POLITÉCNICO
DE PORTALEGRE

Escola Superior
de Tecnologia,
Gestão e Design

AVALIAÇÃO DE MODELOS *TRANSFORMERS* BERT DE PERGUNTA/RESPOSTA COM AJUSTE FINO NUM CORPUS EM PORTUGUÊS EUROPEU

Maria Alexandra Esteves Almeida

TRABALHO DE FINAL DE CURSO

Curso: **Mestrado em Informática**

Docente/Orientador: Valentim Alberto Correia Realinho

Ano Letivo: 2022 | 2023

Setembro | 2024

POLITÉCNICO
DE PORTALEGRE

Escola Superior
de Tecnologia,
Gestão e Design

AVALIAÇÃO DE MODELOS *TRANSFORMERS* BERT DE PERGUNTA/RESPOSTA COM AJUSTE FINO NUM CORPUS EM PORTUGUÊS EUROPEU

Maria Alexandra Esteves Almeida

Unidade Curricular: **Projeto ou Estágio**

Docente: Valentim Alberto Correia Realinho

Curso: **Mestrado em Informática**

Ano Letivo: 2022 | 2023

Setembro | 2024

"O sucesso é o produto de hábitos diários – não de transformações que acontecem uma vez na vida"

James Clear, *Hábitos Atómicos*

Agradecimentos

Na realização do presente trabalho contei com algumas pessoas que me apoiaram de forma direta ou indireta às quais estou profundamente grata.

Quero agradecer especialmente ao meu orientador, Professor Valentim Realinho, que me orientou de maneira exemplar e com bastante rigor, demonstrando sempre o seu interesse no trabalho elaborado. Agradeço a disponibilidade e os meios facultados para o seu desenvolvimento. Agradeço também o seu apoio e incentivo na realização deste trabalho.

À Escola Superior de Tecnologia, Gestão e Design de Portalegre e a todo o seu corpo docente pelo conhecimento e ajuda prestados.

Por fim, deixo o meu agradecimento aos meus amigos e família por estarem sempre presentes em todos os momentos, mesmo quando eu estava mais ausente.

Resumo

Este trabalho visa a avaliação e comparação de modelos baseados na arquitetura de *Transformers* para a tarefa de Pergunta/Resposta que foram pré-treinados com dados em língua portuguesa. Foi feito o ajuste fino de três modelos de linguagem baseados no modelo BERTimbau (modelo BERT pré-treinado para português): o bert-base-cased-squad-v1.1-portuguese (SQUAD-PT-BASE); o faquad-bert-base-portuguese-cased (FAQUAD-PT-BASE) e o bert-large-cased-squad-v1.1-portuguese (SQUAD-PT-LARGE). Estes modelos fazem uso da arquitetura *Transformer*, possuindo respetivamente 108 334 082, 108 334 082 e 333 348 866 parâmetros. Para efetuar o ajuste fino, foi construído um *dataset* no formato SQuAD (*Stanford Question Answering Dataset*) com 265 perguntas em português europeu, acerca da Fundação da Nossa Senhora da Esperança. A validação do desempenho dos modelos foi efetuada através das métricas F1 e *accuracy,* tendo o melhor resultado sido alcançado com o modelo SQUAD-PT-BASE com um F1 de 0,91111 e uma *accuracy* de 0,96154. Os valores obtidos permitem concluir que o ajuste fino efetuado aos modelos pré-treinados, conduzem a resultados para tarefas de Pergunta/Resposta bastante satisfatórios, tendo atingido uma performance melhor do que os modelos originais e de outros trabalhos semelhantes.

Palavras-Chave: Inteligência Artificial, *Transformers*, Pergunta/Resposta, Processamento de Linguagem Natural

Abstract

This work aims to evaluate and compare question/answering Transformer models that have been pre-trained with Portuguese data. Fine-tuning was performed on three language models based on BERTimbau model (BERT model pre-trained for Portuguese): bert-base-cased-squad-v1.1-portuguese (SQUAD-PT-BASE); faquad-bert-base-portuguese-cased (FAQUAD-PT-BASE) and bert-large-cased-squad-v1.1-portuguese (SQUAD-PT-LARGE). These models make use of the Transformer architecture and they have 108 334 082, 108 334 082 and 333 348 866 parameters respectively. For the fine-tuning, a dataset in SQuAD (Stanford Question Answering Dataset) format was constructed with 265 questions in European Portuguese about Fundação da Nossa Senhora da Esperança. The comparison was carried out using F1 and accuracy metrics, with the best result being achieved with the SQUAD-PT-BASE model with an F1 of 0,91111 and an accuracy of 0,96154. These results allow us to conclude that the fine-tuning performed on the pre-trained models leads to very good results for the Question/Answering task, having achieved better performance than the original models and other similar works.

Keywords: Artificial Intelligence, Transformers, Question/Answering, Natural Language Processing

Lista de Abreviaturas, Siglas e Símbolos

AI	Artificial Intelligence
AUC	Area Under the Receiving Operating Curve
BERT	Bidirecional Encoder Representations for Transformers
BoW	Bag-of-Words
CSV	Comma-separated values
DeBERTa	Decoding-enhanced BERT with disentangled attention
DPR	Dense Passage Retriever
DT	Decision Trees
EM	Exact Match
FILCO	Filter Context for Retrieval-Augmented Generation
FN	False Negative
FNSE	Fundação da Nossa Senhora da Esperança
FP	False Positive
GBM	Gradient Boosting Machine
GLUE	General Language Understanding Evaluation
GLoVe	Global Vectors for Word Representation
GPU	Graphics Processing Unit
GPT	Generative Pre-Trained Transformer
GRU	Gated Recurrent Unit
HyDE	Hybrid Dense Retrieval
IMDb	Internet Movie DataBase
IR	Information Retrieval
JSON	JavaScript Object Notation
KNN	K-Nearest Neighbors Algorithm
LCs	Learning Curves
LSI	Latent Semantic Indexing
MAE	Mean Absolute Score
MLM	Masked Language Models
NSP	Next Sentence Prediction
NW	Non Words

PCA	Principal Component Analysis
PLM	Permutation Language Modeling
PLN	Processamento de Linguagem Natural
POS	Part-of-Speech Tagging
QA	Question Answering
RF	Random Forest
RLHF	Reinforcement Learning from Human Feedback
RMScrop	Root Mean Square Propagation
RMSE	Root Mean Squared Error
RNN	Recurrent Neural Networds
RoBERTa	Decoding-enhanced BERT with Disentangled Attention
ROC	Receiver Operating Characteristics
SA	Sentiment Analysis
SGD	Stochastic Gradient Descendent
SMS	Short Message Service
SMT	Statistical Automatic Translation
SQUAD	Stanford Question Answering Dataset
SVD	Singular Value Decomposition
SVM	Support Vector Machine
SWAG	Situation With Adversarial Generations
TF-IDF	Term Frequency Inverse Document
XML	Extensible Markup Language

ÍNDICE GERAL

ÍNDICE DE ANEXOS xi

ÍNDICE DE FIGURAS xii

ÍNDICE DE TABELAS xiii

CAPÍTULO I – INTRODUÇÃO 1
1.1 Enquadramento e Justificação do Tema 1
1.2 Objetivos Gerais e Específicos 2
1.3 Estrutura Geral do Trabalho 3

CAPÍTULO II – REVISÃO DA LITERATURA 5
2.1 Compreensão Linguística 5
2.1.1 Análise Fonética 5
2.1.2 Análise Morfológica 6
2.1.3 Análise Sintática 7
2.1.4 Análise Semântica 8
2.1.5 Análise Pragmática e de Discurso 9
2.2 Corpora Linguísticos 9
2.2.1 Língua Portuguesa 10
2.2.2 Língua Inglesa 12
2.3 Tarefas Básicas de Processamento de Linguagem Natural 13
2.3.1 Identificação de Frases e Átomos 14
2.3.2 Lematização e *Stemming* 15
2.3.3 Remoção de *Stop Words* 15
2.3.4 Marcação das Partes do Discurso 15
2.3.5 Desambiguação 16
2.4 Representação de Texto 17
2.4.1 *Bag of Words* 18
2.4.2 *Bag of N-grams* 19
2.4.3 TF-IDF 19
2.4.4 *Word Embeddings* 20
2.4.5 Redução da Dimensionalidade 22
2.5 Aplicações de Processamento de Linguagem Natural 23
2.5.1 Classificação de Texto 23

2.5.2 Análise de Sentimentos 25
2.5.3 Agrupamento de Texto 25
2.5.4 Tradução Automática 26
2.5.5 Correção Automática de Texto 26
2.5.6 Extração de Informação 26
2.5.7 Resumo de Texto 27
2.5.8 Pergunta/Resposta 27
2.6 *Transformers* 27
2.6.1 Atenção 29
2.6.2 Modelos *Transformers* Mais Conhecidos 30
2.7 Métricas de Avaliação de Modelos 34
2.7.1 Métricas de Classificação 34
2.7.2 Métricas de Regressão 37
2.7.3 Função de Perda (*loss*) 38
2.7.4 Curvas de Aprendizagem 39
2.7.5 Avaliação de Modelos de Pergunta/Resposta 42

CAPÍTULO III – DADOS E METODOLOGIA 45
3.1 Dados 45
3.2 Metodologia 50

CAPÍTULO IV – RESULTADOS OBTIDOS 57
4.1 Modelo SQUAD-PT-BASE 60
4.2 Modelo FAQUAD-PT-BASE 61
4.3 Modelo SQUAD-PT-LARGE 64
4.4 Avaliação em Dados Desconhecidos 66

CAPÍTULO V - DISCUSSÃO DOS RESULTADOS OBTIDOS 69
5.1 Comparação das Curvas de Aprendizagem Obtidas nos Treinos 69
5.2 Comparação dos Resultados Obtidos com Outros Treinos 71

CAPÍTULO VI - CONCLUSÃO 75
6.1 Síntese e Contribuições 75
6.2 Conclusões 76
6.3 Trabalho Futuro 76

REFERÊNCIAS BIBLIOGRÁFICA 79

ANEXOS 87
ANEXO 1 – Serviços de análise de frases (Lx-Parser, Lx-DepParser e Tags usadas) 87
ANEXO 2 – Curvas de Aprendizagem (SQUAD-PT-BASE, learning_rate=1e-04) 94
ANEXO 3 – Curvas de Aprendizagem (SQUAD-PT-BASE, learning_rate=1e-05) 95
ANEXO 4 – Curvas de Aprendizagem (SQUAD-PT-BASE, learning_rate=2e-05) 96
ANEXO 5 – Curvas de Aprendizagem (FAQUAD-PT-BASE, learning_rate=1e-04) 97
ANEXO 6 – Curvas de Aprendizagem (FAQUAD-PT-BASE, learning_rate=1e-05) 98
ANEXO 7 – Curvas de Aprendizagem (FAQUAD-PT-BASE, learning_rate=2e-05) 99
ANEXO 8 – Curvas de Aprendizagem (SQUAD-PT-LARGE, learning_rate=1e-04) 100
ANEXO 9 – Curvas de Aprendizagem (SQUAD-PT-LARGE, learning_rate=1e-05) 101
ANEXO 10 – Curvas de Aprendizagem (SQUAD-PT-LARGE, learning_rate=2e-05) 102

ÍNDICE DE ANEXOS

ANEXO 1 – Serviços de análise de frases (Lx-Parser, Lx-DepParser e Tags usadas) 87
ANEXO 2 – Curvas de Aprendizagem (SQUAD-PT-BASE, learning_rate=1e-04) 94
ANEXO 3 – Curvas de Aprendizagem (SQUAD-PT-BASE, learning_rate=1e-05) 95
ANEXO 4 – Curvas de Aprendizagem (SQUAD-PT-BASE, learning_rate=2e-05) 96
ANEXO 5 – Curvas de Aprendizagem (FAQUAD-PT-BASE, learning_rate=1e-04) 97
ANEXO 6 – Curvas de Aprendizagem (FAQUAD-PT-BASE, learning_rate=1e-05) 98
ANEXO 7 – Curvas de Aprendizagem (FAQUAD-PT-BASE, learning_rate=2e-05) 99
ANEXO 8 – Curvas de Aprendizagem (SQUAD-PT-LARGE, learning_rate=1e-04) 100
ANEXO 9 – Curvas de Aprendizagem (SQUAD-PT-LARGE, learning_rate=1e-05) 101
ANEXO 10 – Curvas de Aprendizagem (SQUAD-PT-LARGE, learning_rate=2e-05) 102

ÍNDICE DE FIGURAS

Figura 1. Árvore de Constituência da frase "*A menina procura o brinquedo.*" 8
Figura 2. Árvore de Dependência da frase "*A menina procura o brinquedo.*" 8
Figura 3 - Estrutura do SQuAD 13
Figura 4. Árvores de constituência da frase "*Eu vi o João com o binóculo*" 17
Figura 5. Arquiteturas Word2Vec 21
Figura 6 – Arquitetura dos *Transformers* 28
Figura 7 – Tipos de funções de atenção 29
Figura 8 - Matriz de confusão 34
Figura 9 - Representação gráfica das métricas ROC e AUC 37
Figura 10 - Exemplo de Overfitting 40
Figura 11 - Exemplo de underfitting 41
Figura 12 - Exemplo de Good Fit 42
Figura 13 - Exemplo de uma pergunta do dataset no formato SQuAD 46
Figura 14 - Distribuição do número de palavras dos contextos, perguntas e respostas (colunas `context`, `question` e `answer`) 47
Figura 15 - Distribuição das perguntas por categoria 48
Figura 16 – Nuvem de palavras do contexto 49
Figura 17 – Nuvem de palavras das perguntas 49
Figura 18 – Nuvem de palavras das respostas 50
Figura 19 - Workflow usado para treino e validação dos modelos 52
Figura 20 - Fluxo para a Previsão com Dados Desconhecidos 56
Figura 21 - Evolução das métricas ao longo das épocas para os melhor resultado (SQUAD-PT-BASE) 61
Figura 22 - Evolução das métricas ao longo das épocas para o melhor resultado (FAQUAD-PT-BASE) 63
Figura 23 - Evolução das métricas ao longo das épocas para o melhor resultado (SQUAD-PT-LARGE) 65
Figura 24 - Exemplo de interação com o utilizador para criação de perguntas 66
Figura 25 - Curvas de aprendizagem para as métricas *loss*, *accuracy* e F1 para os melhores resultados de cada modelo 69

ÍNDICE DE TABELAS

Tabela 1 - Análise Morfossintática da frase "*A menina procura o brinquedo*" 7
Tabela 2 – POS *Tagging* da frase "*A menina procura o brinquedo no quarto.*" 16
Tabela 3 - Bag-of-Words (Exemplo) 18
Tabela 4 - Bag of n-grams (Exemplo) 19
Tabela 5 - TF-IDF (Exemplo) 20
Tabela 6 - Família de Transformers GPT 32
Tabela 7 - Análise estatística do dataset da FNSE 46
Tabela 8 - Modelos pré-treinados utilizados 51
Tabela 9 - Hiperparâmetros utilizados no treino 54
Tabela 10 - Dez primeiros registos do treino dos modelos 59
Tabela 11 - Melhores resultados da validação do modelo SQUAD-PT-BASE 60
Tabela 12 - Melhores resultados da validação do modelo FAQUAD-PT-BASE 62
Tabela 13 - Melhores resultados da validação do modelo SQUAD-PT-LARGE 65
Tabela 14 - Resultados médios obtidos na avaliação com dados desconhecidos 67
Tabela 15 - Resultados de métricas obtidos na avaliação de modelos de Pergunta/Resposta 72
Tabela 16 - Tags lexicais e sintagmáticas 87
Tabela 17 - Tags Multipalavra 89
Tabela 18 – Lx-Parser tagger 90
Tabela 19 – Conjunto de tags de função gramatical 91
Tabela 20 - Conjunto de tags para categorias nominais 92
Tabela 21 - Conjunto de tags para verbos 92
Tabela 22 - Conjunto de Tags para verbos Infinitivos 93

CAPÍTULO I – INTRODUÇÃO

O presente capítulo efetua o enquadramento e justificação do tema deste trabalho, contextualizando a importância crescente dos *Transformers* no Processamento de Linguagem Natural (PLN). São ainda apresentados, neste capítulo, os objetivos gerais e específicos, bem como a organização geral do trabalho.

1.1 Enquadramento e Justificação do Tema

Os *Transformers* são uma arquitetura muito usada em modelos de Processamento de Linguagem Natural e possuem a capacidade de modelar relações contextuais complexas em textos longos. Esta arquitetura é baseada num mecanismo de atenção, que captura as dependências em sequências de entrada, analisando a importância de cada palavra, o que permite que os modelos se concentrem em zonas específicas do processamento de texto (Vaswani et al., 2017). Os *Transformers* ganharam grande popularidade e são, atualmente, bastante utilizados numa grande variedade de tarefas de PLN, tais como a resposta a perguntas (mais conhecida por tarefa de Pergunta/Resposta), a tradução automática ou a classificação de textos, entre muitas outras.

Estes modelos, são treinados em grandes corpora generalistas com uma elevada quantidade de informação, que na maior parte dos casos, se encontram em língua inglesa, ou na melhor das hipóteses, em português do Brasil. Para um modelo conseguir obter melhor desempenho numa determinada tarefa, é necessário efetuar o ajuste fino através do treino com um corpus específico para a tarefa em questão (Howard & Ruder, 2018).

O foco deste trabalho é a utilização da arquitetura dos *Transformers* para tarefas de Pergunta/Resposta, em que, os modelos são capazes de encontrar respostas num contexto fornecido obtido pela recuperação de informação existente em documentos ou páginas *web*. Este tipo de tarefa apresenta desafios significativos, especialmente em cenários onde a compreensão semântica profunda e o contexto são fundamentais para fornecer respostas precisas. Os *Transformers* vieram trazer uma nova perspetiva para lidar com esses desafios, sendo capazes de entender o contexto de forma mais ampla e

gerar respostas mais precisas e contextualmente adequadas (Vaswani et al., 2017). Desta forma, pretende-se entender o Processamento da Linguagem Natural em Português e avaliar os modelos existentes, para a execução da tarefa de Pergunta/Resposta.

