



IPBeja

INSTITUTO POLITÉCNICO
DE BEJA

Sistema de apoio à promoção do turismo rural
Fase de Investigação

Gonçalo Amaro – 17440,
Pedro Tomás – 18962,
Vítor Hugo – 18966

11 de Novembro, 2021

Conteúdo

1	Sites pesquisados	3
2	(Im)possibilidade de uso de APIs	3
3	Alternativas	3
3.1	<i>Web-crawling</i> VS <i>web-scraping</i>	3
3.1.1	Vantagens de <i>web scraping</i>	3
3.1.2	Desvantagens de <i>web scraping</i>	4
3.2	Necessidade de <i>Web Scraping</i>	4
4	Bibliotecas de Python para <i>Web Scraping</i>	4
4.1	Tratamento de output	4
4.1.1	Informações necessárias (previsão)	5
4.1.2	Informações a descartar (previsão)	5
5	Algoritmos de análise/mineração de texto	5
5.1	Algoritmos Clássicos	6
5.2	Algoritmos de <i>machine learning</i>	6
5.2.1	Aprendizagem supervisionada	6
5.2.2	Aprendizagem não supervisionada	6
5.2.3	Aprendizagem de reforço	6
5.3	Decisão sobre o tipo de algoritmo	7
5.3.1	Algoritmos considerados	7
6	Webgrafia	8

1 Sites pesquisados

Foram usados os sites:

- TripAdvisor
- Booking
- Zomato
- Yelp

2 (Im)possibilidade de uso de APIs

Dos quais em nenhum foi observada uma facilidade na obtenção de acesso às suas APIs, e apenas alguns (3/4) oferecem acesso à documentação da(s) mesma(s) facilmente. A maioria requer um contacto, que foi tentado e continua sem resposta até hoje (contactos iniciados por G.A.17440, entre dia 26 e 29 de Out, e hoje sendo dia 1 de Nov).

Os que oferecem documentação foi observado que todos subdividem os seus serviços de API em 3 ou 4 APIs para casos de uso específicos (reservas, dados, etc) em vez de uma com endpoints que oferecem solução para todos os casos.

3 Alternativas

Sendo o caso que a impossibilidade do uso das APIs (que facilitariam o trabalho) temos de recorrer a outras técnicas para obter os dados.

3.1 Web-crawling VS web-scraping

Web crawling, também conhecido como Indexação, é usado para indexar as informações na página usando bots também conhecidos como *tracker*. O *trackers* é essencialmente o que os motores de busca fazem. É uma questão de visualizar uma página como um todo e indexá-la. Quando um Bot rastreia um site, ele passa por todas as páginas e todos os links, até a última linha do site, em busca de qualquer informação.

O *web scraping*, também conhecido como extração de dados da *web*, é semelhante ao *web crawling*, pois identifica e localiza os dados de destino das páginas da *web*. A principal diferença é que, com o *web scraping*, sabemos o identificador de conjunto de dados exato, por exemplo, uma estrutura de elemento HTML para páginas da *web* que estão sendo corrigidas, da qual os dados precisam ser extraídos.

Com isto visto, *web scraping* é o nosso alvo, visto que minimiza lixo e é direcionado. No entanto devemos olhar para:

3.1.1 Vantagens de web scraping

1. Mais rápido: Manusear grandes quantidades de dados poderiam levar dias ou semanas a serem processados através do trabalho manual, com o uso do scraping podemos reduzir substancialmente o esforço e aumentar a velocidade de decisão.
2. Confiável e consistente: Ao fazer o trabalho manual é muito fácil de haver erros, por exemplo, erros tipográficos, informações esquecidas ou inserção nas colunas erradas. O uso do *web scraping* garante consistência e a qualidade dos dados.
3. Ajuda a reduzir a carga de trabalho.
4. Menor custo: Uma vez implementado o *scraping*, o custo total da extração de dados é significativamente reduzido, especialmente quando comparado ao trabalho manual.
5. Manutenção básica: Fazer o *scraping* de dados geralmente não requer muita manutenção.

3.1.2 Desvantagens de *web scraping*

1. Baixa proteção: se os dados na *web* são protegidos, o uso do *scraping* também pode se tornar um desafio e aumentar os custos.
2. Dados estruturados: não vai ser possível fazer *scraping* a 1000 sites diferentes pois cada site tem uma estrutura completamente diferente. Será necessário haver alguma estrutura básica que difira em determinadas situações.

