UNIVERSIDADE DE AVEIRO

Técnicas de Mineração de Texto - 2

Licenciatura em Engenharia de Computadores e Informatica

UC: Projeto em Engenharia de Computadores e Informática

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

Professor Coordenador: Luís Filipe de Seabra Lopes

September 5, 2025

Contents

1 Abstract

Empresas que trabalham com grandes volumes de dados devem manter um padrão definido, de modo a garantir um acesso à informação rápido e eficaz. A proposta deste projeto surgiu de uma necessidade real da Administração dos Portos de Sines e do Algarve, S.A., que tem vindo a enfrentar cada vez mais dificuldades na gestão estatística dos dados face às diversas variações de nomes de empresas que surgem nos seus registos. Este projeto insere-se também no âmbito do projeto NEXUS[1].
O grupo descreve no presente relatório o trabalho realizado no sentido de desenvolver um método automático para a correção e padronização destes nomes, com base na associação de colunas entre tabelas, técnicas de comparação textual e de agrupamento não supervisionado e semi-supervisionado. O sistema recebe como entrada as tabelas de dados a corrigir e, opcionalmente, uma tabela de dados de referência com os nomes e informação padronizada de cada empresa.

2 Introdução

2.1 Contexto

Dado um conjunto de dados no qual grande parte da informação em cada linha é escrita de forma livre, isto é, sem restrições sobre o método de introdução de dados, é natural que ocorram inconsistências, erros de digitação ou falta de padronização. No contexto específico deste projeto, os conjuntos de dados dizem respeito às empresas que entram e saem do Porto de Sines [2].

Como os registos das empresas são efetuados manualmente, é comum que uma mesma entidade surja com variações no nome (por exemplo, diferenças de acentuação ou espaçamento irregular). Essa problemática estende-se também a outros campos, como números de identificação (NIF).

Partindo da existência de uma base de dados de referência contendo os nomes padronizados e a informação fidedigna de cada empresa, é possível proceder à correção automática de grande parte dos dados originais, reduzindo significativamente o esforço de correção manual.

Para identificar e agrupar nomes semelhantes, o sistema desenvolvido recorre a diversas métricas de comparação textual, tanto de similaridade quanto de distância. Entre as principais métricas utilizadas encontram-se a distância de Levenshtein, que quantifica o número mínimo de operações de edição necessárias para transformar uma cadeia de caracteres noutra; o índice de Jaccard, uma métrica de similaridade baseada na interseção e união de conjuntos de caracteres (sendo a distância de Jaccard definida como $1 - J$); e a similaridade de cosseno, geralmente aplicada a vetores de frequência ou representações vetoriais de texto.

Além disso, foi utilizado um modelo de linguagem para vetorização semântica das palavras, tornando a comparação mais robusta. Todas estas técnicas foram integradas num algoritmo de agrupamento (*clustering*), que agrupa variantes semelhantes de

[1] NEXUS: Pacto de Inovação – Transição Verde e Digital para Transportes, Logística e Mobilidade

[2] Todos os conjuntos de dados foram fornecidos pelo Porto de Sines sob um contrato de confidencialidade.

nomes e sugere a forma padronizada, seja esta proveniente da tabela de referência ou inferida com base nos dados disponíveis.

2.2 Objetivos

O objetivo do projeto desenvolvido incide na criação de um sistema que tem como finalidade principal corrigir e padronizar grandes bases de dados. É de realçar que a solução desenvolvida requer a existência dos campos de nome e número de identificação para o seu funcionamento, dado que estas foram as informações usadas no seu desenvolvimento.
Para que o sistema funcione podem ser introduzidos dois tipos de conjuntos de dados:

1. Um ou mais conjuntos de dados a corrigir (Obrigatório)
2. Um ou mais conjuntos de dados que incluam as informações padronizadas de cada empresa (Opcional)

Apesar do sistema desenvolvido ter a possibilidade de operar sem uma tabela de dados de referência, a sua existência leva a resultados melhores, explorados com maior detalhe em 6.

