INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA DE INTELIGÊNCIA ARTIFICIAL




Respostas automáticas na pesquisa de conteúdos educativos

ABEL DE JESUS FERREIRA
outubro de 2022

P.PORTO

Respostas automáticas na pesquisa de conteúdos educativos

Abel Jesus Ferreira

Aluno nº: 1960533

Dissertação para obtenção do Grau de
Mestre em Engenharia de Inteligência Artificial

Orientador: Doutor Luís Felipe Rocha de Faria

Júri:

Presidente:

Doutora Maria Goreti Carvalho Marreiros, Professora Coordenadora com Agregação do Instituto Superior de Engenharia do Instituto Politécnico do Porto

Vogais:

Doutor Carlos Fernando da Silva Ramos, Professor Coordenador Principal do Instituto Superior de Engenharia do Instituto Politécnico do Porto

Doutor Luiz Felipe Rocha de Faria, Professor Coordenador do Instituto Superior de Engenharia do Instituto Politécnico do Porto

Porto, 25 de outubro 2022

<< Vive como se fosses morrer amanhã,
aprende como se fosses viver para sempre.>>

Mahatma Gandhi

Resumo

Perante o constante crescimento do volume de informação disponível na internet, os motores de pesquisa desempenham um papel fundamental na forma como facilmente podemos encontrar a informação que procuramos.

A utilização da Inteligência Artificial no processamento de língua natural aplicada aos motores de pesquisa de conteúdos, tem permitido que estes correspondam de forma mais inteligente, não só na vertente mais habitual para encontrar os documentos mais relevantes, mas também no entendimento das intenções do utilizador, apresentando respostas diretas às questões colocadas.

Nos sistemas de *eLearning*, o crescimento da informação não foge à regra, acompanhando a tendência de transição digital, quer pela criação de novos conteúdos nos mais variados formatos, quer pela transformação digital dos manuais escolares.

Este projeto, EVGuru, pretende explorar as técnicas mais adequadas de Inteligência Artificial aplicada ao processamento de língua natural, que permitam a criação de uma experiência de pesquisa inteligente de conteúdos educativos. O projeto é desenvolvido considerando uma eventual integração na plataforma de *eLearning* Escola Virtual da Porto Editora.

Para tal, esta investigação é acompanhada do desenho da arquitetura dos componentes necessários à implementação de um protótipo, capaz de encontrar uma resposta direta para a questão colocada pelo utilizador, através da pesquisa nos textos de um manual escolar.

Durante a realização do projeto, vários modelos foram testados em diferentes tarefas do processamento de língua natural, como classificação de textos, questões e respostas e geração de questões.

Palavras-chave: *eLearning*, motor de pesquisa, *machine learning*, Inteligência artificial, questões e respostas, *reading comprehension*

Abstract

Due to the constantly growing of the information available on the internet, search engines play a key role in how easily we can find the information we are searching for.

The use of Artificial Intelligence in natural language processing applied to content search engines, has allowed them to match more intelligently, not only in the most usual way to find the most relevant documents, but also in understanding the user's intentions, providing direct answers to the questions asked.

In eLearning platforms, the growth of information is not exception, following the trend of digital transition, either through the creation of new content in the most varied formats, or through the digital transformation of school textbooks.

This project, EVGuru, aims to explore the most appropriate techniques of Artificial Intelligence applied to natural language processing, which allow the creation of an intelligent search experience for educational content. The project is developed considering an eventual integration into the Porto Editora Escola Virtual eLearning platform.

Therefore, this investigation is accompanied by the components architecture design that will be necessary for the implementation of a prototype, capable of finding a direct answer to the question posed by the user, through the search in the texts of a school textbook.

During the project, several models were tested in different tasks of natural language processing, such as text classification, questions and answers and question generation.

Keywords: eLearning, search engines, machine learning, artificial intelligence, question-answering, reading comprehension

Agradecimentos

Este projeto não seria possível concretizar sem a orientação, apoio, motivação e paciência de um conjunto de pessoas que durante esta caminhada, ajudaram a criar as condições necessárias para a sua conclusão.

Em primeiro lugar, quero agradecer ao meu orientador professor Luiz Faria pela disponibilidade que sempre demonstrou e orientação para ajudar a manter o foco no essencial. Agradeço também ao ISEP e a todos os professores que permitiram a criação do MEIA, com um agradecimento especial ao professor Carlos Ramos pela motivação criada aos alunos durante mestrado.

Agradeço à Porto Editora, por permitir a realização deste projeto com a disponibilização de conteúdos digitais importantes para o projeto, com um agradecimento especial ao meu amigo Luís Abreu pela motivação, compreensão e por sempre me ter dado as condições necessárias para a realização do projeto.

Muito obrigado aos meus pais porque tudo o que sou é reflexo da educação e perspetiva sobre a vida que sempre me passaram, nomeadamente na vontade de estar sempre a aprender.

Um agradecimento muito especial à minha companheira pelo que faz acontecer. Pela energia, pela paciência, pela motivação, por fazer com que tudo faça sentido. Agradeço igualmente aos meus filhos pelo apoio e compreensão e que este trabalho seja igualmente uma motivação para eles.

Por último, agradeço também a todos os meus amigos porque sempre estão comigo em todos os momentos.

Conteúdo

Lista de Figuras

Lista de Tabelas

Lista de Acrónimos

EV	Escola Virtual
LMS	*Learning Management System*
IA	Inteligência Artificial
JSON	*Javascript Object Notation*
API	*Application programming Interface*
TF-IDF	*Term Frequency-Inverse Document Frequency*
MOOC	*Massive Open Online Courses*
NLP	*Natural Language Processing*
IR	*Information Retrieva*l
MRC	*Machine Reading Comprehension*
DPR	*Dense Passage Retrievers*
QA	*Question Answering*
IE	*Information Extraction*
ID	*Intent Determination*
SF	*Slot Filling*
DARPA	*Defense Advanced Research Program Agency*
ATIS	*Airline Travel information System*
ER	*Error Rate*
NN	*Nearest Neighbors*
NB	*Naïve Bayes*
DT	*Decision Tree*
SVM	*Support Vector Machines*
CNN	*Convolutional Neural Network*
RNN	*Recurrent Neural Networks*
GRU	*Gated Recurrent Unit*
LSTM	*Long-Short Term Memory*
Bi-LSTM	*BiDirectional Long Short Term Memory*
BERT	*Bidirectional Encoder Representations from Transformers*
CRF	*Conditional Random fields*
BrWaC	*Brazilian Web as Corpus*
MC	*Machine comprehension*
FC	*Fully Connected*
SQuAD	*Stanford Question Answering Dataset*
EM	*Exact Match*
FFN	*Feed Forward Network*
BiDAF	*Bi-Directional Attention Flow*
MLM	*Masked Language Model*
T5	*Text-to-Text Transfer Transformer*
OCR	*Optical Character Recognition*

NER	*Named Entity Recognition*
CUDA	*Computed Unifed Device Architecture*
GPU	*Graphics Processing Unit*

1.Introdução

A plataforma Escola Virtual (EV), é o serviço comercial de ensino online (*eLearning*), disponibilizado pela empresa Porto Editora para os professores e alunos do 1º ciclo de ensino ao ensino secundário.

Um dos objetivos destes sistemas de *eLearning* é disponibilizar uma grande variedade de conteúdos digitais educativos nos mais variados formatos, que no caso da plataforma EV inclui os próprios manuais escolares. O volume e variedade destes conteúdos representa um desafio, no sentido de permitir que o aluno possa encontrar de forma rápida e simplificada os recursos e as respostas que procura.

Este capítulo apresenta uma contextualização do projeto através da descrição do âmbito e enquadramento na plataforma EV, caracterização do problema, objetivos e resultados esperados, seguido da descrição da estrutura do documento.

1.1 Contexto

A digitalização do ensino tem sido uma tendência nos últimos anos e irá ser cada vez mais intensificada nos próximos anos (Insights 2020).

Neste processo de digitalização do ensino, as plataformas de *eLearning* desempenham um papel fundamental já que permitem aos professores e alunos, aceder a um conjunto alargado de conteúdos educativos digitais (aulas virtuais, vídeos, áudios, atividades digitais, entre outros), partilhar esses conteúdos, atribuir tarefas, entre outros cenários de colaboração. Estas funcionalidades são responsabilidade da componente central das plataformas de *eLearning,* denominado de *Learning Management System* (LMS).

É no sistema LMS que estão disponibilizados os cursos e respetivas aulas digitais que o aluno pode frequentar, bem como todos os conteúdos digitais complementares à aprendizagem de um determinado conteúdo, sejam eles conteúdos expositivos ou avaliativos.

A plataforma EV posiciona-se atualmente como uma plataforma de *eLearning* complementar às atividades de ensino em sala de aula e por essa razão, os cursos são configurados tendo por base o curriculum e objetivos de aprendizagem definidos pelo Ministério da Educação para uma dada disciplina do ensino do 1º ao 12º ano de escolaridade. Nesta abordagem, os conteúdos de uma dada disciplina/curso são apresentados ao utilizador num percurso de aprendizagem predefinido pelos professores e gestores de conteúdos que definiram a estrutura do curso.

Os conteúdos digitais na sua forma mais granular (exemplo: um vídeo ou uma atividade) estão também disponíveis para o aluno através de uma pesquisa tradicional por palavras chave de pesquisa e filtros como a disciplina ou ano escolar, permitindo ao aluno procurar conteúdos que abordam um dado tema que pretende apreender ou aprofundar conhecimento.

Nas plataformas de *eLearning* atuais, tipicamente existe uma separação entre os conteúdos digitais que constituem os cursos e os livros ou manuais (*eBooks*) que o utilizador tem na sua biblioteca digital. Esta separação acontece também na funcionalidade de pesquisa, ou seja, uma pesquisa no texto do *eBook* habitualmente acontece num contexto de leitura digital do manual para facilitar o posicionamento na leitura, enquanto que a pesquisa de conteúdos digitais acontece nos cursos que estão disponíveis no *LMS.*

Vivemos numa sociedade digital onde a informação e o conhecimento está hoje mais disponível e acessível do que nunca. Através dos nossos computadores e dispositivos móveis, conseguimos à distância de um simples *click* realizar uma pesquisa e obter no imediato resposta para algo que desconhecemos.

Esta necessidade de obter repostas rápidas é uma característica dos jovens da atualidade, habituados a receber diariamente constantes *headlines* de informação, nomeadamente pela utilização das redes sociais ou agregadores de notícias.

É a pensar nestas novas características dos jovens na pesquisa de conhecimento que este trabalho se desenvolve, pensando potenciar a pesquisa atual através da utilização de Inteligência Artificial (IA) e disponibilizar uma pesquisa inteligente de conteúdos digitais educativos, capaz de entender a intenção do aluno quando efetua uma pesquisa, apresentando, sempre que possível, uma resposta direta e mais direcionada, utilizando para isso tanto os conteúdos digitais dos cursos do LMS, como o texto dos manuais escolares.

Esta abordagem de pesquisa inteligente é algo que podemos hoje encontrar nos motores de pesquisa de páginas web mais utilizados como é o exemplo do *Google*, os quais, beneficiando do enorme volume de informação, conseguem responder diretamente e de forma bastante eficiente às intenções do utilizador, seja a apresentar a definição para um conceito ou a apresentar um mapa de restaurantes mais perto do utilizador.

1.2 Problema

A plataforma EV contém atualmente mais de 500.000 conteúdos digitais, disponíveis para professores e alunos em cerca de 90 cursos focados nas disciplinas lecionadas do 1º ao 12º do ensino secundário.

A funcionalidade de pesquisa de conteúdos educativos atualmente existentes na EV, permite aos alunos e professores utilizar palavras chaves e filtros de pesquisa sobre um banco de recursos digitais previamente catalogados com metainformação e indexados em documentos *JavaScript Object Notation* (JSON) na plataforma *Solr* (Seeley 2006).

A plataforma *Apache Solr*, é uma plataforma implementada sobre *Apache Lucene* (D. Cutting 1999), delegando nesta biblioteca a responsabilidade de indexação e pesquisa, assim como implementando funcionalidades adicionais sendo uma das mais relevantes a standard *Application Programming Interface* (API) para *XML, JSON e HTTP*, entre outras como *Facet Search*, replicação dos índices ou *Caching*.

Apache Lucene é fundamentalmente um motor de indexação e pesquisa, utilizando para a indexação uma abordagem de índices invertidos (Cutting and Pedersen 1989) a par dos modelos *Term Frequency-Inverse Document Frequency* (TF-IDF) (Robertson 2004) e BM25 também descrito por (Robertson and Zaragoza 2009), para o calculo de similaridade de cossenos para determinar o *score* dos documentos de pesquisa relativamente à expressão de pesquisa utilizada.

Com a utilização do *Solr* na EV como plataforma para indexação e pesquisa, o utilizador tem a possibilidade de pesquisar um ou vários termos compondo a expressão de pesquisa e, se necessário, utilizar alguns dos *facets* (Hostetter 2006) disponíveis para filtrar mais os resultados.

Estes modelos de indexação são extremamente eficazes para encontrar documentos que apresentam similaridade entre os termos utilizados na pesquisa e os termos existentes nos documentos, no entanto não tentam inferir a intenção do utilizador para, de forma ainda mais eficaz, apresentar diretamente a resposta que melhor satisfaz a intenção da consulta efetuada.

Por outro lado, a pesquisa baseada nestes métodos, não responde a questões de similaridade como por exemplo uma pesquisa utilizando o termo “carro” com textos que incluem “automóvel”, não sendo considerada a semântica dos termos.

Na Figura 1 é apresentado um exemplo de uma pesquisa realizada na EV utilizando a expressão “constituintes moleculares” sendo devolvidos 129 resultados. Embora neste exemplo provavelmente o aluno irá encontrar a informação que procura no primeiro recurso digital da lista (recurso com o *score* mais elevado), a pesquisa baseada nas palavras chave devolve também em terceiro lugar um recurso da área de português, quando o mais provável é estar à procura de conteúdos na área de biologia.

Figura 1 - Exemplo de pesquisa na Escola Virtual

No cenário em que o utilizador não encontra no recurso digital a informação pretendida, terá de consultar o conteúdo um a um entre as possíveis dezenas de recursos obtidos no resultado de pesquisa.

Para além desta dificuldade, a metainformação associada aos recursos é algo limitada, já que implica uma catalogação manual dos milhares de recursos, sem tirar partido da informação presente no conteúdo do manual escolar digital. A pesquisa atual, não pesquisa no conteúdo de texto dos manuais escolares.

No cenário em que o aluno procura uma resposta para uma definição, pretende encontrar a data de um evento, ou conhecer uma determinada fórmula, esta informação pode ser encontrada dentro do texto de um manual escolar, no entanto o aluno não a irá encontrar de forma direta na pesquisa atual.

1.3 Objetivos

O aumento constante da informação digitalizada e acessível é uma enorme vantagem da sociedade do conhecimento atual, no entanto esta massificação da informação, traz também grandes desafios quanto à fiabilidade da informação que é consumida, sendo cada vez maior a preocupação da sociedade relativamente à qualidade dos conteúdos digitais que consumimos na internet para obtenção de conhecimento.

Nos últimos anos são também cada vez mais as plataformas de *eLearning* comerciais e não comerciais, como é o caso das *Massive Open Online Courses* (MOOC), com oferta de conteúdos educativos digitais.

Tirando partido da qualidade dos conteúdos educativos e manuais digitais escolares existentes na plataforma EV, este projeto é apresentado em duas vertentes:

1. Um estudo sobre a utilização de diversas técnicas de *Natural Language Processing* (NLP) e forma de as combinar, de modo a conseguir compreender as intenções do utilizador no cenário de pesquisa de conteúdos e, tendo por base o texto do manual escolar combinado com os recursos digitais existentes, extrair a informação necessária para apresentar sempre que possível uma resposta ao aluno que satisfaça a sua intenção de pesquisa.
2. O desenho da arquitetura e desenvolvimento do protótipo de um sistema complementar e integrado na pesquisa da EV, que permita avaliar as técnicas utilizadas neste contexto. Para facilitar a leitura deste documento, este protótipo é designado no 3 de EVGuru.

De modo a proporcionar ao aluno uma experiência de pesquisa menos tradicional e mais aproximada com um assistente virtual, este projeto contempla também a disponibilização de uma componente de *user interface* desenhada para esse efeito.

Dado o elevado número de recursos digitais existentes, bem como a diversidade de manuais escolares, optou-se por definir como âmbito deste projeto especificamente a disciplina de biologia e geologia de 10º ano. É expectável que o modelo desenvolvido seja facilmente generalizado para utilização no âmbito de outras disciplinas e anos escolares com características semelhantes. A opção por esta disciplina teve por base, uma orientação editorial interna da empresa Porto Editora, aliada ao facto de ser uma disciplina de natureza descritiva e rica em definições.

No estudo e experimentação das técnicas de NLP, aquelas que assumem maior relevância neste projeto e detalhadas nos próximos capítulos são: modelos de classificação de intenções, modelos de similaridade de textos para *Information Retrieval* (IR) e *Machine Reading Comprehension* (MRC) que será utilizado para responder a questões utilizando segmentos de texto do manual.

Uma fase que antecede o estudo e experimentação das técnicas de NLP é a análise dos dados existentes tendo em vista a criação dos *datasets* de treino necessários para os diferentes modelos, nomeadamente:

- Análise de *datasets* públicos em português que possam ser utilizados para as diferentes tarefas.
- Análise do conteúdo do manual digital e extração da informação do PDF, para formato semiestruturado para indexação.
- Análise do banco de questões/respostas existente na plataforma EV para utilização como *dataset* de treino MRC.
- Criação de *dataset* com segmento de textos do manual e perguntas sobre a informação contida nesses segmentos, com as respetivas para o treino da tarefa de MRC.

Assim, numa primeira fase, tendo em consideração a complexidade em preparar estes *datasets*, serão utilizados alguns *dataset*s públicos como é o caso do SQuAD (Rajpurkar, et al. 2016) para o MRC.

Uma vez avaliados e selecionados os modelos a utilizar nos *datasets* públicos, estes modelos serão treinados com os *datasets* preparados com os conteúdos da disciplina de biologia e geologia 10º e reavaliada a eficácia e relevância dos resultados obtidos.

Concluída a fase de avaliação e implementação das componentes de IA a utilizar, deverá ser efetuada a integração com o sistema de pesquisa da EV.

Posto isto, os principais objetivos do projeto resumem-se nos seguintes pontos:

1. Estudo, avaliação e seleção das técnicas de NLP para análise de intenções nas expressões de pesquisa.
2. Estudo, avaliação e seleção das técnicas de NLP para modelos de questões/resposta, nomeadamente em MRC.
3. Extração de texto semiestruturado do PDF do manual escolar e indexação para construção do IR.
4. Criação dos *datasets* de biologia e geologia 10º para MRC.
5. Desenvolvimento de novo *User Interface* para a pesquisa inteligente.
6. Realização de casos de uso reais que permitam avaliar a relevância dos resultados.

1.4 Estrutura do documento

Este documento inicia no capítulo 1 com uma introdução contextualizando o problema relativamente à pesquisa de conteúdos com os motores de pesquisa tradicionais, apresentando as motivações para a execução do projeto e objetivos de concretização.

No capítulo 2, é efetuada análise do estado da arte relativamente a pesquisas inteligentes e técnicas de IA utilizadas pelos mesmos, nomeadamente na obtenção de respostas diretas para as pesquisas, com atenção especial á tarefa de MRC. No final do capítulo são apresentadas conclusões na identificação das técnicas que são mais utilizadas.

O capítulo 3, descreve a arquitetura proposta para o protótipo do sistema a desenvolver e os seus componentes, detalhando cada componente relativamente às técnicas de IA utilizadas para as diferentes tarefas de *NLP,* incluindo *text classification, information retrieval, question-answering* e *question generation.*

No capítulo 4, são apresentadas as experiências e resultados realizados na avaliação dos modelos com melhores resultados para a execução das tarefas propostas, concluindo o capítulo com o resumo das conclusões sobre os resultados obtidos.

O documento finaliza com o capítulo 5 onde são apresentadas conclusões gerais quanto à concretização dos objetivos propostos, assim como são apresentadas as áreas previstas para futura investigação na tarefa de *question-answering.*

2. Estado da arte

Neste capítulo é apresentada uma análise relativa aos motores de pesquisa de conteúdos e abordagens técnicas utilizadas pelos mesmos, começando por apresentar as abordagens mais tradicionais, nas quais se enquadra o atual motor de pesquisa da Escola Virtual, até às abordagens que apresentam comportamentos inteligentes recorrendo às técnicas mais atuais de Inteligência Artificial (IA), nomeadamente na área do *Natural Language Processing* (NLP) e *Deep Learning*.

Para a realização desta revisão dos estudos existentes, as pesquisas foram realizadas utilizando *Google*, *Google Scholar, IEEE Xplore* e *B-On,* tentando responder às seguintes perguntas:

Q1: Estão os sistemas de *eLearning* a utilizar pesquisas inteligentes de conteúdos?

Q2: Quais as técnicas de IA utilizadas nas pesquisas inteligentes?

Q3: Quais as técnicas de IA utilizadas para dar respostas automáticas a partir do texto de um livro?

2.1 Soluções de eLearning com pesquisa inteligente

São várias as áreas de aplicação de IA nas soluções de eLearning (Neelakandan 2019), nomeadamente, respostas a questões em tempo real, utilização de NLP para a interação com os conteúdos digitais ou percursos de aprendizagem personalizados, sendo este último um dos que tem recebido maior atenção de investigação e desenvolvimento (Aldahwan and Alsaeed 2020).

No estudo realizado, não é habitual encontrar plataformas de LMS com abordagens de pesquisa inteligente, nomeadamente com a utilização de NLP.

A grande maioria das soluções, como é o caso do LMS *open source Moodle* com o *Moodle Global Search,* solução que utiliza a plataforma de pesquisa Solr, utilizam soluções tradicionais de pesquisa baseadas em indexação de palavras chave e *ranking* de documentos. No entanto, foi possível verificar que existe uma clara tendência na utilização de técnicas de IA por algumas das plataformas classificadas pelo site *elearningindustry.com* no top 20 das plataformas LMS preferidas pelos clientes.

A plataforma *Docebo* por exemplo, disponibiliza uma pesquisa que já incorpora NLP, permitindo entender questões colocadas em língua natural e pesquisar no conjunto de cursos e recursos da plataforma os resultados mais relevantes para o aluno, permitindo responder a questões como *"Where can I find information on writing non-discriminatory job adverts?*", onde o sistema é capaz de identificar as palavras chave que devem ser utilizadas para a realização da pesquisa (Powell n.d.).

Algumas soluções de LMS como o *Absorblms*, coloca a IA no centro da maior parte das funcionalidades que apresenta, com assistentes inteligentes para atividades de apoio à gestão da plataforma, recomendações automáticas de cursos para os alunos, assim como uma pesquisa com *ranking* de resultados baseada na experiência de utilização dos vários utilizadores da plataforma (Figura 2).

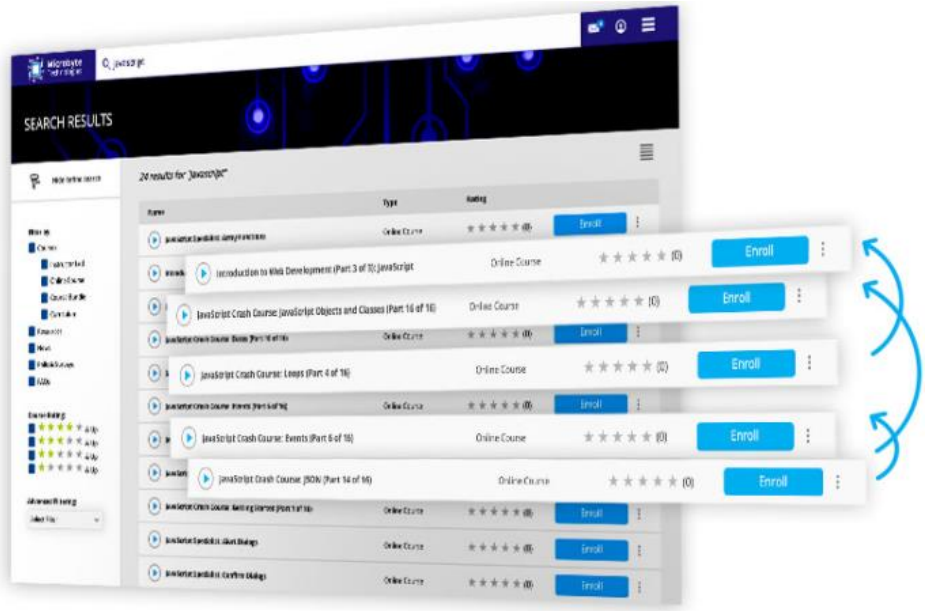

Figura 2 - Pesquisa no *Absorblms*

2.2 Motores de pesquisa convencionais

Os motores de pesquisa desempenharam um papel fundamental na evolução e utilização da internet como fonte de informação (Varian 2006). Imaginar a internet sem motores de pesquisa, seria como imaginar um armazém com milhões de documentos ricos em informação e conhecimento, mas de cada vez que necessitássemos de um desses documentos levaríamos tanto tempo no processo de procura que, ou desistíamos ou quando finalmente fosse encontrado, essa informação já seria obsoleta ou irrelevante.

O rápido crescimento do número de websites na internet criou a necessidade e oportunidade para a criação do primeiro motor de pesquisa na internet, *Archie,* disponibilizado em 1990 (Vaughan 2017). Nos anos que se seguiram outros surgiram como *Yahoo*, *Infoseek*, *Lycos* ou *Altavista*, no entanto, a grande mudança acontece com o lançamento do *Google* disponibilizado ao público em 1998 (Brin and Page 1998), o qual passa a dar uso intensivo à estrutura e informação presente no *hypertext* das páginas web*,* quer para a indexação quer para a criação de um *pagerank* permitindo obter resultados de pesquisa mais relevantes.

De acordo com o website *worldwidewebsize.com*, em 2020 existiam aproximadamente cerca de 6 mil milhões de páginas indexadas.

Tipicamente uma pesquisa consiste em alguém escrever um ou mais termos que constituem a expressão de consulta num motor de pesquisa e é a este que cabe a responsabilidade de devolver a resposta sobre a forma de uma lista de resultados (exemplo: documentos, produtos, vídeos) ordenados de acordo com um *score* calculado.

Embora nos motores de pesquisa atuais, as tarefas de pesquisa tenham evoluído para tarefas mais complexas, como por exemplo as que podemos assistir com facilidade na utilização dos motores de pesquisa web como o *Bing* ou *Google*, com tarefas de pergunta e resposta, os motores de pesquisa mais utilizados de âmbito fechado num dado website, como é o caso das pesquisas num motor de pesquisa de um site de *eCommerce* ou de *eLearning*, a tarefa principal ainda consiste na indexação utilizando índices invertidos e em devolver uma lista ordenada de resultados baseada nas palavras chave, filtros de pesquisa e no *score* calculado nomeadamente utilizando TF-IDF e similaridade do cosseno.

De modo muito geral, os índices invertidos funcionam criando uma tabela de relação (índice) entre cada palavra dos documentos e a posição onde estas se encontram em cada documento.

Para o cálculo do *score* os motores de pesquisa convencionais, como por exemplo os que são baseados em *Lucene*, com algumas pequenas variantes, realizam o cálculo do score aplicando as seguintes formulas principais.

O cálculo do *Term Frequency*, é realizado utilizando a fórmula:

$$tf(t,d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}} \quad (1)$$

Em que $f_{t,d}$ corresponde ao número de vezes que o termo surge no documento a dividir pelo número total de termos.

O *Inverse Document Frequency* (IDF), é calculado utilizando a fórmula:

$$idf(t,D) = \log \frac{N}{|\{d \in D : t \in d\}|} \quad (2)$$

Em que N é o número total de documentos (exemplo, frases) no corpus de texto total D, e $|\{d \in D : t \in d\}|$ é o número total de documentos em D onde surge o termo t.

O TF-IDF consegue então ser calculado com:

$$tfidf(t,d,D) = tf(t,d) \cdot idf(t,D) \quad (3)$$

Este valor é utilizado como medida de peso para permitir a criação dos vetores esparsos[1] de representação de cada termo no espaço.

Num cenário de pesquisa, as palavras ou sequência de palavras da pesquisa terão uma representação vetorial assim como as sequências de palavras dos documentos. Uma vez estando definidos estes vetores, a similaridade é calculada, por exemplo, utilizando a fórmula de cálculo do cosseno entre dois vetores:

$$\cos(t) = \frac{v \cdot w}{|v||w|} \quad (4)$$

A similaridade é então utilizada para a atribuição do *score* ao resultado de pesquisa de forma a que os resultados com melhor *score* sejam apresentados no topo.

2.3 Motores de pesquisa inteligentes

Ao contrário dos motores de pesquisa convencionais com métodos muito focados nas palavras chave utilizadas pelo utilizador para execução de uma consulta numa base de dados ou sistema de indexação, um sistema de pesquisa inteligente tem a capacidade de entender a intenção do utilizador e responder às questões colocadas.

Os assistentes virtuais que dispomos hoje nos nossos dispositivos como *Siri*[2], *Alexa*[3] ou Google *assistente*, incluem a pesquisa inteligente como uma das suas tarefas principais para rapidamente responder às questões colocadas pelos utilizadores.

Nos anos mais recentes, ao efetuarmos uma pesquisa nos motores de pesquisa web mais utilizados, é normal que a primeira informação apresentada seja uma resposta direta à intenção mais provável do utilizador ou resposta a uma questão colocada (Guha, McCool and Miller 2003).

Por outro lado, os modelos de similaridade baseados em vetores esparsos criados com TF-IDF ou BM25, com os avanços dos modelos de *deep learning* a par da cada vez maior capacidade de computação, têm tendência a ser substituídos por métodos de pesquisa que tiram partido destes modelos, nomeadamente na utilização de representações vetoriais densas, como são os *Dense Passage Retrievers* (DPR) apresentados no trabalho de (Karpukhin, et al. 2020).

No exemplo da Figura 3, para uma pesquisa efetuada no Google de "constituintes moleculares", o sistema considera que o utilizador pretende saber para que servem os "constituintes moleculares" e apresenta no imediato uma resposta para essa intenção e só posteriormente apresenta a lista de sites relacionados ordenados pela relevância dos mesmos.

[1] https://www.ibm.com/docs/en/essl/6.1?topic=vectors-sparse-vector
[2] Assistente virtual da Apple Inc.
[3] Assistente virtual da Amazon

Figura 3 - Pesquisa inteligente do Google

Para além dos motores de pesquisa web, também os motores de pesquisa utilizados em domínios de informação mais restritos, tiram partido da IA e em particular de NLP para tornar o comportamento de pesquisa inteligente. Alguns desses exemplos são as empresas *LucidWorks*, *Coveo* ou a *IBM* com o *IBM Watson Discovery*.

Utilizando a definição de sistemas de pesquisa inteligentes da IBM em (IBM n.d.), habitualmente estes são constituídos pelas seguintes funcionalidades:

- **Entendimento da língua natural** – Capacidade de ler e entender uma frase colocada pelo utilizador, para a partir desta construir uma consulta a uma fonte de dados.
- **Aprender a estrutura de documentos** – Através da utilização de *machine learning*, aprender a identificar e a classificar áreas de documentos ou documentos na integra como faturas, contractos, ordens de compra, entre outros.
- **Filtragem de resultados** – Pesquisa por facetas dos dados, restringindo os mesmo na pesquisa de informação específica.
- **Classificação e categorização de conteúdo** – Com a extração de entidades a partir de textos, conseguir identificar automaticamente nomes de pessoas, produtos ou organizações.

