algoritmos matemáticos para cálculo de distância entre valores de probabilidades para identificar o documento mais parecido na fonte de dados destino, que seria o que tivesse menor distância.

Como base de origem eu usei os posts do blog **Nerds Viajantes**, do qual sou editor e como base de destino eu usei um conjunto de páginas da **Wikipedia** e o resultado do trabalho foi encontrar dentro daquele conjunto a página que mais se assemelha-se a cada post do blog.

Apesar de ter estas fontes específicas usadas neste trabalho, a implementação foi feita de forma que seja fácil trabalhar com qualquer conjunto de textos como fonte e origem da recomendação. A coleta e limpeza devem ser feitos de forma específica por causa da variedade das fontes de textos mas o treinamento do modelo e o cálculo da semelhança foi feita em estruturas de dados genéricas abstraídas de forma a executar esta parte tão importante sobre qualquer conjunto de documentos.

2 Coleta de Dados

A coleta de dados envolveu obter de suas origens e gravar localmente os dados das fontes. Como executei o treinamento várias vezes eu achei melhor manter estes dados localmente.

2.1 Posts Nerds Viajantes

A coleta de dados dos posts do blog foi mais simples. O blog é publicado utilizando a ferramenta de gerenciamento de conteúdo *Wordpress*. Para manter o conteúdo ele usa o banco de dados *MySql*. Como sou editor e proprietário do blog, eu fiz um dump do banco de dados através da ferramenta de gerenciamento do serviço de hospedagem e carreguei localmente.

Em meu computador pessoal eu executo o *MySql* através de container da ferramenta *Docker*. Usando uma imagem de *MySql* como base, eu criei uma imagem própria que já carrega o dump gerado com os posts do blog como parte da imagem. Sendo assim, ao criar e executar um container eu já tenho os dados pré-carregados.

Os campos que utilizamos na tabela de posts são:

Nome do campo	**tipo do campo**
ID	bigint unsigned
post_date	datetime
post_content	longtext
post_title	text
post_name	varchar(200)
post_modified	datetime
comment_count	bigint

2.2 Wikipedia

A coleta de dados da Wikipedia deu mais trabalho e digamos que foi muito mais interessante. Tive dois desafios nesta parte: o que baixar e como baixar.

2.2.1 Definindo o subconjunto de páginas

Fazer a coleta de todas as páginas da Wikipedia nunca foi minha intenção. Nem sei quantas páginas há mas imagino que seja um número tão grande que não daria pra fazer nesse trabalho. Se não me engano há formas de baixar o conteúdo inteiro ou parte do conteúdo mas eu queria fazer a coleta utilizando a API oficial deles, então não poderia ser um conjunto tão grande e seria até complexo de gerenciar localmente. Além disso, o subconjunto de páginas que me interessa no trabalho é muito pequeno perto do universo completo da Wikipedia.

Meu blog sempre foi mais um hobby que algo profissional. Não temos muito conteúdo como aqueles que vivem desse ramo profissionalmente e geograficamente meus posts são limitados. A maioria de nossos posts vem dos seguintes

países: Argentina, Chile, Estados Unidos, Nova Zelândia, Islândia, além do Brasil. Decidi então procurar na Wikipedia conteúdo relacionado a estes países, mas não todos. E também não seria qualquer conteúdo, foquei em textos de alguma forma relacionados a Geografia, assunto que eu gosto muito e que influencia o conteúdo do blog, que tem muito conteúdo de lugares relacionados a contato com natureza.

A Wikipedia tem dois tipos de conteúdo importantes, categorias e páginas. Categorias são páginas especiais que tem uma lista de páginas ou subcategorias que a compõem. Páginas contém apenas conteúdo relacionado a um assunto.

Para fazer a coleta, inicialmente pensei em entrar na categoria principal de cada país, pegar a referência da categoria de Geografia (quase todos têm uma) e coletar toda a árvore a partir dela, começando pela Argentina. Logo de cara eu percebi que não era uma boa ideia pois diversas categorias tinham subcategorias ou até mesmo páginas que não eram relacionadas a Geografia. E além disso notei que algumas subcategorias da Argentina envolviam inclusive referências a categorias de coisas do Brasil.

Resolvi então fazer uma espécie de curadoria. No conjunto de categorias armazenadas localmente eu adicionei um status, que poderia ter um dos valores:

- **Check**: Aguardando aprovação para download da árvore embaixo daquela categoria
- **Waiting**: Aguardando download, já aprovado
- **Done**: Download feito com sucesso
- **Skeep**: Marcada para não fazer download no momento

Decidi armazenar localmente conjuntos de dados separados para categorias e páginas. Isto porque a curadoria escolhida para a coleta de categorias envolvia uma parte manual, que seria a escolha de qual categoria eu iria fazer o download das páginas nela contidas. Desta forma ficou mais fácil dividir o processo e fazer as duas coletas separadamente. A coleta de categorias, obviamente, teve que ser feita antes porque esta envolveria a gravação dos metadados de quais páginas seriam coletadas.

O processo completo envolveu duas iterações, uma para coleta das categorias e outra para coleta do conteúdo das páginas.

2.2.2 Coleta de categorias

Utilizando como controle os status mencionados anteriormente, eu segui o seguinte algoritmo:

1. Lê todas as categorias com status **Check**, ou seja, aguardando aprovação
   (a) Para cada categoria cujo assunto me interessava eu marcava com **Waiting**, ou seja, aprovada e aguardando download

   (b) Para cada categoria cujo assunto **não** me interessava eu marcava com **Skeep**, ou seja, ignore no momento

2. Lê todas as categorias aguardando download

3. Para cada categoria aguardando download, executar os passos:

   (a) Coleta os dados da categoria via API da Wikipedia

   (b) Grava cada subcategoria na collection categorias com o status **Check**

      i. Estas ficam aguardando aprovação manual em iteração seguinte

   (c) Grava os metadados de cada página localmente

      i. Ainda não grava os conteúdos, visto que estes são feitos com chamada a API diferente

      ii. O download do conteúdo é explicado na seção seguinte

   (d) Marca a categoria como download feito

4. Volta para o início (passo 1) para recomeçar o processo

Eu executei esta sequência de passos até ter uma quantidade de páginas boa o suficiente para identificar as semelhanças e ao final eu tinha 3015 páginas armazenadas localmente.

Exemplo de um documento contendo uma categoria:

```
{
    'pageid' : 67826,
    'ns': 14,
    'title': "Categoria:Geografia da Argentina",
    'type': "subcat",
    'download': "Done",
    'country': "Argentina"
}
```

2.2.3 Coleta de páginas

Para as páginas eu fiz download do conteúdo tanto em formato *wikitext* como *text*, mesmo que no final só tenha trabalhado a versão *text* porque achei mais fácil fazer *parse*. Foram executados duas iterações separadas, uma para cada tipo de conteúdo.

Para controlar quais páginas eu já tinha coletado ou não eu usei os dois status abaixo:

- **download**: Status do download do wikitext
- **download_text**: Status do download do wikitext

Neste caso como não envolveria a curadoria eu só usei dois valores possíveis:

- **Waiting**: Aguardando download
- **Done**: Download realizado com sucesso

A iteração de coleta neste caso foi muito mais simples e foi executada duas vezes, uma para formato de conteúdo.

1. Ler páginas que estão aguardando download
2. Para cada página aguardando download
   (a) Faz o download da página, gravando o conteúdo localmente
   (b) Marca a página como download realizado com sucesso

Exemplo de documento contendo uma página com conteúdo:

```
{
    'pageid': 22911,
    'ns': 0,
    'title': "Geografia da Argentina",
    'type': "page",
    'download': "Waiting",
    'country': "Argentina",
    'categories': Array
                "{{Sem notas}}{{Info/País/geografia f
                    ísica|...",
    'wikitext': "|preposićão = da |nome = ...",
    'text': "<div class="mw-parser-output"><style
        data-mw-deduplicate="TemplateStyl...",
    'text_clean': "A Geografia da Argentina é um domí
        nio de estudos e conhecimentos sobre..."
}
```

2.2.4 Ferramentas utilizadas

Localmente eu resolvi guardar o conteúdo no *MongoDB*. A escolha se deu pelo fato de que eu não sabia direito como seria meu modelo de dados e com a estrutura flexível de modelo orientado a documentos eu poderia mudar a estrutura a qualquer momento sem me preocupar com um esquema rígido previamente definido. Com o tempo se mostrou uma boa escolha pois acabei adicionando novos campos, alterando valores de documentos anteriores sem grandes problemas. A performance das operações também não se mostrou um problema.

A Wikipedia disponibiliza uma API Rest com vários serviços para coleta de conteúdo. Eu utilizei dois serviços, um para recuperação de dados das categorias e outro para recuperar o conteúdo. Para o conteúdo eu fiz a chamada ao serviço duas vezes por página, um cada formato.

Exemplos de urls dos dois serviços:

- Membros da categoria: https://pt.wikipedia.org/w/api.php?action=query&format=json&list=categorymembers&cmtitle=Category%3AArgentina&cmlimit=200
- Conteúdo da página: https://en.wikipedia.org/w/api.php?action=parse&format=json&pageid=3276454&prop=wikitext&formatversion=2
  - O campo *prop* indica o formato do conteúdo retornado, sendo *wikitext* ou *text*

Para fazer a chamada aos serviços da API eu usei o pacote *requests* do Python, que retorna o resultado em formato *JSON*.

2.3 Verbos em língua portuguesa

Como parte do trabalho eu precisei da lista de verbos em língua portuguesa para analisar o levantamento de tópicos em documentos após remoção de verbos, que no contexto deste trabalho poderiam ser pouco relevantes ou até mesmo atrapalhar na definição dos tópicos.

A lista de verbos foi recuperada fazendo webscraping do site https://www.conjugacao.com.br/verbos-populares. Foi utilizada o pacote requests do Python para fazer a requisição das páginas e o pacote Beautiful Soup para extrair o conteúdo do resultado html retornado.

O total de verbos contidos nesta página é de 5000 e para que fossem retornados todos foram necessárias 50 requisições (são 100 verbos em cada página), uma para cada página de conteúdo no site.

Os verbos são gravados localmente em uma collection do MongoDB para que possam ser utilizados mais de uma vez sem necessidade de download a cada execução. Cada documento gravado no MongoDB tem dois campos, o verbo original e o verbo *stemmed*, conforme exemplo abaixo para o verbo *falar*.

```
{
    'verbo': 'falar',
    'verbo_stemmed': 'fal'
}
```

3 Processamento/Tratamento de Dados

Nem sempre utilizamos o texto inteiro quando queremos extrair conhecimento do seu conteúdo, que no nosso caso significa encontrar tópicos com suas palavras e encontrar documentos em que estes tópicos estão presentes. Muitas palavras não são interessantes, como por exemplo aquelas que não agregam valor mas influenciam o treinamento de modelos, como as conhecidas *stopwords*. Estas podem, inclusive, atrapalhar o processo de aprendizado ao adicionar ruído ao conjunto de dados e aumentar o custo computacional e o tempo de execução.

A limpeza de texto consiste então em remover dos nossos documentos originais este conjunto de palavras que não somente não agregam ao treinamento como atrapalham o seu funcionamento adequado. Parte das tarefas de limpeza é comum a todas as fontes de dados e parte é específica por fonte de dados escolhida.

Estas atividades são na essência uma tentativa de conversão de linguagem de texto para algo mais próximo do que o computador consegue entender melhor, preparando os dados de textos para o processamento de linguagem natural.

3.1 Limpeza comum

Algumas tarefas de limpeza de texto são úteis e devem ser empregadas a qualquer fonte de dados, mesmo que com parâmetros diferentes.

- Remoção de tags HTML: são apenas marcadores para definir a formatação de conteúdo html e não devem fazer parte dos dados de treinamento ou teste
- Remoção de pontuações: servem apenas para definir a estrutura do texto e não devem ser utilizados no treinamento
- Conversão para minúsculo: todo o conteúdo deve ser normalizado convertendo as letras para minúsculas de forma que palavras escritas com *case* diferente não sejam tratadas como entidades distintas
- Remoção de *stopwords*: contempla a remoção de *stopwords* gerais da linguagem dos textos, em nosso caso português. Esta lista é obtida de um pacote de processamento de linguagem natural
- Remoção de palavras adicionais: remover do texto palavras que não são *stopwords* mas atrapalham o treinamento de alguma forma. É uma questão que deve ser avaliada especificamente para cada fonte de dados adicionais pois a lista é diferente em cada fonte de dados. Esta lista é obtida após análise de exploração e conhecimento da fonte de dados
- *Stemming*: redução de palavras para sua forma raiz, necessário para uma mesma palavra que seja escrita de formas distintas seja tratada como apenas uma entidade. Além disso este tratamento diminui a dimensão do conjunto total de palavras e reduz custo computacional. Como exemplo podemos citar as palavras *visitei* e *visita* que seriam reduzidas a *visit*

Para exemplificar a execução de limpeza de texto vamos usar a seguinte frase:

"Após visitar a Lagoa Dourada nós decidimos voltar o hotel. Foi um dia muito bonito, esta foi uma das lagoas mais bonitas que já visitamos."

Após a limpeza temos como resultado a seguinte lista de palavras:

['após', 'visit', 'lago', 'dour', 'decid', 'volt', 'hotel', 'foi', 'dia', 'bonit', 'lago', 'bonit', 'visit']

Importante perceber alguns alterações importantes em relação ao texto inicial:

- Conversão da palavra *Após* para *após* com o *a* minúsculo
- Remoção de *stopwords* como *a, nós, o, um, esta, uma, das, mais, que, já*
- Alteração de *case* e conversão da palavra *Lagoa*, desta forma as duas referências a *lagoa* que estavam escritas de forma diferente agora são tratados como uma mesma entidades
- Remoção de pontuações

3.2 Posts Nerds Viajantes

3.2.1 Limpeza de palavras nos posts

Além da limpeza de texto padrão, foi necessário eliminar outros conjuntos de palavras dos posts do blog. Após execução da análise dos tópicos nas primeiras execuções de treinamento do modelo foi identificado que alguns conjuntos de palavras não fazem sentido ser usados na definição do dicionário.

- Remoção de *caption*: A ferramenta *Wordpress* utilizada uma notação específica para adicionar legendas às imagens, que não corresponde ao padrão html e por isto foi necessária uma limpeza específica deste conteúdo
- Remoção de *stopwords* específicas: Palavras que não fazem parte da lista de *stopwords* da linguagem mas que mesmo assim atrapalham no treinamento do modelo
  - Exemplos: fot, fic, visit, fotograf, dia, algum, par, passei, bem, restaurant
  - A lista completa destas palavras se encontra no arquivo remocao_nerds_viajantes.txt no repositório do projeto.
- Remoção de verbos: A ideia da comparação dos documentos foi mais direcionada a características dos locais visitados e decidi remover os verbos do texto pois estes contribuiam muito na composição dos tópicos

3.2.2 Limpeza de posts inteiros

Além da limpeza de palavras dentro dos posts, foi necessário também remover alguns posts do conjunto a ser utilizado no treinamento.

Nosso blog também tem um pouco de foco em fotografia e por muito tempo disponibilizamos **papéis de parede** de fotos que nós tiramos e gostamos. Para divulgar os papéis de parede escrevemos vários posts e estes não são interessantes para o propósito deste trabalho, por isso resolvi removê-los. A remoção foi relativamente fácil, visto que todos os posts deste conjunto tem o atributo *name* começando com o prefixo *papel-de-parede*.

Outro critério para remoção de posts inteiros foi o **tamanho do documento em tokens**. Em várias referências sobre o algoritmo de identificação de tópicos LDA, é recomendado utilizar documentos cuja quantidade de tokens seja acima de 40 ou 50. Analisando os tamanhos posts do blog eu notei que 42 tokens seria um bom limite e este passou a ser o tamanho a partir do qual os documentos seriam usados no treinamento.

Além dos dois conjuntos acima eu também optei por analisar manual e visualmente o conjunto de posts para remover pontualmente alguns cujo conteúdo não seria interessante no treinamento. Páginas na ferramenta *Wordpress* são consideradas posts e a página de contato, por exemplo, não interessa no nosso contexto. Posts com conteúdo publicitário também foram removidos. Posts com conteúdo de valor temporal (divulgação de um evento, por exemplo) que não tem mais valor atualmente também foram removidos. A lista completa pode ser consultada no módulo de limpeza de posts.

3.3 Wikipedia

No caso dos posts do blog a limpeza de tags html em posts é feita no carregamento dos dados antes do treinamento, mas no caso das páginas da Wikipedia eu resolvi fazer esta limpeza apenas uma vez e gravar em um atributo separado, aproveitando as facilidades do MongoDB para alteração na estrutura de documentos.

Ao fazer um teste com cerca de duas mil páginas o tempo para remover as tags foi mais de um minuto e como seria um trabalho a ser executado a cada treinamento eu achei melhor já fazer este processo previamente.

O campo utilizado como base para a limpeza foi o *text*, que contém o conteúdo em html. Neste campo o conteúdo da página fica dentro de parágrafos (atributo html *p*) e elementos pré-formatados (atributo html *pre*).

Para cada texto eu utilizei o componente *BeautifulSoup* do módulo *bs4* para extrair todos os parágrafos e pré-formatados para, em seguida, gravá-los em um campo separada chamado *text_clean*, que foi usado no identificação de documentos semelhantes.

O código fonte completo pode ser encontrado no módulo de limpeza da Wikipedia.

4 Análise e Exploração dos Dados

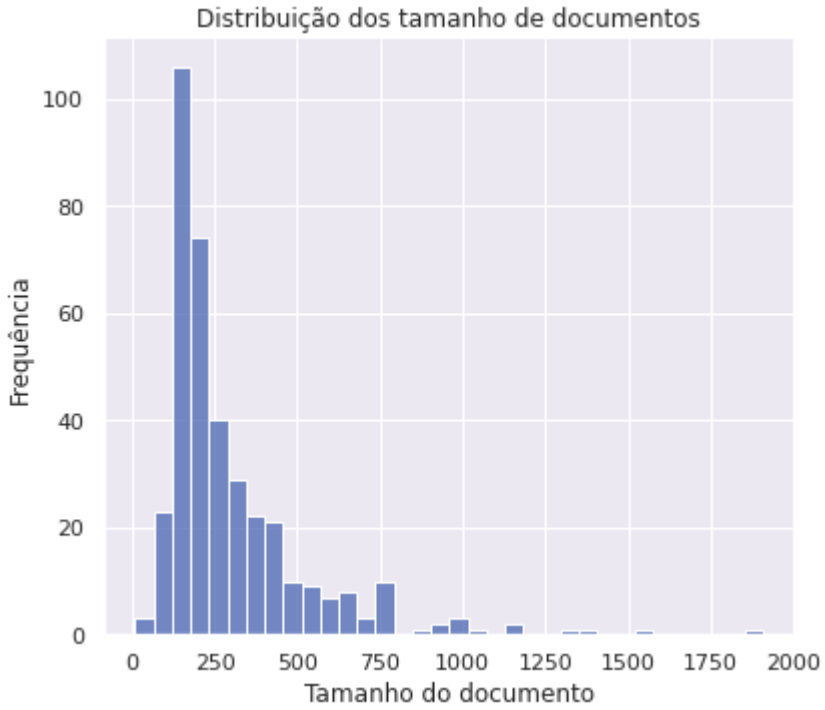
Antes de fazer qualquer trabalho de treinamento dos dados é preciso conhecer os dados em que estamos trabalhando. No contexto deste trabalho duas coisas muito importantes a se compreender são os tamanhos dos documentos a serem trabalhados e o conjunto de palavras que fazem parte do dicionário completo. As análises feitas demonstradas a seguir foram feitas na fonte de dados de posts do blog pois esta foi usada para criar o dicionário e fazer o treinamento do modelo.

