UNIVERSIDADE DE AVEIRO

Técnicas de Mineração de Texto - 2

Licenciatura em Engenharia de Computadores e Informatica

UC: Projeto em Engenharia de Computadores e Informática

Equipa: Gabriel Boia (113167), Guilherme Matos (114252), Rafael Dias (114258), Tiago Costa (114629) e Tiago Almeida (113106)

Supervisor: Luís Filipe de Seabra Lopes

December 7, 2024

Relatório do Projeto: 1ª Versão

December 7, 2024

Contents

1 Introdução

Quando empresas trabalham com grandes quantidades de dados, é importante que os mesmos sigam um padrão definido, para que a procura e atualização de informação dentro de conjuntos de dados seja rápida e eficaz. O grupo responsável pelo projeto aqui descrito propôs-se, então, a procurar resolver este problema através de um método automático de correção de bases de dados. Os dados de treino utilizados foram fornecidos pela empresa que submeteu a proposta do projeto: Administração dos Portos de Sines e do Algarve, S.A.
Posto isto, o presente relatório tem como objetivo dar a conhecer o trabalho realizado neste âmbito, inserido na Unidade Curricular de "Projeto em Engenharia de Computadores e Informática". Este trabalho foi realizado ao longo do ano letivo 2024/2025, sob supervisão do professor Luís Seabra Lopes, e com acompanhamento do mesmo e dos professores responsáveis pela Unidade Curricular.

1.1 Contexto

Dado um conjunto de dados no qual toda a informação em cada linha é escrita de forma livre, isto é, não há restrições sobre o método de introdução de informação em cada entrada, é evidente que eventualmente os dados sejam introduzidos incorretamente ou que, pelo menos, não estejam padronizados. No contexto específico do projeto realizado, os conjuntos de dados são referentes a dados de carga que entram e saem do Porto de Sines.
Sendo assim, se existirem várias entradas para a mesma empresa, é de esperar que, por exemplo, o nome da empresa ocasionalmente esteja escrito de forma diferente, com alguma pontuação a mais, ou a menos. Isto aplica-se para todos os dados, incluindo NIF e outros.
Para isso, se existir (e existe, nos dados fornecidos), uma base de dados com os nomes e informação geral sobre cada empresa padronizados, é possível corrigir os dados originais substancialmente, reduzindo em massa o trabalho manual necessário para corrigir os restantes dados que possam eventualmente não ter sido corrigidos automaticamente.

1.2 Objetivos

O objetivo do projeto desenvolvido incide na criação de uma aplicação que tem como finalidade principal corrigir e padronizar grandes bases de dados. Para este efeito, é necessário que existam dois tipos conjuntos de dados:

1. O conjunto de dados inicial, a corrigir
2. O conjunto de dados que inclui as informações padronizadas de cada entrada (e.g. nome da empresa, NIF, etc.)

O grupo responsável propõe-se a desenvolver esta aplicação, tendo ainda em mente possíveis *features* adicionais, que possam ser úteis.

2 Requisitos

Nesta secção, abordamos a escolha dos *stakeholders* envolvidos no projeto, os requisitos e métodos de recolha dos mesmos e a sua importância.

2.1 Análise dos *Stakeholders*

Tendo em mente o objetivo fundamental do projeto, é possível identificar três *stakeholders*

1. Todos os membros do grupo, como desenvolvedores da aplicação;
2. O professor orientador do projeto (Prof. Luís Seabra Lopes);
3. A empresa que submeteu a proposta do projeto: Administração dos Portos de Sines e do Algarve, S.A.