Com os grandes avanços da utilização de IA no processamento de língua natural, os agentes conversacionais (designados de *chat bots*) são cada vez mais utilizados em diferentes domínios de aplicação. Os motores de pesquisa inteligentes beneficiam também desta evolução e das técnicas utilizadas nestes agentes.

Uma das funcionalidades mais apreciadas pelos utilizadores de assistentes inteligentes aplicados a áreas de domínio fechado, é sem dúvida a capacidade de obter uma resposta imediata para uma determinada pergunta no contexto desse domínio, evitando assim que o utilizador se perca numa lista de resultados de pesquisa.

O desenvolvimento de sistemas e modelos automáticos de perguntas e respostas é um ramo de investigação, habitualmente denominado de *Question Answering* (QA)*,* que combina investigação de várias áreas como *Information Retrieval* (IR)*, Information Extraction* (IR) *e Natural Language Processing* (NLP).

Na pesquisa realizada por Allam em (Allam and Haggag 2012), os sistemas de QA são tipicamente divididos em dois grupos, os de domínio aberto que são capazes de responder a uma grande variedade de questões transversais a vários domínios de informação como é o caso da informação disponível na *world wide web (WWW)* e os de domínio fechado. Estes últimos, à data da pesquisa realizadas, implicavam habitualmente a definição de ontologias para representar o conhecimento do domínio específico.

De acordo com Allam, um sistema de QA típico é composto por três componentes: "*Query Processing Module"* responsável pela classificação da questão, "*Document Processing Module"* cujo objetivo principal é a tarefa de IR e "*Document Processing Module"* sobre o qual recai a responsabilidade de extração da resposta.

A utilização de IA na investigação em QA nos últimos 6 anos tem sido intensificada, assim como os resultados obtidos continuam a ser constantemente melhorados de ano após ano (Cortes, et al. 2021).

Uma das subáreas de investigação em QA é o *Machine Reading Comprehension* (MRC). O conceito de MRC vem do entendimento humano sobre o texto. Ou seja, uma das formas de perceber se um leitor entendeu o texto que acabou de ler, será através da obtenção de respostas para questões sobre o texto que este acabou de ler. Com o MRC o objetivo é conseguir que a máquina seja capaz de ler um conjunto de passagens de texto e responder a questões que lhe sejam colocadas no contexto dessas passagens.

A utilização de MRC tem imensa aplicabilidade em sistemas com NLP e nos motores de pesquisa inteligentes em particular, motivo pelo qual as empresas responsáveis pelos principais motores de pesquisa como o Bing da Microsoft (Microsoft 2019) ou Google (Simonite 2019) já utilizam MRC nos seus produtos.

Assim, tendo em conta o âmbito do projeto e aplicação à EV, nos próximos dois subcapítulos, será apresentado o estado da arte nas duas vertentes identificadas como as mais importantes para conseguir os objetivos principais do projeto, nomeadamente uma revisão sobre o tema da análise de intenções e *slot tagging,* técnicas bastante utilizadas em agentes conversacionais (Tur, Hakkani-Tur and Heck 2010), bem como sobre o tema de *Machine Reading Comprehension* (MRC). Estes serão apresentados com maior detalhe, bem como os trabalhos realizados e técnicas IA mais utilizadas.

Tendo em conta que no âmbito deste projeto é pretendido a implementação de modelos que permitam ler o texto do manual escolar e responder a questões nesse contexto, o foco será maior no estado da arte relativo ao MRC.

2.3.1 Análise de intenções e Slot Tagging

A análise de intenções, frequentemente referida na investigação como *Intent Determination* (ID), é uma das tarefas de classificação necessárias para permitir que a partir de uma determinada questão colocada pelo utilizador, seja possível entender que tipo de questão está a ser colocada, permitindo assim ao sistema determinar o tipo de resposta que será mais adequada apresentar. Por exemplo, na pergunta "Qual o caminho para o Museu de Serralves?", a intenção do utilizador será obter informação sobre o percurso que deve tomar, assim uma classificação possível para a intenção seria *nav.directions*.

O *Slot Tagging* ou *Slot Filling* (SF), corresponde à tarefa a partir da qual é possível identificar as restrições na questão colocada pelo utilizador que devem ser utilizadas na construção da consulta à fonte de dados. No exemplo referido acima, a restrição a aplicar na obtenção do percurso a apresentar como resposta, seria "Museu de Serralves", ou seja, o sistema identificaria por exemplo como *slot*, @TO{Museu de Serralves}.

Assim, do ponto de vista formal, a tarefa de ID consiste numa tarefa de classificação para prever uma dada intenção y^i e a tarefa de SF consiste em atribuir etiquetas a uma dada sequência de entrada $x = (x_1, x_2, \dots, x_T)$ obtendo a sequência de saída $y^s = (y_1^s, y_2^s, \dots, y_T^s)$.

A execução destas duas tarefas são essências para permitir ao sistema determinar a ação que deverá executar.

Nos anos 90, DARPA (*Defense Advanced Research Program Agency*) iniciou o projeto ATIS (*Airline Travel Information System*). Este projeto disponibiliza um *dataset* com enunciados como "*I want to fly to Boston from New York next week*" para a realização de tarefas de ID e SF, sendo este um dos *datasets* públicos mais utilizados e que mais potenciou a investigação na área nos últimos anos (Weld, et al. 2021).

Como medidas de avaliação das tarefas de ID e SF, são habitualmente calculadas um *error rate* (ER) para ID e *F1-Score* para SF.

$$ER_{ID} = \frac{\#\ misclassified\ utterances}{\#\ utterances} \quad (5)$$

Para a avaliação de ID, é utilizada uma métrica simples, calculando a razão entre o número de enunciados com intenção mal classificada (# *missclassified utterances*) e o número de enunciados totais (# *utterances*) no *dataset.*

Para a avaliação do SF, é frequentemente utilizado o *F1-Score*, uma métrica bastante utilizada em tarefas de classificação de sequências em NLP.

Para o cálculo do *F1-Score* são utilizadas as seguintes formulas:

$$Recall = \frac{\#\ correct\ slots\ found}{\#\ true\ slots} \quad (6)$$

$$Precision = \frac{\#\ correct\ slots\ found}{\#\ found\ slots} \quad (7)$$

$$\text{F1-Score} = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (8)$$

O *F1-Score* mede a sobreposição média entre a previsão e as classificações que são verdadeiras. Quando maior for *F1-Score* melhor é a performance do modelo na tarefa em questão.

Nos primeiros trabalhos de investigação realizados para ID e SF, estas tarefas eram habitualmente tratadas de forma independente.

Para a tarefa de ID, podem ser utilizados modelos de *machine learning* como os que foram utilizados no trabalho de Zhang (Zhang and Lee 2003) no qual realizou uma comparação entre *Nearest Neighbors* (NN), *Naïve Bayes* (NB), *Decision Tree* (DT), *Sparse Network of Windows* (SNoW), ou *Support Vector Machines* (SVM), tendo obtido melhores resultados com a utilização de SVM.

Estes modelos de *machine learning* foram os mais utilizados nos trabalhos de investigação para a tarefas de ID até 2015, surgindo a partir desta data um interesse crescente na utilização de diferentes modelos de *deep learning* (Weld, et al. 2021).

No trabalho proposto em (Liu and Lane 2016), os autores utilizam *Attention Mechanism* (detalhado no ponto 2.3.2) com *Recurrent Neural Networks* (RNN) para a construção de um modelo para realizar em simultâneo a tarefa de ID e SF.

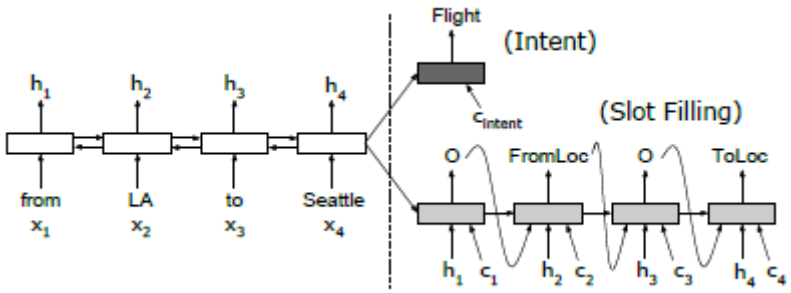

Figura 4 - Modelo de *encoder-decoder* para detenção conjunta de ID e SF utilizando *Attention* e alinhamento (Liu and Lane 2016).

Na arquitetura *encoder-decoder* com modelos utilizando *Attention Mechanism*, estes são capazes de mapear sequencias de diferentes tamanhos quando não existe disponível informação de alinhamento. No entanto, o alinhamento é bastante importante na tarefa de SF, uma vez que

neste caso, o alinhamento é um requisito obrigatório entre a sequência de entrada e a sequência de saída, pelo que estes modelos de RNNs baseados em alinhamento são bastante adequados.

No modelo apresentado é utilizada uma rede *Bi-LSTM* (*BiDirectional Long Short Term Memory Network*) para o *encoder*, e o último estado desconhecido obtido da rede (*hidden state*) é passado ao *decoder* tanto para a classificação de intenção como para etiquetar a sequência de palavras. Ambas as tarefas são treinadas em simultâneo, utilizando no *decoder* os vetores de alinhamento h_i a partir dos *hidden states* do *encoder* e o vetor de atenção c_i calculado como a média pesada dos *hidden states*. Com este modelo, os autores conseguiram obter resultados *state-of-the-art* quer no treino das tarefas individuais quer no treino conjunto, com um *F1-Score* de 95.78 para SF.

Tendo por base o mesmo conceito de treino de um modelo conjunto para as duas tarefas, os autores (Goo, et al. 2018) desenvolveram neste trabalho o conceito para melhorar a performance do modelo na tarefa de SF, considerando que existe uma forte relação entre a classificação de intenção e os *slots* a identificar, introduzindo aquilo que denominam de *slot-gated mechanism*.

O conceito do *slot-gated* passa por combinar os vetores de contexto da intenção e de *slot* fazendo passar ambos por este *gate,* cujo resultado será então utilizado na função final de classificação de cada *slot* identificado na sequência.

O trabalho de (Chen, Zhuo and Wang 2019) propõe um modelo de classificação conjunta de ID e SF baseado em BERT *(Bidirectional Encoder Representations from Transformers)*. BERT é um modelo desenvolvido por uma equipa de investigação da *Google* (Devlin, et al. 2019), pré-treinado com grande volume de dados a partir de textos do BooksCorpus e Wikipedia em inglês, mascarando texto para prever a palavra em falta, assim como na tarefa de prever a próxima frase. Com a utilização do BERT e CRF *(Conditional Random Fields*), os autores conseguiram um dos melhores resultados à data do artigo, com 97.9 de acurácia para ID e *F1-Score* de 96.0 para SF. Desde então, o modelo BERT, seguindo os mesmos modelos de treino, tem sido pré-treinado tendo por base corpus de texto em diferentes línguas entre as quais o português no trabalho de (Souza, Nogueira and Lotufo 2020) onde os autores utilizaram BrWaC (*Brazilian Web as Corpus*) (Wagner, et al. 2018), tendo superado os resultados obtidos com BERT multilingue nas tarefas de: *sentence textual similarity*, *recognizing textual entailment* e *named entity recognition* .

No trabalho de (Wang, Shen and Jin 2018), voltam a utilizar o conceito de que a classificação de intenção da frase e a sequência de etiquetas dos *slots* estão diretamente relacionados e por essa razão apresentam um modelo com duas BiLSTM para assincronamente tratarem cada uma das tarefas, mas partilhando os *hidden states* produzidos por ambas. Com esta abordagem os autores conseguiram obter os melhores resultados publicados até ao momento para o *dataset* ATIS com *F1-Score* de 96.89% para SF e 98.99% de acurácia para ID.

2.3.2 Machine Reading Comprehension

Machine Reading Comprehension (Liu, et al. 2019) pode-se definir como a tarefa de NPL através da qual uma máquina é capaz de ler e entender de modo a conseguir responder a uma questão num dado contexto.

Tendo em conta os objetivos deste projeto de mestrado, a utilização de MRC integrada em cenário de pesquisa, utilizando como contexto os textos dos manuais escolares, será certamente uma das técnicas a ter em consideração para a apresentação de respostas de âmbito fechado. Na Figura 5, é apresentado um exemplo prática dessa utilização num cenário de pesquisa web.

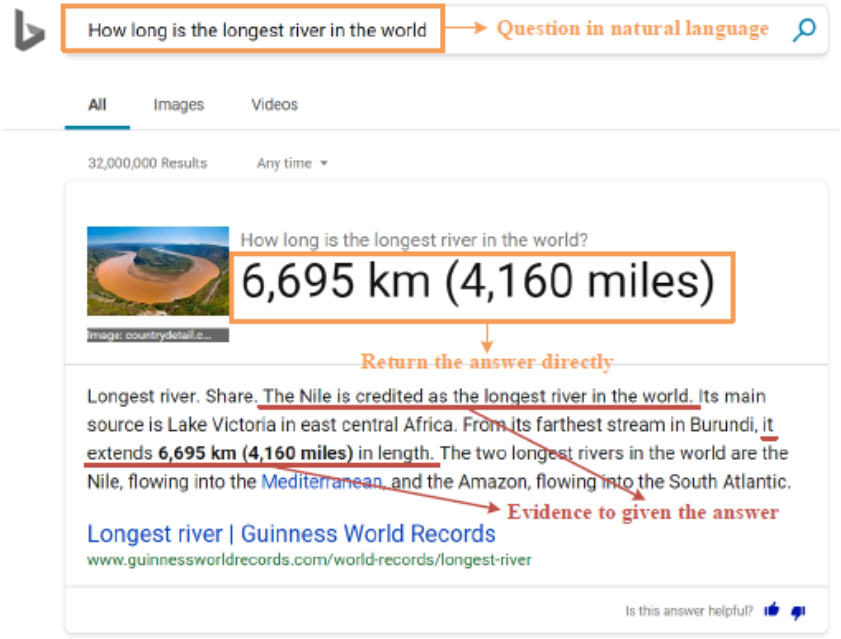

Figura 5 – Do trabalho de (Liu, et al. 2019) exemplo de utilização de MRC numa pesquisa web

No estudo de Liu, et al. podemos verificar que os primeiros sistemas com alguma relevância de *Machine Comprehension* (MC) e *Question Answer* (*QA*) apontam para os anos 70 com o sistema QUALM (Lehnert 1977).

A investigação em MRC esteve bastante parada e praticamente ignorada nos anos 80 e 90, principalmente devido às exigências de capacidade de computação em NLP (Jurafsky and Martin 2000) assim como na dificuldade em encontrar *datasets* com volume e qualidade necessária para a tarefa.

Em 1999, (Hirschman, et al. 1999), com a disponibilização de um *dataset* de MRC com textos de histórias do terceiro ao sexto ano de escolaridade para responder a perguntas do tipo "wh" (*what, where, when, why, and who*), a par de conferências realizadas no início dos anos 2000, como TREC QA (Ahn, et al. 2004) e QA CLEF (Magnini, et al. 2004), vieram promover a investigação e desenvolvimento de QA.

Nesta altura os projetos desenvolvidos eram fundamentalmente sistemas baseados em regras e em técnicas básicas de *machine learning.* São exemplo os trabalhos de (Riloff and Thelen 2000) com o sistema baseado em regras Quarc, ou de (Hirschman, et al. 1999) que utilizou *bag-of-words* para representar as frases das questões e dos textos que definem o contexto, como vetores de representação de palavras e, desta forma, escolher as palavras que pertencem a ambos como candidatos à resposta.

Em 2007 com o aparecimento de alguns *datasets* como DBPedia (Auer, et al. 2007) ou *Freebase* (Bollacker, et al. 2008) e mais tarde em 2011 com o sistema Watson da IBM (Ferrucci, et al. 2010) a ganhar o concurso no jogo *Jeopardy*, a investigação em QA recebe bastante atenção por parte da comunidade de investigação em *NLP.*

No âmbito do trabalho proposto neste projeto, com os conteúdos do manual escolar digital, bem como de todos os conteúdos digitais e meta informação associada que os complementam, será bastante relevante possibilitar ao aluno uma experiência de pesquisa do conhecimento e esclarecimentos que procura, que não seja meramente constituída por mecanismos de indexação de dados tendo em conta os termos de pesquisa.

No trabalho de (Chaudhri, et al. 2013), foi desenvolvido um projeto protótipo ("*Inquire Biology*"), com o objetivo de criar um manual escolar inteligente, capaz de responder a respostas colocadas pelo aluno sobre a informação presente num livro do ensino secundário de biologia. Para além das respostas, o sistema é também capaz de sugerir perguntas que são apresentadas quando o aluno seleciona do texto no manual. Este trabalho incorpora várias técnicas de IA, sendo baseado na representação formal do conhecimento dos conteúdos do livro, NPL para entender as questões colocadas e métodos de raciocínio sobre a base de conhecimento para inferência das respostas a apresentar.

Neste protótipo, os autores utilizaram para representação da base de conhecimento e inferência, o sistema pericial *Knowledge Machine 2.0* (Clark, Porter and Works 2004). Para a geração de respostas, os autores utilizam o grafo da base de conhecimento para gerar automaticamente um banco de questões, sendo estas comparadas com a questão utilizada pelo utilizador e apresentando a questão mais semelhante como sugestão. Uma vez escolhida a questão, o sistema gera uma resposta em língua natural tendo por base a base de conhecimento e o trabalho desenvolvido por Banik, et al. (2012).

Em 2016, com o trabalho de Bahdanau, et al. (2016) surge o mecanismo de atenção em redes neuronais ("*Attention Mechanism*"). Em 2015 e 2016 surgem três novos *datasets*: CNN & Daily Mail (Hermann, et al. 2015), SQuAD (Rajpurkar, et al. 2016) e MS MARCO (Nguyen, et al. 2016). De acordo com o estudo sobre evolução e tendências em MRC realizado por Liu, et al. (2019), a evolução da investigação em *deep learning* a par da disponibilização destes *datasets*, serão as razões principais para o crescente número de estudos de investigação na utilização de diferentes técnicas de inteligência artificial a partir desta data.

O *Attention Mechanism* em *deep learning*, consiste em incorporar no *decoder* de uma rede neuronal, a capacidade de dar maior "atenção" a determinadas palavras do *input* que sejam mais relevantes para determinar o *target* do *decoder*.

Numa arquitetura *encoder-decoder* típica, por exemplo com RNN, o mecanismo de atenção fica colocado entre o *encoder* e o *decoder* (Figura 6).

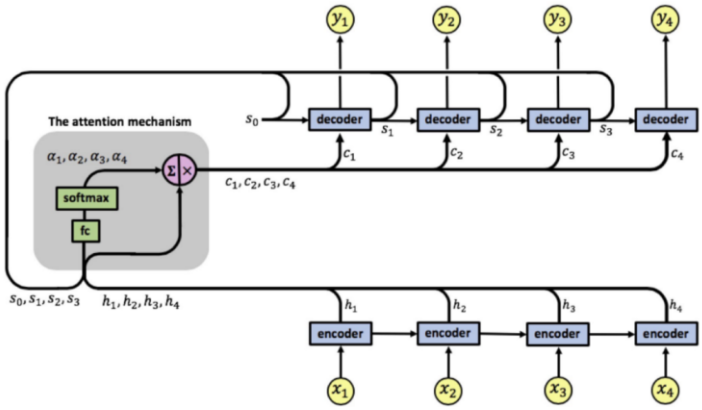

Figura 6 – Diagrama típico do mecanismo de atenção com RNN (Manu 2021)

O mecanismo de atenção, em cada passo i do *decoder,* recebe como input todos os *hidden states* do *encoder* (neste caso h_1, h_2, h_3, h_4) e o estado $s_{(i-1)}$ do *decoder*. Utiliza uma camada *Fully Connected* (FC) e a função *softmax,* equação (9), para calcular os pesos de atenção $\alpha_{i1}, \alpha_{i2}, \alpha_{i3}, \alpha_{i4}$. Estes pesos são utilizados para calcular os vetores de atenção (também denominados vetores de contexto) para permitir ao *decoder* dar maior importância aos termos mais relevantes na predição final.

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \qquad (9)$$

No processo de treino, a camada FC do mecanismo de atenção é treinada em simultâneo com o *encoder* e *decoder* utilizando o algoritmo habitual das redes neuronais de *back propagation* (Rumelhart, Hinton and Williams 1986). Os erros calculados na predição, são propagados para trás ao longo da rede pelo *decoder* passando pela FC do mecanismo de atenção até ao *encoder*.

A utilização do *Attention Mechanism* para a tarefa de MRC começou por ser utilizado inicialmente com *Recurrent Neural Networks* (RNN) (Bahdanau, Cho and Bengio 2016) e com elevado sucesso da rede BiDAF (Seo, et al. 2016) em *MRC.* No entanto, as RNN sendo alimentadas sequencialmente, são bastante demoradas na atividade de treino em textos de grande dimensão, nomeadamente pelo facto de não ser possível paralelizar o seu processamento, devido à sua natureza sequencial.

O elevado tempo de processamento em MRC com RNN, tem levado os investigadores a estudar outras arquiteturas menos dependentes do processamento sequencial, como foi o caso do

trabalho realizado por (Yu, et al. 2018) com a QANet, que, combinando apenas *Convolutional Networks* com *Attention Mechanism*, conseguiu manter o modelo competitivo e reduzindo de 3 a 13 vezes o tempo de treino e de 4 a 9 vezes o tempo de inferência.

O tempo de processamento é algo bastante relevante já que quanto mais reduzido for este tempo, mais rapidamente conseguem os investigadores obter resultados e re-treinar os modelos, assim como treinar os modelos com maior volume de dados, aumentado a performance dos mesmos.

Nos trabalhos mais recentes e com melhores resultados no *SQuAD leader board*, encontram-se os modelos baseados em *Transformers* introduzidos no trabalho *"Attention is All you Need"* por (Vaswani, et al. 2017). Estes modelos, não utilizam nem RNN nem CNN para a tarefa de MRC como apresentado no trabalho de (Zhang, Yang and Zhao 2020).

A investigação em NLP no geral e a elevada aplicabilidade de MRC em diversos casos de uso de interação homem-máquina, como por exemplo os motores de pesquisa ou os sistemas conversacionais, tem levado nos últimos anos grandes empresas como a Google (Radu and Nan 2016) (Devlin, et al. 2019) ou a Microsoft (Shen, et al. 2017) (Group 2017), a investir na investigação na área, pelo que se espera que esta continue a estar em constante evolução.

A grande maioria dos estudos realizados em MRC desde 2016, são trabalhos realizados sobre o SQuAD (*Stanford Question Answering Dataset*) *dataset* o qual é disponibilizado atualmente na versão 2.0. O desafio deste *dataset* é não só responder a questões no contexto dado, mas também abster-se quando não existe uma resposta para a questão colocada. A performance dos trabalhos realizados é habitualmente medida através de dois indicadores: o *F1-Score* e *Exact Match* (EM), apresentando o *Leader Board* à data deste documento *F1-Score* de 93.214 e EM de 90.939 para a rede IE-NET submetida em 4 de junho de 2021.

Nas secções seguintes é apresentada uma arquitetura genérica para a atividade de MRC, seguido da apresentação com maior detalhe das abordagens, arquiteturas utilizadas e resultados alcançados por alguns dos trabalhos mais relevantes realizados na área, onde é possível verificar que os modelos baseados em *Transformers* são atualmente os que melhores resultantes têm obtido em MRC.

Arquitetura típica para a tarefa de MRC

Conforme documentado e resumido no trabalho de Liu (Liu, et al. 2019), na Figura 7 é apresentada uma arquitetura típica para a realização da tarefa de MRC composta por 4 componentes: *embeddings*, *feature extraction*, *context-question interaction* e *answer prediction*.

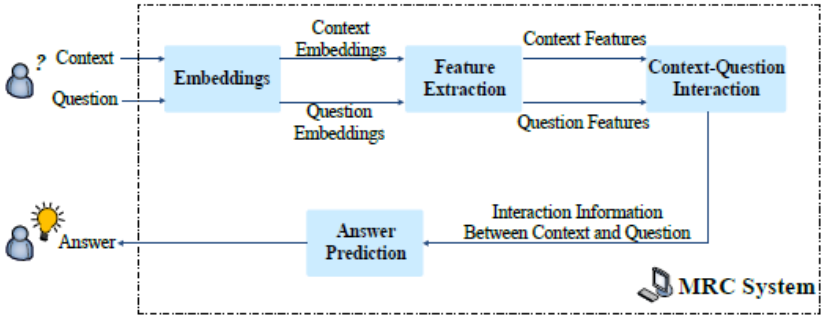

Figura 7 - Arquitetura típica para tarefa de MRC (Liu, et al. 2019)

Nos trabalhos analisados relativos às tarefas de MRC, a utilização das redes neuronais tem sido a prática habitual para a implementação destas diferentes componentes, tanto pelos resultados obtidos como pela capacidade de generalização e escalabilidade.

As componentes apresentadas no diagrama da Figura 7, são implementadas com uma ou várias camadas de diferentes tipos de redes neuronais como: *Feed Forward NetWorks* (FFN), CNN, RNN, *Transformers.*

Tendo em conta a natureza das redes neuronais, todo o processamento exige que os dados que alimentam a rede, sejam fornecidos com representações em vetores numéricos.

Embeddings: Dado que as redes neuronais não aceitam dados em língua natural, existe a necessidade de encontrar representações vectoriais numéricas para serem utilizados como dados de entrada. Tendo por base os textos da questão colocada e do documento onde pretendemos obter a resposta, diferentes técnicas podem ser utilizadas para obter a representação vectorial necessária. Atualmente, estas representações vetoriais podem ser efetuadas recorrendo a técnicas sem noção de contexto como *bag-of-words* ou TF-IDF, através de *embeddings* criados por modelos que incorporam, através de métodos estatísticos, noção de contexto como *word2vec* (Mikolov, Sutskever, et al. 2013) ou *GloVe* (Pennington, Socher and Manning 2014), ou através de modelos pré-treinados em corpus de texto de grande dimensão, como ELMo (Peters, et al. 2018). No estudo realizado, os modelos mais utilizados pelos investigadores para a obtenção de *Embeddings*, são o *GloVe* e o *ELMo*. Embora a maior parte destes modelos de *embeddings* existentes tenham tido por base uma aprendizagem baseada em corpus de textos em inglês, é possível encontrar alguns com representações treinados em português, nomeadamente o *FastText* (Bojanowski, et al. 2017) disponibilizado pelo *Facebook* para várias línguas.

Feature Extraction: De modo a melhor entender o contexto e a questão, o objetivo deste módulo será obter informação mais contextualizada, evidenciando características que serão relevantes para o componente seguinte na rede. Para a realização desta atividade, as *deep neural networks* mais utilizadas são RNN como é exemplo a rede Bi-Directional Attention Flow (BiDAF) em (Seo, et al. 2016) ou CNN na rede QANet proposta por (Yu, et al. 2018) ou a utilização de *Transformers* com é exemplo a rede BERT (Devlin, et al. 2019).

Context-Question-Interaction: Este módulo tem como principal objetivo estabelecer uma correlação entre o contexto e a questão de modo a permitir obter melhores resultados na previsão da resposta. O *Attention Mechanism* é amplamente utilizado nesta atividade, permitindo evidenciar as partes do contexto mais candidatas a pertencer à resposta.

Answer Prediction: Este módulo é responsável por apresentar a resposta final baseada na informação acumulada dos módulos anteriores. Dependendo do tipo de resposta final, este módulo apresenta diferentes resultados para a determinação da resposta. Por exemplo, no tipo de resposta *span extraction,* que consiste em obter a partir de uma passagem de texto o segmento que servirá de resposta, o resultado final é apenas indicação sobre o início e o fim do segmento, enquanto que numa resposta aberta, existem mecanismos mais complexos para gerar a resposta, nomeadamente, incluindo palavras que não constam do contexto.

MRC baseado em RNN

No processamento de língua natural, o texto pode ser visto como uma sequência de palavras, tal como numa sequência temporal. Nas arquiteturas de redes neuronais e em particular no *deep learning*, as RNN tem sido amplamente utilizadas para o desenvolvimento de modelos preditivos com informação sequencial (Zhang, Wang and Liu 2018).

A característica principal das RNN que as tornam adequadas para trabalhar com este tipo de informação, é a capacidade que estas têm de, trabalhando em ciclo, passando o *hidden state* de um ciclo anterior como input para o ciclo seguinte, influenciando assim a predição num dado momento.

Na Figura 8 é ilustrada a principal diferença no fluxo de informação numa RNN e numa *Feed Forward Neural Network*, onde a primeira é capaz de gerar um novo estado a partir dos dados de entrada e do estado anterior, em que $h_t = f_W(h_{t-1}, x_t)$.

Do ponto de vista formal, temos que um dado estado h num momento temporal t, é obtido a partir de uma qualquer função f com os parâmetros W a partir do estado anterior h_{t-1} e um dado vetor de entrada x_t no mesmo momento temporal.

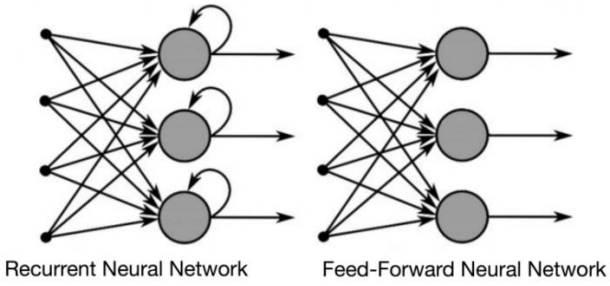

Figura 8- Fluxo de informação numa RNN e numa FFNN[4]

[4] Imagem obtida em https://builtin.com/data-science/recurrent-neural-networks-and-lstm

Esta capacidade das RNN potenciou largamente a utilização de redes neuronais em tarefas de NLP, dado que nas sequências de texto, é muito importante que ao submeter um vetor com a representação de uma dada palavra na sequência, esta seja contextualizada com o *hidden state* obtido das palavras anteriores e assim conseguir melhores resultados na predição da próxima palavra na sequência (Mikolov, Karafiát, et al. 2010).

As RNN são uma classe de redes neuronais em cadeia em que a cada momento temporal t, a rede recebe informação calculada no momento temporal anterior $t-1$. Dado que a saída de uma RNN no momento temporal t é obtido em função de todos os dados de entrada dos momentos temporais anteriores, então podemos dizer que de alguma forma é mantida memória. A parte de uma redes neuronais onde é mantido algum estado ao longo do tempo, é chamado de *memory cell* ou apenas *cell* (Géron 2020). A Figura 9 ilustra o encadeamento e a respetiva transição de informação obtida por uma dada função de ativação A no momento $t-1$ para uma *cell* no momento t.

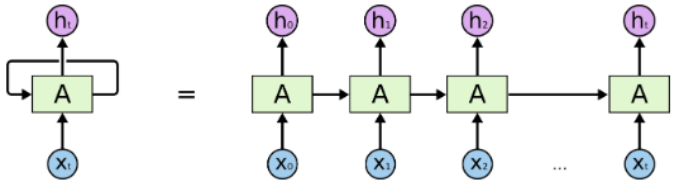

Figura 9 - Arquitetura típica do encadeamento de uma RNN[5]

Com esta arquitetura podemos obter os *outputs* de cada momento temporal, importante por exemplo para uma tarefa de *sequence-to-sequence*, ou obter apenas o resultado final para uma tarefa de classificação.

Um dos problemas identificados nas redes RNN é o facto de que em sequências que sejam algo longas, é o problema do desaparecimento do gradiente (*gradient vanishing*) limitando a RNN em manter alguma relação (memória) dos termos mais distantes. Isto acontece porque à medida que o gradiente vai sendo aplicado no processo de *back propagation* (Rumelhart, Hinton and Williams 1986) , a partir de determinado momento este vai tendendo para zero, fazendo com que a rede deixe de aprender ou aprenda muito lentamente.

