Universidade do Minho

Mestrado em Engenharia Informática

Processamento de Linguagens e Conhecimento

Scripting no Processamento de Linguagem Natural

João Pimentel A80874

21 de Junho de 2020

Resumo

O presente projeto consistiu no desenvolvimento de métodos para extrair e analisar avaliações de jogos, filmes e produtos informáticos, presente no *site* da *IGN Portugal*. Para tal, foram utilizadas as linguagens de programação *Python* e as bibliotecas *BeautifulSoup* e *Spacy*, bem como o algoritmo *TF-IDF*. Numa fase de análise de resultados, observou-se que as soluções implementadas cumpriram todos os requisitos, sendo que a sua melhor, ou pior, *performance* depende das restrições da pesquisa.

Conteúdo

1 Introdução **1**

2 Análise e Especificação **2**
2.1 Descrição do Projeto 2
2.2 Especificação de Requisitos 2

3 Concepção da Resolução **3**
3.1 Extração e Armazenamento de Dados 3
3.2 Preparação dos Dados 3
3.2.1 Versão Base 3
3.2.2 Versão com *Lemmatization* 4
3.2.3 Versão com *Lemmatization* e *Word2Text* 4
3.3 *Term Frequency — Inverse Document Frequency* 4
3.3.1 *Term Frequency* 4
3.3.2 *Inverse Document Frequency* 4
3.3.3 *Resultado Final* 5
3.4 Aplicação *Web* 5

4 Análise de Resultados **8**

5 Conclusões e Trabalho Futuro **9**

Lista de Figuras

1 Página inicial da aplicação 6
2 Página de resultados de uma pesquisa 6
3 Página individual de um documento 7

1 Introdução

O presente relatório é o resultado da resolução do terceiro trabalho prático da unidade curricular de *Scripting* no Processamento de Linguagem Natural, do perfil de Processamento de Linguagens e Conhecimento. O foco deste trabalho passou por extrair informação de um *site* de artigos ou publicações, processando esta informação de modo a permitir o desenvolvimento de um motor de pesquisa multi-palavra com auxílio do algoritmo *TF-IDF*. Para tal, teve-se por base a utilização das funcionalidades fornecidas pela linguagem de programação *Python*, bem como a biblioteca *Flask* para implementação de uma interface gráfica para a pesquisa.

O grande objetivo deste projeto consistiu no aumento da experiência dos alunos no âmbito do desenvolvimento de código *Python* para processamento de textos de linguagem natural. Além disso, o projeto visava a consolidação da utilização de bibliotecas de linguagem natural e de programação de interfaces *web*, realçando a utilidade e simplicidade destas para a resolução deste tipo de problemas.

Nos capítulos seguintes serão demonstrados os problemas, formas de resolução, métodos de desenvolvimento e testes efetuados de modo a obter os resultados ideais para o problema proposto.

2 Análise e Especificação

2.1 Descrição do Projeto

O projeto em questão consistiu no desenvolvimento de dois *scripts* no contexto de processamento de linguagem natural, permitindo extrair informação de páginas *web* e classificar publicações com base na existência de certos termos. Além disso, foi desenvolvida uma pequena aplicação *web* para melhor interação durante a pesquisa de termos nas publicações extraídas.

2.2 Especificação de Requisitos

O principal objetivo deste projeto foi extrair publicações de um *site* à escolha, sendo depois possível pesquisar e classificar as mesmas com base na existência de determinados termos. Assim, serão apresentados, em seguida, os requisitos associados ao desenvolvimento de forma detalhada:

1. Deve ser utilizada a linguagem de programação *Python*.
2. Deve ser escolhido um *site* com artigos ou publicações, de modo a serem extraídos textos para análise, com auxílio de um *web scraper*.
3. O conteúdo extraído deve ser armazenado localmente.
4. Deve ser implementada uma interface *web* para melhor interação aquando da pesquisa de publicações.
5. A ocorrência dos termos em diferentes partes das publicações (título, corpo, entre outros) deve ser diferenciada, tendo cada componente um peso distinto.
6. Para pesquisa dos termos nas publicações, deve ser utilizado o algoritmo *TF-IDF*, sendo os resultados devolvidos por ordem de relevância.