2.2 Táticas de Recolha de Requisitos

Após a definição dos stakeholders, o próximo passo óbvio a seguir foi a recolha dos requisitos do sistema.
Para tal, decidimos fazer uso de várias táticas diferentes, como:

1. Contacto direto com os stakeholders;
2. Reuniões de grupo;
3. Feedback dos professores da unidade curricular.

2.2.1 Contacto direto com os *stakeholders*

A partir do contacto com os *stakeholders* citados, nomeadamente através de reuniões conjuntas, o grupo definiu os requisitos mais importantes, isto é, os que definem o objetivo do trabalho e que devemos alcançar para considerar o projeto como bem-sucedido.

2.2.2 Reuniões de grupo

Através das reuniões com os *stakeholders* obteve-se uma perspetiva de como seria possível alcançar os requisitos mais importantes e, para além disso, definiram-se ainda requisitos secundários que o grupo sentiu que trariam mais valor ao projeto sem comprometer a consecução dos requisitos principais.

2.2.3 *Feedback* dos professores responsáveis pela UC

Por último, as aulas da Unidade Curricular associada a este projeto, mais especificamente as aulas onde houve um contacto mais direto com os professores, capacitaram uma solidificação das ideias previamente discutidas em grupo e também trouxeram novas ideias e pontos de vista úteis no desenvolvimento do trabalho.

2.3 Requisitos Definidos

Uma vez que já se encontram explicitados os métodos de levantamento de requisitos, apresentam-se de seguida os requisitos definidos em última análise.

2.3.1 Requisitos Principais

Descrição do Requisito	**Tipo**
Conseguir corrigir os dados	Funcional
Assegurar precisão na correção	Não Funcional
Ser adaptável (capaz de trabalhar com diferentes datasets)	Funcional
Reduzir significativamente o trabalho manual de correção seguinte	Funcional

Table 1: Resumo dos Requisitos Principais

2.3.2 Requisitos Secundários

Descrição do Requisito	**Tipo**
Baixa complexidade temporal	Não Funcional
Inserir novas entradas nos dados padronizados (quando surgem novas empresas)	Funcional
Capacidade de sugerir entradas quando alguém insere uma entrada que se assemelhe a um campo padronizado	Funcional
Ter uma *User Interface* de forma a poder interagir com o programa de forma fácil	Não Funcional

Table 2: Resumo dos Requisitos Secundários

3 *Background* e Trabalhos Relacionados

3.1 Contextualização do Problema

Este projeto insere-se na área de *data science*, mais concretamente no âmbito de *data cleansing*, que é uma etapa fundamental na preparação dos dados para a sua análise e tratamento. No contexto do nosso projeto, como se trata de algo real relativo a uma empresa, a padronização dos dados é crucial para garantir que seja mantida a integridade dos dados registados no sistema do porto, bem como para facilitar a sua gestão.
Um dos maiores dificuldades no trabalho é lidar com conjuntos de dados de grande dimensão, que estão fora do padrão de registo, muitos ainda com erros ortográficos. Por exemplo, muitos dos campos apresentam apenas abreviações das empresas e, alguns ainda têm caracteres de sinais de pontuação no início ou fim das *strings* de texto.

3.2 Soluções Existentes

Numa pesquisa inicial, o grupo teve a preocupação de procurar trabalhos e técnicas de *data cleansing* que fossem semelhantes ao problema em mãos, de modo a facilitar a recolha de dados inicial, bem como o levantamento de técnicas que poderiam ser aplicadas na resolução do problema. Foram assim encontrados dois projetos considerados relevantes:

1. **SymSpell Algorithm**: Este projeto oferece uma abordagem eficiente para a correção de *strings* relativamente curtas, preparada para lidar com grandes volumes de dados. Este trabalho tem por base o *Symmetric Delete Algorithm*. Este algoritmo reduz a complexidade do problema ao gerar variações mínimas de cada *string*, de modo a simular possíveis erros ortográficos que possam ser cometidos.
   Isto passa pela remoção ou adição de certos caracteres à palavra, ou até mesmo a troca de caracteres adjacentes. Estas variações são então procuradas nas keys de um dicionário de correspondência, que tem como *value* de cada *key* a *string* correta. Caso seja obtida mais do que uma correspondência, é então calculada a distância de Levenshtein da *string* original ao valor da correspondência obtida, de modo a verificar qual a *string* pretendida (sendo possível um match de mais do que uma correspondência legítima). Não sendo obtidos resultados diretos, ou a palavra não é corrigida, ou é feita uma procura mais profunda com a distância de Levenshtein.