Para a realização deste trabalho foi contruído um *dataset* com 265 perguntas, as suas respetivas respostas e contextos o qual foi utilizado para o treino de três modelos de Pergunta/Resposta, baseados no modelo *BERTimbau* (Guillou, 2021b). Os treinos realizados foram avaliados e comparados com o objetivo de se perceber qual é o melhor na geração das respostas. O *dataset* construído foi obtido pela extração da informação da página *web* da Fundação da Nossa Senhora da Esperança, e tem por objetivo, a resposta a perguntas acerca da instituição.

A realização deste trabalho é relevante e justifica-se, pela necessidade de se avançar no desenvolvimento de modelos que compreendam e processem linguagem natural em português europeu, para tarefas de Pergunta/Resposta com maior precisão.

1.2 Objetivos Gerais e Específicos

O objetivo geral deste trabalho é a avaliação de modelos baseados na arquitetura de *Transformers* para tarefas de Pergunta/Resposta, que foram pré-treinados com dados em língua portuguesa, a fim de se verificar qual deles é o melhor no que toca à geração de respostas a perguntas, tendo por base um contexto fornecido. Desta forma, devem ser selecionados modelos pré-treinados em língua portuguesa que tenham demonstrado bons resultados.

No que toca a objetivos específicos, podem-se citar os que se seguem, e que constituem as etapas deste trabalho:

1. Entendimento das tarefas de Processamento de Linguagem Natural e aplicações do processamento de Linguagem Natural;
2. Recolha de dados e criação de um *dataset* em português europeu, o qual irá servir para se efetuar o ajuste fino dos modelos de *Transformer* selecionados;
3. Efetuar o ajuste fino dos modelos selecionados e respetiva validação/avaliação;

4. Discussão dos resultados obtidos com os treinos e com a avaliação com dados desconhecidos.

1.3 Estrutura Geral do Trabalho

Este trabalho encontra-se dividido em seis capítulos. O primeiro capítulo foca-se na introdução e enquadramento do tema e nos seus objetivos. O segundo capítulo, efetua uma revisão da literatura, começando por abordar conceitos relativos à compreensão linguística e aos corpora linguísticos disponíveis, quer em português, quer em inglês. Ainda neste capítulo, são explicadas as tarefas básicas de Processamento da Linguagem Natural, as várias formas de representação de texto e as diversas aplicações de PLN. Por fim, é apresentada a arquitetura de *Transformers* e as métricas de avaliação dos modelos. No terceiro capítulo são apresentados os dados e a metodologia utilizados neste trabalho. É explicado como foi criado o *dataset* utilizado para efetuar o ajuste fino dos modelos, sendo de seguida, apresentada as etapas da metodologia utilizada neste trabalho. No quarto capítulo são mostrados os resultados obtidos pelos treinos realizados e para a avaliação do melhor modelo com dados desconhecidos. No quinto capítulo é feita a discussão dos resultados dos treinos e na avaliação com dados desconhecidos. Finalmente, no sexto capítulo são apresentas as conclusões deste trabalho e aponta direções relativas ao trabalho futuro.

CAPÍTULO II – REVISÃO DA LITERATURA

Este capítulo apresenta uma revisão dos conceitos e estudos mais relevantes para a realização e compreensão deste trabalho. Começa por introduzir os conceitos fundamentais da compreensão linguística e de exemplos de corpora linguísticos de domínio público, tanto para a língua portuguesa como para a língua inglesa. De seguida, são descritas as tarefas mais comuns no Processamento de Linguagem Natural bem como os modelos de representação de texto mais utilizados e as suas aplicações. Finalmente, é abordada a arquitetura dos *Transformers* e as métricas utilizadas na avaliação da performance de modelos de linguagem.

2.1 Compreensão Linguística

A compreensão linguística refere-se à capacidade humana ou de um sistema em entender e interpretar a linguagem escrita ou falada. Envolve a prática da decomposição da língua com recurso a vários níveis de análise, os quais são descritos de seguida.

2.1.1 Análise Fonética

A análise fonética é o ramo da linguística que estuda os sons produzidos pela fala humana onde é feita a divisão do discurso em fonemas, que podem ser classificados em vogais, semivogais ou consoantes. Um fonema, (do grego "*fono"* que significa "*som*" e "*emas"* que significa "unidades distintas") corresponde à unidade mínima com valor distintivo do sistema fonológico, que constitui a lista de fonemas que compõem o alfabeto fonético de uma língua.

A fonologia e a fonética são áreas diferentes, na medida em que, a fonologia apenas tem em conta as variações dialetais se houver impacto na diferença de significado nas palavras, e a fonética tem em conta as diversas formas como os falantes das várias regiões articulam as palavras, os sotaques.

A análise fonética torna-se relevante no Processamento de Linguagem Natural (PLN), em particular em sistemas de interação por voz, quer ao nível do reconhecimento da fala, quer ao nível da síntese de voz.

A ambiguidade fonética está presente na análise fonética em palavras homófonas, ou seja, palavras com pronúncias idênticas. Estas palavras, apesar de terem um som igual, têm significados diferentes. No vocabulário português pode-se exemplificar as palavras "*cozer"* e "*coser"* ou "*houve"* e "*ouve"*. A existência deste tipo de palavras é um desafio na análise fonética.

Assim, a análise fonética constitui a análise de um fonema no que diz respeito à classificação dos fonemas, ao número de sílabas e acento tónico de palavras.

2.1.2 Análise Morfológica

A morfologia (do grego "*morphe"* que significa forma e "*logia*" que significa estudo), consiste na área da linguística que é capaz de descrever e analisar a estrutura interna da palavra e as formas de variação da mesma. A morfologia tem como objeto de estudo a palavra (morfema, que é a unidade mais pequena para análise). A análise morfológica estuda as palavras de forma isolada, classificando e extraindo a classe gramatical (definir se uma palavra é um nome, adjetivo, verbo, determinante, pronome, advérbio, preposição, conjunção ou interjeição), o lema e o radical.

Ao se analisar a estrutura de uma palavra, deve-se ter conta os processos de flexão, derivação e composição, que são os mais comuns. A flexão é a forma que uma palavra pode assumir sem que existam modificações na categoria sintática, sendo que, os nomes, adjetivos, verbos e certos determinantes e pronomes são afetados pela flexão, em número ou género, para exemplificar, o nome "*aluno*" que flexionado em número fica como "*alunos*", e em género como "*aluna*". A derivação faz com que a variação efetuada pelos morfemas conduza à modificação da categoria sintática da palavra ou do significado da mesma, para exemplificar, a palavra "*livraria*" que deriva de "*livro*" sendo que, neste caso, a categoria sintática não é alterada, apenas o significado. A composição cria palavras através da junção de palavras, como por exemplo, "*porta-chaves*" que junta as palavras "*porta*" e "*chaves*" ou a palavra "*Girassol*" que junta as palavras "*gira*" e "*sol*".

Se considerarmos a frase "*A menina procura o brinquedo.*", esta pode ser decomposta morfologicamente nos respetivos lemas, flexões e classes tal como ilustrado na Tabela

1. Esta decomposição foi elaborada com o PORTULAN CLARIN LX-Suite[1] o qual permite anotar morfologicamente frases usando o conjunto de etiquetas constantes no Anexo 1. Neste exemplo, são utilizadas as etiquetas que definem a classe gramatical (DA - artigo definido, CN - nome comum, V - verbo e, PNT - pontuação) e a flexão (fs - feminino singular, pi-3s - presente do indicativo, 3ª pessoa do singular e, ms - masculino singular) de cada uma das palavras da frase.

Tabela 1 - Análise Morfossintática da frase "*A menina procura o brinquedo.*"

	A	menina	Procura	o	brinquedo	.
Lema		MENINO	PROCURAR		BRINQUEDO	
Flexão	Fs	Fs	pi-3s	ms	ms	
Classe	DA	CN	V	DA	CN	PNT

Fonte: Elaboração própria a partir do PORTULAN CLARIN LX-Suite

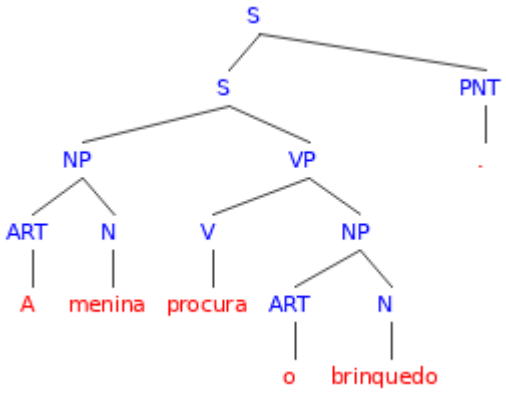
2.1.3 Análise Sintática

A análise sintática estuda o relacionamento entre as palavras numa frase, descrevendo as relações que as mesmas estabelecem entre si. Também pode ser considerada como o ramo da lógica que descreve as regras que determinam as relações das expressões.

A análise sintática permite a construção de árvores de constituência e de dependência. Uma árvore de constituência é uma estrutura de dados em árvore que permite a representação da estrutura sintática de acordo com a análise gramatical, enquanto a árvore de dependência representa apenas a relação entre as palavras. Tomando como exemplo a frase anterior, "*A menina procura o brinquedo.*", a árvore de constituência pode ser representada como na Figura 1, enquanto a árvore de dependências como representado na Figura 2. As etiquetas utilizadas neste exemplo podem ser consultadas no Anexo 1.

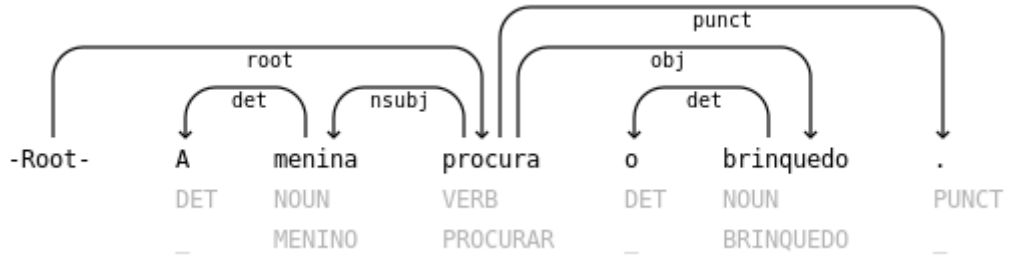
[1] https://portulanclarin.net/workbench/lx-suite/

Figura 1. Árvore de Constituência da frase "*A menina procura o brinquedo.*"



Fonte: Elaborado com o PORTULAN CLARIN LX-Parser[2]

Figura 2. Árvore de Dependência da frase "*A menina procura o brinquedo.*"



Fonte: Elaborado com o PORTULAN CLARIN LX-DepParser[3]

A ambiguidade estrutural da linguagem (mais de uma maneira de analisar a sintaxe de uma frase) conduz a árvores de constituência e de dependência distintas, em que, todas com interpretações são válidas. Lidar com este tipo de ambiguidade requer técnicas para determinar a interpretação mais provável de um determinado contexto.

2.1.4 Análise Semântica

A análise semântica pretende clarificar o significado das palavras e das frases num determinado texto. Enquanto a análise sintática se preocupa com a estrutura gramatical da frase, a análise semântica vai mais além, tentando extrair o significado mais profundo e contextual das expressões linguísticas.

[2] https://portulanclarin.net/workbench/lx-parser/
[3] https://portulanclarin.net/workbench/lx-depparser/

A análise semântica envolve, entre outros, a determinação de conceitos semânticos como a sinonímia (palavras com significado semelhante), a antonímia (palavras com significado oposto ou contrário), a hiperonímia (palavras que representam uma categoria mais ampla ou genérica), hiponímia (palavras que se enquadram dentro de uma definição mais ampla) ou, a polissemia (palavras com vários significados).

A análise semântica envolve também a determinação do uso de figuras de linguagem como a metáfora (comparação simbólica entre palavras distintas) e a metonímia (substituição de uma palavra por outra que está relacionada com ela por proximidade ou associação), as quais representam desafios importantes no Processamento de Linguagem Natural.

A ambiguidade lexical está presente na análise semântica, por exemplo, sempre que se faz uso da polissemia ou de metáforas. Como exemplo, a palavra "*dó*", tanto pode significar tristeza ou compaixão, ou a nota dó da pauta musical. De igual forma, a expressão "*cabelos de oiro*" representa uma metáfora com sentido diferente daquele que as palavras separadas habitualmente possuem. A análise semântica tenta, desta forma, determinar o significado correto com base no contexto da palavra.

2.1.5 Análise Pragmática e de Discurso

A análise pragmática permite determinar a relação entre a linguagem e o contexto. A pragmática é a área da linguística que verifica o uso concreto da linguagem pelos falantes de uma determinada língua, tendo em conta os contextos. Para a pragmática, o importante é a forma como algo é falado e o impacto cultural e social que um diálogo provoca. Esta área engloba os atos de fala e a comunicação não-verbal.

Nesta área, avalia-se as emoções que um diálogo pode gerar de acordo com a pontuação de um texto escrito ou com a forma como o falante fala. Para exemplificar, na frase "*Não queremos saber!*" a simples colocação de uma virgula altera completamente o sentido da frase: "*Não, queremos saber!*".

2.2 Corpora Linguísticos

Os corpora linguísticos são coleções organizadas de documentos numa ou mais línguas, usados para estudos linguísticos, análises computacionais e treino de modelos de

linguagem para vários tipos de tarefas. São apresentados de seguida, diversos corpora linguísticos tanto para a língua portuguesa, como para a língua inglesa.

2.2.1 Língua Portuguesa

- **CETEMPúblico**[4]: corpus em português europeu que inclui o texto de cerca 2 600 edições do jornal Público[5], entre os anos de 1991 e 1998, num total de aproximadamente 180 milhões de palavras (Rocha & Santos, 2000). Está dividido em 1 567 625 extratos, classificados por semestre e secção do jornal da qual provêm. Cada extrato está dividido em parágrafos e frases e tem assinalado os títulos e os autores dos artigos. É disponibilizado em XML (*Extensible Markup Languag*e) de acordo com um esquema próprio (*XML Schema Definition*).
- **CETENFolha**[6]**:** corpus em português do Brasil com cerca de 24 milhões de palavras que tem por base o Jornal Folha de São Paulo[7] do ano de 1994. Está anotado pelo analisador PALAVRAS (Bick, 2000) e dividido em 340 947 extratos classificados por semestre e caderno do jornal.
- **CHAVE**[8]: corpus que contém textos jornalísticos do jornal Público (português europeu) e do jornal Folha de São Paulo (português do Brasil) dos anos de 1994 e 1995 (D. Santos & Rocha, 2005). Estes textos contêm anotações sintáticas geradas automaticamente pelo analisador PALAVRAS e anotações das entidades mencionadas geradas pelo REMBRANDT (Cardoso, 2008).
- **Floresta Sintá(c)tica**[9]: corpus com textos em português europeu e do Brasil com anotações morfossintáticas criadas automaticamente pelo analisador sintático PALAVRAS (Bick, 2000). Serve para treino e avaliação de analisadores morfossintáticos, tendo parte destas anotações sido revistas por peritos linguistas. Está dividido em quatro partes que diferem quanto ao género de

[4] https://www.linguateca.pt/CETEMPublico/
[5] https://www.publico.pt/
[6] https://www.linguateca.pt/cetenfolha/index_info.html
[7] https://www.folha.uol.com.br/
[8] https://www.linguateca.pt/ACDC/
[9] https://www.linguateca.pt/Floresta/levantamento.html

texto, ao modo (escrito ou falado) e quanto ao nível de revisão por peritos linguistas encontrando-se em constante atualização. Essas partes são:

- **Bosque:** é composto por 9 368 frases retiradas dos primeiros extratos do CETENFolha e do CETEMPúblico. Foi totalmente revisto por linguistas e corresponde à parte mais correta da Floresta Sintá(c)tica e, por isso, a mais aconselhada para utilizações, em que não se dá tanta prioridade à quantidade, mas sim à precisão dos resultados.
- **Floresta Virgem**: corresponde ao resto do primeiro milhão de palavras respetivamente do CETENFolha e do CETEMPúblico. A Floresta Virgem não contém as frases pertencentes ao Bosque, visto que, é complementar a este, e a intenção é ir-se aumentando o Bosque e, consequentemente, diminuir a Floresta Virgem.
- **Selva**: contém cerca de 300 000 palavras divididas entre diferentes géneros textuais e as variantes portuguesa e brasileira do português. Subdivide-se em três partes: (i) Selva Falada composta pela transcrição de dois tipos de fala relativos a entrevistas e debates parlamentares; (ii) Selva Literária composta por textos literários do final do século XIX e do início do século XX e, ainda textos contemporâneos e, (iii) Selva Científica composta por capítulos de teses brasileiras e portuguesas.
- **Amazônia**: contém 4.6 milhões de palavras (cerca de 275 mil frases), em português do Brasil, retiradas do sítio colaborativo Overmundo[10], um coletivo virtual que tem como objetivo expressar a produção cultural brasileira.

- **BDCamões[11]**: corpus constituído por uma coleção de documentos literários escritos em português, em formato .txt, com cerca de 4 milhões de palavras provenientes de mais de 200 documentos completos de 83 autores em 14 géneros, cobrindo um período do século XV ao século XXI e, aderindo a diferentes convenções ortográficas (Grilo et al., 2020). Muitos textos deste

[10] http://www.overmundo.com.br/
[11] https://hdl.handle.net/21.11129/0000-000D-F89B-D

corpus estão anotados automaticamente com ferramentas de processamento de linguagem de última geração. Este corpus inclui obras de romance, poesia e crónicas marcantes da cultura portuguesa (como textos de Luís Vaz de Camões ou Fernando Pessoa) e pode ser usado no desenvolvimento de ferramentas de processamento de autoria, classificação de género, verificação gramatical, conversão da ortografia, entre muitos outros.

2.2.2 Língua Inglesa

A língua inglesa tem disponível muitos mais corpora linguísticos. O site English-Corpora.org[12] é um bom recurso, pois reúne um vasto número de corpora linguísticos em língua inglesa, alguns dos quais muito utilizados em PLN, sendo um bom ponto de partida para a procura de corpora para treino de modelos.

De seguida, iremos enumerar outros que pela sua ampla utilização merecem destaque.