4.1 Análise do tamanho dos documentos

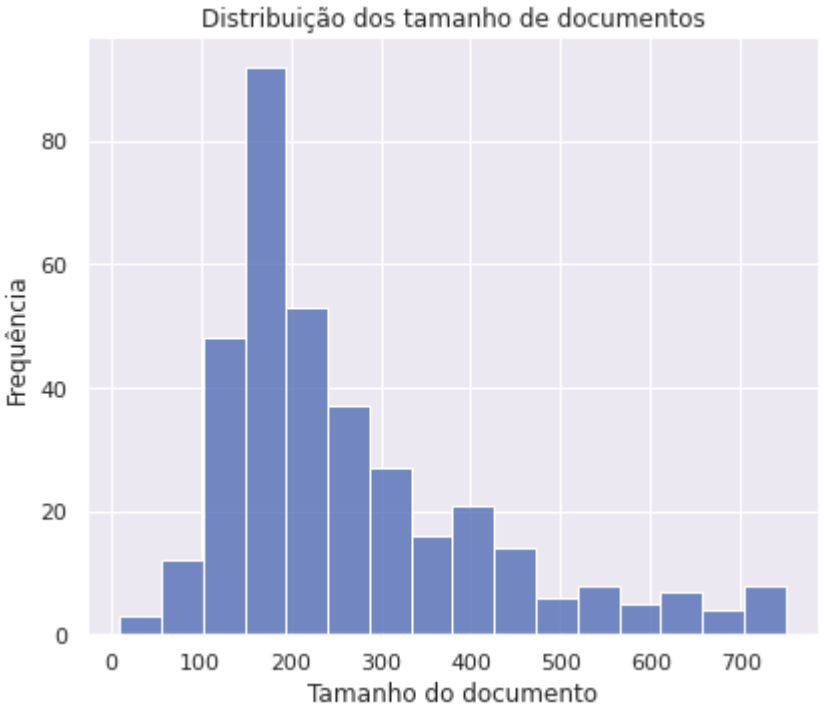
É preciso entender entre os documentos que estamos trabalhando o tamanho dos documentos em tokens principalmente porque documentos muito pequenos podem não ser recomendados para algoritmos com limitação em *corpus* de dados com documentos pequenos. O algoritmo LDA, por exemplo, utilizado para encontrar tópicos em documentos, não tem boa performance com documentos pequenos (em documentação encontrada na internet recomenda-se documentos acima de 40 ou 50 tokens).

4.1.1 Distribuição de tamanhos de documentos

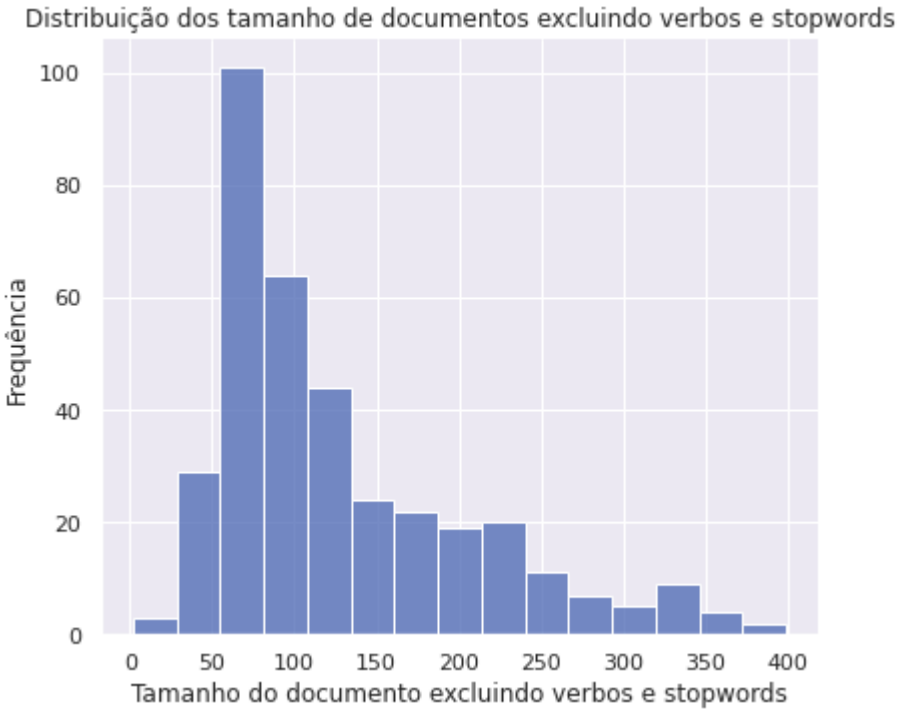
Para melhor compreender o tamanho dos documentos no meu conjunto de textos eu resolvi usar uma histograma para exibir a distribuição de tamanhos dos documentos após executar a limpeza básica de tokens (remoção de tags html e stopwords da linguagem portuguesa). O gráfico a seguir mostra esta distribuicão.



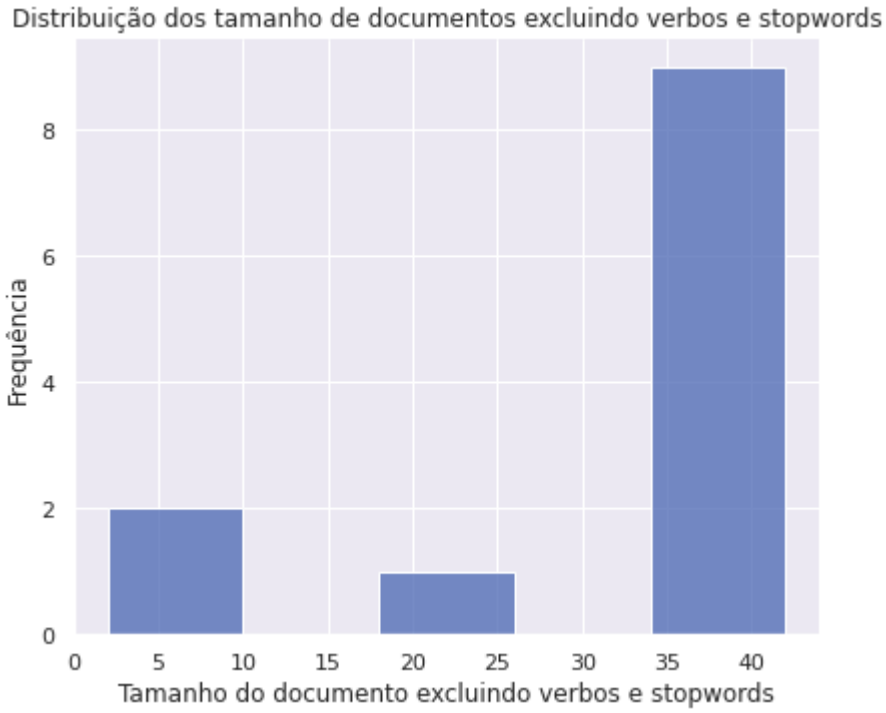
Notei que a maioria dos documentos se concentra em uma faixa de até 750 tokens, então resolvi limitar minha visualização gráfica do eixo x a este valor. Pode-se perceber pelo gráfico abaixo que a faixa de maior concentração de documentos é entre 150 e 200 tokens, o que é um tamanho bom para o algoritmo que escolhemos. Nota-se também que uma quantidade pequena de documentos se encontra abaixo de 50 tokens, o que também é muito bom. Mais adiante veremos uma distribuição estatística para validar este intervalo.



Uma ideia que tive ao longo do desenvolvimento deste projeto foi remover do meu dicionário de treinamento algumas palavras que apareciam com frequência e que não agregavam nada à comparação dos documentos. Após verificar a frequência eu também achei interessante poder remover os verbos do conjunto. Sendo assim eu fiz uma análise do tamanho dos documentos excluindo estes dois conjuntos de palavras. No gráfico abaixo é exibida a distribuição dos tamanhos dos documentos após excluir verbos e *stopwords* específicas. O limite de 400 tokens no eixo x foi usado para detalhar melhor a visualização.

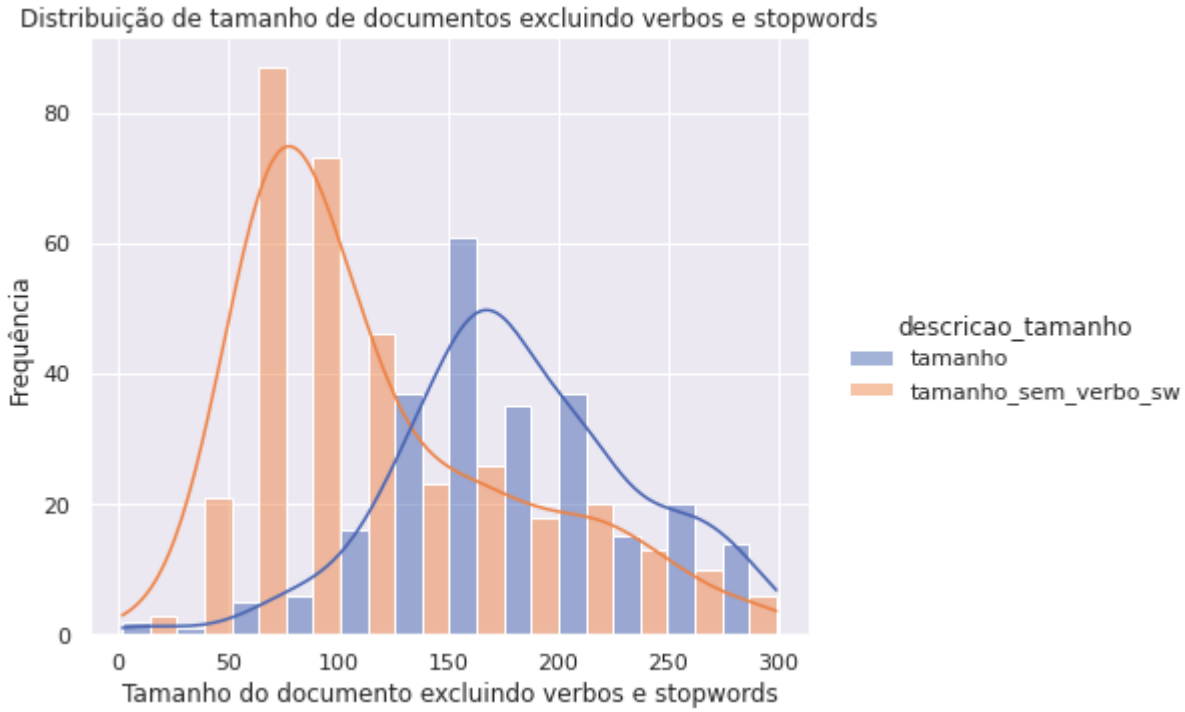


Mesmo limitando o conjunto anterior a 400 tokens dá pra perceber que há uma faixa de cerca de 30 documentos com tamanho entre 25 e 50 tokens e achei melhor detalhar mais. Coloquei um limite em 42 para ver graficamente as faixas de tamanhos abaixo dessa linha. Analisando visualmente percebe-se que 12 documentos estão abaixo deste valor, o que é muito pouco perto do total de 378 documentos que restaram no conjunto de posts do blog após a limpeza de posts explicada no processamento de dados.

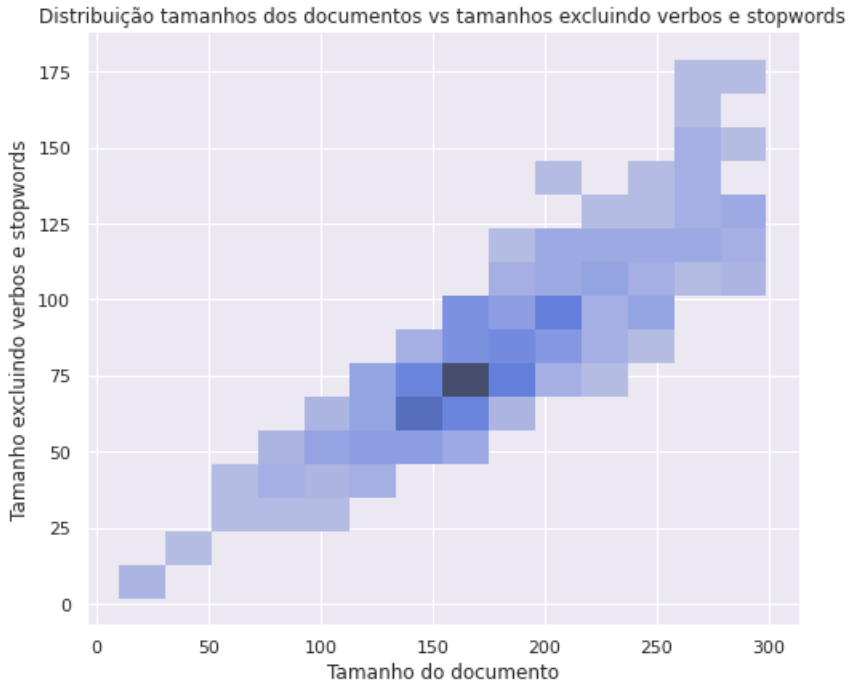


Para ilustrar um pouco mais esta questão da distribuição dos tamanhos dos documentos eu resolvi criar um gráfico com a comparação entre o comportamento dos tamanhos dos documentos originais e o tamanho dos documentos após a remoção de verbos e palavras específicas. Novamente coloquei limite no tamanho dos documentos para melhor visualização da área mais concentrada.

A imagem a seguir nos mostra ao mesmo tempo que antes os tamanhos dos documentos originais se concentrava entre 150 e 200 tokens e após a remoção dos dois conjuntos de dados esta faixa de maior concentração passou para entre 50 e 100.



Outra forma interessante de ver a relação entre os tamanhos dos documentos antes e após a remoção dos conjuntos de dados é utilizar um mapa de calor como este a seguir. As cores mais fortes indicam que maioria dos documentos tinha tamanho entre 150 e 175 antes da remoção e passou a ter tamanhos entre 65 e 75 sem os verbos e *stopwords* específicas.



4.1.2 Análise estatística dos tamanhos dos documentos

Uma outra forma de entender a distribuição dos tamanhos dos documentos é uma descrição estatística. Abaixo está uma tabela com várias métricas relacionadas a estes tamanhos, como média, desvio padrão, valor mínimo, valor máximo e percentis diversos. Optei por incluir, 5%, 10%, 90% e 95% nos percentis para ver como os tamanhos se comportam para minorias e maiorias.

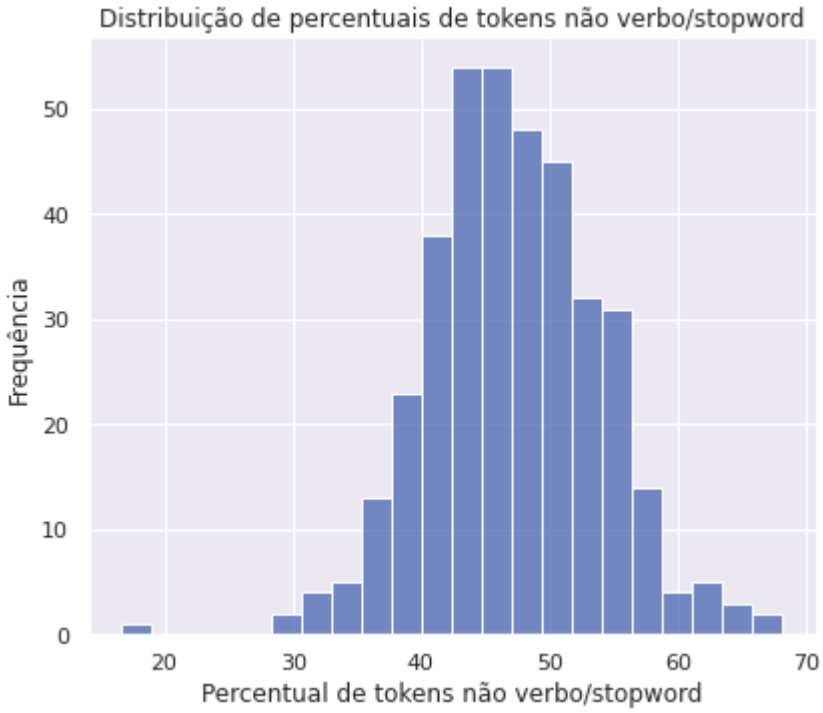
Após a remoção a média do tamanho de documentos ficou em cerca de 145 tokens e o percentil de 5% em 49 nos mostra que ainda assim a maioria dos documentos está com tamanho aceitável.

Métrica	Documento Original	Documento sem verbos ou *stopwords* específicas
count	378.000000	378.000000
mean	303.706349	145.137566
std	234.920633	118.516928
min	10.000000	2.000000
5%	111.700000	49.000000
10%	133.000000	57.700000
25%	165.000000	73.000000
50%	216.000000	101.000000
75%	367.000000	179.000000
90%	589.200000	276.600000
95%	747.450000	349.600000
max	1909.000000	1033.000000

4.2 Análise de palavras das fontes de dados

4.2.1 Distribuição de palavras

Assim como no tamanho dos documentos eu resolvi analisar para cada documento o percentual de palavras que não é verbo nem *stopword* específica do contexto do trabalho. O gráfico abaixo mostra uma distribuição destes percentuais e podemos ver que a maioria está entre 40% e 58%, o que significa que após a remoção dos dois conjuntos de dados o tamanho destes documentos seriam reduzido para esta proporção. Felizmente vimos na análise gráfica dos tamanhos dos documentos que mesmo após o encurtamento os tamanhos da maioria permanecia satisfatório.



4.2.2 Análise estatística das fontes de palavras

A primeira informação e mais simples é o total de palavras no dicionário. Percebe-se pelas informações abaixo que após a remoção de verbos e palavras específicas o dicionário caiu para 48% de seu tamanho original.

- Total de palavras: 114801
- Total de palavras após remoção de verbos e *stopwords* específicas: 54862

A seguir eu procurei saber os 20 tokens mais frequentes do conjunto de dados. No conjunto original as 10 palavras que mais frequentes, assim como a quantidade de vezes que aparecem é:

```
[('fot', 1550), ('fic', 807), ('algum', 750), ('dia',
    720), ('par', 660), ('caminh', 639), ('faz', 634)
    , ('cheg', 611), ('pod', 599), ('visit', 581), ('
    fotograf', 566), ('passei', 560), ('parqu', 557),
    ('trilh', 554), ('outr', 551), ('bem', 538), ('
    cidad', 534), ('pass', 526), ('muit', 502), ('hor
    ', 489)]
```

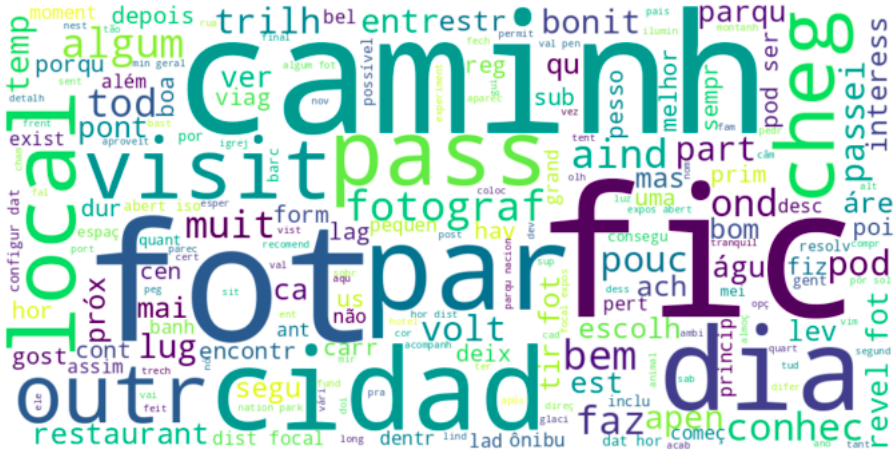
Após a remoção de verbos e algumas *stopwords* específicas os 10 tokens mais frequentes são:

```
[('parqu', 557), ('cidad', 534), ('temp', 484), ('
    pouc', 422), ('lag', 364), ('jalap', 361), ('águ',
     353), ('aind', 350), ('bonit', 348), ('vist',
    338), ('bel', 322), ('pra', 297), ('lug', 297), ('
    fiz', 296), ('áre', 276), ('expos', 273), ('lad',
    268), ('cacho', 253), ('pais', 246), ('qu', 245)]
```

4.2.3 Nuvem de palavras

Uma outra forma de informar visualmente as palavras mais frequentes no dicionário é uma nuvem de palavras. Abaixo vemos duas imagens.