2.3 Enquadramento Académico

Este trabalho foi realizado ao longo do ano letivo 2024/2025, no âmbito da unidade curricular de PECI (Projeto em Engenharia de Computadores e Informática), sob supervisão do professor Luís Seabra Lopes e acompanhamento dos docentes responsáveis.

3 Requisitos

Nesta secção, é abordada a escolha dos *stakeholders* envolvidos no projeto, e são reconhecidos os requisitos e métodos de recolha dos mesmos, bem como a importância individual de cada um.

3.1 Análise dos *Stakeholders*

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

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

3.2 Métodos de Recolha de Requisitos

Após a definição dos *stakeholders*, foram levantados os requisitos relativos ao projeto. Para tal, o grupo fez uso de métodos diferentes:

1. Contacto direto com a empresa interessada, por intermediário do professor orientador
2. Reuniões entre os desenvolvedores
3. Discussão com os professores responsáveis pela Unidade Curricular

3.2.1 Contacto direto com a empresa interessada

A partir do contacto com o Porto de Sines, nomeadamente através de reuniões com o professor orientador, o grupo estabeleceu os requisitos mais importantes, que definem o objetivo do trabalho e que foram considerados fundamentais para o sucesso do projeto.

3.2.2 Reuniões entre os desenvolvedores

Através de reuniões internas, obteve-se uma perspetiva de como seria possível alcançar os requisitos mais importantes e definiram-se ainda requisitos secundários que o grupo sentiu que trariam mais valor ao projeto sem comprometer a realização dos requisitos principais.

3.2.3 *Feedback* dos professores responsáveis pela UC

Por último, as aulas da Unidade Curricular associada ao projeto, mais especificamente as aulas onde houve um contacto mais direto com os professores responsáveis, permitiram uma solidificação das ideias previamente discutidas em grupo e também trouxeram novas ideias e pontos de vista úteis ao desenvolvimento do trabalho.

3.3 Requisitos Definidos

Uma vez que explicados os métodos de levantamento de requisitos, apresentam-se de seguida os requisitos definidos em última análise, juntamente com a sua classificação (funcional ou não funcional. Adicionalmente, na tabela também se encontra a referência à completude do requisito aquando da entrega do projeto. Os requisitos encontram-se divididos em duas tabelas (1 e 2), de acordo com a sua prioridade no contexto do projeto. Os requisitos principais distinguem-se por serem fundamentais para o sucesso do projeto, enquanto que os requisitos secundários foram encarados como *extras* que dariam um valor maior ao projeto.

3.3.1 Requisitos Principais

3.3.2 Requisitos Secundários

Descrição do Requisito	Tipo	Atingido
Corrigir os dados	Funcional	Sim
Assegurar precisão na correção	Não Funcional	Sim
Adaptável (capaz de trabalhar com diferentes *datasets*)	Funcional	Sim
Reduzir significativamente o trabalho manual de correção posterior	Funcional	Sim

Table 1: Resumo dos Requisitos Principais

Descrição do Requisito	Tipo	Realizado
Baixa complexidade temporal	Não Funcional	Sim
Inserir novas entradas nos dados padronizados (quando surgem novas empresas)	Funcional	Não
Capacidade de sugerir entradas quando alguém insere uma entrada que se assemelhe a um campo padronizado	Funcional	Não
Ter uma *User Interface* de forma a poder interagir com o programa de forma fácil	Não Funcional	Sim (interface no terminal)

Table 2: Resumo dos Requisitos Secundários

4 *Background* e Trabalhos Relacionados

4.1 Soluções Existentes

Numa pesquisa inicial, o grupo teve a preocupação de procurar trabalhos e técnicas de limpeza de dados que fossem relevantes 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 frases e palavras relativamente curtas, preparada para lidar com grandes volumes de dados. O trabalho tem por base o *Symmetric Delete Algorithm*. Este algoritmo reduz a complexidade do problema ao gerar variações mínimas de cada frase, 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 chaves de um dicionário de correspondência, que tem como valor de cada chave a frase correta. Caso seja obtida mais do que uma correspondência, é então calculada a distância de Levenshtein da frase original ao valor da correspondência obtida, de modo a verificar qual a frase pretendida (sendo possível 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, menos notório 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 Aprendizagem Automática. 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.