Para resolver esta questão surgem as redes RNN LSTM (*Long Short Term Memory Network)* (Hochreiter and Schmidhuber 1997). Na Figura 10 é apresentado um diagrama ilustrativo do Ciclo de repetições numa rede LSTM.

[5] http://colah.github.io/posts/2015-08-Understanding-LSTMs/

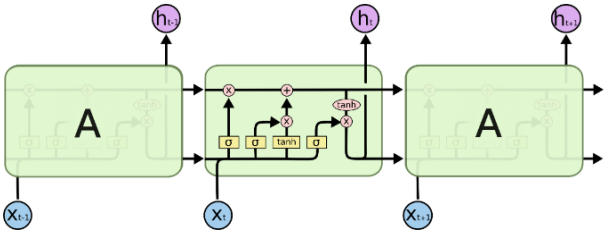

Figura 10 - Ciclo de repetição numa LSTM[5]

Nas redes LSTM cada *cell* é composta por quatro camadas de redes neuronais sendo que três, ativadas pela função sigmoid σ, funcionam como *gate* decidindo qual a informação de que deve ou não passar para a *cell* seguinte. Podemos observar que o *output* da *cell* anterior leva dois caminhos diferentes, sendo que um deles (o do topo da imagem) leva diretamente mais informação para a saída da seguinte.

Assim, tendo em conta a excelente performance das RNN em tarefas de NLP (Sundermeyer, Schlüter and Ney 2012), estas têm sido igualmente uma das opções mais utilizadas na tarefa de *MRC*.

Em 2016, surge BiDAF *(Bi-Directional Attention Flow)* de (Seo, et al. 2016), um dos trabalhos com utilização de *RNN* mais relevantes à data e que tira partido também do mecanismo de atenção. O mecanismo de atenção é conceptualmente bastante adequado para a tarefa de *MRC* já que para obtermos a resposta a uma pergunta dado um determinado contexto, é praticamente de senso comum, que a resposta será encontrada dando atenção diferente a determinadas partes do contexto e da pergunta.

Na Figura 11 estão representadas as seis camadas da arquitetura da rede BiDAF.

As três primeiras camadas combinam de forma independente representações vetoriais numéricas das palavras *(embeddings*) produzidas a partir de níveis de granularidade diferente, quer para a questão quer para o contexto. Estas representações vetoriais, foram obtidas, ao nível das palavras a partir de modelo pré-treinado *GloVe,* tendo as representações obtidas ao nível dos caracteres, sido realizada utilizando CNNs.

Os *hidden states* obtidos da camada *Contextual Embed Layer* são obtidos recorrendo a uma Bi-LSTM capturando desta forma as interações temporais entre cada palavra da sequência quer da questão quer do contexto nos dois sentidos.

Os estados obtidos da camada de contexto são então passados à camada de atenção bidirecional, obtendo os vetores de atenção com os pesos nas palavras mais relevantes quer no sentido da questão para o contexto, quer no sentido do contexto para a questão. O output resultante desta camada, é uma matriz onde cada coluna corresponde a um vetor que combina o vetor de atenção com o vetor da camada *contextual embeding* para cada palavra do contexto. Cada vetor obtido das colunas da matriz, pode ser entendido como a representação da relevância da questão para cada palavra do contexto.

Na *Modeling Layer,* é mais uma vez utilizada uma Bi-LSTM que recebe como entrada a matriz obtida da camada anterior e evidência a interação entre as palavras do contexto já combinadas com as palavras da questão.

No *Output Layer* é obtida uma distribuição de probabilidades de cada palavra do contexto, a partir da qual é obtida a predição do início e fim da sequência de palavras que representam a resposta.

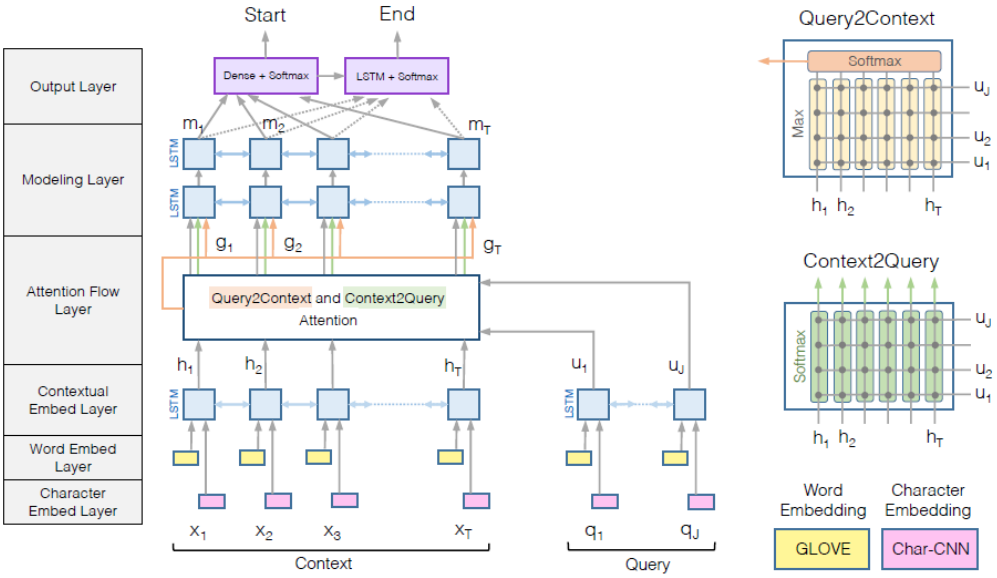

Figura 11 - Bi-Directional Attention Flow (Seo, et al. 2016)

No trabalho de MRC de Wang et. al (2016), foram utilizadas duas camadas de LSTM, a primeira que recebe a passagem de texto e a questão e realiza um pré-processamento combinando as duas sequências e a segunda camada que tenta encontrar o *match* da passagem em relação à questão. Na última camada utilizaram o modelo de *Pointer Network* (Ptr-Net) proposto por (Vinyals, Fortunato and Jaitly 2017), tendo sido utilizado para determinar o início e fim dos *tokens* da passagem para extrair a resposta.

Em 2017, investigadores da *Microsoft* (Shen, et al. 2017) desenvolveram a *ReasoNet (Reasoning Network)*, uma nova arquitetura de rede neuronal para MRC, a qual realiza várias voltas para explorar e raciocinar sobre a relação entre as questões, a passagem de texto e as respostas. Este processo foi inspirado no comportamento de leitura humano, quando existe a necessidade de repetir a leitura para melhor compreender o texto.

Uma das grandes novidades introduzidas pela *ReasoNet,* é o facto de que ao contrário de outras abordagens *multi-turn* em que, independentemente da complexidade das perguntas ou do texto, existe um número pré-definido de repetições, a *ReasoNet* utiliza *Reinforcement Learning* (Kaelbling, Littman and Moore 1996), para determinar quando deve continuar o processo de

raciocínio ou terminar assim que conclui que já existe informação adequada para obter a resposta.

Na Figura 12, é possível visualizar as componentes da arquitetura de rede *ReasoNet.* Aqui, as *RNN* foram utilizadas em duas das componentes:

- Componente de memoria, que corresponde a uma representação vetorial de dimensão fixa de cada palavra da passagem de texto, codificadas através de uma RNN Bidirectional permitindo assim captar para cada palavra o contexto das palavras mais próximas.
- Componente para obter os estados internos da rede, obtidos sequencialmente numa RNN para cada palavra da questão, tendo as células desta como input os estados anteriores e um vetor de atenção x_t calculado a partir do estado atual e dos vetores de memoria de contexto.

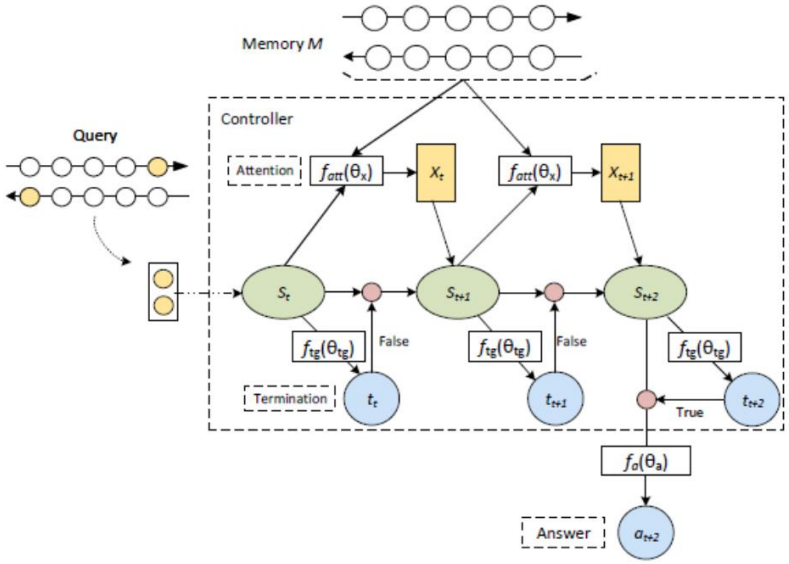

Figura 12 - Componentes da arquitetura ReasoNet (Shen, et al. 2017)

Os resultados obtidos com a *ReasoNet* à data de submissão do artigo, alcançaram resultados melhores do que a maior parte dos modelos existentes na altura, tendo ocupado o segundo lugar do quadro de liderança no desafio do *dataset* SQuAD.

Evidenciando o envolvimento da *Microsoft* na investigação em MRC, no mesmo ano da *ReasoNet* surge a *R-NET* (Group 2017), a qual, como apresentado na Figura 13, utiliza igualmente *RNN* nas principais componentes do modelo. Este modelo consiste em quatro partes: 1) *RNN* como *encoder* para construção separada das representações da questão e da passagem, 2) uma camada com uma porta que combina a questão com a passagem, 3) uma camada de RNN que combina e agrega informação das palavras dentro da passagem, e 4) à semelhança do trabalho de Wang e Jang em 2016, utilizam *Pointer Network* como camada preditiva para obter quais os limites onde se encontrar a resposta adequada na passagem.

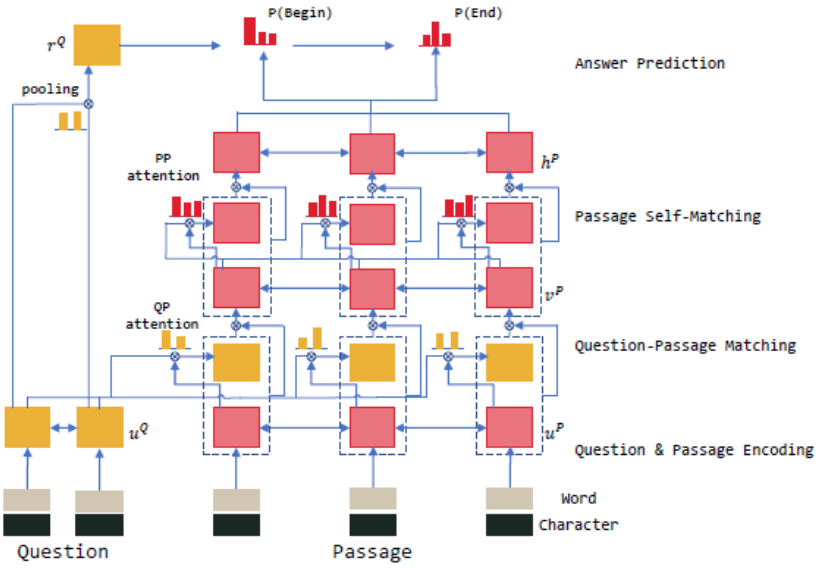

Figura 13 - Componentes da R-NET (Group 2017)

MRC baseado em CNN

As *Convolutional Neural Networks* (CNN) são um tipo específico de redes neuronais, cuja utilização tem obtido resultados surpreendentes nas áreas de visão por computador e processamento de imagem, nomeadamente para o reconhecimento de objetos, classificação de imagens e processamento de vídeo (Liu, Deng and Yang 2019), assim como no processamento de língua natural.

No processamento de língua natural, pela natureza sequencial da informação como é o caso de documentos de texto os quais são compostos por sequencias de palavras, as RNN têm sido a opção mais habitual. No entanto, vários estudos realizados colocam as CNN como alternativa bastante válida, conseguindo inclusive superar as RNN em algumas tarefas (Dauphin, et al. 2017).

Na Figura 15 é apresentada uma arquitetura típica para a tarefas de classificação de sentimento a partir de textos (neste caso em particular mensagens no *Twitter*[6]).

Ao contrário do que acontece numa rede RNN, em que na geração sequencial dos estados internos do processamento de cada palavra estes vão recebendo e acumulando informação de contexto dos estados anteriores, nas CNN o processamento de todos os elementos acontece em simultâneo e por esta razão a noção de contexto de uma palavra necessita de ser trabalhada de forma diferente. Esta característica da forma como os dados de entrada são processadas sequencialmente nas RNN e em simultâneo nas CNN, é também uma vantagem destas últimas já que permitem o processamento em paralelo, sendo assim bastante mais rápidas em determinadas tarefas, quer no treino quer na predição.

[6] https://www.twitter.com

Neste tipo de redes, assume-se que o contexto de uma palavra é dado pelas n palavras mais próximas na frase (à frente ou atrás) e não é relevante toda a sequência de palavras (Dauphin, et al. 2017).

Assim, para extrair as características principais da sequência de texto em análise (*feature extraction)*, é realizada a operação de convolução 1D.

A operação de convolução 1D consiste na aplicação de um filtro de dimensão fixa (é habitual utilização de 2,3 ou 5 dependendo da análise se focar em *2-grams, 3-grams ou 5-grams*) sobre as representações vetoriais de cada palavra (*word embeddings*).

Na Figura 14, é apresentada uma ilustração da operação de convolução sobre a frase "*cat sitting there",* utilizando um filtro de 2 vezes a dimensão do *word embedding*. A designação 1D resulta da aplicação unidirecional do filtro numa janela deslizante, aqui representada no sentido descente dos vetores de entrada. Como resultado, vamos obter a matriz de *feature map* a qual depende do número de filtros aplicados. Este processo da janela deslizante tem também a vantagem de analisar o contexto da palavra tanto em relação aos n termos anteriores como aos posteriores, sendo bastante mais simples do que, por exemplo, no caso das Bi-LSTM.

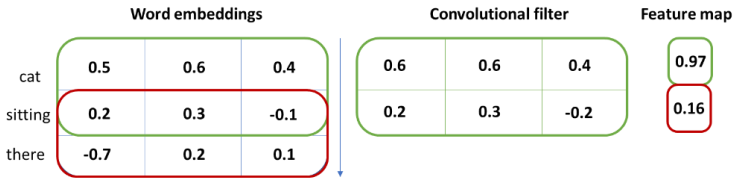

Figura 14 - Ilustração da operação de convolução com filtro 2-gram[7]

Uma vez realizadas as convoluções, é então realizada a operação de *max-pooling* cujo objetivo é transformar a matriz de *feature map* numa representação vetorial de toda a frase "*cat sitting there*".

A operação de *max-pooling* consiste em extrair de cada coluna da matriz *feature map* o valor mais elevado, construindo assim uma representação para a frase.

[7] https://cezannec.github.io/CNN_Text_Classification/

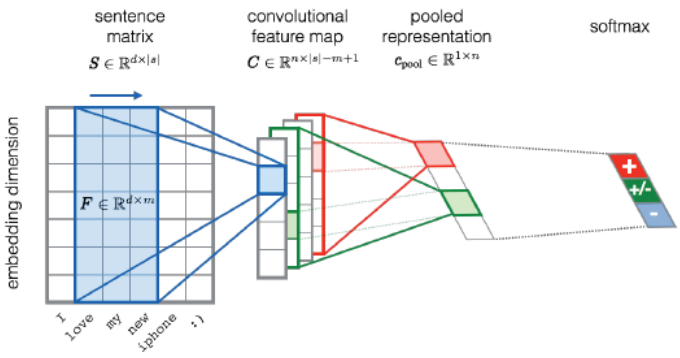

Figura 15 - Arquitetura CNN para análise de sentimento (Severyn and Moschitti 2015)

Na Figura 16 é apresentada uma arquitetura típica de uma CNN para processamento de imagem, composta por várias camadas de operações de convolução e *pooling*, sendo a última camada composta por uma ou mais camadas de redes neuronais densamente ligadas (*fully-connected)*, permitindo assim a realização de tarefas de classificação ou regressão. Esta abordagem é bastante comum igualmente na utilização de CNN no processamento de língua natural (Hu, et al. 2014).

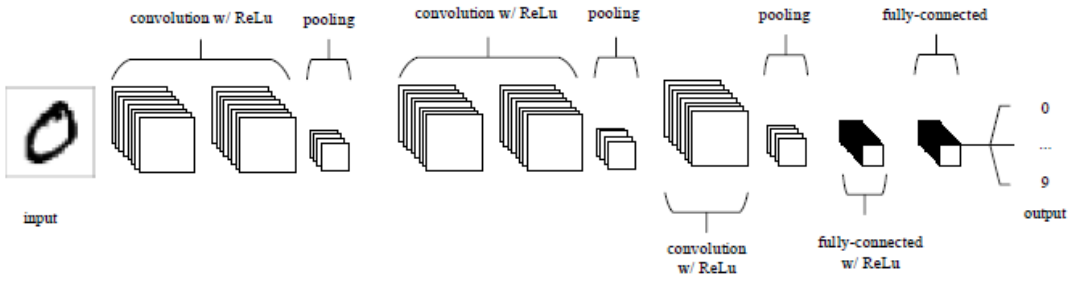

Figura 16 - Arquitetura típica de CNN em várias camadas convolucionais (O'Shea and Nash 2015)

No âmbito da tarefa de MRC um dos trabalhos mais relevantes é o trabalho de (Yu, et al. 2018), no qual apresenta a arquitetura QANet, a qual não utiliza redes recorrentes e apenas utiliza CNN com mecanismo de atenção.

Os autores evidenciam rapidez no treino e na inferência como sendo a grande vantagem desta arquitetura quando comparada no mesmo *dataset SQuAD* com arquiteturas que utilizam RNN. Esta vantagem permite também obter resultados de 84.6 de *F1-score* bastante melhor do que o melhor trabalho publicado à data do artigo que era de 81.8, segundo os autores pelo facto de ter permitido o treino de maiores volumes de dados e ciclos mais rápidos de treino e validação.

A arquitetura da QANet (Figura 17) respeita em grande parte as camadas tipicamente utilizadas na grande maioria das arquiteturas de rede para a tarefa de MRC, nomeadamente: camada de

word embeddings, *feature extraction (embedding encoder blocks),* camada de interação entre a questão e o contexto (*Context-Question-Interaction)*, *encoders* do modelo e camada de *output* que apresenta a probabilidade de cada posição ser o inicio ou o fim do segmento do contexto que servirá de resposta.

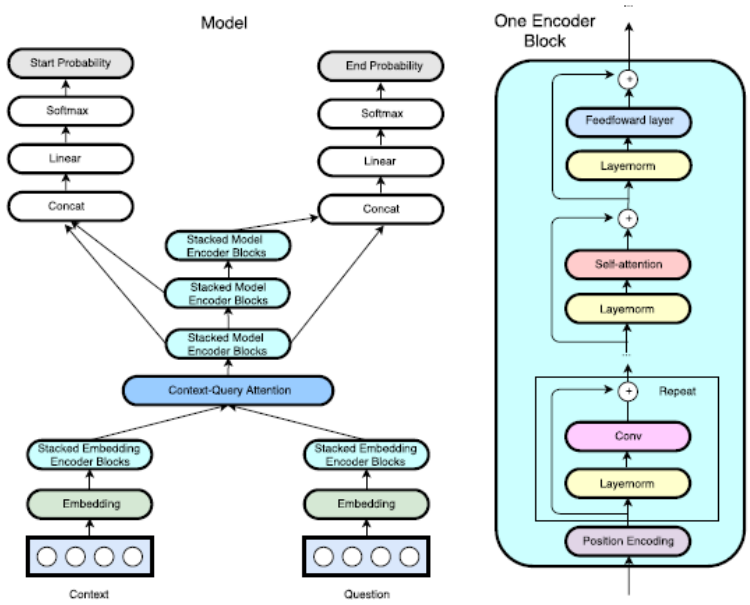

Figura 17 - Visão geral da arquitetura QANet (Yu, et al. 2018)

MRC baseado em Transformers

A arquitetura de *transformer* (Figura 18) surge em 2017 no trabalho de investigadores da Google (Vaswani, et al. 2017).

Este tipo de arquitetura tem gerado imenso entusiasmo, principalmente os modelos pré-treinados de representação de linguagem como o *OpenAI* GPT (Radford, et al. 2018) ou BERT (Devlin, et al. 2019), os quais sem qualquer camada para realização específica de uma tarefa, podem ser ajustados (*fine-tuning*) para ser adaptados a uma variedade de tarefas como por exemplo: *Question Answering, Named Entity Recognition* ou *Language Inference*.

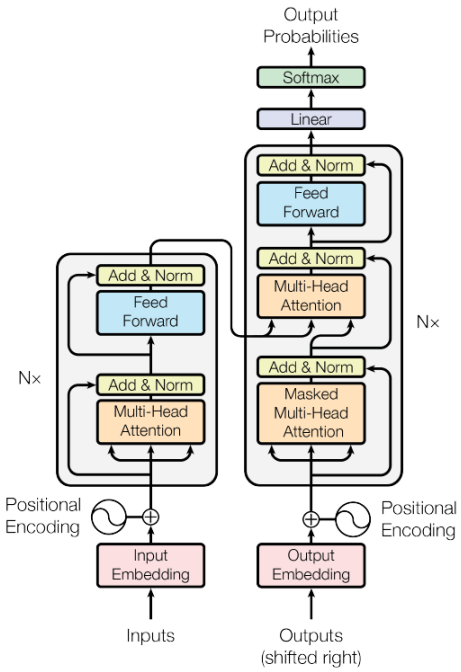

Figura 18 - Arquitetura de Transformer (Vaswani, et al. 2017)

Esta arquitetura surge com o objetivo de substituir os modelos mais utilizados até ao momento no processamento de língua natural baseados em RNN. O principal objetivo foi resolver o tempo computacional necessário nas RNN para o processamento de sequências de texto mais extensas, introduzindo um modelo capaz de receber, utilizando computação paralela, de uma só vez como input toda uma sequência de palavras e produzir uma nova sequencia como output (modelo de *encoder-decoder*).

Uma das tarefas mais habituais nos modelos *encoder*-decoder é a tarefa de tradução. No treino de uma arquitetura de *transformer* para a tarefa de tradução, o *encoder,* representado à esquerda da Figura 18, recebe como input os vetores de representação das palavras da sequência origem para a tradução (por exemplo, uma sequência em francês) e a estes vetores são adicionados vetores *positional encodings* que representam a posição de cada palavra na sequência. Estes *positional encodings* são importantes para introduzir a noção de contexto, dado que a mesma palavra poderá ter significados diferentes quando utilizada em diferentes ordens na sequência de texto.

Os vetores contextualizados de cada palavra são processados em paralelo pela camada de mecanismo de *self-attention*, de forma a obter vetores que representam a relevância de cada palavra entre si na frase de entrada. No final do *encoder* a camada de *feed forward* serve para converter os vetores de atenção numa forma que possa ser utilizada como input para o *decoder*.

Do lado direito da Figura 18, está representado o *decoder*. Na fase de treino, o *decoder,* à semelhança das primeiras duas camadas do *encoder*, recebe as representações espaciais das palavras que são destino da tradução e obtém os vetores posicionais passando de seguida estes

para uma camada de *self-attentio*n obtendo igualmente vetores de representação da relevância interna entre as palavras da frase de destino da tradução.

Os vetores de saída do *encoder* e os vetores de saída do *decoder* são ambos combinados novamente numa camada de *self-attention* cujo objetivo é obter as representações da relevância conjunta entre as representações de atenção da origem e do destino. É nesta camada que é realizado o mapeamento de relação entre as palavras de origem e de destino.

A última camada *Linear* e *softmax*, servem para realizar a transformação tendo por base o alfabeto da língua destino e a respetiva distribuição probabilísta das palavras que melhor representam a tradução.

Um dos trabalhos baseado em *transformers* mais relevantes com aplicação na tarefa de MRC é o modelo apresentado também por investigadores da *Google*, *Bidirectional Encoder Representations from Transformers* (BERT) (Devlin, et al. 2019). No BERT*,* como o próprio nome indica, a arquitetura é baseada numa série de camadas (12 ou 24) de *encoders* da arquitetura de *Transformers.* Os autores obtiveram resultado de 93.2 *F1-Score* no *dataset* SQuAD *1.1* e de 83.1 (5.1 pontos de melhoria) no *dataset* SQuAD *2.0* no teste de *Question-Answer*.

Neste trabalho a novidade relativamente a estudos anteriores (Radford, et al. 2018), foi a introdução do objetivo de incorporar contexto bidirecional nas representações, tendo sido utilizado o modelo *masked language model* (MLM) inspirado na tarefa de preenchimento da palavra em falta (*Cloze task*) (Taylor 1953). O objetivo de prever a palavra em falta, permite fundir o contexto à esquerda e à direita da palavra, pré-treinando assim bidireccionalmente o *Transformer*. Paralelamente ao MLM, o modelo foi também pré-treinado para a tarefa de prever a próxima frase.

No BERT*,* os procedimentos do modelo para treino numa tarefa específica consistem em duas fases (Figura 19): *Pre-training* e *Fine-tuning*.

Durante o *pre-training,* o modelo foi treinado sobre dados não classificados, tendo sido utilizados os *datasets BooksCorpus* (800M de palavras) e *Wikipedia* em inglês (2,500M de palavras). Estes dados são utilizados para iniciar a fase de *fine-tuning* e posteriormente todos os parâmetros são ajustados para a realização de tarefas específicas.

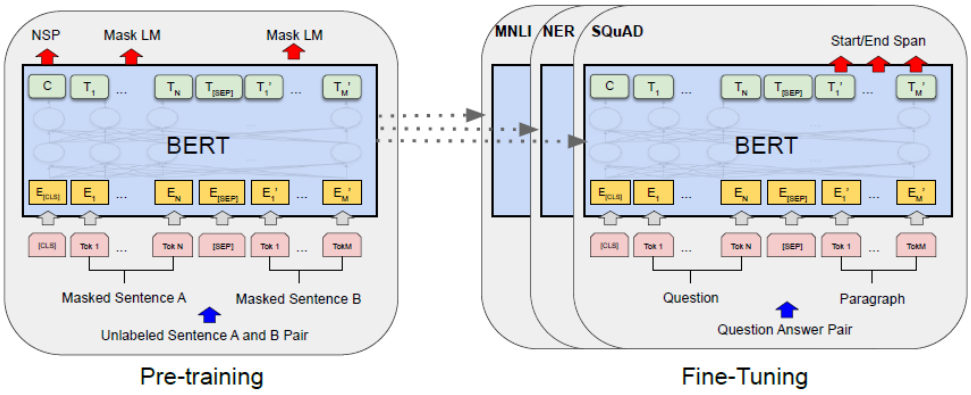

Figura 19 - Procedimentos gerais de pré-treino e ajuste à tarefa no BERT (Devlin, et al. 2019)

Como referido acima, o *pre-training* é realizado em duas tarefas:

- Na tarefa de MLM, o gerador de dados utilizado, esconde aleatoriamente 15% de *tokens* substituindo estes pelo *token* "[MASK]", e o modelo é treinado bidireccionalmente para prever o *token* em falta. Dado que este *token* [MASK] não irá constar nas sequências utilizadas no *fine-tunig,* de modo a evitar o impacto nessa fase, os autores optaram por não transformar sempre em todas as situações o *token* a esconder pelo *token* [MASK], trocando apenas em 80% das situações pelo *token* [MASK], em 10% por um *token* aleatório e 10% o gerador decide manter o *token* inalterado.
- Na tarefa de prever o próximo *token,* o modelo foi treinado para uma tarefa binária de previsão do próximo *token* a partir da geração de um *dataset* de sequências de palavras em que em 50% das vezes o próximo era efetivamente o correto (etiqueta *IsNext*) e nos restantes 50% o incorreto (etiqueta *NotNext*). O vetor resultante está representado em *C* na Figura 19.

Para o *fine-tuning*, nomeadamente na aplicação ao *MRC* o modelo está preparado para receber pares de questão e passagem de texto, aplicando *self-attention* bidirecional sobre cada par. Para cada tarefa específica que se pretende realizar o *fine-tuning*, basicamente são ajustados os dados de entrada de acordo com a tarefa pretendida e adicionadas as camadas necessárias para a tarefa em questão (BERT *Head*).

No trabalho de (Zhang, Yang and Zhao 2020) denominado de *Retro-Reader,* a preocupação principal não foi a criação de um modelo inovador de MRC mas forçar principalmente no desenvolvimento de um modelo de validação das respostas, para resolver um dos problema na investigação em MRC que é o problema das questões para as quais o contexto não tem uma resposta adequada (*answerable questions*).

O verificador de *answerable questions* foi implementado recorrendo a uma numa camada *Feed Foward Network* (FFN), que, com ensino supervisionado, classifica cada questão como tendo ou não uma resposta.

Do ponto de vista da arquitetura base do *Retro-Reader*, os autores seguiram uma arquitetura típica de *transformers*, incluindo a classificação obtida pela *FFN* em três alternativas distintas de classificação da função de perda (*loss function*) conforme detalhado no artigo dos autores.

Com esta abordagem os autores conseguiram ainda melhores resultados, nomeadamente em comparação com o modelo *BERT*, obtendo 91.4 de *F1-Score* no *dataset SQuAD 2.0,* utilizando o *BERT* na componente de pré-treino.

2.4 Conclusão

O ensino digital é uma realidade no quotidiano de alunos e professores, com tendência para continuar a aumentar nos próximos anos. Este crescimento, trás também um crescimento no número de conteúdos disponíveis e a necessidade de criar sistemas que rapidamente respondam com a informação que os alunos procuram.

Neste capítulo de estado da arte, foi possível verificar que é hoje uma realidade a utilização da IA nos sistemas de LMS, assim como é uma realidade a sua utilização de forma cada vez mais intensiva nos motores de pesquisa, nomeadamente nos de domínio aberto, como é o caso dos motores de pesquisa de páginas na internet.

A inteligência da pesquisa é demonstrada pela capacidade de o sistema entender a intenção do utilizador e, consecutivamente, apresentar ao utilizador a resposta adequada como primeiro resultado. Num sistema de domínio de informação fechado como é o caso da Escola Virtual, as respostas que o utilizador procura para os conteúdos educativos estão presentes no texto dos manuais e nas bases de dados de recursos educativos.

A IA aplicada ao processamento de língua natural, tem apresentado resultados com inúmeras aplicações de sucesso, nomeadamente nos motores de pesquisa e sistemas de conversação. Esta aplicabilidade tem também levado a um crescimento significativo nos últimos 6 anos de estudos na área de investigação em *Question-Answer*, com resultados e aplicações muito relevantes, nomeadamente com a utilização dos modelos de redes neuronais baseados em *Transformers*.

3. EVGuru

EVGuru é um assistente de pesquisa virtual construído com o objetivo de apoiar os alunos (utilizadores da plataforma Escola Virtual) na pesquisa inteligente de conteúdos digitais no âmbito de uma disciplina, sendo capaz de entender algumas intenções de pesquisa do utilizador, apresentando respostas diretas às questões colocadas ou apresentando a lista de recursos digitais que melhor poderão responder aos critérios de pesquisa do utilizador.

Embora todo o projeto tenha sido construído com o objetivo de integração na plataforma Escola Virtual beneficiando dos seus conteúdos digitais, os princípios e técnicas utilizadas são válidos para qualquer outro contexto onde existam textos e conteúdos digitais que beneficiem de uma pesquisa inteligente.