A primeira ilustra o dicionário completo. Nota-se a predominância da palavras *fot* e também dos verbos *fic* e *caminh*, removidos ao fazer o treinamento. Na subseção anterior já tivemos a oportunidade de ver como estas palavras são predominantes.



A segunda ilustra a frequência das palavras após a remoção de verbos e *stopwords* específicas.

4.3 Análise de palavras e documentos da Wikipedia

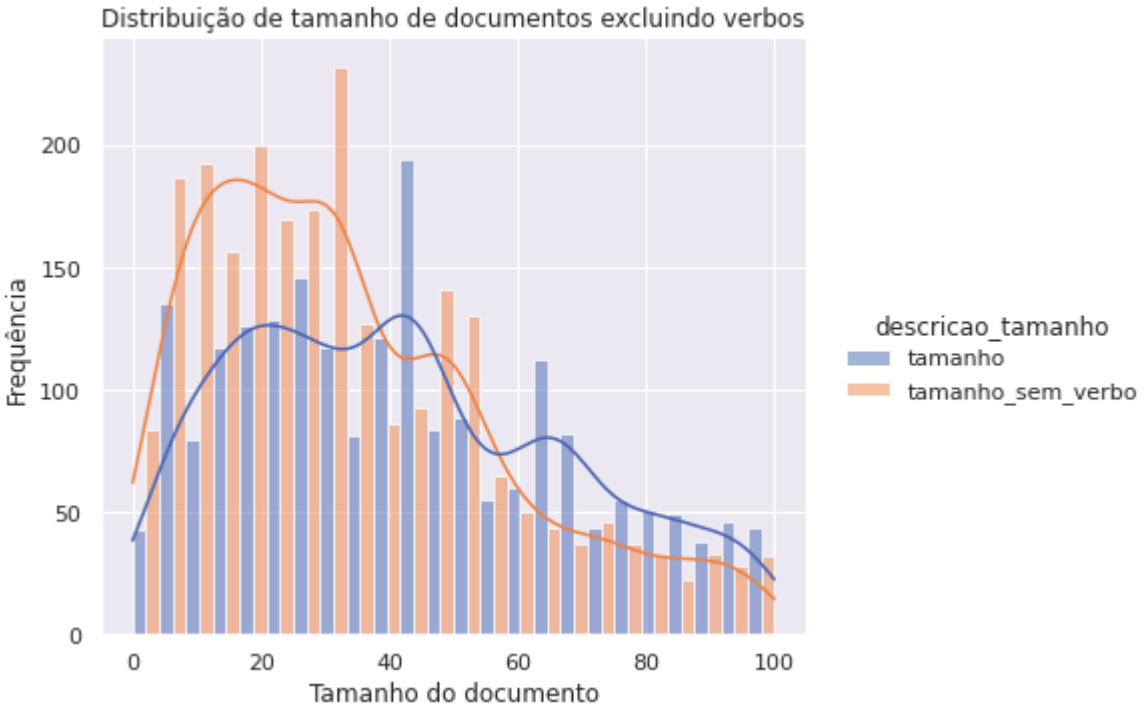
Apesar de não ser o modelo utilizado no treinamento eu fiz uma breve análise sobre as páginas obtidas da Wikipedia.

A primeira informação que levantei foi sobre o total de palavras que aparecem nos documentos. Percebe-se pelas informações abaixo que após a remoção de verbos o dicionário caiu para 62% de seu tamanho original.

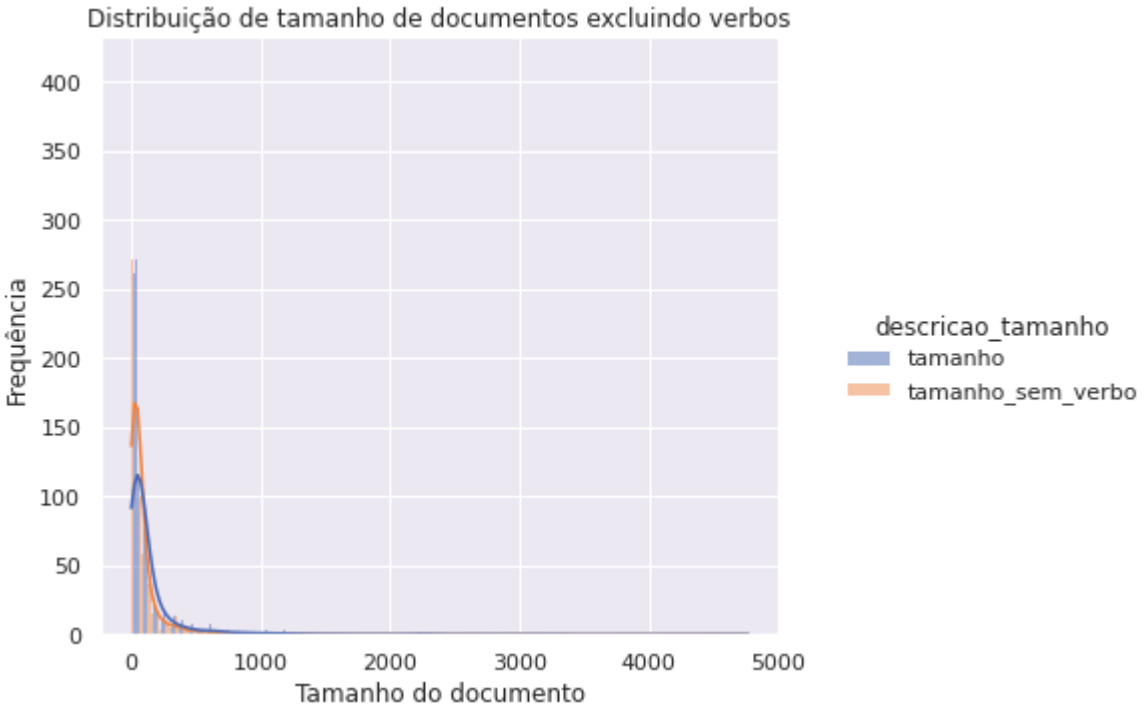
- Total de palavras: 469499
- Total de palavras após remoção de verbos e *stopwords* específicas: 291325

Em seguida fiz a análise comparando a distribuição dos tamanhos dos documentos em tokens na versão original (com stopwords básicas da língua portuguesa removidos) com a versão sem verbos e pude constatar visualmente que há muitos documentos pequenos, conforme gráfico abaixo.

Para conseguir visualizar tive que limitar bastante o tamanho dos documentos e só exibir no gráfico a parte que envolve apenas documentos com menos de 100 tokens.



Para referência segue a distribuição sem limitar o tamanho máximo de tokens no eixo x.



Em seguida fiz um levantamento dos 20 tokens mais frequentes ao longo de todo o dicionário. Sem remoção de verbo a lista é:

```
[('est', 3996), ('reg', 3551), ('áre', 3237), ('local
    ', 2966), ('parqu', 2893), ('rio', 2765), ('brasil
    ', 2543), ('part', 2367), ('grand', 2043), ('cidad
    ', 1811), ('nacion', 1765), ('sul', 1762), ('popul
    ', 1708), ('ilh', 1630), ('mai', 1605), ('nort',
    1587), ('ano', 1565), ('argentin', 1507), ('outr',
     1480), ('form', 1473)]
```

Após a remoção dos verbos em língua portuguesa os 20 mais frequentes foram:

```
[('áre', 3237), ('parqu', 2893), ('rio', 2765), ('
    brasil', 2543), ('grand', 2043), ('cidad', 1811),
    ('nacion', 1765), ('sul', 1762), ('popul', 1708),
    ('ano', 1565), ('argentin', 1507), ('outr', 1480),
     ('km2', 1467), ('águ', 1449), ('espéci', 1271),
    ('provínc', 1214), ('americ', 1144), ('tod', 1087)
    , ('municípi', 1075), ('unid', 1039)]
```

E por último mas não menos importante fiz a análise estatística do tamanho dos documentos do conjunto de páginas e percebi que a quantidade de documentos considerados pequenos para LDA é grande. Por este motivo eu resolvi

executar treinamento e recomendação com dois cenários distintos, um envolvendo todas as páginas da Wikipedia e outro envolvendo um conjunto reduzido apenas com páginas com tamanho maior ou igual a 40 tokens.

Métrica	**Tamanho Original**	**Tamanho sem Verbo**
count	3016.000000	3016.000000
mean	155.669430	96.593170
std	441.581739	257.269247
min	0.000000	0.000000
5%	8.000000	5.000000
10%	15.000000	10.000000
25%	29.000000	20.000000
50%	58.000000	37.000000
75%	123.000000	81.000000
90%	306.000000	197.000000
95%	598.000000	349.250000
max	10018.000000	5699.000000

5 Treinamento de modelo

O treinamento tem como objetivo fazer o ajuste de um modelo que identifique os tópicos que compôem um conjunto de textos (*corpus*) e que seja capaz de encontrar os tópicos mais importantes na definição do conteúdo dos textos. A lista de tópicos dos textos, com suas respectivas probabilidades, serão o insumo para o cálculo de semalhança entre os documentos, conforme veremos mais adiante.

5.1 LDA - Latent Dirichlet Allocation

LDA é um algoritmo de aprendizado não supervisionado que associa tópicos a documentos. Documentos são como textos e cada um pode conter mais de um tópico, assim como um tópico pode estar presente em vários documentos. Tópicos são formados por palavras e uma mesma palavra pode fazer parte de mais de um tópico, ou seja, uma mesma palavra contribui para a formação de vários tópicos. Os tópicos são descobertos durante o treinamento do modelo mas a quantidade de tópicos deve ser especificada a priori (é um dos parâmetros de ajuste).

Após o treinamento do modelo um documento tem uma distribuição discreta de tópicos e um tópico tem uma distribuição discreta de palavras. Na seção *Testes de análise de tópicos* há exemplos de topicos extraídos dos documentos da base de treinamento, assim como palavras que contribuem na definição dos tópicos.

Há vários tipos de uso para o algoritmo LDA, como entender melhor o tipo de documento um determinado conjunto de palavras (notícias, artigo na wikipedia, negócios), quantificar as palavras mais usadas e mais importantes em um texto ou mesmo encontrar semelhanças e recomendações de documentos.

Neste trabalho eu usei LDA para definir a distribuição de tópicos em textos (e consequentemente das palavras destes tópicos) e com base no modelo ajustado calculei a semalhança entre textos de acordo com as probabilidades dos tópicos do modelo treinado encontrados para os documentos das bases de treinamento e comparação. Enfim, com base nestas probabilidades eu calculei a distância entre os documentos e fiz o cálculo de recomendações de textos novos considerados semelhantes aos da base de treinamento.

LDA não tem boa performance com documentos curtos, como tweets, por exemplo, pois ele infere parâmetros a partir da observação de palavras e se não há palavras suficientes não há as condições necessárias para um bom aproveitamento. O algoritmo trabalha com modelos do tipo bag of words, ou seja, não há importância na ordem das palavras. Outros algoritmos funcionam bem com sentenças estruturadas.

5.1.1 Hiperparâmetros

Há vários parâmetros que podemos variar no treinamento de um modelo com LDA mas além do número de tópicos eu resolvi variar os seguintes hiperparâmet-

ros:

- α (alpha): Um valor baixo indica que documentos contém poucos tópicos contribuindo para os mesmos
- η (eta): Um valor baixo indica que tópicos contém poucas palavras contribuindo para os mesmos, enquanto em um valor alto pode haver maior sobreposição de palavras entre tópicos diferentes
- passes: Número de vezes que o algoritmo passa por cada documento no levantamento de tópicos

5.2 Cálculo de coerência

Um dos parâmetros mais importantes para o algoritmo LDA é o número de tópicos escondidos que devem ser extraídos do corpus de treinamento.

Sendo o LDA um algoritmo não supervisionado é muito difícil validar se o conjunto de tópicos selecionados para um determinado conjunto de documentos é útil e faz sentido. Não há uma lista de tópicos corretos previamente selecionados para comparar com os resultados obtidos. Uma forma de medir a eficiência do número de tópicos escolhidos é a **coerência**, que é uma avaliação quantitativa da qualidade dos tópicos aprendidos para um determinado conjunto de documentos.

Um conjunto de afirmações ou fatos é considerado coerente, se apoiarem um ao outro. Assim, um conjunto de fatos coerente pode ser interpretado em um contexto que cobre todos ou a maioria dos fatos. Um exemplo de conjunto de fatos coerentes é "o jogo é um esporte de equipe", "o jogo é jogado com uma bola", "o jogo exige grandes esforços físicos".

A coerência de um tópico mede o grau de semelhança semântica entre as palavras que mais contribuem para a definição daquele tópico. Esta medida ajuda a distinguir entre os tópicos que são **semanticamente interpretáveis** e os tópicos que são **artefatos de inferência estatística**.

Há várias medidas de coerência e a que escolhi foi a **c_v**. Na lista de referências há artigos que explicam esta e outras medidas de coerência. O que importa é que um maior valor indica maior coerência.

5.2.1 Análise de coerências

Conforme podemos ver no código fonte listado na seção seguinte, há alguns parâmetros importantes para o cálculo da coerência:

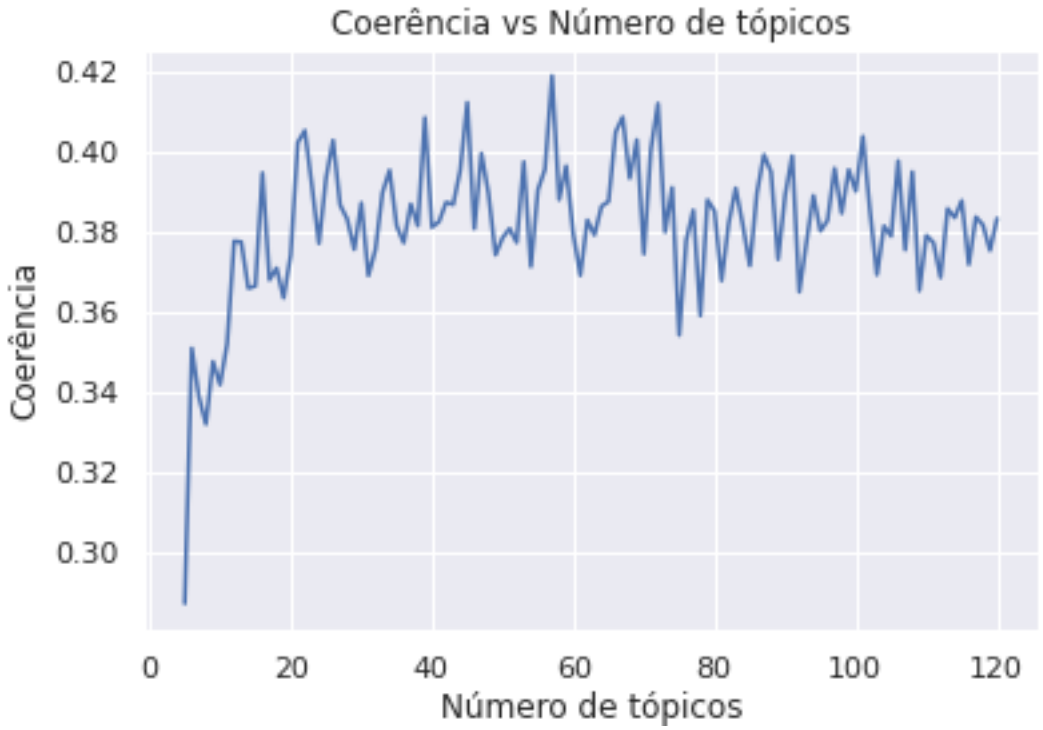
- **documents**: lista de documentos que compõem nosso corpus
- **num_topics**: número de tópicos para os quais queremos calcular a coerência

Apesar de não ser explicitamente utilizados no código, os parâmetros **alpha** e **eta** mencionados anteriormente são também importantes mas neste cálculo inicial optei por mantê-los em seu valor padrão, que é 1/número de tópicos.

A lista de documentos contempla o conjunto completo de textos de nossa fonte de dados de origem. O número de tópicos eu variei entre 5 e 120, de um a um. O resultado gerado foi um arquivo csv contendo os seguintes campos:

- num_topics: número de tópicos no Treinamento
- coherence: coerência para aquele número de tópicos
- tempo_gasto: tempo total gasto no cálculo de coerência para o número de tópicos

Com base no resultado dos cálculos das coerências para o número de tópicos foi gerado o gráfico a seguir, onde já podemos ver que a coerência para de crescer e de certa forma e começa a oscilar para cima e para baixo sem grandes variações a partir de um determinado número de tópicos.



Ordenando os valores das coerências nós obtemos as dez maiores e seus respectivos tópicos, conforme vemos abaixo ordenando começando pela maior.

Esta lista da quantidade de tópicos com maior coerência foi muito importante porque dela eu tirei os números de tópicos que eu escolhi para variar o

parâmetro *num_topics* na execução do ajuste do modelo e comparação entre os documentos.

Número de tópicos	Coerência
57	0.4187279439804192
45	0.4120513470103957
72	0.4118140382511174
67	0.4084148429857527
39	0.4083294457749056
22	0.4050952035903351
66	0.4047401968259075
101	0.4036451569534363
69	0.4027847252786048
26	0.4026587678111619

Abaixo segue uma listagem com o código fonte usado para o cálculo das coerências para um conjunto de documentos e intervalo de números de tópicos.

5.2.2 Código fonte

```
from gensim.corpora import Dictionary
from gensim.models import LdaModel, CoherenceModel

import pandas as pd
import time

class CalculadorCoerencia:

    def __init__(self, fonte_dados_origem):
        self.__documentos = fonte_dados_origem.get_tokens()

    def calcular_coerencia(self, num_topics, corpus, id2word):
        """Compute coherence score given a number of topics."""
        lda = LdaModel(corpus=corpus, id2word=id2word, num_topics=
    num_topics,
            passes=20, random_state=0)
        cm = CoherenceModel(model=lda, texts=self.__documentos,
    dictionary=id2word, coherence='c_v')
        return cm.get_coherence()

    def calcular_coerencias(self, topicos_inicio, topicos_fim,
    debug=False):
        id2word = Dictionary(self.__documentos)
        corpus = [id2word.doc2bow(documento) for documento in self.
    __documentos]

        coherence = pd.DataFrame(index=range(topicos_inicio,
    topicos_fim + 1),
            columns=['coherence', 'tempo_gasto'])

        for i in coherence.index:
            if debug:
```

```
28                 print(f'Calculando coerencia para {i} topicos...')
29             tempo_antes = time.time()
30             c = self.calcular_coerencia(i, corpus, id2word)
31             tempo_depois = time.time()
32             coherence.loc[i, 'coherence'] = c
33             coherence.loc[i, 'tempo_gasto'] = tempo_depois -
    tempo_antes
34 
35         return coherence
```

Listing 1: Código fonte: Cálculo de coerência de tópicos

5.3 Testes de análise de tópicos

Após a definição de modelo com base nos documentos de origem foram feitos testes para avaliação de quais tópicos determinavam melhor os documentos e quais palavras definiam os tópicos com maior probabilidade. Como a execução deste algoritmo é não determinística as palavras (e seus *id's*) que definem um tópico pode ser alteradas de uma execução para outra. O resultado exibido aqui pode não ser o mesmo de execuções seguintes com os mesmos parâmetros mas o objetivo deste teste inicial é identificar palavras que estão contribuindo muito para definição dos tópicos mas que podem não ser interessantes para o objetivo de encontrar documentos semelhantes, então isto não é um problema.

