UNIVERSIDADE DE LISBOA
FACULDADE DE CIÊNCIAS
DEPARTAMENTO DE INFORMÁTICA

Soluções de Aprendizagem Automática e Inteligência Artificial para Extração e Reconhecimento de Entidades em Arquivos Históricos

Diogo Francisco Justino Mano

Mestrado em Engenharia Informática

Trabalho de Projeto orientado por:
Profª. Doutora Márcia Cristina Afonso Barros

2023

Agradecimentos

Em primeiro lugar, quero agradecer à minha orientadora Prof. Márcia Barros, que desde cedo apostou e confiou em mim, guiando-me neste projeto. Ao meu orientador dentro da empresa Adriano Campos por toda a ajuda e disponibilidade ao longo desta caminhada. Um agradecimento especial à Caixa Mágica e ao CEO José Rodrigues que proporcionaram a integração, investigação e desenvolvimento neste projeto desafiador do qual consegui elaborar este projeto tese.

Aos meus pais, pelas inúmeras oportunidades que me presentearam e por serem os conselheiros incondicionais de sempre. Por me terem apoiado em todos os dias e noites de trabalho árduo para conseguir finalizar com o melhor aproveitamento possível este projeto.

Por fim um obrigado caloroso a todos os familiares e amigos que acompanharam todo este processo.

"With great power comes great responsibility." - Stan Lee

Resumo

Neste projeto, o principal objetivo é facilitar o acesso à plataforma do Digitarq, um sistema que contribui para a descrição e gestão das atividades arquivísticas da Direção-Geral do Livro, dos Arquivos e das Bibliotecas (DGLAB).

O projeto está dividido em duas fases distintas. Na primeira fase, pretende-se realizar a migração da base de dados antiga do Digitarq para um novo conjunto de bases de dados. Na segunda fase, o objetivo é criar uma API que permita conectar o novo conjunto de bases de dados à aplicação web do Digitarq.

O primeiro tópico abordado para melhorar o desempenho deste sistema será a migração de uma base de dados relacional que contém metadados de documentos do Digitarq para uma nova base de dados de grafos. Para isso, será implementado um modelo de extração de entidades (NER) para etiquetar informações úteis presentes nesses metadados. O desenvolvimento deste modelo terá como base modelos neuronais pré-treinados de dois tipos de arquiteturas que serão avaliados e comparados: o BERT e o T5. Além disso, as entidades reconhecidas serão importadas para uma base de dados de grafos, seguindo o modelo conceptual do CIDOC-CRM. Simultaneamente, para responder de forma eficiente a pedidos de pesquisa de documentos, os metadados dos documentos das bases de dados antigas serão migrados para uma base de dados Elasticsearch.

O segundo tópico abordado será o desenvolvimento de uma API que permita a criação dos pedidos necessários para a conexão entre as novas bases de dados (grafos e Elasticsearch) e a aplicação web do Digitarq. Neste tópico, todos os endpoints necessários para o funcionamento da aplicação web serão construídos.

Palavras-chave: extração de entidades, BERT, T5, base de dados de grafos, CIDOC-CRM

Abstract

In this project, the primary objective is to enhance access to the Digitarq platform, a system that aids in the description and management of archival activities within the Directorate-General for Books, Archives, and Libraries (DGLAB).

The project is divided into two distinct phases. The first phase aims to migrate the old Digitarq database to a new set of databases. In the second phase, the goal is to create an API that enables the connection between the new databases and the Digitarq web application.

The initial focus to improve the system's performance will be the migration of a relational database containing Digitarq document metadata to a new graph database. To achieve this, an Entity Recognition Model (NER) will be implemented to label valuable information within these metadata. The development of this model will be based on pre-trained neural models of two architectural types, namely BERT and T5, which will be evaluated and compared. Furthermore, the recognized entities will be imported into a graph database, following the CIDOC-CRM conceptual model. Concurrently, to efficiently respond to document search queries, the metadata from the old databases will be migrated to an Elasticsearch database.

The second aspect to be addressed is the development of an API that allows for the creation of necessary requests to connect the new databases (graph and Elasticsearch) with the Digitarq web application. In this context, all the required endpoints for the proper functioning of the web application will be constructed.

Keywords: Entity Extraction, BERT, T5, graph database, CIDOC-CRM

Conteúdo

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Nesta secção será feita uma introdução no qual se apresentam o contexto do trabalho, se resume o trabalho desenvolvido, se identificam as contribuições deste e se apresenta a estrutura do próprio relatório. Também será mencionado sucintamente o enquadramento institucional em que o trabalho decorreu.

1.1 Motivação

A Caixa Mágica é uma entidade reconhecida e de créditos firmados no mercado nacional. Para cada projeto escolhe a melhor estratégia e adapta a melhor tecnologia de forma que seja precisamente aquilo que o cliente pretende, apresentando uma proposta flexível e compatível com as suas necessidades.

A Direção-Geral do Livro, dos Arquivos e das Bibliotecas (DGLAB) [6] desenvolveu diversos sistemas de informação destinados a gerir informação do universo dos arquivos. Destes, o Digitarq assume-se como central na área de atuação da DGLAB por se destinar a gerir dados relativos ao acervo dos arquivos da rede. O objetivo do projeto da DGLAB é melhorar este sistema de informação, sistema que contribuí na descrição e gestão das atividades arquivísticas.

A melhoria principal que vem de encontro ao propósito desta tese incide na migração dos dados antigos do Digitarq e criação de uma API que consiga dar suporte à aplicação web do Digitarq. O problema principal concentra-se no processo de migração, no qual será desenvolvido um modelo neuronal capaz de reconhecer entidades. A dificuldade surge devido à complexidade de criar um modelo que possa identificar entidades em língua portuguesa, com foco em três tipos específicos de entidades: atores, organizações e locais.

A motivação desta tese prende-se com a capacidade de investigar e implementar soluções de machine learning que possibilitem a realização de um dos principais processos desta área que é o reconhecimento de entidades (NER). Uma tarefa desafiante que implica a utilização de ferramentas recentes e a aplicação de conhecimentos dentro da área da inteligência artificial.

A motivação para a investigação e desenvolvimento deste projeto deve-se à dimensão e importância do sistema Digitarq, sistema que facilita a pesquisa arquivística para muitos utilizadores. Sistema que necessita de sofrer desenvolvimentos ao longo dos anos para se manter atual e de

acordo com a linha temporal onde se encontra.

A necessidade de reformularmos este sistema é o motivo que faz com que ambas as entidades tanto a DGLAB como a Caixa Mágica queiram avançar com este projeto.

1.2 Objectivos

O sistema Digitarq [20] tem sido desenvolvido sucessivamente ao longo do tempo, baseado em normas internacionais e orientações subsequentes produzidas pela DGLAB que sofreram alterações durante o seu processo de utilização e desenvolvimento. Simultaneamente, o Digitarq procurou adaptar-se às necessidades decorrentes da prática de gestão de informação na rede de arquivos. Isto condicionou opções de recolha e processamento de informação e resultou numa arquitetura de sistema excessivamente dependente da utilização da aplicação em áreas específicas de atuação da DGLAB. O Digitarq utiliza tecnologia proprietária, sendo um dos objetivos da DGLAB a sua substituição por uma nova aplicação baseada, preferencialmente, em software aberto. A DGLAB pretende substituir o Digitarq por um novo sistema a desenvolver baseado num outro paradigma de descrição.

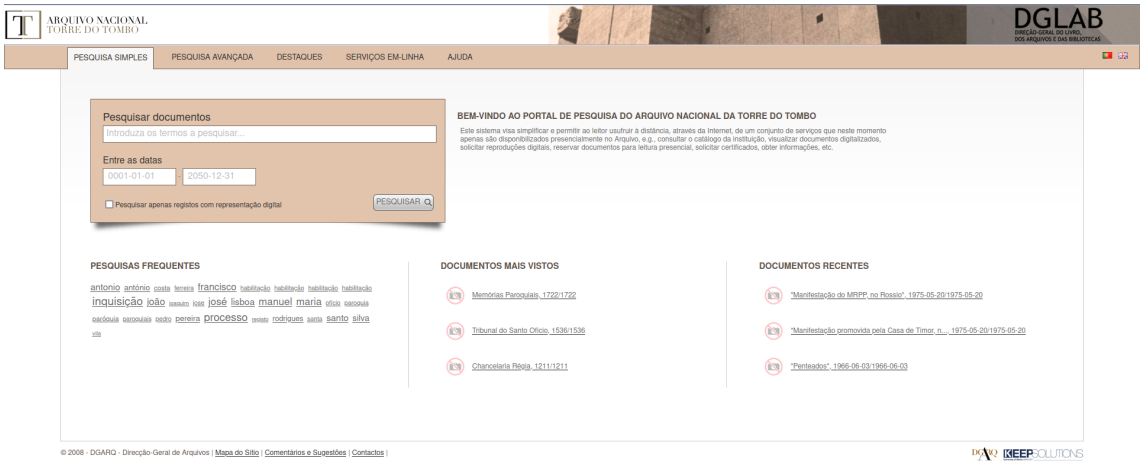

Figura 1.1: Página inicial do Digitarq atual

Num mundo cada vez mais diversificado ao nível de machine learning no desenvolvimento de aplicações web, a DGLAB propôs em primeira instância a extração dos dados antigos do Digitarq que se encontram em bases de dados relacionais.

Em segunda instância o desenvolvimento de um modelo neuronal que possibilite a extração de entidades para que posteriormente seja feita a importação destes dados numa base de dados de grafos com o objetivo de materializar as entidades extraídas da base de dados antiga. Para além da importação de dados numa base de dados de grafos também é necessário que seja feita a importação numa base de dados que facilite e seja eficiente para pedidos de pesquisa.

Por fim, a construção de uma API que permita fazer a ligação entre as novas bases de dados

criadas com a aplicação web.

Este projeto, realizado no âmbito do Mestrado em Engenharia Informática da Faculdade de Ciências da Universidade de Lisboa em conjunto com a Caixa Mágica Software, ambiciona através de investigação e posterior implementação responder ao problema de capacidade de extração de entidades a partir de cadeias de texto. Relativamente a este problema, o principal objetivo é o desenvolvimento de um modelo neuronal treinado para a realização da tarefa de reconhecimento de entidades (NER), de modo a conseguirmos extrair da melhor forma informação semântica útil no processo de migração de dados.

Em suma, o principal objetivo deste projeto é facilitar o acesso à plataforma do Digitarq de modo que este permita ao utilizar uma maior eficiência e rapidez devido aos milhões de documentos que se encontram divididos pelos diferentes arquivos distribuídos pelos dezoito distritos de Portugal Continental.

1.3 Contribuições

As principais contribuições do projeto tese são a renovação do sistema DIGITARQ, que conta com a realização de diferentes tarefas, tais como, numa primeira fase a exportação e análise dos dados exportados das bases de dados antigas do Digitarq, para de seguida proceder ao desenvolvimento do modelo neuronal que possibilite a extração de determinadas entidades consideradas as mais relevantes de identificar (pessoas (PER), organizações (ORG) e lugares (LOC)).

No momento de identificação de entidades a contribuição da tese prende-se apenas com a identificação de entidades do tipo localização, onde será feito o reconhecimento das mesmas, futura avaliação e desambiguação para definir quais as que fazem sentido extrair alguma informação semântica, de modo a minimizar o possível conteúdo sem relevância extraído.

De seguida, a construção de uma API e dos respetivos endpoints que permita conceber a ligação entre as bases de dados e a aplicação web do Digitarq.

A contribuição desta tese incidirá no desenvolvimento da nova plataforma do DIGITARQ que permitirá aos utilizadores do mesmo um sistema renovado que irá facilitar o acesso a documentos e arquivos presentes neste sistema.

1.4 Estrutura do documento

O projeto tese está organizada em cinco capítulos principais. O primeiro é a introdução já apresentada, que serve como uma introdução a vários temas discutidos mais adiante nas seções seguintes.

O segundo capítulo está focado em introduzir o leitor para a componente teórica utilizada no desenvolvimento do projeto, seja conceitos teóricos ou tecnologias utilizadas. O terceiro capítulo gira em torno de referir trabalhos realizados por outros autores relacionados com este projeto tese.

No quarto capítulo é feita uma análise e introdução aos dados que serão utilizados na implementação do projeto. O quinto capítulo refere todos os detalhes da metodologia. No sexto capítulo são introduzidos os resultados e discussão dos mesmos após implementação e realização de testes do

projeto.

Por fim, o sétimo capítulo onde são referidas as considerações finais da tese para destacar as principais descobertas, realizações do presente trabalho, limitações e trabalho futuro.

Relativamente ao planeamento inicial do projeto, que estava divido em cinco fases, ocorreram algumas alterações:

- **Fase Preparatória**: Fase introdutória onde definimos tecnologias e soluções para o problema apresentado. Cumpri com o plano inicial, realizado no tempo definido outubro de 2022.
- **Desenho, arquitetura e implementação**: Fase onde desenhamos a arquitetura do projeto e procedemos à implementação do mesmo. O objetivo do planeamento inicial não foi cumprido ocupando ao invés de novembro de 2022 a janeiro de 2023 prolongou-se até março de 2023.
- **Avaliação e melhoria** Fase onde foi feita a avaliação e melhoria da implementação do projeto. Demorou os mesmos dois meses definidos mas foi realizado em meses diferentes dos definidos inicialmente, abril a maio de 2023.
- **Testes, avaliação e afinações** Fase onde após implementação do projeto procedemos a testes e avaliações para possíveis afinações. Demorou os mesmos dois meses definidos mas foi realizado em meses diferentes dos definidos inicialmente, maio a junho de 2023.
- **Elaboração do relatório final**: Fase na qual se procedeu à escrita do relatório final. O plano inicial não foi cumprido aumentando o prazo desta fase para três meses ao invés de ocupar apenas o mês de junho de 2023 ocupou de junho a setembro de 2023.

Capítulo 2

Contexto

Nesta secção introduzimos as tecnologias e toda a componente teórica relevante e utilizada durante a execução do projeto.

2.1 Aprendizagem Automática

A aprendizagem automática [29], ou machine learning, é uma área da inteligência artificial que se concentra em desenvolver algoritmos e modelos estatísticos que permitem que os computadores aprendam com exemplos e dados. Em vez de programar explicitamente as regras, os computadores são treinados para reconhecer padrões e fazer previsões ou tomar decisões baseadas nesses padrões.

A aplicação prática da aprendizagem automática [32] é ampla, incluindo reconhecimento de voz, identificação de imagens, tradução automática, recomendação de produtos, análise de dados de sensores e detecção de fraudes financeiras. Os modelos de aprendizagem automática são classificados em três tipos principais: supervisionados, não supervisionados e de reforço, dependendo do tipo de dados de treino e do objetivo do modelo.

Apesar dos avanços recentes, a aprendizagem automática ainda enfrenta desafios como a privacidade dos dados, a interoperabilidade dos modelos e a equidade na tomada de decisões automatizadas.

Existem três tipos de aprendizagem automática [32]:

- **Aprendizagem supervisionada:** Neste tipo de aprendizagem, o modelo é treinado com exemplos rotulados, ou seja, dados que já possuem uma resposta correta. O objetivo do modelo é aprender a mapear as entradas para as saídas corretas. Por exemplo, um modelo pode ser treinado para reconhecer imagens de gatos e cachorros. A aprendizagem supervisionada é comumente usada em tarefas como classificação, regressão e previsão.

- **Aprendizagem não supervisionada:** Neste tipo de aprendizagem, o modelo é treinado com exemplos não rotulados. O objetivo do modelo é encontrar padrões ou estruturas nos dados que ajudem a compreender melhor o conjunto de dados. Por exemplo, um modelo pode ser treinado para encontrar grupos de produtos semelhantes num conjunto de dados

de compras. A aprendizagem não supervisionada é comumente usada em tarefas como agrupamento, redução de dimensionalidade e análise de anomalias.

- **Aprendizagem de reforço:** Neste tipo de aprendizagem, o modelo aprende a tomar decisões com base no feedback de um ambiente dinâmico. O modelo é treinado para maximizar uma recompensa ao tomar uma ação num estado específico do ambiente. Por exemplo, um modelo pode ser treinado para jogar xadrez, onde cada jogada é uma ação e a recompensa é ganhar ou perder o jogo. A aprendizagem de reforço é comumente usada em tarefas como jogos, robótica e controlo de processos industriais.

2.1.1 Modelos neuronais

Um modelo neuronal [22], também conhecido como modelo de rede neuronal, é uma representação computacional de um sistema inspirado no funcionamento do cérebro humano. É composto por unidades interconectadas chamadas neurónios artificiais ou unidades de processamento, que são organizadas em camadas e usadas para processar informações e realizar tarefas específicas de aprendizagem automática.

Esses modelos são projetados para aprender padrões e relações complexas nos dados por meio de um processo chamado treino. Durante o treino, o modelo ajusta os pesos e as conexões entre os neurónios com base nos dados de entrada e nos rótulos correspondentes (no caso de tarefas supervisionadas).

Cada tipo de modelo neuronal tem sua própria arquitetura e características específicas, mas todos partilham o princípio básico de usar unidades de processamento interconectadas para aprender e realizar tarefas complexas de aprendizagem automática, como classificação, regressão, reconhecimento de padrões, tradução de idiomas, geração de texto, entre outros.

2.1.2 Treino

O treino de um modelo neuronal é o processo pelo qual o modelo aprende a realizar uma tarefa específica por meio da exposição a um conjunto de dados anotados. O objetivo do treino é ajustar os parâmetros do modelo de forma que ele seja capaz de generalizar padrões e fazer previsões precisas para novos exemplos não vistos anteriormente.

O processo de treino é iterativo e pode levar várias épocas até que o modelo alcance um desempenho satisfatório. Um bom treino é fundamental para obter modelos neuronais com capacidade de generalização e resultados precisos para novos dados.

2.1.3 Inferência

A inferência em um modelo neuronal refere-se ao processo de usar um modelo treinado para fazer previsões ou tomar decisões em novos exemplos de dados. É a etapa em que o modelo é colocado em uso após ter sido treinado com um determinado conjunto de dados.

Durante a fase de treino, um modelo neuronal é exposto a um conjunto de dados anotados, onde aprende a identificar padrões e relações nos dados de entrada para realizar uma tarefa específica, como classificação de imagens, tradução de texto ou geração de sequências.

Após o treino, o modelo é capaz de generalizar o conhecimento adquirido para exemplos de dados não vistos anteriormente. Durante a inferência, o modelo recebe esses novos exemplos como entrada e produz uma saída ou previsão correspondente.

2.1.4 Métricas de avaliação

As métricas nos modelos neuronais são medidas utilizadas para avaliar o desempenho e a qualidade dos modelos durante o **treino** e a **inferência**. Fornecem informações quantitativas sobre como o modelo está a realizar as suas tarefas e podem ajudar a comparar diferentes modelos, ajustar hiperparâmetros e monitorizar o progresso do treino.

As métricas podem variar dependendo da tarefa específica do modelo. Aqui estão alguns exemplos comuns de métricas usadas em modelos neuronais:

- **Precisão (accuracy)**: É a proporção de exemplos corretamente classificados em relação ao total de exemplos. É frequentemente utilizada em problemas de classificação.

$$accuracy = (VP + VN)/(VP + VN + FP + FN)$$

$$VP(verdadeirospositivos, VN(verdadeirosnegativos)$$

$$FP(falsospositvos), FN(falsosnegativos)$$

- **Precisão (precision)**: É a proporção de exemplos positivos verdadeiros (verdadeiros positivos) em relação à soma dos exemplos positivos verdadeiros e dos falsos positivos. É útil quando o foco é minimizar os falsos positivos.

$$precision = VP/(VP + FP)$$

- **Revocação (recall)**: Também conhecida como taxa de verdadeiros positivos ou sensibilidade, é a proporção de exemplos positivos verdadeiros em relação à soma dos exemplos positivos verdadeiros e dos falsos negativos. É relevante quando é importante evitar falsos negativos.

$$recall = VP/(VP + FN)$$

- **Classificação F1 (f1-score)**: É a média harmônica da precision e do recall. É uma métrica que combina as duas métricas anteriores e é útil quando se deseja considerar tanto a precision quanto o recall.

$$recall = (2 * precision * recall)/(precision + recall)$$

- **Perda (loss)**: É uma métrica que quantifica o quão distante à saída do modelo está do valor desejado durante o treino. O objetivo é minimizar a perda para melhorar o desempenho do modelo.
- **Mean Squared Error - MSE**: É uma métrica comumente usada em problemas de regressão que mede a média dos quadrados das diferenças entre as previsões do modelo e os valores reais. Quanto menor o MSE, melhor o desempenho do modelo.