- **The Wikipedia Corpus[13]**: contém o texto completo da Wikipédia possuindo cerca de 2 biliões de palavras existentes em mais de 4.4 milhões de artigos. Permite a criação de corpora menores através de filtragem pelos termos desejados.
- **The Twitter Sentiment Analysis Dataset[14]**: corpus para tarefas de análise de sentimento contendo 1 578 627 tweets, cada um classificado como sentimento positivo ou sentimento negativo.
- **The Movie Corpus[15]**: este corpus contém 200 milhões de palavras recolhidas de mais de 25 000 filmes, desde a década de 1930 até à atualidade. Todos os filmes estão vinculados à sua entrada no IMDb[16]. Serve como um excelente recurso de texto em linguagem informal falada.

[12] https://www.english-corpora.org/
[13] https://www.english-corpora.org/wiki/
[14] https://huggingface.co/datasets/carblacac/twitter-sentiment-analysis
[15] https://www.english-corpora.org/movies/
[16] https://www.imdb.com/

- **Movie Review**[17]: este corpus contém avaliações de filmes, categorizadas com sentimento positivo ou negativo. É geralmente utilizado para tarefas de análise de sentimentos e classificação de texto.
- **SQuAD**[18]: o *Stanford Question Answering Dataset* (SQuAD) é um conjunto de dados que consiste em perguntas feitas por *crowdworkers* num conjunto de artigos da Wikipedia, onde a resposta a cada pergunta corresponde a um segmento de texto (Rajpurkar et al., 2016). Este *dataset* é bastante utilizado para treino de modelos em tarefas de Pergunta/Resposta, por ser de grande dimensão, com mais de 100 000 perguntas formuladas de acordo com um tema específico, contendo a respetiva resposta e contexto associado. Existem versões deste corpus obtidas através da tradução para várias línguas incluindo o português do Brasil. O formato original do *dataset* é JSON, ilustrando-se na Figura 3 um exemplo de uma pergunta.

Figura 3 - Estrutura do SQuAD

```
{
  "id": "56df6bf35ca0a614008f99ff",
  "title": "Christian",
  "context": "In the past, the Malays used to call the Portuguese Serani from
the Arabic Nasrani, but the term now refers to the modern Kristang creoles of
Malaysia.",
  "question": "What term did the Malays use for the Portuguese Serani?",
  "answers": {
    "text": [
      "Nasrani"
    ],
    "answer_start": [
      75
    ]
  }
}
```

Fonte: Elaboração própria

2.3 Tarefas Básicas de Processamento de Linguagem Natural

O Processamento de Linguagem Natural envolve um conjunto de tarefas que permite efetuar as análises linguísticas descritas anteriormente, bem como a utilização de

[17] http://www.cs.cornell.edu/people/pabo/movie-review-data/
[18] https://rajpurkar.github.io/SQuAD-explorer/

modelos de *Machine* e *Deep Learning*. De seguida, são descritas as tarefas mais comuns de PLN.

2.3.1 Identificação de Frases e Átomos

A tarefa de separar palavras num texto possui o nome de atomização (*tokenization* em inglês) e cada unidade resultante dessa tarefa é designada de átomo (*token* em inglês). Esta tarefa, em línguas como inglês, português e muitas outras, é efetuada com base nos espaços existentes entre as palavras. No entanto, nem todas as línguas constroem as suas frases desta forma, como é o caso do japonês, que não forma espaços nos textos, o que torna o processo mais complexo, sendo necessário separar pelos próprios caracteres presentes.

Mesmo em línguas como o português e o inglês, o processo de atomização não é simples na separação de frases, uma vez que, não se podem considerar os pontos como separador, pois estes podem ocorrer no meio de frases, como em palavras como *"Sr."* ou *"Dr."*. Por outro lado, por vezes não são os pontos que terminam as frases, mas sim, outra pontuação, como é o caso dos pontos de exclamação ou de interrogação, que também tem de ser considerados.

Nem todos os constituintes de um texto são considerados na atomização, por vezes, não se considera a pontuação existente, que é essencial para o entendimento pelo utilizador humano, mas que pode acrescentar informação em determinadas aplicações de PLN. As expressões numéricas representam também uma dificuldade para os algoritmos, já que, na generalidade das vezes, as vírgulas e os pontos ocorrem dentro dos números.

Para exemplificar a tarefa de atomização, consideremos o seguinte texto:

"Fui ao supermercado às compras. Encontrei lá a Luísa."

A divisão do texto nas respetivas frases, resulta, neste caso, em:

[*"Fui ao supermercado às compras.", "Encontrei lá a Luísa."*]

Enquanto a atomização resulta em:

[*"Fui", "ao", "supermercado", "às", "compras", ".", "Encontrei", "lá", "a", "Luísa", "."*]

2.3.2 Lematização e *Stemming*

A Lematização (em inglês *Lemmatization*) é usada em PLN para normalizar palavras e reduzir a dimensionalidade do texto, facilitando análises linguísticas e recuperação de informação (Porter, 1980). Tem por objetivo determinar o lema de uma palavra, ou seja, a sua forma canónica que representa a sua raiz ou forma lexical. A Lematização identifica, por exemplo, a flexão em género e verbal identificando que uma palavra flexionada provém de uma determinada palavra raiz. Por exemplo, o lema da palavra "*correndo*" é "*correr*", da palavra "*cavalos*" é "*cavalo*", e da palavra "*menina*" é "*menino*".

Ao contrário da Lematização, o *Stemming* não resulta necessariamente em palavras válidas no vocabulário, mas sim, em formas cortadas ou truncadas que representam a raiz comum de várias palavras relacionadas. Por exemplo, o *Stemming* de "*correndo*", "*correu*" e "*correrá*" resulta em "*corr-*", que é a forma comum dessas palavras. Tal como a Lematização, o *Stemming* é usado em PLN para normalizar o texto e reduzir a dimensionalidade. Este processo não altera o significado das palavras.

2.3.3 Remoção de *Stop Words*

As *Stop Words* são palavras frequentes num texto, como por exemplo, artigos, preposições, conjunções e outros termos que, geralmente, não contribuem muito para o significado semântico. São, habitualmente, filtradas ou removidas durante o pré-processamento de texto em PLN. Contudo, importa referir que a remoção de *Stop Words* nem sempre é necessária ou desejável, dependendo do contexto da análise, pois em alguns casos, as *Stop Words* podem conter informação importante, especialmente em tarefas como a Análise de Sentimentos ou Deteção de Autoria (Jurafsky & Martin, 2023). Exemplos comuns de *Stop Words* em português, incluem os determinantes "*o", "a", "os", "as"*, as preposições "*com", "de", "desde", "sem"* e os pronomes "*deles", "delas", "seus", "suas"*, entre muitos outros.

2.3.4 Marcação das Partes do Discurso

A Marcação das Partes do Discurso (*Part-of-Speech Tagging* ou simplesmente POS *Tagging*) consiste em identificar e rotular as palavras de um texto com etiquetas que representam a sua classe gramatical ou categoria de palavra correspondente. As

categorias de palavras marcadas incluem geralmente substantivos, verbos, adjetivos, advérbios, pronomes, preposições, conjunções, entre outras.

Esta tarefa é fundamental em PLN, pois fornece informação essencial sobre a estrutura gramatical e o significado das palavras num texto, facilitando análises linguísticas, traduções automáticas e reconhecimento de voz ou recuperação de informação.

A Tabela 2 ilustra o POS *Tagging* para a frase "*A menina procura o brinquedo no quarto.*" obtido com o PORTULAN CLARIN LX-Suite[19].

Tabela 2 – POS *Tagging* da frase "*A menina procura o brinquedo no quarto.*"

Token	POS	Descrição
A	DA	Artigo definido
Menina	CN	Nome comum
Procura	V	Verbo
O	DA	Artigo definido
Brinquedo	CN	Nome comum
Em	PREP	Preposição
O	DA	Artigo definido
Quarto	CN	Nome comum
.	PNT	Pontuação

Fonte: Elaboração própria

2.3.5 Desambiguação

A desambiguação refere-se ao processo de resolver ambiguidades que surgem quando uma palavra ou frase pode ter múltiplos significados ou interpretações num determinado contexto. Como referido anteriormente, as ambiguidades podem surgir com a utilização de alguns conceitos semânticos como a polissemia (uma palavra com múltiplos significados) ou a sinonímia (palavras com significado semelhante) e, ainda quando existe mais de uma maneira de analisar a sintaxe de uma frase (ambiguidade

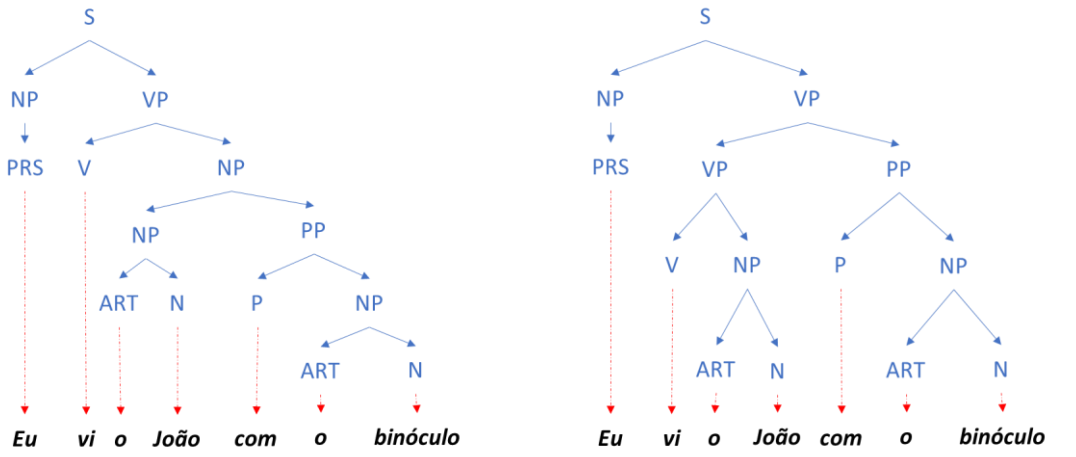
[19] https://portulanclarin.net/workbench/lx-suite/

estrutural), que conduz a árvores de constituência e de dependência distintas todas com interpretações válidas. Como exemplo, consideremos a frase:

"Eu vi o João com o binóculo"

Esta frase tanto pode tomar o sentido de "eu vi o João e ele tinha o binóculo", como "eu usei o binóculo para ver João", o que conduz a duas árvores de constituência distintas, mas ambas válidas, tal como ilustrado na Figura 4.

Figura 4. Árvores de constituência da frase "*Eu vi o João com o binóculo*"



a) sentido: eu vi o João e ele tinha o binóculo (junção do PP ao NP)

b) sentido: eu usei o binóculo para ver o João (junção do PP ao VP)

Fonte: Elaboração própria

Nesta figura, são utilizadas as etiquetas morfossintáticas constantes no Anexo 1, nomeadamente S (frase ou *sentence*), NP (sintagma nominal ou *noun phrase*), VP (sintagma verbal ou *verbal phrase*), PP (sintagma proposicional ou *propositional phrase*) PRS (pronome ou *pronoum*), V (verbo ou *verb*), ART (artigo ou *article*), N (nome ou *name*), P (preposição ou *preposition*).

2.4 Representação de Texto

A representação de texto permite transformar este, em representações vetoriais numéricas de forma serem utilizadas, por exemplo, em algoritmos de aprendizagem máquina ou aprendizagem profunda. De seguida irão ser abordadas as técnicas mais comuns de representação de texto.

2.4.1 *Bag of Words*

O modelo *Bag of Words* (BoW) é utilizado para representar documentos de texto como coleções desordenadas de palavras, ignorando a ordem e a estrutura gramatical. Este modelo conta o número de ocorrências de uma palavra num determinado texto (Qader et al., 2019), destacando-se pela eficiência na representação do mesmo (Mikolov, Sutskever, et al., 2013).

Para exemplificar o modelo BoW, consideremos um texto relativo a comentários sobre filmes. O modelo BoW do texto composto pelo conjunto de comentários, pode ser representado por uma matriz, onde cada linha representa um comentário (ou genericamente um documento de texto) e cada coluna corresponde à frequência de uma palavra, tal como se ilustra na Tabela 3. Em textos grandes, esta matriz apresenta uma dimensionalidade e esparsidade muito elevada, o que representa uma desvantagem. Outra desvantagem, deriva do facto dos vetores serem baseados nas frequências absolutas das palavras, o que pode conduzir à existência de palavras que ocorrem com mais frequência e, que podem ofuscar outras menos frequentes retirando-lhes relevância.

Tabela 3 - Bag-of-Words (Exemplo)

	Palavras						
Comentário	Bom	Este	Filme	Mau	não	um	É
Bom filme	1	0	1	0	0	0	0
Mau filme	0	0	1	1	0	0	0
Não é um bom filme	1	0	1	0	1	1	1
Este filme é um mau filme	0	1	2	1	0	1	1
Bom bom bom filme	3	0	1	0	0	0	0

Fonte: Elaboração própria

O BoW é utilizado em tarefas de PLN em que a ordem das palavras e o seu significado semântico não é relevante para a tarefa que se pretende.

2.4.2 *Bag of N-grams*

O modelo *Bag of n-grams* é uma extensão do modelo BoW que leva em consideração não apenas as palavras individuais, mas também sequências contínuas de palavras de tamanho *n*, conhecidas como *n-grams* (Cavnar & Trenkle, 1994). A Tabela 4 ilustra uma representação *Bag of n-grams* com *n* igual a dois, também conhecida como bigramas.

Tabela 4 - Bag of n-grams (Exemplo)

	2-grams ou bigramas								
Comentário	bom bom	bom filme	este filme	filme é	mau filme	não é	um bom	um mau	é um
Bom filme	0	1	0	0	0	0	0	0	0
Mau filme	0	0	0	0	1	0	0	0	0
Não é um bom filme	0	1	0	0	0	1	1	0	1
Este filme é um mau filme	0	0	1	1	1	0	0	1	1
Bom bom bom filme	2	1	0	0	0	0	0	0	0

Fonte: Elaboração própria

2.4.3 TF-IDF

O modelo TF-IDF (*Term Frequency-Inverse Document Frequency*) tenta resolver o problema da ofuscação das palavras atrás referido usando um fator de escala ou normalização no seu cálculo (Bafna et al., 2016). O TF-IDF usa uma combinação de duas componentes no seu cálculo: (i) o TF ou *Term Frequency* e, (ii) o IDF ou *Inverse Document Frequency* (Jones, 1972). O valor final do TF-IDF é dado pela multiplicação destas duas componentes (Equação 1).

Equação 1 - Cálculo do tf-idf

$$tf - idf(t, d, D) = tf(t, d) \,.\, idf(t, D)$$

em que:

$$tf(t, d) = \frac{número\ de\ vezes\ que\ o\ termo\ t\ ocorre\ no\ documento\ d}{número\ de\ termos\ no\ documento\ d}$$

$$idf(t,\ D) = \log \frac{número\ de\ documentos\ no\ corpus\ D}{número\ de\ documentos\ que\ contém\ o\ termo\ t}$$

Este método é usado, sobretudo, em classificação de texto e em recuperação da informação (Aizawa, 2003). A Tabela 5 ilustra uma representação TF-IDF.

Tabela 5 - TF-IDF (Exemplo)

	Palavras						
Comentário	bom	este	Filme	Mau	Não	Um	É
Bom filme	0,81	0,00	0,58	0,00	0,00	0,00	0,00
Mau filme	0,00	0,00	0,51	0,86	0,00	0,00	0,00
Não é um bom filme	0,39	0,00	0,28	0,00	0,58	0,47	0,47
Este filme é um mau filme	0,00	0,51	0,49	0,41	0,00	0,41	0,41
Bom bom bom filme	0,97	0,00	0,23	0,00	0,00	0,00	0,00

Fonte: Elaboração própria

2.4.4 *Word Embeddings*

Os modelos anteriores são considerados modelos estatísticos de representação de texto, enquanto as *Word Embeddings* utilizam uma abordagem baseada em aprendizagem profunda não supervisionada, treinada em grandes corpora.

Por definição, *Word Embeddings*, também conhecidos por vetores de palavras, são uma abordagem para a representação de documentos e palavras. O objetivo é gerar

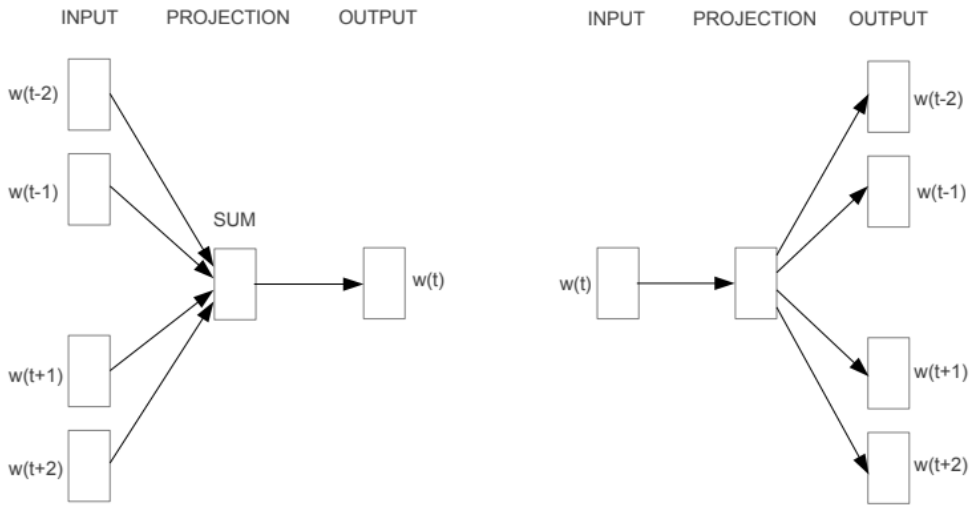
representações de palavras densas, distribuídas e contínuas, que captem similaridade contextual e semântica. Desta forma, palavras com significados semelhantes, ou que, ocorram em contextos semelhantes terão vetores de palavras mais próximos uns dos outros, em comparação com palavras que são semanticamente ou contextualmente diferentes.

Word2Vec

O Word2Vec é uma das abordagens mais populares para criar *word embeddings* que foi desenvolvido pela Google (Mikolov, Chen, et al., 2013). O Word2Vec baseia-se no contexto em que as palavras aparecem em grandes corpora de texto, para assim, aprender as suas representações vetoriais. Esta abordagem utiliza duas arquiteturas diferentes para criar as representações vetoriais (Figura 5):

- ***Continuous Bag of Words* (CBOW)**: neste modelo as representações distribuídas do contexto (ou palavras vizinhas) são combinadas para prever a palavra-alvo (a palavra central) (Figura 5a).
- **Skip-gram**: este modelo faz o inverso do CBOW, sendo usada a palavra alvo (a palavra central) para prever as palavras do contexto de origem (palavras vizinhas) (Figura 5b).

