Geração Automática de Perfis de Pessoas Candidatas a um Emprego

Francisco Antônio Borges Paulino, Paulo Jorge Fernandes Freitas

UC Projeto
Licenciatura em Ciência das Computação, Universidade do Minho
a91666@alunos.uminho.pt, a100053@alunos.uminho.pt

Resumo. Neste projeto, foi desenvolvido um programa de software relativo a geração automática de perfis de candidatos a um emprego, de forma a otimizar e agilizar os processos de recrutamento e avaliação de candidatos. O programa foi implementado na linguagem Python com recurso às bibliotecas "RDFLib", a partir do qual são gerados grafos dos dados de candidaturas e proposta de emprego. Através de várias etapas, o programa tem como objetivo tratar os dados provenientes de candidaturas e da proposta para um dado trabalho, e verificar quais os candidatos mais compatíveis com a proposta de emprego, através do uso de um algoritmo de comparação de triplos, de forma a averiguar a similaridade dos dados presentes nos grafos. Apesar do tema ser bastante desafiador e complexo, foi obtido um programa simples, mas robusto, capaz de organizar os perfis de candidatos e determinar a sua adequação às vagas, com possível otimização no futuro, tanto a nível de interface como em expansão de funcionalidade.

Palavras-Chave: gafos, similaridade, candidatos.

1 Introdução

No âmbito desta Unidade Curricular, foi nos apresentado diversas propostas de projetos a desenvolver, das quais escolhemos o projeto de Geração Automática de Perfis de Pessoas Candidatas a um Emprego.

Este projeto consiste no desenvolvimento de um sistema de software capaz de estabelecer perfis detalhados de candidatos a partir de interações configuradas com base em especificações de características desejadas para uma determinada posição de emprego.

As motivações que levam à realização deste trabalho partem do princípio de tentar otimizar e agilizar os processos de recrutamento e seleção, uma vez que, no mercado de trabalho, tal sistema de software poderia ajudar a lidar com grandes volumes de candidaturas, mostrando-se uma ferramenta de triagem útil.

Os objetivos principais deste projeto consistem em criar um software que possa gerar perfis detalhados de candidatos com base nas especificações predefinidas, coletar in-

formações relevantes de cada candidato de forma a gerar, mais tarde, um perfil adequado e reduzir o tempo e o esforço necessário para a triagem e avaliação de candidaturas, tendo em consideração a proposta de emprego.

Para abordar o projeto realizado, dividimos este relatório em 6 partes, sendo estas a introdução, uma pequena abordagem do que é a geração automática de perfis de pessoas candidatas a um emprego, o processo de como abordamos o nosso projeto, alguns exemplos demonstrativos do nosso caso de estudo, algumas considerações sobre os algoritmos relativos a similaridade e por fim, as conclusões que obtivemos.

2 Geração Automática de Perfis de Pessoas Candidatas a um Emprego

A geração automática de perfis de candidatos envolve o uso de grafos e algoritmos para compilar e analisar informações relativas a candidatos a um emprego. Esses sistemas podem coletar dados de currículos e/ou outras fontes relevantes, de forma a construir um perfil que inclua as habilidades, experiências, qualificações e todas as características relevantes para a vaga de emprego que se está a candidatar e posteriormente verificar se são adequadas.

As aplicações da geração automática de perfis de candidatos focam-se essencialmente na automatização da triagem inicial, de forma a reduzir o tempo e esforço necessários para identificar os candidatos qualificados para um dado emprego, como também para realizar uma avaliação inicial, tendo em conta a comparação automática de perfis dos candidatos com as especificações da vaga de emprego.

3 O Processo de Geração Automática de Perfis de Pessoas Candidatas a um Emprego.

O presente programa foi elaborado em linguagem Python com recurso à biblioteca "RDFLib", uma biblioteca que permite a elaboração e manuseio de grafos.

Para a elaboração do programa, foram definidas etapas com objetivos diferentes e sequenciais entre si, de forma a criar um modelo de melhor compreensão dos processos do programa. Para tal, definimos quatro momentos (funções) no código.

Numa primeira fase, o programa gera um grafo com informações do tipo de dados que irá armazenar. Após isso, o programa lê de uma pasta "candidaturas" todos os ficheiros nela incluídos, e por cada ficheiro presente, é feita uma leitura das informações contidas no arquivo. Nessa leitura, são obtidos dados dos candidatos e são adicionados ao grafo das candidaturas de forma a manter as relações indicadas na candidatura.

No segundo passo, o programa executa um momento parecido com o anterior, porem apenas para um ficheiro "proposta". Nesta fase, são armazenados num grafo, as informações da proposta de emprego de qual os candidatos se aplicam.