Essas são apenas algumas das métricas comumente utilizadas em modelos neuronais. A escolha das métricas adequadas depende do tipo de tarefa, dos dados e dos objetivos específicos do projeto. É importante selecionar as métricas corretas para obter uma avaliação precisa e informativa do desempenho do modelo.

2.2 Pré processamento de dados

O pré-processamento de dados é uma etapa essencial em modelos de reconhecimento de entidades, como processamento de linguagem natural (NLP) ou análise de texto, como neste artigo [28]. Existem várias razões pelas quais o pré-processamento é necessário antes de realizar inferências num modelo de reconhecimento de entidades. Alguns dos motivos mais comuns são:

Limpeza de dados: Os dados brutos geralmente contêm informações irrelevantes, como caracteres especiais, pontuação, espaços extras, números, entre outros. O pré-processamento permite remover ou filtrar essas informações desnecessárias, limpando os dados e melhorando a qualidade da entrada.

Normalização de texto: Os dados brutos podem conter diferentes variações de palavras ou frases com a mesma semântica, como palavras escritas em maiúsculas ou minúsculas, palavras com prefixos ou sufixos diferentes (por exemplo, ”casa”e ”casas”), erros de escrita, entre outros. O pré-processamento pode normalizar o texto, convertendo todas as palavras em minúsculas, removendo sufixos ou prefixos, corrigindo erros de digitação, para que o modelo possa identificar corretamente as entidades independentemente da variação na entrada.

Tokenização: O pré-processamento também envolve a divisão do texto em unidades menores, chamadas de tokens, como palavras individuais ou subpalavras. Isso ajuda a transformar o texto numa sequência estruturada que o modelo pode entender. A tokenização é importante para o reconhecimento de entidades, pois permite identificar os limites das palavras e as relações entre elas.

Remoção de stop words: Stop words são palavras muito comuns, como ”a”, ”o”, ”e”, ”em”, que geralmente não contribuem significativamente para a identificação de entidades. O pré-processamento pode envolver a remoção dessas stop words, reduzindo o ruído e a complexidade do modelo e melhorando o desempenho.

Tratamento de dados desequilibrados: Em alguns casos, os dados podem estar desequilibrados em termos de distribuição de classes de entidades. Por exemplo, pode haver um número muito maior de exemplos de uma classe em comparação com outras. O pré-processamento pode incluir

técnicas de balanceamento de dados, como oversampling (aumento da quantidade de exemplos da classe minoritária) ou undersampling (redução da quantidade de exemplos da classe majoritária), para melhorar a precisão e a capacidade do modelo de identificar entidades de todas as classes de forma equilibrada.

Esses são apenas alguns exemplos dos motivos pelos quais o pré-processamento de dados é necessário antes de fazer inferência num modelo de reconhecimento de entidades. Cada caso pode ter requisitos específicos de pré-processamento, dependendo do domínio, dos dados e dos objetivos do modelo.

2.3 Clustering

Clustering, também conhecido como agrupamento, é uma técnica de aprendizagem automática não supervisionada que envolve a organização de dados não anotados em grupos ou clusters, com base nas suas características e similaridades. O objetivo do clustering é encontrar padrões intrínsecos nos dados e agrupar objetos similares em conjunto, enquanto objetos diferentes são separados em clusters distintos.

Existem várias abordagens para realizar o clustering, sendo os métodos mais comuns o k-means, o DBSCAN (Density-Based Spatial Clustering of Applications with Noise) e o hierarchical clustering (agrupamento hierárquico).

Este artigo [34] aborda o uso da análise de silhuetas para avaliação de desempenho em aplicações de aprendizagem automática, com foco em técnicas de clustering. O estudo destaca a importância do clustering de objetos com base nas semelhanças, é uma tarefa comum em aplicações de aprendizagem automática. Muitos métodos de clustering foram desenvolvidos, entre eles, os métodos baseados em k-means têm sido amplamente utilizados e várias extensões foram desenvolvidas para melhorar o método original de clustering k-means, como k-means ++ e kernel k-means.

- **K-means**:

  No k-means, um dos algoritmos de clustering mais populares, é necessário definir previamente o número de clusters desejado (k). O algoritmo então atribui aleatoriamente k centroides iniciais e itera até convergir numa solução final. Durante cada iteração, os pontos de dados são atribuídos ao centroide mais próximo e os centroides são atualizados com base na média dos pontos de dados atribuídos a eles. O processo continua até que não ocorram mais alterações significativas nos centroides ou que o número máximo de iterações seja alcançado.

- **DBSCAN**:

  O DBSCAN, por sua vez, é um algoritmo baseado em densidade que agrupa pontos de dados em áreas densas, considerando a proximidade e a densidade dos pontos. Pode identificar clusters de forma mais flexível do que o k-means, pois não requer a especificação prévia do número de clusters. O DBSCAN classifica os pontos de dados como núcleo, borda ou ruído, e constrói clusters conectando pontos de dados densos.

- **Agrupamento hierárquico**:

  O agrupamento hierárquico é outra técnica comum, que cria uma hierarquia de clusters. Pode ser dividido em dois tipos principais: agrupamento aglomerativo (bottom-up) e agrupamento divisivo (top-down). No agrupamento aglomerativo, cada ponto de dado começa como um cluster individual e, em cada iteração, os clusters mais próximos são agrupados num único cluster maior. No agrupamento divisivo, todos os pontos começam como um único cluster e, em cada iteração, o cluster é dividido em subclusters menores.

O clustering tem várias aplicações em diferentes áreas, como segmentação de mercado, análise de redes sociais, detecção de anomalias, bioinformática, processamento de imagens e muito mais. É uma técnica útil para explorar e entender a estrutura dos dados, identificar padrões e tomar decisões com base nas características dos grupos formados.

2.4 Scikit-learn

A biblioteca Scikit-learn do python é utilizada devido à simplicidade e variedade de ferramentas que tem disponíveis para análise de dados. Esta biblioteca contêm duas ferramentas principais que são utilizadas regularmente para o clustering de dados: Kmeans 2.3 e TfidfVectorizer.

2.4.1 *TfidfVectorizer*

O TfidfVectorizer é uma classe da biblioteca scikit-learn em Python. É usado para converter uma coleção de documentos de texto numa matriz numérica representando as frequências de termos inversa (TF-IDF) de cada termo nos documentos.

O termo ”TF-IDF”significa Frequência do Termo-Inverso da Frequência do Documento. Essa medida estatística é normalmente utilizada para avaliar a importância de uma palavra num documento dentro de uma coleção de documentos. O objetivo do TF-IDF é destacar as palavras-chave ou termos mais relevantes em documentos, atribuindo um peso mais alto a eles.

O processo de vetorização usando o TfidfVectorizer envolve os seguintes passos:

- **Tokenização:** Os documentos são divididos em tokens (palavras ou termos) individuais.
- **Construção do vocabulário:** O TfidfVectorizer constrói um vocabulário a partir dos tokens únicos encontrados em todos os documentos. A cada token é atribuído a um índice no vocabulário.
- **Cálculo do TF-IDF:** Para cada documento, o TfidfVectorizer calcula a frequência do termo (TF) para cada token e a frequência inversa do documento (IDF) para cada token no vocabulário. O TF mede a importância de um termo num documento específico, enquanto o IDF mede a importância geral de um termo em toda a coleção de documentos.

- **Vetorização:** Os valores de TF-IDF são combinados numa matriz numérica, onde cada documento é representado por um vetor de TF-IDF. Cada elemento no vetor corresponde a um token no vocabulário e representa o peso TF-IDF para esse token no documento.

O TfidfVectorizer também suporta várias opções de pré-processamento, como remoção de stopwords, normalização, filtragem de tokens com base em padrões, entre outros. Ele fornece uma interface fácil de usar para extrair características de texto e é amplamente utilizado em tarefas de processamento de linguagem natural, como classificação de texto, clustering e recuperação de informações.

2.5 Hugging Face

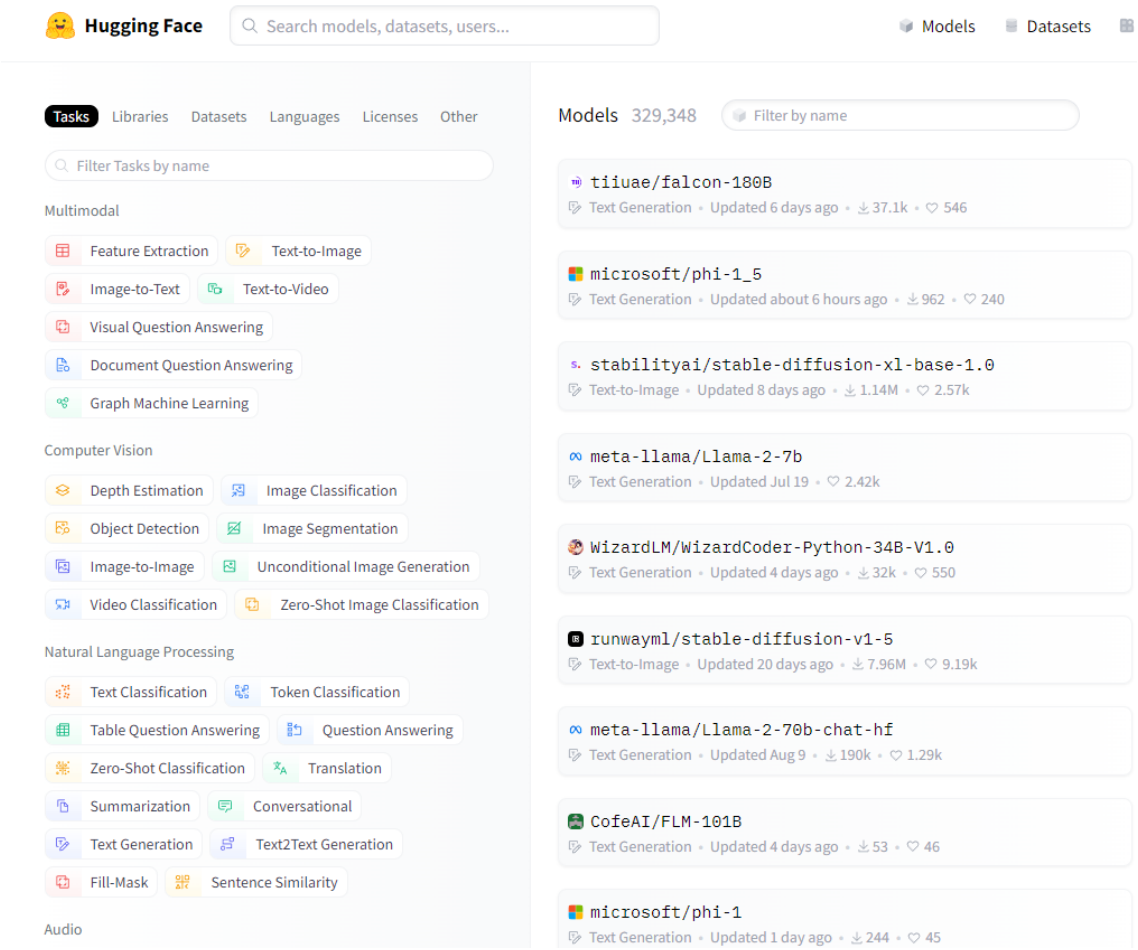

Figura 2.1: Hugging face - Plataforma

A Hugging Face é uma empresa e plataforma de código aberto 2.1 que se concentra no desenvolvimento e fornecimento de ferramentas e bibliotecas para processamento de linguagem natural e aprendizagem automática. São conhecidos principalmente pela contribuição para a comunidade de NLP com a criação da Transformers, uma biblioteca de código aberto amplamente usada para treinar e usar modelos de linguagem pré-treinados.

A principal contribuição da Hugging Face para a comunidade de NLP é a disponibilidade de uma grande variedade de modelos de linguagem pré-treinados, como BERT, GPT-2, RoBERTa e muitos outros. Esses modelos pré-treinados podem ser usados para uma ampla variedade de tarefas de processamento de linguagem natural, tais como classificação de texto, tradução automática, geração de texto e muito mais.

O artigo [23] apresenta a primeira investigação empírica sobre a reutilização de modelos pré-treinados (PTMs). Os autores entrevistaram 12 profissionais do ecossistema PTM mais popular, Hugging Face, para aprender as práticas e desafios da reutilização de PTMs.

Além disso, a Hugging Face fornece uma API de fácil acesso para esses modelos pré-treinados, o que torna mais simples para programadores e investigadores utilizarem esses modelos nos seus próprios projetos e aplicações.

2.5.1 Tranformers

A biblioteca Transformers do Python [7] é uma biblioteca de open source desenvolvida pela Hugging Face. Fornece uma interface fácil de utilizar com modelos de linguagem pré-treinados, especialmente indicadas para tarefas de processamento de linguagem natural (NLP).

A principal funcionalidade da biblioteca ”transformers”é permitir que os programadores utilizem modelos de linguagem pré-treinados para realizar uma variedade de tarefas relacionadas à linguagem, como classificação de texto, geração de texto, sumarização, tradução automática, entre outras.

A biblioteca ”transformers”é baseada em arquiteturas de modelos de linguagem poderosas e bem conhecidas, como o BERT (Bidirectional Encoder Representations from Transformers), GPT (Generative Pre-trained Transformer), RoBERTa, entre outros. Esses modelos são treinados em grandes quantidades de dados textuais e capturam informações semânticas e sintáticas para lidar com várias tarefas de NLP.

A biblioteca ”transformers”também fornece uma ampla gama de recursos, como:

- **Carregamento e uso de modelos pré-treinados:** A biblioteca permite carregar modelos pré-treinados com apenas algumas linhas de código. Esses modelos podem ser usados diretamente para realizar tarefas específicas de NLP.
- **Tokenização:** A tokenização é um passo importante no processamento de linguagem natural, e a biblioteca ”transformers”fornece métodos eficientes para dividir textos em tokens compreendidos pelos modelos pré-treinados.
- **Fine-tuning:** Além de usar modelos pré-treinados diretamente, a biblioteca também oferece suporte para o fine-tuning dos modelos em conjuntos de dados específicos. Isso permite adaptar os modelos pré-treinados para tarefas específicas com dados adicionais de treino.
- **Avaliação de modelos:** A biblioteca ”transformers”também inclui métricas de avaliação usadas por norma para medir o desempenho dos modelos de NLP em diferentes tarefas, como precisão, recall, F1-score, entre outros.

A biblioteca ”transformers”tornou-se extremamente popular e amplamente utilizada na comunidade de NLP devido à sua facilidade de uso, ampla seleção de modelos pré-treinados e recursos abrangentes. Simplifica muito o processo de trabalhar com modelos de linguagem pré-treinados e acelera o desenvolvimento de soluções de NLP de alta qualidade.

2.5.2 Trainer

O Trainer é uma classe fornecida pela biblioteca ”transformers”do Python, desenvolvida pela Hugging Face. Essa classe é projetada para simplificar o processo de treinos de modelos de linguagem pré-treinados para tarefas específicas de processamento de linguagem natural (NLP).

A classe Trainer facilita a configuração e o treino de modelos pré-treinados para tarefas específicas de NLP. Para usá-lo, é necessário fornecer alguns componentes essenciais:

- **Modelo pré-treinado:** É necessário especificar o modelo pré-treinado que será treinado e ajustado para a tarefa específica. A biblioteca ”transformers”fornece uma ampla variedade de modelos pré-treinados para escolher.
- **Tokenizer:** O Trainer requer um tokenizer para converter os textos em sequências de tokens compreendidas pelo modelo. O tokenizer é responsável pela divisão do texto em tokens e pela atribuição de identificadores numéricos a esses tokens.
- **Conjuntos de dados:** É necessário fornecer conjuntos de dados de treino, validação e teste para o treino e avaliação do modelo. Esses conjuntos de dados devem ser preparados de acordo com a estrutura de entrada necessária pelo modelo.
- **Configuração de treino:** O Trainer permite configurar vários parâmetros de treino, como número de épocas, tamanho do lote (batch size), taxa de aprendizagem (learning rate), algoritmo de otimização e assim por diante.

Uma vez que todos os componentes estejam definidos, é possível instanciar a classe Trainer e usá-la para treinar o modelo pré-treinado na tarefa específica. O Trainer cuidará de todos os detalhes de treino, incluindo o processamento dos dados, o cálculo das perdas, a atualização dos pesos do modelo e a avaliação do desempenho do modelo.

Além disso, o Trainer também permite salvar o modelo treinado e o tokenizer, bem como carregar modelos previamente treinados para realizar inferências em novos dados.

O Trainer simplifica consideravelmente o processo de treino de modelos de linguagem pré-treinados, tornando-o mais acessível e fácil de implementar para uma ampla variedade de tarefas de NLP.

2.5.3 Pipeline

A classe pipeline da biblioteca Hugging Face Transformers é uma funcionalidade que simplifica o uso de modelos pré-treinados para várias tarefas de processamento de linguagem natural (NLP) e

outras tarefas relacionadas. Esta classe permite realizar inferências em modelos sem a necessidade de escrever código complexo de pré-processamento e pós-processamento.

A classe pipeline é uma maneira fácil e rápida de usar os modelos pré-treinados para realizar tarefas como: classificação de texto (ex: análise de sentimento), preenchimento de lacunas em texto (ex: previsão de palavras ausentes em uma frase), geração de texto (ex: criar novos textos com base em uma entrada inicial), reconhecimento de entidades nomeadas (NER - Named Entity Recognition), e muitas outras tarefas relacionadas com processamento de linguagem natural.

Ao usar a classe pipeline, não é necessário preocupar-se com detalhes de arquitetura do modelo, carregamento do tokenizador ou escrita de código complexo para pré-processamento e pós-processamento dos dados.

2.6 Base de dados Relacional vs NoSQL

Bases de dados são coleções organizadas de dados que são armazenados eletronicamente num computador. São projetados para permitir o armazenamento, gerenciamento e recuperação eficiente de informações.

As bases de dados podem ser criadas para armazenar qualquer tipo de informação, desde informações pessoais, como nomes e endereços, até informações comerciais, como registos de vendas, inventários ou informações financeiras. São usadas em muitos sistemas de software, como sistemas de gestão de conteúdo, sistemas de gestão de relacionamento com o cliente, e muito mais. Também são usados em conjunto com outras tecnologias, como a inteligência artificial, para realizar análises e obter informações valiosas a partir de grandes conjuntos de dados.

Existem diferentes tipos de bases de dados, incluindo bases de dados relacionais e bases de dados NoSQL [26], neste artigo são apontadas algumas das divergências referidas de seguida.

As bases de dados NoSQL têm ganho popularidade ao longo dos anos devido às suas inúmeras vantagens em relação às bases de dados relacionais. Ao contrário das bases de dados relacionais, foram projetadas para lidar com grandes volumes de dados, oferecendo escalabilidade horizontal e flexibilidade em relação ao modelo de dados.

Relativamente a flexibilidade, as bases de dados relacionais requerem uma estrutura rígida de tabelas e colunas, as bases de dados NoSQL permitem que os dados sejam armazenados em diferentes formatos, como documentos, gráficos, colunas, entre outros. Isso permite que os dados sejam armazenados de forma mais natural, sem a necessidade de reestruturar as tabelas para cada tipo de dado.