Os testes foram feitos utilizando a fonte de dados do blog, que foi a que usei para o treinamento do modelo. A quantidade de tópicos utilizada foi 57, que foi o número de melhor valor nos testes de coerência. Cenários de execução:

- Sem limpeza de *stopwords* e sem remoção de verbos
- Com limpeza de *stopwords* e sem remoção de verbos
- Com limpeza de *stopwords* e com remoção de verbos

Para ilustrar os resultados eu coloquei aqui listas menores mas os resultados completos podem ser encontrados na pasta docs/relatorio/analisetopicos do repositório do projeto.

Por exemplo, ao ilustrar as palavras que mais contribuem para um determinado tópico, aqui estou exibindo apenas as 5 enquanto na pasta acima mencionada eu coloquei as 20 mais importantes.

5.3.1 Código fonte

O código fonte para análise de tópicos está listado abaixo:

```
1 from fonte_dados.fabrica import FabricaFonteDados
2 from treinamento.treinamento_lda import TreinamentoLda
3 from util import constants
4 
5 def treinar_modelo(fonte_origem, num_topics,
    limpar_stopwords_especificas=False, limpar_verbos=False):
```

```
    fabrica = FabricaFonteDados()

    # Carregar fonte de dados de origem
    fonte_dados_origem = fabrica.get_fonte_dados(fonte_origem)
    fonte_dados_origem.carregar_dados(limpar_stopwords_especificas=
    limpar_stopwords_especificas, limpar_verbos=limpar_verbos)
    documentos_origem = fonte_dados_origem.get_tokens()

    # Ajuste de modelo
    treinamento_lda = TreinamentoLda(num_topics=num_topics, passes
    =2)
    resultado_lda = treinamento_lda.ajustar_modelo(
    documentos_origem)
    lda = resultado_lda.modelo_lda
    return lda


# Fonte NERDS_VIAJANTES, 57 tópicos, sem limpeza de stopwords espec
    íficas, sem limpeza de verbos
lda_inicial = treinar_modelo(fonte_origem=constants.NERDS_VIAJANTES
    , num_topics=57)

# Fonte NERDS_VIAJANTES, 57 tópicos, com limpeza de stopwords espec
    íficas, sem limpeza de verbos
lda_limpeza_stopwords_especificas = treinar_modelo(fonte_origem=
    constants.NERDS_VIAJANTES, num_topics=57,
    limpar_stopwords_especificas=True)

# Fonte NERDS_VIAJANTES, 57 tópicos, com limpeza de stopwords espec
    íficas, com limpeza de verbos
lda_limpeza_verbos = treinar_modelo(fonte_origem=constants.
    NERDS_VIAJANTES, num_topics=57, limpar_stopwords_especificas=
    True, limpar_verbos=True)
```

Listing 2: Código fonte: Análise de tópicos

No repositório do projeto pode ser consultado o código fonte completo do Notebook utilizada para a análise dos tópicos.

5.3.2 Execução inicial após limpeza de textos básica

A primeira análise de tópicos foi feita excluindo remoções de *stopwords* específicas ou dos verbos da limpeza de conteúdo dos posts, ou seja, nesta análise estes conjuntos fizeram parte do dicionário completo. Os resultados aqui obtidos, inclusive, serviram de insumo para levantar a lista de palavras específicas a serem removidas, assim como da necessidade de remoção dos verbos, cuja lista completa foi obtida de site mencionado na seção de coleta.

Nos resultados abaixo, assim como nos cenários seguintes, para cada tópico é exibida um tupla onde o primeiro elemento é o id do tópico e o segunda a lista de 5 palavras que mais contribuíram na definição do tópico, assim como seus respectivos pesos.

[(51,

```
 '0.039*"fot" + 0.011*"fotograf" + 0.007*"pod" + 0.007*"
     ser" + 0.007*"revel"'),
(44,
 '0.026*"lago" + 0.010*"lenćol" + 0.010*"caminh" +
     0.010*"passei" + 0.009*"par"'),
(36, '0.022*"fot" + 0.008*"fic" + 0.008*"algum" +
    0.007*"tir" + 0.007*"par"'),
(3,
 '0.016*"fot" + 0.011*"fotograf" + 0.009*"trilh" +
     0.008*"parqu" + 0.007*"caminh"'),
(33,
 '0.014*"fot" + 0.014*"expos" + 0.012*"mov" + 0.010*"
     temp" + 0.009*"prać"'),
(53,
 '0.031*"fot" + 0.016*"igrej" + 0.014*"fotograf" +
     0.012*"expos" + 0.010*"abert"'),
(17,
 '0.011*"fot" + 0.010*"viag" + 0.007*"passei" + 0.007*"
     bem" + 0.007*"cidad"'),
(46,
 '0.019*"estr" + 0.012*"prai" + 0.012*"algum" + 0.010*"
     parqu" + 0.008*"par"'),
(11,
 '0.013*"restaurant" + 0.012*"lag" + 0.011*"fot" +
     0.010*"algum" + 0.009*"com"'),
(50,
 '0.022*"vulc" + 0.013*"puert" + 0.012*"jalap" + 0.011*"
     lag" + 0.010*"sal"')]
```

Na listagem acima podemos ver várias palavras que contribuíram muito para o levantamento de tópicos neste cenário de execução mas que julgamos não serem interessantes na comparação com documentos de outras fontes.

Um exemplo é a palavra **fot**, que se refere a foto. Além de falar sobre viagens, o blog sempre se propôs também a ter conteúdo de fotografia e esta palavra aparece em boa parte dos documentos. No entanto não julgamos ser interessante para o propósito de comparação.

5.3.3 Execução após remoção de palavras específicas

No cenário anterior identificamos um conjunto de palavras que estavam contribuindo muito na definição de tópicos mas que não estavam agregando na obtenção de documentos semelhantes de acordo com o objetivo inicial. Neste cenário os tópicos foram identificados após remover este conjunto de palavras específicas.

```
[(26,
```

```
  '0.010*"parqu" + 0.009*"glaci" + 0.008*"trilh" +
     0.008*"pont" + 0.007*"caminh"'),
 (4,
  '0.025*"estr" + 0.020*"parqu" + 0.017*"milh" + 0.011*"ô
     nibu" + 0.010*"anchorag"'),
 (29,
  '0.011*"lago" + 0.010*"esqu" + 0.008*"local" + 0.007*"
     mar" + 0.007*"nev"'),
 (10,
  '0.011*"torr" + 0.011*"caminh" + 0.010*"sub" + 0.009*"
     vist" + 0.008*"refúgi"'),
 (23,
  '0.026*"jalap" + 0.017*"toalh" + 0.012*"cool" + 0.010*"
     refresc" + 0.010*"towel"'),
 (44,
  '0.028*"urs" + 0.008*"catarat" + 0.007*"pesso" +
     0.007*"grand" + 0.006*"parqu"'),
 (2,
  '0.010*"tour" + 0.009*"bar" + 0.007*"botec" + 0.006*"
     hor" + 0.006*"gost"'),
 (20,
  '0.018*"cidad" + 0.008*"prédi" + 0.008*"local" +
     0.007*"gost" + 0.007*"prać"'),
 (7,
  '0.011*"mir" + 0.008*"centr" + 0.007*"café" + 0.006*"
     chap" + 0.006*"blog"'),
 (24,
  '0.012*"parqu" + 0.010*"pont" + 0.009*"pass" + 0.008*"
     park" + 0.008*"temp"')]
```

5.3.4 Execução após remoção de verbos

O último cenário foi a análise de tópicos após a remoção de verbos.

```
[(13,
  '0.017*"mund" + 0.014*"pra" + 0.012*"blog" + 0.012*"
     brasíl" + 0.011*"blogu"'),
 (17,
  '0.033*"lag" + 0.024*"parqu" + 0.012*"barc" + 0.010*"á
     gu" + 0.009*"bonit"'),
 (41,
  '0.015*"expos" + 0.015*"águ" + 0.015*"temp" + 0.014*"
     pouc" + 0.012*"próx"'),
 (0,
  '0.024*"urs" + 0.017*"iso" + 0.016*"câm" + 0.016*"boa"
     + 0.014*"parqu"'),
```

```
(25,
 '0.019*"parqu" + 0.012*"park" + 0.011*"ônibu" + 0.011*"
    temp" + 0.010*"vist"'),
(21,
 '0.022*"prać" + 0.015*"liberdad" + 0.012*"temp" +
    0.012*"cidad" + 0.012*"denal"'),
(32,
 '0.027*"cidad" + 0.021*"igrej" + 0.021*"histór" +
    0.018*"jalap" + 0.012*"são"'),
(48,
 '0.046*"parqu" + 0.023*"pret" + 0.020*"barc" + 0.019*"
    estad" + 0.016*"cacho"'),
(34,
 '0.017*"prat" + 0.011*"opć" + 0.011*"vist" + 0.011*"
    pais" + 0.008*"cardápi"'),
(3,
 '0.045*"prai" + 0.011*"bonit" + 0.011*"mari" + 0.010*"
    cidad" + 0.010*"igrej"')]
```

6 Semelhança entre documentos para textos desconhecidos

6.1 Cálculo de semelhança (Similarity query)

Após o ajuste de um modelo usando LDA, uma das coisas que podemos fazer com base no resultado é obter a distribuição de tópicos para um documento. Isto na prática significa obter a lista de tópicos que o algoritmo encontrou para aquele texto. Esta lista contém duas informações: o identificador do tópico e o quanto o algoritmo entende que o tópico contribui para aquele documento, medido no valor de probabilidade que o tópico esteja naquele documento.

Com base na distribuição de probabilidades de tópicos em dois documentos nós podemos calcular a semelhança entre eles. Para o cálculo da semelhança nós usamos uma métrica chamada **distância Jensen-Shannon**, que determina o quão próximos estatisticamente falando dois documentos estão, comparando a divergência entre a distribuição de tópicos entre eles. Quanto menor o valor da distância, maior a semelhança entre os dois documentos de acordo com a distribuição de probabilidade. Esta distância é simétrica, ou seja, a distância entre dois documentos A e B é a mesma de B e A, o que está de acordo com o propósito deste trabalho.

Para distribuições discretas P e Q, a **divergência Jensen-Shannon**, JSD, é definida como:

$$JSD(P||Q) = 1/2D(P||M) + 1/2D(Q||M)$$

onde $M = 1/2(P + Q)$

A raiz quadrada da **divergência Jensen-Shannon** é a **distância Jensen-Shannon**: $\sqrt{JSD(P||Q)}$

Quanto menor a **Jensen-Shannon distance** maior é a semelhança entre duas distribuições, ou seja, maior a semelhança entre dois documentos.

Para encontrar os documentos mais semelhantes a um novo texto nós calculamos as probabilidades dos tópicos do novo texto e calculamos a **distância Jensen-Shannon** deste para os textos aos quais queremos comparar (usando as probabilidades dos tópicos deles) e ordenamos pelas menores distâncias para obter os mais semelhantes.

6.2 Dicionário de dados na comparação

Para obter a distribuição de tópicos de um documento é preciso ter a distribuição de frequência das palavras daquele documento. A distribuição de frequência de palavras em um documento é calculada utilizando, em conjunto, o dicionário usado como base para o treinamento e a lista de palavras que compõem o documento.

Ao obter a distribuição de frequência de palavras em um novo texto (que não estava no conjunto utilizado no treinamento), a implementação do algoritmo LDA que escolhemos *gensim* apenas considera as palavras existentes no

dicionário original que foram usadas para treinar o modelo, descartando aquelas que não existiam. Estas últimas não serão consideradas na distribuição de tópicos. Isto é um problema na definição dos tópicos do novo texto mas importante para identificar semelhanças com os documentos originais.

Uma forma de mitigar o problema acima é tentar fazer com que os dados de treinamento sejam o mais abrangente possível. No caso deste trabalho eu considerei que isso não seria um grande problema, já que o objetivo foi encontrar textos na base destino que fossem semelhantes aos textos da base de origem.

7 Resultados

A análise dos resultados da recomendação de páginas da Wikipedia com base em posts do blog Nerds Viajantes foi um dos grandes desafios deste trabalho, senão o maior. O problema é que não há uma forma de verificar a qualidade das recomendações que não envolva subjetividade. A alternativa que tive foi fazer uma validação manual de parte do conteúdo, além de analisar estatística e graficamente parte dos resultados. Mas mesmo esta análise numérica não é suficiente para avaliar a eficiência do treinamento e o mecanismo de recomendação.

7.1 Plano de execução de testes

Para executar o levantamento de documentos semelhantes além de usar os posts do blog como fonte origem de dados de treinamento e as páginas da Wikipedia como fonte de dados de comparação eu resolvi variar os seguintes parâmetros, com a descrição dos valores utilizados.

- **topics**: 57, 45, 72, 67, 39, 22, 101
- **eta**: 0.005, 0.05, 0.5, 1
- **alpha**: 0.01, 0.1, 1
- **passes**: 2, 10

Combinando os 378 posts da fonte de origem com estes possíveis valores eu obtive 378 * 7 * 4 * 3 * 2 = **63504 resultados**. Este número foi efetivamente multiplicado por 2 porque fiz os testes uma vez para todas páginas da Wikipedia e outra execução apenas para páginas contendo apenas páginas que após a limpeza de dados básica tinha tamanho maior ou igual a 40 tokens. O resultado destas duas execuções foi colocado em *collections* separadas no MongoDB e neste relatório são usados os termos como **full** para referenciar o conjunto completo e **gte40** para o conjunto limitado pelo tamanho.

A execução dos testes foi feita através do módulo **Executor**, explicado na seção de *Apêndice*. O código para execução deste módulo é o seguinte:

```
from main.especificacao_testes import EspecificacaoTestes
from main.executor import ExecutorTreinamento
from repository.resultado import ResultadoRepo

# Cria especificaćão de testes
filename = 'especificao_completa.yml'
especificacao_testes = EspecificacaoTestes(filename)
cenarios = especificacao_testes.get_testes()

# Executa treinamento
executor = ExecutorTreinamento(cenarios)
resultado_df = executor.executar_treinamento()

# Grava resultados no MongoDB
documentos = resultado_df.to_dict('records')
```

```
resultado_repo = ResultadoRepo()
resultado_repo.insert_many(documentos)
```

Listing 3: Código fonte: Treinamento de modelo usando LDA

A lista de testes a serem executados foi determinadas em arquivo *yml*, formato escolhido pela facildade de definir variações de valores de forma declarativa. Abaixo está a listagem do arquivo utilizado (que está no repositório do projeto).

```
-
  fonte_origem: NV NERDS_VIAJANTES
  fonte_destino: WIKIPEDIA WIKIPEDIA
  topics:
    - 57
    - 45
    - 72
    - 67
    - 39
    - 22
    - 101
  eta:
    - 0.005
    - 0.05
    - 0.5
    - 1
  alpha:
    - 0.01
    - 0.1
    - 1
  passes:
    - 2
    - 10
```

Listing 4: Especificação de execução de testes

7.2 Variação do comportamento da semelhança entre documentos

Um dos meus objetivos foi analisar a variação do comportamento da semelhança entre os documentos de acordo com os valores dos parâmetros de treinamento. Conforme explicado anteriormente, a proximidade se dá pela da distância entre os documentos, que envolve um cálculo baseado nas probabilidades que os documentos tem de terem em seu contexto identificados cada tópico. Quanto menor a distância entre os documentos, mais próximos os mesmos e desta forma existe maior probabilidade de serem semelhantes.

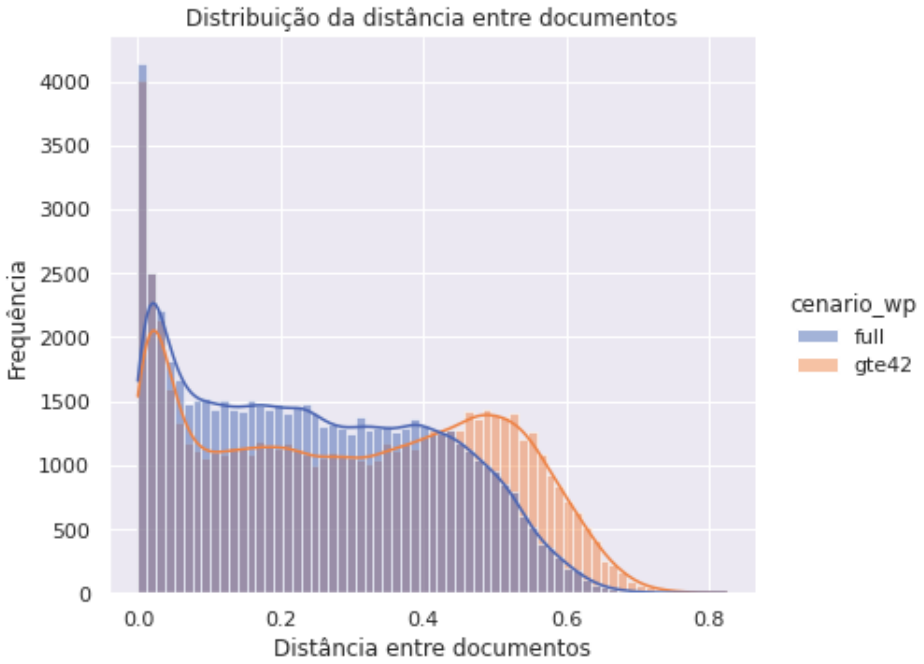
Com base nisto eu resolvi analisar o comportamento da média e desvio padrão das distâncias de acordo com os parâmetros de treinamento ou outro fator que pudesse influenciar na distância.

A tabela a seguir mostra que pelos resultados obtidos a média das distâncias entre documentos na fonte de origem utilizando o conjunto completo de páginas da Wikipedia foi menor que ao utilizar o conjunto de páginas limitado

pelo tamanho. Isto mostra que mostra que o algoritmo considerou os documentos semelhantes mais próximos dos utilizados no treinamento para o conjunto completo.

Conjunto Wikipedia	Média de distâncias	Desvio padrão
full	0.244593	0.169004
gte40	0.293772	0.196154

O histograma a seguir nos dá a mesma informação anterior, mas de forma gráfica. Podemos ver que para o conjunto completo há uma concentração maior nas menores distâncias enquanto que o conjunto limitado tem maior concentração nas distâncias maiores.