Figura 5. Arquiteturas Word2Vec



a) CBOW b) Skip-gram

Fonte: (Mikolov, Chen, et al., 2013)

Glove

O GloVe (*Global Vectors for Word Representation*) é um método que combina técnicas baseadas em contagem e em contexto (Pennington et al., 2014). Ao contrário das abordagens puramente baseadas em contexto, como o Word2Vec, que consideram apenas a proximidade no espaço vetorial, o GloVe inclui também informação sobre a frequência de coocorrência das palavras. Para isso, é construída uma matriz de coocorrência palavra-palavra onde, cada elemento indica com que frequência duas palavras aparecem juntas num determinado contexto. Esta abordagem resulta em *Word Embeddings* que capturam não apenas a semelhança entre as palavras, mas também as suas relações de frequência no corpus, e permitem representar o contexto semântico das palavras de forma mais precisa.

FastText

O FastText foi introduzido pela Facebook em 2016, como uma extensão e melhoria do Word2Vec, que visa capturar informação tanto ao nível da palavra, como ao nível de subpalavras (Bojanowski et al., 2017). Ao considerar as subpalavras, o FastText captura informação morfológica e semântica mais refinadas, o que o torna especialmente útil para lidar com palavras raras ou morfologicamente ricas em línguas como inglês, alemão ou finlandês. Outra característica distintiva do FastText é a sua capacidade em lidar com palavras desconhecidas ou fora do vocabulário (*out-of-vocabulary*, conhecido pela sigla OOV). Ao considerar as subpalavras, mesmo palavras que não estejam presentes no conjunto de treino podem ser representadas como uma combinação das suas subpalavras.

2.4.5 Redução da Dimensionalidade

A redução da dimensionalidade é uma técnica essencial em PLN, de forma a lidar com a alta dimensionalidade dos dados, especialmente em tarefas que envolvem grandes vocabulários, ou representações densas de texto, como os modelos de *Word Embeddings,* vistos anteriormente.

Dos vários métodos de redução da dimensionalidade em PLN, destacam-se os seguintes:

- ***Principal Component Analysis* (PCA)**: o PCA procura capturar as relações semânticas e estruturais entre as palavras, identificando as direções de maior variância nos dados, projetando-os num espaço de dimensão menor, sem a perda de informação (Berry et al., 1995; Jolliffe & Cadima, 2016). Esta técnica tem por objetivo facilitar o trabalho humano da interpretação.
- ***Latent Semantic Indexing* (LSI)**: o LSI visa capturar relações semânticas latentes entre termos e documentos. Baseia-se na aplicação de técnicas de decomposição de matrizes, como a *Singular Value Decomposition* (SVD), para reduzir a dimensionalidade de um espaço de termos-documentos. Isso permite a descoberta de tópicos subjacentes nos documentos, facilitando a recuperação de informação e, melhorando a precisão em tarefas como recuperação de informação e agrupamento de documentos (Deerwester et al., 1990; Shah & Patel, 2016).

2.5 Aplicações de Processamento de Linguagem Natural

O Processamento de Linguagem Natural é aplicado nos mais diversos contextos, desde a execução de tarefas de tradução automática, análise de sentimentos ou até resumo de texto. Nestas tarefas existem várias métodos e algoritmos que permitem o Processamento de Linguagem Natural. Tanto as tarefas como os algoritmos de processamento irão ser explicados de seguida.

2.5.1 Classificação de Texto

A classificação de texto refere-se ao processo de categorização de documentos textuais em classes ou categorias pré-definidas com base no conteúdo. A sua utilização é muito variada e inclui, entre outros, a análise de sentimentos, a deteção de spam ou a categorização de notícias para verificar o tema da notícia.

A classificação de texto, é obtida através de aprendizagem máquina supervisionada, com uso de algoritmos de classificação. Estes algoritmos são treinados com um conjunto de dados categorizado que aprende a relação entre os textos e as classes e utilizam como entrada uma representação de texto em vetores numéricos, tal como

ilustrado na Secção 2.4. Alguns dos algoritmos mais comuns utilizados em classificação de texto, incluem:

- **Naive Bayes**: algoritmo simples e eficaz que se baseia no teorema de Bayes para calcular a probabilidade de um documento pertencer a uma determinada classe com base na frequência das palavras no documento (McCallum & Nigam, 1998; Porter, 1980).
- ***Support Vector Machine* (SVM)**: as SVM's são algoritmos de aprendizagem supervisionada que constroem um hiperplano de separação entre as diferentes classes no espaço de características, maximizando a margem entre os pontos de dados de diferentes classes (Joachims, 1998; Vapnik & Chervonnkis, 1964).
- ***Decision Trees* (DT) e *Random Forests* (RF)**: as DT são modelos que dividem o espaço de características em regiões hierárquicas com base em regras de decisão (Quinlan, 1986), já as RF são compostas por múltiplas árvores de decisão e combinam as suas previsões para melhorar a precisão e evitar *overfitting* (explicado na Secção 2.7.4) (Breiman, 2001).
- ***K-Nearest Neighbors Algorithm* (KNN)**: algoritmo utilizado em classificação de texto para atribuir rótulos de classe a novos documentos, com base na similaridade em documentos previamente categorizados. Caso os vizinhos mais próximos sejam maioritariamente de uma classe, então a amostra em causa será dessa mesma classe (Bishop, 2006).
- ***Gradient Boosting Machine* (GBM)**: as GBM, como o XGBoost (Chen & Guestrin, 2016), o LightGBM (Ke et al., 2017) e CatBoost (Prokhorenkova et al., 2018), são métodos de aprendizagem máquina que constroem uma sequência de modelos de forma iterativa, melhorando gradualmente a previsão combinada (Friedman, 2001).

Os algoritmos anteriores não têm em consideração a ordem ou sequência das palavras, sendo, por isso limitados, já que, a ordem em que as palavras ocorrem numa frase influencia o significado. As *Recurrent Neural Networds* (RNN) são um tipo de arquitetura de rede neuronal projetada para lidar com este problema. A principal característica das RNN é a capacidade de manter e propagar informação ao longo do

tempo, permitindo que elas capturem dependências de longo prazo numa dada sequência. As *Long Short-Term Memory* (LSTM) (Hochreiter & Schmidhuber, 1997) e as *Gated Recurrent Unit* (GRU) (Cho, Van Merriënboer, et al., 2014) são variantes das *Recurrent Neural Networks* (RNN) que são particularmente eficazes para lidar com dados sequenciais, como texto, pela sua arquitetura que permite a propagação de informação e a captura de dependências temporais complexas entre palavras numa frase ou entre átomos de um documento (Rumelhart et al., 1986).

2.5.2 Análise de Sentimentos

A Análise de Sentimentos (*Sentiment Analysis* ou SA), é uma subárea da classificação de texto que pretende identificar e classificar opiniões, emoções e atitudes expressas em textos. O objetivo, é, pois, determinar se um texto exprime um sentimento positivo, negativo ou neutro e pode ser efetuada ao nível do documento, da frase e ainda do aspeto (Medhat et al., 2014). As técnicas de classificação de sentimentos podem ser divididas em três abordagens, sendo elas (Maynard & Funk, 2012):

- Baseada em aprendizagem máquina através da utilização de algoritmos de classificação supervisionados, como os visto na secção anterior, mas também não supervisionados quando não se dispõe de um corpus de treino etiquetado;
- Baseada em léxico através da utilização de dicionários ou léxicos predefinidos que associam palavras a polaridades sentimentais para determinar o sentimento geral do texto;
- Abordagens híbridas que combinam as duas técnicas anteriores.

2.5.3 Agrupamento de Texto

O Agrupamento de Texto, também conhecido como *clustering* de texto, é uma técnica de análise de dados que agrupa documentos de texto semelhantes em *clusters* ou grupos com base nas suas características textuais (Jain et al., 1999).

Existem várias abordagens para o agrupamento de texto, incluindo métodos baseados em distância, como o *k-means* (Dhillon & Modha, 2001) e hierárquicos, como o aglomerativo ou divisivo. Estes métodos utilizam geralmente medidas de similaridade,

como a distância euclidiana, a similaridade do cosseno ou a distância de Jaccard, para agrupar documentos com base nas suas características textuais.

2.5.4 Tradução Automática

A Tradução Automática de texto é a área que se dedica ao desenvolvimento de sistemas capazes de traduzir automaticamente textos de uma língua para outra, sem intervenção humana. A tradução automática pode ser realizada por meio de abordagens baseadas em regras, onde são utilizadas regras gramaticais e vocabulário para realizar a tradução ou, por abordagens baseadas em dados, onde são utilizados modelos estatísticos ou de aprendizagem máquina treinados em grandes corpora bilíngues (Lopez, 2008).

2.5.5 Correção Automática de Texto

A Correção Automática de Texto visa identificar e corrigir erros ortográficos, gramaticais e de digitação em textos de forma automatizada. Desempenha um papel crucial em várias aplicações, como nos processadores de texto, navegadores da *web*, correio eletrónico e mensagens de texto, melhorando a qualidade e a compreensão da comunicação escrita. A correção automática de texto envolve a análise de padrões de linguagem e o uso de algoritmos para sugerir correções apropriadas com base num vocabulário conhecido e em regras gramaticais. São também utilizados modelos estatísticos e de aprendizagem máquina para prever correções com base em padrões de erros comuns encontrados em textos (Jurafsky & Martin, 2023).

2.5.6 Extração de Informação

A Extração de Informação é a área que tem por objetivo extrair conhecimento estruturado e relevante a partir de documentos de texto não estruturados. São utilizadas várias técnicas e abordagens de PLN, tais como, análise sintática, análise semântica e reconhecimento de entidades nomeadas, que permitem identificar e extrair informações relevantes dos textos. Essas informações são então organizadas de forma a permitir a alimentação de sistemas de apoio à decisão numa variedade de campos, incluindo a medicina, negócios, entre muitos outros.

2.5.7 Resumo de Texto

O Resumo de Texto é o processo de condensar um documento de texto numa versão menor mantendo as informações essenciais e o seu significado original. A sua utilização é bastante variada, sendo usada em sistemas de recuperação de informação, na criação de resumos automáticos de documentos ou na geração de legendas para imagens. São habitualmente utilizadas duas abordagens distintas: (i) extrativa, onde partes relevantes do texto original são selecionadas e reorganizadas, e (ii) abstrativa, onde um novo resumo é gerado a partir da compreensão do conteúdo do texto original (Allahyari et al., 2017).

2.5.8 Pergunta/Resposta

A tarefa de Pergunta/Resposta (*Question Answering* ou QA) é uma área especializada na Recuperação de Informação (*Information Retrieval* ou IR), que procura a construir sistemas capazes de compreender perguntas formuladas em linguagem natural e fornecer respostas relevantes e precisas (Abdi et al., 2018). O método mais comum de Pergunta/Resposta é o método extrativo, o qual recebe um documento de entrada (chamado contexto) e uma pergunta sobre o documento e, extrai o segmento do texto do documento que contém a resposta (Nassiri & Akhloufi, 2023). Atualmente, também existem sistemas capazes de gerar a resposta sem a necessidade de contexto fornecido pelo utilizador gerando-se as respostas com base nos dados com que foram treinados.

A capacidade de resposta dos sistemas de Pergunta/Resposta melhorou significativamente nos últimos anos graças aos modelos de aprendizagem profunda e, em particular à arquitetura dos *Transformer,* a qual será descrita na secção seguinte. Com efeito, os sistemas de Pergunta/Resposta estão a tornar-se numa área cada vez mais relevante em PLN, sendo atualmente, utilizados em diversas áreas na recuperação de informação, na extração de entidades, em *chatbots* ou em sistemas de diálogo (Nassiri & Akhloufi, 2023).

2.6 *Transformers*

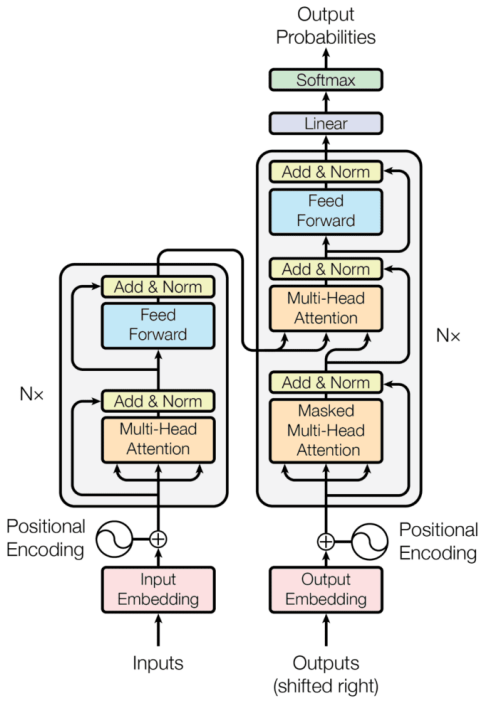
Os *Transformers*, são uma arquitetura capaz de processar dados sequenciais, como texto, áudio ou imagens. A sua principal vantagem é a capacidade de processar dados

sequenciais de forma paralela, o que os torna muito eficientes para treinar modelos em grandes conjuntos de dados. Por outro lado, a arquitetura dos *Transformers* é escalável, permitindo a adição de camadas adicionais para aumentar a profundidade do modelo e aumentar a capacidade de aprendizagem (Vaswani et al., 2017).

Os *Transformers* são utilizados numa grande quantidade de tarefas, como a classificação de texto, tradução automática e resposta a perguntas, e utilizam mecanismos de atenção para capturar as dependências de longo alcance numa sequência de entrada e codificação posicional, que permite que o modelo se concentre em partes específicas da entrada durante o processamento (Khan et al., 2022).

A arquitetura dos *Transformers*, ilustrada na Figura 6, consiste numa componente de codificação composta por várias camadas de *encoders* e uma componente de descodificação composta por várias camadas de *decoders*.

Figura 6 – Arquitetura dos *Transformers*



Fonte: (Vaswani et al., 2017)

A entrada do modelo é uma sequência de vetores de palavras (*embedding*) que são passados para uma série de camadas de *encoder*s, representados no lado esquerdo da Figura 6. Cada camada de *encoder* é composta por um módulo de autoatenção (*multi-head attention*) e um módulo de *feed-forward*. O módulo *multi-head attention* calcula pesos de atenção para cada palavra em relação a todas as outras palavras da sequência,

permitindo que o modelo dê mais importância às palavras mais relevantes para a tarefa em questão. O módulo de *feed-forward*, por sua vez, processa cada palavra individualmente gerando uma sequência de representações contínuas $z = (z_1, ..., z_n)$.

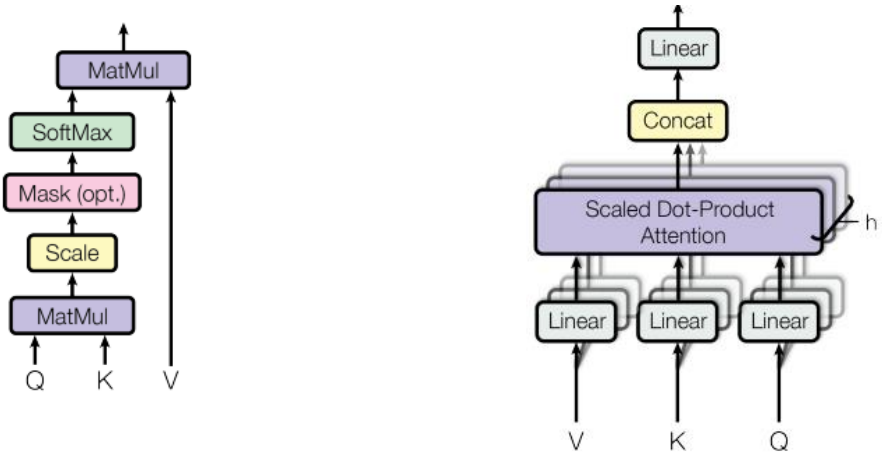
O *decoder* (representado no lado direito da Figura 6) gera uma sequência de saída $(y_1, ..., y_m)$ consumindo elemento a elemento a sequência z gerada pelas camadas de *encoder*. A cada passo o modelo é autorregressivo, consumindo os elementos gerados anteriormente como entrada adicional ao gerar o próximo.

2.6.1 Atenção

Uma função de atenção pode ser descrita como o mapeamento de uma consulta (*query*) e um conjunto chaves (*key*) e valores (*value*) para uma saída, onde a consulta, as chaves, os valores e a saída são representados pelos vetores q, k e v (Vaswani et al., 2017). A saída é calculada como uma soma ponderada de valores onde, o peso atribuído a cada valor é calculado por uma função de compatibilidade da consulta com a chave correspondente.

Existem dois tipos distintos de funções de atenção as quais são descritas de seguida: a *Scaled Dot-Product Attention* e a *Multi-Head Attention* (Figura 7).

Figura 7 – Tipos de funções de atenção



a) Scaled Dot-Product Attention b) Multi-Head Attention

Fonte: (Vaswani et al., 2017)

Scaled Dot-Product Attention

A *Scaled Dot-Product Attention* é calculada sobre um conjunto de consultas simultâneas agrupadas, considerando que as chaves e os valores são agrupados em matrizes representadas por Q, K e V respetivamente. A entrada consiste em consultas e chaves de dimensão d_k e os valores de dimensão d_v. São calculados os produtos escalares das consultas Q com todas as chaves K, dividindo cada um por $\sqrt{d_k}$, aplicando-se de seguida uma função *softmax* para obter os pesos dos valores V (Equação 2).

Equação 2 - Cálculo da Scaled Dot-Product Attention

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Multi-head Attention

A *Multi-head Attention* utiliza mais de um mecanismo de atenção para o mesmo conjunto de entradas. Cada um desses mecanismos é conhecido como uma cabeça (*head*) sendo a escolha do número de cabeças um dos hiperparâmetros de modelos deste tipo. Cada saída dessas cabeças é concatenada e passada por uma camada linear, podendo ser expressa pela Equação 3.