Numa base de dados relacional, a escalabilidade vertical é comum, onde a adição de capacidade é feita através da atualização de hardware existente. Relativamente a escalabilidade horizontal, tem algumas fragilidades, já o NoSQL permite adicionar mais nós ao cluster, aumentando a capacidade da base de dados. Isto significa que, à medida que a quantidade de dados cresce, o NoSQL pode lidar com isso de forma mais eficiente do que uma base de dados relacional.

As bases de dados NoSQL também oferecem alta disponibilidade, replicando os dados em vários nós. Isso significa que, se um nó falhar, os dados ainda estarão disponíveis em outro nó,

garantindo que os dados estejam sempre disponíveis. Além disso, são frequentemente usados em softwares de grande escala que precisam de respostas rápidas, como sistemas de comércio virtual e jogos online.

Uma das bases de dados NoSQL que conferem melhor eficiência ao nível de pesquisa e constituição arquivística são as bases de dados de grafos [37].

Em resumo, as bases de dados NoSQL são uma excelente escolha quando se trata de lidar com grandes volumes de dados, escala e flexibilidade de dados. Embora as bases de dados relacionais ainda tenham a sua utilidade em muitos casos, é importante entender as vantagens do NoSQL e considerá-lo ao escolher o tipo de banco de dados para um projeto específico.

2.6.1 Microsoft SQL Server - relacional

O Microsoft SQL Server é um sistema de gestão de bases de dados relacional (RDBMS) desenvolvido pela Microsoft. Projetado para armazenar, recuperar, gerir e manipular dados, seguindo o modelo de dados relacional. O SQL Server é uma das soluções mais populares no mercado para a criação e administração de bases de dados, sendo amplamente utilizado numa variedade de aplicações e cenários.

Aqui estão algumas características e informações-chave sobre o Microsoft SQL Server:

- **Modelo de Dados Relacional**: O SQL Server segue o modelo relacional, no qual os dados são armazenados em tabelas, e as relações entre as tabelas são definidas por chaves primárias e estrangeiras. Isto permite a organização estruturada e eficiente dos dados.

- **Linguagem SQL**: O SQL Server utiliza a linguagem SQL (Structured Query Language) para realizar consultas, inserções, atualizações e exclusões de dados. O SQL é uma linguagem padrão da indústria para interagir com bases de dados relacionais.

- **Edições Diferentes**: O SQL Server está disponível em várias edições, desde versões adaptadas para pequenas empresas até edições corporativas de alto desempenho. Algumas das edições incluem a Express (gratuita), a Standard, a Enterprise e a Developer.

- **Ferramentas de Gestão**: A Microsoft oferece diversas ferramentas para gerir bases de dados SQL Server, incluindo o SQL Server Management Studio (SSMS) para desenvolvimento e administração, e o SQL Server Data Tools (SSDT) para desenvolvimento de bases de dados e aplicativos.

- **Segurança**: O SQL Server oferece recursos avançados de segurança, incluindo autenticação, autorização, criptografia de dados, auditoria e controlo de acesso detalhado.

- **Integração com Plataforma Microsoft**: O SQL Server é projetado para funcionar de forma integrada com outras tecnologias e produtos da Microsoft, como o sistema operacional Windows, a plataforma de cloud Azure, o .NET Framework e muito mais.

- **Recursos Avançados**: O SQL Server oferece uma variedade de recursos avançados, como replicação para manter cópias atualizadas de bases de dados em várias localidades, serviços de análise de dados, serviços de relatórios, suporte a análise de big data e integração com linguagens como R e Python.

- **Escalabilidade**: O SQL Server permite escalabilidade vertical (adicionando mais recursos a um único servidor) e horizontal (distribuindo dados entre vários servidores).

- **Opções de Implementação**: O SQL Server pode ser implementação localmente em servidores próprios, em servidores virtuais ou na cloud usando o Microsoft Azure.

O Microsoft SQL Server é usado numa ampla gama de cenários, desde pequenas aplicações locais até sistemas corporativos críticos que requerem alta disponibilidade e desempenho. A versatilidade, recursos avançados e integração com o ecossistema Microsoft fazem deste uma escolha popular para empresas e desenvolvedores que trabalham com bases de dados relacionais.

É possível aceder a este tipo de base de dados diretamente através de bibliotecas do python, como por exemplo o pymssql

Pymssql

A biblioteca pymssql do python permite a conexão a uma base de dados Microsoft SQL Server através do conhecimento do ip do servidor, da port, do username, da password, e do nome da base de dados. Conseguimos assim estabelecer uma ligação a base de dados e através de um comando SQL obtermos os dados que queremos utilizar.

Esta biblioteca é útil na extração dos dados da base de dados relacionais do Digitarq.

2.6.2 Base de dados de grafos - NoSQL

Bases de dados de grafos são um tipo específico de sistema de gestão de bases de dados NoSQL que são projetados para armazenar, gerir e consultar dados que possuem estruturas de grafo. Um grafo é uma representação visual de objetos (chamados de nós ou vértices) conectados por relações (chamadas de arestas) entre eles. As bases de dados de grafos são ideais para modelar e consultar dados que possuem relações complexas e interconexões.

Aqui estão algumas características e conceitos-chave das bases de dados de grafos:

- **Nós e Arestas**: Os dados em uma base de dados de grafos são representados como nós (entidades) e arestas (relações). Cada nó pode ter propriedades associadas ao mesmo, e as arestas representam as conexões ou relacionamentos entre os nós.

- **Modelo de Dados Flexível**: As bases de dados de grafos permitem modelar uma variedade de cenários, desde redes sociais até redes de infraestrutura, sistemas de recomendação e muito mais. Isso ocorre porque o modelo de grafos é naturalmente adequado para representar relacionamentos complexos.

- **Consulta de Padrões**: Uma das principais vantagens das bases de dados de grafos é a capacidade de realizar consultas complexas sobre padrões. Isso significa que é possível procurar por caminhos específicos, relacionamentos entre nós e até mesmo padrões mais abstratos dentro do grafo.

- **Desempenho em Validações de Relacionamento**: Em comparação com bases de dados relacionais, as bases de dados de grafos são mais eficientes para validações que envolvem a navegação de relações. É especialmente útil em cenários onde as relações são cruciais, como redes de transporte, mapas ou sistemas de recomendação.

- **Algoritmos de Análise de Grafos**: Muitas bases de dados de grafos vêm com algoritmos de análise integrados que permitem calcular métricas e informações relevantes sobre a estrutura do grafo. Isso é útil para identificar padrões, influenciadores, clusters e outras informações importantes.

- **Escalabilidade**: Bases de dados de grafos geralmente oferecem boas opções de escalabilidade, permitindo a adição de mais nós e servidores conforme a necessidade.

- **Exemplos de Uso**: Estas bases de dados são frequentemente usadas em redes sociais (modelagem de amizades, seguidores), sistemas de recomendação (entendendo preferências do utilizador), análise de fraudes (detecção de padrões suspeitos) e qualquer cenário em que as relações entre os dados sejam cruciais.

Exemplos populares de bases de dados de grafos incluem Neo4j, Amazon Neptune e Microsoft Azure Cosmos DB (modo de API de Grafo). Estas bases de dados são particularmente valiosas quando os dados têm uma estrutura de relacionamento complexa e a análise dessas conexões é uma parte essencial da aplicação.

- **Neo4j:** é uma base de dados de grafos OpenSource. As bases de dados de grafos foram criadas especificamente para possibilitar o armazenamento de relacionamentos e a navegação por eles. Os relacionamentos são elementos distintos que agregam a maior parte do valor para base de dados de grafos. Estas base de dados utilizam nós para armazenar entidades de dados e arestas para armazenar os relacionamentos entre as entidades, a este conjunto entre nós e arestas dá se o nome de grafo que por sua vez formam a base de dados de grafos.

2.7 Elastic Search

O Elasticsearch é um mecanismo de análise e pesquisa de texto completo de OpenSource altamente escalável. Permite armazenar, pesquisar e analisar grandes volumes de dados rapidamente e quase em tempo real. Geralmente é usado como o mecanismo/tecnologia subjacente que alimenta aplicativos com recursos e requisitos de pesquisa complexos, como é o caso do mecanismo de pesquisa do Digitarq.

Elasticsearch é uma base de dados NoSQL. Isso significa que armazena dados de forma não estruturada e que não pode usar SQL para consultá-los.

O artigo [39] aborda o uso do Elasticsearch para pesquisar documentos. O Elasticsearch é um mecanismo de busca baseado no Apache Lucene, uma biblioteca de software de recuperação de informações gratuita e de código aberto. O Elasticsearch fornece um mecanismo de busca distribuído de texto completo com uma interface web HTTP e documentos JSON. O artigo examina a API REST do Elasticsearch e demonstra operações básicas de pesquisa utilizando apenas solicitações HTTP.

2.7.1 Haystack

Haystack é uma estrutura Python para processamento de linguagem natural (NLP) que se concentra na pesquisa semântica e na resposta a perguntas (QA). O Haystack permite que os utilizadores façam perguntas em linguagem natural e recebam uma resposta informativa de uma grande coleção de documentos - tudo em segundos.

Ao usar o Haystack por cima do Elasticsearch, é possível obter o melhor dos dois mundos. As capacidades de armazenamento e pesquisa de palavras-chave do mecanismo funcionam bem com os mais recentes modelos de linguagem baseados no Transformer para oferecer uma experiência de pesquisa poderosa para os utilizadores.

2.8 Docker

O Docker é uma plataforma OpenSource para desenvolvimento, envio e execução de aplicativos. O Docker permite a separação de aplicativos relativamente à sua infraestrutura para que possa fornecer software rapidamente. Com o Docker, é possível administrar a infraestrutura da mesma forma que gerir os seus aplicativos.

O Docker vai ser essencial para alocar em containers várias imagens de bases de dados do Elasticsearch através da criação de um ficheiro docker-compose.yml, de modo a facilitar a utilização dos mesmos.

2.9 FastAPI

FastAPI é uma estrutura Python que contém um conjunto de ferramentas que permite usar uma interface REST para chamar funções usadas com frequência para implementar aplicativos. É acedido através de uma API REST para chamar blocos de construção comuns para um aplicativo.

O objetivo da utilização da FastAPI serve para construir uma interface para aceder à base de dados de grafos e à base de dados do Elasticsearch.

2.10 Postman

O Postman é uma plataforma de API para criar e usar APIs. Simplifica cada etapa do ciclo de vida da API e agiliza a colaboração para a criação de APIs melhores com mais eficiência.

Permite armazenar especificações da API, para nos ajudar na realização de testes e verificação do funcionamento da mesma.

2.11 Google Colab

O Google Colab, ou Google Colaboratory, é uma plataforma de cloud gratuita oferecida pelo Google que permite escrever e executar scripts. É baseado no ambiente de notebooks Jupyter, o que significa que é possível criar documentos interativos que combinam código, texto explicativo, visualizações e outros elementos.

O artigo [17] apresenta uma análise detalhada do Google Colaboratory (também conhecido como Colab), um serviço de nuvem baseado no Jupyter Notebooks para disseminação de educação e pesquisa em aprendizagem automática. Colab fornece um ambiente de execução totalmente configurado para desenvolvimento de modelos de aprendizagem automática e acesso gratuito a uma GPU robusta. Este artigo realiza uma análise do Colab em termos de recursos de hardware, desempenho e limitações, usando o Colab para acelerar o desenvolvimento de modelos de aprendizagem automática para visão computacional e outras aplicações centradas em GPU.

Uma das principais vantagens do Google Colab é que ele fornece acesso gratuito a recursos computacionais, incluindo CPUs e GPUs. Isso é especialmente útil para a construção e treino de modelos neuronais, que geralmente requerem muito poder de processamento.

O Google Colab oferece suporte a várias bibliotecas populares de aprendizagem automática, como TensorFlow, Keras, PyTorch e scikit-learn, entre outras. Essas bibliotecas fornecem ferramentas e funções para projetar, treinar e avaliar modelos neuronais.

É possível importar as bibliotecas necessárias, carregar conjuntos de dados, pré-processar os dados, definir a arquitetura do modelo, compilar o modelo com uma função de perda e um otimizador, treinar o modelo com um conjunto de dados de treino, avaliar o desempenho do modelo para um determinado conjunto de teste e fazer previsões usando o modelo treinado.

O Google Colab também permite que você utilize as GPUs disponíveis para acelerar o treino de modelos neuronais, o que é particularmente útil quando você lida com conjuntos de dados grandes ou modelos complexos.

Além disso, o Colab oferece integração com outros serviços do Google, como Google Drive, o que facilita o compartilhamento e o armazenamento de notebooks e conjuntos de dados.

Em resumo, o Google Colab é uma plataforma poderosa e acessível para a construção e treino de modelos neuronais. Ele oferece recursos computacionais gratuitos, suporte a bibliotecas populares de aprendizagem automática e a capacidade de colaborar e compartilhar notebooks.

2.12 Wandb

O Wandb é a abreviação de "Weights & Biases", uma plataforma [12] para acompanhamento e visualização de exepriências de aprendizagem automática. O Wandb é projetado para ajudar os desenvolvedores a rastrear, registar e visualizar os resultados dos seus modelos de forma eficiente.

Algumas características e funcionalidades do wandb incluem:

- **Registo de métricas e resultados:** O wandb permite registar métricas de treino, validação e teste, como perda, precisão, recall, F1-score, entre outros. Além disso, é possível registar hiperparâmetros, gráficos, imagens e outros tipos de dados relevantes para análise posterior.
- **Visualização interativa:** Os dados registados pelo wandb são exibidos através de uma interface web interativa que permite visualizar e comparar resultados de diferentes modelos. Gráficos, tabelas e outros elementos visuais ajudam a entender o desempenho do modelo ao longo do tempo e em diferentes configurações.
- **Integração com várias bibliotecas:** O wandb é compatível com várias bibliotecas populares de aprendizagem automática, como PyTorch, TensorFlow, scikit-learn e Hugging Face. Pode ser facilmente integrado ao código existente para rastrear automaticamente métricas e resultados durante o treino do modelo.
- **Colaboração e partilha:** O wandb permite compartilhar os resultados do modelo com outras pessoas de forma colaborativa. É possível compartilhar links para modelos específicos ou fornecer acesso a equipas e colaboradores.

Em resumo, o wandb é uma ferramenta poderosa para rastrear e visualizar dados de modelos treinados, fornecendo uma interface intuitiva para acompanhar o progresso do treino, comparar resultados e colaborar com outros membros da equipa.

2.13 OpenStreetMap

O OpenStreetMap (OSM) é um projeto colaborativo de mapeamento geográfico de open source. Lançado em 2004 e tem como objetivo criar um mapa mundial detalhado, livre e acessível por qualquer pessoa. A principal característica do OpenStreetMap é que os mapas e os dados associados são criados e atualizados pelos próprios utilizadores da comunidade OSM em vez de serem criados por empresas privadas ou governos.

A comunidade do OpenStreetMap é composta por milhões de voluntários em todo o mundo que contribuem com informações geográficas de diversas maneiras, tais como:

- **Pontos de interesse:** Edifícios, parques, restaurantes, lojas e outros locais de interesse são marcados no mapa.
- **Estradas e vias:** A rede de estradas e vias, incluindo autoestradas, ruas, trilhas e caminhos, são mapeados em detalhe.

- **Limites e fronteiras:** Divisões políticas, como países, estados e cidades, são mapeadas.
- **Características naturais:** Rios, lagos, montanhas e outras características geográficas são registadas no mapa.
- **Transporte público:** Informações sobre rotas de autocarro, metro e outras formas de transporte público também podem ser adicionadas.

Os dados do OpenStreetMap estão disponíveis sob a licença Open Database License (ODbL), que permite a qualquer pessoa utilizar, compartilhar e modificar os dados, desde que atribua o devido crédito aos contribuidores originais.

O projeto OpenStreetMap é amplamente utilizado por indivíduos, empresas e organizações numa variedade de aplicativos e serviços, incluindo aplicativos de navegação, planeamento de rotas, mapeamento personalizado, análise geográfica e muito mais. Devido à natureza colaborativa do OSM, os mapas continuam a ser atualizados e aprimorados constantemente pela comunidade global de contribuidores.

O artigo [38] discute os desafios e oportunidades do uso do OpenStreetMap (OSM) em combinação com aprendizagem automática. O OSM é um serviço de mapeamento editável, disponível gratuitamente e baseado na comunidade, criado como uma alternativa às fontes autoritárias. Dado que é editado principalmente por voluntários com diferentes habilidades de mapeamento, a conhecimento e a qualidade das anotações são heterogêneas em diferentes localizações geográficas.

O artigo avalia métodos recentes baseados em aprendizagem automática para melhorar e usar dados do OSM. Estes métodos visam melhorar a cobertura e a qualidade das camadas do OSM, normalmente utilizando sistemas de informações geográficas (GIS) e tecnologias de detecção remota, ou usar as camadas existentes do OSM para treinar modelos baseados em dados de imagem para servir aplicações como navegação e classificação de locais.

2.13.1 Nominatim

O Nominatim é um serviço de geocodificação e pesquisa de localização baseado no OpenStreetMap (OSM). Permite converter endereços ou descrições de lugares em coordenadas geográficas (geocodificação) e também permite pesquisar lugares com base em coordenadas geográficas (reverso de geocodificação).

O Nominatim utiliza os dados do OpenStreetMap para realizar essas funcionalidades. Indexa e armazena os dados geográficos do OSM, como nomes de ruas, cidades, pontos de interesse e outros elementos geográficos, de forma a facilitar a pesquisa por informações de localização.

Principais recursos e funcionalidades do Nominatim: geocodificação, reverso de geocodificação, pesquisa de Lugares, detalhes da localização

O Nominatim é amplamente utilizado em aplicativos e serviços que requerem geocodificação e pesquisa de localização, como mapas, aplicativos de navegação, serviços de localização em tempo real, planeamento de rotas e muito mais. Inda oferece uma maneira conveniente de obter informações geográficas precisas com base nos dados do OpenStreetMap.

Capítulo 3

Trabalho relacionado

Nesta secção o objetivo é analisar e relacionar trabalho realizado por outros que se enquadrem com aquilo que pretendemos desenvolver.

3.1 Natural Language Processing - NLP

Processamento de linguagem natural (NLP - Natural Language Processing) [21] é uma área da inteligência artificial que se concentra na interação entre computadores e a linguagem humana. É o estudo de como as máquinas podem processar, entender e produzir linguagem natural, como a linguagem falada e escrita.

O NLP envolve técnicas de análise sintática, semântica e pragmática, e utiliza algoritmos para analisar, interpretar e gerar linguagem natural. O objetivo é permitir que as máquinas entendam a linguagem humana de forma semelhante à forma como os humanos a entendem.

Algumas das tarefas que o NLP pode realizar incluem a tradução automática, resumo automático de texto, reconhecimento de voz, análise de sentimento, classificação de texto, entre outras. Por exemplo, um sistema de chatbot pode utilizar técnicas de NLP para entender a linguagem natural dos utilizadores e responder de forma adequada.

O NLP é uma área em constante evolução, com a crescente disponibilidade de grandes datasets e o desenvolvimento de modelos neuronais cada vez mais avançados. Com isso, espera-se que a aplicação do NLP continue a crescer e se expanda em muitas áreas, desde assistentes pessoais e chatbots até análises mais avançadas de dados em setores como saúde, finanças e segurança.

3.1.1 Named Entity Recognition - NER

A extração de entidades (NER - Named Entity Recognition) é um sub processo do processamento de linguagem natural (NLP) destinado a identificar/extrair entidades como pessoas, locais, organizações e outros tipos de conceitos presentes numa determinada cadeia de texto.

O principal objetivo do reconhecimento de entidades é extrair informações estruturadas de cadeias de texto não estruturadas, este processo é uma etapa crucial em muitas tarefas de processamento de linguagem natural (NLP). Através do artigo [33] é possível entender o conceito que é o NER e as diferentes abordagens que existem na utilização deste sub processo.