3 Concepção da Resolução

Ao longo deste capítulo serão apresentados os métodos associados a cada tarefa do projeto desenvolvido, bem como a forma de funcionamento destes, problemas encontrados e forma de resolver os mesmos.

3.1 Extração e Armazenamento de Dados

O primeiro passo para a resolução do problema passa por extrair informação de um determinado *site* com várias publicações. Assim, o *site* escolhido foi a IGN Portugal, um local onde são colocadas várias avaliações de filmes, jogos e equipamentos informáticos como comandos e teclados.

Escolhido o local para extração da informação, foi necessário definir uma estratégia de obtenção dos dados, já que o *site* possui várias páginas e painéis de conteúdo infinito, ou seja, o conteúdo vai aparecendo à medida que o utilizador faz *scroll* na janela. Além disso, na página inicial das avaliações apenas aparece o título de cada uma, a sua sinopse, classificação de 1 a 10, uma imagem e a data de publicação. Desta forma, de modo a extrair o texto da avaliação, foi necessário armazenar o *link* para a página da avaliação completa. Note-se, ainda, que a avaliação pode possuir mais do que uma página de texto, tendo sido necessário marcar as páginas já visitadas por avaliação, de modo a não se entrar num ciclo infinito.

Desta forma, de modo a extrair os vários elementos numa fase inicial, ou seja, a extração de toda a informação da avaliação, à exceção do subtítulo e corpo em si, foi definido um método com auxílio da biblioteca *BeautifulSoup*, após estudo do formato da página *HTML*. Note-se que, devido à existência de um painel infinito, foi utilizada a biblioteca *selenium* com um *driver* para o motor de busca *Google Chrome*, o qual abria uma janela, percorria a página até ao final, e extraía a informação pretendida. É importante mencionar que apesar deste painel ser infinito, o *site* tinha um limite de conteúdo que seria visível, pelo que o número de avaliações extraídas foi de 600 unidades. Finalizado o processo de extração, estes dados foram armazenados num ficheiro *JSON* para permitir uma mais fácil busca pelos corpos de cada análise.

No que toca à extração dos corpos das avaliações, foi definido um ciclo que percorria todas as avaliações presentes no ficheiro previamente armazenado, aplicando um *scrapper* à página referente à análise em questão. Além de extrair o conteúdo textual da página, como o subtítulo da avaliação e o seu texto, é necessário confirmar a existência de mais páginas, sendo isto efetuado com uma pesquisa por uma divisão da classe *paginator* e obtenção das suas âncoras. Como mencionado anteriormente, de modo a evitar um ciclo infinito, as páginas já visitadas são marcadas. Finalmente, tendo toda a informação pretendida, os dados são armazenados num novo ficheiro *JSON*, permitindo uma rápida utilização para os passos seguintes do projeto.

3.2 Preparação dos Dados

Após a informação estar armazenada localmente, é necessário preparar a mesma para a aplicação do algoritmo de pesquisa. Assim, tendo em conta a realização de mais do que uma versão, de modo a permitir comparações nos resultados, em seguida serão explicados os processos de preparação para cada versão.

3.2.1 Versão Base

Na versão base optou-se por não aplicar qualquer tipo de *lemmatization*. Assim, foi removido qualquer elemento do texto que não fosse uma palavra ou número, com auxílio de uma expressão regular. Após esta filtragem, o resultado obtido foi uma lista de palavras, pelo que se aplicou uma função para converter os caracteres para minúsculos, permitindo uma análise uniforme e mais exata dos termos a pesquisar. Note-se que este processo é aplicado ao título, subtítulo, sinopse e corpo da avaliação, sendo estes os elementos a estudar.