3.2 Necessidade de *Web Scraping*

Com o acima dito, é necessário recorrer a soluções de *Web Scraping*. A comunidade internauta reparou no mesmo, visto que em reação ao observado existem dezenas de projetos e tutoriais de *Web Scraping* das variadas plataformas de turismo e reservas. Infelizmente, as mesmas não ajudam no processo e cada uma têm um forma de atuação bastante diferente.

4 Bibliotecas de Python para *Web Scraping*

Para fazer *Web Scraping* vamos usar Python, pela sua facilidade de uso e multifaceta “*Development speed is more important than execution speed*”. Com Python também temos as opções de criar cadernos Jupyter onde o próprio código e os resultados são “encadernados” com parágrafos de texto fazendo o próprio projeto o seu pequeno relatório de progresso e resultados; como também a criação de ambientes virtuais (*containers*), onde os pacotes usados ficam registados e instalados localmente, garantindo assim a portabilidade.

Para tal linguagem existem 5 grandes bibliotecas para a resolução deste caso:

- Requests
- BeautifulSoup
- Scrapy
- lxml
- Selenium

Cada uma tendo vantagens e desvantagens diferentes, um pequeno exemplo é o Selenium (a mais conhecida), que é a biblioteca mais completa e poderosa das listadas, mas tem um nível de complexidade maior e requer um setup inicial maior e mais trabalhoso, requer *WebDrivers* para a execução das tarefas, pode ser “manhosa” com Firefox, sendo preferencial usar *Chromium-based Browsers* como o Chrome e o novo Edge (necessário ainda o *ChromeDriver* e o *EdgeDriver*), visto que esta ferramenta não é apenas de *web scraping* mas sim de automação de testes.

Tentaremos evitar essa pela sua complexidade e extras desnecessários às nossas necessidades e custo temporal do setup inicial. Caso seja necessário usar, não vamos hesitar. Para cada website pode ser necessário usar bibliotecas diferentes por necessidade ou por obtenção de informações/blog posts/etc. . . que facilitem ou melhorem o output desejado.

4.1 Tratamento de output

Os outputs do conteúdo *scraped* podem vir em .xml ou .csv (ou outras mas essencialmente essas duas). Tentaremos ao máximo usar .csv, e transformar qualquer .xml em .csv, visto que um maior numero de ferramentas gráficas (Excel, PowerBI, etc..) e/ou bibliotecas de Python (Pandas, matplotlib, etc) para análise de dados tratam melhor ficheiros separados por virgulas.

Será feita também uma análise e extração de keywords nos textos das descrições e *reviews*. Para tál existem variados algoritmos que podemos usar, alguns “clássicos” outros até de *machine learning*.

Inicialmente todas as informações (dados e meta-dados) são dados como relevantes, após consideração e ponderação durante análises iniciais do decorrer do estudo poderemos descartar dados que não consideremos relevantes. No entanto nada nos impede de tentar prever ou imaginar quais esses serão e posteriormente avaliar o nosso julgamento para ver o que foi aprendido.

4.1.1 Informações necessárias (previsão)

Qualquer dado diretamente relacionado com o estabelecimento como horas de funcionamento, localização, preços, classificação e quantidade de *feedback*. Também a relação das *keywords* apanhadas do *feedback* em relação ao oferecido.

4.1.2 Informações a descartar (previsão)

Provavelmente informações pessoais de clientes e empregados não nos serão minimamente importantes.

5 Algoritmos de análise/mineração de texto

Os algoritmos de análise de texto podem ser considerados ferramentas de mineração de texto, isto é, o processo de descoberta de conhecimento potencialmente útil e inicialmente desconhecido, ou seja, a extração de conhecimento útil utilizando bases textuais.

O processo de mineração de texto é dividido em quatro etapas bem definidas:

- Seleção
- Pré-processamento
- Mineração
- Assimilação

Na seleção, os documentos relevantes devem ser escolhidos e mais tarde processados. No pré-processamento ocorrerá a conversão dos documentos em uma estrutura compatível com minerador, bem como ocorrerá um tratamento especial do texto. Na mineração, o minerador irá detetar os padrões com base no algoritmo escolhido. E por fim, na assimilação, os utilizadores irão utilizar o conhecimento gerado para apoiar as suas decisões.

A etapa pré-processamento pode ser dividida em quatro tarefas:

- Remoção de *StopWords*
- Conflação
- Normalização de sinónimos
- Indexação

Na etapa remoção de *stopwords* os termos com pouca ou nenhuma relevância para o documento serão removidos. São palavras auxiliares ou conectivas, ou seja, não são discriminantes para o conteúdo do documento.