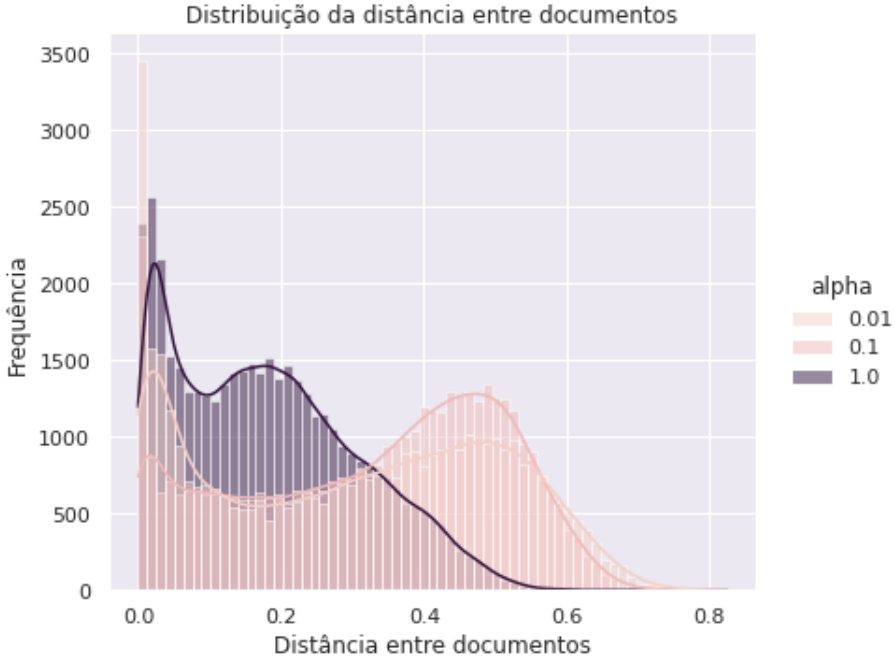


O parâmetro **alpha** indica o comportamento da quantidade de tópicos que influenciam um conjunto de textos. Um valor baixo significa que os documentos tem um pequeno número de tópicos entre eles, ao passo que um valor alto indica que os documentos tem um maior número de tópicos entre eles, ou seja, os documentos têm uma tendência a serem mais distintos em relação a outros.

Pelos valores abaixo percebemos que ao aumentar o valor de *alpha* diminuem o valor das distâncias, ou seja, quanto dizemos ao algoritmo que há mais tópicos dentro do conjunto de textos de treinamento, a tendência é que o algoritmo entenda que a proximidade com os documentos mais parecidos da fonte de comparação seja maior.

Alpha	Média de distâncias	Desvio padrão
0.01	0.299975	0.202525
0.10	0.327551	0.181500
1.00	0.180020	0.126922

O mesmo comportamento acima pode ser verificado graficamente. Com o valor de *alpha* igual a 1, por exemplo, há uma concentração muito maior na área de distâncias menores.

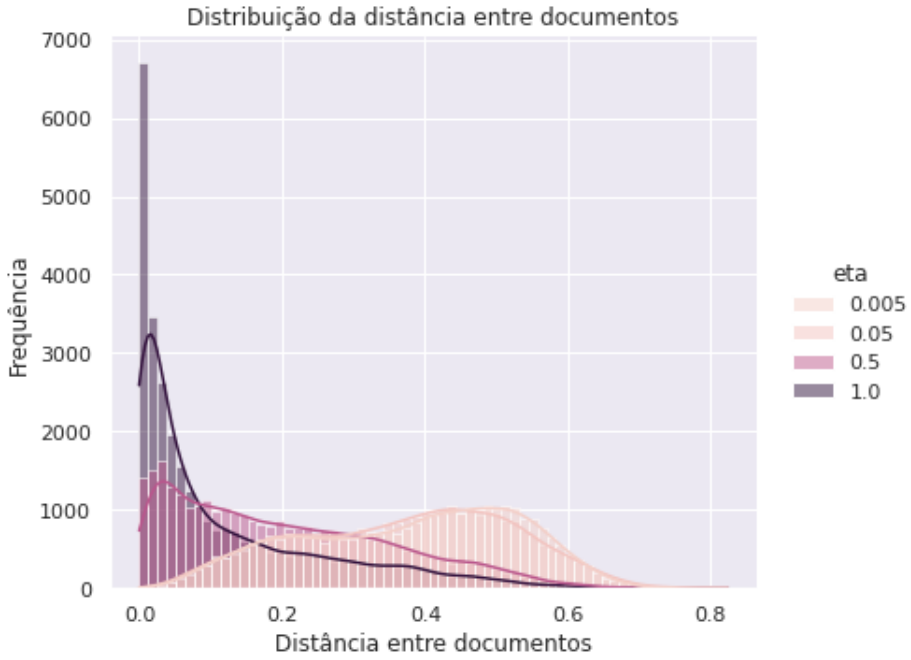


O parâmetro **eta** indica o comportamento da quantidade de palavras que influenciam na definição dos tópicos. Um valor baixo significa que poucas palavras influenciam a definição de tópicos e vice-versa.

Pelos valores abaixo percebemos que ao aumentar o valor de *eta* diminuem o valor das distâncias, ou seja, quanto dizemos ao algoritmo que há mais palavras atuando na definição de tópicos dentro do conjunto de textos de treinamento, a tendência é que o algoritmo entenda que a proximidade com os documentos mais parecidos da fonte de comparação seja maior.

Eta	Média de distâncias	Desvio padrão
0.005	0.379512	0.150678
0.050	0.371781	0.150373
0.500	0.205030	0.152114
1.000	0.120406	0.138177

O mesmo comportamento acima pode ser verificado graficamente. Com o valor de *eta* igual a 1, por exemplo, há uma concentração muito maior na área de distâncias menores.



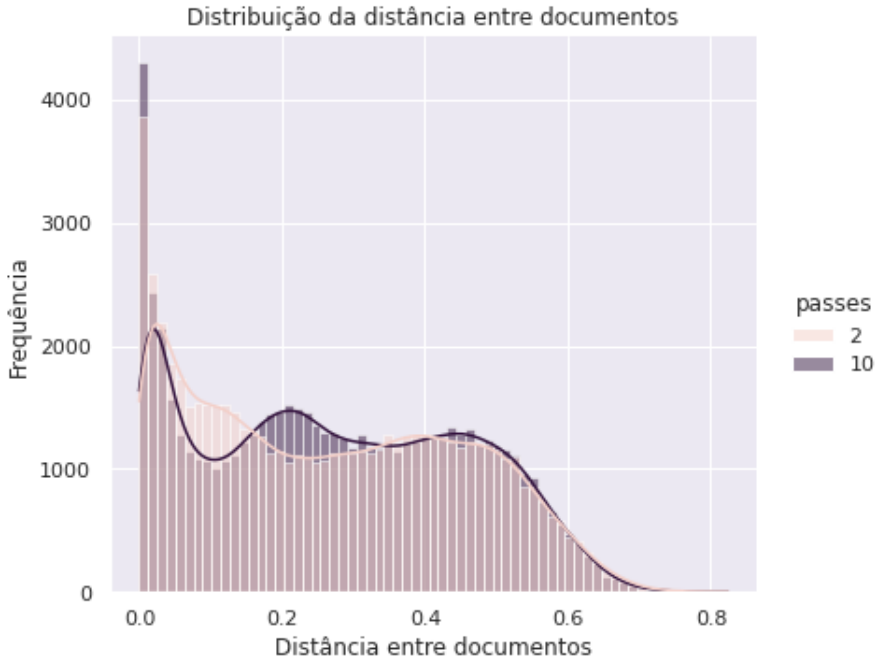
O número de tópicos a serem encontrados é outro parâmetro muito importante. Foi explicado anterior como o cálculo de coerência foi usado para definir a lista de números de tópicos que usamos em nossos testes. Abaixo segue uma tabela com o comportamento da variação das distância de acordo com o número de tópicos. Percebemos que com menos tópicos este indicador de semelhança aumenta, já que na média temos distâncias menores.

Número de tópicos	Média de distâncias	Desvio padrão
22	0.234775	0.161470
39	0.266610	0.173423
45	0.270563	0.178036
57	0.276523	0.184559
67	0.285299	0.187543
72	0.279334	0.191407
101	0.271173	0.208681

O parâmetro **passes** indica o número de vezes que o algoritmo passa pelo conjunto de textos ao fazer o treinamento. Os valores abaixo indicam que a diferença entre os dois valores utilizados (2 e 10) foi muito pequena, causando pouca influência na noção de proximidade entre os documentos.

Número de tópicos	Média de distâncias	Desvio padrão
2	0.265562	0.186465
10	0.272802	0.182900

O mesmo comportamento anterior pode ser verificado na distribuição abaixo.



7.3 Variabilidade de documentos recomendados

Outro fator interessante de analisar é a variabilidade de documentos recomendados ao variar os parâmetros de treinamento. No conjunto de resultados cada documento de origem tem **336 valores**, número este que vem da combinação de valores possíves das variações dos parâmetros número de tópicos, *alpha*, *eta*, *passes* e de 2 conjuntos de páginas diferentes.

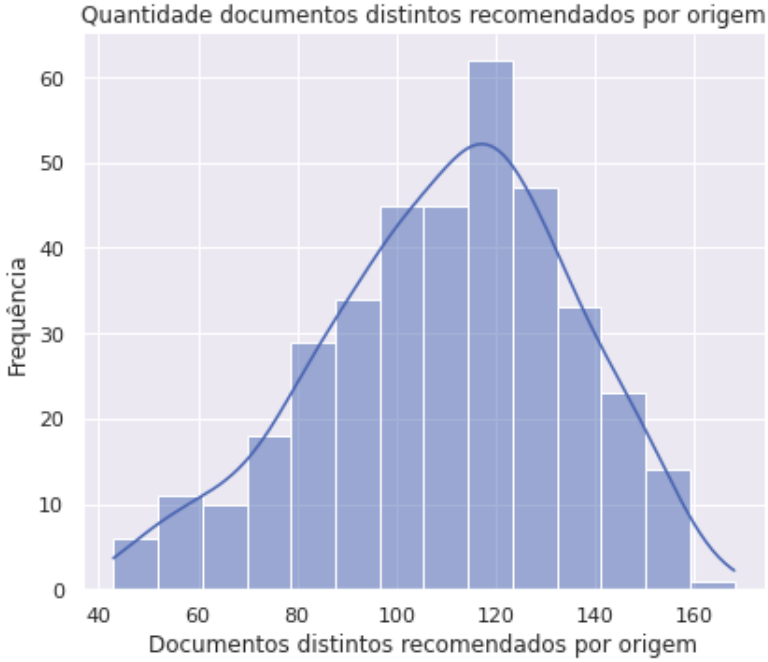
Agrupando por documento de origem eu contei a quantidade de documentos destinos diferentes que foram recomendados como o mais semelhante e abaixo está a lista ordenada contendo os 5 com menor variação e os 5 com maior.

No geral o documento com *id* 7923 teve 43 documentos de destino recomendados com o mais semelhante. Isto quer dizer que o post do blog com este *id* teve 43 páginas na Wikipedia recomendadas como a mais parecida entre as 336 comparações realizadas para este documento no total.

No outro extremo temos o documento de origem com *id* 6798, que teve 168 documentos distintos recomendados para este documento. Isto quer dizer que a variação nos parâmetros influenciou muito na identificação de semelhança para este documento.

Id do documento de origem	Documentos destino diferentes
7923	43
7131	45
8116	46
7108	48
8146	51
...	...
8896	157
5641	158
134	158
3721	158
6798	168

O gráfico a seguir mostra a distribuição da quantidade de documentos distintos por documento de origem. Notamos que a maior concentração de documentos semelhantes distintos recomendados por origem está perto de 120.



Vamos então ver como esta variabilidade é influenciada pelos parâmetros de treinamento do algoritmo para entender quais tem maior peso.

Começando pelo número de tópicos, vamos ver os 5 que mais se destacam nos dois extremos, os que têm maior e menor variação. Cada combinação de documento de origem e número de tópicos aparece 48 vezes no resultado (3 *alpha* * 4 *eta* * 2 *passes* * 2 *cenario_wp*).

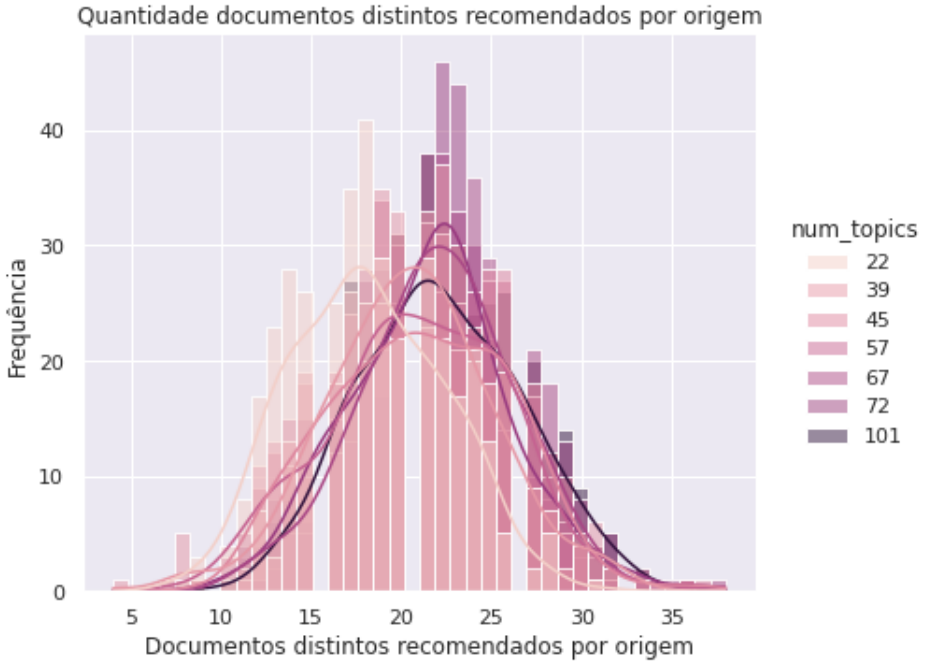
O documento de *id* 5599 se mostrou o que menos é influenciado pela variação dos demais parâmetros quando o número de tópicos foi *39*, tendo apenas 4

documentos distintos recomendados nos 48 resultados para esta combinação.

No outro extremo temos o documento com *id* 8896, que teve com 67 tópicos teve 38 recomendações distintas, uma variação muito grande.

Id do documento de origem	Número de tópicos	Documentos destino diferentes
5599	39	4
14565	22	7
5599	57	7
14565	39	8
13988	39	8
...	...	...
10475	72	36
8896	57	36
10399	101	36
14946	45	37
8896	67	38

Vamos ver agora a distribuição gráfica destes valores recomendados por tópico.



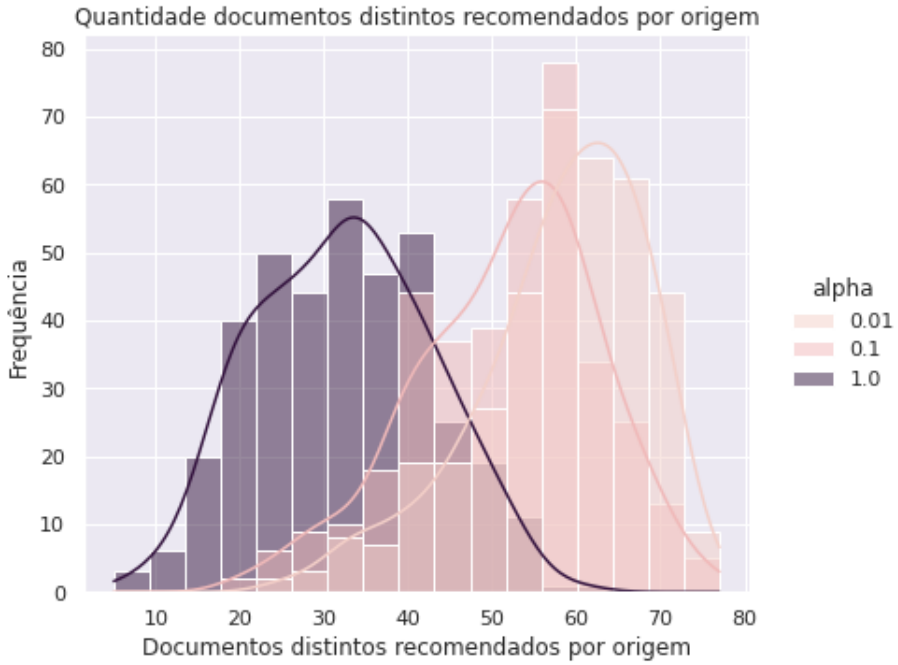
Em seguida vemos os 5 que mais se destacam nos dois extremos em relação ao parâmetro *alpha*, os que têm maior e menor variação. Cada combinação de documento de origem e valor de alpha aparece 112 vezes no resultado (7 *num_topics* * 4 *eta* * 2 *passes* * 2 *cenario_wp*).

Assim como na comparação por tópicos, o documento de *id* 5599 se mostrou o que menos é influenciado pela variação dos demais parâmetros quando o valor alpha *1.00*, tendo apenas 5 documentos distintos recomendados nos 112 resultados para esta combinação.

No outro extremo temos o documento com *id* 6798, que teve com valor *alpha* 0.10 teve 77 recomendações distintas.

Id do documento de origem	**Alpha**	**Documentos destino diferentes**
5599	1.00	5
13617	1.00	8
13360	1.00	9
13356	1.00	10
15389	1.00	11
...	...	...
8896	0.10	76
6637	0.10	76
14946	0.01	77
8896	0.01	77
6798	0.10	77

O gráfico abaixo nos mostra que ao utilizar valor *alpha* 1.0 a tendência é que a quantidade de documentos distintos recomendados por documento de origem seja muito menor, causando menor variabilidade. Lembrando que este valor mais alto indica pro algoritmo que mais tópicos definem os contextos dos textos. Para valores *alpha* 0.1 e 0.01 a variabilidade é maior mas com pouca diferença entre um e outro na distribuição.

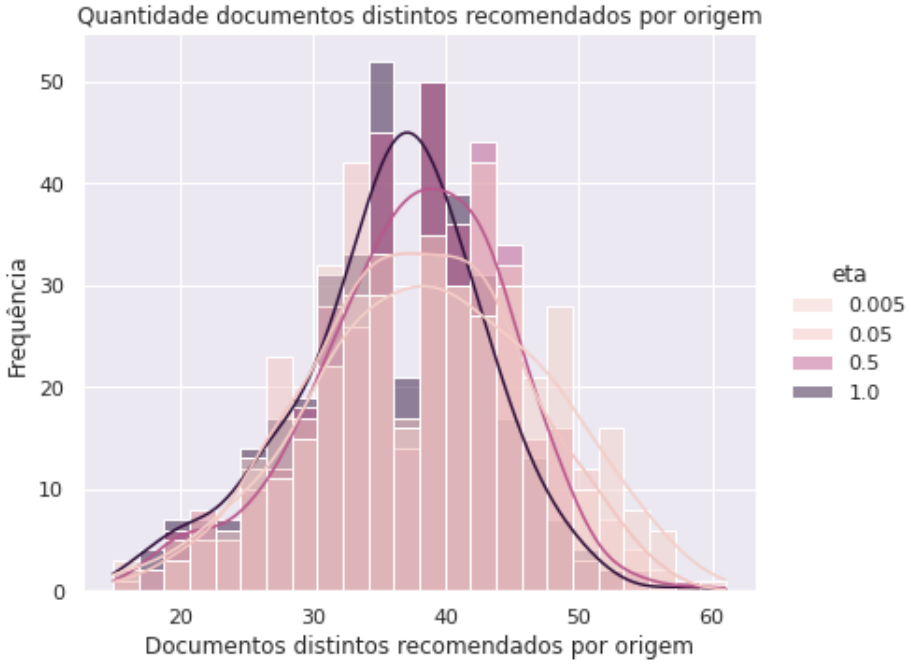


O próximo parâmetro analisado é o *eta*, onde também analisamos os 5 que mais se destacam nos dois extremos em relação ao parâmetro. Cada combinação de documento de origem e valor de *eta* aparece 84 vezes no resultado (7 *num_topics* * 3 *alpha* * 2 *passes* * 2 *cenario_wp*).

Aqui o documento com menor variação é o com *id* 3339, que para o valor *eta* 0.005 teve apenas 15 documentos distintos. Nota-se aqui que mesmo não sendo o que mais se destaca houve também pouca variação para o documento com *id* 5599. No outro extremo temos o documento com *id* 5641, que teve com valor *eta* 0.005 teve 61 recomendações distintas, o que é um número grande comparado com as 84 execuções para este par.