Equação 3 - Multi-head Attention

$$MultiHead(Q, K, V) = Concat(head_1, \dots, head_h)\, W^o$$

em que:

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$$

2.6.2 Modelos *Transformers* Mais Conhecidos

Desde que os *Transformers* surgiram, várias implementações foram criadas, sendo que, nesta secção, serão descritas duas das famílias de modelos de linguagem mais referidas na literatura: o GPT (*Generative Pre-trained Transformer*) e o BERT (*Bidirectional Encoder Representations from Transformers*).

GPT (*Generative Pre-trained Transformer*)

O GPT é uma família de modelos de linguagem baseados em *Transformers* desenvolvidos pela OpenAI[20]. Esses modelos têm-se destacado devido ao seu desempenho excecional numa ampla variedade de tarefas, tais como a geração de texto, a tradução automática, a resposta a perguntas ou a sumarização de documentos.

A arquitetura do GPT é unidirecional, o que significa que, cada palavra só tem em conta as palavras anteriores na sequência, garantindo que o modelo não tenha acesso a informação futura durante o processo de geração. O GPT não possui um *encoder* e *decoder* separados como no modelo tradicional de *Transformers*, mas sim, uma única pilha de camadas que codifica e gera texto de forma contínua. Cada camada do GPT contém múltiplas cabeças de atenção e camadas totalmente conectadas, permitindo que o modelo capture dependências de longo alcance e aprenda representações ricas e contextuais de texto.

O GPT, nas suas várias versões, é pré-treinado sem supervisão em grandes corpora de texto, seguido de ajuste fino discriminativo supervisionado para uma grande variedade de tarefas de PLN. O GPT-2 foi um dos primeiros modelos da família GPT a ganhar destaque, possuindo 1.5 biliões de parâmetros (Radford et al., 2019). Mais tarde, o GPT-3 veio a representar uma evolução significativa, possuindo 175 biliões de parâmetros, tornando-se, à data, o maior modelo de linguagem criado. Em 2022, surge o GPT-3.5 e em 2023 o GPT-4 dos quais se possui pouca informação. A Tabela 6 resume as características dos modelos da família GPT (Brown et al., 2020).

[20] https://openai.com/

Tabela 6 - Família de Transformers GPT

Modelo	Arquitetura	Parâmetros	Dados de Treino	Lançamento
GPT-1	12 camadas, decoder Transformer com 12 cabeças (sem encoder), seguido por linear-softmax	117 milhões	BookCorpus: 4.5 GB de texto proveniente de 7 000 livros não publicados de vários géneros	11 de junho de 2018
GPT-2	Idêntica ao GPT-1 mas com normalização modificada	1.5 biliões	WebText: 40 GB de texto, 8 milhões de documentos provenientes de 45 milhões de páginas web votadas no Reddit	14 de fevereiro de 2019
GPT-3	Idêntica ao GPT-2 mas com modificações para permitir maior escala	175 biliões	499 biliões de tokens consistindo em CommonCrawl (570 GB), WebText, Wikipédia em inglês e dois corpora de livros	28 de maio de 2020
GPT-3.5	Não divulgada	175 biliões	Não divulgados	15 de março de 2022
GPT-4	Treinado também com previsão de texto e RLHF[21] aceita texto e imagens como entrada. Mais detalhes não divulgados	1.7 triliões (estimativa)	Não divulgados	14 de março de 2023

Fonte: Elaboração própria

BERT (*Bidirectional Encoder Representations from Transformers*)

O BERT é uma família de modelos de linguagem da Google que foi projetado para treinar representações bidirecionais profundas de texto não categorizado. Este modelo foi pré-treinado com dados para diferentes tarefas de PLN, tendo por base a predição da próxima frase (*Next Sentence Prediction* ou NSP), em que os pares de frases são criados e o modelo é treinado para verificar se a próxima frase é lógica, e a tarefa *Masked Language Modeling* (MLM) que mascara aleatoriamente palavras numa frase e o modelo tem de prever essas palavras (Devlin et al., 2018).

A arquitetura do modelo BERT foi implementada com base num codificador *Transformer* bidirecional de multicamadas. Esta arquitetura permite que o modelo

[21] Reinforcement Learning from Human Feedback

tenha em conta o contexto à esquerda e à direita de cada palavra o que melhora a compreensão da linguagem natural.

O ajuste fino deste modelo foi feito com base nos *datasets* GLUE (*General Language Understanding Evaluation*) (Wang et al., 2018) que é usado para avaliar a compreensão da linguagem natural, SQuAD (Rajpurkar et al., 2016) e SWAG (*Situation With Adversarial Generations*), que é um conjunto de dados projetado para avaliar a capacidade de o modelo compreender o contexto em situações adversas (Zellers et al., 2018).

RoBERTa

O modelo RoBERTa (*Decoding-enhanced BERT with Disentangled Attention*), surgiu a partir do modelo BERT com uma abordagem melhorada do pré-treino. Este modelo tem como objetivo ultrapassar as limitações do modelo BERT (Liu et al., 2019). No treino foi utlizada uma maior quantidade de dados (quase dez vezes mais dados que o modelo original BERT) e um tempo de treino prolongado para se obter uma maior capacidade de generalização. O ajuste fino levou em conta vários valores, como a taxa de aprendizagem, com objetivo de melhorar a avaliação do modelo.

O modelo RoBERTa foi avaliado em várias tarefas de PLN com desempenho superior ao modelo original tendo em conta os diversos *datasets* de avaliação como o SQuAD e o GLUE.

DeBERTa

O DeBERTa (*Decoding-enhanced BERT with disentangled attention*), surgiu a partir do modelo RoBERTa, através de um treino aprimorado com metade dos dados utilizados no RoBERTa, e da apresentação de tuas técnicas relevantes (He et al., 2020). A primeira, é um mecanismo de atenção em que cada palavra é representada por dois vetores que codificam o conteúdo e a posição, o que traz melhorias na interpretação de contextos. Em segundo, é utilizado um decodificador que substitui a camada *softmax* de saída para prever os átomos codificados para o pré-treino do modelo.

O modelo DeBERTa pode ser comparativamente melhor que o modelo RoBERTa-Large, em várias tarefas de Processamento de Linguagem Natural, como a análise de sentimentos ou em tarefas de compreensão de texto.

2.7 Métricas de Avaliação de Modelos

Existem várias métricas de avaliação para avaliar os diversos tipos de modelos existentes. Estas métricas permitem validar a performance de um modelo e a forma como este é capaz de responder à tarefa proposta.

2.7.1 Métricas de Classificação

Nas tarefas de classificação o objetivo é prever a categoria a que uma observação pertence. O resultado da observação são valores categóricos e discretos que habitualmente são tidos como rótulos ou classes.

Matriz de confusão

A matriz de confusão é usada na classificação binária e é representada por uma tabela que possui duas linhas e duas colunas, que retrata a avaliação do desempenho de um modelo. Na horizontal, representa-se a classe prevista pelo modelo, e na vertical a classe real. A matriz de confusão pode ser representada como na Figura 8.

Esta matriz é dividida em quatro áreas: TP que indica os casos que foram classificados corretamente na Classe Positiva, TN que representa os casos bem classificados na Classe Negativa, FP que revela os casos mal classificados na Classe Positiva e FN que indica os casos incorretamente classificados na Classe Negativa (Amidi & Amidi, 2018)

Figura 8 - Matriz de confusão

		Classe Prevista	
Classe Real		**TP** True Positive	**FN** False Negative
		FP False Positive	**TN** True Negative

Fonte: Elaboração própria

Accuracy

A *accuracy* representa o desempenho geral do modelo. Esta métrica indica o quão verdadeiro, exato e correto é o modelo e é calculada com base na Equação 4.

Equação 4 - *Accuracy*

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Precision

A *precision* mede a qualidade dos modelos e o quão precisos estes são. Esta avalia a percentagem de itens que o sistema avalia como sendo da classe positiva (Equação 5).

Equação 5 - *Precision*

$$precision = \frac{TP}{TP + FP}$$

Recall

A *recall* mede a percentagem de itens que fazem parte da entrada do modelo e que foram corretamente classificados pelo mesmo (Equação 6). Esta equação indica a quantidade de classes previstas corretas (Jurafsky & Martin, 2023).

Equação 6 - *Recall*

$$recall = \frac{TP}{TP + FN}$$

Specificity

A *specificity* indica o quão capaz o modelo é de prever a classe negativa corretamente (Equação 7) (Amidi & Amidi, 2018).

Equação 7 - *Specificity*

$$specificity = \frac{TN}{TN + FP}$$

F-Score

Combina a *precision* com a *recall* de forma harmónica (Equação 8). β pondera a importância da *recall* e da *precision*, sendo que, quando se obtém β > 1 a *recall* terá

melhores valores e quando β < 1 a *precision* tem valores mais elevados. Quando β = 1 a *precision* e a *recall* estão equilibrados e pode-se considerar a

Equação 9 (Grandini et al., 2020; Jurafsky & Martin, 2023).

Equação 8 - *F_β -Score*

$$F_\beta - score = (1 + \beta^2)\frac{precision\,.recall}{(\beta^2 precision) + recall}$$

Equação 9 - *F1-score*

$$F_1 - score = 2\frac{precision\,.recall}{precision + recall}$$

Receiver Operating Characteristics (ROC)

A *Receiver Operating Characteristics* (ROC) mostra a capacidade de um modelo distinguir as classes positivas das negativas através de um gráfico. Quanto mais próxima a curva estiver do canto superior esquerdo melhor será o desempenho do modelo. Esta métrica é calculada por duas equações, sendo que, a Equação 11 é representada na horizontal, e a Equação 10 é representada na vertical (Amidi & Amidi, 2018; Kowsari et al., 2019). Na Figura 9 pode ser observada a representação gráfica da métrica ROC.

Equação 10 - *True Positive Rate*

$$TPR = \frac{TP}{TP + FN}$$

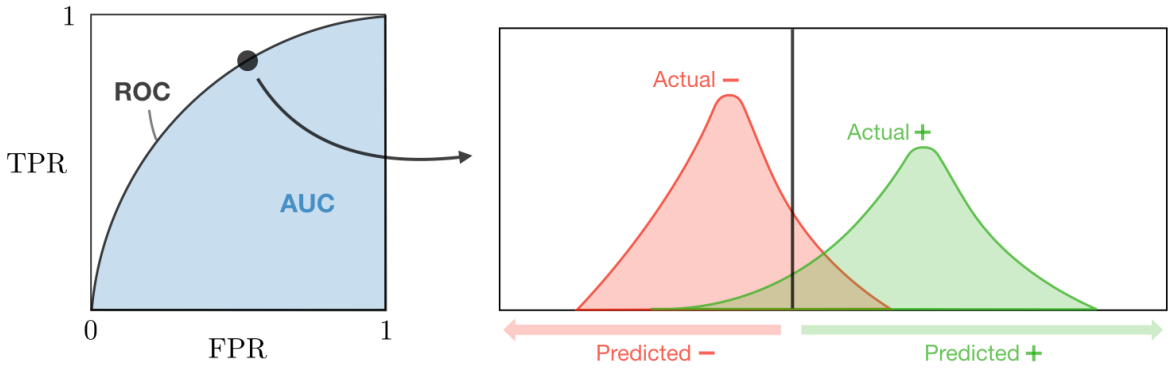
Equação 11 - *False Positive Rate*

$$FPR = \frac{FP}{TN + FP}$$

Area Under the Receiving Operating Curve (AUC)

A *Area Under the Receiving Operating Curve* (AUC) é observada por toda a área abaixo da curva formada pelo gráfico da métrica ROC. Esta, fornece a medida do desempenho dos modelos de classificação tendo em conta as taxas da Equação 10 e Equação 11. Quanto maior for a AUC melhor será o desempenho do modelo (Figura 9).

Figura 9 - Representação gráfica das métricas ROC e AUC



Fonte: (Amidi & Amidi, 2018)

2.7.2 Métricas de Regressão

Nas tarefas de regressão tem-se por objetivo prever um valor contínuo de saída (também conhecidos como variável dependente ou alvo), com base numa ou mais entradas (conhecidas como variáveis independentes ou características), tais como, prever o preço de uma casa com base em características como a área e o número de quartos, ou prever a temperatura de amanhã com base em dados históricos de temperatura e padrões climáticos.

Mean Absolute Error (MAE)

A *Mean Absolute Error* (MAE) avalia a *precision* das previsões através do cálculo da média das diferenças absolutas entre as previsões e os valores reais (Karunasingha, 2022; W. Wang & Lu, 2018). O cálculo que é efetuado para se obter o resultado desta métrica pode ser observado na Equação 12. Esta métrica é bastante robusta, tendo em conta que, trata os erros de forma absoluta e que tem menor viés, comparativamente à métrica RMSE que será explicada de seguida.

Equação 12 - *Mean Absolute Error*

$$MAE = \frac{\sum_{n=1}^{N} |\widehat{r_n} - r_n|}{N}$$

Root Mean Squared Error (RMSE)

A *Root Mean Squared Error* (RMSE) é a raiz quadrada da média dos quadrados dos erros, e é uma medida que representa o quão dispersas estão as previsões dos números reais (Equação 13).

Equação 13 - *Root Mean Squared Error*

$$RMSE = \sqrt{\frac{\sum_{n=1}^{N}(\widehat{r_n} - r_n)^2}{N}}$$

2.7.3 Função de Perda (*loss*)

A função de perda (ou *loss*) desempenha um papel fundamental nos algoritmos de aprendizagem máquina, na medida em que, possibilita a melhoria continua da performance dos algoritmos. Na realidade, a *loss* determina o desvio do valor previsto face ao real, sendo que, quanto mais distante do valor real for o previsto maior será o resultado dessa métrica, sendo que, o valor desejável nesta métrica é 0. Durante o treino, o objetivo é minimizar a função de perda, o que é feito através de um processo de otimização onde, os parâmetros do modelo são ajustados para reduzir o valor da função de perda. Esta métrica tem a vantagem de poder ser utilizada em modelos de aprendizagem máquina para melhoria no treino, com fim de se chegar o mais próximo possível dos resultados desejáveis (Wang et al., 2022).

Logarithmic Loss

Também conhecido por *Cross Entropy Loss,* é muito utilizado na regressão e em problemas de classificação binária. O *Logarithimic Loss* pode ser definido pela fórmula da Equação 14. Nesta fórmula, $L(y, \hat{y})$ é o *Logarithmic Loss* para um determinado exemplo, y é a classe real e $\hat{y}$ a classe prevista.

Equação 14 - *Logarithmic Loss*

$$L(y, \hat{y}) = -[y * log(\hat{y}) + (1 - y) * log(1 - \hat{y})$$

Softmax Cross-Entropy Loss Function

A métrica *Softmax Cross-Entropy Loss* é semelhante à anterior, mas utilizada em problemas de classificação multiclasse. Esta normaliza as saídas do modelo em probabilidades para as classes. A função pode ser definida pela Equação 15.

Equação 15 - *Softmax Cross-Entropy Loss*

$$L(y, \hat{y}) = -\sum_{i=1}^{c} y_i * log\left(\frac{e^{\hat{y}_i}}{\sum_{j=1}^{c} e^{\hat{y}_j}}\right)$$

Exponential Loss function

A função que faz o cálculo da *Exponential Loss* é contínua e diferenciável. Este cálculo conduz o modelo à melhoria e penaliza de maneira variável as amostras mal classificadas, atribuindo maior penalização a amostras menos assertivas. Pode-se definir pela Equação 16.

Equação 16 - Exponential *Loss*

$$L\big(y, f(x)\big) = e^{-y*f(x)}$$

2.7.4 Curvas de Aprendizagem

A aprendizagem de dados por parte de modelos pode ter vários comportamentos que, geralmente, são observados através das curvas de aprendizagem (*Learning Curves* – LC) sobre os dados de treino e de validação.

As curvas de aprendizagem são uma descrição matemática que reflete a performance em tarefas repetidas, por exemplo, um modelo quando é treinado com um *dataset* por sucessivas épocas aprende a executar a tarefa proposta. O seu comportamento pode ser observado pelo gráfico gerado. Esta descrição matemática gera um gráfico, em que o eixo *x* representa o tempo ou o número de épocas de treino do modelo, e o eixo *y* representa a métrica que está a ser avaliada, como a *loss, F1* ou a *accuracy*, por exemplo.

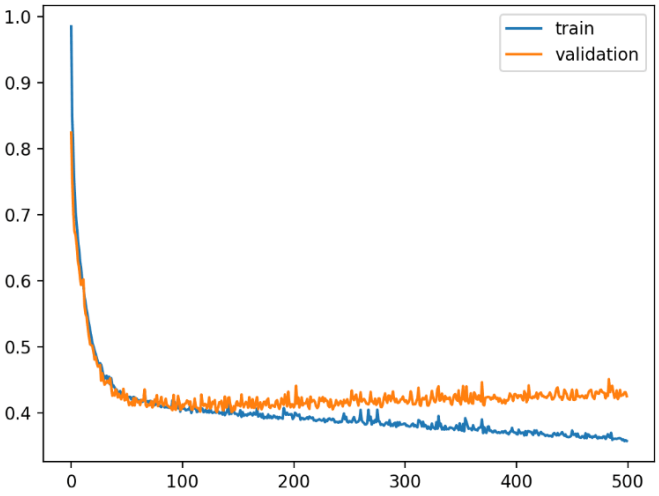
A observação destas curvas permite o diagnóstico de problemas de aprendizagem como o *overfitting* ou o *underfitting* (Anzanello & Fogliatto, 2011).

Overfitting

O *overfitting* é um dos maiores problemas no treino de modelos, porque é natural que, a uma determinada altura do treino o modelo não seja capaz de melhorar a sua performance. Este problema ocorre quando o modelo aprende o conjunto de dados muito bem, o que dificulta a generalização de dados, que pode ser verificado pelo desempenho na validação do modelo.

Este comportamento ocorre quando o modelo é treinado por muitas épocas, ou quando o modelo possui mais capacidades de resolução do problema do que seria necessário (Jabbar & Khan, 2015). Na Figura 10 é demonstrado um exemplo de *overfitting,* em que as curvas de treino e validação surgem afastadas no final do treino e, a partir da época número 100, o modelo já não é capaz de melhorar as suas capacidades. No final do treino o modelo demonstra uma performance pior do que aquela que tem na época número 100.