4.2 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 atual projeto.
Uma limitação do **SymSpell Algorithm** passa pelo facto de estar projetado para lidar com palavras ou frases de dimensões reduzidas, que não é o caso com o tipo de dados a corrigir. Também o facto deste algoritmo ser projetado para lidar com pequenas variações previsíveis nas frases ou palavras o torna mais difícil de aplicar no contexto do projeto atual, visto que as variações que surgem dos nomes de empresas são muitas vezes constituídas por erros imprevisíveis e de grande dimensão, com muitas palavras a mais ou em falta.
Em relação ao **CPClean**, um dos seus maiores problemas é a escassez de documentação do projeto.
No entanto, ainda que relevantes, os projetos mencionados foram utilizados apenas como repositórios de técnicas de análise e comparação de palavras ou frases, como as distâncias de Levenshtein e a utilização de dicionários de correspondências, que proporcionou a redução dos tempos de resolução de campos de dados que já tenham sido resolvidos anteriormente.

5 Design e Soluções Viáveis

5.1 Design do Sistema

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

- **Tabelas a serem corrigidas**: Contém os nomes que necessitam de validação e padronização. (Obrigatório)
- **Tabelas referência (Standard)**: Conjunto de nomes previamente validados, utilizados como referência, caso as tabelas sejam introduzidas. (Opcional)

O sistema recebe como entrada a(s) tabela(s) mencionada(s). Os nomes das empresas são extraídos de cada tabela e processados com o objetivo de realizar uma limpeza, removendo ruído indesejável. Opcionalmente, o utilizador pode fornecer uma tabela de referência, a partir da qual é construída uma base de verdade (*Ground Truth*) para avaliar o desempenho do algoritmo de agrupamento. Esta base de verdade é formada agrupando os nomes das empresas que partilham o mesmo número de identificação na tabela de referência. Caso esta tabela não seja fornecida, não é criada qualquer base de verdade, e o agrupamento prossegue sem validação automática.
Os nomes a corrigir são convertidos em vetores semânticos utilizando o módulo Python

SentenceTransformers[3]. Estes vetores são normalizados e indexados com recurso à biblioteca **FAISS** [4] - explorada em detalhe em 5.4 - permitindo comparações rápidas e eficientes entre nomes com base na sua similaridade vetorial.
O fluxo do algoritmo de agrupamento é ilustrado na Figura 3 e segue os seguintes passos principais:

1. **Criação de um novo grupo**: Para cada nome ainda não associado, é criado inicialmente um novo grupo com esse nome.

2. **Procura de nomes semelhantes**: O sistema utiliza a biblioteca **FAISS** para identificar os 1000 nomes semânticamente mais semelhantes com base nos vetores indexados. Estes 1000 nomes podem originar dos nomes a analisar ou dos nomes já tidos no agrupamento. Para cada nome candidato, é calculada a similaridade com o nome a ser atualmente analisado.