3.2.2 Versão com *Lemmatization*

Em seguida, foi definida uma versão que aplicava *lemmatization* ao conteúdo a estudar, tal como no caso anterior. A primeira etapa foi a remoção de termos que não fossem nem palavras, nem números, ou seja, qualquer sinal de pontuação ou semelhante. Em seguida, os vários caracteres foram transformados na sua versão com letra minúscula, tal como no caso anterior. Por fim, foi aplicada uma função de *lemmatization*, tirando proveito da biblioteca *spacy* e do seu núcleo para língua portuguesa. Note-se que, numa fase inicial, foi ponderada a utilização de *stemming* no lugar de *lemmatization*, mas os resultados obtidos não eram os melhores, sendo que a biblioteca *spacy* permite resultados mais próximos da realidade. Além disso, em vez de uma remoção de prefixos e sufixos, as palavras são analisadas com base numa versão de dicionário, algo que, na sua grande maioria, esta biblioteca permite com resultados bastante bons. É importante mencionar que, sendo esta uma operação bastante pesada computacionalmente, foi definido um *script* para tratar dos dados presentes no ficheiro completo das análises, gerando um novo pronto a ser utilizado pela aplicação.

3.2.3 Versão com *Lemmatization* e *Word2Text*

Foi, ainda, definida uma versão que convertia valores numéricos na sua versão textual. Isto pode ser importante explorar já que para um utilizador, a pesquisa por 100 e *cem* são iguais, algo que não acontece para uma linguagem de programação. Desta forma, antes da aplicação de qualquer processo de processamento da versão anterior, foi aplicada uma substituição dos valores numéricos pela sua versão textual, com auxílio da biblioteca *num2words*. Tendo em conta que os textos a analisar estão escritos em português, também a representação dos números tem que estar, pelo que a função utilizada recebeu como argumento da variável *lang* a língua portuguesa. Após esta substituição, foram aplicadas todas as outras transformações da versão com *lemmatization*, sendo, mais uma vez, os resultados armazenados em ficheiro.

3.3 *Term Frequency — Inverse Document Frequency*

No que toca ao algoritmo de pesquisa de termos nos vários documentos, o primeiro passo é o carregamento dos dados vindos do ficheiro em estudo. Em seguida, é efetuada a separação do texto em várias palavras, ou seja, a conversão de *strings* em listas de termos. De modo a simplificar as operações, por cada documento é armazenado o total de palavras neste, ou seja, a soma do total de palavras no título, subtítulo, sinopse e corpo, bem como os totais destes campos.

3.3.1 *Term Frequency*

Feito o processamento inicial, podem ser calculadas as métricas em estudo. Assim, começando pela métrica *TF*, ou seja, *Term Frequency*, o objetivo é obter a frequência de um termo num determinado documento. Desta forma, a ideia passou por definir um dicionário cujas chaves fossem os identificadores dos ficheiros, sendo os valores um novo dicionário dividido pelas partes dos documentos: título, subtítulo, sinopse e corpo. Cada uma destas, é composta por um outro dicionário, sendo este um contador de ocorrências dos termos no campo da avaliação em questão. Este dicionário permite a obtenção do número de ocorrências de um termo num determinado documento e parte do mesmo. Assim, no momento do cálculo, é necessário dividir este valor pelo total de palavras do documento em estudo, algo simples e efetuado de forma eficiente tendo em conta a utilização de um dicionário para alocar os documento e o seu total de palavras, como visto anteriormente.

3.3.2 *Inverse Document Frequency*

No que diz respeito à métrica de *DF* (*Document Frequency*), esta carateriza-se por indicar a contagem de ocorrências de um termo no conjunto de documentos em análise. Assim, a ideia passa por possuir um dicionário cujas chaves são os vários termos em estudo e os respetivos valores são a contagem de documentos onde o termo aparece. No entanto, a métrica que se pretende obter é *IDF*, ou seja, *Inverse Document Frequency*. Para tal, por cada termo esta é obtida pela aplicação de

um logaritmo à divisão do total de documentos pela métrica de *DF* previamente calculada, sendo utilizado para a divisão o valor 1 no caso do termo não existir no dicionário.