Na etapa seguinte, conflação, realiza-se uma normalização morfológica, ou seja, realiza-se uma combinação das palavras que são variantes morfológicas em uma única forma de representação. Um dos procedimentos mais conhecidos de conflação é a radicalização (Stemming). Nela as palavras são reduzidas ao seu radical, ou seja, as palavras variantes morfológicamente serão combinadas em uma única representação, o radical. A radicalização pode ser efetuada com o auxílio de algoritmos de radicalização, sendo os mais utilizados o algoritmo de Porter (Porter Stemming Algorithm) e algoritmo de Orengo (Stemmer Portuguese ou RLSP).

Após a conflação, na etapa de normalização de sinónimos, os termos que possuem significados similares serão agrupados em um único termo, por exemplo, as palavras ruído, tumulto e barulho serão substituídas ou representadas pelo termo barulho. Na normalização de sinónimos, é formado um vocabulário controlado que se refere à utilização de termos adequados para representar um documento, sendo esses termos pré-definidos e específicos a um determinado assunto de uma área. Isso facilita a procura, pois os termos são normalmente utilizados pelos utilizadores.

E, por fim, na etapa indexação atribui-se uma pontuação para cada termo, garantindo uma única instância do termo no documento. No processo de atribuição de pesos devem ser considerados dois pontos:

- Quanto mais vezes um termo aparece no documento, mais relevante ele é para o documento
- Quanto mais vezes um termo aparece na coleção de documentos, menos importante ele é para diferenciar os documentos

5.1 Algoritmos Clássicos

Os algoritmos clássicos são instruções passo a passo de modo a que, dada uma entrada específica, é possível rastrear e determinar exatamente a saída.

- Algoritmos clássicos especificam as regras exactas para encontrar a resposta geral
- Um algoritmo clássico usa código e dados para prever a resposta correta para uma pergunta
- Algoritmo clássico produz uma saída com base nas etapas descritas no algoritmo
- Em algoritmos clássicos, normalmente é necessário um grande número de exemplos para determinar a que distância a equação está da equação desejada. É por isso que “*big data*” é uma grande negócio hoje em dia
- Um algoritmo clássico não dá uma solução depois de chegar a uma solução optima para um problema

5.2 Algoritmos de *machine learning*

Os algoritmos de *machine learning* são partes de código que ajudam as pessoas a explorar, analisar e localizar o significado em conjuntos de dados complexos. Cada algoritmo é um conjunto finito de instruções passo a passo inequívocas que um computador pode seguir para atingir um determinado objetivo. Num modelo de *machine learning*, o objetivo é estabelecer ou descobrir padrões que as pessoas possam utilizar para fazer previsões ou categorizar informações.

Os algoritmos de *machine learning* utilizam parâmetros baseados em dados de preparação, um subconjunto de dados que representa o conjunto maior. À medida que os dados de preparação se expandem para representar o mundo de forma mais realista, o algoritmo calcula resultados mais precisos.

Algoritmos diferentes analisam os dados de diversas formas. Geralmente, são agrupados consoante as técnicas de *machine learning* para as quais são utilizados:

- Aprendizagem supervisionada
- Aprendizagem não supervisionada
- Aprendizagem por reforço

5.2.1 Aprendizagem supervisionada

Na aprendizagem supervisionada, os algoritmos fazem previsões com base num conjunto de exemplos etiquetados fornecidos por si. Esta técnica é útil quando sabe como deverá ser o resultado. Por exemplo, fornece um conjunto de dados que inclui populações de cidades por ano nos últimos 100 anos e deseja saber qual será a população de uma cidade específica dentro de quatro anos. O resultado utiliza etiquetas que já existem no conjunto de dados: população, cidade e ano.

5.2.2 Aprendizagem não supervisionada

Na aprendizagem não supervisionada, os pontos de dados não são etiquetados. O algoritmo etiqueta- Ao organizar os dados ou ao descrever a sua estrutura. Esta técnica é útil quando não sabe como deverá ser o resultado. Por exemplo, fornece dados de cliente e deseja criar segmentos de clientes que gostam de produtos semelhantes. Os dados que está a fornecer não são etiquetados e as etiquetas no resultado são geradas com base nas semelhanças descobertas entre os pontos de dados.