3. **Atribuição a grupos**: Se a similaridade entre o nome atual e os candidatos for acima de um limiar definido, estes são agrupados ao grupo criado (mencionado no primeiro ponto). Caso o nome semelhante já pertença a um grupo originado a partir da base de verdade, o seu index é guardado, bem como o valor da sua similaridade, como um possível *best match*. No final da comparação com todos os nomes, o novo grupo criado vai extender o grupo do seu *best match* ou, caso não tenha sido encontrado um, vai ser anexado no final do agrupamento como um novo grupo distinto. É relevante mencionar que nos nomes inseridos no conjunto de dados sobre o qual se trabalhou, muitas das entradas referentes ao nome da empresa incluíam nomes pessoais (possivelmente por serem referentes a empresas familiares). Para isso, o grupo criou uma regra que reconhece se a entrada é um nome e, caso positivo, aumenta o limiar de similaridade de forma a que os nomes só entrem no mesmo grupo caso sejam realmente semelhantes. A necessidade para esta correção surge porque os vetores gerados para nomes de pessoas são sempre relativamente semelhantes, por serem semânticamente parecidos.

4. **Novo ciclo**: Os nomes que foram anexados são marcados como vistos para serem ignorados em futuras iterações, e isto repete-se para cada nome a ser analisado.

[3] `https://sbert.net/`
[4] `https://faiss.ai/index.html`

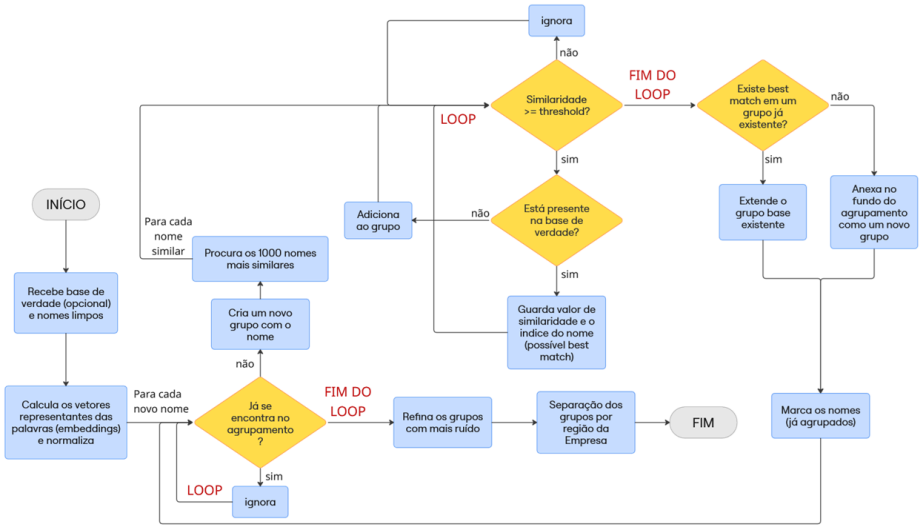

Figure 1: Fluxo do Sistema de Agrupamento

Pseudocodigo do Algoritmo

Entradas:

- `BASE`: lista de listas de nomes associados, ex. `[[N1a, N1b, ...], [N2a, N2b, ...]]`
- `AGRUPAR`: lista de nomes únicos a agrupar

Inicialização:

1. Criar `MAPA` ← dicionário {nome → índice do grupo em `BASE`}
2. `NOMES` ← todos os nomes presentes em `BASE` e `AGRUPAR`
3. `VECTORES` ← vetores semânticos de `NOMES` com *SentenceTransformer*
4. `GRUPOS` ← cópia de `BASE`
5. `AGRUPADOS` ← Verdadeiro para n em `NOMES` se n existe em `GRUPOS` caso contrário Falso

Para cada nome n em `NOMES`:

1. Se `AGRUPADOS[indice(Nomes,`n`)]` == Verdadeiro, continuar para o próximo nome
2. `NOVO` ← { n }
3. `CANDIDATOS` ← 1000 nomes mais semelhantes a n em `VECTORES`, retorna (índice, similaridade)
4. `maxMatch` ← 0; `bestMatch` ← `None`
5. Para cada (i, s) em `CANDIDATOS`:

a. Se $s <$ `threshold`, continuar para o próximo

b. Se `NOMES[`i`]` $\in$ `MAPA` (ou seja, está na base de verdade):