Este capítulo apresenta as várias componentes que constituem a solução apresentada, iniciando na secção 3.1 com uma visão geral sobre a arquitetura do sistema e descrição geral das responsabilidades dos mesmos, seguindo com as secções 3.2 e 3.3 com uma apresentação detalhada sobre as abordagens e técnicas utilizadas em cada componente. Existem duas camadas aplicacionais de componentes que atuam de forma independente. Na secção 3.2 é apresentado o primeiro grupo e componentes responsáveis pelo extração, seleção e tratamento dos textos dos livros e recursos objeto das pesquisas, seguindo-se a 3.3 com a apresentação dos componentes que caracterizam o comportamento do assistente de pesquisa virtual, tanto na vertente de *User Interface* como no comportamento inteligente da pesquisa. Os resultados das experiências efetuadas serão apresentados no capítulo 4.

3.1 Arquitetura geral do sistema

Como referido anteriormente, o projeto EVGuru pretende ser disponibilizado na plataforma Escola Virtual com o objetivo de apoiar o aluno na pesquisa inteligente de conteúdos onde este poderá encontrar respostas ou informação complementar para as suas dúvidas ou, sempre que possível, obter uma resposta direta à questão colocada. É na capacidade do EVGuru realizar esta última tarefa que o poderemos enquadrar nos denominados *Question-Answering Systems* (QA).

Um sistema de QA é composto tipicamente por três módulos distintos (Allam and Haggag 2012): "*Query Processing Module*" que trata da classificação do tipo de questão e poderá efetuar a reformulação da questão, "*Document Processing Module*" cuja responsabilidade é execução da tarefa de *information retrieval* obtendo da base de textos as passagens mais relevantes para a questão colocada e "*Answer Processing Module*" que trata por fim de extrair a resposta.

A arquitetura deste sistema de pesquisa inteligente, combina as componentes habituais de uma arquitetura de um sistema de QA (*Question-Answering System*), com componentes de pesquisa tradicional e a geração de questões QG (*Question Generation Systems*).

O diagrama da Figura 20 apresenta uma visão geral sobre os principais componentes que constituem as duas camadas aplicacionais necessárias ao funcionamento do sistema: *Close domain text preparation* e *Virtual search assistant*.

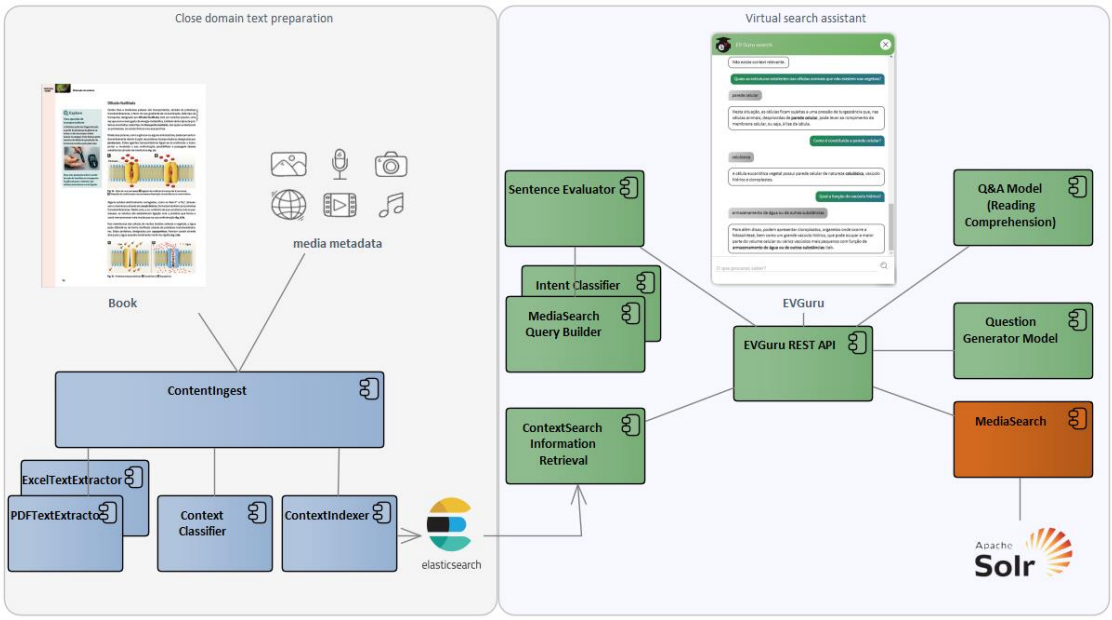

Figura 20 - Visão geral da arquitetura do sistema

A camada aplicacional "*Close domain text preparation*" assume a responsabilidade de, a partir de ficheiros de *input* com dados semiestruturados dos manuais escolares e recursos multimédia de uma dada disciplina/ano que pretendemos que sejam alvo de ação do assistente de pesquisa virtual, efetuar a extração, o pré-processamento e a seleção dos textos relevantes para a construção da base de textos que irão definir o domínio da pesquisa de respostas do assistente.

O conteúdo dos manuais escolares é disponibilizado em ficheiros PDF (*Adobe Portable Document Format*), enquanto que os textos contendo definições e conceitos abordados em recursos multimédia são disponibilizados em ficheiros no formato XLSX (*Microsoft Excel)*.

Nesta camada o componente "*ContentIngest"* recebe os diferentes ficheiros de *input* e coordena as ações de: extração dos textos PDF e XLSX delegando essa atividade no "*PDFTextExtractor"* e "*ExcelTextExtractor"*, classificação binária dos textos extraídos com sendo relevantes ou não relevantes através do componente "*ContextClassifier"* que disponibiliza um modelo de classificação previamente treinado na tarefa de *sentiment analysis* com *dataset* manualmente construído para o efeito, concluindo com a ação final de armazenamento e indexação numa base de dados de documentos *ElasticSearch*[8].

A camada aplicacional "*Virtual search assistant"*, assume a responsabilidade de expor uma *interface* gráfico para permitir ao utilizador colocar as questões ou pesquisas em língua natural e devolver as respostas ou resultados de pesquisa relevantes.

Nesta camada o *input* do utilizador é efetuado num *user interface web* com a apresentação habitual dos sistemas conversacionais e toda a interação com os diferentes componentes é efetuada através de uma REST API[9] implementada pelo componente "*EVGuru REST API*".

A cada frase inserida pelo utilizador (*user input sentence)* é efetuada uma chamada à REST API que utiliza o componente "*Sentence Evaluator"* o qual coordena a avaliação da frase, nomeadamente efetuando uma primeira classificação básica através do componente "*Intent Classifier*" para devolver se estamos perante uma pergunta ou não e, não sendo uma pergunta, através de técnicas de NLP efetuar uma análise da frase e construção de uma *query* que será devolvida pelo componente "*Search Query Builder"* para a posterior pesquisa de recursos no banco de recursos "*MediaSearch*" (componente atualmente já existente na plataforma da Escola Virtual).

O resultado devolvido pelo "*Sentence Evaluator"* irá determinar uma de três ações possíveis a executar:

- "*getAnswer*" - Esta classificação de ação é obtida quando a frase analisada é classificada como sendo uma questão. A ação será executada utilizando os componentes "*Information Retrieval*" cuja responsabilidade será obter os 10 textos com melhor classificação em termos de relevância para a questão colocada e o "*Q&A Model*" o qual utiliza o modelo baseado na arquitetura de *Transformers* BERT afinado para a tarefa de MRC que utiliza como contexto cada um dos 10 textos obtidos para obter o segmento de texto melhor classificado para devolver a resposta ao utilizador.
- "*search*" – Esta classificação de ação é obtida quando a fase analisada é classificada como sendo a intenção do utilizador obter uma pesquisa de determinados recursos (tipo de recurso, disciplina, ano, tema, tópico). Neste cenário o componente "*Search Query*

[8] https://www.elastic.co/
[9] https://www.redhat.com/en/topics/api/what-is-a-rest-api

Builder" transforma a consulta colocada pelo utilizador em língua natural, numa *query* a ser executada pelo "*MediaSearch"*.

- "getQuestions" – Esta classificação é obtida quando o sistema não obtém com sucesso nenhuma das duas anteriores. Esta ação é executada pelo componente "*Question Generator Model*" cuja responsabilidade consiste na geração automática de questões a partir dos 10 textos mais relevantes obtidos pelo "*Information Retrieval*". São geradas 5 questões apresentadas ao utilizador para que este possa escolher o que pretende ver respondido. As questões são geradas utilizando um modelo treinado utilizando o modelo pré-treinado *Transformers* T5 (Raffel, et al. 2020) para a tarefa de geração de questões.

3.2 Close domain text preparation

Como referido na secção 3.1 nos módulos que compõem o EVGuru fazem parte os módulos habituais de um sistema de QA.

Maioritariamente, os sistemas de QA podem ser classificados em dois tipos (Ajitkumar, Khillare and C. 2016): *Open domain* e *Closed domain. Open domain* pretendem responder a qualquer tipo de pergunta sobre uma variedade de domínios, nomeadamente sobre conteúdos encontrados na web, como é possível verificar atualmente nos resultados de uma pesquisa no *Google*, e os *Close domain* onde o espaço de pesquisa para a resposta é limitado a um domínio de informação como por exemplo medicina.

No cenário de utilização que se pretender dar ao EVGuru, as questões colocadas pelo utilizador serão efetuadas no âmbito de uma determinada disciplina (Biologia, História, Geografia, entre outras) e neste sentido, poderemos enquadrá-lo como um *Close domain QA System.*

Os textos a considerar para a base de pesquisa, são os textos presentes nos manuais escolares e os textos de catalogação dos recursos digitais disponíveis na plataforma (vídeos, áudios, SCORM[10], entre outros).

Os textos associados aos recursos são pré-produzidos por peritos da área disciplinar e disponibilizados em formato Excel estabelecendo a relação com o ficheiro do recurso, enquanto que os textos dos manuais são disponibilizados no formato PDF.

O tratamento dos ficheiros Excel e respetiva submissão desses textos para a base de textos é uma atividade trivial implementada pelo componente "*ExcelTextExtractor"* e descrita na secção 3.2.1 .

Tendo em conta a estrutura do formato PDF e a forma como os manuais escolares conjugam texto, imagens, tabelas e outros elementos na sua construção, apresenta-se como um desafio complexo na tarefa de extração e seleção dos textos para compor a base de textos que será utilizada nas pesquisas. Esta complexidade implicou o estudo e comparação de resultados com

[10] https://scorm.com/

diferentes bibliotecas preparadas para trabalhar com ficheiros PDF, bem como a construção de algoritmos de limpeza e classificação dos textos de modo a obter as passagens relevantes para a pesquisa e extração das respostas. Na Figura 21 é possível visualizar os textos que uma biblioteca habitual para trabalhar com ficheiros PDF identifica, assinalando a verde os textos que deverão ser considerados para a base de dados de documentos de pesquisa e a azul os que terão de ser descartados.

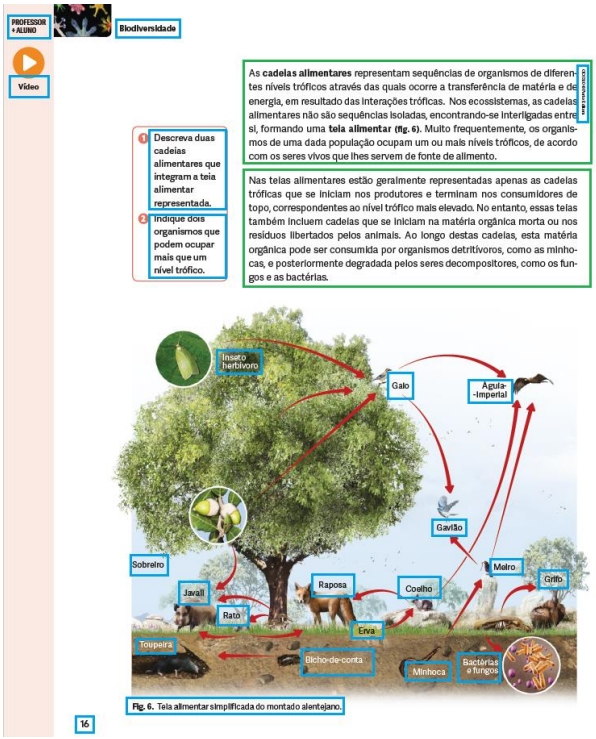

Figura 21 - Página de manual escolar PDF e extração do texto[11]

Uma vez identificados os textos relevantes para a constituição dos contextos de pesquisa das respostas, estes são submetidos para a base de dados de documentos. Estas atividades são implementadas pelos componentes "*PDFTextExtractor*", "*Context Classifier*" e *"Context Indexer"* descritos em detalhe nas secções 3.2.2 , 3.2.3 e 3.2.4 respetivamente.

3.2.1 ExcelTextExtractor

Dada a forma estruturada destes ficheiros *Excel* com a meta informação dos recursos armazenados na base de dados de recursos, este componente serve apenas para ler o conteúdo dos ficheiros *Excel* e submeter os textos obtidos para armazenamento pelo componente *ContentIndexer* descrito em 3.2.4 .

[11] Manual Odisseia Biologia e Geologia 10º da Porto Editora.

3.2.2 PDFTextExtractor

No processo editorial de produção de um manual escolar, depois de realizado todo o trabalho de edição e compilação dos diversos materiais que compõem o manual e efetuada a revisão por autores e editores, o resultado final termina na produção de um ficheiro PDF, o qual é utilizado também para a impressão do manual em papel.

De modo a tirar partido deste conteúdo já em formato digital e evitar a necessidade de trabalho manual adicional por parte dos gestores de conteúdos, os textos do manual são obtidos diretamente a partir dos ficheiros PDF dos manuais relacionados com as disciplinas para as quais se pretende criar a base de dados de documentos para o módulo de QA.

Para a extração dos parágrafos de texto foi necessário analisar e escolher a biblioteca que cumprisse com os requisitos necessários e qualidade na extração destes textos, tendo sido analisadas três bibliotecas *python*: pdfminer[12], PyPDF4[13], pdfplumber[14]. Para além destas, foram também realizadas experiências com abordagem de OCR (*Optical Character Recognition*) utilizando para o efeito a biblioteca *Tesseract*[15].

Estas bibliotecas foram analisadas considerando os seguintes critérios:

1. Palavras e frases completas – O texto obtido deverá ser igual ao texto visualizado na leitura do PDF.
2. Parágrafos completos – O texto obtido em parágrafos distintos de modo isolar tanto quanto possível os conceitos.
3. Respeitar a ordem de leitura – O texto deverá ser obtido pela ordem natural de leitura (da esquerda para a direita, de cima para baixo).

Para a análise destas bibliotecas foi utilizado como exemplo o PDF de um manual escolar de biologia 10 ano de escolaridade.

pdfminer

Esta biblioteca é bastante focada na extração de texto a partir de PDF e permite obter alguma informação sobre o texto extraído como por exemplo a localização ou tipo de fonte.

Nas experiências realizadas, foram encontrados dois problemas principais: Frases quebradas antes do fim efetivo da frase e sequências de textos repetidas.

A Figura 22 apresenta um pequeno fragmento com texto do manual escolar, seguido do exemplo de texto obtidos pela biblioteca, evidenciando em diferentes cores, exemplos de duplicações, assim como o caracter de quebra de linha "\n" sempre presente.

[12] https://pypi.org/project/pdfminer/
[13] https://github.com/claird/PyPDF4
[14] https://github.com/jsvine/pdfplumber
[15] https://github.com/tesseract-ocr/tesseract

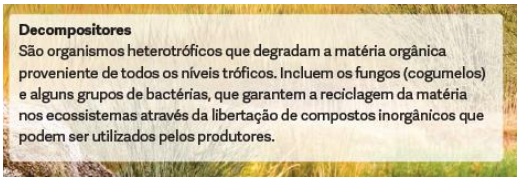

Figura 22 - fragmento de página do manual

Exemplo:

```
Decompositores\n

São organismos heterotróficos que degradam a matéria orgânica \n
proveniente de todos os níveis tróficos. Incluem os fungos (cogu-\n
Decompositores\n
melos) e alguns grupos de bactérias, que garantem a reciclagem \n
São organismos heterotróficos que degradam a matéria orgânica \n
da matéria nos ecossistemas através da libertação de compostos \n
proveniente de todos os níveis tróficos. Incluem os fungos (cogumelos) \n
inorgânicos que podem ser utilizados pelos produtores.\n
e alguns grupos de bactérias, que garantem a reciclagem da matéria \n
nos ecossistemas através da libertação de compostos inorgânicos que \n
podem ser utilizados pelos produtores.\n
```

PyPDF4

As experiências realizadas com PyPDF4 relevaram um problema grave com o PDF. Embora para a grande maioria das páginas os resultados fossem interessantes, em determinadas páginas, provavelmente pelas características de construção do PDF[16], alguns textos não eram de todo extraídos.

No exemplo seguinte apresentamos o texto obtido para o mesmo fragmento da figura Figura 22.

Neste exemplo o resultado obtido com o PyPDF4 foi melhor que o pdfminer pelo facto de evitar as duplicações de texto, no entanto a utilização desta biblioteca foi descartada já que falhava na extração de texto em algumas páginas.

Exemplo:

```
Decompositores\n
São organismos heterotróficos que degradam a matéria orgânica \n
proveniente de todos os níveis tróficos. Incluem os fungos (cogumelos) \n
e alguns grupos de bactérias, que garantem a reciclagem da matéria \n
nos ecossistemas através da libertação de compostos inorgânicos que \n
podem ser utilizados pelos produtores.\n
```

[16] A análise da estrutura de construção dos PDF não foi objeto deste projeto

Tessaract

Uma vez identificados os problemas das soluções anteriores, partiu-se para a experimentação com uma abordagem diferente, por *OCR*.

O objetivo na utilização desta abordagem, parte da perspetiva na obtenção de melhores resultados utilizando uma tecnologia que retirasse a componente da estrutura complexa do PDF e se baseasse apenas na imagem (observação) do texto de cada página.

O motor da biblioteca *Tessaract* 4 utiliza uma rede neuronal LSTM ao contrário do seu antecessor que trabalha utilizando reconhecimento de padrões de caracteres (Tesseract Git n.d.).

Esta abordagem implica uma conversão prévia das páginas do manual para imagens distintas, para de seguida ser aplicado nas imagens pretendidas o OCR para a extração de texto. Este processo de conversão prévia em imagem é um dos pontos fracos desta solução, pelo tempo de processamento demasiado elevado para a realização desta tarefa, demorando num computador com processador i5 e 16Gb de RAM, cerca de 4 minutos para gerar as 207 imagens das páginas do manual. No entanto esta tarefa apenas tem de ser executada uma vez por cada manual.

Do ponto de vista da extração do texto, a solução resolve a questão das duplicações obtidas com a biblioteca *pdfminer* e não apresenta falhas de texto como as que se verificam com a PyPDF4. No entanto, foram verificadas situações onde, devido à construção gráfica da página, algumas palavras são extraídas com erros ortográficos e noutras situações são extraídos caracteres que não fazem parte dos parágrafos. Na Figura 23, é apresentado um fragmento que pelo facto de incluir um pequeno texto na vertical (assinalado a azul), cria ruído na extração do texto.

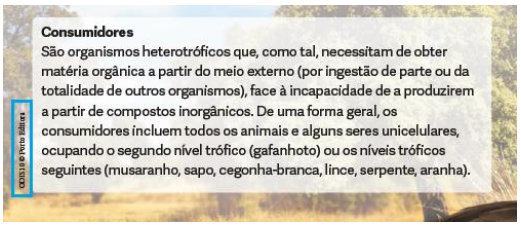

Figura 23 - fragmento de manual com texto vertical

Exemplo:

```
Consumidores\n
São organismos heterotróficos que, como tal, necessitam de obter\n
matéria orgânica a partir do meio externo (por ingestão de parte ou da\n
totalidade de outros organismos), face à incapacidade de a produzirem\n
a partir de compostos inorgânicos. De uma forma geral, os\n
consumidores incluem todos os animais e alguns seres unicelulares,\n
ocupando o segundo nível trófico (gafanhoto) ou os níveis tróficos\n
seguintes (musaranho, sapo, cegonha-branca, lince, serpente, aranha).\n
. E ET\n
voar isa ei '\n
\n
+ q\n
```

pdfplumber

Esta biblioteca apresenta uma grande variedade de métodos, que vão desde a extração de texto à semelhança das bibliotecas apresentadas anteriormente, até métodos que permitem "navegar" por cada elemento da estrutura do PDF, até à extração de palavra a palavra ou caracter a caracter, com toda a informação associada a estes elementos (posição, tamanho, orientação, …)

Nas primeiras experiências realizadas, a abordagem foi seguir a utilização dos métodos disponibilizados pela biblioteca para a extração completa do texto de uma página, utilizando o método *extract_text()* disponível na class *pdfplumber.Page*. À semelhança das analisadas anteriormente, existiam páginas onde os resultados correspondiam aos requisitos, no entanto em determinadas páginas os resultados foram desastrosos. O exemplo seguinte apresenta um resultado de extração de texto para o fragmento da Figura 22 onde é possível observar bastantes erros na extração obtida.

Exemplo:

```
Decompositores\n
São organismos heterotróficos que degradam a matéria orgânica \n
proveniente de todos os níveis tróficos. Incluem os fungos (cogu-\n
Decompositores\n
melos) e alguns grupos de bactérias, que garantem a reciclagem \n
dSaã om oartgéarinai snmoso se choestesirsottermófaicso ast qrauveé ds edgar
alidbaemrta aç ãmoa dtéer ciao omrpgâonsticoas\n
ipnroorvgeânniiecnotse q duee t pooddoes mos s neírv uetisil itzraódfiocso\n
ps.e Ilnocsl uperomd uotso fruensg.os (cogumelos)\n
e alguns grupos de bactérias, que garantem a reciclagem da matéria\n
nos ecossistemas através da libertação de compostos inorgânicos que\n
podem ser utilizados pelos produtores.\n
```

Perante estes resultados e dado que esta biblioteca permite bastante flexibilidade e granularidade quanto aos elementos do PDF que podem ser trabalhados, foi efetuada uma análise mais detalhada aos métodos disponíveis na biblioteca, com o objetivo de identificar abordagens diferentes para a resolução do problema.

Nesta análise aos métodos disponíveis, foi identificado o método ***extract_words*** da classe ***Page***. Este método utiliza a propriedade ***chars*** da classe ***Page***, propriedade esta que devolve uma lista de todos os caracteres encontrados na página, contendo um dicionário com toda a informação associado ao caracter, como apresentado no exemplo seguinte.

Exemplo:

```
{
    "fontname": "KFEYLX+AdelleSans-Bold",
    "adv": 0.621,
    "upright": True,
    "x0": 47.1213,
    "y0": 721.7451000000001,
    "x1": 52.7103,
    "y1": 730.7451000000001,
```

```
    "width": 5.588999999999999,
    "height": 9.0,
    "size": 9.0,
    "object_type": "char",
    "page_number": 15,
    "stroking_color": None,
    "non_stroking_color": [0.13736, 0.13636, 0.13164],
    "text": "P",
    "top": 77.12889999999993,
    "bottom": 86.12889999999993,
    "doctop": 11387.364899999999
}
```


Na informação associada a cada caracter, os elementos `"x0"`,`"y0"` e `"x1"`,`"y1"`, representam as coordenadas no plano cartesiano, que permitem definir a caixa delimitadora do espaço ocupado pelo caracter na página, conforme ilustrado na Figura 24.

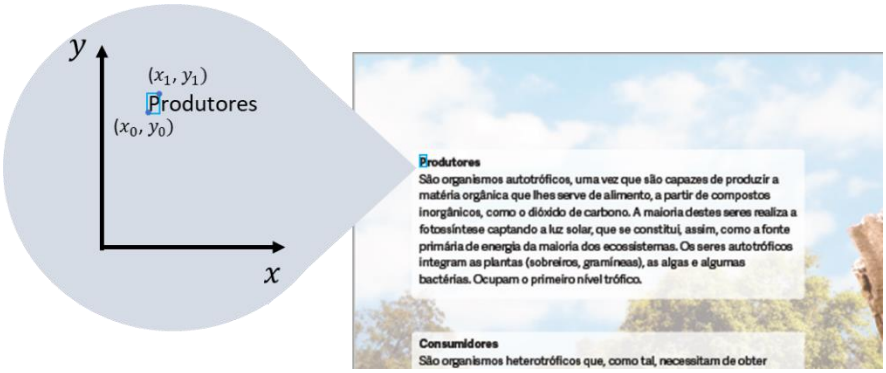

Figura 24 – coordenadas da caixa delimitadora de cada caracter devolvidas pelo pdfplumber

O método *extract_words* devolve para uma determinada página, uma lista de todas as palavras encontradas. As palavras são tratadas como sequências de caracteres em que a distância entre $x_1$de um caracter e o x_0 do caracter seguinte é menor que a tolerância definida como parâmetro de entrada do método. Para cada palavra, à semelhança do que acontece para os caracteres, a biblioteca devolve as coordenadas da caixa delimitadora da palavra.

As experiências realizadas com este método de extração das palavras de uma página, permitiram concluir que a biblioteca era bastante eficaz a este nível, obtendo com bastante precisão todas as palavras existentes na página. Outro aspeto interessante do método, é o facto de permitir à priori excluir palavras que tenham uma determinada orientação (horizontal ou vertical), o que é bastante útil para excluir palavras escritas na vertical.

No entanto, a utilização deste método por si só não permite responder aos requisitos definidos, já que todo o texto da página é obtido como uma lista de palavras, não permitindo identificar a forma como essas palavras estão agrupadas na página, nomeadamente nos diferentes parágrafos.

Analisando visualmente a estrutura e organização do texto nas páginas quanto à construção dos parágrafos ou grupos de parágrafos no manual, considerando que cada palavra não é mais do que um ponto (x, y) num plano cartesiano e que os parágrafos podem ser vistos como *clusters*

de palavras que partilham uma distância máxima com a palavra mais próxima, foi desenvolvido um algoritmo simplificado baseado no algoritmo de aprendizagem não supervisionada, DBSCAN (Ester, Kriegel and Xu 1996).

Utilizando a definição da *Wikipédia* (Wikipédia n.d.), as técnicas de *clustering* permitem efetuar a análise automática de dados e extrair dessa análise informação sobre a forma como esses dados podem ser agrupados, formando grupos que partilham determinados critérios de semelhança. A estes grupos é dada a designação de *clusters*.

O algoritmo DBSCAN é um algoritmo de complexidade relativamente simples quando comparado com outros algoritmos de *clustering* e consiste na identificação de grupos em que os elementos partilham uma distância máxima e em que o grupo é definido por uma dada densidade. A distância máxima e a densidade são parâmetros de entrada do algoritmo. O algoritmo inicia um *cluster* pela seleção de um elemento selecionado aleatoriamente e para esse elemento identifica, baseado na distância e densidade) os elementos mais próximos. Estes elementos mais próximos são classificados como elementos *core* ou não, dependendo da densidade da vizinhança desse elemento. O critério de paragem na construção de um *cluster* acontece quando não existem mais elementos *core* para selecionar. No final da construção do *cluster* são adicionados os elementos cuja densidade não cumpre o critério, mas cumpre o critério da distância. Este processo repete-se para todos os dados disponíveis formando todos os *clusters*.

Utilizando esta abordagem, foram realizadas experiências utilizando a biblioteca *Scikit-Learn*[17] com a classe *cluster.DBSCAN*, no entanto os resultados não foram os desejados. Na Figura 25 são apresentados os atributos devolvidos pela biblioteca *pdfplumber* e que definem a caixa delimitadora do espaço ocupado por uma palavra.

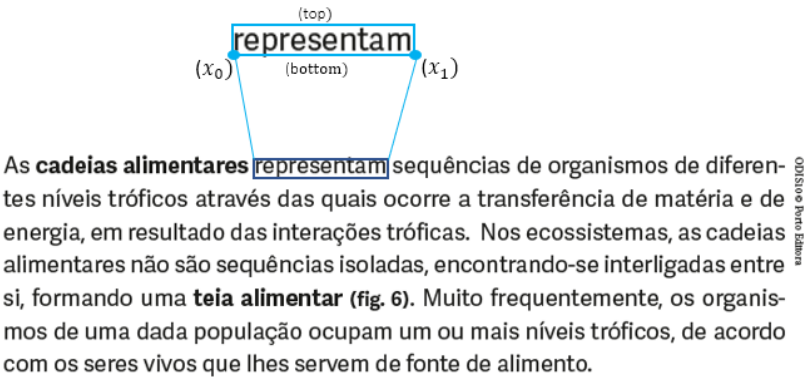

Figura 25 - coordenadas da caixa delimitadora da palavra devolvido pelo *pdfplumber*

Na utilização da classe *cluster.DBSCAN* foram utilizados como *dataset* os quatro valores: $x_0, x_1, top, bottom$ para cada palavra da página.

Foram realizadas experiências mantendo um número mínimo no *cluster* de 3 (palavras) e com uma distância de 20, 30, 40 e 50, no entanto os resultados nunca foram satisfatórios.

[17] https://scikit-learn.org/

Aumentando a distância resolvia o problema de separação em alguns *clusters* não desejados, no entanto passava a considerar alguns parágrafos distintos como sendo um só. Também no caso de parágrafos com baixa densidade de palavras, em algumas situações dependendo da dimensão das palavras, o algoritmo não conseguia classificar todas as palavras no mesmo *cluster* (Figura 26.

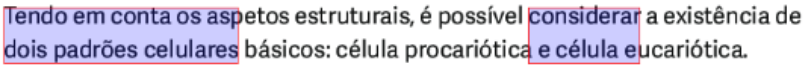

Figura 26 – Palavras por classificar em *clusters* com DBSCAN

Outro problema identificado na construção destes *clusters* de palavras com o algoritmo DBSCAN, é o facto de não ser adequado que o cálculo da distância entre cada palavra seja efetuado a partir do ponto médio de cada palavra. Ou seja, em palavras maiores a distância calculada pelo ponto médio poderia ser rapidamente ultrapassada e ajustando o valor da distância, passaria esta a ser demasiado grande para palavras com um número reduzido de caracteres.

Aproveitando os conceitos base destes algoritmos baseados nas distâncias entre os elementos como o já referido DBSCAN ou algoritmos de *clustering* hierárquicos como *single-linkage clustering*, *complete linkage clustering* ou *average linkage clustering* (Murtagh and Contreras 2012), foi desenvolvido o algoritmo apresentado em **Algoritmo 1**, o qual recebe como parâmetro de entrada apenas a distância máxima pré-definida e, enquanto não encontrar um *cluster* para a palavra, realiza três passagens de comparação das distâncias entre as palavras, considerando em cada iteração a comparação entre a palavra sem classificação w_t e as palavras já classificadas $wc = (w_1, w_2, \dots, w_{t-1})$ calculando a distância em três pontos diferentes da caixa delimitadora, nas abcissas : x_0 , x_1 e ponto médio das abcissas $\frac{(x_0+x_1)}{2}$.