3.3.3 *Resultado Final*

Finalizados os cálculos das métricas de *TF* e *IDF*, é possível calcular a métrica *TF-IDF*, sendo esta obtida pela multiplicação das anteriores. Note-se, contudo, que existem pesos diferentes para os vários campos dos documentos, pelo que devem ser diferenciados nos cálculos. Assim, o cálculo deve ser efetuado por partes. No que toca à métrica de *IDF*, o seu valor é independente do local do documento onde a palavra ocorre. Já a métrica de *TF* depende desse local, sendo necessário atribuir um peso a cada componente do documento. Note-se que a definição do dicionário dos valores de *TF*, separando as entradas do documento em entradas pelas componentes do mesmo, permite uma fácil compreensão de como se podem obter os valores. Deste modo, assumindo uma soma de pesos de 1 para os 4 campos, o título ficou com o maior peso, sendo este de 0.4. O subtítulo ficou com um peso de 0.3, seguido da sinopse com 0.2. Por fim, o corpo da avaliação ficou com a menor porção, sendo a sua métrica de *TF-IDF* multiplicada por 0.1.

Compreendido o cálculo das métricas, falta compreender como são processados os termos a pesquisar e os resultados enviados e filtrados. No que toca aos termos, estes sofrem o mesmo processamento que as palavras dos textos, ou seja, no caso da versão base os termos apenas são convertidos para minúsculas. Já no caso da versão com *lemmatization*, após a conversão para minúsculas, o termo é convertido no seu lema. Quanto aos resultados, é gerado um dicionário com os resultados do cálculo do algoritmo *TF-IDF*, onde cada chave é um termo em pesquisa, sendo depois gerado um novo dicionário cuja chave é o identificador do documento. A este é somado o valor calculado pelo algoritmo por cada termo em pesquisa, ou seja, numa pesquisa por vários termos, o valor associado a cada documento passa pela soma dos resultados do algoritmo por todos os termos para aquele documento específico. É importante destacar que, de modo a só apresentar ao utilizador os documentos importantes, os resultados são ordenados de forma decrescente, sendo filtrados os documentos cuja métrica seja 0.

3.4 Aplicação *Web*

O último requisito do projeto era a definição de uma interface *web* para simplificar a pesquisa de termos nos vários documentos. Desta forma, foi utilizado o módulo *Flask*, permitindo desenvolver uma pequena aplicação com apenas 3 janelas, sendo uma interface para pesquisa, uma interface de listagem de resultados e uma apresentação de uma avaliação.

Pela observação da Figura 1, a página inicial da aplicação é bastante simples, possuindo apenas uma caixa de texto para o utilizador indicar os vários termos que pretende pesquisar, devendo estes ser separados por espaços. Note-se que na imagem se pode ver uma pesquisa por *Canalizadores Italianos*, sendo, nas versões com *lemmatization*, convertido para *canalizador italiano*, antes de efetuar a pesquisa nos documentos.