O reconhecimento de entidades pode ser realizado através da utilização de uma variedade de técnicas incluindo *rule-based models, dictionary-based methods, e machine learning-based methods. Machine learning-based methods* são mais utilizados uma vez que aprendem a reconhecer entidades de dados pré-treinados e previamente anotados, conseguindo assim adaptar-se a novos dados e idiomas com mais facilidade.

Para executar o processo de extração de entidades existem diversos modelos de inteligência artificial. A maioria destes modelos são treinados para a língua inglesa. Mesmo aqueles que são treinados com datasets na língua portuguesa, costumam ser treinados com datasets de menor tamanho. Como é o caso do BERT [8], T5 [11], RoBERTa [10], ELECTRA [9]

Para a construção de um modelo de extração de entidades entidades encontramos alternativas. O artigo [35], onde optaram por pré-treinar um modelo *Portuguese BERT* afinando o mesmo através de um modelo *BERT-CRF*, todavia durante este processo utilizaram um conjunto de dados anotados muito pequeno o que acabou por resultar num pré-treino pobre, enfraquecendo assim o modelo criado. Pode ser uma opção a considerar uma vez que o código feito está disponível online e podemos voltar a pré-treinar e a afinar este modelo que é algo que os autores referem na secção de trabalho futuro.

Este artigo sugere uma opção com bons argumentos que pode solucionar relativamente ao desafio da extração de entidades em português [41]. A ideia principal deste estudo e implementação seria traduzir dados não legendados de um idioma de *low-resource* para um idioma de *high-resource* (por exemplo de português para inglês). No processo de tradução é utilizado um modelo multi-lingual de tradução pré-treinado (M2Mlarge), que inclui a língua portuguesa. Posteriormente é aplicado aos dados traduzidos o modelo de extração de entidades (NER), os resultados (entidades) obtidos serão traduzidos de novo para o idioma original, estas entidades são por sua vez associadas às palavras que contêm essa etiquetação.

O artigo [19] tem como objetivo encontrar alternativas para a extração de entidades em texto não etiquetado na língua portuguesa. Os autores implementam quatro modelos alternativos com o intuito de perceber qual apresenta melhor resultados ao nível da extração. Durante o processo de obtenção de resultados experimentais recorrem a dados na língua inglesa e portuguesa e para além de compararem os quatro modelos, comparam também a capacidade de extração de entidades para ambos os idiomas. Considerámos esta opção relevante uma vez que os resultados obtidos foram positivos apesar dos resultados para o idioma inglês terem sido melhores, mas que se justifica pela quantidade de palavras utilizadas no pré-treino do modelo para a língua inglesa comparativamente ao da língua portuguesa.

3.1.2 NER - Reconhecimento específico de Localizações

O trabalho realizado uma das entidades que será reconhecida é a Localização e para isso foi feita a análise de dois artigos que propões diferentes abordagens na identificação de localizações.

O artigo [40] aborda o problema de identificar entidades de localização em coleções temporais de artigos de notícias. Os autores propõem um novo modelo de aprendizagem automática para

LER que é capaz de identificar entidades de localização com mais precisão do que os métodos anteriores. O modelo proposto pelos autores é baseado num modelo neuronal supervisionado. O modelo é treinado através de um conjunto de dados de artigos de notícias anotados com entidades de localização. O modelo é capaz de identificar entidades de localização com uma precisão de 95%.

O artigo [27] propõe um método de reconhecimento de entidades (NER) para textos geológicos baseado em GeoBERT. O modelo proposto usa um método de ajuste de duas etapas, onde o modelo BERT pré-treinado é ajustado com conhecimento de domínio geológico na primeira etapa e, na segunda etapa, um pequeno número de amostras é usado para completar a tarefa NER em relatórios geológicos. O modelo proposto alcança uma pontuação F1 muito alta em comparação com os modelos base no conjunto de dados construído.

Ambas as abordagens são interessantes porém como o objetivo passa por identificar três tipos de entidades distintas (localizações, pessoas e organizações) iria por estar a criar um modelo independente para a identificação de entidades do tipo localização o que iria aumentar o tempo de implementação.

3.1.3 BERT e T5

BERT

O BERT (Bidirectional Encoder Representations from Transformers) é um modelo de linguagem pré-treinado desenvolvido pelo Google [13] que utiliza a arquitetura de redes neuronais conhecida como Transformers. Especificamente, o BERT é projetado para entender e representar o contexto das palavras numa frase, levando em consideração o contexto tanto à esquerda quanto à direita de cada palavra.

No contexto do reconhecimento de entidades, o BERT pode ser utilizado como um componente importante para realizar a tarefa de identificar e classificar diferentes tipos de entidades num texto, como nomes de pessoas, organizações, locais, datas, entre outros. O modelo é treinado para uma tarefa de preenchimento de lacunas, na qual uma palavra ou uma entidade é mascarada numa frase e o BERT tenta prever qual é a palavra ou entidade correta com base no contexto circundante.

T5

A arquitetura T5 (Text-to-Text Transfer Transformer) é um modelo de aprendizagem automática desenvolvido pelo Google AI Language [16]. Diferente de modelos como o BERT, que são pré-treinados numa única tarefa, o T5 é projetado para realizar uma ampla gama de tarefas de processamento de linguagem natural (NLP) utilizando uma abordagem ”text-to-text”.

A ideia central do T5 é transformar todas as tarefas de NLP num formato padronizado de entrada e saída de texto. Por outras palavras, todas as tarefas são reformuladas como problemas de transformação de texto, onde uma sequência de texto é fornecida como entrada e a tarefa consiste em gerar a sequência de texto de saída correta. Essa abordagem unificada permite que o modelo seja treinado de forma consistente em várias tarefas.

BERT VS T5

O artigo [31] é relevante, pois aborda a arquitetura T5 (Text-to-Text Transfer Transformer) e compara o seu desempenho com o modelo BERT em tarefas de processamento de linguagem natural (NLP).

Ao longo do artigo, discutem a abordagem ”text-to-text”do T5 e como ela difere do BERT. Enquanto o BERT é pré-treinado numa única tarefa de previsão de mask words, o T5 adota uma abordagem mais ampla, reformulando todas as tarefas de NLP como problemas de transformação de texto. Essa abordagem unificada permite que o T5 seja aplicado numa variedade de tarefas de NLP, fornecendo resultados competitivos.

Os resultados apresentados no artigo mostram que o T5 supera o desempenho do BERT em várias tarefas, como tradução automática, resumo de texto e resposta a perguntas. Além disso, o estudo destaca a capacidade do T5 de realizar transferência de aprendizagem efetiva, aplicando conhecimentos prévios aprendidos numa tarefa para melhorar o desempenho em outras tarefas relacionadas.

Comparando os dois modelos, o T5 demonstra uma maior flexibilidade e adaptabilidade, enquanto o BERT destaca-se em tarefas específicas de NLP. A abordagem ”text-to-text”do T5 permite que ele seja ajustado para uma ampla variedade de tarefas, proporcionando melhor desempenho e generalização. No entanto, o BERT pode ser mais adequado para tarefas mais especializadas, onde a modelagem de contexto local é crucial.

3.2 INESC - projeto Episa

O projeto EPISA – Inferência de Entidades e Propriedades para Arquivos Semânticos [14] [30] – traça um caminho essencial no panorama nacional para a acessibilidade do património cultural português e da informação em geral.

O objetivo do projeto era criar um novo modelo de descrição para arquivos e promover a criação semiautomática de meta dados. O seu principal objetivo é incorporar os arquivos nacionais na rede global de dados semânticos vinculados.

O projeto está a ser desenvolvido pelo INESC TEC (Instituto de Engenharia de Sistemas e Computadores, Tecnologia e Ciência), em parceria com a Universidade de Évora e a DGLAB (Direção Geral do Livro, Arquivo e Bibliotecas).

O objetivo deste projeto prende-se com a necessidade de uma nova geração de ferramentas de descrição que inclua bibliotecas, arquivos e museus, com um grão mais fino, mais flexível e especialmente mais facilmente processável por máquinas. O projeto é muito semelhante ao desenvolvido neste projeto uma vez que os dados são guardados numa base dados de grafos seguindo o Modelo de Referência Conceitual do CIDOC-CRM.

Em suma, o projeto EPISA tem como principais objetivos criar um novo modelo de descrição para arquivos, baseado em modelos já propostos para museus e arquivos, apoiar a descrição arquivística na criação semiautomática de meta dados, migrar descrições arquivísticas padrão ISAD

existentes para padrões compatíveis com o modelo CIDOC-CRM (museus) e RiC (ICA), incorporar os arquivos na rede global de dados vinculados semânticos, vincular dados de arquivo a outras fontes de informação, propor interfaces que possibilitem uma interação mais eficiente, para especialistas, para usuários em geral e para máquinas, promover a presença de arquivos nacionais em agregadores internacionais, facilitar o acesso a arquivos/património cultural e facilitar o acesso à informação.

O projeto EPISA acaba por se tratar de um projeto semelhante ao nosso porém não houve implementação uma vez que o projeto se tratou apenas de investigação e que neste momento se encontra finalizado.

3.3 CIDOC-CRM

O Modelo de Referência Conceitual do CIDOC (CRM) [15] é uma ferramenta teórica e prática para integração de informações no campo do património cultural. Pode ajudar analistas, administradores e o público a explorar questões complexas em relação ao nosso passado em conjuntos de dados diversos e dispersos. O CIDOC CRM consegue isto através do fornecimento de definições e uma estrutura formal para descrever os conceitos e relacionamentos implícitos e explícitos usados na documentação do património cultural e de interesse geral para a consulta e exploração de tais dados. Esses modelos também são conhecidos como ontologias formais, essas descrições formais permitem a integração de dados através de múltiplos recursos.

O CIDOC CRM foi desenvolvido com o intuito de promover uma compreensão das informações do património cultural, fornecendo uma estrutura semântica comum e extensível para a integração de informações do património cultural. O objetivo principal passa por ser uma linguagem comum para especialistas para que estes deste modo possam formular requisitos para sistemas de informação e assim servir como um guia de boas práticas de modelagem conceitual. Dessa forma é possível formular as relações necessárias para intervir entre diferentes fontes de informação sobre o património cultural.

O modelo concetual do CIDOC CRM é um grafo onde os nós são entidades e as arestas são relações. O CIDOC é utilizado no momento de identificação e denominação de entidades dos dados exportados das bases de dados antigas do Digitarq. Este modelo concetual contém 99 classes e cerca de 198 relações, as classes existentes estão hierarquizadas de modo que se possam relacionar umas com as outras.

O modelo CIDOC será o modelo utilizado no processo de migração dos dados para a base de dados de grafos do Neo4j, isto é, a cada coluna da tabela *Components* da base de dados antiga do Digitarq são atribuídas entidades referentes a classes do CIDOC e por sua vez relações com outras entidades, cada classe representa um nó e cada relação representa uma aresta formando assim a base de dados de grafos.

- **Neo4J**: O artigo [18] apresenta uma avaliação de bases de dados de grafos e mapeadores de objetos-grafo (OGM) em arquivos digitais compatíveis com CIDOC CRM. O artigo discute

a escolha da base de dados de grafos como uma solução para representar um modelo de dados baseado em CRM para o novo software de gestão de arquivo digital. O artigo também compara várias bases de dados de grafos, com base na sua maturidade, recursos e desempenho em tarefas padrão, bem como os OGM disponíveis para interagir com cada base de dados de modo orientado a objetos. O artigo conclui que o Neo4j é a melhor escolha para a gestão de arquivos digitais compatíveis com CIDOC CRM.

3.4 API de Localizações no processo de desambiguação

O processo de desambiguação de localizações através de API de localizações é uma das principais soluções para identificar lugares em textos através do uso de uma API (Interface de Programação de Aplicativos) que fornece informações geográficas. A desambiguação é o processo de identificar o significado correto de um lugar quando há várias possibilidades de interpretação, por exemplo, quando um nome de cidade ou país pode ser encontrado em diferentes locais geográficos.

A API de localizações é uma interface fornecida por um serviço ou plataforma que contém uma base de dados com informações geográficas, como nomes de lugares, coordenadas geográficas e outros detalhes relevantes. Essa API permite que os utilizadores acedam e consultem essas informações.

No artigo [25] é apresentada uma solução para identificar e desambiguar localizações de conjuntos de dados. Utilizam o GeoTxt uma ferramenta ou método para detecção e desambiguação de localizações geográficas em mensagens de texto. É usado para identificar endereços de lugares, como nomes de cidades, estados, países, pontos de referência geográficos, entre outros, em textos não estruturados, como tweets, posts em redes sociais, notícias e documentos.

Este tipo de desambiguação é utilizado quando se pretende extrair entidades do tipo localização de dados não estruturados.

Capítulo 4

Análise

Nesta secção será referido e explicado detalhadamente os dois conjuntos de dados mais importantes neste projeto. Serão estes conjuntos de dados que terão influência no decorrer do mesmo: O conjunto de dados das bases de dados antigas do digitarq e os conjuntos de dados que vamos utilizar no desenvolvimento do modelo neuronal.

4.1 Dados do Digitarq

A base de dados do digitarq é atualmente uma base de dados do tipo relacional, que se divide por 18 arquivos de acordo com o distrito, destes os principais são o Arquivo Nacional da Torre do Tombo e o Arquivo Distrital do Porto.

Os arquivos têm a coleção de património cultural mais relevante, largamente digitalizada e acedida tanto por investigadores de História como por leigos dos países de língua portuguesa e de outras. O vasto volume de metadados de descrição arquivística ajudam-nos a encontrar e contextualizar os documentos que se procuram.

Os metadados nestes arquivos são na maior parte descrições textuais do contexto e conteúdo de documentos. Entretanto os objetos de arquivo evoluíram para incluir cada vez mais informação nascida digital, ou seja, dados desenvolvidos a partir de documentos físicos são cada vez menos, e os requisitos de interoperabilidade em repositórios de património cresceram.

4.1.1 Components - Tabela da base de dados do Digitarq

Os arquivos da DGLAB contam com um número total de 60 milhões de documentos. Dentro de cada base de dados a única tabela de metadados que pretendemos migrar é a tabela 'Components' constituída por 109 colunas onde cada linha corresponde aos metadados de um documento. Os conjuntos de dados do DIGITARQ que irão ser migrados são privados.

Na figura 4.1 está disponível a análise de dados: tipo de dados por cada coluna, nome da coluna e rótulo que indica o conteúdo pertencente a cada uma.

Os documentos estão organizados de forma hierárquica, sendo denominada a primeira hierarquia de Fundo. Contendo dentro do mesmo as hierarquias seguintes (Subfundo, Série, Unidade de Instalação, Documento composto, Documento Simples, Colecção).

Destas 109 colunas apenas algumas contém valores não numéricos ou valores tipificados, uma vez que parte destas colunas são metadados numéricos que servem somente para identificação do ID (coluna ”ID”) do documento, identificação do pai do documento (coluna ”ParentID”) e identificação do pai de todos os pais (coluna ”RootParentID”), e outros são valores tipificados como é o caso do valor do nível de descrição de um código de referência que são metadados fixos para cada documento.

No total apenas 26 colunas não contém valores tipificados ou numéricos. Em todas estas colunas de metadados existem descrições de metadados dos documentos que podem conter informação semântica útil.

Colunas que contém cadeias de texto com informação semântica útil: ”UnitTitle”, ”GeogName”, ”Author”, ”AuthorAddress”, ”Recipient”, ”RecipientAddress”, ”Scrivener”, ”Notary”, ”ScopeContent”, ”BiogHist”, ”OtherFindAid”, ”Note”, ”RelatedMaterial”, ”CustodHist”, ”Appraisal”, ”Abstract”, ”Repository”, ”Custom1”, ”Custom6”, ”Custom7”, ”Custom11”, ”Custom14”, ”Custom16”, ”Custom17”, ”Custom18”, ”Custom22”

	Column Name	Type	Rótulo
1	ID	bigint	
2	ParentID	bigint	[identificador do registo de nível imediatamente anterior ao qual pertence]
3	RootParentID	bigint	[identificador do registo do fundo ao qual pertence]
4	Published	bit	Publicado
5	Revised	bit	Revisto
6	Available	bit	Disponível
7	Active	bit	
8	AllowUnitDatesInference	bit	Autorizar inferência de datas extremas
9	AllowExtentsInference	bit	Autorizar inferência de extensões
10	TakenBy	nvarchar	
11	DescriptionLevel	nvarchar	Nível de descrição
12	UnitId	nvarchar	Referência
13	CompleteUnitId	nvarchar	Código de referência
14	CountryCode	nvarchar	Código do país
15	RepositoryCode	nvarchar	Código da entidade detentora
16	UnitTitle	nvarchar	Título
17	UnitTitleType	nvarchar	Tipo de título
18	UnitDateInitial	nvarchar	Data extrema inicial
19	UnitDateInitialCertainty	bit	Certeza da data inicial
20	UnitDateFinal	nvarchar	Data extrema final
21	UnitDateFinalCertainty	bit	Certeza da data final
22	UnitDateBulk	nvarchar	Datas predominantes
23	UnitDateNotes	nvarchar	Datas descritivas
24	Dimensions	nvarchar	Dimensão e suporte
25	GenreForm	nvarchar	Tipologia e suporte
26	GeogName	nvarchar	Localidade
27	PhysFacet	nvarchar	Aspecto físico
28	MaterialSpec	nvarchar	Detalhes físicos específicos
29	LangMaterial	nvarchar	Idioma e escrita
30	Author	nvarchar	Autor intelectual
31	Recipient	nvarchar	Destinatário
32	AuthorAddress	nvarchar	Morada
33	RecipientAddress	nvarchar	Morada destinatário
34	Scrivener	nvarchar	Escrivão
35	Notary	nvarchar	Notário
36	ScopeContent	nvarchar	Âmbito e conteúdo
37	BiogHist	nvarchar	História administrativa/biográfica/familiar
38	OtherFindAid	nvarchar	Instrumentos de pesquisa
39	Note	nvarchar	Notas
40	RelatedMaterial	nvarchar	Unidades de descrição relacionadas
41	PhysTech	nvarchar	Características físicas e requisitos técnicos
42	AcqInfo	nvarchar	Fonte imediata de aquisição ou transferência
43	Arrangement	nvarchar	Sistema de organização
44	CustodHist	nvarchar	História custodial e arquivística
45	AltFormAvail	nvarchar	Existência e localização de cópias
46	Appraisal	nvarchar	Avaliação e selecção
47	AccessRestrict	nvarchar	Condições de acesso
48	LegalStatus	nvarchar	Estatuto Legal
49	Accruals	nvarchar	Ingressos adicionais
50	UseRestrict	nvarchar	Condições de reprodução
51	ProcessInfo	nvarchar	Nota de edição

51	ProcessInfo	nvarchar	Nota de edição
52	DescRules	nvarchar	Regras ou convenções
53	PreferCite	nvarchar	
54	SeparatedMaterial	nvarchar	Material separado
55	Abstract	nvarchar	Resumo
56	Repository	nvarchar	Entidade detentora
57	FilePlan	nvarchar	Planos de classificação
58	ContainerTypeTermID	bigint	Tipo u.i.
59	OtherDescriptiveData	nvarchar	Notas de migração
60	PhysLoc	nvarchar	Cota atual
61	PreviousLoc	nvarchar	Cota antiga
62	OriginalsLoc	nvarchar	Existência e localização de originais
63	OriginalNumbering	nvarchar	Cota original
64	Creator	nvarchar	Criado por
65	Created	datetime	Data de criação
66	Modified	datetime	Última modificação
67	Username	nvarchar	Alterado por
68	AssociationCount	int	
69	Custom1	nvarchar	Título paralelo
70	Custom2	nvarchar	
71	Custom3	nvarchar	
72	Custom4	nvarchar	Datas de acumulação
73	Custom5	nvarchar	Suporte
74	Custom6	nvarchar	Autor material
75	Custom7	nvarchar	Colaborador
76	Custom8	nvarchar	Funções, ocupações e actividades
77	Custom9	nvarchar	Mandatos/fontes de autoridade
78	Custom10	nvarchar	Estrutura interna/genealogia
79	Custom11	nvarchar	Contexto geral
80	Custom12	nvarchar	Tradição documental
81	Custom13	nvarchar	Tipologia documental
82	Custom14	nvarchar	Marcas
83	Custom15	nvarchar	Monogramas
84	Custom16	nvarchar	Selos
85	Custom17	nvarchar	Inscrições
86	Custom18	nvarchar	Assinaturas
87	Custom19	nvarchar	Eliminação
88	Custom20	nvarchar	Datas de eliminação
89	Custom21	nvarchar	Escrita
90	Custom22	nvarchar	Produtor
91	Custom23	nvarchar	Roda AIP ID
92	Custom24	nvarchar	
93	Custom25	nvarchar	
94	Custom26	nvarchar	
95	Custom27	nvarchar	
96	Custom28	nvarchar	
97	Custom29	nvarchar	
98	Custom30	nvarchar	
99	UnitIdToOrder	nvarchar	
##	ViewCount	bigint	
##	UserReaders	nvarchar	
##	ProfileReaders	nvarchar	
##	DescriptionLevelToOrder	int	
##	ThumbnailImageVaultID	nvarchar	
##	FNAAProducingEntityID	nvarchar	Identificador da entidade produtora
##	FNAAHoldingEntityID	nvarchar	Identificador da entidade detentora
##	FNAAFunctionID	nvarchar	Função
##	FNAACHRID	nvarchar	Registos patrimoniais de classificação
##	CompleteUnitIdToSort	nvarchar	

Figura 4.1: Colunas da tabela Components

4.2 Dados utilizados no treino dos modelos – 5 datasets (língua portuguesa)

Utilizar datasets para treinar modelos de reconhecimento de entidades é fundamental [36] porque esses modelos dependem de dados de treino para aprender a identificar e classificar diferentes entidades num texto. Os datasets fornecem exemplos rotulados de texto nos quais as entidades de interesse foram identificadas e anotadas. Esses dados de treino permitem que o modelo aprenda a reconhecer os padrões linguísticos associados a essas entidades e, assim, façam previsões precisas em novos textos.