Figura 10 - Exemplo de Overfitting



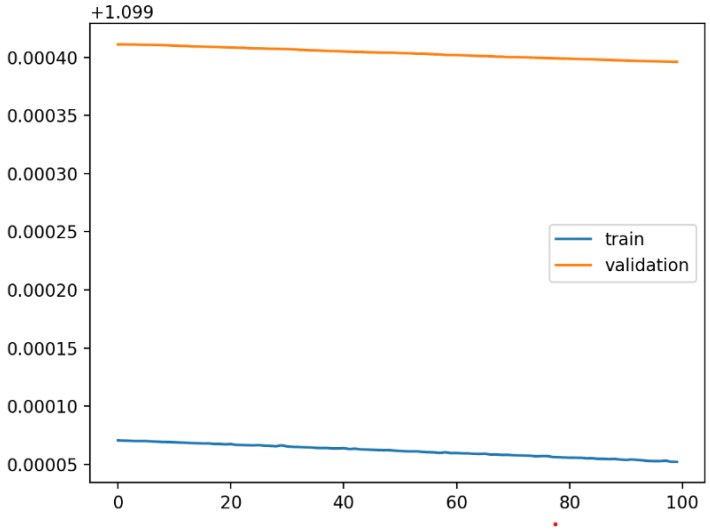
Fonte: (Brownlee, 2019)

Underfitting

Este termo é o oposto de *overfitting* e ocorre quando o modelo não é capaz de aprender com o conjunto de dados fornecido. Este comportamento pode ser observado quando existe uma curva plana de aprendizagem, quando os valores ao final do treino ainda são mais altos do que se esperava à partida ou quando as curvas de treino e validação estão muito afastadas (Jabbar & Khan, 2015).

O *underfitting* é um comportamento habitual quando o modelo não possui capacidades para aprender a tarefa determinada, ou quando o treino não foi efetuado com a quantidade de épocas adequada. A Figura 11 demostra um exemplo de uma curva de aprendizagem com comportamento de *underfitting*.

Figura 11 - Exemplo de underfitting

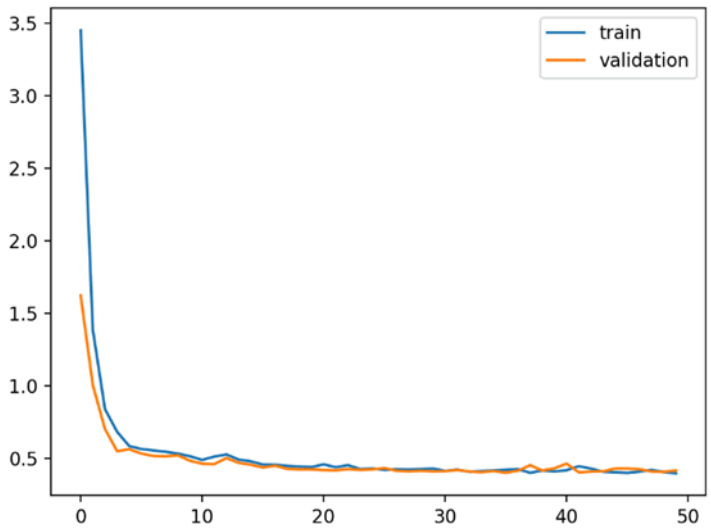


Fonte: (Brownlee, 2019)

Good Fit

Uma curva de bom comportamento acontece quando o modelo aprende corretamente os dados fornecidos, havendo um equilíbrio entre o *overfitting* e o *underfitting*. Neste caso, os valores das métricas vão melhorando gradualmente até à estabilidade, no final, onde se encontram os melhores resultados da métrica. Este comportamento é o pretendido para um treino de modelos *Transformers* sendo identificado quando existe pouca diferença entre as curvas de aprendizagem. A Figura 12 representa uma curva de aprendizagem com um ajuste adequado.

Figura 12 - Exemplo de Good Fit



Fonte: (Brownlee, 2019)

2.7.5 Avaliação de Modelos de Pergunta/Resposta

Existem duas métricas que são utilizadas com maior predominância para definir a performance de modelos de Pergunta/Resposta: a *Exact Match* (EM) e a F1. Estas métricas são calculadas para cada par Pergunta/Resposta, sendo que, quando podem existir várias respostas corretas para uma determinada pergunta, é calculada a pontuação máxima sobre todas as respostas corretas. As pontuações globais de EM e F1 são calculadas para um modelo pela média das pontuações individuais de cada pergunta.

Exact Match (EM)

Como o nome indica, esta métrica indica se a resposta prevista é igual à resposta correta depois de ambas serem normalizadas, o que é feito, habitualmente, através da transformação para minúsculas, eliminação de espaços desnecessários, remoção de artigos e, remoção de acentos e pontuação.

F1

Métrica comum para problemas de classificação e amplamente usada em tarefas de Pergunta/Resposta, sendo calculada, neste caso, sobre os átomos individuais da resposta prevista em relação à resposta correta, sendo a base do cálculo o número de

átomos comuns entre a resposta prevista e a resposta correta. De acordo com a Equação 9, a F1 é calculada usando a *precision* e a *recall*, sendo que, no caso da sua utilização em tarefas de PLN de Pergunta/Resposta, a *precision* é calculada como o quociente entre o número de átomos corretos na resposta prevista e o número total de átomos na resposta prevista (Equação 17), e o *recall* como o quociente entre o número de átomos corretos na resposta prevista e o número total de átomos na resposta correta (Equação 18).

Equação 17 – Cálculo da precision em tarefas de Pergunta/Resposta

$$precision = \frac{Número\ de\ átomos\ corretos\ na\ resposta\ prevista}{Número\ total\ de\ átomos\ na\ resposta\ prevista}$$

Equação 18 – Cálculo da recall em tarefas de Pergunta/Resposta

$$recall = \frac{Número\ de\ átomos\ corretos\ na\ resposta\ prevista}{Número\ total\ de\ átomos\ na\ resposta\ correta}$$

Os modelos de Pergunta/Resposta extrativa, como os utilizados neste trabalho, localizam a resposta correta para uma dada pergunta num determinado contexto. Quando a resposta correta não se encontra no contexto fornecido, estes modelos tendem a fornecer respostas não confiáveis das perguntas (Rajpurkar et al., 2018). Nestes casos, quando a resposta não está no contexto, o modelo pode não retornar nenhuma resposta, já que, não se encontrou o que esperava, ou então, retorna uma parte do contexto na tentativa de responder à pergunta colocada. Esta última situação, costuma ocorrer quando existe uma ou mais palavras da pergunta no contexto.

CAPÍTULO III – DADOS E METODOLOGIA

Este capítulo descreve a construção do conjunto de dados e da metodologia que foi utilizada para treino e validação de modelos, que permitiram demonstrar a capacidade atual dos modelos pré-treinados disponíveis na resposta a perguntas colocadas em linguagem natural sobre um determinado tema, neste caso, informação acerca da Fundação Nossa Senhora da Esperança (FNSE).

3.1 Dados

Foi construído um dataset para fazer o ajuste fino (*fine-tuning*) aos modelos *Transformers* escolhidos, com base na informação contida na página *web* da Fundação Nossa Senhora da Esperança (FNSE). A informação foi recolhida de forma automática através de *web scrapping*. O processo de *web scrapping* recolheu o conteúdo de cada página *web* associando-a à respetiva área do sítio, num total de 20 categorias. Como resultado deste processo, obteve-se um ficheiro Comma Separated Values (.csv) com 798 linhas e com cinco colunas: id, link de partida, categoria, link da página e conteúdo da página já limpo de todo o código HTML. A opção nesta fase pelo formato .csv, deve-se ao facto de facilitar a criação/edição manual das perguntas.

Após este processo, foram então criadas, manualmente, 265 perguntas relativas à informação contida nas páginas *web*. Procurou-se formular questões acerca das várias temáticas existentes para se ter a maior cobertura possível do domínio em causa. Cada pergunta é definida pelo seu contexto e a respetiva resposta de acordo com o formato SQuAD.

Depois de todas as perguntas estarem formuladas, o .csv foi convertido para o formato SQuAD (descrito na Secção 2.2.2) através de um *script* Python. O campo `id` do SQuAD foi preenchido com um número sequencial, o campo `context` com o contexto de onde pode ser retirada a resposta, o campo `title` com a categoria, e os campos `question` e `answers` com as perguntas criadas e com as respetivas respostas. A Figura 13 ilustra um exemplo de uma das perguntas criadas no formato SQuAD.

Figura 13 - Exemplo de uma pergunta do dataset no formato SQuAD

```
[
  {
    "id": 183,
    "context": "As despesas da Fundação são as que resultem da execução dos
presentes Estatutos.",
    "question": "De que resultam as despesas da Fundação?",
    "answers": {
      "text": [
        "execução dos presentes Estatutos"
      ],
      "answer_start": [
        47
      ]
    },
    "title": "Estatutos"
  }

]
```

Fonte: Elaboração própria

O conjunto de dados, no formato SQuAD, contabiliza em memória 10,5Kb. Na Tabela 7 pode observar-se uma análise estatística do *dataset* criado em termos de palavras, unigramas e bigramas. Em termos de palavras, o *dataset* contém 9 642 palavras (média de 36,4 por contexto) nos contextos, 2 263 nas perguntas (média de 8,5 por pergunta) e 3 666 nas respostas (média de 13,8 por resposta). Os unigramas e os bigramas foram determinados removendo todas as *stop words* e todos os números. Assim, os contextos contêm 4 335 unigramas (média de 16,4 por contexto) e 4 074 bigramas (15,4 por contexto), as perguntas contém 899 unigramas (média de 3,4 por pergunta) e 636 bigramas (média de 2,4 por resposta, enquanto as respostas contêm 1 837 unigramas (média de 6,9 por contexto) e 1 582 bigramas (média de 6 por resposta).

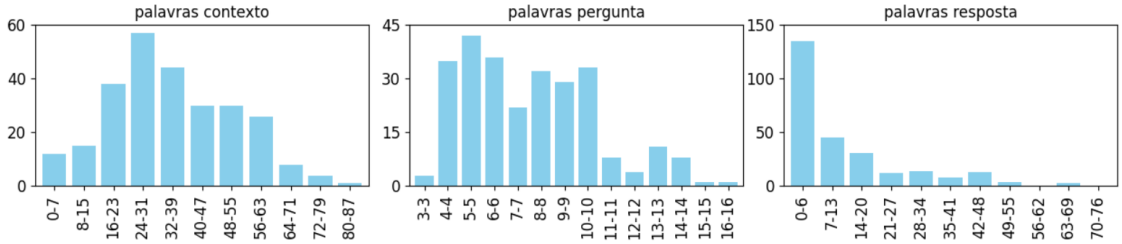
Tabela 7 - Análise estatística do dataset da FNSE

	Palavras		**Unigramas**		**Bigramas**	
	Total	**Média**	**Total**	**Média**	**Total**	**Média**
Contextos	9 642	36,4	4 335	16,4	4 074	15,4
Perguntas	2 263	8,5	899	3,4	636	2,4
Respostas	3 666	13,8	1 837	6,9	1 582	6

Fonte: Elaboração própria

A Figura 14 representa a distribuição do número de palavras nos contextos, perguntas e respostas, podendo-se observar que a maioria dos contextos têm entre 24 e 31 palavras, as perguntas entre 4 a 10 palavras e a maior parte das respostas têm no máximo 6 palavras. O eixo horizontal indica a frequência de palavras e o eixo vertical a quantidade de elementos da coluna do *dataset.*

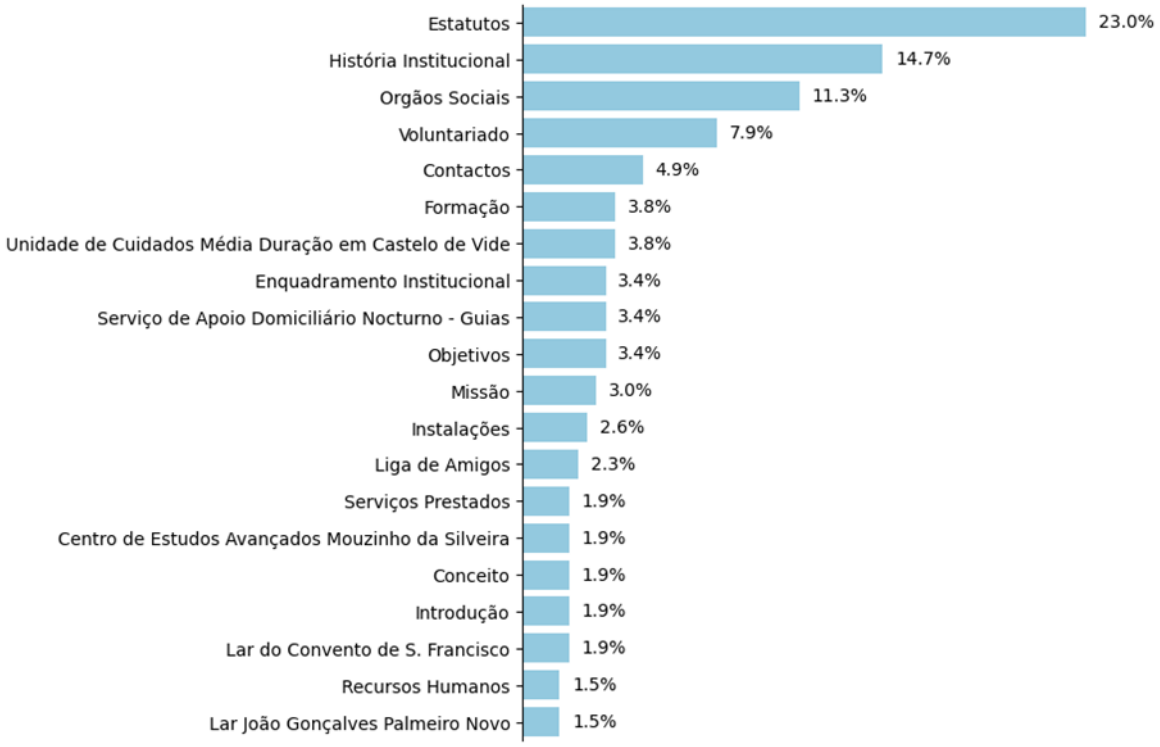
Figura 14 - Distribuição do número de palavras dos contextos, perguntas e respostas (colunas `context`, `question` e `answer`)



Fonte: Elaboração própria

A Figura 15 representa a distribuição das perguntas por cada uma das 20 categorias, sendo a categoria com mais perguntas a relativa aos Estatutos (23%), logo seguida da História Institucional (14,7%), Órgãos Sociais (11,3%) e Voluntariado (7,9%). Estas categorias representam, no total, mais de 50% das perguntas do *dataset*. Também se pode observar que as perguntas não se encontram balanceadas por categoria o, que no contexto deste trabalho, não é relevante.

Figura 15 - Distribuição das perguntas por categoria



Fonte: Elaboração própria

A Figura 16, a Figura 17 e a Figura 18, representam as nuvens de palavras dos contextos, perguntas e respostas respetivamente. Na construção destas nuvens de palavras não foram consideradas as *stop words* nem os números. A análise destas figuras demonstra que as três palavras mais comuns nos contextos são "*conselho*", "*fundação*", e "*administração*", nas perguntas, "*fnse*" (iniciais de Fundação Nossa Senhora da Esperança), "*conselho*" e "*fundação*", enquanto nas respostas as palavras mais frequentes são "*conselho*", "*administração*" e "*actividades*".

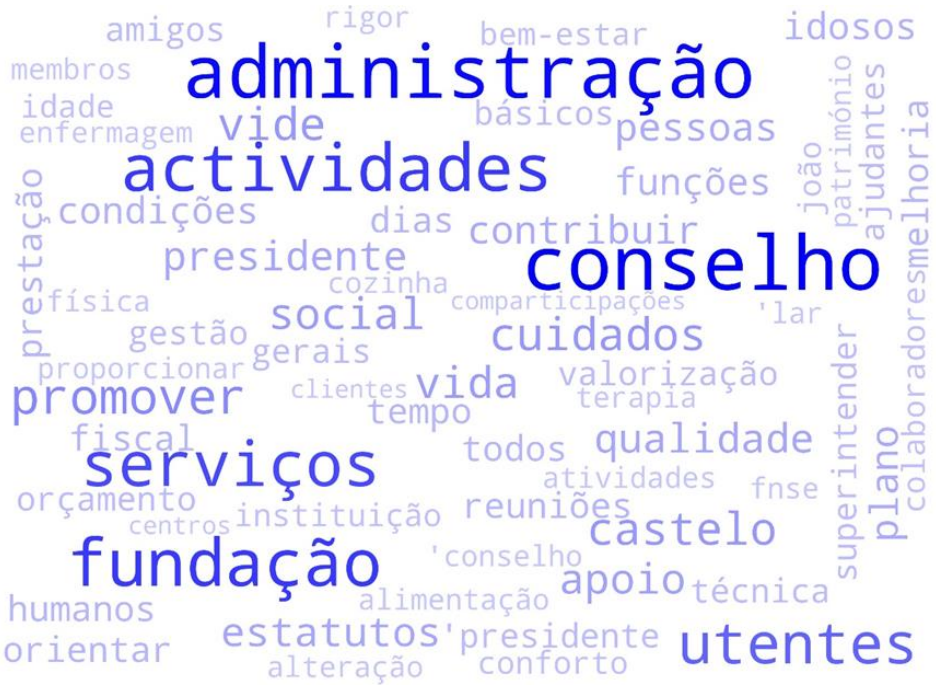
Figura 16 – Nuvem de palavras do contexto

Fonte: Elaboração própria

Figura 17 – Nuvem de palavras das perguntas

Fonte: Elaboração própria

Figura 18 – Nuvem de palavras das respostas



Fonte: Elaboração própria

3.2 Metodologia

Como referido anteriormente, o *dataset* criado foi utilizado para fazer o ajuste fino de três modelos de linguagem pré-treinados disponíveis no Hugging Face[22] e escolhidos de acordo com a sua performance em tarefas de Pergunta/Resposta em português. Esses modelos, são:

- **bert-base-cased-squad-v1.1-portuguese[23]**: modelo treinado no *dataset* SQuAD v1.1 em português, sobre o modelo de linguagem BERTimbau base (Guillou, 2021b). Este modelo obteve 0,8250 na métrica F1 e 0,7049 como *exact match* quando foi feito o treino inicial.
- **faquad-bert-base-portuguese-cased[24]**: modelo treinado durante 3 épocas no *dataset* FaQuAD[25] (Sayama et al., 2019), sobre o modelo de linguagem BERTimbau base. Foi utilizada uma taxa de aprendizagem de 3e-05, um tamanho de lote (*batch size*) de treino de 64 e 32 de avaliação, o optimizador

[22] https://huggingface.co/
[23] https://huggingface.co/pierreguillou/bert-base-cased-squad-v1.1-portuguese
[24] https://huggingface.co/eraldoluis/faquad-bert-base-portuguese-cased
[25] https://huggingface.co/datasets/eraldoluis/faquad

Adam com β_1=0,9 e β_2=0,999, e um *weight decay* de 0,01. Este modelo conseguiu obter um F1 igual a 0,83091 e 0,74532 como *exact match*.