- Se $s >$ `maxMatch`, então:
  - `maxMatch` $\leftarrow s$
  - `bestMatch` $\leftarrow$ `MAPA[NOMES[`i`]]`

c. Caso contrário:

- Adicionar `NOMES[`i`]` a `NOVO`
- `AGRUPADOS[`i`]` $\leftarrow$ Verdadeiro

6. Se `bestMatch` == `None`:

   - Adicionar `NOVO` a `GRUPOS`

7. Caso contrário:

   - Adicionar `NOVO` ao grupo `GRUPOS[bestMatch]`

Pós-processamento:

- Refinamento de grupos com maior ruído
- Eventual separação adicional por região geográfica

Retorna: `GRUPOS`

5.2 Pós-processamento:

- **Deteção e remoção de ruído:**
  - Para os grupos com maior número de elementos, calcular a média de similaridade entre os seus membros.
  - Identificar elementos cuja similaridade com os restantes do grupo seja significativamente inferior à média.
  - Esses elementos são movidos para um novo grupo individual (tratados como possíveis outliers).
- **Separação por região geográfica:**
  - Para cada grupo, verificar se os nomes associados pertencem a diferentes países (inferência por nome).

5.3 Remoção de Ruído em Clusters

Para reduzir o ruído presente nalguns clusters resultantes da fase de agrupamento, aplicou-se um processo de *refinamento de grupos*. Este processo visa identificar nomes que, apesar de inicialmente agrupados juntos, apresentam uma similaridade significativamente inferior com os restantes elementos do cluster.

A metodologia baseia-se numa métrica de similaridade combinada, definida da seguinte forma:

- **Similaridade de Levenshtein Normalizada:** Calcula a distância de Levenshtein entre os nomes, após remoção de prefixos legais (e.g., "Sociedade", "Empresa") e termos geográficos (e.g., "Brasil", "Angola"). A similaridade é então dada por $1 - \frac{\text{distância}}{\text{comprimento máximo}}$.

- **Jaccard Ponderado:** Baseado no índice de Jaccard entre os tokens dos nomes. Tokens considerados genéricos (prefixos legais e regiões) são ignorados na interseção. A interseção ponderada considera apenas tokens distintos e relevantes.

- **Combinação Final:** A similaridade final entre dois nomes é uma média ponderada:

$$\text{similaridade}(n_1, n_2) = 0{,}7 \cdot \text{Levenshtein}(n_1, n_2) + 0{,}3 \cdot \text{Jaccard}(n_1, n_2)$$

O algoritmo percorre os elementos de um cluster inicial e reagrupa os nomes em subconjuntos onde todos os pares de nomes possuem uma similaridade combinada superior a 0,75. Caso um nome não atinja esse limiar com nenhum grupo existente, é colocado num novo grupo.
Este processo permite reduzir significativamente o ruído, isolando nomes que foram agrupados de forma incorreta devido a coincidências superficiais.

5.4 Biblioteca FAISS

FAISS (*Facebook AI Similarity Search*) é uma biblioteca desenvolvida pelo Facebook AI Research[5] que permite a realização de pesquisas eficientes de vetores em espaços de alta dimensão. Foi desenhada para escalar a tarefas de busca aproximada por similaridade, com foco em desempenho e precisão, sendo amplamente utilizada em aplicações de recuperação de informação, recomendação e agrupamento semântico.
No contexto deste sistema, **FAISS** é utilizada para identificar rapidamente nomes semanticamente semelhantes através das seguintes funcionalidades principais:

- **Normalização de vetores**: Após a geração dos vetores semânticos (através de *sentence embeddings* criados com *SentenceTransformers*), os vetores são normalizados com **FAISS** para que a métrica de similaridade baseada em produto interno (*inner product*) funcione de forma equivalente à distância de cosseno.

- **Indexação eficiente**: Os vetores são armazenados num índice **FAISS**, permitindo consultas extremamente rápidas mesmo com milhares de nomes.