Algoritmo 1 *Clustering* distância ao vizinho mais próximo com distância pré-definida

Entrada: Conjunto de todas as palavras na página: $D = (w_1, w_2, \dots, w_k)$
Distância máxima: d_{max}
Saída: Lista de todos os grupos de palavras representando os parágrafos: $[C_1, C_2, \dots, C_n]$
conj_paragrafos_class ← *{ }* # inicia com dicionário de parágrafos classificados vazio
cluster ← *0*
ultimo_cluster ← *0*
para cada $w_t \in D$*:*
  distancia_min_todos_grupos ← *999* # inicia com valor irreal o valor mínimo obtido de w
  # em relação a todos os grupos. Útil para a criação
  # de novos *clusters*
  n_*passagens* ← *0*
enquanto *distancia_min_todos_grupo* > d_{max} ***e*** *n_passagens < 3:*
  para cada *nr_paragrafo,* wc ∈ *conj_paragrafos_class:*
    distancia_min_grupo ← *999* # inicia com valor irreal o valor mínimo entre w e as
    # palavras do conjunto wc já classificado
    para cada $wc_k \in wc$*:* # as coordenadas de cada palavra são: $x_0, x_1, bottom, top$
      $xwc_k \leftarrow wc_k\{x_1\}$,***se*** $wc_k\{x_1\} < w_t\{x_0\}$ ***ou*** $wc_k\{bottom\} = w_t\{bottom\}$ ***se não*** $xwc_k \leftarrow wc_k\{x_0\}$
      $xw_t \leftarrow xw_t\{x_0\}$, ***se*** *n_passagens = 0*
      $xw_t \leftarrow xw_t\{x_1\}$, ***se*** *n_passagens = 1*

17: $xw_t \leftarrow \frac{xw_t\{x_0\} + xw_t\{x_1\}}{2}$ **se** *n_passagens = 2*
18: $d = \sqrt{(xw_t - xwc_k)^2 + (w_t\{bottom\} - wc_k\{bottom\})^2}$
19: *distancia_min_grupo* ← d, **se** d < *distancia_min_grupo*
20: **se** *distancia_min_grupo < distancia_min_todos_grupos:*
21: *distancia_min_todos_grupos* ← *distancia_min_grupo*
22: *cluster* ← *nr_paragrafo* *# foi possível enquadrar a palavra num dos clusters existentes*
23: *n_passagens* ← *n_passagens + 1*
24: **se** *distancia_min_todos_grupos* > d_{max}*:*
25: *ultimo_cluster* ← *ultimo_cluster + 1* *# novo cluster*
26: *cluster* ← *ultimo_cluster*
27: *conj_paragrafos_class { cluster }* ← ***adiciona*** w_t

Com o algoritmo apresentado em Algoritmo 1, foi possível obter a extração automática dos parágrafos de uma forma semelhante à que seria visualmente efetuada por um humano conforme apresentado no capítulo de resultados em 4.3.1 .

3.2.3 Context Classifier

Analisando o conjunto de segmentos de texto extraídos do PDF, nem todos estes segmentos são relevantes do ponto de vista da informação e conhecimento de domínio, para que possam ser considerados para a base de dados de textos e assim fazer parte dos contextos de pesquisa para a componente de QA.

O manual escolar, dependendo do manual e da abordagem dos autores, apresenta as páginas organizadas em diferentes secções, nomeadamente em: Índice com a organização do manual, introdução ao tópico, conteúdos de aprendizagem, atividades práticas, resumos, exercícios. Em todas as secções existem imagens, tabelas e outros objetos, com textos associados.

Como já referido, um dos objetivos neste projeto é que a base de textos seja criada utilizando os conteúdos digitais com o mínimo de intervenção humana possível. Por esta razão, o sistema deve ser capaz de selecionar de forma automática, de entre os textos extraídos, quais os textos que devem ser considerados. Para este efeito, foram utilizadas técnicas de *machine learning* clássicas e de *deep learning* para a classificação dos segmentos de textos (parágrafos) como sendo texto relevante ou não relevante. O objetivo desta tarefa consiste na criação de um modelo viável para a classificação automática das sequências de textos que devem candidatas para a base de textos que irá servir de repositório para a tarefa de *information retrieval.*

Independentemente do modelo a utilizar, os resultados obtidos em *machine learning* dependem de uma série de atividades prévias ao treino do modelo, nomeadamente toda a fase de preparação dos dados. A Figura 27 apresenta as várias fases necessárias até à construção de um modelo final.

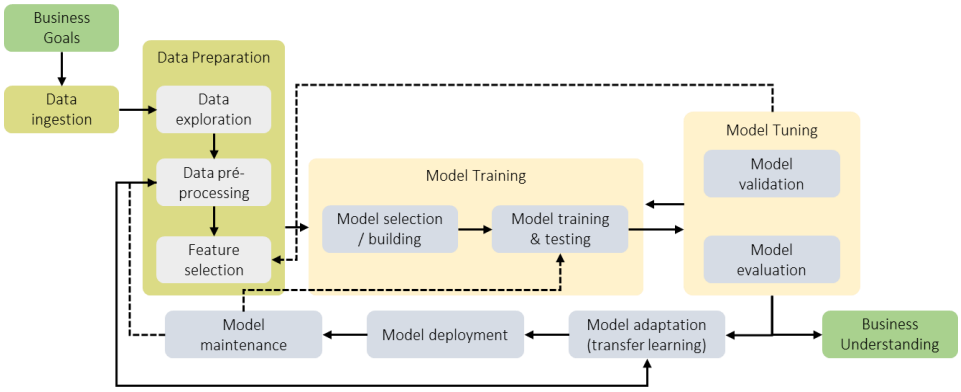

Figura 27 - Ciclo de vida de um processo de *machine learning*[18]

Nas subsecções seguintes são apresentadas as fases utilizadas no âmbito deste projeto para a construção do modelo de classificação da componente *Context Classifier*, iniciando pela fase de preparação dos dados onde estão incluídas as atividades de pré-processamento (limpeza dos dados, *tokenization, stemming, lemmatization*), até apresentação dos modelos de classificação utilizados.

Pré-processamento – Limpeza dos dados

Nesta fase são retirados ou substituídos no texto, caracteres ou sequências de caracteres extraídos pela biblioteca e que numa análise breve podem ser criadas regras, utilizando por exemplo expressões regulares, para limpar ou corrigir o texto. Para além destas regras de remoção ou substituição, podem ser também mantidas em ficheiro à parte, um conjunto de sequências de palavras que se pretende ignorar. Por exemplo, no manual em análise, surge em todas as páginas do manual a sequência "PROFESSOR + ALUNO" a qual se pretende que seja ignorada. Na Tabela 1 - Exemplos de pré-processamento do textoTabela 1 são apresentados alguns exemplos do pré-processamento efetuado.

Tabela 1 - Exemplos de pré-processamento do texto

Expressão regular	Resultado esperado
^(Fig.+[0-9]+|Fig.+ [0-9]+).+	Remover todas as legendas de figuras.
(.[a-zA-Zóéáãí]+)- (.[a-zA-Zóéáãí])	Corrigir as palavras que foram extraídas quebradas conforme apresentado na Figura 28. As **células procarióticas** constituem os organismos procariontes, de que são exemplo as bactérias. São, geralmente, as células mais pequenas e com uma organização mais simples, caracterizada pela ausência de núcleo e de organe-los membranares. Nestas células, o DNA encontra-se disperso no citoplasma, Figura 28 - Exemplo de palavra quebrada
(\(fig\. [0-9]+[a-zA-Z]*\)) \((fig\. [0-9]+|Fig\. [0-9]+)\)	Remover referências a figuras no meio dos textos.
(https?://.* |https?://(.*/.*)+(|\.\!\?)|https?://(.*/.*) +$)	Remover hiperligações para páginas na internet.

[18] Adaptado de conteúdo do mestrado do MEIA

Assumiu-se também que todas as sequências de texto teriam de terminar com os sinais de pontuação “.”,”?”,”!” ou “:”, eliminando assim algumas sequências de caracteres indesejáveis que surgiam no final de algumas sequências.

Pré-processamento – Tokenization

Em NLP, o processo de *tokenization* traduz-se fundamentalmente na separação de uma sequência de *input* nos chamados *tokens.* Esta separação pode ocorrer recorrendo a diferentes critérios obtendo como resultado uma lista de diferentes *tokens*.

Existem inúmeras bibliotecas de *tokenization* (algumas específicas para determinadas línguas) as quais podem ser utilizadas dependendo dos resultados que se pretende. Na Figura 29, são apresentadas duas listas de *tokens* obtidos para a mesma frase, utilizando métodos diferentes da biblioteca *nltk*[19].

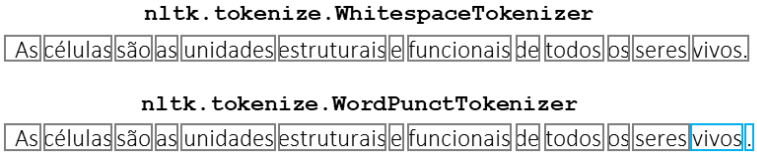

Figura 29 - Diferentes *tokens* obtidos para a mesma frase

Para o modelo de *logistic regression* a opção foi pela utilização do *WordPunctTokenizer* da biblioteca *nltk*, a qual separa as palavras e os sinais de pontuação em *tokens* diferentes. Uma vez obtida a lista de *tokens*, dado que o modelo apenas recebe de dados numéricos de entrada, existe a necessidade de criação de um dicionário de dados capaz de efetuar a transformação dos *tokens* num índice numérico.

No modelo contruído com RNN foi utilizada a classe *Tokenizer* do *Tensorflow*[20]. Esta classe pode receber uma lista de sequências de palavras e devolver uma lista de sequência de índices para um dicionário de palavras, removendo tudo o que são caracteres que não fazem parte das palavras.

Para o BERT foi utilizado o *Tokenizer* do modelo pré-treinado ‘neuralmind/bert-base-portuguese-cased’ (Souza, Nogueira and Lotufo 2020) carregado utilizando a classe *BertTokenizer* do *Huggingface Transformers*[21].

Pré-processamento – Stemming e Lemmatization

Na fase de pré-processamento de dados, o *stemming* consiste em remover sufixos às palavras com o objetivo de obter a forma original das mesmas, à qual em *NLP* se designa de *stemm*. Nas tarefas de NLP o *stemming,* a par da lematização, são também formas de reduzir a dimensionalidade já que permitem reduzir o número de tokens a considerar no vocabulário.

[19] https://www.nltk.org/
[20] https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/text/Tokenizer
[21] https://huggingface.co/

Tabela 2 - Exemplos de transformação *stemming*

Palavra	*stemm*
estruturais	*estrutur*
unidades	*unidad*
funcionais	*funcion*

A lematização, consiste na redução da palavra à sua forma canônica designado de lema. Este processo permite, não só, mas também, remover por exemplo o plural das palavras.

Tabela 3 - Exemplos de transformação *lemmatization*

Palavra	Lema
células	*célula*
funcionais	*funcionar*
seres	*ser*

Para o modelo *logistic regression*, dado que o objetivo foi criar uma rápida *baseline* com indicadores de performance para a classificação do *dataset*, optou-se apenas pela utilização de *stemming* já que permite uma maior redução de dimensionalidade. Para a realização da tarefa de *stemming* foi utilizada a classe *SnowballStemmer* da biblioteca *nltk*.

Já no caso dos modelos com redes neuronais, considerou-se que, pelo facto de estarem a ser utilizados *word embeddings* pré-treinados existiriam vantagens na utilização das palavras originais, não foi utilizada qualquer técnica de *steemming* ou de *lemmatization*.

Classificação dos segmentos de textos com Logistic Regression

Para permitir criar uma *baseline* de comparação entre a performance dos modelos de classificação em análise, iniciou-se pela utilização de um algoritmo de aprendizagem supervisionada clássico de *machine learning, logistic regression*.

Logistic regression é um dos métodos bastante utilizados por investigadores e estatísticos na análise de dados, podendo ser utilizado quer para problemas de classificação binária, ou estendido para multi-categoria, assim como para problemas de regressão, permitindo obter de forma natural as probabilidades de classificação (Maalouf 2011). Outro aspeto bastante importante é o facto de ser um modelo bastante rápido no treino e previsão, sendo um fator facilitador para a criação da *base line* pretendida nesta fase.

A hipótese do *logistic regression* (Nasteski 2017) é definida por:

$$h_\theta(x) = g(\theta^T x) \quad (10)$$

Onde a função g é definida pela seguinte função logística:

$$g(z) = \frac{1}{1 + e^{-z}} \quad (11)$$

Os valores da função variam entre [0;1], pelo que, ao contrário de modelos com outras funções de regressão que devolvem o valor discreto que pode ser superior a 1 ou inferior a 0, o *logistic regression* devolve a probabilidade de a classificação ser 0 quando a imagem da função é inferior a 0.5 ou 1 quando é superior a 0.5.

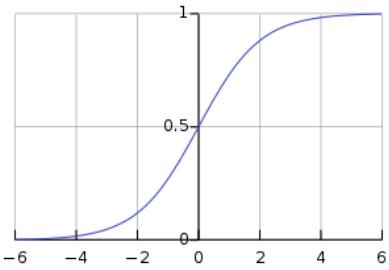

Figura 30 - *sigmoid* da função logística

O cálculo da otimização dos pesos θ é efetuado pela minimização do custo obtido pela função:

$$j(\theta) = \frac{1}{m}\sum_{i=1}^{m}[-y^{(i)}\log\left(h_\theta\left(x^{(i)}\right)\right) - (1 - y^{(i)})\log\left(1 - h_\theta(x^{(i)})\right)] \quad (12)$$

Dado que estes modelos apenas trabalham com informação numérica, uma vez realizado o pré-processamento, é necessário proceder à criação de um dicionário de índices de *tokens* o qual permita efetuar a conversão bidirecional entre o índice e o *token* respetivo. Para este efeito foi utilizada da biblioteca *Scikit-Learn* a classe *TfidfVectorizer* a qual produz uma matriz de vetores esparsos seguindo o modelo TF-IDF apresentado em 2.2 . Estas representações vetoriais, assim como as etiquetas binárias de objetivo da classificação, foram utilizadas para o treino do modelo *Logistic Regression* implementando um *pipeline* de treino com as classes *Pipeline* e *LogisticRegression* da biblioteca *Scikit-Learn*.

Classificação dos segmentos de textos com Deep Learning

Durante algumas décadas as técnicas computacionais para abordar problemas de NLP foram baseadas em algoritmos clássicos de *machine learning*, como por exemplo *Logistic Regression, Naive Bayes,* SVN entre outros, baseados em representações vetoriais esparsas como *Bag-of-Words* ou *TF-IDF,* nomeadamente em tarefas de classificação (Young, et al. 2018). Na última década a utilização de algoritmos e diferentes arquiteturas baseados em redes neuronais demonstraram superar os resultados obtidos pelos modelos clássicos em tarefas como *part-of-speech tagging (POS)*, *named entity recognition* (NER) *ou semantic role labeling* (SRL) (Collobert, et al. 2011). Com o acentuar da investigação em *deep learning aplicada ao* NLP (Goldberg 2017),

os últimos anos têm sido de enorme entusiasmo da comunidade de investigadores na utilização destas técnicas nas mais variadas tarefas, principalmente com o surgimento de modelos pré-treinados como o BERT, T5 ou GPT.

A utilização destes modelos tira partido do conceito de *transfer learning* em que um dado modelo treinado para uma tarefa específica, é posteriormente utilizado para a realização de uma nova tarefa relacionada, beneficiando da aprendizagem efetuada pelo modelo anterior (Torrey and Shavlik 2010). Estes modelos pré-treinados, apresentam enormes vantagens na sua utilização, dado que tipicamente são bastante complexos (milhões de parâmetros) e foram treinados recorrendo a grandes volumes de dados, exigindo elevado poder de computação. Em circunstâncias normais, estes recursos não estão ao dispor da grande maioria dos investigadores, pelo que tem permitido massificar a sua utilização e alcançar resultados de novas descobertas que de outra forma não seriam possíveis.

No desenvolvimento deste componente *Context Classifier* foram consideradas duas abordagens com *deep learning.* A primeira abordagem recorrendo a uma arquitetura de *Recurrent Neural Networks (RNN),* mais especificamente LSTM, utilizando para a camada de *embedding* o modelo pré-treinado *Facebook fastText* (Bojanowski, et al. 2017) (existem várias versões entre as quais português) e a uma segunda abordagem recorrendo a uma implementação com uma arquitetura de *Transformers* utilizando o modelo pré-treinado BERT (com modelo treinado em português 'neuralmind/bert-base-portuguese-cased').

Para o treino de ambos os modelos, foi construído um *dataset* contendo a cerca de 2500 frases extraídas dos manuais escolares de biologia e geologia 10º ano de escolaridade, com cada frase previamente classificada manualmente com a indicação "S" considerar e "N" para ignorar.

Tabela 4 - Exemplo de texto extraído do manual e correspondente classificação

#	Texto	Considerar
1	"As diferentes populações, e as relações que estabelecem entre si numa determinada área, formam uma comunidade. Uma comunidade encontra-se sempre em interação com os outros fatores do meio ambiente, constituindo, em conjunto com estes, um ecossistema."	S
2	"Os diferentes níveis de organização biológica podem ser estudados em qualquer ecossistema. As florestas constituem um laboratório natural para a identificação desses níveis. Que níveis de organização biológica podem ser identificados numa floresta?"	N

Pela observação dos exemplos na Tabela 4, é possível verificar a importância do contexto semântico e gramatical para a classificação atribuída. Uma das grandes vantagens dos modelos de *deep learning* quando comparados com os modelos de *machine learning* clássicos é também a capacidade de automaticamente serem extraídas características na análise dos dados que no caso dos modelos tradicionais têm, em muitos casos, de ser extraídas manualmente. Os

resultados dos modelos de *deep learning* aplicados a tarefas de NLP tendem a ser melhores, quanto melhores foram as representações vetoriais das palavras (*embeddings).*

Ao contrário de representações vetoriais como *bag-of*-words as quais resultam em vetores esparsos e que dependendo do vocabulário podem ser de grande dimensionalidade, os *embeddings* são vetores densos de dimensão fixa.

Os *word embeddings* tentam seguir o conceito da hipótese distribucional, onde palavras que ocorrem e são usadas no mesmo contexto tendem a ter significado similar (Harris 1954). Ou seja, estes vetores tendem a capturar informação dos seus vizinhos.

Dado que o objetivo inicial para o projeto EVGuru será trabalhar com conteúdos em português, para ambos os modelos, uma das preocupações foi encontrar modelos pré-treinados que permitissem obter *embeddings* tendo sido treinados com corpus de textos em português. No caso do modelo com *LSTM* foi utilizado *Facebook fastText* que disponibiliza *embeddings* em 152 línguas diferentes e no caso do BERT este modelo disponibiliza os seus próprios *embeddings*.

Classificação dos segmentos de textos RNN

Existem várias opções de arquiteturas de redes neuronais que poderiam ser utilizadas para tarefas de NLP, sendo, conforme apresentado no 2, as mais utilizadas as RNN e as CNN. Tendo em conta que o foco principal do projeto não é nos modelos de classificação, mas sim nas componentes mais relacionadas com as tarefas de *information retrieval* e QA descritos nas secções 3.2.4 e 3.3.5 respetivamente, a opção foi pela utilização da arquitetura que dada a sua capacidade natural para tratar dados sequênciais, aparentava ser adequada para o tratamento das sequências de palavras em textos, tendo a opção sido a utilização de uma rede BiLSTM.

As arquiteturas de rede LSTM permitem manter durante mais tempo a informação de memória de contexto num determinado momento temporal. As BiLSTM em particular, permitem obter esse contexto quer à esquerda quer à direita da palavra em análise. Esta informação de contexto é algo bastante importante para o problema de classificação em análise, já que, segundo os critérios utilizados para a etiquetagem manual do *dataset* a informação que determina se esta será ou não para considerar, em várias situações encontra-se nas zonas mais próximas do início ou final do texto.

Na Figura 31 são evidenciadas a azul as sequências de palavras que mais contribuiriam para a decisão humana na etiquetagem da frase.

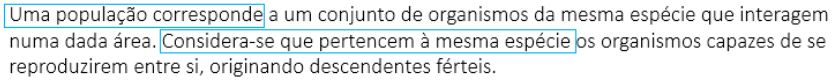

Figura 31 - Excerto de texto assinalando partes do texto mais relevantes para a classificação

De modo a capturar o contexto do texto nos dois sentidos, foi utilizada uma arquitetura de BiLSTM a qual é composta por duas camadas LSTM, uma que atua da esquerda para a direita na sequência e outra da direita para a esquerda (Figura 32).

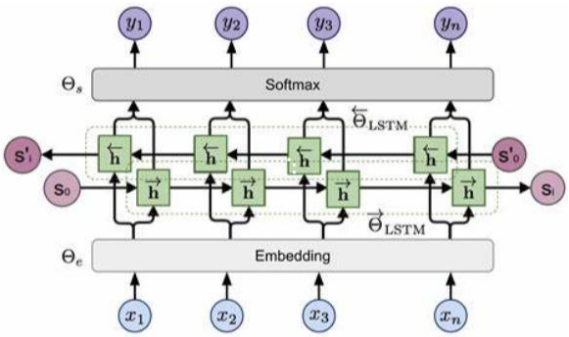

Figura 32 - Arquitetura de uma *BiLSTM*[22]

A arquitetura final do modelo de classificação implementado tendo por base RNN, ficou então constituído por uma camada de *embedding* previamente carregada com a matriz de *embeddings* com vetores de dimensão 300 criados a partir do *fastText*, uma camada BiLSTM com *512 cells* (tamanho máximo definido para as sequências de texto) que devolver apenas um vetor final e não os resultados em cada momento temporal, finalizando com uma camada densa *feed forward* ativada com a função de ativação *sigmoid* que realiza a classificação final. Dado que estamos perante uma classificação binária, a função de custo utilizada foi a *binary_crossentropy*, optimizador *Adam* (Kingma and Diederik P. 2017) e a métrica escolhida foi *accuracy* (métrica que calcula a relação entre o número de classificações corretas sobre o total de instâncias).

Na Figura 33 são apresentadas as camadas que compõe a arquitetura final do modelo implementado.

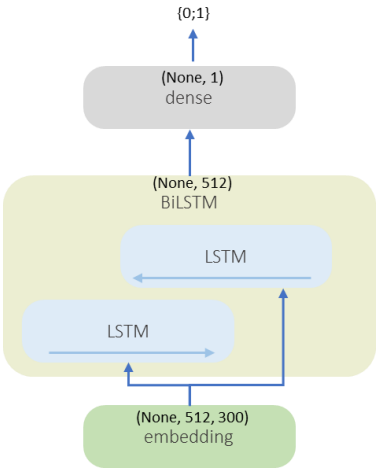

Figura 33 - Classificador LSTM dos textos extraídos

[22] https://www.topbots.com/generalized-language-models-cove-elmo/

Embora atualmente o *SOTA (State-of-the-Art)* das redes neuronais para as tarefas de *NLP* esteja nos trabalhos realizados com a arquitetura de *Transformers,* nomeadamente após o trabalho de (Vaswani, et al. 2017), as *RNN* dada a sua arquitetura adequada para trabalhar com sequências de textos, continua a ser uma solução interessante para tarefas relativamente simples como a de classificação de textos.

Classificação dos segmentos de textos - BERT

Tendo em conta o sucesso recente dos modelos baseados em *Transformers* e a disponibilidade de bibliotecas que permitem a utilização facilitada de modelos pré-treinados com enorme sucesso como o BERT descrito na secção "MRC baseado em *Transformers*", o objetivo na utilização deste modelo foi comprovar a efetiva facilidade na sua utilização, como funciona o processo de *fine-tuning* do modelo e verificar a eficiência dos resultados comparativamente com os resultados obtidos com o modelo testado com *fastText embeddings* + BiLSTM (os resultados desta experimentação serão apresentados no capítulo 4).

A arquitetura do BERT é constituída por várias camadas *encoders* da arquitetura de *Transformers*, existindo atualmente dois modelos de BERT disponíveis:

1. ***BERT base*** – consiste em 12 *layers* de *Transformers encoders,* 12 *attention heads, feedforword-networks 768 hidden units* e 110 milhões de parâmetros.
2. ***BERT large –*** consiste em 24 *layers* de *Transformers encoders, 16 attention heads, feedforword-networks 1024 hidden units* e 340 milhões de parâmetros.

O BERT, como acontece nas modelos de redes neuronais, espera que o *input* de palavras seja efetuado a partir do índice dos *tokens* previamente obtidos da sequência de palavras. Para a obter estes *tokens* e respetivos *embeddings* os autores utilizaram *WordPiece* (Wu, et al. 2016) com um vocabulário com 30.000 palavras. Existem dois *tokens* especiais utilizados pelo BERT, o *token* [CLS] e o *token* [SEP].

1. [CLS] é utilizado no início da série de sequências de *input* e no processo de aprendizagem o vetor de *embedding* gerado para este *token* é utilizado para tarefas de classificação. Pode ser interpretado como sendo o *embedding* da própria sequência.
2. [SEP] é utilizado para separar sequências de *input* importante por exemplo como veremos na secção 3.3.5 em tarefas de *QA*, onde existe a necessidade enviar no *input* a sequência de palavras relativo à questão e a sequência de palavras relativa ao contexto onde pretendemos encontrar o semento de texto da resposta.

Dependendo do modelo BERT utilizado (*base* ou *large*) à saída do modelo podemos obter *embeddings* de dimensão 768 ou 1024 para cada *token* da sequência. Para a construção do modelo de classificação pretendido, apenas necessitamos de colocar o *token* [CLS] no início da sequência e *token* [SEP] no final da sequência.

Com referido anteriormente, uma das bibliotecas que atualmente disponibiliza mais recursos para trabalhar com a arquitetura de *Transformers* e com os mais variados modelos pré-treinados em diversos tipos de tarefas é a biblioteca *Huggingface Transformers*.

Os modelos BERT pré-treinados necessitam que as sequências de *input* sejam fornecidas contendo três tensores com os dados de treino:

1. ***Input_ids*** – índices dos *tokens* de cada sequência de palavras dos textos do *dataset*. No caso do BERT, as sequências podem ter no máximo 512 *tokens*, pelo que, no caso de sequências inferiores, as posições à direita do último *token* são preenchidas com o valor 0 corresponde ao índice do *token* [PAD] numa operação designada de *padding*.
2. ***attention_mask –*** esta máscara binária identifica as posições onde estamos perante um *token* correspondente a uma palavra para considerar ou não. No caso dos tokens [CLS], [SEP] ou de uma palavra a considerar o valor deve ser 1, nos restantes deverá ser 0.
3. ***Token_type_ids –*** É uma máscara binária que identifica a que sequência o *token* pertence. Quando apenas existe uma sequência todos os valores são 0.

Esta biblioteca facilita bastante o desenvolvimento já que em apenas algumas linhas de código, é possível obter de uma só vez a partir de todo o texto do *dataset,* todas as estruturas de *input_ids, attention_mask* e *token_type_ids* que são necessárias. No Trecho de Código 1 é apresentado um exemplo de utilização da biblioteca.

```
      from transformers import BertTokenizer

      tokenizer = BertTokenizer.from_pretrained('neuralmind/bert-base-portuguese-
      cased')
      tokens = tokenizer(df['text'].tolist(), max_length=seq_len, truncation=True,
      padding='max_length', add_special_tokens=True,
                        return_tensors='np')
      print(tokens.keys())


output: dict_keys(['input_ids', 'token_type_ids', 'attention_mask'])
```

Trecho de Código 1 - utilização da classe BertTokenizer

Na Figura 34 é apresentada uma representação simplificada da arquitetura implementada do modelo de classificação com *BERT*. O modelo final obtido é um modelo de aprendizagem semi-supervisionada dado que embora seja necessário um *dataset* etiquetado para a classificação, incluí a camada do *BERT* o qual foi treinado sem recurso a dados previamente etiquetados.

O modelo é composto pela camada de entrada do modelo *BERT* utilizando apenas as estruturas *input_ids* e *attention_mask*. Neste cenário de classificação como apenas é passada uma sequência de texto, não existe a necessidade de passar a estrutura *token_type_ids.*

À saída do modelo BERT foram acrescentadas duas camadas, uma camada linear de FFN e uma camada para a saída da classificação de duas classes (0 e 1) utilizando a função *ReLU* a qual se

define pela função de ativação $\max(0, x)$ sendo x o valor de entrada de cada "neurónio". Como resultado vamos ter um valor calculado pela função de ativação para cada uma das classes. Com a função $argmax(y_0, y_1)$ no final, obtemos qual das duas classes tem o valor maior representando o resultado final da classificação. Ao conjunto das camadas adicionadas à camada BERT são designadas de BERT *Head*.

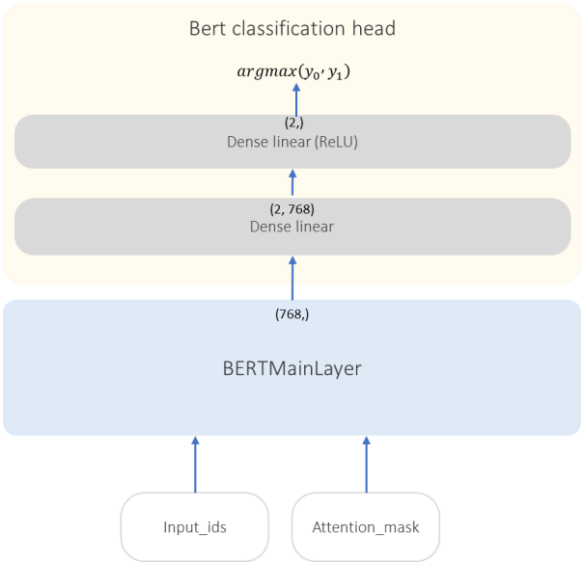

Figura 34 - Arquitetura simplificada do classificador com *BERT*

3.2.4 Context Indexer

Uma vez selecionados os textos considerados importantes para servirem de contexto para a pesquisa e obtenção de respostas, este componente tem a responsabilidade de efetuar o armazenamento desses documentos de forma otimizada para que sejam utilizados pelo componente *ContextSearch* no processo de *Information Retrieval* (IR).

O processo de IR é algo que de modo informal qualquer indivíduo poderá executar diariamente, por exemplo, quando seleciona um dos seus cartões bancários para procurar o número que necessita para realizar uma compra online. Embora o exemplo anterior possa ser considerado de alguma forma *IR*, do ponto de vista dos estudos académicos *IR* pode ser definido como:

> *Information Retrieval* é encontrar materiais (habitualmente documentos) de natureza não estrutura (tipicamente texto) que satisfaça uma determinada necessidade de informação, a partir de coleções de materiais armazenados em larga escala (normalmente armazenados em computadores). (Christopher D., Prabhakar and Hinrich 2008)

Nos sistemas de QA com MRC, os resultados do modelo podem ser excelentes quando testamos uma pergunta e respetiva resposta para um dado contexto previamente selecionado, no entanto quando a resposta tem de ser encontrada numa coleção de contextos, a qualidade das

respostas pode baixar drasticamente. No capítulo 4, são apresentados alguns resultados que comprovam esta realidade.

Com o objetivo de selecionar a melhor abordagem para a pesquisa dos textos relevantes, foram utilizadas duas bibliotecas para avaliar, no âmbito deste trabalho, quais os algoritmos mais adequados para pesquisa. Assim, foram analisadas duas abordagens, uma com abordagem mais clássica baseada em índices invertidos para *n-gram exact match* e TF-IDF ou BM25 *scores,* e outra baseada em algoritmos que utilizam similaridade e proximidade entre representações vetoriais (*embeddings*), obtidos a partir de modelos de redes neuronais como alguns dos que já foram anteriormente referidos neste documento.

1. ***ElasticSearch*** – Projeto iniciado em 2000 tem como motor de indexação e pesquisa o *Apache Lucene* que implementa uma pesquisa bastante otimizada com utilização de índices invertidos. De modo simplificado, um índice invertido de documentos de texto, estabelece uma relação entre as palavras objeto de pesquisa e os documentos onde essas palavras podem ser encontradas. A cada resultado de pesquisa é atribuído um *score* relativamente à *query* de pesquisa, utilizando nomeadamente a similaridade do cosseno entre os vetores esparsos construídos utilizando BM25. Na última versão anunciada em 2022 (*ElasticSearch 8*)[23], já disponibiliza uma implementação de ANN *(Approximate Nearest Neighbor)* com o algoritmo HNSW *(Hierarchical Navigable Small World*), permitindo a pesquisa utilizando vetores de grande dimensão como os *embeddings*.
2. ***FAISS (Facebook AI Similarity Search)*** [24] – Esta biblioteca foi disponibilizada pelo *Facebook* em 2017 e tem como objetivo disponibilizar as funcionalidades de indexação e pesquisa apenas baseado nos *embeddings* dos materiais de pesquisa, sejam textos, imagens ou outros cujas representações vetoriais tenham sido aprendidas de forma automática por algoritmos de redes neuronais. Esta biblioteca foi construída com o grande objetivo de escalabilidade, permitindo manter níveis de utilização de memória fixos, ajustando a relação entre a *accuracy* e o tempo de resposta.