5.2.3 Aprendizagem de reforço

A aprendizagem por reforço utiliza algoritmos que aprendem com resultados e decide a ação a realizar em seguida. Após cada ação, o algoritmo recebe comentários que o ajudam a determinar se a escolha feita foi correta, neutra ou incorreta. É uma boa técnica utilizada para sistemas automatizados que precisam de tomar muitas decisões pequenas sem orientação humana. Por exemplo, se estiver a criar um carro autónomo, quer que este cumpra a lei e mantenha as pessoas seguras. À medida que o carro ganha experiência e um histórico de reforço, aprende a permanecer dentro da faixa, a não ultrapassar o limite de velocidade e a travar quando encontrar peões.

Os algoritmos de *machine learning* ajudam a responder a perguntas demasiado complexas para responder através de uma análise manual. Existem muitos tipos diferentes de algoritmos de *machine learning*. Contudo, por norma, os casos de utilização destes algoritmos enquadram-se numa destas categorias.

- Algoritmos de classificação de duas classes (binários) dividem os dados em duas categorias. São úteis para perguntas com apenas duas respostas possíveis mutuamente exclusivas, incluindo perguntas de sim/não
- Algoritmos de classificação multiclasse (multinomial) dividem os dados em três ou mais categorias. São úteis para perguntas com três ou mais respostas possíveis mutuamente exclusivas
- Algoritmos de detecção de anomalias identificam os pontos de dados que estão fora dos parâmetros definidos para o que é considerado “normal”
- Algoritmos de regressão preveem o valor de um novo ponto de dados com base em dados históricos
- Algoritmos de séries temporais mostram as alterações a um determinado valor ao longo do tempo. Com a análise e a previsão de série temporal, os dados são recolhidos a intervalos regulares ao longo do tempo e utilizados para fazer previsões e identificar tendências, sazonalidade, periodicidade e irregularidade
- Algoritmos de *clustering* dividem os dados por vários grupos ao determinar o nível de semelhança entre os pontos de dados
- Algoritmos de classificação utilizam cálculos de previsão para atribuir dados a categorias predefinidas

5.3 Decisão sobre o tipo de algoritmo

Após alargada análise dos tipos e subtipos de algoritmos de mineração de texto (análise de texto), é óbvia a escolha em algoritmos de machine learning com aprendizagem não supervisionada para execução da nossa análise; a questão está em qual serão usados visto que muitos para os nossos casos poderão ter de ser sujeitos a pré-processamento; o que removeria as nossas requeridas dimensões de análise textual. No entanto vai continuar a haver pré-processamento como algoritmos de redução de dimensionalidade, a diferença comparando com a frase anterior é que a pré-modelação não afetará negativamente os resultados e possíveis associações.

5.3.1 Algoritmos considerados

Dos variados algoritmos vistos e disponíveis na internet ou em bibliotecas de Python (como SciKitLearn, Tensorflow, Keras), demos a decisão de considerar os seguintes algoritmos como candidatos a uso e/ou pertencentes aos grupos de algoritmos usados para comparação de resultados:

- LDA (Latent Dirichlet Allocation): um modelo de distribuição gaussiana, muito usado por empresas de software sobre feedback e bug reports para associação de resultados do QA,
- K-Means Clustering: muito usado para fazer clusters de keywords em redes sociais,
- Naive Bayes Classifier: dos mais usados diariamente, é muito usado para filtros de spam, identificação de características em texto como idade e género e posterior sorteio e categorização,
- KNN (K-Nearest Neighbor): usado para agrupar dados relacionais com os contactos, relatórios, correspondência e emails em empresas,
- SVM (Support Vector Machines): este é usado nos mesmos lugares que regressões lineares, porém mais rápido ou poderoso, é usado para agrupar pontos como texto com imagens ou tópicos de texto em sites de vendas de 2ª mão,
- Decision Tree: usado na análise de feedback positivo e queixas de clientes,
- Association Rules: muito usado na associação de tópicos com relatórios, como medicações com resultados de análises clínicas.

Pode-se notar que aqui foram escolhidos algoritmos de *machine learning* já comuns a grupos que realizaram tarefas semelhantes e que são algoritmos simples e rápidos não sendo mais um algoritmo de uma família de algoritmos (mais complexos ou não, mas que têm muita variedade), tais como redes neurais, algoritmos genéticos ou algoritmos lineares (como regressões lineares).

6 Webgrafia

Devido à enormidade de *websites* visitados apenas conseguimos colocar o motor de busca:

- [Google](#)