- **Pesquisa por similaridade**: **FAISS** permite encontrar os k vetores mais próximos de um dado vetor de entrada, retornando os índices e os valores de similaridade correspondentes. Esta funcionalidade é essencial para agrupar nomes com base na proximidade semântica.

A utilização da biblioteca **FAISS** permitiu uma redução significativa no tempo de execução e uma maior escalabilidade do sistema, especialmente quando aplicado a volumes muito grandes de dados.

[5]Agora Meta Research

5.5 Soluções Alternativas

Durante a evolução do projeto, outras soluções foram consideradas e testadas, pelo que são aqui explorados os aspetos positivos e negativos levantados com o uso destas soluções:

1. **Distância de Levenshtein como única técnica**
    - **Vantagem**: Simplicidade na implementação, já que todas as entradas seriam corrigidas com base na proximidade.
    - **Desvantagem**: Não é considerada a semântica dos nomes, o que pode levar a agrupamentos incorretos. Além disso, erros comuns ou abreviações podem não ser corrigidos de forma adequada.
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**: Requer uma grande quantidade de dados curados e tempo de treino. Além disso, devido à escassez desses dados neste contexto, a utilização de *Machine Learning* seria menos eficaz do que abordagens baseadas em regras e similaridade textual.
3. **Criação de um dicionário manual**
    - **Vantagem**: Total controlo sobre as correções.
    - **Desvantagens**: Processo manual e inviável para grandes volumes de dados. Impossibilidade de prever todas as variações, dada a elevada imprevisibilidade.
4. **Uso de MinHash com Similaridade de Jaccard**
    - **Vantagem**: Permite calcular a similaridade entre nomes com base na sobreposição de substrings (shingles), sendo extremamente eficiente para detectar pequenas variações ortográficas.
    - **Desvantagem**: Pode não capturar bem relações semânticas entre palavras. A performance depende da escolha de parâmetros, como por exemplo o número de funções *hash*. Esta abordagem revelou ser substancialmente mais lenta que a abordagem final, com *SentenceTransformers* e **FAISS**.

5.6 Solução Final

A solução adotada baseia-se na utilização de algoritmos de agrupamento, para agrupar as entradas semelhantes no campo referente ao nome da empresa. Através desta abordagem, é possível identificar e consolidar variações do mesmo nome, mesmo quando contêm erros ortográficos, abreviações ou formatos distintos.
Para isso, é realizada inicialmente um pré-processamento dos nomes, onde cada entrada é normalizada com base em regras definidas - remoção de pontuação, passagem a letras minúsculas, entre outros. Após essa etapa, são extraídas representações vetoriais dos nomes, de modo a permitir a comparação mais eficaz entre nomes.

Posteriormente, é aplicado um algoritmo de agrupamento com um limiar de semelhança adequado, de forma a garantir que apenas nomes suficientemente semelhantes são agrupados. Cada grupo resultante sem entrada na tabela de referência é então associado a um nome tido como nome representante do grupo, selecionado com base na entrada mais próxima do centro do próprio grupo.
Esta abordagem tem como vantagem principal a sua capacidade de lidar com variações imprevisíveis nos dados, sem depender de um dicionário fixo de nomes. Além disso, permite escalar a limpeza a conjuntos de dados de grandes dimensões, adaptando-se dinamicamente às entradas presentes.
Todo este sistema funciona em conjunto com uma interface de terminal simples que permite a escolha dos ficheiros a serem introduzidos, bem como a visualização do progresso do programa. A interface dá ainda a escolher ao utilizador se este pretende receber informação relativa à avaliação da precisão da correção realizada sobre os dados.