Id do documento de origem	Eta	Documentos destino diferentes
3339	0.005	15
15389	0.005	15
5599	0.050	15
7185	1.000	16
13870	0.050	16
...	...	...
8896	0.005	57
6798	0.005	58
8896	1.000	58
14946	0.500	59
5641	0.005	61

A distribuição abaixo nos mostra que a maior concentração de documentos distintos é semelhante para qualquer valor de *eta*, perto da faixa de 35 documentos. A diferença acontece na quantidade de valores nesta faixa, que aumenta quando se aumenta o *eta* e na faixa com maior quantidade de documentos distintos um valor mais baixo do parâmetro se destaca, com maior concentração).

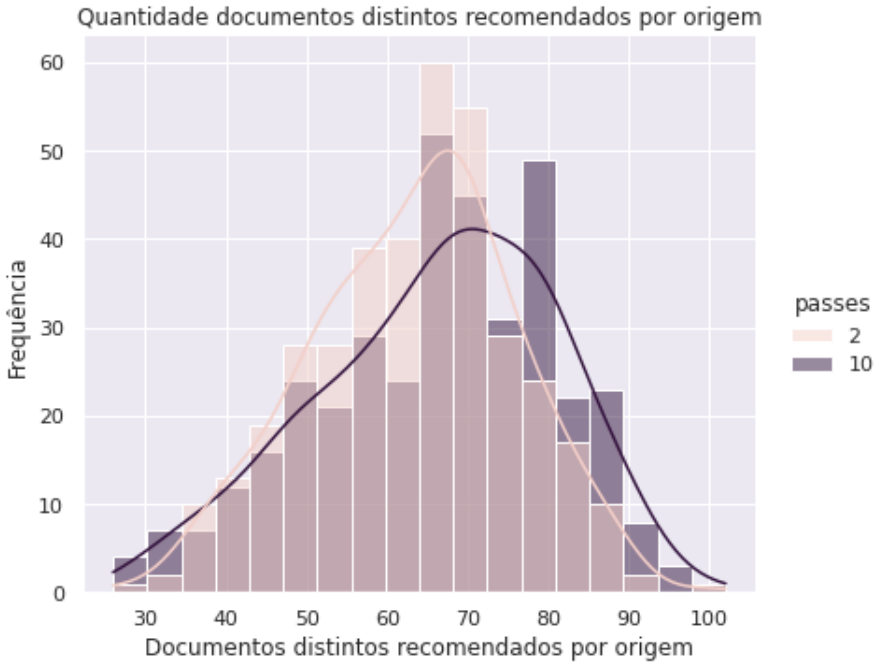


O próximo parâmetro analisado é o *passes*, onde abaixo analisamos os 5 que mais se destacam nos dois extremos em relação ao parâmetro. Cada combinação de documento de origem e valor de *eta* aparece 168 vezes no resultado (7 *num_topics* * 3 *alpha* * 4 *eta* * 2 *cenario_wp*).

Aqui o documento com menor variação novamente é o com *id* 5599, que para o valor de *passes* 2 teve apenas 26 documentos distintos. No outro extremo temos o documento com *id* 3326, que com valor de *passes* 10 teve 102 recomendações distintas.

Id do documento de origem	Passes	Documentos destino diferentes
5599	2	26
7923	27	
7131	10	27
7108	10	29
8116	10	30
...	...	...
6766	10	94
6798	10	95
8018	10	96
8896	2	100
3326	10	102

A análise gráfica abaixo nos mostra que apesar de ter uma diferença na concentração na faixa com maior frequência, a variação para este parâmetro não se mostrou muito significativa.



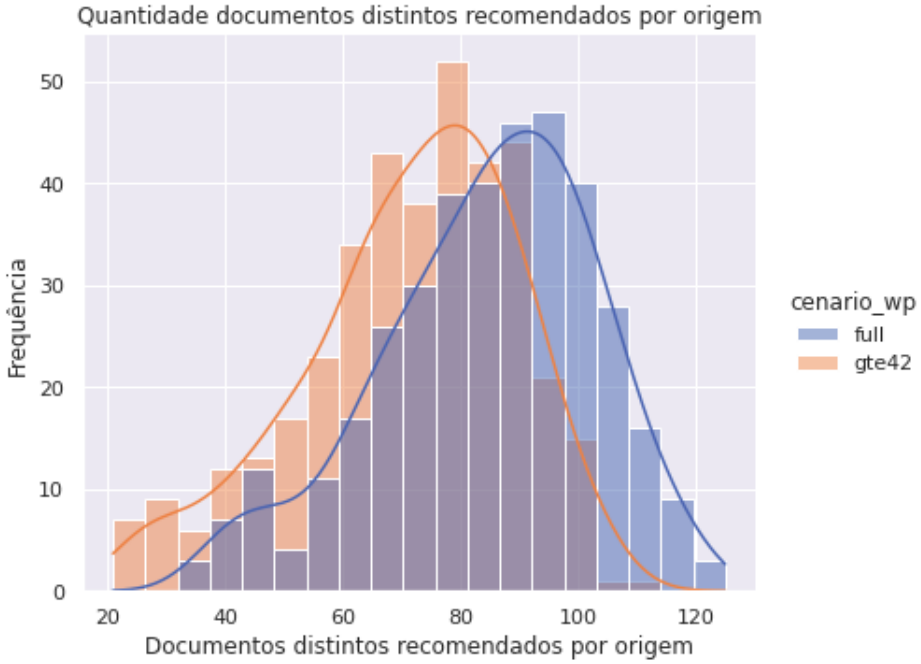
Em seguidas analisamos os extremos de variação ao limitar o tamanho dos documentos utilizados na comparação. Assim como no caso anterior, cada combinação de documento de origem e cenário Wikipedia aparece 168 vezes no resultado (7 *num_ topics* * 3 *alpha* * 4 *eta* * 2 *passes*).

Aqui o documento com menor variação novamente é o com *id* 8116, que para a base de comparação reduzida teve apenas 21 documentos distintos. No outro extremo temos o documento com *id* 6798, que na base de comparação completa 125 recomendações distintas.

Um fato interessante a se notar é que os 5 que menos variam ocorreram na base reduzida, enquanto os 5 maiores aparecem na base de comparação completa.

Id do documento de origem	cenario_wp	Documentos destino diferentes
8116	gte42	21
7131	gte42	21
7923	gte42	24
7344	gte42	24
7108	gte42	25
...	...	...
5232	full	118
5093	full	118
3326	full	120
8377	full	122
6798	full	125

A análise gráfica abaixo nos mostra que ao excluir da comparação documentos com pouca quantidade de tokens a variação na quantidade de documentos distintos recomendados por documento de origem dimimui. Por aqui nota-se o mesmo que percebemos na tabela acima, que é a predominância da base reduzida no extremos de menor variação e da base completa na área de maior variação. Isto mostra uma tendência do algoritmo em "ter mais dúvidas" quando usamos a base completa. Mas a diferença neste caso não ocorre por somente por causa do tamanho dos documentos, há também a diminuição nas opções disponíveis para comparação.



7.4 Tópicos e palavras dominantes

7.4.1 Tópicos dominantes e seu percentual de contribuição em cada documento

No algoritmo LDA, cada documento é composto por múltiplos tópicos, mas tipicamente apenas um é dominante. Uma análise que pode ser feita após o treinamento é ver qual o tópico dominante de cada documento, seu percentual de contribuição naquele documento e as palavras chave do tópico.

Número do documento	Tópico dominante	Percentual de contribuição	Palavras chave	Documento
0	19.0	0.9949	bonit, temp, cidad, vist, cervej, aind, pra, g...	[santiag, mus, art, pré, colombi, vár, mes, pe...
1	6.0	0.9947	congr, cidad, ambi, marin, noss, opç, cad, blo...	[santiag, bacan, dentr, opç, divers, blog, aqu...
2	8.0	0.9924	glaci, temp, pouc, lag, seward, vist, pais, ai...	[palaci, moned, soment, moned, chilen, santiag...
3	55.0	0.4061	câm, cidad, loj, estanh, cen, expos, são, peç,...	[hoj, seç, seman, câm, ent, últ, chil, argenti...
4	30.0	0.9891	parqu, ônibu, temp, montanh, expos, fiz, cent,...	[monument, valley, unid, dentr, indígen, trib,...
5	1.0	0.9958	sorvet, cidad, aind, histór, mund, uma, pouc, ...	[cristób, 880m, nível, 280m, nível, santiag, s...
6	54.0	0.8293	routeburn, track, hut, mus, pais, câm, porqu, ...	[frutill, cidad, bel, alemã, não, muse, alemán...
7	4.0	0.9936	argentin, prat, boa, bariloch, cord, cidad, ma...	[qu, país, boa, sempr, argentin, buen, air, de...
8	17.0	0.9967	lag, parqu, barc, águ, bonit, cacho, pouc, táb...	[lag, andin, petrohué, antecipad, lug, bellav,...
9	8.0	0.9113	glaci, temp, pouc, lag, seward, vist, pais, ai...	[catedr, metropolit, santiag, atr, religi, pop...

Em determinados momentos pode ser interessante ver os textos que melhor representam um tópico. A tabela a seguir mostra 10 tópicos e para cada tópico

mostra o maior percentual de contribuição daquele tópico, as palavras chaves que melhor o representam e parte do texto onde o tópico é mais significativo.

Tópico	% Contribuição	Palavras chave	Texto representativo
0.0	0.9943	urs, iso, boa, câm, parqu, cinderel, moment, voad, voo, cordilh	[selv, intenç, denal, nation, park, parqu, alasc, aind, parqu, animal, inclusiv, urs, espéci, gr...
1.0	0.9974	sorvet, cidad, aind, histór, mund, uma, pouc, pesso, são, temp	[boa, gulose, princip, sorvet, sempr, sorvet, bel, horizont, cidad, dentr, sorvet, aless, uma, i...
2.0	0.7644	alt, geral, caraç, cidad, santu, igrej, santiag, parqu, matriz, ônibu	[rapid, santiag, vall, estaçã, esqu, montanh, passe, princip, cidad, ônibu, hop, hop, off, final...
3.0	0.9978	prai, bonit, mari, cidad, igrej, pouc, praç, rua, vist, fiz	[noss, fern, noronh, tour, apes, bacan, poi, noronh, fim, agênc, caminhon, 4x4, depois, turist, ...
4.0	0.9936	argentin, prat, boa, bariloch, cord, cidad, mas, afinal, portugu, jauj	[qu, país, boa, sempr, argentin, buen, air, depois, bariloch, cidad, boa, bariloch, jauj, cardáp...
5.0	0.9974	lago, lençol, maranh, cervej, cidad, parqu, águ, aind, temp, chei	[lençol, maranh, antig, junh, 2012, promoç, aére, bom, são, luí, maranh, lençol, maranh, lago, c...
6.0	0.9947	congr, cidad, ambi, marin, noss, opç, cad, blog, sit, bel	[santiag, bacan, dentr, opç, divers, blog, aquí, blog, destemper, diss, mes, mes, varand, ótim, ...
7.0	0.9984	glaci, temp, quilômetr, pouc, lad, barc, chuv, lag, bonit, jalap	[sul, argentin, chil, contém, dezen, glaci, uma, boa, barc, lag, argentin, nest, glaci, upsal, s...
8.0	0.9978	glaci, temp, pouc, lag, seward, vist, pais, aind, montanh, bonit	[patagôn, calafat, dest, estânc, dentr, opç, cristin, apes, rebanh, ovelh, porqu, dentr, áre, pa...
9.0	0.9951	animal, temp, vist, selv, expos, macr, long, lad, lag, yellowston	[yellowston, nation, park, parqu, nacion, unid, selv, sempr, chanc, animal, talv, bich, princip,...

7.5 Análise de conteúdo das recomendações

Uma coisa difícil em aprendizado não supervisionado é a validação da qualidade dos resultados obtidos. Neste trabalho seria avaliar o quanto faz sentido as recomendações de páginas da Wikipedia para cada post do blog. Com base nos resultados anteriores eu resolvi analisar de forma geral e também pontualmente algumas recomendações.

A validação que faço aqui é muito subjetiva, com base no meu entendimento se o conteúdo recomendado é relevante ou não e os critérios podem variar de post para post). Alguns critérios podem ser semelhança geográfica, semalhança de assunto, tipo de localidade, e por aí vai.

Para dar um exemplo, um dos posts analisados abaixo é de uma visita ao Parque Nacional Denali, no Alasca. O que seria conteúdo relevante relacionado? Páginas falando do próprio Denali teriam relevância altíssima. Outras falando de outros parques nacionais (tipo de localidade) ou de outros lugares do Alasca (mesm estado) seriam também considerados relevantes por mim.

7.5.1 Vinte combinações com maior e menor frequência de recomendação

Uma das análises que quis fazer foi verificar quais as 20 combinações de post de origem com página recomendada apareceram mais e menos nos resultados.

Abaixo segue a tabela com as 20 combinações mais frequentes. A maioria das recomendações é muito relevante mas algumas são curiosas. Percebe-se uma tendência de semalhança entre posts do Valle Nevado e o Cerro Catedral, que é em outra região mas é conteúdo muito relevante para o post de origem. Outras semelhanças curiosas (mas estas não parecem fazer sentido) é associar posts da Praia do Rosa (em Santa Catarina) com a página da Wikipedia *Território organizado dos Estados Unidos*, os relacionados a um hotel de Punta Cana com a página *American Academy of Arts and Letters* e os dos Lençóis Maranhenses com a *Reserva da Biosfera do Cinturão Verde de São Paulo*. A maioria, no entanto, pelo próprio título dos textos em ambas as fontes faz todo sentido.

Título documento origem	Título documento destino	Recomendações
Vídeo - Episódio Valle Nevado Ski Resort	Cerro Catedral	159
Santiago - Cerro Santa Lucía	Cerro Santa Lucía	127
Alasca - Visitando o Denali National Park and Preserve	Parque Nacional e Reserva de Denali	119
Praia do Rosa - Trilha para o Mirante do Canto Sul	Território organizado dos Estados Unidos	106
Valle Nevado Ski Resort - Minha Experiência com o Esqui	Cerro Catedral	100
Punta Cana - Hard Rock Hotel & Casino Punta Cana	American Academy of Arts and Letters	100
Região dos Lagos Chilenos - Roteiro Volta ao Lago Llanquihue	Llanquihue	99
Revelando a Foto - Horseshoe Bend	Horseshoe Bend (Arizona)	96
Revelando a Foto - Zabriskie Point	Zabriskie Point	94
Revelando a Foto - Pescador na Praia Vermelha	Território organizado dos Estados Unidos	93
Alasca - Vida Selvagem no Denali National Park and Preserve	Parque Nacional e Reserva de Denali	93
El Calafate - Passeio de Barco por Todos Los Glaciares	Glaciar Upsala	92
Catas Altas - Um Passeio Pela Cidade e Região	Reserva Particular do Patrimônio Natural Santuário Caraça	90
Revelando a Foto - Especial de Natal	Castelo da Cinderela	89
Lençóis Maranhenses - Lagoa Verde	Reserva da Biosfera do Cinturão Verde de São Paulo	86
Lençóis Maranhenses - Post Índice	Reserva da Biosfera do Cinturão Verde de São Paulo	85
Hard Rock Hotel & Casino Punta Cana - Suíte Caribe	American Academy of Arts and Letters	84
Praia do Rosa - Trilha para a Praia Vermelha	Território organizado dos Estados Unidos	81
Frutillar - Um Pedaço da Alemanha no Chile	Frutillar	80
Vídeo - Episódio Hard Rock Hotel & Casino Punta Cana	American Academy of Arts and Letters	79

Abaixo segue uma tabela com as vinte combinações com menor frequência. Na verdade aqui estão apenas vinte mas há outras combinações que ocorreram apenas uma vez mas não justifica mostrar aqui. Por esta amostragem aqui dá pra ver que estas combinações aconteceram muito com posts sobre descritivos de fotografia (a série *Revelando a Foto* tinha como objetivo mostrar técnicas aplicadas a uma foto específica), que nada eram relacionadas com o tema de

natureza que foi o foco do conteúdo das páginas da Wikipedia selecionadas para comparação.

Título documento origem	**Título documento destino**	**Recomendações**
Catas Altas - Um Passeio Pela Cidade e Região	Parque Estadual de Grão Mogol	1
Revelando a Foto - Dia das Crianças	Lista de vencedoras do Miss USA	1
Revelando a Foto - Flores no Epcot	Lista de rios dos Estados Unidos	1
Revelando a Foto - Flores no Epcot	Rio Anatoki	1
Revelando a Foto - Dia das Crianças	25.º Distrito Congressional da Califórnia	1
Revelando a Foto - Flores no Epcot	Rio Allen	1
Revelando a Foto - Flores no Epcot	Rio Ahuriri	1
Revelando a Foto - Dia das Crianças	Hawke's Bay	1
Revelando a Foto - Flores no Epcot	Parque Nacional Vicente Pérez Rosales	1
Revelando a Foto - Dia das Crianças	Pico Pikes	1
Revelando a Foto - Dia das Crianças	Tanaga	1
Revelando a Foto - Flores no Epcot	Alicanto	1
Revelando a Foto - Flores no Epcot	Ilhas Bounty	1
Revelando a Foto - Dia das Crianças	Parque Estadual Delta do Jacuí	1
Revelando a Foto - Dia das Crianças	Rio Big (Costa Oeste)	1
Revelando a Foto - Flores no Epcot	Títulos e estatísticas da MLS	1
Revelando a Foto - Dia das Crianças	Cabo Farewell (Nova Zelândia)	1
Revelando a Foto - Flores no Epcot	Futaleufú	1
Revelando a Foto - Dia das Crianças	Estação Ecológica de Santa Bárbara	1
Revelando a Foto - Dia das Crianças	Antillanca (grupo vulcânico)	1

7.5.2 Análise de recomendações com foco nos posts de origem

Na seção de variabilidade de documentos recomendado vimos que o documento com *id* 7923 (*Jalapão - Alimentação*) foi o que teve o menor número de documentos distintos recomendados para ele em toda minha bateria de execuções, tendo 43 distintas páginas da Wikipedia recomendadas como a mais semelhante pra ele. Vamos então ver, dentre este conjunto, quais as páginas que tiveram maior número de recomendações.

Este post fala sobre alimentação em uma atração turística do estado do Tocantins, então sendo assim eu acho que das 5 páginas mais recomendadas para ele a primeira, quarta e quinta tem conteúdo relevante, enquanto a segunda e terceira nem tanto.

Título documento destino	**Recomendações**
Política Nacional de Resíduos Sólidos	62
Horseshoe Bend (Arizona)	32
Portal:Nova Zelândia/Colabore	32
Monumento Natural das Árvores Fossilizadas do Tocantins	26
Projeto Cerrado Verde	26

No outro extremo da quantidade de páginas distintas recomendados para um mesmo post temos o com *id* 6798 (*Revelando a Foto - Detalhes em Caeté*), com 168 recomendações distintas das 336 feitas para este post. Trata-se de um post atípico em nosso blog, que fala de como foi tirada um foto em uma passeio de um dia para uma cidade próxima a BH. Mesmo entre as 5 páginas mais recomendadas não há uma sequer que eu ache que tenha relevância para o assunto do post.

Título documento destino	**Recomendações**
Bible Belt	8
Praça Nauro Machado	7
Chonchi	6
Rakahanga	5
Reserva Biológica Poço das Antas	5

Nas análises anteriores, ao analisar o comportamento baseado em parâmetros, alguns posts se destacaram por ter menos variação entre os recomendados para eles. Um destes foi o post de *id* 5599 (*Alasca - Visitando o Denali National Park and Preserve*), que teve a menor variação de documentos recomendados ao analisar junto os parâmetros *num_topics* e *alpha*. Resolvi analisar então as páginas de destino que mais foram recomendadas para este post, ordenadas pelo quantidade de recomendações de cada página.