Para facilitar a utilização destes dois *retrievers*, foi utilizada a biblioteca *python haystack*[25]*.* Ambas as implementações podem ser utilizadas individualmente por configuração, existindo uma *Factory*[26], responsável por devolver o *retriever* que se pretende utilizar.

[23] https://www.elastic.co/blog/introducing-approximate-nearest-neighbor-search-in-elasticsearch-8-0
[24] https://engineering.fb.com/2017/03/29/data-infrastructure/faiss-a-library-for-efficient-similarity-search/
[25] https://github.com/deepset-ai/haystack
[26] https://en.wikipedia.org/wiki/Factory_(object-oriented_programming)

Elastic Search – BM25

A utilização do *ElasticSearch* foi uma utilização básica tendo sido implementada a classe *ElasticSearchBM25Retriever*, cujas responsabilidades são: Estabelecer a ligação com o *cluster* de *ElasticSearch* (no âmbito do protótipo ficou apenas um nó local), criar a estrutura *ElasticSearch* dos documentos a indexar, limpar todos os documentos e efetuar as pesquisas. No Trecho de Código 2, é apresentado um exemplo da estrutura de documento submetida para o *ElasticSearch.*

```
{
    "content": "A Terra é o único planeta onde reconhecemos a existência de
vida, um fenómeno possível graças à presença de água líquida e à existência
de temperaturas propícias ao seu desenvolvimento. Os seres vivos, bem como
os ambientes onde se integram, constituem a biosfera, um subsistema da Terra
que tem sofrido mudanças profundas desde o aparecimento dos primeiros
organismos no planeta.",
    "meta": {
                "page":"11",
                "topic":"Diversidade Biológica"
            }
}
```

Trecho de Código 2 - Exemplo de documento a indexar no ElastSearch

FAISS – Dense Passage Retrieval

Dense Passage Retrieval (DPR) aplicado a QA foi introduzido em 2020 por (Karpukhin, et al. 2020), demonstrando que em cenários de sistemas de QA em *open domain* em que a componente de *information retrieval* tem um papel preponderante, é possível obter resultados melhores utilizando apenas *embeddings* quando comparados com os resultados obtidos utilizando as tradicionais representações vetoriais esparsas com TD-IDF ou BM-25.

Utilizando o exemplo dos autores, se considerarmos a questão *"Who is the bad guy in lord of the rings?"* que pode ser respondida a partir do seguinte contexto *"Sala Baker is best known for portraying the villain Sauron in the Lord of the Rings trilogy."*, um sistema baseado na comparação dos termos da pesquisa terá alguma dificuldade em perceber que existe uma forte relação semântica entre a expressão *"bad guy"* e *"villain"*.

Para o treino de um DPR é necessário que sejam criados os *embeddings* das passagens de texto (contexto) onde pretendemos executar a pesquisa, assim como os *embeddings* para o conjunto de questões possíveis do *dataset*. Seguindo a recomendação dos autores, a criação destes *embeddings* pode ser efetuada utilizando o *embedding* do *token* [CLS] obtido com o *BERT*. A semelhança entre a passagem e as questões é calculada recorrendo ao produto escalar entre os vetores obtidos para as passagens e questões. A criação de um *dataset* para o treino do modelo é composto por um conjunto de questões e está relacionada com uma passagem positiva (relevantes para a questão) e um conjunto alargado de passagens negativas (irrelevantes para a questão). Sendo p_i^+ as passagens positivas, p_i^- as passagens negativas e q_i as questões, temos

que as instâncias do treino são definidas no seguinte conjunto $D = \{q_i, p_i^+, p_{i,1}^-, \dots, p_{i,n}^-\}_{i=1}^m$. O processo de treino consiste na otimização da seguinte probabilidade:

$$L\left(q_i, p_i^+, p_{i,1}^-, \dots, p_{i,n}^-\right) = -\log \frac{e^{sim(q_i, p_i^+)}}{e^{sim(q_i, p_i^+)} + \sum_{j=1}^{n} e^{sim(q_i, p_{i,j}^-)}} \quad (13)$$

Para a implementação do *retriever* com a biblioteca FAISS, foi preparado um *dataset* a partir do SciQ[27] traduzido para português, de modo a criar as instâncias de treino necessárias com as questões, passagens positivas e passagens negativas, de um modelo DPR em português.

Com o objetivo de melhor a qualidade dos *embeddings,* foi efetuado o *fine-tuning* baseado no modelo '*neuralmind/bert-base-portuguese-cased*' para a tarefa de *masked language model* (MLM) a partir de um *dataset* com passagens de texto do livro de Biologia e Geologia. O objetivo foi criar um modelo de linguagem com maior conhecimento sobre vocabulário de domínio da disciplina em causa, o designamos de *BioGeoModel*. Este modelo, foi então utilizado no DPR para o *embedding* das passagens de texto e das questões.

Neste treino, foi utilizada fundamentalmente a biblioteca *python haystack* com a classe *DensePassageRetriever* a qual permite receber os parâmetros e hiper-parâmetros necessários para a realização do treino e validação do modelo, incluindo a possibilidade de receber diretamente o nome do ficheiro de treino e de validação (respeitando a estrutura definida pela biblioteca).

Uma vez tendo o modelo treinado, as passagens obtidas quer pelo *PDFTextExtractor* e pelo *ExcelTextExtractor* são armazenadas utilizando a biblioteca FAISS obtendo os *embeddings* dessas sequências de texto através do novo modelo. O processo de pesquisa é realizado pelo componente *ContextSearch* descrito em 3.3.1 .

3.3 Virtual search assistant

O *Virtual search assistant* é a face visível do projeto EVGuru. Trata-se de uma aplicação web desenvolvida com a biblioteca *React JS* + *Typescript*, a qual desempenha as funções de *interface* entre o utilizador e os componentes de IA com os quais comunica por via de uma API de serviços *REST* em *python*.

Na Figura 35 - Exemplo do comportamento do prótotipo EVGuru em contexto com a EV.Figura 35 é apresentado um *printscreen* do protótipo EVGuru em funcionamento.

[27] https://huggingface.co/datasets/sciq

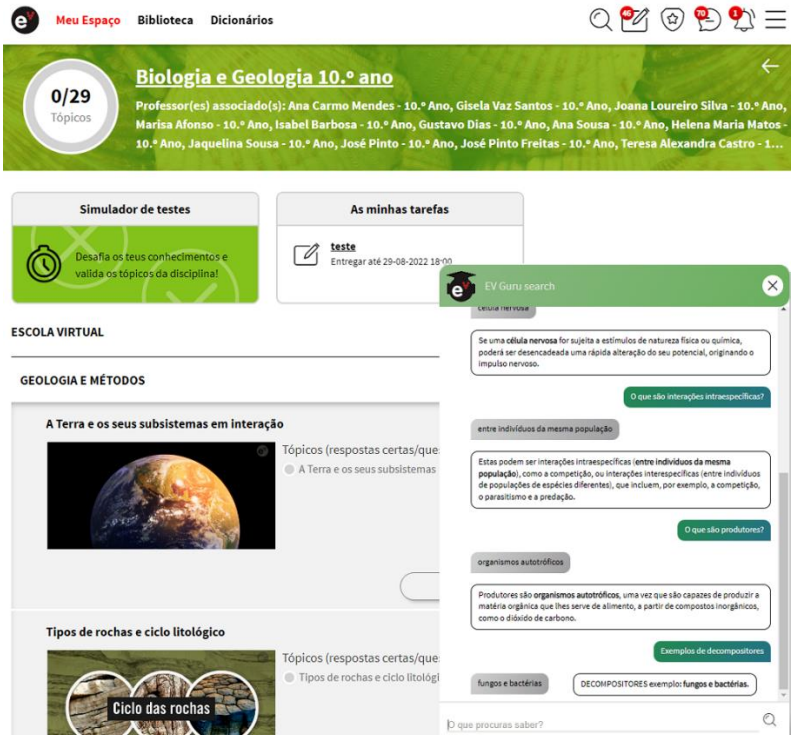

Figura 35 - Exemplo do comportamento do prótotipo EVGuru em contexto com a EV.

Arquitetura de serviços REST adotada permite flexibilidade e independência relativamente à camada de apresentação, permitindo que a inteligência da pesquisa possa ser utilizada em diferentes aplicações e funcionalidades, independentemente da tecnologia em que estas foram implementadas. Por outro lado, a própria aplicação web *React JS*, facilmente poderá ser adaptada para permitir ser incorporada em qualquer página web numa lógica de *widget* apresentado em qualquer elemento de html que possa servir de contentor (*container*) para a apresentação, como por exemplo o elemento html .

3.3.1 Context Search

Este é um dos componentes mais básicos da arquitetura do EVGuru. Tem a responsabilidade de interagir com a *Factory* de *retrievers* para obter um *ElasticSearchBM25Retriever* ou *FAISSRetriever* beneficiando do trabalho realizado por estes na fase de armazenamento das passagens de texto. Paralelamente, uma vez obtida a reposta, efetua a transformação da resposta obtida pelos *retrievers* numa estrutura mais adequada para ser trabalhada na apresentação.

3.3.2 Sentence Evaluator

O objetivo deste componente é servir de *facade* [28] para interagir com os componentes *IntentClassifier* e *MediaSearchQueryBuilder*. Basicamente, perante uma frase escrita pelo utilizador, este componente obtém a classificação da intenção através do *IntentClassifier* e, no caso de uma intenção para construção de *query* de recursos, passa essa intenção ao *MediaSearchQueryBuilder* que apenas devolve a construção dos critérios de pesquisa que podem ser submetidos ao componente já existente na plataforma EV, *MediaSearch* para obtenção de uma lista de recursos multimédia relevantes.

3.3.3 Intent Classifier

Este componente concentra a responsabilidade de identificar as intenções do utilizador, permitindo assim direcionar convenientemente a pesquisa para os conteúdos e modelo de IA que mais faça sentido para devolver resultados ao utilizador. Na Figura 36, são apresentadas as intenções que foram consideradas na definição inicial do projeto.

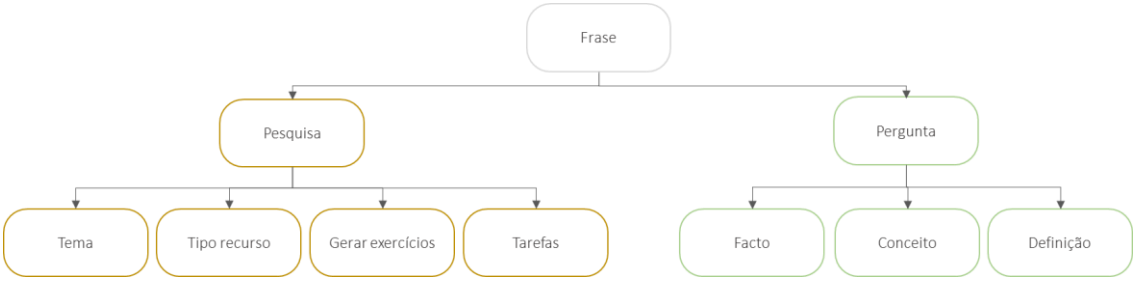

Figura 36 - Exemplo de intenções

No âmbito e tempo disponível para o projeto não foi possível implementar um modelo capaz de classificar todas as intenções permitindo posteriormente avançar para o *slot-tagging* utilizando por exemplo, *Named Entity Recognition* (NER). Para a classificação de todas as intenções seria necessário a criação de um *dataset* suficientemente grande de textos (frases) devidamente etiquetado, o qual não foi possível obter. Este será um dos aspetos a melhorar no futuro.

Como prova de conceito, foram considerados apenas a duas primeiras classificações "Pergunta" ou "Pesquisa". Para isso, foi construído um *dataset* com 2100 termos e frases recolhidas do motor de pesquisa atual da EV e etiquetados manualmente nas categorias "Pesquisa" ou "Pergunta". De modo a aumentar o volume destes dados, foi utilizado o *dataset TREC Question Classification dataset*[29] (Ellen M 2002) com 5500 questões. Dado que as questões *TREC* estão em inglês, foi utilizada a biblioteca *python GoogleTrans* para efetuar a tradução das primeiras 3700 questões para português. Estas 3700 questões foram então concatenadas às 2100 etiquetadas manualmente para complementar o *dataset* de classificação de "Pesquisa" ou "Pergunta".

[28] https://en.wikipedia.org/wiki/Facade_pattern
[29] https://www.kaggle.com/datasets/ananthu017/question-classification

Utilizando este *dataset* foi então implementação um modelo de classificação de *machine learning* clássico, *Logistic Regression*.

Para as duas primeiras classificações de topo, poderia ser seguida uma abordagem obrigando o utilizador a utilizar o sinal de pontuação "?" no final sempre que pretende-se realizar uma pergunta, mas tratando-se de uma ferramenta para alunos que poderão ser muito jovens, o sistema deverá ser capaz de inferir a partir do padrão da estrutura da sequência de palavras, se estamos ou não perante uma pergunta.

3.3.4 MediaSearch Query Builder

Este componente foi também desenhado na arquitetura base do sistema, no entanto não tem atualmente uma implementação. O seu objetivo é gerar a *query* adequada para a pesquisa de recursos disponíveis através do componente atualmente já existente na plataforma EV *MediaSearch*.

O componente deverá reagir às seguintes intenções:

1. Pesquisa →Tema
2. Pesquisa →Tipo de recurso
3. Pesquisa → Gerar exercícios

Para estas intenções, recebe as entidades extraídas utilizando um modelo de *NER* e com esta informação constrói a *query* indicada para a pesquisa na base de dados de recursos.

3.3.5 QA Evaluator

Este componente é responsável pela identificação do segmento de texto mais provável para corresponder à resposta a dar ao utilizador, tendo para o efeito sido utilizado um modelo implementado com a arquitetura BERT.

Com já referido no capítulo 2, quando surgiu em 2019 a arquitetura BERT desenvolvida pelos investigadores da *Google*, o modelo pré-treinado obteve os melhores resultados até então, nomeadamente com a utilização do *dataset* SQuAD *2.0 (dataset* de referência para a investigação em *Machine Reading*). Pouco tempo depois de ser lançado, a equipa da *Google* disponibilizou o código do modelo em *open-source* e disponibilizou o modelo para *download*. Existem outros modelos de linguagem pré-treinados bastante relevantes atualmente nesta área como T5 o GPT (nas suas variantes), as quais são mais adequados para geração de texto. No entanto, no âmbito deste projeto, optou-se pela abordagem mais simples obtendo apenas a sequência de texto que se encontra no contexto dado e que responde à pergunta do utilizador. Por esta razão, pelos resultados reconhecidos pelos investigadores e sua aplicabilidade e pelo facto de existirem várias bibliotecas e recursos para trabalhar com arquitetura BERT, esta foi a opção para o módulo de QA.

BERT fine-tuning para Machine Reading

Na Figura 37 é apresentado um diagrama representativo das camadas envolvidas no *fine-tuning* do modelo BERT para a tarefa de *Question-Answering*.

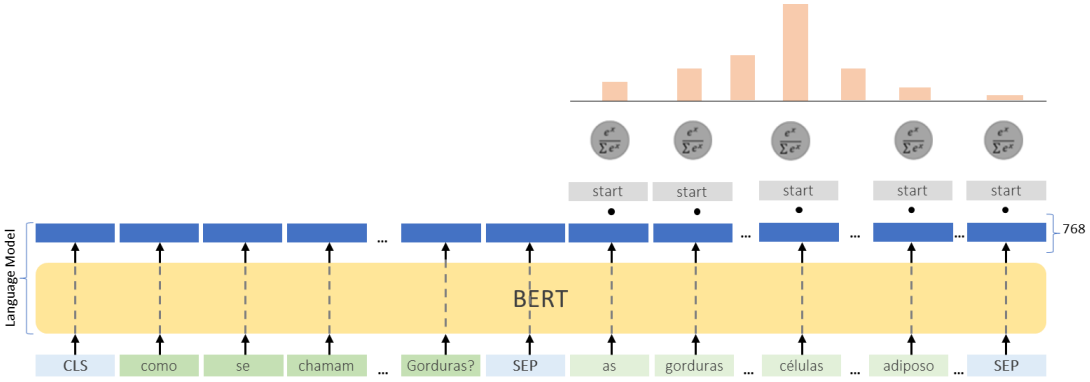

Figura 37 *BERT fine-tuning* para *QA* (adaptação de diagrama[30])

Como descrito no trabalho original dos autores, para o processo de *fine-tuning* do modelo BERT para a tarefa de QA, o modelo recebe duas sequências de texto numa única sequência de entrada. A primeira sequência com a pergunta que pretendemos colocar e a segunda sequência com a passagem de texto sobre a qual pretendemos procurar a resposta, habitualmente designado de contexto, separando estas duas sequências com o *token* especial [SEP], conforme o seguinte exemplo:

> **[CLS]** Como se chamam as células onde ficam armazenadas as gorduras? **[SEP]** As gorduras encontram-se armazenadas nas **células do tecido adiposo** dos animais ou em vários órgãos vegetais, como as sementes ou os frutos. O seu armazenamento é vital para a sobrevivência dos organismos, uma vez que essa gordura pode ser mobilizada para a produção de energia ou funcionar como isolante térmico. Os lípidos são compostos ternários constituídos por carbono, hidrogénio e oxigénio, podendo, no entanto, integrar outros elementos. Constituem um grupo muito heterogéneo que inclui as gorduras, os fosfolípidos, os esteroides e outras moléculas insolúveis em água. As gorduras, ou triglicerídeos, são constituídas por um álcool – o glicerol –, ligado através de ligações éster a três moléculas de ácidos gordos. Estes ácidos são constituídos por longas cadeias de átomos de carbono ligados a átomos de hidrogénio com um grupo carboxilo numa extremidade da cadeia. Da ligação éster que se estabelece entre cada grupo carboxilo do ácido gordo e o grupo hidroxilo do glicerol resulta uma molécula de água. **[SEP]**

O processo consiste basicamente num problema de classificação, cujo objetivo é identificar qual o *token* inicial e *token* final da resposta. No topo da Figura 37 à direita, é apresentada uma representação das camadas adicionais ao modelo de linguagem pré-treinado que são adicionadas para a tarefa na fase de *fine-tuning* de QA com BERT.

Para isso surgem adicionalmente os vetores inicial $S \in \mathbb{R}^H$ e vetor final $E \in \mathbb{R}^H$, calculados durante o treino de *fine-tuning,* com os pesos para o índice que corresponde ao *token* de início da resposta e o outro com os pesos para o índice que corresponde ao *token* de fim do segmento

[30] https://medium.com/saarthi-ai/build-a-smart-question-answering-system-with-fine-tuned-bert-b586e4cfa5f5

de texto da resposta. Como *output* da última camada do modelo pré-treinado temos os *embeddings* T_i do modelo de linguagem. A probabilidade P_i de que a palavra i (pertencente ao segundo segmento) seja a palavra do início do segmento de resposta é dada pelo produto escalar entre $S \cdot T_i$ sobre o qual é aplicada a função *softmax* conforme a seguinte equação:

$$P_i = \frac{e^{S \cdot T_i}}{\sum_j e^{S \cdot T_i}} \quad (14)$$

O processo é o mesmo para determinar a probabilidade do último *token* da sequência. O *score* atribuído ao segmento de texto candidato a ser o segmento que corresponde à resposta correta é calculado efetuando a seguinte soma: $S \cdot T_i + E \cdot T_j$.

Na implementação deste componente, foram realizadas experiências detalhadas no capítulo 4, utilizando modelos disponíveis no repositório de modelos pré-treinados do *Huggingface.* Mais uma vez a preocupação foi encontrar modelos que permitissem comprovar a aplicabilidade e a forma como estes generalizam quando aplicados aos textos de manuais escolares escritos em português.

Embora a utilização do modelo de linguagem pré-treinado reduza significativamente o tempo de processamento necessário para o treino de um modelo de QA *end-to-end*, para conseguirmos obter um modelo com performance aceitável para que possa ser efetuada a disponibilização para os utilizadores, são necessários grandes volumes de dados. Dada a exigência de processamento para estes modelos, este componente foi implementado recorrendo ao modelo de *QA* disponível em *Hugginge face* 'pierreguillou/bert-large-cased-squad-v1.1-portuguese'. Este modelo de *QA* foi treinado utilizando o modelo de linguagem *BERTimbau*, tendo como *dataset* de treino o SQuAD *v1.1* com 100.000 questões/respostas em português disponibilizado pelo *Deep Learning Brasil group*[31]. No Trecho de Código 3, vemos um excerto de código com a importação do modelo.

```
      from transformers import BertForQuestionAnswering, AutoTokenizer, pipeline
      nome_modelo = 'pierreguillou/bert-large-cased-squad-v1.1-portuguese'
      modelpt = BertForQuestionAnswering.from_pretrained(nome_modelo)
      tokenizerpt = AutoTokenizer.from_pretrained(nome_modelo)
      nlp_pt = pipeline('question-answering', model= modelpt, tokenizer=tokenizerp
t)
      result = nlp_pt({
              'question': question,
              'context': context
          })
```

Trecho de Código 3 - Carregamento do modelo BERT para QA

[31] http://www.deeplearningbrasil.com.br/

O componente é responsável pelo carregamento do modelo, recebe os pedidos e devolve as respostas devidamente formatadas, assim como implementa métodos de avaliação da performance do modelo segundo algumas métricas, nomeadamente *exact match* e utiliza o conjunto de métricas *ROUGE (Recall-Oriented Understudy for Gisting Evaluation)* (Lin 2004) obtendo *F1-Score* e *Recall* para *uni-grams* e *bi-grams*.

3.3.6 Question Generator

Embora a geração de questões não tivesse sido algo inicialmente previsto para o projeto, durante a execução do mesmo foram realizadas novas descobertas sobre a aplicação de *deep learning* nas mais variadas tarefas de NLP, tendo sido esta uma das que despertou atenção, nomeadamente a sua aplicação no contexto de *eLearning*.

Dependendo da classificação de intenção calculada pelo componente *IntentClassifier*, o sistema pode utilizar este componente para baseado nas frases inseridas pelo utilizador, gerar questões automáticas sobre os textos mais relevantes encontrados pelo *retriever*.

O objetivo com este componente, será proporcionar ao utilizador uma experiência em que perante um cenário em que o sistema não consegue classificar a sequência de texto inserida como sendo uma questão ou gerar uma *query* de pesquisa, aleatoriamente utilizar este componente para apresentar ao utilizador um conjunto de 3 a 4 perguntas que poderão corresponder à pergunta que o utilizador pretenderia colocar, numa abordagem de "Será que o que pretendes saber é: questão 1? questão 2? …?".

Ao contrário dos modelos de QA apresentados anteriormente em que a saída (resposta) corresponde a um segmento de texto que está presente na passagem de texto fornecida, para a geração de questões o *output* corresponde efetivamente à geração de linguagem. O *Natural Language Generation* (NLG) é umas das áreas de NLP com forte investigação nos últimos anos e com resultados impressionantes como é o caso do modelo GTP-3 da *OpenAI*[32].

O desenvolvimento deste componente teve por objetivo efetuar uma breve avaliação quanto à relevância de questões geradas por um modelo capaz de gerar texto como GPT ou T5, modelos que são treinados *end-to-end* com a arquitetura *encoder-decoder* de *Transformers*. Tendo em conta o objetivo meramente experimental, foi efetuado o *fine-tuning* para a geração de questões utilizado o modelo T5 pré-treinado.

O modelo T5 (*Text-to-Text Transfer Transformer*) é um modelo treinado utilizado aprendizagem não supervisionada para tarefa genérica de *sequence to sequence*. Conforme proposto pelos autores (Raffel, et al. 2020), a utilização do modelo pré-treinado para a realização de tarefas específicas como, classificação, QA, sumarização ou tradução, implica sempre que para um dado texto de *input* seja providenciado um texto objetivo (*target*). Do ponto de vista da arquitetura utilizada, com pequenos ajustes em alguns parâmetros, os autores do modelo utilizaram basicamente a arquitetura de *Transformers* original (Vaswani, et al. 2017). Para a criação do

[32] https://openai.com/

dataset com o volume de testes pretendidos os autores utilizaram o *Common Crawl*[33] que é um repositório disponível online de forma livre, com informação obtida das páginas na internet. Utilizando este repositório, uma vez extraído e limpo o texto selecionado apenas da língua inglesa, foi construído o *dataset* com cerca de 750Gb ao qual deram o nome de *Colossal Clean Crawled Corpus* (C4) e está disponível como parte dos *TensorFlow Dataset*[34].

Para realizar a predição relativa a uma das tarefas específicas pretendidas, o modelo pré-treinado necessita que no *input* seja adicionado um prefixo específico dependendo da tarefa em questão como por exemplo: para tradução "*translate English to German:"*, para sumarização "*summarize:"*. Para carregar o modelo foi utilizada a biblioteca de *transformer* disponível em *Huggingface* com a class *T5ForConditionalGeneration*, a qual implementa uma camada extra linear, designada de *Language Model Head,* cuja saída é da dimensão do vocabulário. De acordo com a documentação da biblioteca é aconselhável a utilização desta classe em tarefas de *sequence-to-sequence*.

Utilizando os modelos disponíveis no *Huggingface* foi selecionado o modelo de linguagem T5 *'unicamp-dl/ptt5-base-portuguese-vocab'* pré-treinado utilizando o corpus de texto BrWaC, no entanto, não foi encontrado qualquer modelo em português treinado para a geração de questões. Utilizando a abordagem de *Transfer Learning*, tendo por base este modelo de linguagem, foi utilizado o SQuAD *v1.1-pt* disponibilizado pelo *Deep Learning Brasil group* para a criação de um *dataset* de textos e questões para o *fine-tuning* de um modelo T5 para geração de questões.

A estrutura base do SQuAD é disponibilizada no seguinte formato:

```
{   "version": 1.1,
    "data": [
        {
            "paragraphs": [
                {
                    "context":"Edward Wilson (1929-…) Investigador…",
                    "qas": [
                       {
                           "answers":[
                             {
                                 "answer_start": 23,
                                 "text": "Investigador da Universidade…"
                             },…],
                           "question":"Quem é Edward Wilson?"
                       },…
                 },…]
        },…]
}
```

Trecho de Código 4 - Excerto no formato *SQuAD v1.1*

[33] https://commoncrawl.org/
[34] https://www.tensorflow.org/datasets

Para a criação do *dataset* de treino do *T5* foram considerados os elementos *"context"* e a lista de questões em *"qas" "question"* para cada contexto. Como para cada contexto podem existir várias questões, estas são separadas utilizando o *token* especial <sep>.

O modelo de linguagem foi utilizado tanto como *tokenizer* (classe *T5TokenizerFast*) tanto como base do modelo a treinar, utilizando fundamentalmente as bibliotecas *PyTorch* e *Transformers.* O *tokenizer* é também responsável por garantir que todas as sequências têm o mesmo comprimento realizando internamente a operação de *padding*. O modelo a realizar *fine-tuning* para a tarefa de geração de questões, deve ser carregado com a classe *T5ForConditionalGeneration*. O treino do modelo foi realizado durante cerca de 2h utilizando uma máquina com GPU RTX5000 com 16Gb, 8xCPU, 32Gb RAM numa subscrição da plataforma de *cloud computing Paperspace* [35] . Os resultados das experiências com o modelo serão apresentados no capítulo 4.

[35] https://www.paperspace.com/

4. Experimentação e Avaliação dos resultados obtidos com os modelos de IA

Neste capítulo são apresentadas as atividades realizadas para o treino dos diferentes modelos utilizados em cada componente da arquitetura do sistema apresentada no capítulo 3, revendo os *datasets* utilizados e a origem dos mesmos, o pré-processamento, o treino dos modelos e as métricas de validação utilizadas.

De modo geral, a performance dos modelos *deep learning* está diretamente relacionada com o volume e qualidade dos dados utilizados para treino e validação. A aplicação destas técnicas ao NLP não é exceção. Paralelamente, a complexidade destes modelos, alguns com milhões de parâmetros para otimizar, requerem uma capacidade de processamento só possível em *hardware* que não está à disponível nos computadores pessoais. Portanto, este capítulo está organizado iniciando com a descrição das principais dificuldades de processamento e *datasets* disponíveis para o treino, seguindo com a análise ao vocabulário dos modelos, efetuando a comparação entre o vocabulário do manual escolar utilizado e o vocabulário dos modelos pré-treinados. Por último, é efetuada a descrição das abordagens de treino e resultados obtidos para os modelos de Classificação, *Dense Passage Retrieval*, *Machine Reading* e Geração de Questões.

4.1 Principais dificuldades

O desenvolvimento de projetos na área de IA, nomeadamente os que envolvem a utilização de *deep learning*, são bastante exigentes quanto ao volume e qualidade dos dados necessários, bem como quanto à capacidade e velocidade de processamento. Embora algumas abordagens como *transfer learning* tenha vindo ajudar mitigar um pouco esta dificuldade, estes aspetos continuam a ser uma dificuldade na execução dos projetos.

4.1.1 Capacidade de processamento

Os resultados obtidos nos últimos anos pelos modelos referidos neste documento baseados na arquitetura de *transformers*, nomeadamente GPT*,* T5 e BERT, como descrito, são modelos bastante complexos e que necessitam de grandes volumes de dados para o treino integral. Felizmente adotando a técnica de *Transfer Learning,* podemos beneficiar dos modelos pré-treinados e realizar apenas o *fine-tuning* para as tarefas mais específicas. De qualquer modo, para o *fine-tuning* de algumas tarefas mais específicas, como as que foram utilizadas neste projeto de *Mask Language Modeling, Reading Comprehension*, *Dense Passage Retrieval* ou *Text Generation*, as necessidades de processamento são bastante elevadas para um computador pessoal normal, nomeadamente sem GPU. A título de exemplo, num portátil com 16GB de RAM e Core i5 10ª geração, o *fine-tuning* de um modelo BERT na tarefa de *sentiment analysis* com o *dataset* do *IMDB*[36] demorou mais de 3 horas de processamento.

[36] https://www.imdb.com/

Após esta conclusão, ainda foram analisadas formas de efetuar o processamento utilizando GPU (*Graphics Processing Unit*), mas como as placas gráficas dos equipamentos disponíveis são AMD[37], não é possível utilizar o CUDA (*Computed Unified Device Architeture*) que apenas funciona com placas gráficas da NVIDIA[38]. No caso das placas AMD existe a possibilidade de utilização da biblioteca *ROCm*[39], mas esta apenas funciona em Linux e todo o ambiente disponível de momento para a realização do projeto é o ambiente Windows.

Perante estas dificuldades foram analisadas algumas soluções *cloud computing*[40] como *Google Colab*[41] ou *Paperspace*[42], tendo sido esta última a opção escolhida.

Dada a exigência verificada em termos de recursos computacionais, este aspeto da infraestrutura necessária para o treino e disponibilização dos modelos em ambiente de produção, será um aspeto bastante importante a merecer análise detalhada na eventualidade da disponibilização de uma solução final deste sistema.