6 Testes e Resultados

6.1 Avaliação dos Dados e Mapeamento Inicial

Durante a fase inicial, foi criada uma correspondência entre os nomes presentes nas tabelas fornecidas e os respetivos números de identificação. Nomes considerados variantes de um nome padrão foram identificados através de regras de similaridade e pré-processamento, nomeadamente através da limpeza e comparação por distância de Levenshtein. Com esta abordagem foi possível encontrar um nome padrão em cerca de 65% dos casos. No entanto, falta de numeros de identificação e nomes mais complexos ou com um maior número de variações não eram corrigidos corretamente.

6.2 Criação e Pós-processamento do Agrupamento

Ao adicionar um novo nome ao processo de agrupamento, o seguinte procedimento é adotado:

1. O nome é inicialmente inserido num grupo temporário, contendo apenas esse nome.

2. Em seguida, são identificados os 1000 nomes mais semelhantes com base na métrica de similaridade. Apenas os nomes com pontuação superior a um limiar de pontuação correspondente a 0.81 são considerados para inclusão no grupo temporário.

3. A inclusão desses nomes ao grupo temporário ocorre com as seguintes exceções:

    - Caso o nome candidato já tenha sido adicionado anteriormente a um grupo, ele é ignorado.
    - Caso o nome candidato pertença a um grupo originado a partir da base de verdade, também não é adicionado ao grupo temporário. Em vez disso, o índice desse nome e a sua respetiva similaridade são armazenados numa variável denominada *best match*.

4. No fim do processo:

    - Caso um *best match* não tenha sido identificado, o grupo temporário é anexado como um novo grupo ao agrupamento existente.
    - Caso um *best match* tenha sido identificado, o grupo temporário não é adicionado como um novo agrupamento, mas sim utilizado para expandir o grupo base associado ao *best match*.

5. Pos processamento: Explicado anteriomente no 5.2

O limiar definido foi escolhido de acordo com um conjunto de testes realizados, dos quais se concluiu que este valor dava os melhores resultados de **F1-Score**, **Recall** e Precisão.

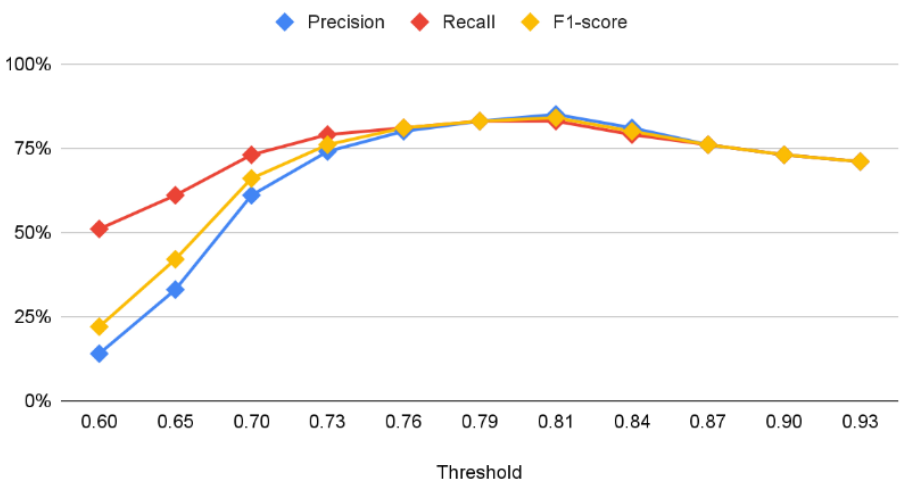

Figure 2: Escolha do limiar

Métricas utilizadas para avaliar a qualidade dos agrupamentos

- **Similaridade Intra-cluster**: considera a similaridade de um dado nome com os nomes pertencentes ao mesmo grupo.
- **Similaridade Inter-cluster**: considera a similaridade de um dado nome com os nomes de outros grupo.