- **bert-large-cased-squad-v1.1-portuguese**[26]: modelo treinado no *dataset* SQuAD v1.1 em português sobre o modelo de linguagem BERTimbau large (Guillou, 2021a). Este modelo obteve 0,8443 na métrica F1 e 0,7268 como *exact match* no treino com *o dataset* SQuAD.

Todos estes modelos têm como base o modelo de linguagem BERTimbau (Souza et al., 2020). O BERTimbau é um modelo BERT pré-treinado para português disponível em dois tamanhos: base (conhecido como bert-base-portuguese-cased) e large (conhecido como bert-large-portuguese-cased) com 108 334 082 e 333 348 866 parâmetros respetivamente. A Tabela 8 resume os modelos pré-treinados utilizados no contexto deste trabalho.

Tabela 8 - Modelos pré-treinados utilizados

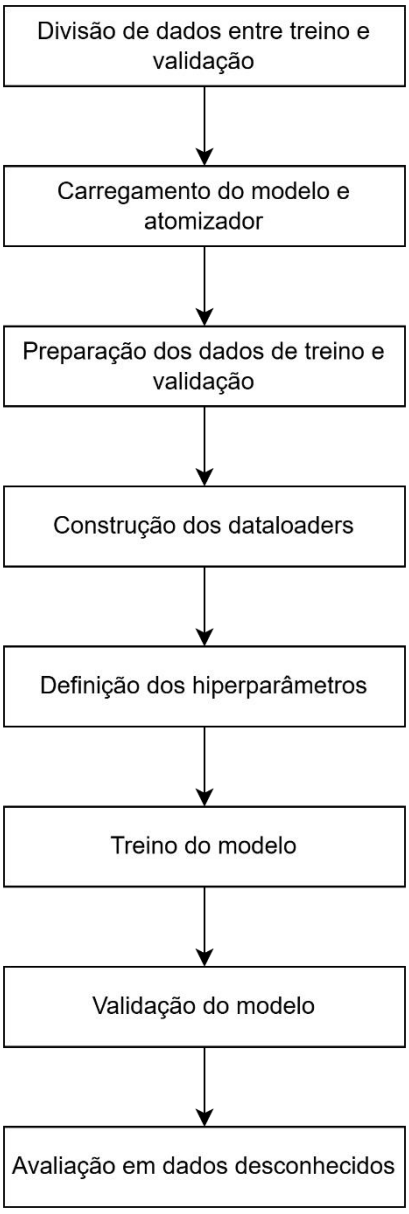
Código	Modelo	Tamanho do Vocabulário	Número de Layers	Número de Parâmetros
SQUAD-PT-BASE	bert-base-cased-squad-v1.1-portuguese	29 794	12	108 334 082
FAQUAD-PT-BASE	faquad-bert-base-portuguese-cased	29 794	12	108 334 082
SQUAD-PT-LARGE	bert-large-cased-squad-v1.1-portuguese	29 794	24	333 348 866

Fonte: Elaboração própria

A Figura 19 ilustra o *workflow* com as etapas necessárias para efetuar o treino bem como a validação de cada modelo, as quais serão descritas de seguida.

[26] https://huggingface.co/pierreguillou/bert-large-cased-squad-v1.1-portuguese

Figura 19 - Workflow usado para treino e validação dos modelos



Fonte: Elaboração própria

- **Divisão de dados entre treino e validação**. Esta etapa consiste no carregamento do *dataset* e a sua divisão entre treino e validação. A parte do treino é utilizada para efetuar o treino do modelo, para este se adaptar aos contextos e perguntas do domínio em causa. A parte de validação serve para validar se o modelo consegue responder de maneira correta acerca das perguntas, que são distintas das de treino. Foi usada a função `train_test_split` da biblioteca `sklearn.preprocessing` para fazer essa

divisão, considerando 20% dos dados para validação e 80% dos dados para treino. Neste processo de divisão também foi feita a mistura dos dados.

- **Carregamento do modelo e atomizador**. Nesta etapa são carregados o modelo pretendido e o correspondente atomizador através do método `from_pretrained` das classes `AutoModelForQuestionAnswering` e `AutoTokenizer` da biblioteca `transformers`.
- **Preparação dos dados de treino e validação**. Na preparação dos dados de treino e validação efetuou-se a atomização das perguntas e respetivos contextos, bem como o cálculo das posições em que se iniciam as respostas, para que o modelo consiga aprender com os dados. A posição da resposta considerada é a posição do átomo inicial e depois calcula-se o átomo final da resposta. Neste processo, também é validado se a resposta está enquadrada no respetivo contexto. Na preparação dos dados de validação não se têm em conta as posições das respostas, para que, o modelo não tenha esta informação e consiga efetuar uma avaliação segura.
- **Construção dos *dataloaders***. Os *dataloaders* são usados para carregar os dados de forma eficiente, sendo considerados essenciais no processo de treino e avaliação dos modelos. De entre as suas vantagens, pode-se destacar as seguintes:
    - **Leitura de dados em lotes (*batching*)**: os *dataloaders* dividem o conjunto de dados em pequenos lotes (*batches*) sendo estes carregados para memória à medida que vão sendo necessários, o que permite o suporte para grande volume de dados;
    - **Leitura assíncrona e em paralelo**: os *dataloaders* podem carregar os dados de forma assíncrona e em múltiplas *threads*, o que ajuda a manter o ciclo de treino rápido e eficiente, minimizando o tempo de espera pela leitura de dados;
    - **Embaralhamento (*shuffling*)**: os dados são embaralhados antes de cada época de treino ajudando a evitar que o modelo se ajuste demasiado aos dados de treino (*overfitting*) e melhore a generalização.

- **Definição dos hiperparâmetros**. Nesta etapa são definidas as combinações de parâmetros a utilizar nos treinos os quais se encontram resumidos na Tabela 9. No modelo SQUAD-PT-LARGE foram utilizadas menos combinações de parâmetros devido às limitações existentes a nível da GPU disponível no computador utilizado para treino.

Tabela 9 - Hiperparâmetros utilizados no treino

Parâmetro	Valor
Tamanho do conjunto de validação (tam_tst)	20%
Tamanho máximo da sequência (max_length)	400
Stride	100
Gradientes acumulados	3
Número de passos de aquecimento (num_warmup_steps)	1
Quantidade de respostas a considerar (n_best)	2
Tamanho máximo da resposta (max_resp_tam)	400
Optimizador	[SGD, AdamW] SGD, valores padrão AdamW, $\beta 1$=0.9, $\beta 2$=0.999, weight decay=0.01
Taxa de aprendizagem (learning_rate)	[1e-04, 1e-05, 2e-05]
Modelos	[SQUAD-PT-BASE, FAQUAD-PT-BASE, SQUAD-PT-LARGE]
Número de épocas de treino (num_epochs)	SQUAD-PT-BASE e FAQUAD-PT-BASE: [3, 5, 6, 8, 10, 15, 20] SQUAD-PT-LARGE: [3, 5, 6, 8, 10, 15]
Tamanho do batch (batch_size)	SQUAD-PT-BASE e FAQUAD-PT-BASE: [4, 8, 10, 16, 32, 48] SQUAD-PT-LARGE: [4, 8, 10, 16, 20]

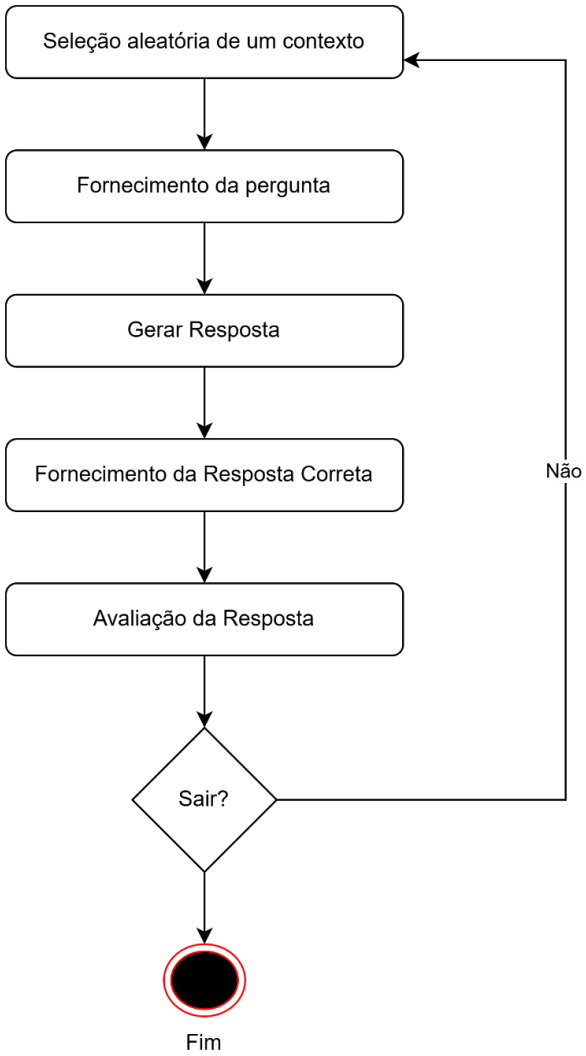
Fonte: Elaboração própria

- **Treino do modelo**. Nesta etapa são efetuados os treinos dos modelos para cada combinação de hiperparâmetros definida. No treino, primeiro coloca-se o modelo no modo `train` e faz-se o respetivo treino com 80% dos dados, como anteriormente definido. A *loss* é a métrica utilizada no ajuste de pesos do treino para que a sua taxa diminua ao longo do processamento. Os pesos, são então melhorados com o optimizador AdamW com uma taxa de aprendizagem de [1e-04, 1e-05, 2e-05], β_1=0,9 e β_2=0,999 e um *weight decay* de 0,01.
- **Validação do modelo.** Depois do modelo treinado, é efetuada a validação com os restantes 20% dos dados colocando-se o modelo no modo `eval`. Os *logits* (pontuações brutas produzidas pela camada de saída do modelo) são armazenadas para depois se calcular as métricas de validação. Para cada pergunta é calculada a melhor resposta (tendo em conta a pontuação de cada uma) que depois é levada para o cálculo das métricas F1*, accuracy e loss.* Para o cálculo foi necessário definir os rótulos com a ajuda da função `LabelBinazer` para posterior comparação das respostas previstas com as respostas originais.
- **Avaliação em Dados Desconhecidos:** Posteriormente à validação do modelo é feita uma avaliação adicional com perguntas desconhecidas, utilizando o modelo que conduziu a melhores resultados. Para o efeito, deve ser criada uma aplicação que siga os seguintes passos (Figura 20):

    1. Seleção aleatória de um contexto: Aqui é selecionado um dos contextos existentes no *dataset* criado com os dados da FNSE de maneira aleatória.
    2. Fornecimento da pergunta: Neste passo, o utilizador terá de fornecer uma pergunta que pode ou não ter a resposta no contexto.
    3. Gerar resposta: Nesta etapa, o modelo irá tentar gerar uma resposta tendo por base o contexto e a pergunta realizada. É de notar, que o modelo procura a resposta no contexto que foi selecionado no passo 1, portanto, caso a resposta não esteja nesse contexto, o modelo não conseguirá dar a resposta esperada.

4. Fornecimento da resposta correta: Neste momento, o utilizador deve fornecer a resposta que esperava que fosse dada pelo modelo, que pode ou não ser a mesma que a resposta já gerada pelo modelo no passo 3.
5. Avaliação da resposta. Aqui, são comparadas as respostas existentes nos passos 3 e 4 para se verificar se são iguais. A aplicação construída, avalia as respostas pelas métricas F1 e *exact match* (explicadas na Secção 2.7.5), sendo que, quanto mais próximas as respostas forem melhores também serão as métricas. Neste passo, também é obtido o *score* da resposta prevista pelo modelo que deverá ter um valor entre 0 e 1, a depender da confiança que o modelo teve na geração da resposta.

Figura 20 - Fluxo para a Previsão com Dados Desconhecidos



Fonte: Elaboração própria

CAPÍTULO IV – RESULTADOS OBTIDOS

Após a criação do *dataset* descrito no capítulo anterior, foram construídos testes de acordo com a metodologia referida, igualmente, no capítulo anterior.

Todo o código utilizado no treino e validação dos modelos foi escrito em Python usando várias bibliotecas, sendo as mais relevantes, a numpy 1.24.3 (Harris et al., 2020), a pandas 2.1.0 para análise de dados (McKinney, 2010; The pandas development team, 2020), a matplotlib 3.8.0 para visualização de dados (Hunter, 2007), a scikit-learn 1.3.0 (Pedregosa et al., 2012), a torch 2.0.1 (Paszke et al., 2019) e a transformers 4.28.0 (Wolf et al., 2019) para tarefas e algoritmos de treino e avaliação dos modelos. Este código foi executado no sistema operativo Ubuntu num computador NVIDIA DGX Station com 2 CPUs Intel Xeon E5-2698V4, 20 núcleos de 2,2 GHz, 256 GB de memória e 4 GPUs NVIDIA Tesla V100 com capacidade de processamento de 500 Tera Flops em vírgula flutuante.

Os testes realizados com o optimizador SGD conduziram a valores das métricas bastante inferiores ao AdamW, o que é consistente com vários estudos que apontam para a melhor performance do AdamW. Como exemplo, no melhor resultado do SGD para o modelo SQUAD-PT-BASE, conseguiu-se uma *loss* nos dados de validação de 1,5597, um F1 0,3825 e uma *accuracy* de 0,5800, valores francamente inferiores aos do AdamW. Por este motivo, no resto deste capítulo irão ser considerados apenas os resultados obtidos com o AdamW.

No total foi efetuado o treino de 3 117 épocas correspondentes a 342 combinações distintas dos hiperparâmetros, nomeadamente, o tamanho do lote (`batch_size`), o número de épocas (`num_epochs`) e a taxa de aprendizagem (`learning_rate`) num tempo total de treino de 10:21:14. Para cada época treinada foi guardada informação, com os resultados obtidos, nomeadamente a *loss*, a F1 e a *accuracy*, quer sobre os dados de treino, quer sobre os dados de validação. A Tabela 10 ilustra a informação recolhida com os 10 primeiros registos. Esta informação permitiu construir, entre outras, as curvas de aprendizagem e as análises que ajudam a explicar o comportamento de cada um dos modelos considerados (SQUAD-PT-BASE, FAQUAD-PT-BASE e SQUAD-PT-LARGE), as quais irão ser feitas de seguida.

No final deste capítulo, são ainda apresentados os resultados obtidos com o modelo que conduziu aos melhores resultados, fazendo a predição sobre dados, de acordo com o referido na última etapa do *workflow* da Figura 19.

Tabela 10 - Dez primeiros registos do treino dos modelos

Model	Start	learning rate	batch	num epochs	epoch	loss (train)	loss (validation)	F1 (train)	F1 (validation)	accuracy (train)	accuracy (validation)	time (secs)
SQUAD-PT-BASE	20/05/2024 14:15	1e-4	4	3	1	0,72727	0,28932	0,77643	0,82979	0,85377	0,92308	14
SQUAD-PT-BASE	20/05/2024 14:15	1e-4	4	3	2	0,15539	0,28932	0,95859	0,82979	0,98585	0,92308	13
SQUAD-PT-BASE	20/05/2024 14:15	1e-4	4	3	3	0,03539	0,28932	0,98415	0,82979	0,98585	0,92308	13
SQUAD-PT-BASE	20/05/2024 14:15	1e-5	4	3	1	0,90917	0,72331	0,71620	0,64744	0,79245	0,80769	14
SQUAD-PT-BASE	20/05/2024 14:15	1e-5	4	3	2	0,39266	0,43973	0,88144	0,76389	0,92925	0,88235	14
SQUAD-PT-BASE	20/05/2024 14:16	1e-5	4	3	3	0,26529	0,43973	0,91533	0,76389	0,92925	0,88235	13
SQUAD-PT-BASE	20/05/2024 14:16	2e-5	4	3	1	0,78329	0,72331	0,75638	0,64744	0,83019	0,80769	14
SQUAD-PT-BASE	20/05/2024 14:16	2e-5	4	3	2	0,20842	0,35700	0,95679	0,80952	0,96698	0,90566	14
SQUAD-PT-BASE	20/05/2024 14:16	2e-5	4	3	3	0,10331	0,21699	0,98020	0,86957	0,98585	0,94231	14
SQUAD-PT-BASE	20/05/2024 14:17	1e-4	8	3	1	0,75246	0,57120	0,70810	0,72549	0,83019	0,84906	9

Fonte: Elaboração própria

4.1 Modelo SQUAD-PT-BASE

Para este modelo, foi efetuado o treino de 1 206 épocas correspondentes a 126 combinações distintas do tamanho do lote (`batch_size`), número de épocas (`num_epochs`) e taxa de aprendizagem (`learning_rate`) num tempo total de treino de 2:51:28.

O Anexo 2, o Anexo 3 e o Anexo 4, apresentam as curvas de aprendizagem para cada uma destas 126 combinações dos hiperparâmetros. O Anexo 2 corresponde aos testes com uma taxa de aprendizagem igual a 1e-04, o Anexo 3 a uma taxa de aprendizagem de 1e-05 e o Anexo 4 a uma taxa de aprendizagem de 2e-05. Em cada anexo, as colunas representam as curvas de aprendizagem para cada tamanho de lote (4, 8, 10, 16, 32 e 48) e as linhas o número de épocas considerado (3, 5, 6, 8, 10, 15 e 20) respetivamente. O melhor valor da *loss* de validação (valor mais baixo) encontra-se representado com um traço mais grosso para se poder destacar mais facilmente dos restantes.