Em primeira instância sabemos que os dados de treino que queremos utilizar necessitam de suportar a língua portuguesa uma vez que os dados do Digitarq estão nesta língua. Em segunda

instância necessitávamos que estes datasets conseguissem reconhecer três entidades: pessoas (PER), ogranizações (ORG) e localizações (LOC). Uma vez que a DGLAB não tem datasets já pré anotados optamos por utilizar 5 diferentes datasets que encontramos através de alguma pesquisa.

Em todos os datasets apresentados apenas foram consideradas quatro labels que todos tinham presentes (person - PER, localização - LOC, organização - ORG e 0 - sem identificação). Todos os datasets foram divididos percentualmente entre dataset de treino, dataset de teste e dataset de validação.

4.2.1 WikiNEuRal

O dataset WikiNEuRal [1] é um conjunto de dados de alta qualidade para reconhecimento de entidades nomeadas multilíngues. O conjunto de dados WikiNEuRal é gerado por meio de uma técnica que utiliza uma base de conhecimento lexical multilíngue (BabelNet) e arquiteturas baseadas em transformadores (BERT) para produzir anotações de alta qualidade para NER multilíngue. O dataset suporta as 9 línguas cobertas (de, en, es, fr, it, nl, pl, pt, ru).

- **Análise do dataset:**

  Labels: 'O': 0, 'B-PER': 1, 'I-PER': 2, 'B-ORG': 3, 'I-ORG': 4, 'B-LOC': 5, 'I-LOC': 6, 'B-MISC': 7, 'I-MISC': 8

  Para conter as quatro labels mencionadas em cima foi necessário ajustar o dataset e conjunto de labels disponibilizadas pelo dataset.

- **Tamanho do dataset para a língua portuguesa:** 106 mil palavras, 2.53 milhões de tokens, 44 mil entidades do tipo PER, 17 mil entidades do tipo ORG, 112 mil entidades dpo tipo LOC

- **Valores definidos para cada dataset:** dataset de teste 10.2 mil linhas, dataset de treino 80.6 mil linhas, dataset de validação 10.1 mil linhas.

- **Exemplo:**

  Tokens :[”O”, ”seu”, ”subsolo”, ”,”, ”rico”, ”em”, ”extensas”, ”reservas”, ”de”, ”água”, ”,”, ”serviu”, ”em”, ”dada”, ”altura”, ”para”, ”abastecer”, ”a”, ”Fábrica”, ”da”, ”Pólvora”, ”de”, ”Barcarena”, ”.”]

  Ner_tags: [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 5, 6, 6, 6, 6, 0]

4.2.2 MultiNERD

O dataset MultiNERD [5] é um conjunto de dados multilíngue, multigênero e de granularidade fina utilizado para reconhecimento (e desambiguação) de entidades nomeadas. Suporta 10 idiomas e 15 categorias de NER e 2 gêneros textuais. As 15 categorias de NER são: Pessoa (PER), Localização (LOC), Organização (ORG), Animal (ANIM), Entidade biológica (BIO), Corpo celeste (CEL), Doença (DIS), Evento (EVE), Comida (FOOD), Instrumento (INST), Mídia (MEDIA), Planta

(PLANT), Entidade mitológica (MYTH), Tempo (TIME) e Veículo (VEHI). O conjunto de dados MultiNERD cobre 10 idiomas: chinês, holandês, inglês, francês, alemão, italiano, polonês, português, russo e espanhol.

- **Análise do dataset:**

  Labels: ”O”: 0,”B-PER”: 1,”I-PER”: 2,”B-LOC”: 3,”I-LOC”: 4,”B-ORG”: 5,”I-ORG”: 6,”B-ANIM”: 7,”I-ANIM”: 8,”B-BIO”: 9,”I-BIO”: 10,”B-CEL”: 11,”I-CEL”: 12,”B-DIS”: 13,”I-DIS”: 14,”B-EVE”: 15,”I-EVE”: 16,”B-FOOD”: 17,”I-FOOD”: 18,”B-INST”: 19,”I-INST”: 20,”B-MEDIA”: 21,”I-MEDIA”: 22,”B-PLANT”: 23,”I-PLANT”: 24,”B-MYTH”: 25,”I-MYTH”: 26,”B-TIME”: 27,”I-TIME”: 28,”B-VEHI”: 29,”I-VEHI”: 30,”B-SUPER”: 31,”I-SUPER”: 32,”B-PHY”: 33,”I-PHY”: 34

  Para conter as quatro labels mencionadas em cima foi necessário ajustar o dataset e conjunto de labels disponibilizadas pelo dataset.

- **Tamanho do dataset para a língua portuguesa:** 177 mil palavras (única informação mencionada)

- **Valores definidos para cada dataset:** foi necessário dividir o dataset em 20% para teste, 20% para validação e os restantes 60% para treino uma vez que este não vinha dividido.

- **Exemplo:**

  Tokens :[”Faz”, ”fronteira”, ”com”, ”Bagaladi”, ”,”, ”Calanna”, ”,”, ”Campo”, ”Calabro”, ”,”, ”Cardeto”, ”,”, ”Fiumara”, ”,”, ”Laganadi”, ”,”, ”Montebello”, ”Ionico”, ”,”, ”Motta”, ”San”, ”Giovanni”, ”,”, ”Roccaforte”, ”del”, ”Greco”, ”,”, ”Sant”, ””, ”Alessio”, ”in”, ”Aspromonte”, ”,”, ”Santo”, ”Stefano”, ”in”, ”Aspromonte”, ”,”, ”Villa”, ”San”, ”Giovanni”, ”.”]

  Ner_tags: [0, 0, 0, 3, 0, 3, 0, 3, 4, 0, 3, 0, 3, 0, 3, 0, 3, 4, 0, 3, 4, 4, 0, 3, 4, 4, 0, 3, 4, 4, 4, 4, 0, 3, 4, 4, 4, 0, 3, 4, 4, 0]

4.2.3 HAREM

O conjunto de dados HAREM [2] é um corpus em língua portuguesa comumente usado para tarefas de reconhecimento de entidades nomeadas. Ele inclui cerca de 93 mil palavras, de 129 textos diferentes, de vários gêneros e variedades linguísticas. O HAREM é uma competição de avaliação para o reconhecimento de entidades nomeadas em português. As entidades mencionadas (categorias) do Harem são as seguintes: PESSOA; ORGANIZACAO; LOCAL; TEMPO; VALOR; ABSTRACAO; ACONTECIMENTO; COISA; OBRA; OUTRO.

- **Análise do dataset:**

  Labels: ”O”, ”B-PESSOA”, ”I-PESSOA”, ”B-ORGANIZACAO”, ”I-ORGANIZACAO”, ”B-LOCAL”, ”I-LOCAL”, ”B-TEMPO”, ”I-TEMPO”, ”B-VALOR”, ”I-VALOR”, ”B-ABSTRACCAO”,

”I-ABSTRACCAO”, ”B-ACONTECIMENTO”, ”I-ACONTECIMENTO”, ”B-COISA”, ”I-COISA”, ”B-OBRA”, ”I-OBRA”, ”B-OUTRO”, ”I-OUTRO

Para conter as quatro labels mencionadas em cima foi necessário ajustar o dataset e conjunto de labels disponibilizadas pelo dataset.

- **Tamanho do dataset para a língua portuguesa:** 257 linhas
- **Valores definidos para cada dataset:** dataset de treino 121 linhas, dataset de teste 128 linhas, dataset de validação 8 linhas.
- **Exemplo:**

  Tokens :[”MONEY”, ”1”, ”O”, ”escritor”, ”Clive”, ”Cussler”, ”,”, ”autor”, ”das”, ”aventuras”, ”de”, ”Dirk”, ”Pitt”, ”,”, ”assinou”, ”um”, ”contrato”, ”de”, ”US”, ”$”, ”14”, ”milhões”, ”com”, ”a”, ”Simon”, ”&”, ”Schuster”, ”para”, ”a”, ”publicação”, ”de”, ”dois”, ”livros”, ”.”]

  Ner_tags: [0, 0, 0, 0, 1, 2, 0, 0, 0, 0, 0, 1, 2, 0, 0, 0, 0, 0, 9, 10, 10, 10, 0, 0, 3, 4, 4, 0, 0, 0, 0, 0, 0, 0]

4.2.4 Paramopama

O conjunto de dados Paramopama [4] [24] é um conjunto de dados rotulados para reconhecimento de entidades nomeadas em português. Ele contém 5.000 palavras anotadas com 9 tipos de entidades nomeadas diferentes, incluindo pessoas, organizações e locais. O conjunto de dados foi criado a partir de notícias em português e é útil para treinar modelos de NER em português.

- **Análise do dataset:**

  Labels: ”O”, ”PERSON”, ”LOCATION”, ”ORGANIZATION”, ”TIME

  Para conter as quatro labels mencionadas em cima foi necessário ajustar o dataset e conjunto de labels disponibilizadas pelo dataset.

- **Tamanho do dataset para a língua portuguesa:** 12 mil palavras, 310 mil de tokens, 7 mil entidades PER, 7 mil entidades ORG, 17 mil entidades LOC
- **Valores definidos para cada dataset:** foi necessário que dividir o dataset em 20% para teste, 20% para validação e os restantes 60% para treino uma vez que este não vinha dividido.
- **Exemplo:**

  Tokens :[”A”, ”bandeira”, ”nacional”, ”,”, ”adotada”, ”em”, ”1947”, ”,”, ”é”, ”baseada”, ”na”, ”bandeira”, ”do”, ”Congresso”, ”Nacional”, ”Indiano”, ”,”, ”desenhada”, ”por”, ”Pingali”, ”Venkayya”, ”.”]

  Ner_tags: [O, O, O, O, O, TEMPO, TEMPO, O, O, O, O, O, O, ORGANIZACAO, ORGANIZACAO, ORGANIZACAO, O, O, O, PESSOA, PESSOA, O]

4.2.5 LegalGLUE

O dataset Legal General Language Understanding Evaluation (LegalGLUE) [3] é uma coleção de datasets para avaliar o desempenho do modelo num conjunto diverso de tarefas de NLU jurídicas de maneira padronizada. Consiste em quatro datasets já existentes que cobrem três tipos de tarefas e um total de 23 idiomas diferentes. O objetivo do LegalGLUE é fornecer uma avaliação justa e abrangente do desempenho do modelo numa ampla variedade de tarefas de NLU jurídicas. O dataset LegalGLUE reconhece entidades como nomes de pessoas, organizações e locais.

- **Análise do dataset:**

  Labels: 'O', 'B-PESSOA', 'I-PESSOA', 'B-ORGANIZACAO', 'I-ORGANIZACAO', 'B-LOCAL', 'I-LOCAL', 'B-JURISPRUDENCIA', 'I-JURISPRUDENCIA', 'B-LEGISLACAO', 'I-LEGISLACAO', 'B-TEMPO', 'I-TEMPO'

  Para conter as quatro labels mencionadas em cima foi necessário ajustar o dataset e conjunto de labels disponibilizadas pelo dataset.

- **Tamanho do dataset para a língua portuguesa:** 52.400 palavras de treino e 5,000 de validação e de treino

- **Valores definidos para cada dataset:** foi necessário que dividir o dataset em 20% para teste, 20% para validação e os restantes 60% para treino uma vez que este não vinha dividido.

- **Exemplo:**

  Tokens :["Número", "do", "Acórdão", "ACÓRDÃO", "1160/2016", -", "PLENÁRIO", "Relator", "AUGUSTO", "NARDES", "."]

  Ner_tags: ["O", "O", "O", "JURISPRUDENCIA", "JURISPRUDENCIA", "O", "ORGANIZACAO", "O", "PESSOA", "PESSOA", "O"]

Por fim, realiza-se a concatenação de todos os datasets: datasets de treino, datasets de testes e datasets de validação de cada um destes cinco datasets de forma a criar um único datasets para utilizar no desenvolvimento do modelo neuronal direcionado para a tarefa de reconhecimento de entidades.

Capítulo 5

Metodologia

A metodologia do projeto encontra-se dividido em duas secções distintas: (1) migração dos dados antigos do Digitarq, (2) criação de uma API que consiga dar suporte à aplicação web do Digitarq.

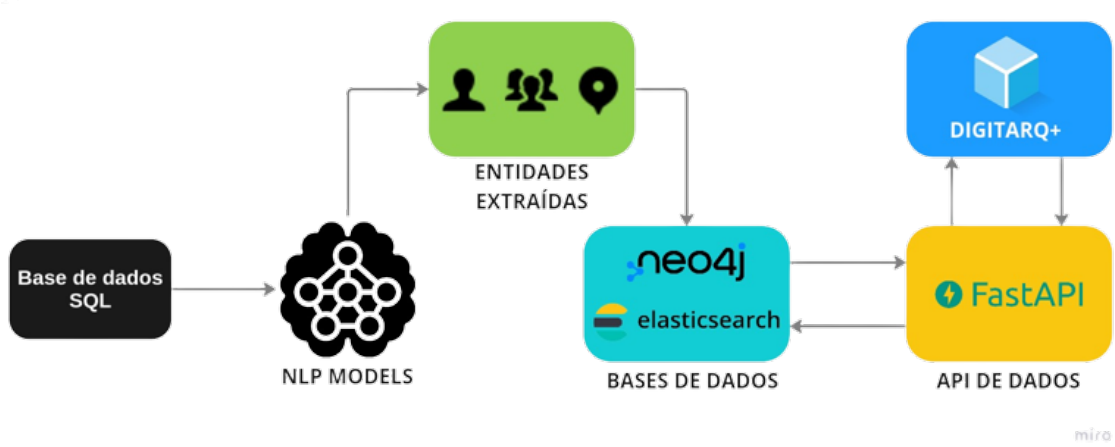

Figura 5.1: Metodologia do projeto

O projeto tem como esquema ilustrativo a figura 5.1, que representa as diferentes tarefas do mesmo.

No processo de migração dos dados antigos do Digitarq irão existir diferentes tarefas, uma primeira fase onde vai decorrer a extração dos dados das bases de dados SQL, juntamente com a análise e pré-processamento dos mesmos de modo a perceber que informação é necessária migrar e extrair para as novas bases de dados, de seguida uma segunda fase onde será desenvolvido um modelo de aprendizagem automática que permite realizar a tarefa de reconhecimento de entidades (NER), neste caso de três tipos de entidades, pessoas(PER), organizações(ORG) e localizações(LOC), que se pretende extrair. Após as entidades estarem extraídas, será realizado a importação dos dados extraídos nas bases de dados do Neo4j e do Elasticsearch. O processo de migração é realizado de forma individual para cada uma das 18 bases de dados incluídas no DGLAB.

No processo de criação de uma API serão utilizadas duas bases de dados para conseguir fa-

cilitar os pedidos necessários de criar. Para satisfazer pedidos de pesquisa (semântica e com parâmetros de pesquisa) é necessário, para além das bases de dados do Neo4j, de uma base de dados elasticsearch que providencie para estes tipos de pedidos uma resposta rápida e eficiente. Esta API serve para fazer a ligação necessária entre as bases de dados e a aplicação web seja para pedidos get (pesquisa), post (inserção), delete (remoção) ou put (atualização) como representado na figura 5.2.

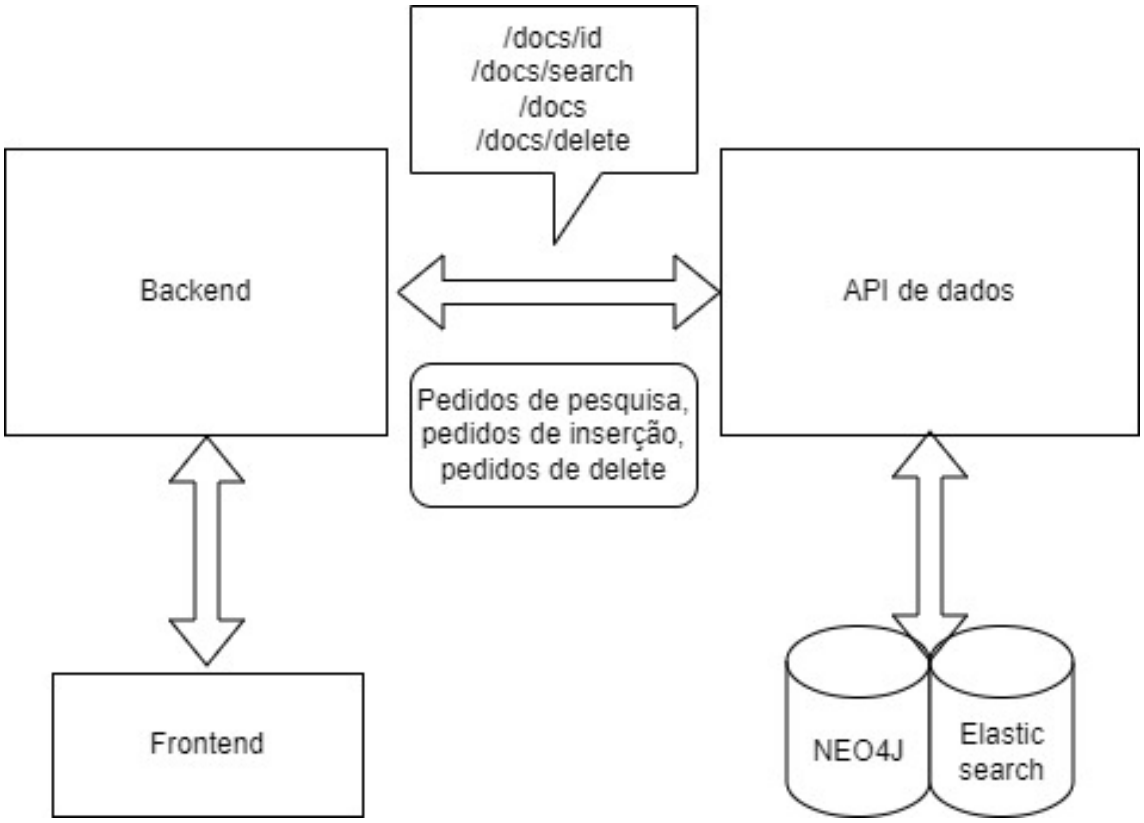

Figura 5.2: API do projeto

5.1 Conexão e exportação à base de dados antiga do digitarq

O primeiro passo da migração inicia-se pela conexão e posterior exportação dos dados do digitarq pertencentes às 18 base de dados relacionais do digitarq que pretendemos migrar. Serão extraídos apenas dados da tabela Components da base de dados.