6.3 Avaliação de Precisão do Agrupamento

Ao longo do presente relatório é destacado que o sistema desenvolvido permite opcionalmente a introdução de uma tabela de referência (*standard*). Se uma tabela de referência for fornecida, a qualidade dos grupos de nomes aumenta, uma vez que os grupos iniciais são feitos com base nessa tabela. Ao utilizar 50% desses dados, escolhidos aleatoriamente, é possível avaliar a qualidade do algoritmo verificando se os grupos criados são semelhantes aos restantes 50%.

- **Versão 1**: Agrupamento sem grupos base
- **Versão 2**: Agrupamento com 50% dos grupos da base de verdade como base

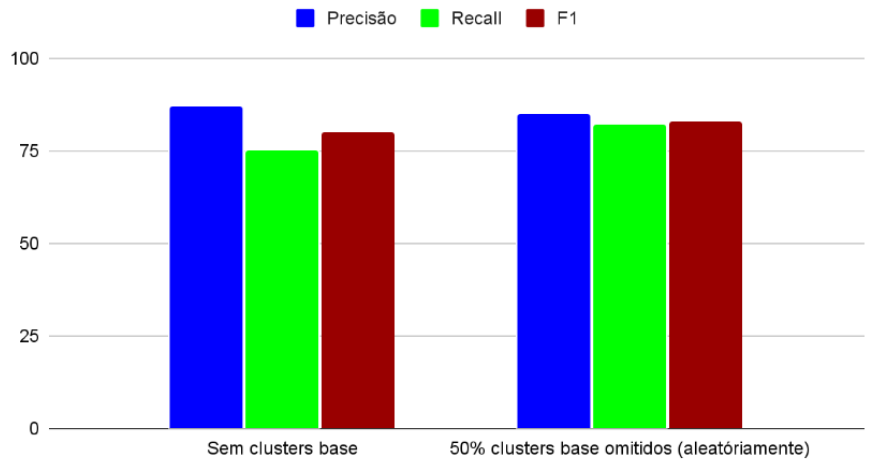

Figure 3: Relevância da existência de grupos com origem na base de verdade no resultado final

6.4 Criação do Mapa Final de Sinónimos

Com base nos grupos, é criado automaticamente um mapa final de sinónimos que associa cada nome presente nas tabelas ao nome padrão correspondente, com base nos grupos obtidos. O mapa resultante gera uma nova coluna nas tabelas que associa a cada nome o seu nome padrão, de acordo com os grupos criados.

6.5 Comparação de resultados com o método de MinHash

Como referido anteriormente, na Secção 5.5, a utilização da biblioteca **FAISS** resultou numa solução mais eficiente. Para além de uma redução significativa no tempo de execução, os resultados obtidos também apresentaram melhores resultados conforme as métricas utilizadas na avaliação do sistema.

MinHash	Sem base de verdade	50% base de verdade	100% base de verdade
Precision	0.05	0.06	0.77
Recall	0.93	0.92	0.91
F1 Score	0.09	0.11	0.83

Figure 4: Resultados MinHash

ST+Faiss	Sem base de verdade	50% base de verdade
Precision	0.87	0.85
Recall	0.75	0.82
F1 Score	0.80	0.83

Figure 5: Resultados ST+FAISS

7 Conclusão

Em última análise, o grupo responsável pelo desenvolvimento do projeto, bem como o professor supervisor do mesmo, consideraram que o sistema desenvolvido cumpre os requisitos definidos na sua quase totalidade.
Ao longo do desenvolvimento foram testadas várias soluções e, por fim, o grupo optou por uma solução que permite uma elevada percentagem de correção, bem como uma complexidade temporal reduzida. O programa desenvolvido é flexível, nomeadamente pelo facto de ser totalmente opcional a introdução de uma tabela de referência.
Posto isto, a solução desenvolvida pode ser facilmente adaptada a contextos reais, necessitando de eventuais ajustes dependendo do modo de utilização pretendido. Além de cumprir os requisitos técnicos, o desenvolvimento do projeto proporcionou ao grupo um aprofundamento nas áreas do âmbito, especialmente a área de ciências de dados.

8 Bibliografia