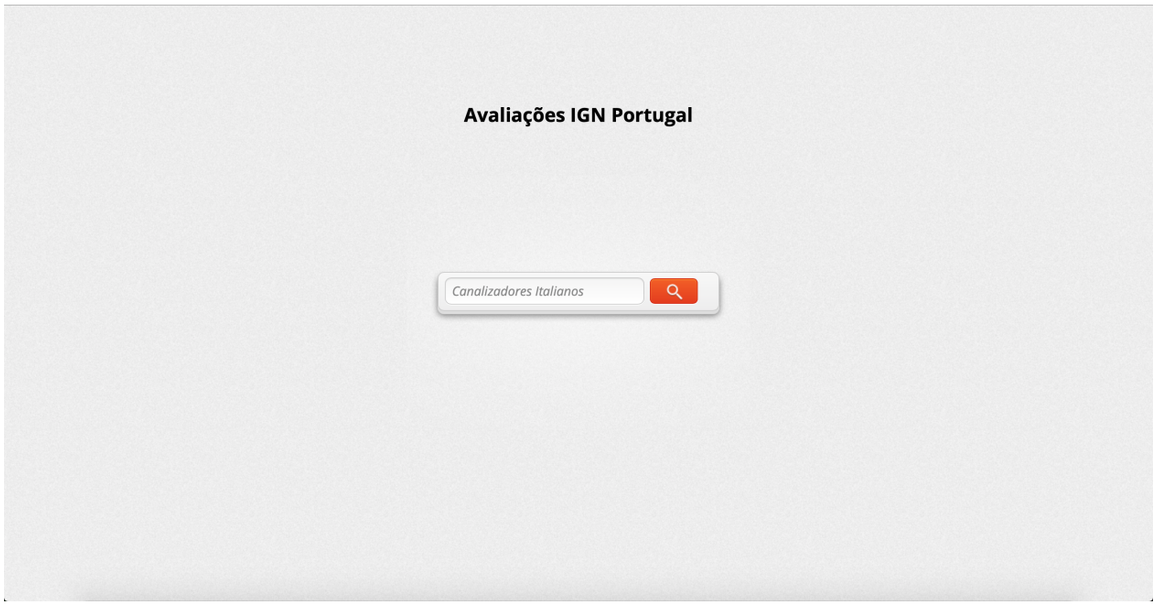

Figura 1 - Página inicial da aplicação.

Após a indicação dos termos, o utilizador fica com um ecrã semelhante ao da Figura 2, onde lhe são apresentados os vários documentos e os respetivos valores da aplicação do algoritmo de pesquisa.

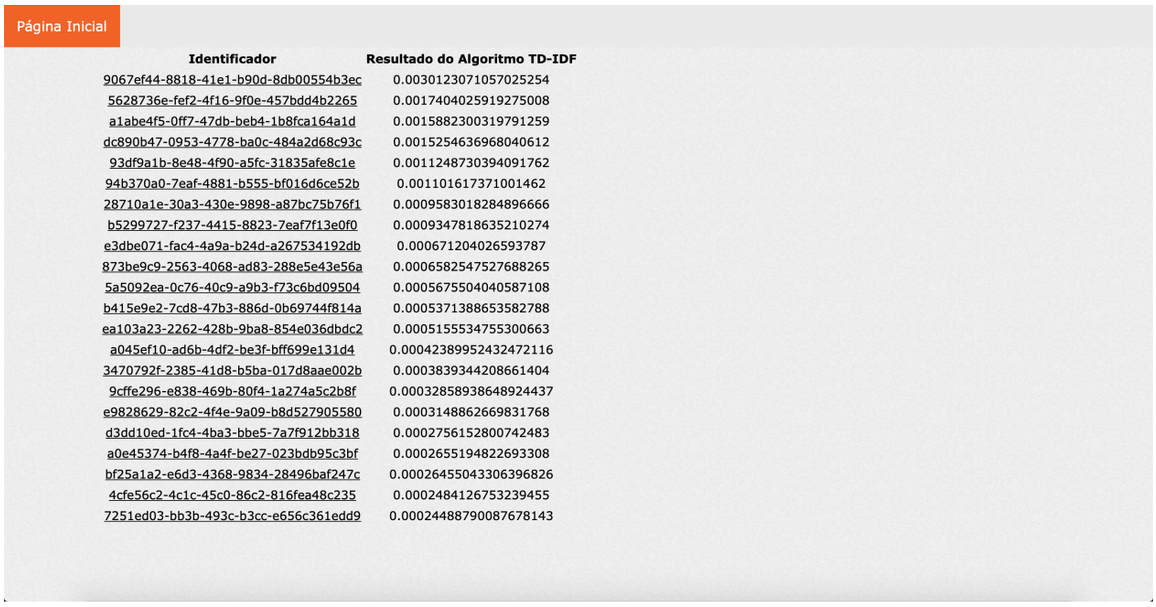

Figura 2 - Página de resultados de uma pesquisa.