A tabela a seguir mostra que a página da Wikipedia *Parque Nacional e Reserva de Denali*, com *id* 1403174 foi recomendada 119 vezes para o post acima, o que foi satisfatório visto que a página descreve exatamente o parque nacional que é foco do post em questão.

A segunda página mais recomendada é sobre o Monte Denali, que é assunto discutido internamente no post, enquanto que a quinta mais recomendada é sobre uma estrada no Alasca que fica próxima ao parque e de certa forma é conteúdo semelhante.

A terceira e quarta paǵinas que mais se destacam são de parques nacionais americanos. Mesmo não sendo próximos ao Denali não deixa de ser conteúdo relevante.

Título documento destino	Recomendações
Parque Nacional e Reserva de Denali	119
Monte Denali	46
Parque Nacional de Yosemite	41
Parque Nacional de Yellowstone	17
Dalton Highway	8

Na analisar de variação de posts recomendados considerando o parâmetro *eta*, o post que mais se destacou foi o de *3339* (*Lençóis Maranhenses - Lagoa Verde*). Resolvi então analisar as 5 páginas da Wikipedia que mais eram recomendadas para ele, também para verificar a relevância.

A página da Wikipedia *Reserva da Biosfera do Cinturão Verde de São Paulo* se destaca, com 86 recomendações para este post. Curiosamente apesar do número alto e da semelhança encontrada no treinamento, eu acho que as páginas mais relevantes para este post seriam e quarta e a quinta, que se referem a locais relacionados natureza e do estado do Maranhão. A quinta página, inclusive, se trata do parque nacional onde se encontra a Laguna Verde e se mostra a recomendação mais relevante. Quando lidamos com aprendizado de máquina nem sempre o primeiro resultado pode ser o mais relevante para o que queremos.

Título documento destino	Recomendações
Reserva da Biosfera do Cinturão Verde de São Paulo	86
Reserva Lagoa São Paulo	35
Lagoa Dom Helvécio	20
Área de Proteção Ambiental das Reentrâncias Maranhenses	13
Parque Nacional dos Lençóis Maranhenses	9

Um outro post que achei interessante analisar foi o de *14565* (*Vídeo - Episódio Salar de Atacama*), que conforme vimos em análise anterior teve baixa variação de páginas recomendadas no cenário de testes com menor número de tópicos (22). Ao analisar as páginas que mais lhe foram recomendadas eu achei o resultado muito interessante pois dos 10 listados abaixo 7 são do Atacama, região do assunto do post. A primeira da lista não é do Atacama mas também fala de vulcão, assunto tratado no conteúdo de origem. Mesmo os dois últimos, que seriam os menos relevantes, ainda são interessantes para o assunto.

Título documento destino	Recomendações
Osorno (vulcão)	54
Salar de Atacama	51
Gêiseres de Tatio	37
Vulcão Peña Blanca	20
San Pedro de Atacama	20
Puna atacamenha	19
Lascar (vulcão)	11
Vale do Salinas	9
Cerro Catedral	6
Montanhas Rochosas Canadianas	6

7.5.3 Análise de recomendações com foco nas páginas recomendadas

Esta seção tem como objetivo fazer uma análise de resultados agrupando pelas páginas recomendadas.

A primeira análise foi verificar quais páginas são recomendadas para o maior número de posts distintos. A tabela abaixo mostra que a página *Horseshoe Bend (Arizona)* foi recomendada para 174 posts distintos.

Página da Wikipedia	**Posts distintos**
Horseshoe Bend (Arizona)	174
ISO 3166-2:NZ	149
Glaciar Taku	148
Lista de premiações especiais do Miss USA	140
Castelo da Cinderela	138
Monte Fitz Roy	130
Parque Nacional e Reserva de Denali	130
ISO 3166-2:CL	120
Glaciar Upsala	120
Lago Yellowstone	117
Praça Pedro II (São Luís)	116
Área de Estatística Combinada	100
Campo de gelo do sul da Patagónia	99
Acidente na mina de Pike River em 2010	98
Wikipédia:Lista de predefinições/Esboços/Geografia	98
Lago Ranco	94
Geographic Names Information System	94
Lowcountry	93
Deserto de Chihuahua	93
Rio Mendoza	92

Com base na tabela anterior eu resolvi analisar para que tipo de post a página *Horseshoe Bend (Arizona)* estava sendo muito recomendada.

Em primeiro lugar aparece o post *Revelando a Foto - Horseshoe Bend* com 96 recomendações, o que é muito bom pois se trata do mesmo lugar. O segundo post que mais aparece é o *Revelando a Foto - Lake Powell* com 49 recomendações e eu considero este outro bom resultado, já que ambos os posts são de lugares localizados na cidade de Page, no Arizona. Da terceira mais recomendada para baixo começa a ficar menos interessante mas é muito interessante notar que a página *Horseshoe Bend (Arizona)* da Wikipedia foi sucessivamente recomendada como semelhante aos posts do Parque Estadual do Jalapão.

Título documento origem	Recomendações
Revelando a Foto - Horseshoe Bend	96
Revelando a Foto - Lake Powell	49
Jalapão - Cânion Sussuapara	34
Jalapão - Transporte	33
Jalapão - Alimentação	32
Jalapão - Fervedouro da Glorinha	32
Jalapão - Cachoeira da Formiga	32
Jalapão - O que vem por aí	31
Jalapão - O que Levar?	31
Jalapão - Acampamento Korubo	31

Outro página que achei interessante analisar foi a do *Campo de Gelo Sul da Patagonia*, próxima um locais onde eu tenho uma grande quantidade de posts. Quase todos os posts da tabela são de lugares que têm geleiras (exceto o da *Cachoeira da Maça*), o que faz todo sentido para a recomendação de uma página sobre o Campo de Gelo. E quatro deles são especificamente da região do campo de gelo.

Título documento origem	Recomendações
Kenai Fjords National Park - Trilha para o Harding Icefield	21
El Calafate - Passeio de Barco por Todos Los Glaciares	9
Kenai Fjords National Park - Passeio de Barco no Alasca	9
Kenai Fjords National Park - Trilha do Exit Glacier	9
Cachoeira da Maçã, Bela Surpresa da Cabeça de Boi	8
Revelando a Foto - Barco no Lago Espejo Chico	8
Do Chile a Argentina sob Cinzas Vulcânicas	8
El Calafate - Tour ao Glaciar Perito Moreno	8
El Calafate - Estancia Cristina	7
Circuito W - Lago e Glaciar Grey	7

Uma outra página que apareceu bastante como recomendada e chamou minha atenção foi a do Fitz Roy, uma montanha na Argentina. Resolvi analisar e os 9 primeiros posts são da cidade onde está a montanha, o que faz todo sentido. Apenas o décimo posts, de uma antiga cervejaria de Belo Horizonte, não faz sentido a recomendação.

Título documento origem	Recomendações
Vídeo - Episódio Laguna Torre	77
Vídeo - Episódio Laguna de Los Tres	74
El Chalten - Trilha para a Loma del Pliegue Tumbado	68
Revelando a Foto - Laguna de Los Tres	64
El Chaltén - Volta à Laguna de los Tres	51
El Chaltén - Trilha para Laguna Torre	45
El Chaltén - Trilha para Laguna de los Tres	40
Revelando a Foto - Patagônia sem Filtro	38
El Chalten - Trilha para a Laguna del Diablo	36
Belo Horizonte - Visita ao Pátio Cervejeiro da Backer	35

7.5.4 Análise de semelhança entre documentos

Como já disse anteriormente, me chamou atenção a página do *Horseshoe Bend* da Wikipedia ser recomendada com tanta frequência. Relemebrando, os dois primeiros posts para os quais ela foi recomendada eram da cidade de Page (Arizona), o que faz todo sentido. Já os três seguintes eram sobre o Jalapão, um parque estadual localizado no Tocantins, região norte do Brasil.

Uma boa forma de tentar compreender este tipo de semalhança é listar os tópicos que mais contribuem para um texto, assim como as palavras que mais contribuem para aqueles tópicos. Estas informações são retornadas através da API que faz o treinamento do modelo LDA do módulo *gensim*, que utilizamos neste projeto.

Abaixo listamos os tópicos que mais contribuem para estes 5 primeiros posts e também para a página em questão. Nota-se na listagem abaixo o quanto o tópico 16 é dominante em quatro destes posts e o 29 em outro. Para o texto da página da Wikipedia o post 16 também é dominante, por mais que ele tenha outros tópicos também significatios.

```
Topicos dominantes dos documentos de origem:
['16 = 0.9358', '25 = 0.0576']
['16 = 0.9957']
['29 = 0.6963', '16 = 0.2232', '56 = 0.0739']
['16 = 0.9953']
['16 = 0.9952']

Topicos dominantes dos documentos de destino:
['16 = 0.4912', '29 = 0.1189', '19 = 0.1015', '20 =
    0.0756', '52 = 0.0586']
```

Vendo que os tópicos 16 e 29 se destacam muito, resolvi listar as palavras que mais contribuem para a formação destes tópicos. Percebe-se pela listagem abaixo que estes tópico fazem todo sentido para os posts do Jalapão e deve ser dominante para o Horseshow Bend por causa de alguma palavra nele contida.

```
Palavras dominantes no tópico 16:
```

```
['jalap = 0.1216', 'cacho = 0.0302', 'águ = 0.0170', '
    korub = 0.0168', 'velh = 0.0149', 'pra = 0.0140', '
    temp = 0.0133', 'lei = 0.0125', 'expos = 0.0118', 'dun
     = 0.0109']

Palavras dominantes no tópico 29:
['vall = 0.0314', 'esqu = 0.0204', 'cidad = 0.0146', 'tr
    = 0.0141', 'lagun = 0.0123', 'aul = 0.0103', 'fiz =
    0.0098', 'ski = 0.0092', 'pouc = 0.0089', 'santiag =
    0.0088']
```

Outra associação frequente que achei curiosa foi de uma reserva perto das cidade de São Paulo com os posts dos Lençois Maranhenses. Para o post *Lençois Maranhenses - Laguna Verde*, por exemplo, as cinco páginas a seguir foram as mais recomendadas, seguidas pela frequência:

- Reserva da Biosfera do Cinturão Verde de São Paulo - 86
- Reserva Lagoa São Paulo - 35
- Lagoa Dom Helvécio - 20
- Área de Proteção Ambiental das Reentrâncias Maranhenses - 13
- Parque Nacional dos Lençóis Maranhenses - 9

Uma coisa curiosa que dá pra notar pela listagem abaixo é que o tópico dominante no post (23) não é o principal entre as páginas mas aparece quase sempre os primeiros.

```
Topicos dominantes dos documentos de origem:
['23 = 0.9969']

Topicos dominantes dos documentos de destino:
['5 = 0.1445', '48 = 0.1265', '1 = 0.1034', '56 =
    0.0746', '47 = 0.0629']
['1 = 0.3182', '23 = 0.3118', '3 = 0.2009', '47 =
    0.1232', '7 = 0.0327']
['1 = 0.1431', '23 = 0.1185', '48 = 0.1097', '5 =
    0.1003', '32 = 0.0935']
['48 = 0.2415', '5 = 0.1617', '25 = 0.0722', '17 =
    0.0668', '23 = 0.0577']
['48 = 0.3327', '23 = 0.1629', '7 = 0.1074', '32 =
    0.0849', '30 = 0.0756']
```

Analisando as palavras que mais contribuem para o tópico dominante acima citado nós podemos perceber várias palavras que fazem sentido no post dos Lençois Maranhenses mas que também estão nos contextos das páginas mais recomendadas (note as palavras *verd*, *lago*, *águ*, etc). Neste caso ficou mais

claro o motivo da semelhança entre posts e páginas que a princípio não teriam muita ligação.

```
Palavras dominantes no tópico 29:
['lago = 0.0629', 'verd = 0.0171', 'águ = 0.0166', 'prat
    = 0.0150', 'pouc = 0.0146', 'lenćol = 0.0141', 'gent =
     0.0127', 'barre = 0.0126', 'cidad = 0.0113', 'maranh
    = 0.0112']
```

7.6 Datasets de resultados

Foram disponibilizados no repositório do projeto três *datasets* com resultados das recomendações.

O arquivo resultados_dataset.csv contém o dataset completo. Tem a página recomendada para cada post na fonte de dados de origem e cada parâmetro utilizado.

Contém os campos: **id_cenario, fonte_origem, id_documento_origem, titulo_documento_origem, fonte_destino, id_documento_destino, titulo_documento_destino, distancia_destino, num_topics, passes, eta, alpha, cenario_wp**.

O arquivo top5_posts_por_recomendado.csv contém um dataset reduzido contendo os cinco posts da base de origem para os quais uma página da Wikipedia foi recomendada como semelhante com mais frequência.

Contém os campos: **titulo_documento_origem, titulo_documento_destino, Recomendacoes**.

Já o arquivo top5_recomendacoes_por_post.csv contém um dataset reduzido contendo as cinco páginas da Wikipedia que mais foram recomendadas para cada post da base de origem como semelhante.

Contém os campos: **titulo_documento_origem, titulo_documento_destino, Recomendacoes**.

8 Links

Aqui estão links para recursos que exibem o trabalho realizado.

- Vídeo Youtube: https://www.youtube.com/watch?v=apd9m_XjA6E
- Repositório Git: https://github.com/heldergr/tcc-pucmg-2

9 Referências

9.1 Artigos

- 6 Tips for Interpretable Topic Models: https://towardsdatascience.com/6-tips-to-optimize-an-nlp-topic-model-for-interpretability-20742f3047e2
- Collocations in NLP using NLTK Library: https://towardsdatascience.com/collocations-in-nlp-using-nltk-library-2541002998db
- Evaluate Topic Models: Latent Dirichlet Allocation (LDA): https://towardsdatascience.com/evaluate-topic-model-in-python-latent-dirichlet-allocation-lda-
- Introduction to NLP: https://towardsdatascience.com/introduction-to-nlp-part-1-preprocessi
- Intuitive Guide to Latent Dirichlet Allocation: https://towardsdatascience.com/light-on-math-machine-learning-intuitive-guide-to-latent-dirichlet-allocation-437c
- LDA and Document Similarity: https://www.kaggle.com/ktattan/lda-and-document-similarity
- Removing stopwords using nltk: https://stackabuse.com/removing-stop-words-from-strings-in-p
- Stemming in Portuguese using nltk: http://www.nltk.org/howto/portuguese_en.html
- Text Cleaning Methods for Natural Language Processing: https://towardsdatascience.com/text-cleaning-methods-for-natural-language-processing-f2fc1796e8c7
- Text Cleaning in Natural Language Processing(NLP): https://medium.com/analytics-vidhya/text-cleaning-in-natural-language-processing-nlp-bea2c27035a6
- Text Summarization for Topic modeling and clustering: https://towardsdatascience.com/nlp-preprocessing-clinical-data-to-find-sections-461fdadbec77
- Topic Modeling using Latent Dirichlet Allocation (LDA): https://towardsdatascience.com/nlp-topic-modeling-to-identify-clusters-ca207244d04f
- Topic Modeling with LSA, PLSA, LDA & lda2vec: https://medium.com/nanonets/topic-modeling-with-lsa-psla-lda-and-lda2vec-555ff65b0b05
- Topic Modeling Visualization: https://www.machinelearningplus.com/nlp/topic-modeling-visualization-how-to-present-results-lda-models/
- Unsupervised topic model in Python (gensim): https://towardsdatascience.com/introduction-to-nlp-part-5b-unsupervised-topic-model-in-python-ab04c186f295

9.2 Ferramentas e tecnologias

- Docker: https://docs.docker.com/get-started/
- Gensim LDA Model: https://radimrehurek.com/gensim/models/ldamodel.html
- Latex tutorial: https://www.overleaf.com/learn/latex/Sections_and_chapters
- MongoDB: https://docs.mongodb.com/guides
- Processamento de Linguagem Natural: https://en.wikipedia.org/wiki/Natural_language_processing
- Python
  - bs4: https://www.crummy.com/software/BeautifulSoup/bs4/doc/
  - matplotlib: https://matplotlib.org/
  - numpy: https://numpy.org/
  - pandas: https://pandas.pydata.org/
  - pymongo: https://pymongo.readthedocs.io/en/stable/
  - regex: https://docs.python.org/3/library/re.html
  - seaborn: http://seaborn.pydata.org/tutorial/aesthetics.html
  - venv: https://docs.python.org/3/tutorial/venv.html
  - wordcloud: https://amueller.github.io/word_cloud/
- API Wikipedia: https://pt.wikipedia.org/wiki/Wikip%C3%A9dia:Central_de_pesquisas/Portal_de_dados/API

10 Apêndice

10.1 Código fonte

O código fonte do projeto está localizado na pasta src/python/notebooks do repositório git projeto (referência também na seção de links). Foram escritos módulos específicas para diferentes tipo de tarefa. Abaixo estão listados os módulos e suas respectivas responsabilidades:

- *coleta*: Coleta dos dados das fontes utilizadas no treinamento
- *explore*: Exploração estatística e gráfica dos documentos das fontes de dados para entendimento e processamento das informações
- *fonte_dados*: Implementações das fontes de dados, responsáveis por fazer tratamentos específicos e entregar estes dados da forma como o algoritmo projetado espera
- *limpeza*: Limpeza de documentos completos e partes de documentos cujos conteúdos sejam irrelevantes e atrapalham o processamento ou resultado
- *main*: Executor principal, que carrega as fontes de dados, ajusta o modelo e calcula os documentos semelhantes
- *praticas*: Práticas para efeito de aprendizado e exploração de tecnologia, executadas durante a execução do trabalho
- *repositorio*: Camada de acesso a dados persistidos localmente
- *similarity*: Cálculo de similaridades entre os documentos
- *treinamento*: Ajuste de modelo de treinamento e levantamento de tópicos para os documentos
- *util*: Código com funções de propósito geral

Além dos módulos para execução de tarefas específicas, na raiz da pasta de código fonte estão notebooks e scripts Python que comandam a execução de tarefas:

- *analise-palavras-fontes-dados.ipynb*: Análise exploratória das palavras das fontes de dados
- *calculo_coerencias.py*: Execução do cálculo de coerência por número de tópicos
- *lda-gensim-executor.ipynb*: Execução de treinamento e cálculo de documentos semelhantes com base em especificação de testes
- *nerds-viajantes-analise-coerencia_topicos.ipynb*: Análise estatística e gráfica dos dados de coerência gerados pela execução do script *calculo_coerencias.py*

- *nerds-viajantes-lda-analise-topicos.ipynb*: Análise de tópicos e palavras que mais contribuem no modelo gerado pelo treinamento
- *nerds-viajantes-visualizacao-resultados.ipynb*: Visualização gráfica e estatística dos resultados gerados na execução do treinamento e cálculo de semalhantes
- *wikipedia_ download.ipynb*: Orquestração e download de categorias e páginas da Wikipedia

10.2 Algoritmo e estruturas de dados

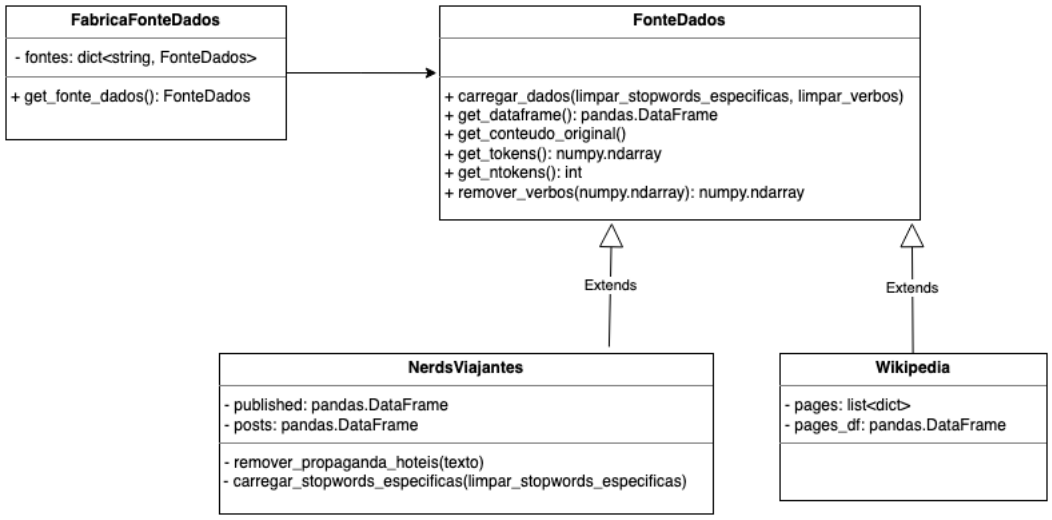
Nesta seção eu descrevo as partes que eu julgo serem as mais importantes do projeto.