4.1.2 Datasets e modelos pré-treinados

As arquiteturas de *deep learning* são formadas por diversas camadas de redes neuronais capazes de extrair informação e reconhecer padrões a partir de dados não estruturados. A qualidade dos dados que alimentam o treino destas redes, deverá ser suficientemente diversificado no domínio que pretendemos tratar, assim como o volume de dados deverá ser suficientemente grande para que o modelo possa generalizar de forma conveniente para informação não vista. Uma das formas de mitigar o problema da falta de dados suficientes é a utilização duma abordagem de *Transfer Learning* através da utilização de modelos pré-treinados.

As principais dificuldades com os *datasets* e modelos pré-treinados encontrados para a realização deste projeto na vertente de QA, foram as seguintes:

- Apenas foi encontrado 1 *dataset* SQuAD em português relevante.
- Não encontrado qualquer *dataset* em português vocacionado para vocabulário de disciplinas escolares.
- Apenas encontrado 1 modelo de linguagem *BERT* relevante em português do Brasil.
- Apenas encontrado 1 modelo de *Reading Comprehension* relevante em português do Brasil.
- Não encontrado qualquer modelo de *DPR* em português.

Na Tabela 5, é apresentada uma lista dos *datasets* utilizados, assim como a fonte dos mesmos e as tarefas onde estes foram utilizados.

[37] https://www.amd.com/
[38] https://www.nvidia.com/
[39] https://rocmdocs.amd.com/en/latest/
[40] https://en.wikipedia.org/wiki/Cloud_computing
[41] https://colab.research.google.com/
[42] https://www.paperspace.com/

Tabela 5 - Lista de *datasets* e modelos onde foram utilizados

Dataset	*Fonte*	*Tafefa*	*Descrição*
SQuAD v1.1 pt[43]	*Online*	*Reading Comprehension*	*Dataset* utilizado pelos autores do modelo de *Reading Comprehension, 'bert-large-cased-squad-v1.1-portuguese'*
SciQ	*Online*	DPR	A partir deste *dataset* foi criado um novo para treino do DPR. O *dataset* foi traduzido para português utilizando a biblioteca *python Googletrans.*
SQuAD v1.1 pt	*Online*	*Question Generation*	Utilizado para o treino do modelo *T5* para geração de questões.
TREC Question Classification dataset	*Online*	*Text Classification*	A partir deste dataset foram traduzidas 3.700 questões para português utilizando a biblioteca *python Googletrans.*
Lista de *keywords* de pesquisa na EV	*Online*	*Text Classification*	Obtido a partir do *Google Analytics* a lista de termos e sequências de palavras mais pesquisadas numa semana na EV. Utilizado em conjunto com o *TREC.*
Lista de textos relevantes	Manual	*Text Classification*	A partir dos textos extraídos do manual, foi criado um *dataset* com 2.415 passagens etiquetadas com "S","N" dependendo
Lista de textos relevantes	Manual	Text Classification	A partir dos textos extraídos do manual, foi criado um dataset com 2.415 passagens etiquetadas com "S" ou "N" dependendo se a passagem deve ou não ser considerada.
Questões para avaliação do modelo MRC	Manual	Reading Comprehension	Criado dataset com 102 questões para os textos extraídos o manual para avaliação do modelo de MRC.

[43] https://github.com/nunorc/squad-v1.1-pt

4.2 Análise ao vocabulário dos modelos

Como referido ao longo do capítulo 3, as tarefas de NLP implementadas nos diferentes componentes da arquitetura do sistema, tiveram como principal abordagem o *fine-tuning* de modelos pré-treinados BERT para classificação e *Machine Reading* e o modelo *T5* para a geração de questões. A principal preocupação foi encontrar modelos pré-treinados com corpus de texto em português dado que, numa primeira fase, os conteúdos escolares que serão utilizados serão na língua portuguesa.

Os modelos de linguagem BERT e T5 selecionados para o projeto, foram treinados pelos autores recorrendo a textos do BrWaC (*Brazilian Web as Corpus*). O BrWac é composto por cerca de 3,53 milhões de documentos, com cerca de 2,68 biliões de *tokens* e foi construído utilizando as ferramentas[44] da comunidade *WaCky*[45] para efetuar a extração dos textos de sites do Brasil especificamente em domínios *.br* .

Dado que, no âmbito deste projeto, os textos a utilizar serão de domínios específicos de algumas das disciplinas do ensino do 1º ao 12º ano de escolaridade, tendo em consideração os textos da disciplina de Biologia e Geologia 10º ano, foi realizada uma análise comparando o vocabulário encontrado nestes textos com o vocabulário dos modelos pré-treinados multilingue ou especificamente treinados em português. O resultado desta análise serviu como um dos critérios para a seleção do modelo de linguagem pré-treinado a utilizar.

A partir dos textos do manual escolar de Biologia e Geologia 10º ano, depois de extraídos dos ficheiros *PDF* e classificados pelo modelo descrito na secção 3.2.3 , foi obtida a lista de *tokens* utilizando a classe *word_tokenize* da biblioteca *nltk* e para estes *tokens* (excluídos *tokens* de apenas um caracter) foi efetuado um processo de lematização utilizando um modelo pré-treinado para português com a biblioteca *spacy*[46]. Deste processo resultou um vocabulário de 7442 palavras distintas presentes nos textos selecionados do manual escolar.

O vocabulário no modelo de linguagem da arquitetura BERT utiliza *WordPiece tokenizer*, enquanto que o *T5* utiliza *SentencePiece tokenizer*. Ambos são baseados num algoritmo de *tokenization* de subpalavras, no entanto os *tokens* de subpalavras são representados de forma diferente. No caso dos *tokens* obtidos com *WordPiece* é utilizado o prefixo "##" (duplo cardinal) nos *tokens* que não representam o início das palavras, enquanto que nos *tokens* obtidos com *SentencePiece* é utilizado o prefixo "_" (underscore) para os *tokens* que representam o início das palavras.

Exemplo de *WordPiece tokenizer*:

humidificação = ['hum', '##idi', '##ficação']

[44] https://wacky.sslmit.unibo.it/doku.php?id=start
[45] http://wackybook.sslmit.unibo.it/
[46] https://spacy.io/models/pt

Exemplo de *SentencePiece tokenizer*:

humidificação = ['_hu', 'mid', 'ificação']

O processo de *tokenization* por subpalavras apresenta algumas vantagens relativamente ao processo por palavras completas, já que por exemplo, em cenários de grandes volumes de textos com milhões de palavras, permite reduzir o número de *tokens*, assim como mitigar o problema de *OOV (out-of-vocabulary)* já que para palavras em falta é provável que existam subpalavras no vocabulário com *embedding* próximo (similaridade) da palavra em falta.

Tanto o *WordPiece* como o *SentencePiece* baseiam-se no algoritmo de criação de subpalavras *Byte-Pair Encoding (BPE)* (Sennrich, Haddow and Birch 2022), em que as subpalavras são compostas dependendo da frequência desses pares no corpus em análise. Quer isto dizer, que ao efetuar a *tokenization* de palavras menos frequentes teremos a palavra mais fragmentada em diferentes *tokens.*

De modo a avaliar o nível de fragmentação das palavras do vocabulário obtido dos textos do manual com o vocabulário dos diferentes modelos pré-treinados, foi efetuada a contagem do número de fragmentos resultantes do *tokenizer* de cada modelo para cada palavra do vocabulário do manual.

Na Figura 38 é apresentada a comparação entre o número de fragmentos dos *tokens* obtidos com o modelo BERT treinado especificamente em português (*bert-base-portuguese-cased*) e com o modelo multilingue (*bert-base-multilingual-cased*), para o total de palavras no vocabulário do manual de Biologia e Geologia ("Vocabulário"). É possível verificar que o modelo treinado especificamente em português inclui mais palavras sem fragmentação de *tokens*.

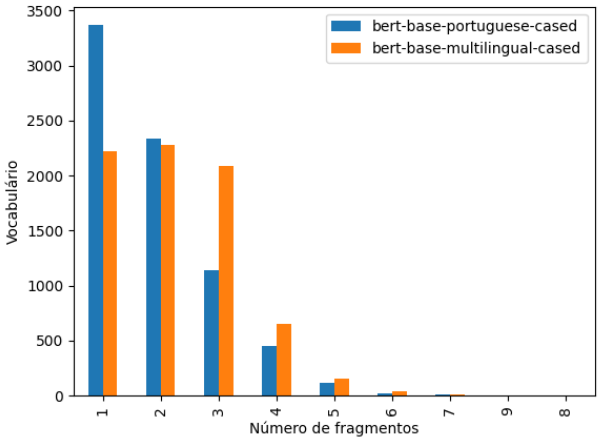

Figura 38 - Comparação entre o vocabulário do modelo BERT multilingue e vocabulário do modelo treinado com BrWaC

Na Figura 39, é apresentada a comparação entre o vocabulário do modelo T5 treinado especificamente em português (*ptt5-base-portuguese-vocab*) e vocabulário de um modelo

multilingue (*google/mt5-base*). Nos modelos T5 comparados, a diferença na fragmentação dos *tokens* é ainda mais acentuada entre o modelo multilingue e o modelo treinado especificamente para português, do que a observada na comparação anterior entre os modelos *BERT*.

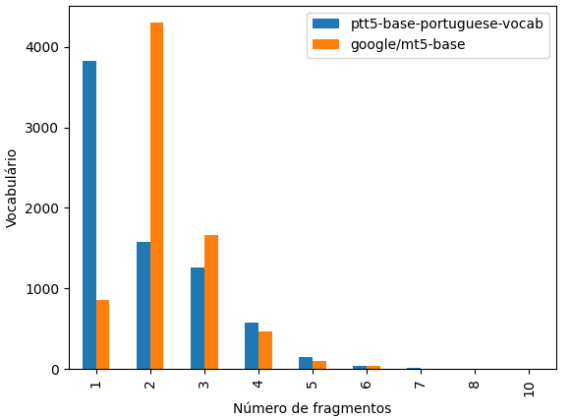

Figura 39 - Comparação entre modelo T5 multilingue e modelo treinado com BrWaC

A análise destes dados permitirá concluir que os modelos treinados com *datasets* em português terão melhores representações da semântica de cada *token* o que será bastante importante no *fine-tuning,* por exemplo para *MRC* ou classificação de textos com estes modelos partir de *datasets* criados em português.

4.3 Avaliação de algoritmos e modelos

No ciclo de vida do processo de desenvolvimento em IA, esta é uma das fases mais importantes e aquela onde a performance dos modelos é avaliada. Parte da metodologia para obtenção do "melhor" modelo passa pela realização de várias experiências numa abordagem de tentativa erro, avaliando os resultados e ajustando os parâmetros disponíveis, repetindo este ciclo de treino e validação até encontrar o modelo que apresenta as melhores métricas de avaliação.

Nesta secção são apresentadas as experiências e resultados obtidos, assim como algumas das métricas de avaliação utilizadas.

4.3.1 Algoritmo de extração de texto do PDF

Grande parte das experiências realizadas foram apresentadas na secção 3.2.2 , dado que foi importante nessa fase utilizar os resultados obtidos com bibliotecas e algoritmos disponíveis para justificar a necessidade de desenvolvimento de um algoritmo específico.

Na Figura 40, é apresentada uma imagem que ilustra as caixas delimitadoras dos textos obtidos.

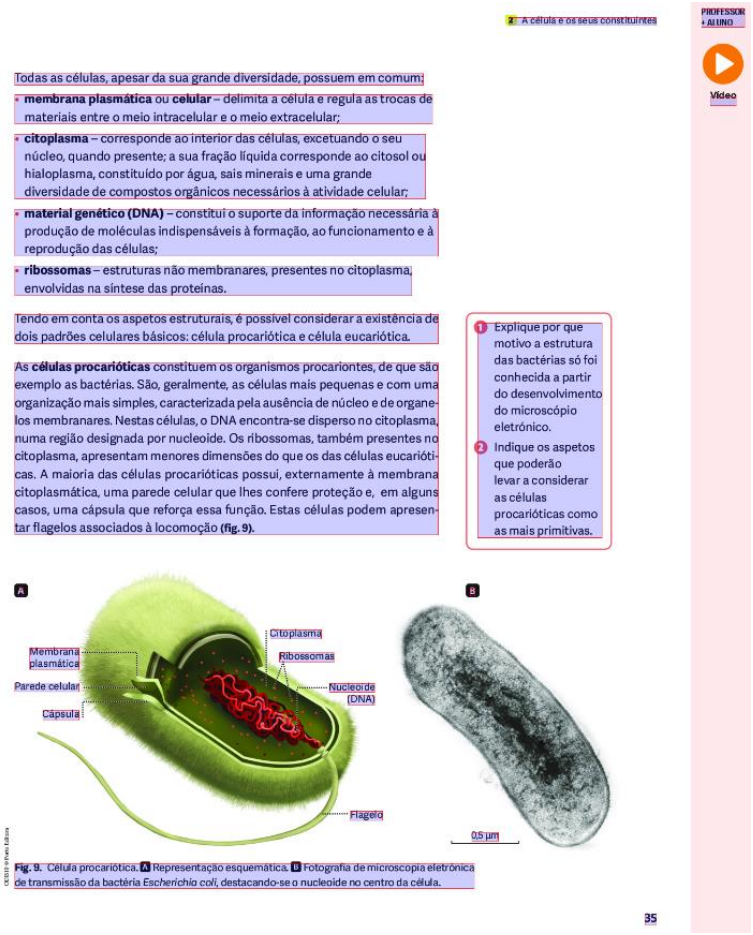

Figura 40 - Representação das caixas delimitadoras dos parágrafos obtidos[47]

A avaliação deste algoritmo foi meramente visual, utilizando para isso o desenho de retângulos sobrepostos sobre a imagem da página PDF, com o objetivo de alcançar visualmente a representação mais semelhante daquilo que seria a separação efetuada por um humano.

4.3.2 Modelos de classificação

Para a avaliação de modelos de classificação, a par das métricas *F1-Score, recall* e *precision* cujas fórmulas (6), (7) e (8) foram apresentadas em 2.3.1 , é também importante apresentar uma breve descrição sobre a Matriz de Confusão bastante utilizada na avaliação destes modelos.

A Matriz de Confusão (Figura 41) pode ser utilizada em problemas de classificação binária ou multi-classe e permite visualizar na forma de matriz, qual a performance do modelo de classificação, permitindo obter uma comparação entre a classe da previsão obtida pelo modelo com a classe real.

[47] Manual escolar Odisseia Biologia e Geologia 10º da Porto Editora

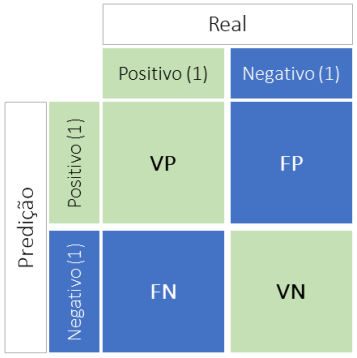

Figura 41 – Representação da Matriz de Confusão

A partir dos valores representados na matriz, podemos calcular métricas como *accuracy, precision, recall, f1-score*, entre outras.

Esta matriz para um problema de classificação binária, apresenta 4 valores:

- (VP) O valor positivo previsto pelo modelo **coincide** com o valor positivo real. Ou seja, a previsão do modelo foi 1 e o valor real também é 1.
- (FP) O valor positivo previsto pelo modelo **não coincide** com o valor negativo real. Ou seja, a previsão do modelo é 1 mas o valor real é 0.
- (FN) O valor negativo previsto pelo modelo **não coincide** com o valor positivo real. Ou seja, a previsão do modelo é 0 mas o valor real é 1.
- (VN) O valor negativo previsto pelo modelo **coincide** com o valor negativo real. Ou seja, a previsão do modelo é 0 e o valor real é 0.

Esta visualização, para além de permitir uma rápida perceção sobre a performance do modelo, permite também ter uma visualização sobre o balanceamento dos dados entre as diferentes classes.

Uma das métricas mais utilizadas e que permite quantificar a performance geral do modelo é a *accuracy.* Esta métrica permite calcular a razão entre as classificações corretamente efetuadas e o total de observações a classificar e pode ser obtida da seguinte forma:

$$\text{accuracy} = \frac{VP + VN}{VP + FP + VN + FN} \quad (15)$$

Classificação da qualidade do texto para information retrieval

As experiências realizadas, tiveram como objetivo avaliar os modelos descritos em 3.2.3 , permitindo selecionar o modelo suficientemente eficaz para a classificação dos textos extraídos dos *PDFs* dos manuais escolares, de forma a submeter para a base de dados de passagens de

texto, apenas aqueles com conteúdo relevante para as eventuais questões dos alunos sobre conceitos, definições ou outros factos relacionados com os tópicos da disciplina em análise.

Nesta avalização foram comparados os três modelos, o modelo clássico de *machine learning Logistic Regression*, com os modelos de *deep learning,* um com LSTM e outro com BERT.

O *dataset* utilizado era composto por 2.416 passagens de textos extraídas dos manuais Biologia e Geologia 10º ano. No total, o *dataset* está pouco balanceado sendo composto por 1.466 passagens classificadas como não importantes ("N") e 949 ("S") como importantes para o sistema de *QA.*

Classificação com Logistic Regression

Para o treino deste modelo foram realizadas as atividades de pré-processamento que incluíram a remoção de alguns caracteres não desejados e a técnica de *stemming*. O *dataset* foi separado na proporção de 80% (1932) para treino e 20% (483) para teste, utilizando o método s*tratified* disponibilizado pela biblioteca *sklearn* de modo a garantir a mesma distribuição entre as duas classes no *dataset* de treino e de test.

Na Figura 42 é apresentada a Matriz de Confusão obtida com o modelo e na Figura 43 os valores das métricas calculadas, nomeadamente a *accuracy* de 0,86.

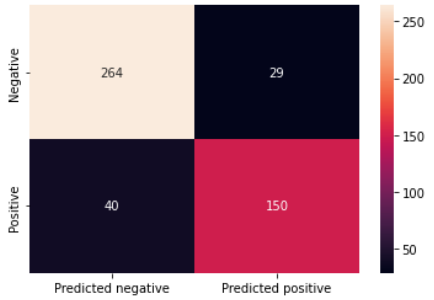

Figura 42 - Matriz de confusão para *Logistic Regression*

```
== Classification Report (Logistic -- Test data) ==
              precision    recall  f1-score   support

           N       0.87      0.90      0.88       293
           S       0.84      0.79      0.81       190

    accuracy                           0.86       483
   macro avg       0.85      0.85      0.85       483
weighted avg       0.86      0.86      0.86       483
```

Figura 43 - Métricas com *Logistic Regression*

Classificação com LSTM

Nesta abordagem foram efetuadas várias experiências utilizando uma arquitetura de LSTM e BiLSTM. As experiências com estas duas arquiteturas obtiveram sempre melhores resultados utilizando os *embeddings FastText* do que utilizando uma abordagem de treino da camada de *embedding*. A abordagem para obter o melhor modelo nesta arquitetura para o *dataset* disponível, consistiu na execução de múltiplas sessões de treino com validação e teste,

efetuando alterações em alguns dos Hiper parâmetros dos modelos, nomeadamente *dropout* e *epochs*.

Tendo em conta que a saída destes modelos tem de ser uma saída numérica, o valor 0 representa a classificação para textos que não devem ser utilizados para a base de textos com conteúdos relevantes para QA e 1 para os textos relevantes.

Analisando os resultados da Figura 44, verificamos que o modelo de classificação com LSTM apresenta resultados bastante baixos, por exemplo com um *f1-score* de 0,61 para a categoria 1 o que indicia que uma predição nesta categoria poderá ser considerada meramente a obtenção de um resultado aleatório (probabilidade próxima de 0,5).

	precision	recall	f1-score	support
0	0.74	0.86	0.80	293
1	0.72	0.53	0.61	190
accuracy			0.73	483
macro avg	0.73	0.70	0.70	483
weighted avg	0.73	0.73	0.72	483

Figura 44 - Resultados com *LSTM*

Na Figura 45 e Figura 46, são apresentados os resultados obtidos com BiLSTM. Neste caso, embora os resultados não sejam ótimos continuam abaixo do modelo com *Logistic Regression*, o modelo apresenta resultados coerentes na aprendizagem para ambas as categorias com valor de 0,78 de *accuracy*. Tendo em conta a reduzida dimensão do *dataset* (2.416 passagens de texto), com apenas 39% das instâncias para a classe 1, dada a forma como estes modelos de *deep learning* aprendem, será de esperar que se consiga obter facilmente melhores resultados aumentando o volume de dados de treino com passagens de texto de outros manuais escolares.

	precision	recall	f1-score	support
0	0.91	0.70	0.79	293
1	0.66	0.90	0.76	190
accuracy			0.78	483
macro avg	0.79	0.80	0.78	483
weighted avg	0.81	0.78	0.78	483

Figura 45 - Resultados com *BiLSTM*

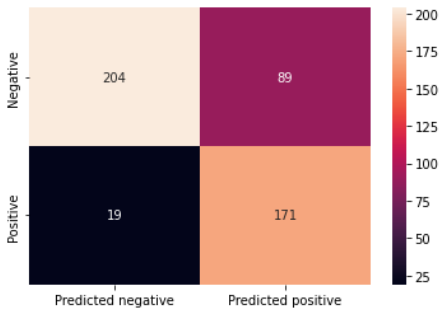

Figura 46 - Matriz de confusão com *BiLSTM*

Classificação BERT fine-tuning

O modelo de classificação com BERT descrito em 3.2.3 , foi treinado com o mesmo *dataset* dos anteriores e foram utilizados os parâmetros *default* do modelo durante apena 4 *epochs.* Com esta configuração, efetuando uma divisão do *dataset* com 20% das instâncias para teste e 16% para validação, foram obtidos os resultados apresentados na Figura 47 e Figura 48, com 0,87 de *accuracy*.

	precision	recall	f1-score	support
0	0.84	0.97	0.90	293
1	0.93	0.73	0.82	190
accuracy			0.87	483
macro avg	0.89	0.85	0.86	483
weighted avg	0.88	0.87	0.87	483

Figura 47 - Resultados com *BERT finetuned*

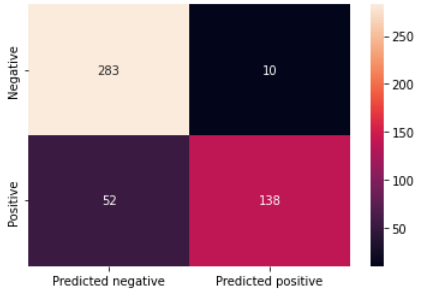

Figura 48 - Matriz de confusão com o modelo *BERT*

Mesmo sem realização de mais experiências com diferentes configurações dos hiper parâmetros, o modelo BERT apresentou logo na primeira tentativa os melhores resultados das quatro abordagens testadas. Tendo em conta a reduzida dimensão do *dataset* de treino, estes resultados comprovam a eficácia do *transfer learning*, nomeadamente neste caso com a utilização de um modelo pré-treinado em português e que são também reveladores da qualidade dos *embeddings* do modelo.

4.3.3 Question Answer

Machine Reading Comprehension

Atendendo ao facto de não existirem *datasets* de perguntas e respostas no domínio das disciplinas escolares, disponíveis para o *fine-tuning* do modelo de linguagem pré-treinado na arquitetura BERT para a tarefa de MRC, foi avaliado o modelo pré-treinado com o *dataset* SQuAD *1.1pt,* especificamente treinado em português para a realização desta tarefa.

Para conseguir uma melhor avaliação do modelo, foram criadas manualmente 102 questões para os textos do livro de Biologia 10º ano conforme os exemplos da Tabela 6.

Tabela 6 - Exemplos contexto, questões e respostas do *dataset* de avaliação

Contexto	
Questão	Resposta
Edward Wilson (1929-…) Investigador da Universidade de Harvard, conhecido pelo seu trabalho em Ecologia e em Evolução, tem sido um acérrimo defensor da biodiversidade a nível global.	
Quem é Edward Wilson?	Investigador da Universidade de Harvard
A designação das enzimas integra, frequentemente, o nome do substrato sobre o qual atua, acrescentando-se o sufixo ase; por exemplo, as proteases hidrolisam as proteínas e a sacarase promove a dissociação da sacarose.	
O que está integrado na designação das enzimas?	o nome do substrato sobre o qual atua

Tal como apresentado na secção 2.3.2 , na tarefa de *MRC* as métricas de avaliação mais utilizadas são *Exact Match* (EM) e *F1-Score*.

O cálculo do *EM* mede a percentagem de respostas (segmentos de texto) obtidos da previsão do modelo que são exatamente iguais às respostas criadas manualmente para o *dataset* de teste. Embora esta métrica transmita uma avaliação importante quanto à performance do sistema, dado que duas sequências de texto podem variar num ou noutro termo mantendo o sentido, o facto de termos um EM relativamente baixo (por exemplo 60%), não permite por si só aferir se, de modo geral, as respostas são ou não corretas. Por esta razão foi também utilizado como métrica o *F1-Score, Recall* e *Precision* calculado para *Rouge-N (*neste caso foram utilizados *uni-grams*, *bi-grams)* (Lin 2004) utilizando para isso a biblioteca *python ROUGE*[48].

As métricas *ROUGE* consistem no cálculo de *F1-Score, Recall* e *Precision* tendo por base diferentes formas de contabilizar as previsões positivas ou negativas numa sequência de texto.

No caso de *Rouge-N*, baseia-se no cálculo de *Recall* contabilizando os *n-grams* da sequência candidata (resultando da previsão do modelo) e a sequência de referência (habitualmente produzida por um humano). Formalmente é definida da seguinte forma:

$$\text{Rouge-N} = \frac{\sum_{S \in \{ReferenceSummaries\}} \sum_{grams_n \in S} Count_{match}(gram_n)}{\sum_{S \in \{ReferenceSummaries\}} \sum_{grams_n \in S} Count\,(gram_n)} \qquad (16)$$

Onde n corresponde ao tamanho dos *n-grams* e $Count_{match}(gram_n)$ é o número máximo de *n-grams* coexistente na sequência candidata e na sequência de referência.

Assim é possível representar o cálculo de *Recall* como:

[48] https://pypi.org/project/rouge/

$$Recall = \frac{\text{número de } n\text{-grams coexistentes}}{\text{número de } n\text{-grams de referência}} \quad (17)$$

O cálculo da *Precision* como:

$$Precision = \frac{\text{número de } n\text{-grams coexistentes}}{\text{número de } n\text{-grams da candidata}} \quad (18)$$

O cálculo de *F1-Score* pode então ser calculado a partir da *Recall* e *Precision* conforme indicado na equação (8).

Dada a falta de *datasets* de treino, de modo a criar uma *baseline* para os resultados de *MRC*, foi utilizado o modelo pré-treinado *"pierreguillou/bert-large-cased-squad-v1.1-portuguese"*. A validação efetuada com este modelo para as $N = 102$ questões resultou em $\hat{y} = 70$ respostas corretas, obtendo-se $\frac{\hat{y}}{N} = 68{,}63\%$ *accuracy* para o *exact match.* Na Tabela 7 são apresentados os valores para as métricas *F1-Score*, *Recall* e *Precision*.

Tabela 7 - Métricas de validação para *n-grams*

n-grams	F1-Score	Recall	Precision
uni-grams	0,86	0,85	0,90
Bi-grams	0,71	0,71	0,73

Na Figura 49 é possível verificar que o modelo falha mais quando o número de *tokens* da resposta esperada é maior.

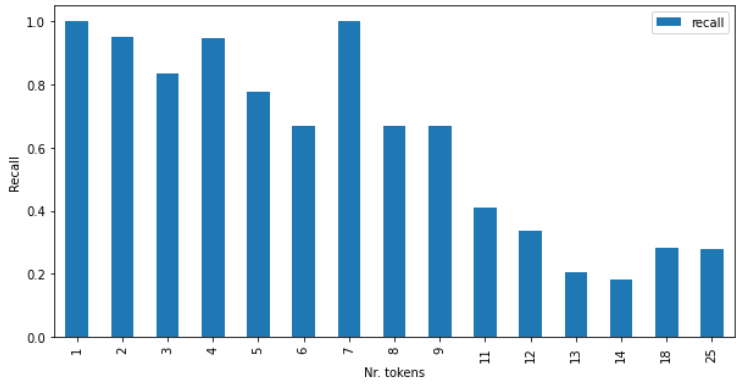

Figura 49 - Relação entre o *recall* e o número de *tokens* da resposta esperada

Retriever

As métricas anteriores foram calculadas num cenário de contexto fechado, isto é, existe apenas uma passagem para a pesquisa da resposta e não existe *retriever*. Neste cenário o modelo recebe especificamente a pergunta e o contexto onde sabemos à priori que existe a resposta, no entanto,

no âmbito do projeto, foi também necessário avaliar os resultados do modelo de *MRC* quando a funcionar em conjunto com o *retriever* procurando os segmentos das respostas em todos os textos do manual que foram armazenados.

Com a utilização de *retriever* com algoritmo de *score* baseado no *BM25,* para as $N = 102$ questões resultou em $\hat{y} = 54$ respostas corretas, obtendo-se $\frac{\hat{y}}{N} = 52{,}94\%$ *accuracy* para o *exact match.* Como é possível verificar na Tabela 8 e Tabela 9, a performance geral do sistema de QA cai significativamente.

Tabela 8 - Metricas de avaliação para o modelo com retriever (BM25)

n-grams	F1-Score	Recall	Precision
uni-grams	0,59	0,59	0,60
Bi-grams	0,47	0,47	0,48

Tabela 9 - Metricas de avaliação para o modelo com *retriever (DPR multilingue)*

n-grams	F1-Score	Recall	Precision
uni-grams	0,46	0,46	0,49
Bi-grams	0,34	0,35	0,35

Estes resultados podem ser considerados algo desanimadores, no entanto, para uma melhor validação, seria necessário incluir nesta análise a validação de um "perito" no domínio para verificar quais são de facto as respostas verdadeiramente certas ou erradas.

Por exemplo, para a seguinte questão:

"Para o que serve a gordura nos animais?"

A resposta baseada no *dataset* utilizado para validação seria:

"produção de energia ou funcionar como isolante térmico"

Obtida a partir da passagem de texto:

> *"As gorduras encontram-se armazenadas nas células do tecido adiposo dos animais ou em vários órgãos vegetais, como as sementes ou os frutos. O seu armazenamento é vital para a sobrevivência dos organismos, uma vez que essa gordura pode ser mobilizada para a* **produção de energia ou funcionar como isolante térmico***. Os lípidos são compostos ternários constituídos por carbono, hidrogénio e oxigénio, podendo, no entanto, …"*

Enquanto que o sistema de QA, obteve como resposta:

"protetora e energética"

Obtida a partir da passagem de texto que o *retriever DPR* considerou mais relevante para a questão:

> *"Os lípidos caracterizam-se pela sua insolubilidade na água. As gorduras têm uma função* ***protetora e energética****. Os fosfolípidos desempenham uma função estrutural, entrando na constituição das membranas plasmáticas."*

Embora com sequências de texto bastante diferentes e consequentemente com métricas penalizadoras do ponto de vista desta análise discreta, numa análise semântica ambas as respostas estariam corretas.

4.3.4 Dense Passage Retrieval

O objetivo do treino específico de um modelo de *Dense Passage Retrieval* (DPR), teve como objetivo avaliar eventuais melhorias nos resultados do sistema de QA, utilizando um DPR especificamente treinado em português, quando comparado com os resultados apresentados na Tabela 9, onde foi utilizado um modelo *DPR* pré-treinado multilingue[49] [50]. Para isso foi utilizado o modelo de linguagem *BioGeoModel* (*finetuned* nos textos do manual escolar de Biologia e Geologia) como *encoder* das passagens e das questões.

Na Tabela 10 são apresentados os resultados obtidos do sistema *QA* com este modelo.