Na terceira fase do código, existe um momento de comparação onde verifica-se a existência de dados iguais entre as informações de proposta e informações do candidato. Caso uma informação seja igual nos dois grafos, então o candidato possui uma propriedade que é procurada na proposta. O subgrafo de cada candidato é obtido a partir da pesquisa dos triplos onde o sujeito é o candidato em questão. Após a extração do subgrafo, é executada uma comparação entre o subgrafo de cada candidato e o grafo "proposta". A comparação consiste em um "Algoritmo de comparação de triplos", isto é, são comparados os predicados e os objetos de cada triplo presentes nos grafos. Este algoritmo foi alterado de forma a conseguir comparar predicados semelhantes, visto que, possuem denominações diferentes entre os dois grafos, porem acerca dos mesmos dados. Após a comparação são atribuídos pontos aos candidatos de acordo com a rubrica e a validação da mesma. Após verificar a similaridade entre todos os candidatos e a proposta de emprego, são apresentados os candidatos e a sua pontuação, por ordem decrescente pelo resultado obtido na comparação.

O último momento consiste na possibilidade de o utilizador ser capaz de verificar os dados dos candidatos, da proposta de emprego, e ainda uma descrição detalhada da pontuação de cada candidato e as rubricas validadas na comparação.

4 Um Caso de Estudo

Tendo em conta o IO do programa e o seu funcionamento, precisamos de alguns candidatos e as suas informações, e também de informações de uma vaga de emprego. Para isso, usaremos Alberta Soares e Francisco Manuel, dois candidatos distintos a uma vaga.

Tomemos o seguinte exemplo de dados de candidatos e proposta:

Seja dois candidatos com os seus dados:

```
Nome: Alberta Soares
Idade: 24
Sexo: F
Nacionalidade: PT, JP, BR
Experiencia Proficional: Informatica 36
Estudos: Engenharia Informatica
Hablilidades: Cplusplus 3, SQL 1, PYTHON 2, Ingles 2
Preferencias de trabalho: Todos
Qualidades interpessoais: Trabalho em equipa
Carta de Conduçao: N
Email: Albertasoa@hmail.com
Telemovel: +351 999554332
Endereço: Rua de Cima, Braga
```

Figura 1 - A candidatura de Alberta Soares

```
Nome: Francisco Manuel
Idade: 53
Sexo(M/F): M
Nacionalidade: PT
Experiencia Proficional: Informatica 134
Estudos: Engenharia Imformatica
Hablilidades: C 5, Cplusplus 3, SQL 1, Python 3
Preferencias de trabalho: Presencial
Qualidades interpessoais: multi-tasking
Carta de Conduçao: B, A, D
Email: xico@xicodev.com
Telemovel: 9999998
Endereço: Lisboa, Lisboa
```

Figura 2 - A candidatura de Francisco Manuel

E seja a seguinte proposta de emprego:

```
Idade: 16-26
Sexo: F, M
Nacionalidade: PT, BR, JP
Area de Trabalho: Informatica 32, Computacao 10
Estudos Preferencial: Engenharia Informatica
Estudos Relativos: Ciencias Computacao
Habilidades Preferencial: Cplusplus 5
Habilidades Relativas: SQL 3
Regime de Trabalho: Todos
Qualidades Preferenciais: Trabalho em equipa
Qualidades Relativas: Pontual
Carta de Conducao_Preferencial: Ligeiros
Carta de Conducao_Relativas: Pesados
Localidade Preferencial: Braga
Localidade Relativa: Porto
```

Figura 3 - Informações da vaga de emprego

Temos que, após a execução da leitura dos candidatos, da proposta e da sua comparação, obteve-se o seguinte resultado de pontuação:

```
Iniciando leitor de candidaturas...
(leitor_candidaturas)>> A ler candidaturas de: C:\Users\paulo\OneDrive\Ambiente de Trabalho\projeto\candidaturas .
 Foram encontrados 2 arquivos neste diretorio.

Iniciando leitor de proposta...
(leitor_proposta)>> Foi detetado ficheiro de referencia.
(leitor_proposta)>>Os dados da vaga de emprego foram guardados.

Iniciando avaliador de candidaturas...
1 --> Francisco Manuel -- 285 pontos
2 --> Alberta Soares -- 122 pontos
```

Figura 4 - Resultado da atribuição de pontuações

E os descritivos das pontuações são:

```
A mostrar os descritivos das pontuações de cada candidato.
(Pontuação)>> O candidato Alberta Soares teve pontuação das seguintes categorias:
  hasIdade validou 24 e obteve 12 pontos,
  hasSexo validou F e obteve 2 pontos,
  hasNacionalidade validou PT e obteve 2 pontos,
  hasExp_Profissao validou Informatica e obteve 72 pontos,
  hasEstudos validou Engenharia Informatica e obteve 10 pontos,
  hasHabilidades validou Cplusplus e obteve 9 pontos,
  hasHabilidades validou SQL e obteve 1 pontos,
  hasPreferencia_Trab validou Remoto e obteve 3 pontos,
  hasQual_Interpessoais validou Trabalho_em_equipa e obteve 4 pontos,
  hasEndereco validou Braga e obteve 7 pontos,
 No total obteve 122 pontos.

(Pontuação)>> O candidato Francisco Manuel teve pontuação das seguintes categorias:
  hasSexo validou M e obteve 2 pontos,
  hasNacionalidade validou PT e obteve 2 pontos,
  hasExp_Profissao validou Informatica e obteve 268 pontos,
  hasHabilidades validou Cplusplus e obteve 9 pontos,
  hasHabilidades validou SQL e obteve 1 pontos,
  hasPreferencia_Trab validou Presencial e obteve 3 pontos,
 No total obteve 285 pontos.
```