O projeto lida com uma quantidade enorme de dados e informação relativos aos metadados dos documentos presentes nas bases de dados, para facilitar a migração dos mesmos serão utilizadas técnicas que permitem filtrar, e pré-processar os dados que eventualmente possam conter informação útil no processo de extração de entidades. Existem colunas com valores pré-formatados que não necessitam de qualquer extração podendo ser migrados sem qualquer validação.

- **Clustering dos dados:** Durante esta fase será feito o clustering e processamento dos dados exportados anteriormente. O objetivo nesta fase é descobrir padrões presentes nos dados

obtidos, para que eventualmente possa existir informação semântica útil para se extrair. o método de clustering que será usado é o k-means, onde posteriormente os resultados serão validados manualmente.

- **Verificação dos dados de forma manual:** De forma a auxiliar a tarefa anterior será também feita uma análise dos dados de forma manual, com intuito de tal como na tarefa anterior definirmos padrões que nos possam ser úteis no processo de reconhecimento de entidades.

- **Pré-processamento dos dados:** Os modelos de extração de entidades recebem um input textual para procederem à inferência e obtenção das entidades presentes no mesmo. De forma a facilitar este processo é preferível optar por pré processar o texto dos dados extraídos, retirando stopwords, espaços duplicados, caracteres e símbolos textuais que dificultem o modelo a fazer inferência durante a tarefa de reconhecimento de entidades.

5.2 Desenvolvimento do modelo de extração de entidades

O principal objetivo após exportação e análise dos dados é conseguir obter alguma informação semântica que fosse útil extrair, e de seguida com as entidades extraídas ser possível materializar as mesmas com base no CIDOC-CRM e por fim importar na base de dados de grafos do Neo4j.

Os documentos são semiestruturados geralmente, grande parte da sua informação textual segue a mesma estrutura. Como tal, é possível fazer uma extração baseada em expressões regulares. Por exemplo para extrair o nome do Réu num conjunto de documentos sobre crime:

```
reg_expression = r'(?<=[rR][eé]u:).*?(?=,)'
```

Todavia durante o projeto não é expectável que todos os documentos sigam uma estrutura rígida que permita uma extração através de expressões regulares. Como tal, vai ser considerado o uso de modelos neuronais para reconhecimento e extração de entidades.

Existem Modelos Neuronais já pré-treinados, a grande maioria para língua inglesa, que dado um input textual fazem a extração das entidades. Os principais desafios na extração vão ser:

- A necessidade de a extração ser modelada para conteúdo na língua portuguesa

- E conseguir mapear para além de entidades gerais que a maioria dos modelos neuronais existentes têm, conseguir também a modelagem para entidades específicas.

Os modelos neuronais permitem maior generalização, eficácia na captação de padrões complexos, porém implicam alguns requisitos ao nível da infraestrutura, tempos de treino, avaliação no processo de modelação, e também são dependentes da qualidade dos dados.

Nesta fase será desenvolvido por isso um modelo neuronal, através do treino de modelos pré treinados já existentes como é o caso do BERT e do T5. O modelo que é pretendido desenvolver tem como tarefa principal realizar reconhecimento de entidades (NER). Durante esta tarefa será possível identificar três tipos de entidades pessoas (PER), organizações (ORG) e localizações

(LOC). Para isso, será feito treino e afinamento de um modelo de aprendizagem automática com datasets anotados 4.2.

- **Extração de localizações:** Nesta fase será feita a extração e verificação através de thresholds (valores de confiança mínimo dos resultados da localização obtidos através da API) das localizações extraídas de modo a filtrar aquelas que não tenham relevância no processo de extração. O objetivo com a extração de localizações é identificar o endereço, a latitude e a longitude das mesmas. Para isso será procedido um processo de desambiguação que utilizará uma API de pesquisa de localizações denominada OpenStreetMap 2.13.

- **Identificação de localizações através de uma API de pesquisa (Desambiguação):** Nesta fase será utilizado o OpenStreetMap que permitirá extrair dados relevantes da localização como endereço, latitude e longitude. Isto também permitirá desambiguar as localizações uma vez que caso não se obtenham resultados no processo de pesquisa será possível considerar que a localização não existe.

- **Extração de pessoas e organizações:** Nesta fase será feita a extração de entidades do tipo pessoa e organização uma vez que são entidades que fazem sentido identificar para importar na base de dados de grafos. Para isso serão utilizados thresholds que permitem fazer a validação das mesmas.

5.3 Importação

Na fase de importação serão utilizados os dados exportados e analisados das bases de dados relacionais antigas do Digitarq.

- **Modelo com as classes do Neo4j:** O objetivo após a extração de entidades é conseguir atribuir as entidades identificadas a um nó específico para ser possível assim construir a base de dados de grafos. Para esse efeito de acordo com o Modelo de Referência Conceitual do CIDOC serão atribuídas aos três tipos de entidades extraídas a classe respetiva.

  Nesta fase serão classificadas individualmente cada coluna da tabela *Components* da base de dados exportada com a sua respetiva classe pertencente ao Modelo de Referência Conceitual do CIDOC. De seguida serão também criadas relações entre as classes identificadas de acordo uma vez mais com o CIDOC.

  A identificação de classes terá em consideração os resultados obtidos durante o processo de extração de entidades.

  Nesta fase para ser possível saber a que classe pertence cada metadado do documento é necessário realizar uma validação juntamente com a DGLAB, para que seja feita a atribuição de classes e relações entre classes que se pretende configurar.

- **Funções de inserção de dados na base de dados do Neo4j:** Após a identificação das classes (nós) e relações entre as colunas da tabela *Components* da base de dados exportadas

serão desenvolvidas funções que permitem a inserção destes grafos na base de dados de grafos do Neo4j.

- **Bases de dados do Elastic search:** Para ser possível responder de forma eficiente aos pedidos que eventualmente a aplicação web do Digitarq é necessário a importação também dos dados numa base de dados do elasticsearch uma vez que são base de dados que respondem de forma rápida e eficiente a pedidos de pesquisa simples.
- **Bases de dados do Neo4j:** Utilizando o modelo onde foram definidas as classes e as relações entre as colunas dos dados exportados será agora após classificação dos mesmos utilizadas as funções de inserção para importar os dados na nova base de dados de grafos do Neo4j.

5.4 API - conexão entre as bases de dados e a aplicação web

Por fim, para ser criada a ligação entre as bases de dados e a aplicação web é necessário criar uma API que englobe todos os pedidos necessários para que assim seja realizada esta ligação.

- **Criação de endpoints:** Nesta fase serão construídos endpoints que correspondam às necessidades do funcionamento da página web.
- **Criação de funções de get, post e insert às bases de dados:** Para conectar os pedidos da API às bases de dados existentes de acordo com o pedido que for executado

Capítulo 6

Resultados e Discussão

Neste capítulo do trabalho, os resultados obtidos após a implementação são analisados e discutidos, com o intuito de verificar se foram gerados resultados valiosos em cada uma das experiências realizadas. Além dos resultados, este capítulo também apresenta as razões que provavelmente levaram à melhoria ou retrocesso dos valores de desempenho.

Este capítulo também tem como objetivo explicar cada uma das experiências analisadas, pois diferentes cenários são realizados por diferentes motivos, podendo ser para verificar se uma mudança nos dados ou no método que leva a melhorias ou simplesmente para verificar se uma mudança de implementação que parece óbvia tem um impacto positivo ou negativo no desempenho.

6.1 Conexão e exportação à base de dados antiga do digitarq

O objetivo inicial foi realizar a conexão às bases de dados antigas do Digitarq. As bases de dados às quais se pretende fazer a ligação são bases de dados do tipo relacional MicrosoftSQL Server, para esse efeito foi criado um script que permite fazer a mesma.

Foi criado um script que permite fazer a conexão à base de dados, através de uma biblioteca do python concebida para fazer conexão às base de dados relacionais MicrosoftSQL Server, o *pymssql*, uma interface de base de dados simples que se baseia no FreeTDS para fornecer uma interface Python DB-API (PEP-249) para o Microsoft SQL Server.

Esta biblioteca permitiu fazer a conexão individual a cada uma das bases de dados através da inicialização de variáveis tais como IP do server, número da porta, o username, a password, e o nome da base de dados.

Após a conexão é possível extrair os dados sem qualquer tipo de restrição utilizando uma chamada SQL, para obter todos os dados pertencentes à tabela *Components* da respetiva base de dados.

```
conn = pymssql.connect(server, port, username, password, database)
cursor = conn.cursor(as_dict=True)

cursor.execute('SELECT {} FROM {} WHERE {} ORDER BY',
        column_db, table_db, order_db)
```

```
conn.close()
```

6.1.1 Clustering dos dados

Antes de ser iniciada a extração de entidades é necessário identificar quais as colunas referentes aos metadados do documento que continham informação semântica útil que necessite de ser extraída. Para esse processo foi necessário recorrer ao clustering das colunas.

Juntamente com a DGLAB foi possível distinguir as colunas que têm metadados formatados, onde não é necessário realizar clustering das mesmas, das que têm metadados em cadeias de string sem qualquer tipo de formatação específica. Através desta análise foi facilitado o processo de clustering pois não é necessário percorrer o total de colunas.

Colunas que necessitam da realização de clustering devido a sua densidade textual: 'Scope-Content' e 'UnitTitle'.

O método utilizado para o processo de clustering foi através de um dos algoritmos mais populares nesta tarefa, o k-means, foi construído um script no qual realizaram-se vários testes com valores de cluster (k) entre 2 e 800.

De modo a obter melhores resultados dentro do script, foi pré processado o texto com o objetivo de eliminar todas as *stopwords* ('de', 'a', 'o', 'que', 'e', 'do', 'da', 'em', 'um', 'para'), para o sucedido foi utilizada a biblioteca nltk do python.

Já com os dados pré processados foi utilizada a classe *TfidfVectorizer* classe que converte o dataframe numa matriz TF-IDF, onde a cada palavra é lhe atribuída um valor TF-IDF, que é uma medida estatística que tem o intuito de indicar a importância de um documento em relação a uma coleção de documentos.

Por conseguinte, foi utilizada a classe k-means que basicamente faz o clustering dos dados, ou seja, agrupa os dados separando as amostras em n grupos de variância igual.

Finalmente, foram exibidos os resultados num excel com três colunas, as colunas não formatadas com os metadados do documento, o número do cluster a que pertence o documento, e por fim as top n palavras mais importantes no documento, isto foi feito com o intuito de ser possível visualizar melhor os resultados obtidos e a partir daí perceber se existe alguma informação semântica útil de etiquetar.

6.1.2 Verificação dos dados de forma manual

Nesta secção é verificado detalhadamente de forma manual os resultados obtidos após o clustering de forma a obter que tipo de informação é possível extrair destas colunas que contém alguma densidade textual.

Após alguma análise detalhada dos resultados obtidos, observou-se que para uma extração inicial é possível com facilidade extrair três tipos de entidades (classes do CIDOC-CRM): ACTOR, GROUP, and PLACE. Tendo em conta os resultados desta validação e sabendo que o tipo de classes que normalmente os modelos de extração de entidades conseguem extrair (PER - pes-

soas, LOC - localizações, ORG - organizações), é possível concluir que as três classes do CIDOC identificadas em resultados obtidos no processo de clustering estão diretamente relacionadas com entidades normalmente etiquetadas por modelos neuronais.

Por exemplo para o seguinte texto foi possível observar a partir do clustering os resultantes presentes na figura 6.1:

”As fotografias reunidas neste conjunto são reveladoras da actividade presidencial do general **Craveiro Lopes** entre os anos de 1951 e 1958. Neste período fez visitas ao **Brasil**, à **Madeira**, Açores, Guiné, Cabo Verde, São Tomé, **Angola**, **Moçambique**, Madrid, Guimarães, Évora, Porto, Mem Martins-Algueirão, Vila Nova de Famalicão, Moura, Hidro-Eléctrica do Cávado, **Hidro-Eléctrica do Douro**, Instituto de Medicina Tropical, Instituto de Socorros a Náufragos, Fundação Nacional para a Alegria no Trabalho, feiras, exposições, barragens e palácios e participou em congressos, manobras militares e funerais. Colaboraram na constituição destes álbuns os seguintes fotógrafos e casas fotográficas: Ismael e Beatriz Ferreira. Fotógrafos da Imprensa, Agência Nacional de Actualidades Fotográficas, Fotos de Monteiro e Machado, Foto-Cetóbriga de Américo Ribeiro, A. Valente, **Foto Beleza**, Foto-Cine de Abílio Fernandes, Fotarte de M. Teixeira, Fotografia Alvão, **Póvoa & Pinto**, **Firmino dos Santos**, **Paul Popper Ltd.**, e F. Marques da Costa.”

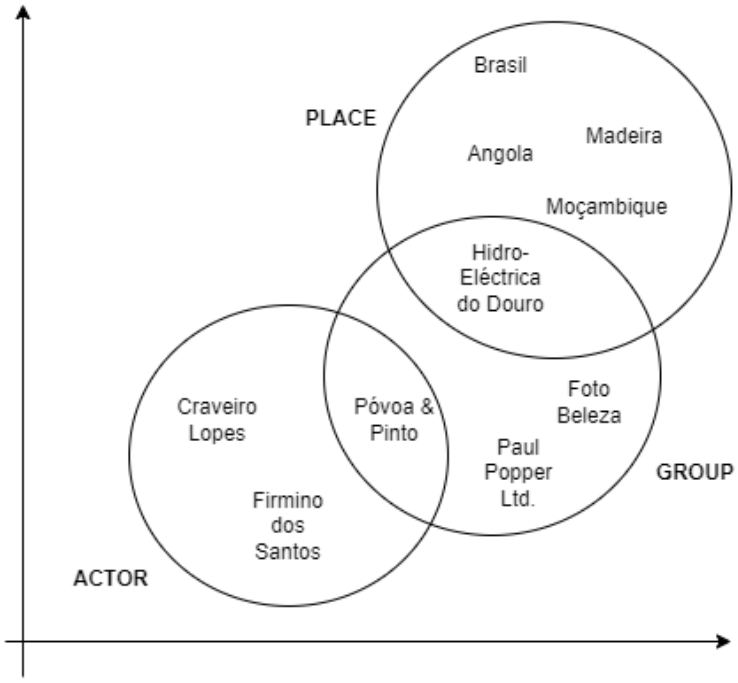

Figura 6.1: Exemplo de clustering

6.1.3 Pré-processamento dos dados

O pré-processamento de dados é uma etapa essencial em modelos de reconhecimento de entidades. Existem várias razões pelas quais o pré-processamento é necessário antes de realizar inferências num modelo de reconhecimento de entidades. Logo antes de se proceder à criação do modelo

foram pré processados os dados exportados.

O processo iniciou-se pela limpeza dos dados 2.2, através da remoção de caracteres especiais 6.2, remoção de espaços extras, e de stop-words, melhorando assim a qualidade dos mesmos. Este foi o único pré processamento que foi feito uma vez que as outras possibilidades de pré processamento não vinham ao acordo das necessidades.

Figura 6.2: Exemplo de pré-processamento (remoção de caracteres especiais)

6.2 Desenvolvimento do modelo de extração de entidades

No decorrer do desenvolvimento do modelo de extração de entidades foram avaliadas duas arquiteturas distintas com o intuito de perceber qual delas daria mais valias e melhores resultados. A primeira arquitetura baseada no modelo de processamento de linguagem natural BERT da Google mais concretamente o bert-base-multilingual-cased e a segunda arquitetura baseada no modelo T5 da google mais concretamente o mt5-base.

Para ambas as arquiteturas o processo de desenvolvimento do modelo neuronal foi idêntico. A única tarefa onde ocorreram ligeiras alterações foi no treino do mesmo.

6.2.1 Objetivo e datasets do modelo neuronal

O propósito do modelo neuronal que se pretende construir é que ele consiga realizar a tarefa de **reconhecer entidades**, tendo em conta que terá que extrair essas entidades de texto em língua portuguesa e que terá que conseguir obter três tipos de entidades: Pessoas (PER), Organizações (ORG), e Localizações (LOC).

O passo seguinte será coletar dados anotados que permitam treinar e avaliar o modelo. Todavia, no decorrer da implementação foi reconhecido que não existem dados anotados relativos aos dados do DIGITARQ então foi necessário optar por encontrar datasets anotados open source que possibilitassem a realização da tarefa de treino do modelo neuronal. Para isso, foram utilizados cinco datasets em língua portuguesa preparados para treinar modelos com o objetivo de extrair entidades dos três tipos de entidades que se pretendem reconhecer 4.2.

Todos estes datasets contêm suporte na língua portuguesa e incluem as três entidades que se pretendem identificar (pessoas, localizações e organizações). Todos os datasets foram pré-processados de modo a só conter apenas os dados relativos às três entidades que queremos utilizar. Após serem colocados todos os datasets no mesmo formato concatenamos para formar um só dataset. Os datasets já se encontram divididos entre dataset de treino, de validação e de testes.

Antes destes datasets serem utilizados no processo de treino do modelo neuronal foi necessário tokenizar os datasets. Ou seja, utilizar uma função que divida os caracteres que formam os datasets em tokens para que estes possam ser utilizados pelo modelo, que utiliza tokens ao invés de frases. Para esse efeito foi utilizada a função AutoTokenizer dos transformers que permite fazer essa divisão por tokens e depois foi necessário ajustar as restantes colunas do dataset para estarem de acordo com a passagem de tokens em vez de palavras.

6.2.2 Treino do modelo neuronal

No treino do modelo utilizaram-se máquinas do google colab para que fosse possível usufruir de GPU´s, uma vez que aceleram o processo de treino de modelos neuronais. Apesar da tarefa de treino ser semelhante para ambas as arquiteturas, existem algumas divergências.

Para além do facto de ter sido treinado o modelo para duas arquiteturas diferentes também foram feitas duas comparações onde antes de serem treinados com os cinco datasets optei por num dos casos fazer um pré-treino com dados não anotados exportados das bases de dados antigas do Digitarq com o objetivo de dar a conhecer ao modelo o texto sobre o qual iria realizar a extração de entidades. O pré-treino foi realizado com cerca de quinhentos mil documentos do Digitarq individualmente para cada uma das arquiteturas.

No total foram treinados quatro modelos neuronais, dois por cada arquitetura. Durante este desenvolvimento utilizei ferramentas da biblioteca tranformers do python desenvolvida pelo HuggingFace que permitiu utilizar modelos de linguagem pré-treinados para realizar uma variedade de tarefas relacionadas.