2. **CPClean**: Este projeto, que é bastante menos popular que o supracitado, passa por uma estratégia que tem como objetivo a limpeza dos dados pretendidos, de modo a melhorar a exatidão de modelos de correção através de *machine learning*. Assim, o seu objetivo não é a correção direta dos campos de dados, mas sim a sua limpeza e preparação que auxilia na correção feita posteriormente.

3.3 Inovação e diferenciação

Apesar da relevância de ambos os projetos, estes têm as suas limitações e problemas, quer no seu todo, quer no contexto do nosso trabalho.
Uma limitação do **SymSpell Algorithm** passa pelo facto de estar projetado para lidar com palavras ou *strings* de dimensões reduzidas, que não é o caso com o tipo de dados que pretendemos corrigir, que podem ser de grande dimensão.
Em relação ao **CPClean**, um dos seus maiores problemas é a escassez de documentação do projeto. Para além disso, o grupo decidiu afastar-se, numa fase inicial, de métodos que façam uso direto de *machine learning* para correção dos campos de dados, ainda que não excluindo esta alternativa de resolução. Este afastamento veio do feedback dos professores da Unidade Curricular, aquando do início da exploração do tema.
No entanto, consegui-se retirar destes projetos conhecimentos úteis para a correção dos campos de dados, como o uso das distâncias de Levenshtein e a utilização de um dicionário de correspondências com a intenção de reduzir os tempos de resolução de campos de dados com erros que já tenham sido vistos/resolvidos anteriormente.

4 Design e Soluções Viáveis

4.1 Design do Sistema

O sistema desenvolvido para validar e corrigir nomes baseia-se em três componentes principais:

- **Dataset a ser corrigido**: Contém os nomes que precisam de ser analisados e corrigidos.
- **Dataset padrão**: Lista com os nomes que representam as versões corretas e consistentes, usada como referência.
- **Dicionário JSON de correção**: Mapeia nomes incorretos conhecidos para as suas versões corrigidas, permitindo correções automáticas para entradas já identificadas.

O fluxo do sistema é ilustrado na Figura 1. A arquitetura segue as seguintes etapas principais:

1. **Consulta inicial**: O sistema inicialmente verifica no *dataset* padrão se existe alguma correspondência.
2. **Consulta inicial no dicionário**: O sistema verifica no JSON se o nome possui uma correção já registada.
3. **Correção por similaridade**: Caso não seja encontrado, é aplicada a métrica de distância de Levenshtein para determinar a entrada mais próxima no *dataset* padrão.
4. **Atualização contínua**: Se uma correspondência for encontrada, a nova associação é adicionada ao dicionário.
5. **Registo de erros não resolvidos**: Quando nenhuma correspondência aceitável é identificada, a entrada é guardada para revisão manual.