Figura 5 - Descritivos das pontuações

5 Outros algoritmos

Durante a pesquisa de algoritmos para a elaboração do programa encontramos vários algoritmos que permitiam, de forma mais complexa e aplicando mais conhecimento, realizar a comparação entre grafos, dentro dos quais:

- Isomorfismo de Grafos – Algoritmo de Ulman e Algoritmo de VF2 – Dois grafos são iguais se forem isomorfos, onde para dois grafos serem isomorfos, precisam possuir a mesma quantidade de vértices e arestas. Estes algoritmos não são aplicáveis no programa devido a diferença de quantidades de vértices a arestas entre os grafos que comparamos.
- Isomorfismo de Subgrafos - Algoritmo de Backtracking – Semelhante aos anteriores, este algoritmo verifica se um subgrafo mantém a estrutura do grafo pai. Este algoritmo também não é aplicável pois não pretendemos comparar um grafo e o seu subgrafo, mas sim um grafo "a" e o subgrafo de um grafo "b".
- Maximum Common Subgraph (MCS) - Algoritmo de Bron-Kerbosch – Este algoritmo pretende determinar o maior subgrafo comum entre dois grafos. Este algoritmo seria um forte candidato para ser usado no comparador do programa, porem, como a estrutura do subgrafo e do grafo são diferentes em algumas propriedades (ie. Subgrafo candidato: HasHabilidade, Grafo proposta: HasHabilidade_Pref), logo impossibilitando a utilização do algoritmo na estrutura atual do programa.

6 Conclusões e Trabalho Futuro

Numa fase inicial deste projeto, o nosso maior desafio foi a compreensão do tema proposto, uma vez que era algo que nunca tínhamos abordado previamente. Contudo, com a ajuda do nosso orientador e após algumas semanas de pesquisas, começamos a entender um pouco melhor o objetivo e a grande utilidade que oferece no recrutamento para o mercado de trabalho.

Posteriormente, outro desafio que nos deparamos foi a compreensão e o uso das bibliotecas "RDFLib", uma vez que foi a primeira vez que tentamos definir uma ontologia para a estrutura de grafos que queríamos obter, que representaria os perfis dos utilizadores, com base nas suas candidaturas. Para além disso, a procura e a implementação de um algoritmo que conseguisse satisfazer as nossas necessidades no programa que implementamos, também demonstrou ser uma dificuldade.

Apesar das dificuldades referidas, pensamos que implementamos um programa simples, contudo robusto, que dado um grupo de candidaturas e uma proposta de emprego, consegue organizar os perfis dos candidatos, verificar as semelhanças de cada um com

a proposta de emprego e obter uma lista dos candidatos ordenada pelos mais adequados até aos menos adequados.

Em termos futuros, o programa poderia ser expandido em vários aspetos. Poderia ser implementada uma interface mais "amiga" do utilizador, ou seja, mais apelativa e intuitiva tanto para os candidatos como para os recrutadores. No que diz respeito ao programa em si, poderia aceitar mais rúbricas nos candidatos, de forma a obter ainda mais pontos de avaliação, com o objetivo de otimizar a seleção dos candidatos, bem como a implementação de mais métodos comparativos e/ou de similaridade entre grafos, que respeitasse a estrutura atual, mas oferecesse um programa mais dinâmico e com mais opções.

Referências

1. Python Software Foundation. (2024). *Python 3 Documentation*. Recuperado de https://docs.python.org/3/ (Último acesso em 28-05-2024)

2. RDFLib. *RDFLib Documentation*. Recuperado de https://rdflib.readthedocs.io/en/stable/ (Último acesso em 28-05-2024)

3. Segaran, T., Evans, C., & Taylor, J. (2009). *Programming the Semantic Web*. O'Reilly Media.

4. Messmer, B.T., & Bunke, H. (1993). RA network based approach to exact and inexact graph matching. Institut für Informatik und angewandte Mathematik, Universität Bern.

5. Kessler, R., Béchet, N., Roche, M., El-Bèze, M., & Torres-Moreno, J. M. (2008). Automatic profiling system for ranking candidates answers in human resources. In OTM Workshops 2008 (pp. 625-634)