No desenvolvimento dos modelos neuronais para ambas as arquiteturas, utilizei a classe ”Trainer”da biblioteca transformers, esta classe permite a configuração e o treino de modelos pré-treinados para tarefas específicas de NLP, neste caso a extração de entidades. Através desta classe consegui configurar diferentes parâmetros para a realização do treino do modelo, modelo que é treinado através da função train() desta classe:

BERT

No desenvolvimento do modelo neuronal baseado no modelo de processamento de linguagem natural bert-base-multilingual-cased foram configurados os parâmetros de treino da seguinte forma:

- Modelo Base (model) - modelo de linguagem pré-treinado a ser treinado neste caso o modelo bert-base-multilingual-cased carregado através da classe AutoModelForTokenClassification

- Argumentos de treino (args) - objeto onde definimos os argumentos de configuração para o treino.
- Dataset de treino ($train\_dataset$) - dataset de treino dos cinco datasets 4.2
- Dataset de avaliação ($eval\_dataset$) - dataset de avaliação dos cinco datasets 4.2
- Tokenizer (tokenizer) - tokenizer associado ao modelo pré-treinado. É responsável por pré-processar e tokenizar os dados de entrada, convertendo-os em sequências de tokens compreendidas pelo modelo. Neste caso é inicializado através da classe AutoTokenizer para o modelo bert-base-multilingual-cased
- Coletor de dados ($data\_collator$) - definir como os dados são agrupados e formatados antes de serem passados para o modelo para isso utilizamos a classe DataCollatorForTokenClassification
- Métricas ($compute\_metrics$) - função onde definimos as métricas que pretendemos calcular durante a avaliação do modelo

Nos argumentos de treino (args) é possível definir vários parâmetros de acordo com a forma como se pretende treinar o modelo, aqueles que defini para treinar o modelo neuronal baseado no modelo de processamento de linguagem natural bert-base-multilingual-cased foram:

- $output\_dir$: O diretório de saída onde os modelos treinados e outros artefatos serão salvos.
- num_train_epochs: O número total de épocas (iterações completas pelo conjunto de treino) para treinar o modelo. Definimos este valor como 1.
- $per\_device\_train\_batch\_size$: O tamanho do lote (batch size) para cada dispositivo de treino. Isso define quantos exemplos são processados simultaneamente durante o treino. Definimos o valor 16 uma vez que as máquinas do google colab tinham dificuldade em suportar valores superiores.
- $per\_device\_eval\_batch\_size$: O tamanho do lote (batch size) para cada dispositivo de avaliação. Isso define quantos exemplos são processados simultaneamente durante a avaliação. Definimos o valor 16 uma vez que as máquinas do google colab tinham dificuldade em suportar valores superiores.
- $learning\_rate$: A taxa de aprendizagem para o otimizador durante o treino. Isso controla o tamanho dos ajustes feitos nos pesos do modelo durante o processo de otimização. Neste modelo utilizamos o valor 2e-5.
- $weight\_decay$: Um termo de decaimento dos pesos, usado para evitar o overfitting durante o treinamento. O valor definido foi 0.01.

- $evaluation\_strategy$: A estratégia de avaliação do modelo durante o treinamento. Pode ser definida como ”no”, ”steps”ou ”epoch”, indicando se a avaliação deve ser realizada após um certo número de etapas ou a cada época. No nosso caso utilizamos a estratégia ”steps”.

- $eval\_steps$: O número de etapas (batches) entre cada avaliação do modelo durante o treino. De modo a conseguirmos avaliar durante diferentes etapas optamos por colocar um valor de 1000 etapas.

- $save\_steps$: O número de etapas (batches) entre cada save do modelo durante o treino. De modo a conseguir salvar várias versões do modelo treinado optamos por colocar o valor 10000.

Na função das métricas, função onde foram definidas as métricas que se pretendem calcular durante a avaliação do modelo, utilizei a biblioteca ”seqeval”para calcular essas métricas. Onde apenas se quis obter os valores de precisão, recall, F1 e accuracy.

T5

No desenvolvimento do modelo neuronal baseado no modelo de processamento de linguagem natural mt5-base foram configurados os parâmetros de treino da seguinte forma:

- Modelo Base (model) - modelo de linguagem pré-treinado a ser treinado neste caso o modelo bert-base-multilingual-cased carregado através da classe MT5EncoderForTokenClassification

- Argumentos de treino (args) - objeto onde definimos os argumentos de configuração para o treino.

- Dataset de treino ($train\_dataset$) - dataset de treino dos cinco datasets 4.2

- Dataset de avaliação ($eval\_dataset$) - dataset de avaliação dos cinco datasets 4.2

- Tokenizer (tokenizer) - tokenizer associado ao modelo pré-treinado. É responsável por pré-processar e tokenizar os dados de entrada, convertendo-os em sequências de tokens compreendidas pelo modelo. Neste caso é inicializado através da classe MT5Tokenizer para o modelo mt5-base

- Coletor de dados ($data\_collator$) - definir como os dados são agrupados e formatados antes de serem passados para o modelo para isso utilizamos a classe DataCollatorForTokenClassification

- Métricas ($compute\_metrics$) - função onde definimos as métricas que pretendemos calcular durante a avaliação do modelo

Nos argumentos de treino (args) e na função das métricas utilizei os mesmos valores e a mesma função que no modelo que seguiu a arquitetura BERT.

6.2.3 Avaliação e ajustes do modelo

Nesta secção utilizei o wandb 2.12 uma plataforma que auxilia na visualização e análise de métricas tais como a precision, o recall, o F1-score e a accuracy. Para avaliar o modelo foi utilizada a função evaluate() e para obter os resultados das métricas utilizei a função predict(test_tokenized) onde foi passado o dataset de testes tokenizado, estas funções pertencem à classe Trainer.

No processo de avaliação optei por comparar as quatro principais métricas que definem a capacidade de um modelo neuronal: a precisão, o recall, o F1-score e a accuracy, distinção entre métricas mencionada na secção 2.1.4. Foram avaliados em primeira instância as diferenças entre os modelos que tiveram um pré treino com dados não anotados do digitarq e aqueles que não tiveram.

(a) Recall (b) F1

Figura 6.3: Resultados de recall e f1

(a) Precision (b) Accuracy

Figura 6.4: Resultados da precision e accuracy

Resultados expressos em valores para cada modelo que conseguimos verificar através das figuras 6.3 e 6.4:

- Modelo T5 pré-treinado com dados do digitarq (cor roxa) - recall = 0.858, f1 = 0.842, precision = 0.826, accuracy = 0.979
- Modelo Bert pré-treinado com dados do digitarq (cor verde) - recall = 0.853, f1 = 0.843, precision = 0.833, accuracy = 0.979
- Modelo T5 (cor cinzenta) - recall = 0.651, f1 = 0.625, precision = 0.603, accuracy = 0.949

- Modelo Bert (cor castanha) - recall = 0.659, f1 = 0.642, precision = 0.626, accuracy = 0.955

Através da análise destes resultados foi possível perceber que existia uma diferença bastante considerável quando se procedia a um pré-treino do modelo com dados não anotados do digitarq, uma vez que o modelo assim tinha conhecimento sobre que dados iria treinar revelando valores métricos muito superiores.

Para além desta análise foi interpretado que para ambas as arquiteturas t5 e bert os resultados são semelhantes o que obriga a proceder para testes de inferência para ambos os modelos (roxo e verde) de forma a perceber qual destes tem melhores resultados com dados reais.

6.2.4 Testes de inferência do modelo

Na realização de testes de inferência do modelo utilizei um conjunto de dados das bases de dados do DIGITARQ exportadas para verificar a qualidade do modelo. Para esse efeito foi utilizada a classe pipeline da biblioteca Tranformers do Hugging Face que permite carregar o modelo neuronal que se quer e passar-lhe um input para obter os resultados de acordo com a tarefa do modelo.

Estes testes foram realizados para os modelos das duas arquiteturas que foram pré-treinados com dados não anotados do digitarq, ou seja, os dois modelos com melhores resultados obtidos na avaliação de métricas.

Figura 6.5: Exemplo do resultado de um teste de inferência

Os testes foram realizados para diferentes conjuntos de dados retirados diretamente da base de dados do DIGITARQ. Um dos exemplos, é possível observar através da figura 6.5, onde é dado como input de entrada um conjunto de texto e onde se obtém como output as entidades extraídas com o respetivo tipo de entidade. Por exemplo, ”Craveiro Lopes”foi etiquetado como sendo do tipo ”PER”(person). Para esta frase obtivemos resultados exatamente iguais para ambas as arquiteturas

Após análise de diversos resultados foi possível concluir que os resultados eram bastante semelhantes, porém optei pelo modelo da arquitetura bert que em comparação com a arquitetura t5 tem mais utilização no que toca a modelos de reconhecimento de entidades.

6.3 Extração de localizações

Nesta secção é referido o processo de extração de localizações. Referindo dois processos de validação: um onde são definidos thresholds e outro de desambiguação através do recurso a uma API de pesquisa de localizações

6.3.1 Validação das entidades extraídas

Após o reconhecimento das entidades através do modelo serão validadas as entidades extraídas. O modelo quando realiza inferência retorna um array constituído pelas entidades, onde cada entidade indica: o tipo de entidade (LOC, PER, ORG), o score (valor de 0 a 1 que indica a confiança que o modelo tem na atribuição do tipo de entidade mencionada), o start e o end (valores que indicam o token onde inicia e onde termina a entidade extraída na frase dada ao modelo).

A validação inicial que é feita através do score, onde foi definido um valor de threshold que permite filtrar entidades que tenham um score, uma certeza superior a um valor compreendido entre 0 e 1. Para se saber que valor devo colocar foi realizada uma análise manual de modo a verificar a partir de que valor faria mais sentido guardar essa entidade extraída. Para esse efeito foram realizadas validações manuais para diferentes valores, como por exemplo 65%, 75% e 85%, para os quais, como é possível ver nas figuras 6.6, 6.7 e 6.8, foi verificado que percentagens inferiores a 70% ainda registavam entidades que não faziam sentido e tinhas uma validação pouco robusta, foi possível verificar que para valores superiores a 80% existiam entidades que acabavam por não ser consideradas, entidades estas que faria sentido guardar.

O valor pelo qual optei foi 0.75 ou seja apenas entidades com uma percentagem de certeza de 75%. Esta análise foi realizada através de testes de inferência com diferentes valores de threshold para diferentes exemplos reais de dados do digitarq.

Para além desta filtragem foi realizada desambiguação as localizações de forma que a extração de entidades contivesse a menor quantidade de lixo possível.

Figura 6.6: Exemplo quando Threshold igual a 65%

Figura 6.7: Exemplo quando Threshold igual a 75%

Figura 6.8: Exemplo quando Threshold igual a 85%

6.3.2 Identificação de localizações através de uma API de pesquisa (Desambiguação)

No processo de desambiguação de localizações recorri a uma API de pesquisa denominada por Nominatim, que permite através de um input realizar uma pesquisa de localização. A pesquisa permite obter os seguintes resultados: o endereço da localização, as coordenadas da mesma, e o score (valor entre 0 e 1 que define a certeza da própria pesquisa).

As entidades extraídas onde não são obtidos resultados de pesquisa através da API não são guardadas. Para melhorar ainda mais a qualidade dos dados extraídos é definido também um threshold de acordo com o score da API. Para isso realizei uma análise manual para perceber que valor faria sentido filtrar os resultados. Foi definido o valor 0.65, ou seja, apenas resultados da pesquisa da API que tenham uma certeza de 65% são guardados.

Foram realizadas duas pesquisas uma por Lisboa e outra por Campo Grande como é possível visualizar pelos resultados da tabela 6.1, onde realizei duas pesquisas uma por Lisboa e outra por Campo Grande. O resultado dessa pesquisa permite obter diferentes parâmetros de resultados (endereço, coordenadas, tipo de endereço, e importância(score)). Nesta fase de desambiguação o parâmetro a importância (score) foi o parâmetro utilizado para validar a certeza da localização pesquisada na API. Visualizando os resultados da tabela uma pesquisa por Lisboa tem uma importância de 71,5%, seguindo o threshold definido é possível conferir e ter certeza que a pesquisa da localização por parte da API tem uma percentagem elevada. Por outro lado, na pesquisa por Campo Grande verifica-se que a importância é apenas de 49,9%, seguindo o threshold definido conferi que não é possível ter uma certeza elevada relativamente a esta pesquisa uma vez que ”Campo Grande”existem muitos resultados possíveis.

Após esta filtragem foram importadas as entidades que sobraram na base de dados do neo4j e

do elastic search.

Exemplos de resultados obtidos pela API		
	Lisboa	**Campo Grande**
Latitude	38.7440523	-20.4640173
Longitude	-9.151827931741947	-54.6162947
Importância (score)	0.715	0.499
Endereço	Lisboa, Portugal	Campo Grande, Região Geográfica Imediata de Campo Grande, Região Geográfica Intermediária de Campo Grande, Mato Grosso do Sul, Região Centro-Oeste, Brasil
Tipo de endereço	Cidade	Munícipio

Tabela 6.1: Exemplos de resultados obtidos pela API

6.4 Modelo com as classes do Neo4j

O objetivo com a migração de dados para uma base de dados de grafos prende-se com a capacidade de criar uma base de dados que possibilite o armazenamento de nós classificados, o relacionamento e a navegação entre os mesmos. Seguindo por base o Modelo de Referência Conceitual do CIDOC, através do Neo4j foi possível definir as classes que identificam cada nó e também as relações entre classes.

Os dados que se pretendem importar do Digitarq são metadados dos documentos e partimos do documento no que diz respeito à criação de classes e relações. Seguindo o anexo **notas_mapeamento.pdf**, sabemos que quando se refere à descrição arquivística representada no Digitarq diz respeito tanto à dimensão material (física) como conceptual das unidades de descrição (documentos). Ou seja, quando se pretende guardar os metadados de um documento sei que terei que dividir o mesmo em duas classes visualizadas na figura 6.9:

- Classe que representa o nó físico do documento - classe do CIDOC E24 Physical Human-Made Thing
- Classe que representa o nó conceptual do documento - classe do CIDOC E28 Conceptual Object

- Para além destas duas classes vamos ter também uma classe que representa o nó de descrição do documento - classe do CIDOC E31 Document

Os nós referidos em cima vou relacionar de acordo com as relações presentes no modelo CIDOC. Para cada nó estarão associados os metadados referentes à parte física, conceptual ou descritiva do documento de acordo com o anexo **classe_neo4j.pdf**

- **Localizações extraídas:** Relativamente às localizações extraídas a classe que define o nó da mesma é a E53 Place, que estará associada ao metadado no qual foi extraída. Por sua vez ao nó E53 Place estarão associados nós referentes às características da localização: endereço e coordenadas.

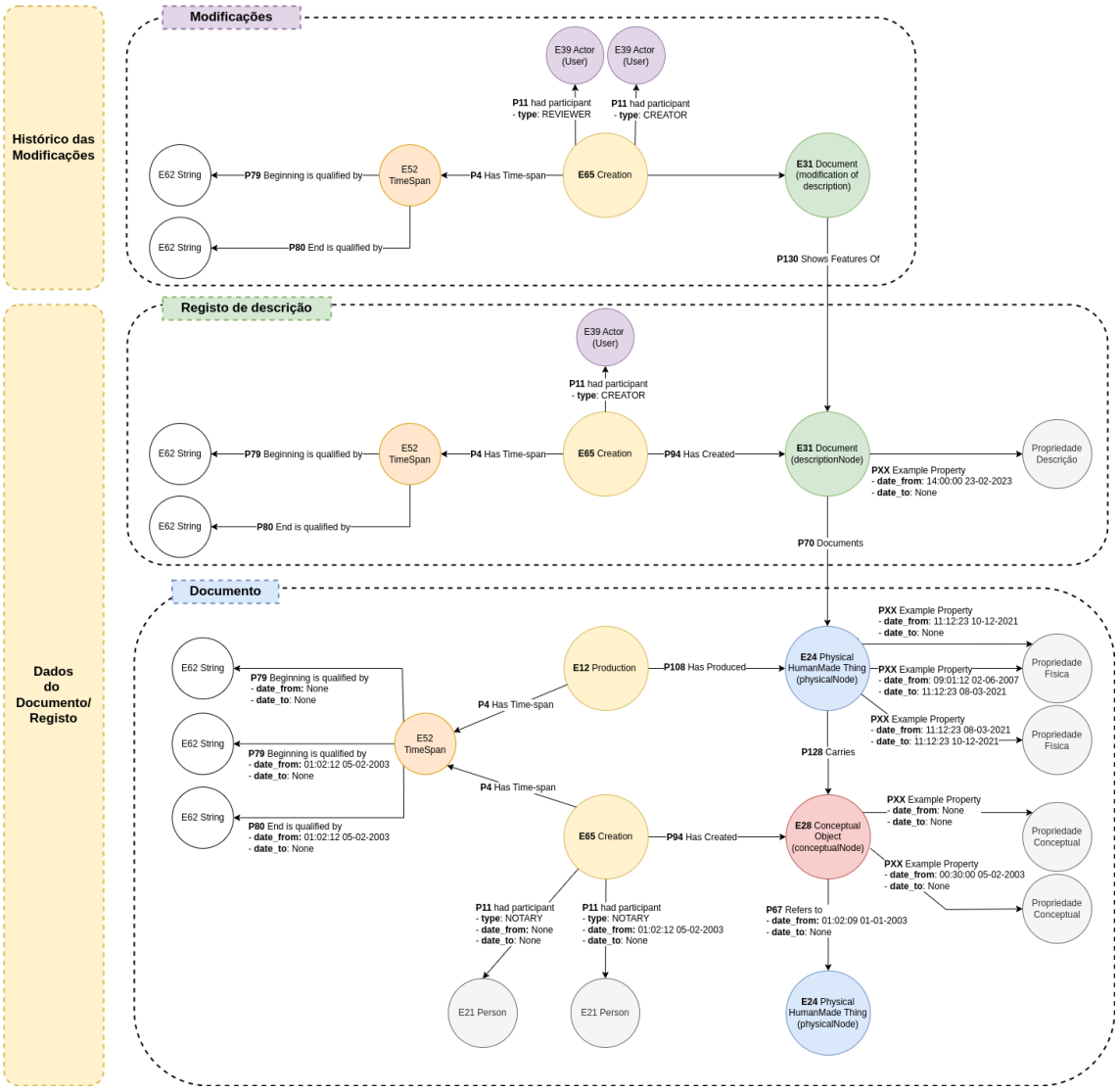

Figura 6.9: Divisão das diferentes classes do documento

6.5 Funções de inserção de dados na base de dados do Neo4j

Ao definir classes e relações para os elementos do documento foi necessário criar funções de inserção na base de dados nova do Neo4j. Tendo em conta a informação obtida anteriormente, é possível saber a associação que se deve fazer entre cada coluna dos metadados exportados das bases de dados do Digitarq e um dos três nós principais de cada documento. Com base nesta informação foi criada individualmente para cada coluna da base de dados antiga do Digitarq uma função de inserção com as classes e relações necessárias para a introdução da mesma.

```
def insertIDParentIDRootParentID(identifier, linguistic_node, physical_node, key):
    try:
        identifier_node = E42_Identifier.nodes.get(name=identifier)
    except:
        identifier_node = E42_Identifier(name=identifier).save()
    try:
        type_node = E55_Type.nodes.get(name=type)
    except:
        type_node = E55_Type(name=type).save()
    linguistic_node.identified_by.connect(identifier_node)
    identifier_node.has_type.connect(type_node)
```

Figura 6.10: Exemplo de função de inserção das colunas ID, ParentID e RootParentID

Como é possível verificar na figura 6.10, neste caso para as colunas ”ID”, ”ParentID”e ”Root-ParentID”as classes e as relações definidas eram iguais por isso ao invés de implementar funções independentes para cada coluna, foi possível implementar apenas uma comum às três.

- Como argumentos de entrada foram definidos de forma comum a todas as tabelas, o conteúdo da coluna, o nó de descrição do documento, o nó físico do documento e a key de identificação.
- Criação do nó (caso não exista) com o conteúdo da coluna, com a sua classe definida. Neste caso classe *E42 Identifier*.
- Criação do nó ou nós que serão relacionados. Neste caso apenas um nó da classe *E55 Type*.
- Por fim conectar através das relações implementadas na secção 6.4 os nós criados. Neste caso conexão do nó com o conteúdo da coluna e com o objeto linguístico do objeto através da relação *P1 identified by* e com o tipo associado através da relação *P2 has type*

6.6 Criação do script de migração

O desenvolvimento deste script tem como passo inicial a exportação de dados e a transformação dos mesmos referida nas secções anteriores, com uma ligeira diferença visto que para ser um processo mais eficiente e rápido ao invés de se obter um *dataframe* com os dados exportados, obtêm-se um *cursor*.