10.2.1 Fonte de dados

Durante a implementação do projeto, um objetivo foi manter a rotina principal (treinamento, cálculo de probabilidade dos tópicos e semelhança entre documentos) independente dos dados de treinamento e também dos dados de teste. O motivo disto é tirar o acoplamento da estrutura dos dados, do conteúdo e da forma como é feita a coleta.

Para isto foi proposta uma estrutura de dados onde foi definido o conceito de fonte de dados, que faz com que qualquer origem de dados escolhida pudesse ser trabalhada sem mudança no código fonte do algoritmo de recomendação.

Uma fonte de dados é uma abstração que tem como funcionalidade carregar os documentos de algum lugar e disponibilizar para o algoritmo de treinamento. Para ficar genérico foi criada a classe *FonteDados* contendo o contrato de uma fonte, ou seja, quais métodos uma classe concreta precisa ter para fornecer os dados que o algoritmo precisa para o treinamento e também para montar o resultado com a semelhança entre os textos.



Cada fonte de dados específica deve carregar seus próprios dados e manter uma estrutura interna de forma a fornecer os documentos quebrados por tokens (palavras), assim como fornecer também um DataFrame contendo os seguintes dados:

- *id_documento*: id do documento na fonte de dados
- *nome*: nome do documento na fonte de dados
- *titulo*: titulo descritivo do documento na fonte de dados
- *documento*: conteudo original do documento
- *n_characters*: tamanho do documento em caracteres
- *tokens*: tokens do documento apos limpeza
- *n_tokens*: tamanho do documento em tokens

Desta forma as fontes de dados fornecem os dados de forma semelhante e mesmo trabalhando com mídias diferentes o algoritmo de recomendação não precisa ser alterado. Para trabalhar com uma nova fonte de dados basta criar uma subclasse de *FonteDados* e implementar a parte específica que carrega os dados, faz limpeza e retorna os dados necessários no treinamento.

```
1 from repository.verbos import VerbosRepo
2
3 class FonteDados:
4
5     def carregar_dados(self, limpar_stopwords_especificas=True,
      limpar_verbos=True):
6         pass
7
```

```
8       def get_dataframe(self):
9           return pd.DataFrame({'tokens': [], 'n_tokens': []})
10
11      def get_tokens(self):
12          return self.get_dataframe()['tokens']
13
14      def get_ntokens(self):
15          return self.get_dataframe()['n_tokens']
16
17      """
18      Remove verbos da lista de tokens
19      """
20      def remover_verbos(self, tokens):
21          verbos_repo = VerbosRepo()
22          todos_verbos = verbos_repo.find_all_stemmed()
23          tokens_sem_verbos = [token for token in tokens if token not
         in todos_verbos]
24          return tokens_sem_verbos
```

Listing 5: Código fonte: Abstração de Fonte de Dados

10.2.2 Treinamento LDA

O módulo de treinamento LDA é responsável por ajustar um modelo para determinação de tópicos para os documentos de um conjunto de textos, assim como as palavras que mais contribuem para a formação de cada tópico. Recebe como parâmetros os documentos, o número de tópicos a serem escolhidos, além dos parâmetros do LDA passes, alpha e eta já explicados na seção de treinamento.

O treinamento consiste nos seguintes passos:

1. Criar um dicionário com o conjunto total de palavras de todos os documentos. Este dicionário contém um mapeamento de um id interno para cada palavra
2. Com base no dicionário, cria um *corpus*, que é um mapeamento de frequência de ocorrência de cada palavra no conjunto inteiro de textos
3. Faz o treinamento do modelo usando a classe *LdaModel* do pacote *gensim*
4. Retorna como resultado do treinamento uma tupla com três elementos:
    - Dicionário de palavras
    - *corpus*
    - Resultado do modelo treinado

Segue o fragmento de código que faz este trabalho de treinamento:

```
1       def ajustar_modelo(self, documentos, alpha=1e-2, eta=0.5e-2):
2           # Cria dicionario
3           id2word = dicionario.criar(documentos)
4
```

```
        # Cria corpus com base nos documentos e no mapeamento de id para palavra
        # Para cada documento faz a contagem de vezes que um determinado topico
        # aparece no documento
        corpus = [id2word.doc2bow(documento) for documento in documentos]

        print(f'Ajustando modelo com {self.num_topics} topicos e {self.passes} passes')

        # Treina modelo LDA
        lda = LdaModel(corpus=corpus, id2word=id2word,
                num_topics=self.num_topics, passes=self.passes,
                alpha=alpha, eta=eta,
                random_state=0)

        return ResultadoTreinamentoLda(id2word, corpus, lda)
```

Listing 6: Código fonte: Treinamento de modelo usando LDA

Para referência, um link do código fonte do módulo de treinamento.

10.2.3 Cálculo de semelhança

O módulo *similarity* é responsável por calcular os documentos mais parecidos utilizando a **distância de Jensen-Shannon**. Ele é carregado com uma lista de probabilidades dos documentos que são as opções às quais eu quero procurar os mais parecidos a um documento que passo como parâmetro no método *get_most_similar_documents()*.

Exemplificando, neste trabalho eu queria obter as páginas das Wikipedia que mais se parecessem com os posts do blog. Sendo assim eu carreguei a classe *SimilarityCalculator* com as probabilidades dos tópicos para cada página da Wikipedia e para cada post do blog eu chamei o método *get_most_similar_documents()* para obter as páginas que mais se parecessem com este post.

```
from scipy.spatial import distance
import numpy as np

class SimilarityCalculator:

    def __init__(self, matriz_esparsa_alvo) -> None:
        self.__matriz_esparsa_alvo = matriz_esparsa_alvo

    def get_matriz_esparsa_alvo(self):
        return self.__matriz_esparsa_alvo

    """
    Params: probabilidades_documento: Probabilides do documento para o qual se deseja calcular a distancia
    """
    def jensen_shannon2(self, probabilidades_documento):
```

```
16          sim = [distance.jensenshannon(data,
    probabilidades_documento) for data in self.
    __matriz_esparsa_alvo]
17          return np.array(sim)
18
19      """
20      This function implements the Jensen-Shannon distance above
21      and retruns the top k indices of the smallest jensen shannon
        distances
22      between one document and all document probabilities in the
        sparse matrix
23      """
24      def get_most_similar_documents(self, probabilidades_documento,
        k=10, return_distances=False):
25          # Calcula lista de distancias jensen shannon
26          sims = self.jensen_shannon2(probabilidades_documento)
27
28          # Ordena para obter os documentos mais proximos de acordo
        com a distancia Jensen Shannon
29          most_similar = sims.argsort()[:k]
30
31          if return_distances:
32              return (most_similar, sims[most_similar])
33          else:
34              return most_similar
```

Listing 7: Código fonte: Cálculo de posts semelhantes

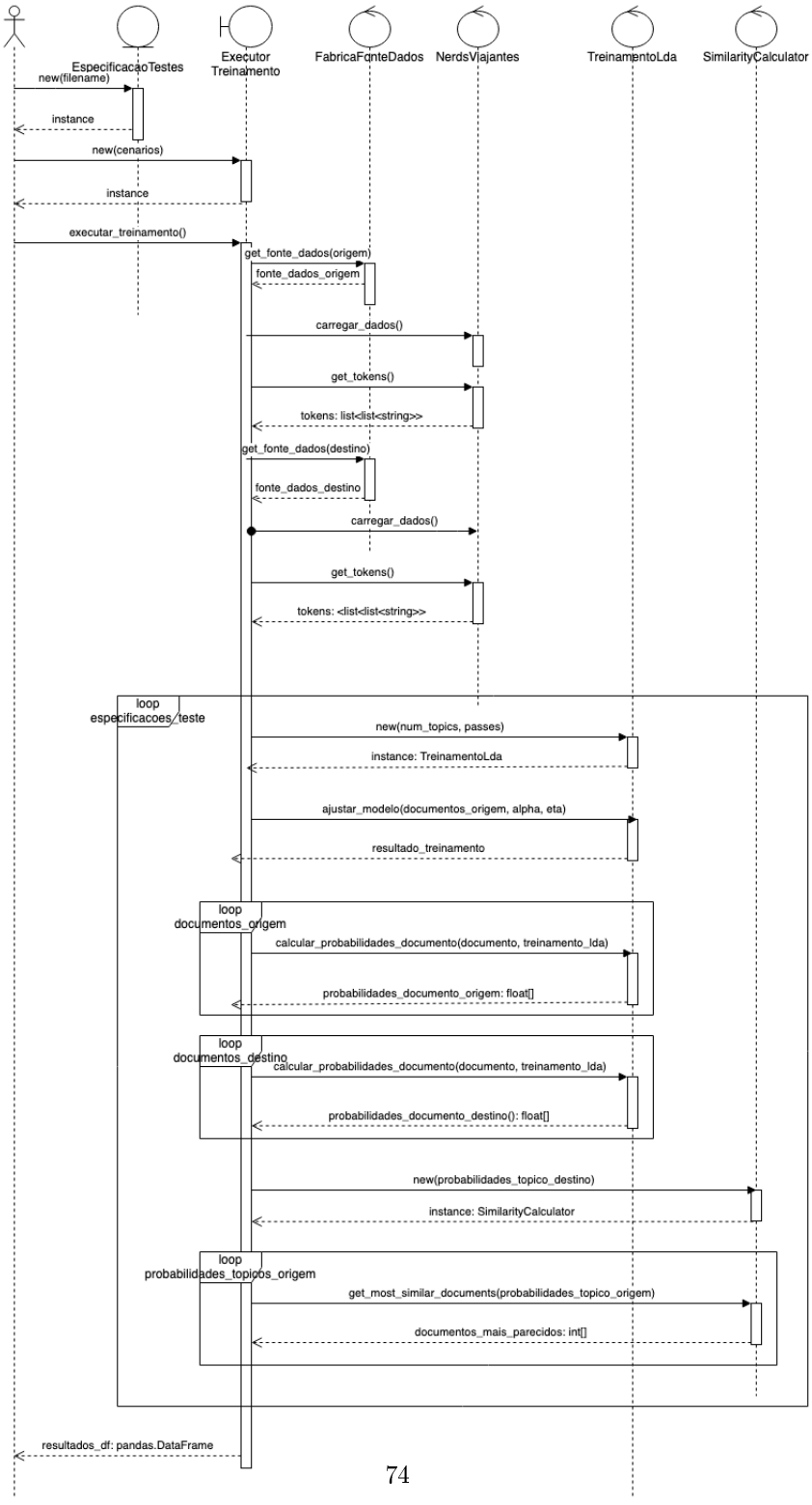
10.2.4 Executor

Foi criado um módulo executor para que pudessem ser variados os parâmetros de ajustes do modelo e gerar resultados para cada conjunto gerado por esta variação, permitindo uma comparação posterior dos resultados obtidos. Este módulo recebe como parâmetro uma lista de cenários de testes. Cada cenário contém as informações:

- fonte de dados de origem
- fonte de dados de destino
- topics
- alpha
- eta
- passes

Para cada combinação destes valores é feito o ajuste de modelo para os documentos de origem e o cálculo do documento de destino mais semelhante de cada documento de origem com base nas probabilidades que o documento contenha cada tópico.

O diagrama de sequência abaixo ilustra as operações que são realizadas e em qual ordem.


EspecificacaoTestes
Executor Treinamento
FabricaFonteDados
NerdsViajantes
TreinamentoLda
SimilarityCalculator
new(filename)
instance
new(cenarios)
instance
executar_treinamento()
get_fonte_dados(origem)
fonte_dados_origem
carregar_dados()
get_tokens()
tokens: list<list<string>>
get_fonte_dados(destino)
fonte_dados_destino
carregar_dados()
get_tokens()
tokens: <list<list<string>>
loop
especificacoes_teste
new(num_topics, passes)
instance: TreinamentoLda
ajustar_modelo(documentos_origem, alpha, eta)
resultado_treinamento
loop
documentos_origem
calcular_probabilidades_documento(documento, treinamento_lda)
probabilidades_documento_origem: float[]
loop
documentos_destino
calcular_probabilidades_documento(documento, treinamento_lda)
probabilidades_documento_destino(): float[]
new(probabilidades_topico_destino)
instance: SimilarityCalculator
loop
probabilidades_topicos_origem
get_most_similar_documents(probabilidades_topico_origem)
documentos_mais_parecidos: int[]
resultados_df: pandas.DataFrame

O resultado gerado pela execução do treinamento é um DataFrame com as seguintes colunas:

- *id_cenario*: identificador do cenário de testes
- *fonte_origem*: descrição da fonte de origem
- *id_documento_origem*: identificador do documento na fonte de origem
- *titulo_documento_origem*: título do documento na fonte de origem
- *fonte_destino*: descrição da fonte de destino
- *id_documento_destino*: identificador do documento na fonte de destino que tem a menor distância para o da fonte de origem referenciado na coluna id_documento_origem
- *titulo_documento_destino*: título do documento na fonte de destino
- *distancia_destino*: distância calculada entre o documento na fonte de origem e o docmento na fonte de destino
- *num_topics*: número de tópicos utilizado no modelo ajustado
- *passes*: parâmetro passes do algoritmo LDA
- *eta*: parâmetro eta do algoritmo LDA
- *alpha*: parâmetro alpha do algoritmo LDA

```
from fonte_dados.fabrica import FabricaFonteDados
from similarity.similarity import SimilarityCalculator
from treinamento.treinamento_lda import TreinamentoLda

import itertools
import pandas as pd

fabrica = FabricaFonteDados()

class ExecutorTreinamento():

    def __init__(self, cenarios_testes):
        self.__cenarios_testes = cenarios_testes

    def __obter_especificacoes(self, cenario):
        topics = cenario['topics']
        etas = cenario['eta']
        alphas = cenario['alpha']
        passess = cenario['passes']
        return itertools.product(topics, etas, alphas, passess)

    def executar_treinamento(self):
        dfs = []

```

```
25          for cenario in self.__cenarios_testes:
26              fonte_origem = cenario['fonte_origem']
27              print(f'Carregando dados de origem {fonte_origem}...')
28              fonte_dados_origem = fabrica.get_fonte_dados(
    fonte_origem)
29              fonte_dados_origem.carregar_dados()
30              documentos_origem = fonte_dados_origem.get_tokens()
31
32              fonte_destino = cenario['fonte_destino']
33              print(f'Carregando dados de destino {fonte_destino}...'
    )
34              fonte_dados_destino = fabrica.get_fonte_dados(
    fonte_destino)
35              fonte_dados_destino.carregar_dados()
36              documentos_destino = fonte_dados_destino.get_tokens()
37
38              for especificacao in self.__obter_especificacoes(
    cenario):
39                  num_topics, eta, alpha, passes = especificacao
40
41                  id_cenario = f'{fonte_origem}-{fonte_destino}-{
    num_topics}_topics'
42                  print('
    ------------------------------------------------')
43                  print(f'Executando testes para cenario {id_cenario
    }, eta={eta}, alpha={alpha}, passes={passes}')
44
45                  # Ajusta modelo LDA para os documentos de origem. O
     resultado gerado contem o dicionario, o corpus de dados e o
    modelo gerado
46                  print("Ajustando modelo para dados de origem...")
47                  treinamento_lda = TreinamentoLda(num_topics=
    num_topics, passes=passes)
48                  resultado_lda = treinamento_lda.ajustar_modelo(
    documentos_origem, alpha=alpha, eta=eta)
49
50                  print("Calculando probabilidades para topicos de
    origem...")
51                  probabilidades_topicos_origem = [treinamento_lda.
    calcular_probabilidades_documento(dnv, resultado_lda) for dnv
    in documentos_origem]
52
53                  print("Calculando probabilidades para topicos de
    destino...")
54                  probabilidades_topicos_destino = [
55                      treinamento_lda.
    calcular_probabilidades_documento(dw, resultado_lda) for dw in
    documentos_destino]
56
57                  print("Executando calculo de similaridade entre
    documentos...")
58                  similarity_calculator = SimilarityCalculator(
    probabilidades_topicos_destino)
59
60                  print("Calculando documentos mais parecidos...")
61                  destinos_mais_parecidos = [similarity_calculator.
    get_most_similar_documents(pto, return_distances=True) for pto
```

```
    in probabilidades_topicos_origem]
62
63                  destino_mais_parecido = [dmp[0][0] for dmp in
    destinos_mais_parecidos]
64                  distancia_mais_parecido = [dmp[1][0] for dmp in
    destinos_mais_parecidos]
65
66                  print("Montando resultado...")
67                  destino_df = fonte_dados_destino.get_dataframe()
68
69                  titulos_destino = destino_df['titulo'].values
70                  titulos_mais_parecidos = [titulos_destino[dmp] for
    dmp in destino_mais_parecido]
71
72                  ids_documentos_destino = destino_df['id_documento'
    ].values
73                  ids_documentos_mais_parecidos = [
    ids_documentos_destino[dmp] for dmp in destino_mais_parecido]
74
75                  resultado_df = pd.DataFrame(data = {
76                      'id_cenario': id_cenario,
77                      'fonte_origem': fonte_origem,
78                      'id_documento_origem': fonte_dados_origem.
    get_dataframe()['id_documento'].values,
79                      'titulo_documento_origem': fonte_dados_origem.
    get_dataframe()['titulo'].values,
80                      'fonte_destino': fonte_destino,
81                      'id_documento_destino':
    ids_documentos_mais_parecidos,
82                      'titulo_documento_destino':
    titulos_mais_parecidos,
83                      'distancia_destino': distancia_mais_parecido,
84                      'num_topics': num_topics,
85                      'passes': passes,
86                      'eta': eta,
87                      'alpha': alpha
88                  })
89                  dfs.append(resultado_df)
90
91          print('-----------------------------------------------------'
    )
92          print('Fim de execucao de testes.')
93          return pd.concat(dfs)
```

Listing 8: Código fonte: Executor

10.3 Tecnologias

- **Docker**: execução de ferramentas em containers com base em imagens das mesmas. Muito útil para evitar a complexidade de instalar softwares localmente. Utilizada para execução de:
  - MongoDB
  - Mysql

- **Git**: Versionamento do código fonte
- **Latex**: escrita de documentação e geração de relatório PDF
- **Python**: linguagem de programação utilizada para escrever todo o código deste trabalho, sendo escolhida a versão 3.8. Os principais pacotes que utilizei estão listados abaixo (a lista completa pode ser encontrada no arquivo requirements.txt no repositório do projeto):
  - bs4
  - gensim
  - matplotlib
  - mysql
  - nltk
  - numpy
  - pandas
  - pymongo
  - seaborn
  - wordcloud
- **Visual Studio Code**: IDE utilizada para escrever o código fonte