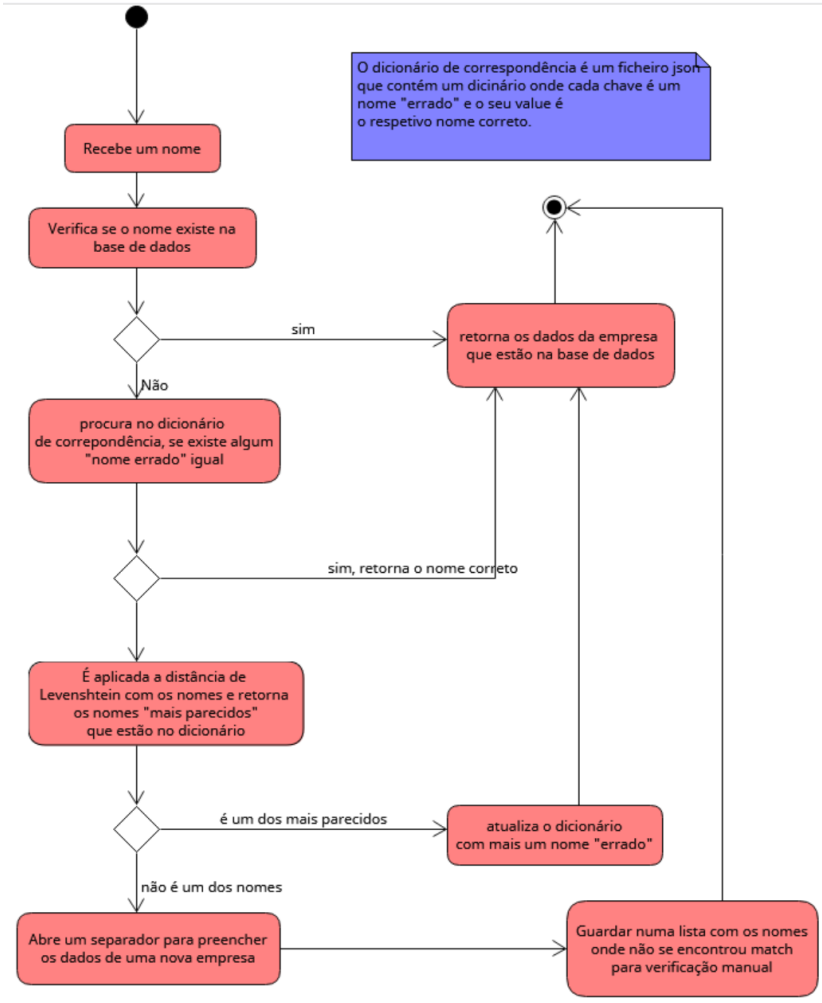

Figure 1: Fluxo do Sistema de Correção de Nomes

4.2 Soluções Alternativas

Durante o desenvolvimento, outras soluções foram consideradas, cada uma com vantagens e desvantagens:

1. **Uso de apenas a distância de Levenshtein**
   - **Vantagem**: Simplicidade na implementação, já que todas as entradas seriam corrigidas com base na proximidade.
   - **Desvantagem**: Correções repetidas ou erros tipográficos frequentes poderiam causar inconsistências, já que não há memória de correções passadas, além disso o grau de incerteza seria maior.
2. **Uso de Machine learning**
   - **Vantagem**: Um modelo treinado poderia identificar padrões de erros mais complexos, aprendendo a corrigir mesmo entradas fora do comum.

- **Desvantagem**: Exige grande quantidade de dados e tempo para treinar o modelo. Além disso, não é ideal para aplicações onde a precisão manual e revisões humanas são mais adequadas, além disso o seu uso neste projeto seria desnecessário e excessivamente complexo, considerando a natureza do problema.

3. **Criação de um dicionário manual**

   - **Vantagem**: Total controlo sobre as correções.
   - **Desvantagem**: Processo manual e inviável para grandes volumes de dados.

4.3 Solução escolhida

Em última análise, optou-se pela combinação do dicionário JSON com a métrica de distância de Levenshtein por diversos motivos:

- **Eficiência**: Correções rápidas para nomes já conhecidos, reduzindo a necessidade de cálculos repetidos.
- **Adaptabilidade**: O sistema está a ser melhorado continuamente à medida que novas associações são registadas no JSON.
- **Facilidade de revisão**: Entradas não resolvidas são organizadas para análise manual, garantindo precisão.

Esta abordagem equilibra eficiência automatizada e controlo manual, atendendo aos requisitos do projeto e garantindo um sistema flexível e preciso para validação de nomes.