Tabela 10 - Metricas de avaliação para o modelo com *retriever (DPR português)*

n-grams	F1-Score	Recall	Precision
uni-grams	0,48	0,47	0,50
Bi-grams	0,36	0,36	0,36

Pelas mesmas razões apresentadas em 4.3.3 , complementarmente às métricas obtidas, os resultados deveriam ser avaliados semanticamente para uma mais correta avaliação. No entanto, mesmo com a reduzida dimensão do *dataset* utilizado no *fine-tuning* do *BioGeoModel*, verifica-se alguma melhoria com a utilização deste modelo de linguagem, pelo que faria sentido em trabalho futuro, voltar a efetuar o *fine-tuning* na tarefa de MLM utilizando um corpus de textos maior no domínio da Biologia e Geologia, assim como nos domínios de outras disciplinas que venham a ser âmbito do projeto.

[49] https://huggingface.co/voidful/dpr-ctx_encoder-bert-base-multilingual
[50] https://huggingface.co/voidful/dpr-question_encoder-bert-base-multilingual

4.3.5 Geração de questões

Para o modelo de geração de questões, foram realizadas experiências tendo sido efetuado o *fine-tuning* de um modelo pré-treinado *T5* disponível no *Huggingface, 'unicamp-dl/ptt5-base-portuguese-vocab'.*

Com referido em 3.3.6 , o *dataset* para treino do modelo foi criado a partir do *SQuAD v1.0 pt,* tendo resultado num *dataset* de 15.116 questões e passagens. Na Tabela 11 - Exemplo do *dataset* de treino para o modelo de geração de questõesTabela 11 são apresentados alguns exemplos de contextos e questões utilizadas no treino.

Tabela 11 - Exemplo do *dataset* de treino para o modelo de geração de questões

contexto	questões
De acordo com um estudo de DNA autossômico realizado por Hodgson et al. (2014), ...	Quando Hodgson publicou seu estudo de DNA?
Na época, estruturas como álgebras de Lie e quaterniões hiperbólicos chamavam a atenção para a necessidade de …	O que as álgebras de Lie e os quaterniões hiperbólicos demonstram como necessidade?
Em 1980, quando o estádio precisou ser redesenhado para atender aos critérios da Uefa...	Por que o estádio precisou ser redesenhado em 1980?
Israel tem 18.096 quilômetros (11.244 mi) de estradas pavimentadas e 2,4 milhões …	Quantos quilómetros de estradas pavimentadas tem Israel?
Os números oficiais (a partir de 21 de julho de 2008, 12:00 CST) declararam que 69.197 foram confirmados mortos, …	Quantas pessoas foram confirmadas mortas?

Para o processo de treino, o *dataset* criado foi dividido em 80% para treino e 20% para validação, para contextos com tamanho máximo de 512 *tokens* e para 64 *tokens* de comprimento máximo para as questões. O modelo foi treinado com *batch size 4* e para 7 *epochs*. O modelo foi treinado utilizando a biblioteca *Transformers* do *Huggingface* com as classes *TrainingArguments* e *Trainer*.

Os resultados obtidos foram bastante promissores, embora a relevância das questões tenha ainda de ser avaliada do ponto de vista pedagógico. O modelo é capaz de gerar questões gramaticalmente bastante corretas. Na Tabela 12, são apresentados exemplos das questões geradas para o cenário em que o utilizador apenas pesquisa utilizando o termo *"fotossíntese"*. As passagens utilizadas como *input* para o modelo de geração das questões, são selecionadas do *ElasticSearch* utilizando o *retriever* descrito em 3.2.4 .

Tabela 12 - Exemplos de questões geradas

Questões geradas
Em que ano Ruben e Kamen confirmaram a origem do oxigénio libertado na fotossíntese?
Quem descobriu a influência dos comprimentos diferentes da luz no efeito da fotossíntese?
Quando Van Niel descobriu que o oxigénio libertado provinha da água?
Que tipo de reações a fotossíntese integra?
Qual é o processo de troca gasosa no decorrer da fotossíntese numa alga verde?

4.3.6 Conclusões da experimentação

Os diferentes modelos e abordagens utilizados para implementação das tarefas de responsabilidade dos principais componentes, levou a conclusões distintas que importa separar tendo em conta o âmbito de cada uma das mesmas:

- Extrair texto dos manuais escolares em formato PDF – A realização desta tarefa demonstrou ser mais complexa do que inicialmente prevista. As bibliotecas *python* analisadas permitem extrair texto, separando-o inclusive em palavras ou mesmo caracteres, no entanto, todas as que foram analisadas apresentavam alguns problemas provocando repetições de texto ou extraindo sequências não desejáveis. Por outro lado, a extração do texto obtido com as bibliotecas, é uma extração em bruto, não estruturando o texto em parágrafos distintos, algo que se considerou importante para a fase de QA. Para resolver estas dificuldades, utilizando a biblioteca *pdfplumber* e o desenvolvimento de um algoritmo adaptado de *clustering* baseado em DBSCAN, foi possível obter uma extração dos parágrafos de texto (passagens de texto) de forma automática, de modo muito semelhante ao que seria realizado por um humano.

- Classificação das passagens de texto relevantes – Nem todo o texto extraído do PDF pode ser considerado texto relevante para a realização das tarefas dos componentes do assistente de pesquisa virtual. A forma como o texto é distribuído numa página de um manual escolar pode ser bastante diversificada incluindo tabelas e imagens, entre outros elementos, assim como com páginas compostas por textos, por exemplo exercícios e atividade laboratoriais, que deverão ser descartados para a base de dados de passagens a utilizar em QA. Foram testados três modelos de classificação, o algoritmo clássico de *machine learning Logistic Regression* e dois modelos de *deep learning*, um baseado em arquitetura de RNN implementado com BiLSTM e *Fasttext embeddings,* e outro, o BERT com um modelo de linguagem pré-treinado em português adicionando uma camada (BERT *Head)* para a classificação. Tendo em conta a reduzida dimensão do *dataset* de treino, o modelo com BiLSTM não conseguiu obter bons resultados, tendo sido o modelo com BERT o que apresentou a melhor performance. Estes resultados permitiram comprovar a eficácia do *transfer learning,* permitindo que com um *dataset* pequeno se

consiga resultados interessantes na tarefa de classificação com um modelo bastante complexo como é o BERT.

- Classificação de intenções – Para a realização desta tarefa, na implementação do protótipo, o tema da classificação de intenções não foi tão aprofundado como inicialmente previsto. Foram apresentadas conceptualmente as intenções que seriam âmbito de classificação, no entanto, apenas foi implementada uma classificação básica para determinar se a sequência de termos de entrada inserida pelo utilizador corresponderia a uma pesquisa ou a uma pergunta. Para este classificador, foi utilizado *Logistic Regression* que se mostrou bastante eficiente nesta atividade simples de classificação, demonstrando que, tendo em conta a velocidade de treino e inferência, continua a ser um modelo bastante relevante para esta tarefa.

- *Question and Answering* – Na perspetiva das funcionalidades deste protótipo, esta é provavelmente aquela que, tendo em vista a eventual utilização em ambiente real, suscitava maior expectativa inicial quanto à relevância efetiva dos resultados. No QA, existem duas tarefas importantes, *Information Retrieval* (IR) e *Machine Reading Comprehension* (MRC). Quanto ao MRC, a opção continuou a ser a de utilização do BERT, aprofundando os conhecimentos teóricos sobre o *fine-tuning* para esta tarefa, e realizando experiências com modelos pré-treinados em português do Brasil, disponibilizados no *Huggingface*. Para avaliação dos modelos, foi criado um *dataset* com 102 questões do livro de Biologia, tendo obtido resultados com 68,63% de *accuracy* e de 0,86 e 0,71 de *F1-score* calculados com *uni-grams* e *bi-grams* respetivamente. Estes resultados são bastante interessantes, apesar de caíram drasticamente com a introdução do IR. Para a implementação do IR foram utilizadas duas abordagens, uma com pesquisa baseada em BM-25 e outra em *Dense Passage Retrievers* (DPR). Para o conjunto de textos e questões utilizadas para a avaliação, os resultados caíram quase para metade. No entanto, foi possível observar que, em alguns casos, embora a sequência de termos que constitui a resposta obtida pelo *Reader* seja diferente da resposta esperada na comparação dos *n-grams*, numa análise semântica, aquela seria igualmente correta. Esta constatação, leva a concluir que do ponto de vista conceptual um IR baseado em DPR será a opção mais adequada para QA, mas será necessário melhorar o treino do modelo utilizado. Por outro lado, para além da análise discreta, será importante incluir também a de um perito para uma análise semântica na avaliação das respostas obtidas.

- *Geração de Questões* – Esta não era uma tarefa inicialmente prevista no contexto do projeto, no entanto, na fase de investigação foram surgindo vários documentos e artigos neste âmbito o que motivou a realização de algumas experiências no treino e teste para a geração de texto. Para isso, foi utilizado um modelo de linguagem T5 pré-treinado em português e efetuado o *fine-tuning* para a tarefa de geração de questões. Com este *fine-tuning* foi possível criar um modelo final capaz de receber uma passagem de texto e uma

sequência de termos de pesquisa e a partir destes, gerar uma lista de várias questões relacionadas. As questões geradas, são do ponto de vista da análise sintática e semântica corretas, no entanto necessitam de ser avaliadas do ponto de vista da relevância pedagógica para uma efetiva utilização.

5. Conclusão e trabalho futuro

Os objetivos do projeto consistiram, por um lado na realização de um trabalho de investigação sobre as técnicas de IA atualmente mais utilizadas e que apresentam os melhores resultados em *NLP,* especificamente nas tarefas de *Intent Classification, Information Retrieval* e *Machine Reading Comprehension*, por outro no desenho da arquitetura e protótipo funcional dos componentes necessários para o desenvolvimento de um sistema de pesquisa inteligente de textos e recursos digitais, com respostas automáticas ás questões colocadas pelos alunos da plataforma de *eLearning* Escola Virtual.

A investigação iniciou com a análise de soluções de pesquisa utilizadas em sistemas de LMS existentes como o *Moodle* com o *Moodle Global Search*, ou os sistemas *Docebo* e *AbsorbLMS,* estes últimos como exemplo de plataformas com utilização de IA. Mantendo o foco na análise dos motores de pesquisa e abordando a forma como estes funcionam num processo típico de *Information Retrieval*, foi apresentada uma visão sobre os motores de pesquisa tradicionais baseados em índices invertidos e cálculo da relevância dos resultados de pesquisa com TF-IDF ou BM-25, até aos mais recentes modelos de DPR*,* com base em representações vetoriais do texto, capazes de absorver informação da relação semântica entre as palavras. A utilização nos últimos anos de diferentes arquiteturas de modelos de *deep leaning*, nomeadamente a utilização de RNN*,* GRU*,* CNN até à recente arquitetura de *transformers* (2017), têm permitido avanços significativos na capacidade de produzir cada vez melhores representações vetoriais (*embeddings)*, melhorando assim os resultados em várias tarefas de NLP.

Os motores de pesquisa inteligentes, deixaram de ser apenas uma forma de encontrar documentos mais relevantes, sendo agora capazes de "entender" a intenção do utilizador e apresentar diretamente a resposta para uma questão colocada para uma sequência de termos de pesquisa. Nesta área, a investigação realizada abordou o tema do MRC. Aqui o objetivo foi encontrar as melhores técnicas disponíveis para permitir a aplicação do conceito aos textos de manuais escolares e à meta informação disponível nos conteúdos digitais da plataforma EV. Existem fundamentalmente três grandes modelos pré-treinados com os melhores resultados nesta área, nomeadamente o T5, GPT (1,2 e 3) e o modelo BERT. Os dois primeiros, embora possam ser utilizados com bastante sucesso em várias tarefas, são mais adequados para a geração de texto, dado que foram treinados *end-to-end,* com algumas variações, na arquitetura *encoder-decoder* de *transformers.* Já o BERT*,* treinado apenas com *encoder,* é mais adequado, por exemplo, para tarefas de *text classification, named entity recognition* ou *question-answering*.

A investigação realizada foi fundamental para o desenho da arquitetura do sistema de pesquisa inteligente EVGuru, com funcionalidades que incluem modelos treinados para as tarefas que foram âmbito da investigação, com experiências realizadas também na geração de textos, nomeadamente na geração de questões.

O desenvolvimento do protótipo funcional tendo por base a arquitetura definida, permitiu aprofundar o conhecimento sobre as técnicas e modelos de *machine learning* utilizados, experienciar as principais dificuldades na utilização destes modelos e *datasets* disponíveis, avaliar as bibliotecas e modelos pré-treinados, assim como validar a utilização da técnica de *transfer learning* realizando o *fine-tuning* dos modelos para as tarefas específicas de cada componente da arquitetura.

O modelo BERT foi o modelo mais utilizado e com melhores resultados nos componentes principais, nomeadamente para as tarefas de *text classification* e *question-answering,* tendo ainda sido utilizado para os *embeddings* no treino do modelo de *DPR.*

A tarefa de QA é a tarefa de maior importância no protótipo e aquela que mereceu maior atenção e investigação na área. Os resultados obtidos para a tarefa de MRC com os modelos pré-treinados em português foram bastante interessantes sem alterações aos parâmetros *default*, embora com a utilização da componente de IR, á primeira vista, as métricas de *exact match accuracy*, *F1-score, recall* e *precision* para *uni-gram* e *bi-gram,* tenham caído para aproximadamente metade.

Os resultados de MRC são bastante interessantes e motivadores para a realização de um trabalho futuro de *fine-tuning,* quer no treino do modelo de linguagem incorporando vocabulário específico das disciplinas, quer adicionando mais questões de domínio específico ao *dataset* de treino utilizado. Quanto à componente de IR, os resultados e estudo realizado aponta para que o caminho seja na utilização de DPR, no entanto, aqui existe ainda bastante trabalho a realizar com investigação mais profunda na área, assim como na preparação de melhores *datasets* com informação de domínio para *fine-tuning* do modelo utilizado.

Para trabalho futuro, para além da necessidade de melhorar a componente de QA, será necessário aplicar alguma da investigação realizada para concluir o módulo de *Intent Classifier*, implementando todas as classificações de intenções previstas e respetiva ligação à componente de *MediaSearch Query Builder.*

Quanto à capacidade de o sistema produzir respostas às questões colocadas pelo utilizador, a abordagem analisada neste trabalho é apenas uma das abordagens existentes, no entanto é também algo limitativo para determinado tipo de questões. Para permitir que o sistema seja capaz de responder a outro tipo de questões que não sejam apenas contruídas com a extração de uma sequência contínua de termos do contexto dado, existem algumas áreas a investigar no futuro, nomeadamente: analisar mais em detalhe os modelos GPT e a capacidade destes para a geração de respostas com termos não observados no contexto, investigar a área de *text-summarization* em que medida esta poderá ser interessante para utilização nas pesquisas efetuadas pelo utilizador e por último, a criação de uma base de conhecimento através da construção de um grafo de termos cujas ligações e respetivos pesos, sejam obtidos diretamente dos vetores de atenção produzidos durante o treino numa arquitetura de *transformers*. No que concerne a esta última, pretende-se, uma vez criado o referido grafo, avaliar a criação de um motor de inferência para a construção de respostas capazes de relacionar os conceitos.

Referências

Ahn, David , Valentin Jijkoun, Gilad Mishne, Karin Mûller, Maarten de Rijke, Stefan Schlobach, M Voorhees, e L Buckland. 2004. "Using Wikipedia at the TREC QA Track." *TREC.* Citeseer.

Ajitkumar, M., S.A. Khillare, e Namrata C. 2016. "Question Answering System, Approaches and Techniques: A Review." *International Journal of Computer Applications* 34-39.

Aldahwan, Nouf, e Noor Alsaeed. 2020. "Use of Artificial Intelligent in Learning Management System (LMS): A Systematic Literature Review." *International Journal of Computer Applications* 975-8887.

Allam, Ali Mohamed Nabil, e Mohamed Hassan Haggag. 2012. "The Question Answering Systems: A Survey." *International Journal of Research and Reviews in Information Sciences (IJRRIS)* 12.

Auer, Sören, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, e Zachary Ives. 2007. "Dbpedia: A nucleus for a web of open data." Em *The semantic web*, 722-735. Springer.

Bahdanau, Dzmitry, Kyunghyun Cho, e Yoshua Bengio. 2016. "Neural Machine Translation by Jointly Learning to Align and Translate."

Banik, Eva, Eric Kow, Nikhil Dinesh, Vinay Chaudhri, e Umangi Oza. 2012. "Natural Language Generation for a Smart Biology Textbook." *INLG 2012 Proceedings of the Seventh International Natural Language Generation Conference.* 125-127.

Bojanowski, Piotr, Edouard Grave, Armand Joulin, e Tomas Mikolov. 2017. "Enriching Word Vectors with Subword Information." *Transactions of the Association for Computational Linguistics* 135-146.

Bollacker, Kurt , Colin Evans, Praveen Paritosh, Tim Sturge, e Jamie Taylor. 2008. "Freebase: a collaboratively created graph database for structuring human knowledge." *Proceedings of the 2008 ACM SIGMOD international conference on Management of data.* 1247-1250.

Brin, Sergey, e Lawrence Page. 1998. "The Anatomy of a Large-Scale Hypertextual Web Search Engine." *Computer Networks* 107-117.

Chaudhri, Vinay K., Britte Cheng, Adam Overtholtzer, Jeremy Roschelle, Aaron Spaulding, Peter Clark, Mark Greaves, e Dave Gunning. 2013. "Inquire Biology: A Textbook that Answers Questions." *AI Magazine* 55-72.

Chen, Qian, Zhu Zhuo, e Wen Wang. 2019. "BERT for Joint Intent Classification and Slot Filling."

Christopher D., Manning, Raghavan Prabhakar, e Schütze Hinrich. 2008. *Introduction to Information Retrieval.* Cambridge University Press.

Clark, Peter, Bruce Porter, e Boeing Phantom Works. 2004. "Km: the knowledge machine 2.0: Users manual." *Department of Computer Science, University of Texas at Austin.*

Collobert, Ronan, Jason Weston, Leon Bottou, Michael Karlen, Koray Kavukcuoglu, e Pavel Kuksa. 2011. "Natural Language Processing (Almost) from Scratch." *Journal of Machine Learning Research.*

Cortes, Eduardo Gabriel, Vinicius Woloszyn, Dante Barone, Sebastian Möller, e Renata Vieira. 2021. "A systematic review of question answering systems for non-factoid questions." *Journal of Intelligent Information Systems.*

Cutting, D., e J. Pedersen. 1989. "Optimization for dynamic inverted index maintenance." *Proceedings of the 13th annual international ACM SIGIR conference on Research and development in information retrieval.* New York, NY, USA: Association for Computing Machinery. 405-411.

Cutting, Doug. 1999. *Apache Lucene.* Acedido em 2022. https://lucene.apache.org/.

Dauphin, Yann N., Angela Fan, Michael Auli, e David Grangier. 2017. "Language Modeling with Gated Convolutional Networks."

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, e Kristina Toutanova. 2019. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding."

Ellen M, Voorhees. 2002. "Overview of TREC 2002." National Institute of Standards and Technology.

Ester, Martin, Hans-Peter Kriegel, e Xiaowei Xu. 1996. "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise." *second international conference on knowledge discovery & data mining.*

Ferrucci, David, Eric Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya A Kalyanpur, Adam Lally, J William Murdock, Eric Nyberg, e John Prager. 2010. "Building Watson: An overview of the DeepQA project." *AI magazine* 59-79.

Géron, Aurélien. 2020. *Hands-on Machine Learning with Scikit-Learn, Keras & Tensorflow.* O´Reilly.

Goldberg, Yoav . 2017. "Neural network methods for natural language processing." Em *Synthesis Lectures on Human Language Technologies*, 309.

Goo, Chih-Wen, Guang Gao, Yun-Kai Hsu, Chih-Li Huo, Tsung-Chieh Chen, Keng-Wei Hsu, e Yun-Nung Chen. 2018. "Slot-Gated Modeling for Joint Slot Filling and Intent Prediction." *Proceedings of the 2018 Conference of the North American Chapter of the Association*

for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers). New Orleans, Louisiana: Association for Computational Linguistics. 753–757.

Group, Natural Language Computing. 2017. "R-NET: Machine Reading Comprehension with Self-matching Networks."

Guha, R., Rob McCool, e Eric Miller. 2003. "Semantic search." *Proceedings of the 12th international conference on World Wide Web.* New York, NY, USA: Association for Computing Machinery. 700–709.

Guillou, Pierre. 2021. "Portuguese BERT large cased QA (Question Answering), finetuned on SQUAD v1.1."

Harris, Zellig S. 1954. "Distributional Structure." *WORD* 146-162.

Hermann, Karl Moritz, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, e Phil Blunsom. 2015. "Teaching Machines to Read and Comprehend." *Advances in Neural Information Processing Systems.* Curran Associates, Inc.

Hirschman, Lynette, Marc Light, Eric Breck, e John D Burger. 1999. "Deep read: A reading comprehension system." *Proceedings of the 37th annual meeting of the Association for Computational Linguistics.* 325-332.

Hochreiter, Sepp, e Jürgen Schmidhuber. 1997. "Long Short-Term Memory." *Neural Computation* 1735-1780.

Hostetter, Chris. 2006. "Faceted Searching With Apache Solr." *apache.org.* 13 de 10. Acedido em 16 de 01 de 2022. http://people.apache.org/~hossman/apachecon2006us/faceted-searching-with-solr.pdf.

Hu, Baotian, Zhengdong Lu, Hang Li, e Qingcai Chen. 2014. "Convolutional Neural Network Architectures for Matching Natural Language Sentences." *Advances in Neural Information Processing Systems.* Curran Associates, Inc.

IBM. s.d. *IBM.* https://www.ibm.com/cloud/learn/intelligent-search.

Insights, Global Market. 2020. *Global Market Insights.* Acedido em 29 de 01 de 2022. https://www.gminsights.com/industry-analysis/elearning-market-size.

Jurafsky, Daniel, e James H. Martin. 2000. *Speech and Language Processing.* Pearson Education India.

Kaelbling, L. P., M. L. Littman, e A. W. Moore. 1996. "Reinforcement Learning: A Survey." *Journal of Artificial Intelligence Research* 237-285.

Karpukhin, Vladimir, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, e Wen-tau Yih. 2020. "Dense Passage Retrieval for Open-Domain Question Answering."

Kingma, e Diederik P. 2017. “Adam: A Method for Stochastic Optimization.”

Lehnert, Wendy Grace. 1977. *The process of question answering.* Yale University.

Lin, Chin-Yew. 2004. “ROUGE: A Package for Automatic Evaluation of Summaries.” *Text Summarization Branches Out.* Association for Computational Linguistics. 74-81.

Liu, Bing, e Ian Lane. 2016. “Attention-Based Recurrent Neural Network Models for Joint Intent Detection and Slot Filling.”

Liu, Shanshan, Xin Zhang, Sheng Zhang, Hui Wang, e Weiming Zhang. 2019. “Neural Machine Reading Comprehension: Methods and Trends.” *APPLIED SCIENCES-BASEL* 3698.

Liu, Xiaolong, Zhidong Deng, e Yuhan Yang. 2019. “Recent progress in semantic image segmentation.” *Artificial Intelligence Review* 1089-1106.

Maalouf, Maher. 2011. “Logistic regression in data analysis: an overview.” *International Journal of Data Analysis Techniques and Strategies* 281.

Magnini, Bernardo , Simone Romagnoli, Alessandro Vallin, Jesus Herrera, Anselmo Penas, Victor Peinado, Felisa Verdejo, e Maarten Rijke. 2004. “The multiple language question answering track at CLEF 2003.” *Comparative Evaluation of Multilingual Information Access Systems. CLEF 2003.* Berlin Heidelberg: Springer . 471-486.

Manu. 2021. *A simple overview of RNN, LSTM and Attention Mechanism.* 30 de 01. Acedido em 29 de 01 de 2022. https://medium.com/swlh/a-simple-overview-of-rnn-lstm-and-attention-mechanism-9e844763d07b.

Microsoft. 2019. *Bringing the power of machine reading comprehension to specialized documents.* 25 de July. Acedido em 22 de 01 de 2022. https://www.microsoft.com/en-us/research/blog/bringing-the-power-of-machine-reading-comprehension-to-specialized-documents/.

Mikolov, Tomas, Ilya Sutskever, Kai Chen, Greg S Corrado, e Jeff Dean. 2013. “Distributed Representations of Words and Phrases and their Compositionality.” *Advances in Neural Information Processing Systems.* Curran Associates, Inc.

Mikolov, Tomas, Martin Karafiát, Lukas Burget, Jan Cernocký, e Sanjeev Khudanpur. 2010. “Recurrent neural network based language model.” *Proceedings of the 11th Annual Conference of the International Speech Communication Association, INTERSPEECH 2010.* 1045-1048.

Murtagh, Fionn, e Pedro Contreras. 2012. “Algorithms for hierarchical clustering: an overview.” *WIREs Data Mining and Knowledge Discovery* 86-97.

Nasteski, Vladimir. 2017. “An overview of the supervised machine learning methods.” *HORIZONS.B* 51-62.

Neelakandan, Naveen. 2019. *elearningindustry.* 6 de 9. Acedido em 17 de 01 de 2022. https://elearningindustry.com/artificial-intelligence-based-platform-impact-future-elearning.

Nguyen, Tri, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, e Li Deng. 2016. "MS MARCO: A Human Generated MAchine Reading COmprehension Dataset."

Nothman, Joel, Nicky Ringland, Will Radford, Tara Murphy, e James R. Curran. 2013. "Learning multilingual named entity recognition from Wikipedia." *Artificial Intelligence* 151-175.

O'Shea, Keiron, e Ryan Nash. 2015. "An Introduction to Convolutional Neural Networks."

Pascanu, Razvan, Tomas Mikolov, e Yoshua Bengio. 2013. "On the difficulty of training recurrent neural networks." *Proceedings of the 30th International Conference on Machine Learning* 1310-1318.

Pennington, Jeffrey, Richard Socher, e Christopher Manning. 2014. "GloVe: Global Vectors for Word Representation." *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP).* 1532–1543.

Peters, Matthew E., Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, e Luke Zettlemoyer. 2018. "Deep contextualized word representations."

Powell, Matt. s.d. *docebo.* https://www.docebo.com/learning-network/blog/increase-revenue-extended-enterprise-learning-2/.

Radford, Alec, Karthik Narasimhan, Tim Salimans, e Ilya Sutskever. 2018. "Improving Language Understanding by Generative Pre-Training." 12.

Radu, Soricut, e Ding Nan. 2016. "Building Large Machine Reading-Comprehension Datasets using Paragraph Vectors."

Raffel, Colin, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, e Peter J. Liu. 2020. "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer."

Rajpurkar, Pranav, Jian Zhang, Konstantin Lopyrev, e Percy Liang. 2016. "SQuAD: 100,000+ Questions for Machine Comprehension of Text."

Riloff, Ellen, e Michael Thelen. 2000. "A rule-based question answering system for reading comprehension tests." *ANLP-NAACL 2000 Workshop: Reading Comprehension Tests as Evaluation for Computer-Based Language Understanding Systems.*

Robertson. 2004. "Understanding inverse document frequency: on theoretical arguments for IDF." 503-520.

Robertson, Stephen, e Hugo Zaragoza. 2009. “The Probabilistic Relevance Framework: BM25 and Beyond.” *Foundations and Trends in Information Retrieval* 333-389.

Rumelhart, David E., Geoffrey E. Hinton, e Ronald J. Williams. 1986. “Learning representations by back-propagating errors.” 533-536.

Seeley, Yonik. 2006. *Apache Solr.* Acedido em 2022. https://solr.apache.org/.

Sennrich, Rico, Barry Haddow, e Alexandra Birch. 2022. “Neural Machine Translation of Rare Words with Subword Units.”

Seo, Minjoon, Aniruddha Kembhavi, Ali Farhadi, e Hannaneh Hajishirzi. 2016. “Bidirectional Attention Flow for Machine Comprehension.”

Severyn, Aliaksei, e Alessandro Moschitti. 2015. “Twitter Sentiment Analysis with Deep Convolutional Neural Networks.” *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval.* New York, NY, USA: Association for Computing Machinery. 959–962.

Shen, Yelong, Po-Sen Huang, Jianfeng Gao, e Weizhu Chen. 2017. “ReasoNet: Learning to Stop Reading in Machine Comprehension.” *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* 1047-1055.

Simonite, Tom. 2019. *Google Search Now Reads at a Higher Level.* 25 de 10. Acedido em 22 de 01 de 2022. https://www.wired.com/story/google-search-advancing-grade-reading/.

Souza, Fábio, Rodrigo Nogueira, e Roberto Lotufo. 2020. “BERTimbau: Pretrained BERT Models for Brazilian Portuguese.” Em *Intelligent Systems*, 403-417. Springer International Publishing.

Sundermeyer, Martin, Ralf Schlüter, e Hermann Ney. 2012. “LSTM neural networks for language modeling.” *Interspeech 2012.* 194-197.

Taylor, Wilson L. 1953. ““Cloze Procedure”: A New Tool for Measuring Readability.” *Journalism Quarterly* 415-433.

s.d. *Tesseract Git.* Acedido em 09 de 2022. https://github.com/tesseract-ocr/tesseract.

Torrey, Lisa , e Jude Shavlik . 2010. *Transfer Learning.*

Tur, Gokhan, Dilek Hakkani-Tur, e Larry Heck. 2010. “What is left to be understood in ATIS?” *2010 IEEE Spoken Language Technology Workshop.* Berkeley, CA, USA: IEEE. 19-24.

Varian, Hal R. 2006. “The Economics of Internet Search.” *Rivista di politica economica.*

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, e Illia Polosukhin. 2017. “Attention is All you Need.” *Advances in Neural Information Processing Systems.*

Vaughan, Steven. 2017. *Hewlett Packard Enterprise.* 24 de 03. Acedido em 18 de 01 de 2022. https://www.hpe.com/us/en/insights/articles/how-search-worked-before-google-1703.html.

Vinyals, Oriol, Meire Fortunato, e Navdeep Jaitly. 2017. "Pointer Networks."

Wagner, Jorge, Rodrigo Wilkens, Marco Idiart, e Aline Villavicencio. 2018. "The brWaC Corpus: A New Open Resource for Brazilian Portuguese."

Wang, Shuohang, e Jing Jiang. 2016. "Machine Comprehension Using Match-LSTM and Answer Pointer."

Wang, Yu, Yilin Shen, e Hongxia Jin. 2018. "A Bi-model based RNN Semantic Frame Parsing Model for Intent Detection and Slot Filling."

Weld, H., X. Huang, S. Long, J. Poon, e S. C. Han. 2021. "A survey of joint intent detection and slot-filling models in natural language understanding."

s.d. *Wikipedia - Clustering.* Acedido em 2022. https://en.wikipedia.org/wiki/Cluster_analysis#Connectivity-based_clustering_(hierarchical_clustering).

s.d. *Wikipédia.* Acedido em 10 de 2022. https://pt.wikipedia.org/wiki/Clustering.

Wu, Yonghui, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, e Macherey Macherey. 2016. "Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation."

Young, Tom, Devamanyu Hazarika, Soujanya Poria, e Erik Cambria. 2018. "Recent Trends in Deep Learning Based Natural Language Processing [Review Article]." *IEEE Computational Intelligence Magazine* 55-75.

Yu, Adams Wei, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, e Quoc V. Le. 2018. "QANet: Combining Local Convolution with Global Self-Attention for Reading Comprehension."

Zhang, Dell, e Wee Sun Lee. 2003. "Question classification using support vector machines." *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval.* New York, NY, USA: Association for Computing Machinery. 26–32.

Zhang, Lei, Shuai Wang, e Bing Liu. 2018. "Deep learning for sentiment analysis: A survey." *WIREs Data Mining and Knowledge Discovery.*

Zhang, Zhuosheng, Junjie Yang, e Hai Zhao. 2020. "Retrospective Reader for Machine Reading Comprehension."