Como existem dezoito arquivos diferentes com bases de dados independentes este script tinha que ser corrido de forma independente para cada um, para este sucedido foram criados ficheiros de configuração json.

6.6.1 Criação dos ficheiros de configuração

Os ficheiros json de configuração criados contêm valores de configuração de acesso à base de dados: ip, port, username, password, database e table, independentes para cada um dos arquivos. E ao mesmo tempo existe um ficheiro comum que contém as configurações da base de dados em si com uma label que contém o nome da coluna e um value com a respetiva função de inserção para essa determinada coluna.

Os ficheiros de configuração são lidos, é feita a exportação dos dados da base de dados do arquivo selecionado para um cursor e os dados são transformados onde é feita a identificação de entidades para posterior atribuição de classes.

6.6.2 Script de migração

A migração é feita de forma independente para cada Arquivo, o script é posto a correr individualmente para cada ficheiro json de configuração. Inicialmente são criados todos os nós pais referentes aos Arquivos existentes, denominados com a classe *E22 HumanMadeObject*. Agora percorremos o cursor, linha a linha e criamos para cada linha o nó filho (parte física do documento) pertencente à classe *E22 HumanMadeObject*, caso este não tenha valor na coluna *ParentID* relaciona-se diretamente com o nó identificador do arquivo e é denominado de fundo, caso contrário conecta-se ao nó com o *ID* igual ao *ParenteID* deste. Esta relação será *P46 Is Composed Of*.

Os nós físicos identificadores do documento por sua vez relacionam-se com o objeto linguística desse mesmo documento que será um nó da classe *E33 Linguistic Object*, esta ligação é feita através da relação *P128 Carries*.

O nó físico do documento relaciona-se diretamente com todos os valores que identificam fisicamente o mesmo. Por outro lado, o nó linguístico do documento relaciona-se diretamente com todos os valores que identificam linguisticamente o mesmo.

Na figura 6.11 é possível visualizar alguns nós e relações na plataforma do Neo4J.

6.7 Criação das bases de dados do Elastic search

A utilização da base de dados de grafos possibilita o armazenamento de relacionamentos entre classes e a navegação por eles. No contexto deste projeto permite a pesquisa avançada por uma classe que permite a identificação de um determinado documento ou de vários documentos que contenham essa classe e por sua vez a relação dessa classe com outros elementos identificadores dos documentos que se pretendem pesquisar.

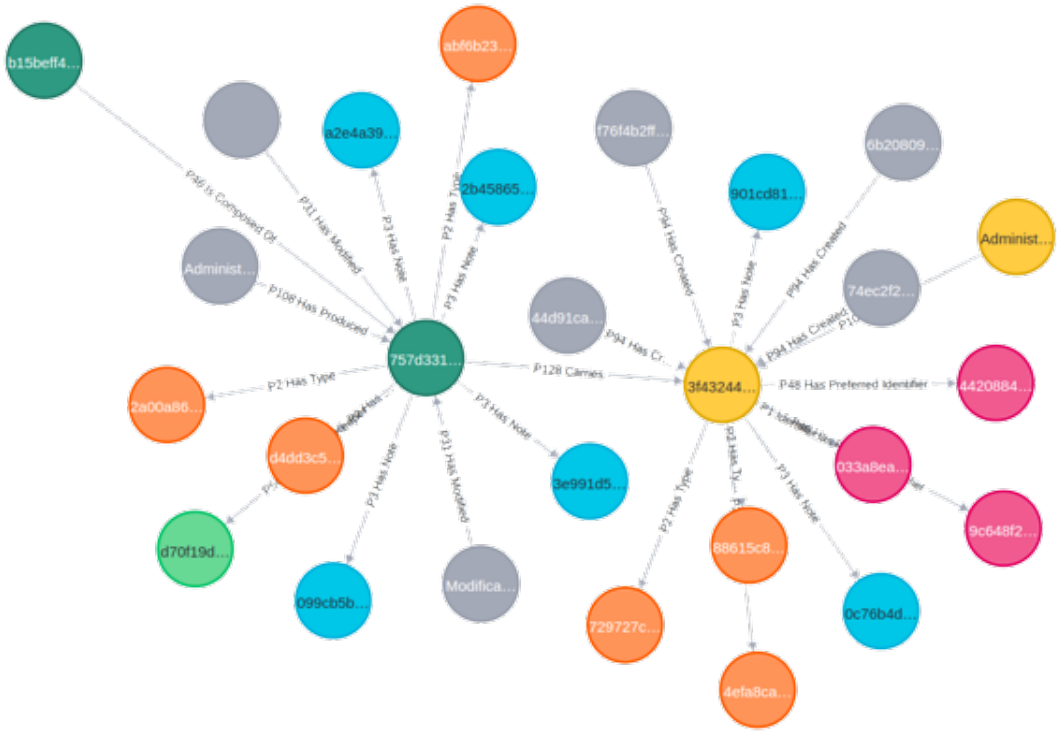

Figura 6.11: Exemplo da visualização da base de dados de grafos

Porém para uma pesquisa simples de documentos a base dados de grafos não é a mais prática, portanto para este efeito foi escolhida uma base de dados orientada a documentos, pois permite indexação de texto para pesquisas lexicais, e indexação de vetores para pesquisa densa. O Elasticseach foi a escolha para exercer como base de dados NoSQL, em conjunto utilizou-se o haystack uma estrutura python que permite que a pesquisa por documentos na base de dados possa ser feita através de linguagem natural, facilitando a implementação da mesma.

A melhor maneira de implementar a base de dados do Elasticsearch é recorrendo ao Docker que permite criar clusters do Elasticsearch, permite flexibilidade e permite ter as bases de dados do Elasticsearch a correrem no docker ao invés da necessidade de termos que correr numa máquina. A necessidade desta utilização originou a criação de um ficheiro docker-compose.yml.

6.7.1 Criação do ficheiro docker-compose.yml

O arquivo docker-compose.yml cria um cluster Elasticsearch seguro de três nós com autenticação e criptografia de rede habilitadas. O ficheiro vem com configurações de acesso predefinidas essa configuração expõe a porta 9200 em todas as interfaces da rede. Devido à forma como o Docker lida com as portas, uma porta que não está vinculada ao localhost deixa o cluster Elasticsearch acessível publicamente, possivelmente ignorando qualquer configuração de firewall. Para não possibilitar o acesso a hosts externos, definimos o valor de ES_PORT no arquivo .env para 127.0.0.1:9200. Assim, o Elasticsearch só estará acessível a partir da própria máquina host.

Para correr o ficheiro apenas temos que inserir no terminal ”docker-compose up”. Eventual-

mente na necessidade da utilização de uma máquina para guardar estes dados apenas temos que correr este comando para inserir a base de dados do Elasticsearch.

6.7.2 Inicialização da base de dados

Agora que já foi criado o cluster do Elasticsearch podemos proceder a inicialização da base de dados do Elasticsearch. Para esse efeito é necessário criar um index que identifica a base de dados que estamos a inicializar. Nesta criação podemos utilizar a biblioteca do haystack que permite a criação de document stores com um index definido. O haystack contém uma variedade de ferramentas que permitem manusear a base de dados de uma forma eficiente e facilitada, seja na inserção de documentos ou na obtenção de documentos.

Inicializamos quatro indexes um relativo aos documentos, que contém exatamente aquilo que a base de dados antiga do digitarq continham, outros dois relativos às entidades extraídas (lugares e atores) e outro relativo a eventos. O motivo pelo qual guardamos estes quatro indexes no elastic search deve-se à pesquisa que será realizada por parte da aplicação web.

6.7.3 Migração dos dados antigos do Digitarq para a base de dados do Elasticsearch

Para além de migrarmos dados para a base de dados do neo4j também iremos migrar para a base de dados do elasticsearch.

Para o index dos documentos vamos inserir diretamente as colunas dos metadados existentes nas bases de dados antigas do Digitarq.

Para o index dos atores vamos inserir as entidades extraídas pelo modelo do tipo pessoa (PER) e organização (ORG).

Para o index dos lugares vamos inserir as entidades extraídas pelo modelo do tipo localização (LOC).

O index dos eventos não vamos migrar nada pois não é uma entidade na qual houvesse interesse retirar informação.

6.8 Construção da API - FastAPI

O objetivo, após a migração das bases de dados antigas do Digitarq, é a criação de uma API que forneça a conexão entre o backend da aplicação web às duas bases de dados necessárias para o funcionamento da aplicação web (Bases de dados do Neo4j e base de dados do Elasticsearch) 6.12.

Para a construção desta API optamos pela *web framework* FastAPI, a escolha desta denotou-se pelo facto de ser uma framework com alta performance, com facilidade ao nível da implementação, robusta e intuitiva.

Figura 6.12: Pagina inicial do Digitarq atualizado

6.8.1 Criação de endpoints

A aplicação web necessitava de executar alguns pedidos. Alguns que necessitavam de acesso à base de dados do neo4j, outros a base de dados do elastic search e outros que necessitavam de ambas.

Ao todo criamos cinco pedidos para cada um dos quatro indexes inicializados no elasticsearch referidos em cima (documentos, atores, lugares e eventos).

- Pedido get de um documento/ator/lugar/evento - pedido que usa como recurso a base de dados do neo4j
- Pedido get de todos os documentos/atores/lugares/eventos - pedido que usa como recurso a base de dados do elastic search
- Pedido delete que elimina de ambas as bases de dados documento/ator/lugar/evento
- Pedido post e put que inserem/atualizam em ambas as bases de dados um documento/ator/lugar/evento.

Capítulo 7

Conclusão, Limitações e Trabalho Futuro

Em resumo, conseguimos encontrar soluções para os principais objetivos deste projeto, a fim de facilitar e melhorar o acesso à plataforma do Digitarq.

O primeiro objetivo alcançado foi a migração das antigas bases de dados relacionais desta plataforma, que continham informações sobre os metadados dos documentos, para uma base de dados de grafos. Isso foi realizado por meio do desenvolvimento de um modelo neuronal que permitiu o reconhecimento de entidades com relevância semântica, possibilitando assim a construção subsequente de uma base de dados de grafos seguindo o modelo conceptual do CIDOC-CRM. Além disso, conseguimos migrar esses metadados para uma base de dados Elasticsearch.

O segundo objetivo cumprido foi a criação de uma API que estabeleceu a conexão entre as bases de dados e a aplicação web do Digitarq, permitindo aos usuários pesquisar documentos de forma eficaz, este objetivo foi atingido através da implementação de uma API e da criação dos endpoints necessários para a realização de pedidos requisitados pela aplicação web.

Todavia surgiram limitações ao longo da implementação do projeto que poderiam ser aperfeiçoadas num trabalho futuro. A principal centra-se no desenvolvimento do modelo neuronal, onde não tivemos suporte de dados anotados do Digitarq o que provavelmente melhorava bastante a capacidade do modelo, por sua vez também poderíamos ter recolhido mais datasets na internet que com certeza melhoravam a capacidade do mesmo. Outra limitação que está relacionada com o desenvolvimento do modelo neuronal seria a capacidade de acesso a máquinas com grande capacidade de GPU que iriam diminuir os tempos de treino aumentando assim a rapidez com que poderíamos executar vários treinos e até mesmo treinos mais complexos.

Relativamente a trabalho futuro, acrescenta-se a capacidade de realizar a migração completa das bases de dados do DIGITARQ antigas, uma vez que é um processo demorado não conseguimos realizar a migração total das bases de dados. Sendo assim, resultados tais como: número total de documentos analisados, tamanho total da base de dados (nós e relações criadas), total de entidades extraídas e tamanho da base de dados do elasticsearch , são resultados que não foram obtidos e contabilizados.

Por fim outra limitação que tivemos e que poderia ser melhorada como trabalho futuro seria o

uso de uma API de localizações utilizada no processo de desambiguação paga que tivesse melhores recursos e resultados de pesquisa.

Bibliografia

[1] Dataset babelscape/wikineural. https://huggingface.co/datasets/Babelscape/wikineural. Acedido: 2023-07-12.

[2] Dataset harem. https://huggingface.co/datasets/harem. Acedido: 2023-07-12.

[3] Dataset jfrenz/legalglue. https://huggingface.co/datasets/jfrenz/legalglue. Acedido: 2023-07-12.

[4] Dataset paramopama. https://github.com/davidsbatista/NER-datasets/tree/master/Portuguese. Acedido: 2023-07-12.

[5] Dataset tner/multinerd. https://huggingface.co/datasets/tner/multinerd. Acedido: 2023-07-12.

[6] Direção-geral do livro, dos arquivos e das bibliotecas. https://dglab.gov.pt/. Acedido: 2023-02-15.

[7] Hugging face - transformers. https://huggingface.co/docs/transformers/index. Acedido: 2023-06-19.

[8] Hugging face – bert. https://huggingface.co/docs/transformers/model_doc/bert. Acedido: 2023-06-19.

[9] Hugging face – electra. https://huggingface.co/docs/transformers/model_doc/electra. Acedido: 2023-06-19.

[10] Hugging face – roberta. https://huggingface.co/docs/transformers/model_doc/roberta. Acedido: 2023-06-19.

[11] Hugging face – t5. https://huggingface.co/docs/transformers/model_doc/t5. Acedido: 2023-06-19.

[12] Weights biases: The ai developer platform - wandb. https://wandb.ai/site. Acedido: 2023-06-19.

[13] Francisca Adoma Acheampong, Henry Nunoo-Mensah, and Wenyu Chen. Transformer models for text-based emotion detection: a review of bert-based approaches. *Artificial Intelligence Review*, pages 1–41, 2021.

[14] Margarida Gouveia Augusto. Avaliação da migração de registos de arquivo para dados ligados no projeto episa. 2022.

[15] C Bekiari, G Bruseker, M Doerr, CE Ore, S Stead, and A Velios. Volume a: Definition of the cidoc conceptual reference model. version 7.1, 2021.

[16] Diedre Carmo, Marcos Piau, Israel Campiotti, Rodrigo Nogueira, and Roberto Lotufo. Ptt5: Pretraining and validating the t5 model on brazilian portuguese data. *arXiv preprint arXiv:2008.09144*, 2020.

[17] Tiago Carneiro, Raul Victor Medeiros Da Nóbrega, Thiago Nepomuceno, Gui-Bin Bian, Victor Hugo C De Albuquerque, and Pedro Pedrosa Reboucas Filho. Performance analysis of google colaboratory as a tool for accelerating deep learning applications. *IEEE Access*, 6:61677–61685, 2018.

[18] Lázaro Costa, Nuno Freitas, and João Rocha da Silva. An evaluation of graph databases and object-graph mappers in cidoc crm-compliant digital archives. *Journal on Computing and Cultural Heritage (JOCCH)*, 15(3):1–18, 2022.

[19] Ivo Fernandes, Henrique Lopes Cardoso, and Eugenio Oliveira. Applying deep neural networks to named entity recognition in portuguese texts. In *2018 Fifth International Conference on Social Networks Analysis, Management and Security (SNAMS)*, pages 284–289. IEEE, 2018.

[20] Luís Miguel Ferros, Miguel Ferreira, and José Carlos Ramalho. Digitarq e o novo módulo de interoperabilidade oai-pmh. In *Actas do Congresso Nacional de Bibliotecários, Arquivistas e Documentalistas*, number 10, 2010.

[21] Andrea Galassi, Marco Lippi, and Paolo Torroni. Attention in natural language processing. *IEEE transactions on neural networks and learning systems*, 32(10):4291–4308, 2020.

[22] Neha Gupta et al. Artificial neural network. *Network and Complex Systems*, 3(1):24–28, 2013.

[23] Wenxin Jiang, Nicholas Synovic, Matt Hyatt, Taylor R Schorlemmer, Rohan Sethi, Yung-Hsiang Lu, George K Thiruvathukal, and James C Davis. An empirical study of pre-trained model reuse in the hugging face deep learning model registry. *arXiv preprint arXiv:2303.02552*, 2023.

[24] C Mendonça Júnior, Hendrik Macedo, Thiago Bispo, Flávio Santos, Nayara Silva, and Luciano Barbosa. Paramopama: a brazilian-portuguese corpus for named entity recognition. *Encontro Nac. de Int. Artificial e Computacional*, 2015.

[25] Morteza Karimzadeh, Wenyi Huang, Siddhartha Banerjee, Jan Oliver Wallgrün, Frank Hardisty, Scott Pezanowski, Prasenjit Mitra, and Alan M MacEachren. Geotxt: a web api to leverage place references in text. In *Proceedings of the 7th workshop on geographic information retrieval*, pages 72–73, 2013.

[26] Douglas Kunda and Hazael Phiri. A comparative study of nosql and relational database. *Zambia ICT Journal*, 1(1):1–4, 2017.

[27] Hao Liu, Qinjun Qiu, Liang Wu, Wenjia Li, Bin Wang, and Yuan Zhou. Few-shot learning for name entity recognition in geological text based on geobert. *Earth Science Informatics*, 15(2):979–991, 2022.

[28] Kiran Maharana, Surajit Mondal, and Bhushankumar Nemade. A review: Data preprocessing and data augmentation techniques. *Global Transitions Proceedings*, 3(1):91–99, 2022.

[29] Batta Mahesh. Machine learning algorithms-a review. *International Journal of Science and Research (IJSR).[Internet]*, 9(1):381–386, 2020.

[30] Sérgio Nunes, Tiago Silva, Cláudia Martins, and Rita Peixoto. Episa platform: A technical infrastructure to support linked data in archival management. In *Workshops and Doctoral Consortium of the 26th International Conference on Theory and Practice of Digital Libraries*, pages 86–97, 2022.

[31] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551, 2020.

[32] Iqbal H Sarker. Machine learning: Algorithms, real-world applications and research directions. *SN computer science*, 2(3):160, 2021.

[33] Hemlata Shelar, Gagandeep Kaur, Neha Heda, and Poorva Agrawal. Named entity recognition approaches and their comparison for custom ner model. *Science & Technology Libraries*, 39(3):324–337, 2020.

[34] Meshal Shutaywi and Nezamoddin N Kachouie. Silhouette analysis for performance evaluation in machine learning with applications to clustering. *Entropy*, 23(6):759, 2021.

[35] Fábio Souza, Rodrigo Nogueira, and Roberto Lotufo. Portuguese named entity recognition using bert-crf. *arXiv preprint arXiv:1909.10649*, 2019.

[36] Zhensu Sun, Li Li, Yan Liu, Xiaoning Du, and Li Li. On the importance of building high-quality training datasets for neural code search. In *Proceedings of the 44th International Conference on Software Engineering*, pages 1609–1620, 2022.

[37] Yelda Unal and Halit Oguztuzun. Migration of data from relational database to graph database. In *Proceedings of the 8th International Conference on Information Systems and Technologies*, pages 1–5, 2018.

[38] John E Vargas-Munoz, Shivangi Srivastava, Devis Tuia, and Alexandre X Falcao. Openstreetmap: Challenges and opportunities in machine learning and remote sensing. *IEEE Geoscience and Remote Sensing Magazine*, 9(1):184–199, 2020.

[39] R Vidhya and G Vadivu. Research document search using elastic search. *Indian Journal of Science and Technology*, 9(37), 2016.

[40] Jiexin Wang, Adam Jatowt, Michael Färber, and Masatoshi Yoshikawa. Improving question answering for event-focused questions in temporal collections of news articles. *Information Retrieval Journal*, 24:29–54, 2021.

[41] Jian Yang, Shaohan Huang, Shuming Ma, Yuwei Yin, Li Dong, Dongdong Zhang, Hongcheng Guo, Zhoujun Li, and Furu Wei. Crop: Zero-shot cross-lingual named entity recognition with multilingual labeled sequence translation. *arXiv preprint arXiv:2210.07022*, 2022.