Por fim, se o utilizador clicar num dos identificadores apresentados nos resultados, este é redirecionado para uma página onde pode aceder à avaliação, sendo esta apresentada num formato semelhante ao visível na Figura 3.

Figura 3 - Página individual de um documento.

4 Análise de Resultados

De modo a comparar as 3 versões implementadas, foram testados alguns termos distintos. A primeira versão desenvolvida permite obter resultados para comparações literais dos termos, ou seja, o resultado depende da existência do exato termo em pesquisa nos documentos. No entanto, nem sempre uma pesquisa destas é o que o utilizador pretende. A título de exemplo, caso o utilizador pretenda encontrar todos os documentos que mencionem *milhões*, este também pode querer documentos que contenham a palavra *milhão*, sendo que a *lemmatization* permite este tipo de resultados. O mesmo acontece se for pesquisado algum tipo de verbo, sendo, desta forma, possível detetar documentos com diferentes tempos verbais do mesmo termo.

Além deste tipo de pesquisa, se um utilizador pretender pesquisar por unidades, ou seja, se pretender encontrar todos os documentos com o termo *onze*, o número 11 também representa o mesmo, aos olhos do utilizador. Desta forma, a versão final, ou seja, a versão que trata de converter todos os números para a sua versão textual, aplicando, posteriormente, *lemmatization* é a que trará melhores resultados.

Em jeito de resumo, as três implementações produzem resultados bastante bons, sendo a versão inicial a mais dependente dos termos em si. No entanto, não se pode dizer que as versões com *lemmatization* são sempre melhores que a versão base, uma vez que o processo de descoberta dos lemas das palavras nem sempre é o correto, existindo erros que podem alterar os resultados. Assim, se o pretendido for uma obtenção de documentos com a existência literal dos termos em si, a versão base é a que trará melhores resultados. No caso de serem pretendidos resultados menos dependentes de a palavra estar num determinado tempo verbal, singular ou plural, a utilização de lemas traz vantagens extremas. Por fim, a versão com conversão de números para a sua forma textual foi feita em jeito de extra aos lemas, permitindo melhorar os resultados em geral em pesquisas por números.

5 Conclusões e Trabalho Futuro

Ao longo do presente relatório encontra-se representado o resultado do terceiro trabalho prático da unidade curricular de *Scripting* no Processamento de Linguagem Natural. Neste contexto foram abordados os passos referentes ao desenvolvimento e implementação de vários métodos que permitissem a extração, processamento e ordenação de documentos com base no algoritmo de *Term Frequency — Inverse Document Frequency*.

O desenvolvimento do projeto permitiu aprimorar os conhecimentos previamente adquiridos e postos em prática no decorrer das aulas da Unidade Curricular. Desta forma, foi possível melhorar uma versão da aplicação do algoritmo em estudo, desenvolvido em contexto de aula, sendo este agora capaz de diferenciar a ocorrência de termos em diferentes partes do documento. Além disso, a conversão de termos nos seus lemas mostrou-se capaz de permitir a obtenção de resultados bastante interessantes para este tipo de problemas, estando presente alguma margem para erros devido aos algoritmos e módulos utilizados. Foi, também, notória a facilidade de extração de informação de páginas *web*, permitindo a construção de conjuntos de dados para consequente estudo, com auxílio de bibliotecas de *web scrapping*.

Em suma, a realização deste trabalho exigiu a aplicação de todos os conhecimentos lecionados em contexto de aula, permitindo que fossem cumpridos todos os objetivos propostos no enunciado, bem como a realização de extras e comparação do seu impacto no resultado final.