A Tabela 11 mostra os melhores valores não só da *loss* (valor mais baixo), mas também da F1 (valor mais alto) e da *accuracy* (valor mais alto), bem como as combinações dos hiperparâmetros (número de épocas, tamanho do lote e taxa de aprendizagem) que conduziram a esses valores.

Tabela 11 - Melhores resultados da validação do modelo SQUAD-PT-BASE

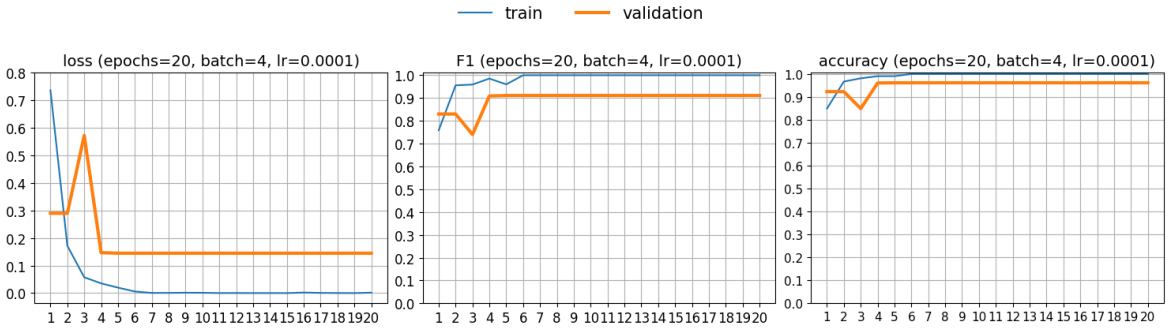
Métrica	Melhor valor	Hiperparâmetros [(num_epochs, batch_size, learning_rate), ...]
loss (validação)	0,14466	[(20, 4, 1e-4)]
F1 (validação)	0,91111	[(20, 4, 1e-4)]
accuracy (validação)	0,96154	[(20, 4, 1e-4)]

Fonte: Elaboração própria

Como se pode observar, a partir desta tabela, neste modelo, os melhores valores das métricas foram obtidos com a mesma combinação dos hiperparâmetros, nomeadamente um número de épocas igual a 20, um tamanho de lote igual 4 e uma taxa de aprendizagem igual a 1e-4.

A Figura 21 representa as curvas de aprendizagem da métrica *loss*, F1 e *accuracy* ao longo das várias épocas para o conjunto de dados de treino (representados a azul) e de validação (representados a laranja), para este mesmo conjunto de hiperparâmetros.

Figura 21 - Evolução das métricas ao longo das épocas para os melhor resultado (SQUAD-PT-BASE)



Fonte: Elaboração própria

Na observação desta figura, pode verificar-se, na terceira época de validação, um pico na *loss,* voltando esta a diminuir estando as curvas de treino e de validação relativamente próximas e constantes a partir da sétima época. O pico existente demonstra que o ajuste da curva de validação não foi o mais adequado.

Os valores obtidos da F1 e da *accuracy* de validação são bastante bons (0,91111 e 0,96154 respetivamente) o que indica que o modelo conseguiu aprender o *dataset* com que foi treinado.

4.2 Modelo FAQUAD-PT-BASE

Para este modelo, foi igualmente efetuado o treino de 1 206 épocas correspondentes a 126 combinações distintas do tamanho do lote (`batch_size`), número de épocas (`num_epochs`) e taxa de aprendizagem (`learning_rate`) num tempo total de treino de 2:46:48.

O Anexo 5, o Anexo 6 e o Anexo 7, apresentam as curvas de aprendizagem para cada uma destas 126 combinações dos hiperparâmetros. O Anexo 5 corresponde aos testes com uma taxa de aprendizagem igual a 1e-04, o Anexo 6 a uma taxa de aprendizagem de 1e-05 e o Anexo 7 a uma taxa de aprendizagem de 2e-05. Em cada anexo, as colunas representam as curvas de aprendizagem para cada tamanho de

lote (4, 8, 10, 16, 32 e 48) e as linhas o número de épocas considerado (3, 5, 6, 8, 10, 15 e 20) respetivamente. O melhor valor da *loss* de validação (valor mais baixo) encontra-se representado com um traço mais grosso para se poder destacar mais facilmente dos restantes.

A Tabela 12 mostra os melhores valores não só da *loss* (valor mais baixo), mas também, da F1 (valor mais alto) e da *accuracy* (valor mais alto), bem como as combinações dos hiperparâmetros (número de épocas, tamanho do lote e taxa de aprendizagem) que conduziram a esses valores.

Tabela 12 - Melhores resultados da validação do modelo FAQUAD-PT-BASE

Métrica	**Melhor valor**	**Hiperparâmetros [(num epochs, batch size, learning rate), ...]**
loss (validação)	0,21699	[(10, 16, 1e-4), (15, 16, 1e-4), (20, 10, 01e-4)
F1 (validação)	0,88406	[(10, 16, 1e-4), (15, 16, 1e-4)]
accuracy (validação)	0,94231	[(10, 16, 1e-4), (15, 16, 1e-4), (20, 10, 1e-4)]

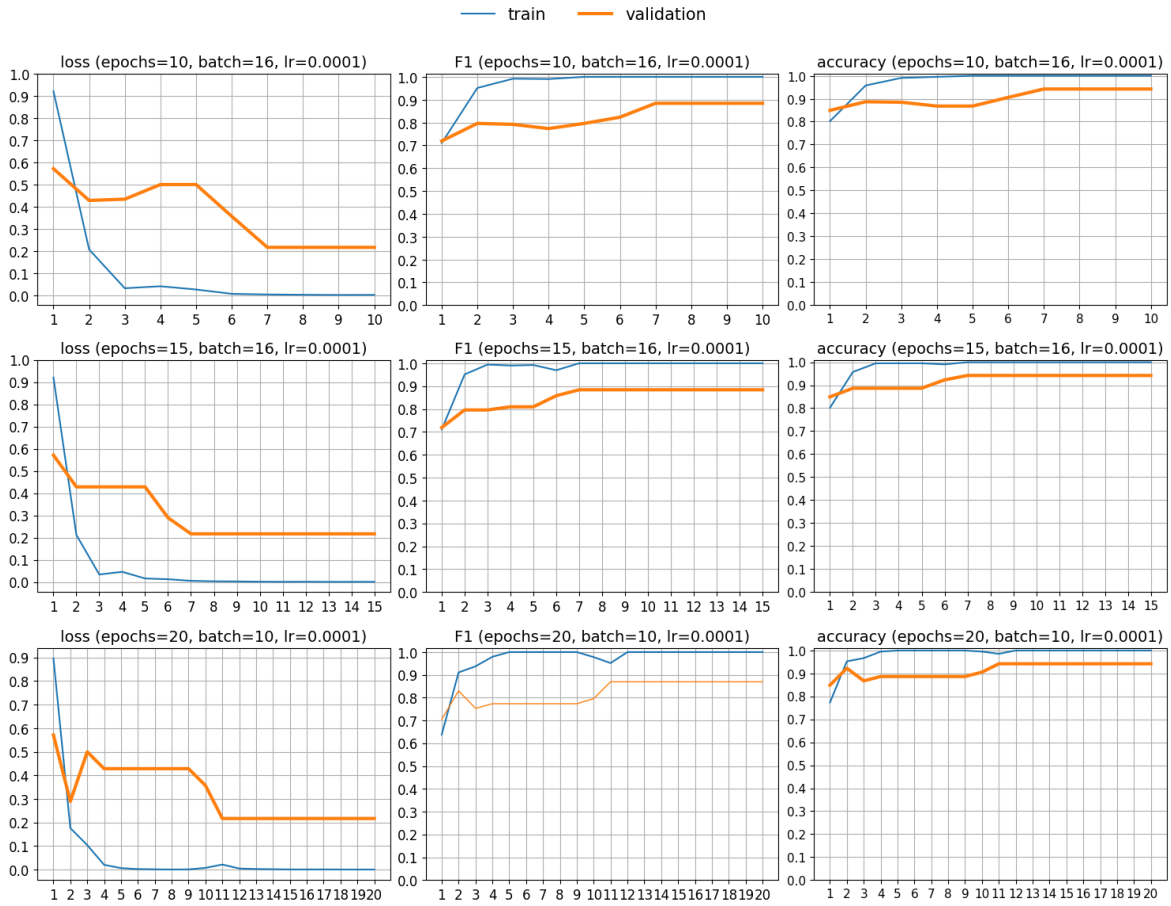
Fonte: Elaboração própria

Com o modelo FAQUAD-PT-BASE também foi possível obter métricas algo próximas do ideal, principalmente, no que toca às métricas F1 e *accuracy*. A métrica *loss* ainda tem um valor ligeiramente distante de 0, mas mesmo assim não deixa de ter um valor aceitável.

Como se pode observar na tabela anterior, o melhor valor da *loss* de validação (0,21699) verificou-se em três combinações dos hiperparâmetros.

A Figura 22 representa a evolução do *loss*, F1 e *accuracy* ao longo das épocas para o conjunto de dados de treino (representados a azul) e de validação (representados a laranja).

Figura 22 - Evolução das métricas ao longo das épocas para o melhor resultado (FAQUAD-PT-BASE)



Fonte: Elaboração própria

Cada linha da figura anterior representa uma combinação diferente dos hiperparâmetros as quais serão comentadas de seguida.

Para a combinação de 10 épocas de treino, 16 de tamanho do lote e 1e-4 de taxa de aprendizagem (primeira linha), o gráfico da métrica *loss* gerou curvas de aprendizagem distantes ao longo do treino, sendo que, apenas na sétima época, é que as curvas conseguiram aproximar-se, assim, este não é o melhor treino do modelo.

Para a combinação de 15 épocas de treino, 16 de tamanho do lote e 1e-4 de taxa de aprendizagem (segunda linha), é possível verificar que as curvas de aprendizagem estiveram sempre com valores próximos ao longo de todo o treino, nas três métricas de validação. Este treino obteve os melhores valores finais para todas as métricas o que prova que é o melhor treino a considerar para o modelo FAQUAD-PT-BASE.

Para a combinação de 20 épocas de treino, 10 de tamanho do lote e 1e-4 de taxa de aprendizagem (terceira linha), as métricas F1 e *loss* mostraram curvas de aprendizagem mais distantes para o treino e validação, em relação às combinações anteriores. A métrica F1 não gerou os melhores valores para o treino com o modelo FAQUAD-PT-BASE, para além disso, a curva não obteve um bom ajuste, havendo picos na curva de validação, nas épocas iniciais. Desta forma, este não é o melhor treino a considerar para o modelo FAQUAD-PT-BASE. Este modelo, necessitou de valores de tamanho do lote maiores, em comparação com o anterior, para se conseguir chegar aos melhores valores de treino.

4.3 Modelo SQUAD-PT-LARGE

Para este modelo, foi efetuado o treino de 705 épocas correspondentes a 90 combinações distintas do tamanho do lote (`batch_size`), número de épocas (num_epochs) e taxa de aprendizagem (`learning_rate`) num tempo total de treino de 4:42:58.

O Anexo 7, o Anexo 8 e o Anexo 9, apresentam as curvas de aprendizagem para cada uma destas 90 combinações dos hiperparâmetros. O Anexo 7 corresponde aos testes com uma taxa de aprendizagem igual a 1e-04, o Anexo 8 a uma taxa de aprendizagem de 1e-05 e o Anexo 9 a uma taxa de aprendizagem de 2e-05. Em cada anexo, as colunas representam as curvas de aprendizagem para cada tamanho de lote (4, 8, 10, 16 e 20) e as linhas o número de épocas considerado (3, 5, 6, 8, 10 e 15) respetivamente. O melhor valor da *loss* de validação (valor mais baixo) encontra-se representado com um traço mais grosso para se poder destacar mais facilmente dos restantes.

A Tabela 13 mostra os melhores valores não só da *loss* (valor mais baixo), mas também, da F1 (valor mais alto) e da *accuracy* (valor mais alto), bem como as combinações dos hiperparâmetros (número de épocas, tamanho do lote e taxa de aprendizagem) que conduziram a esses valores.

Tabela 13 - Melhores resultados da validação do modelo SQUAD-PT-LARGE

Métrica	Melhor Valor	Hiperparâmetros [(num epochs, batch size, learning rate), ...]
loss (validação)	0,21699	[(5, 8, 1e-4)]
F1 (validação)	0,86957	[(5, 8, 1e-4)]
accuracy (validação)	0,94231	[(5, 8, 1e-4)]

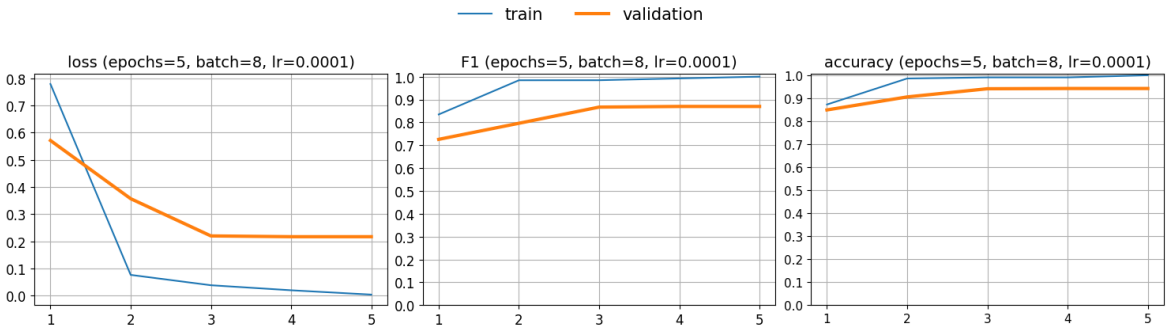
Fonte: Elaboração própria

Como se pode observar os melhores valores das métricas foram obtidos com a mesma combinação dos hiperparâmetros, nomeadamente, um número de épocas igual a 5, um tamanho de lote igual 8 e uma taxa de aprendizagem igual a 1e-4.

Neste treino conseguiram-se valores da F1 e da *accuracy* de validação de 0,86957 e 0,94231 respetivamente, o que comprova que, este treino consegue fornecer respostas bastante exatas para o *dataset* com o qual foi treinado.

A Figura 23 representa as curvas de aprendizagem da *loss*, F1 e *accuracy* ao longo das várias épocas para o conjunto de dados de treino (representados a azul) e de validação (representados a laranja), para este mesmo conjunto de hiperparâmetros.

Figura 23 - Evolução das métricas ao longo das épocas para o melhor resultado (SQUAD-PT-LARGE)



Fonte: Elaboração própria

Neste treino, todas as métricas obtiveram resultados satisfatórios. As métricas F1 e *accuracy* alcançaram valores muito perto do 1, sendo que, a *accuracy* teve um valor de 0,94231 que é bastante alto. A curva da métrica *loss* não obteve resultados tão próximos de 0 como seria o ideal em comparação com o resultado obtido para o modelo SQUAD-PT-BASE.

As curvas de aprendizagem demonstram um comportamento de ajuste adequado, tendo em conta que, a curva de treino está próxima da de validação ao longo de todo o treino. Desta forma, este é o melhor treino a considerar para o modelo SQUAD-PT-LARGE.

4.4 Avaliação em Dados Desconhecidos

Como referido no último passo da Figura 19, foi construída uma aplicação em Python para se poder fazer a avaliação com dados desconhecidos, sendo ilustrado na Figura 24 um exemplo de output da mesma.

Figura 24 - Exemplo de interação com o utilizador para criação de perguntas

```
---------------- Pergunta nº 2 ----------------

CONTEXTO:
 A Fundação tem a sua sede na Rua Sequeira Sameiro, na Freguesia de São João
Baptista de Castelo de Vide

--> Escreva uma pergunta: Em que rua fica a FNSE?

Resposta prevista: Rua Sequeira Sameiro
Score obtido:0.9552

--> Indique a resposta verdadeira: Rua Sequeira Sameiro

---MÉTRICAS:---
  exact match:100.0
  f1:         100.0


--> Deseja sair? (S/N): N
```

Fonte: Elaboração Própria

No total, foram feitas 105 perguntas as quais ficaram guardadas num ficheiro .csv com os resultados. Este ficheiro contém as colunas: contexto gerado aleatoriamente a partir do *dataset* da FNSE (coluna `context`), pergunta efetuada pelo utilizador (coluna `question`), resposta prevista pelo modelo (coluna `predicted`), resposta verdadeira considerada pelo utilizador (`actual`), F1, *exact match* e *score* (*score* obtido para a resposta prevista).

Os resultados obtidos foram de 0,8490 de F1, 0,8286 de *exact match* e *score* de 0,8306, considerando todas as perguntas feitas, como na Tabela 14.

Tabela 14 - Resultados médios obtidos na avaliação com dados desconhecidos

Métrica	**Valor**
Total de perguntas	105
F1	0,8490
Exact match	0,8286
Score médio	0,8306

Fonte: Elaboração Própria

Os resultados apresentados na tabela indicam que, o modelo foi capaz de responder de forma adequada às perguntas colocadas. As previsões feitas pelo modelo mostraram-se muito próximas das respostas verdadeiras, como evidenciado pelas métricas F1 e *exact match*. Além disso, o modelo demonstrou muita confiança nas respostas geradas, o que é refletido pelo elevado valor do *score* médio. Como verificado, a maioria das perguntas foram corretamente respondidas, o que resulta em valores de métricas próximas dos valores máximos.

As métricas, estiveram ligeiramente abaixo do que foi obtido na validação, já que, desta vez, foram feitas 105 questões, comparadas às 53 perguntas que faziam parte dos dados de validação. Desta vez, também foram efetuadas perguntas que a resposta estava fora do contexto o que causou valores inferiores aos já obtidos.

Foram feitas 12 perguntas em que a resposta não estava no contexto, sendo que, nestas perguntas o modelo ou não previu resposta, ou retornou uma resposta que não era a esperada. Além disso, deve-se considerar que as perguntas que deveriam ter como resposta "sim" ou "não", terão outro tipo de resposta prevista ou, não serão respondidas. No entanto, o modelo conseguiu retornar uma resposta correta na maior parte das perguntas colocadas, o que demostra uma boa capacidade da generalização dos dados.

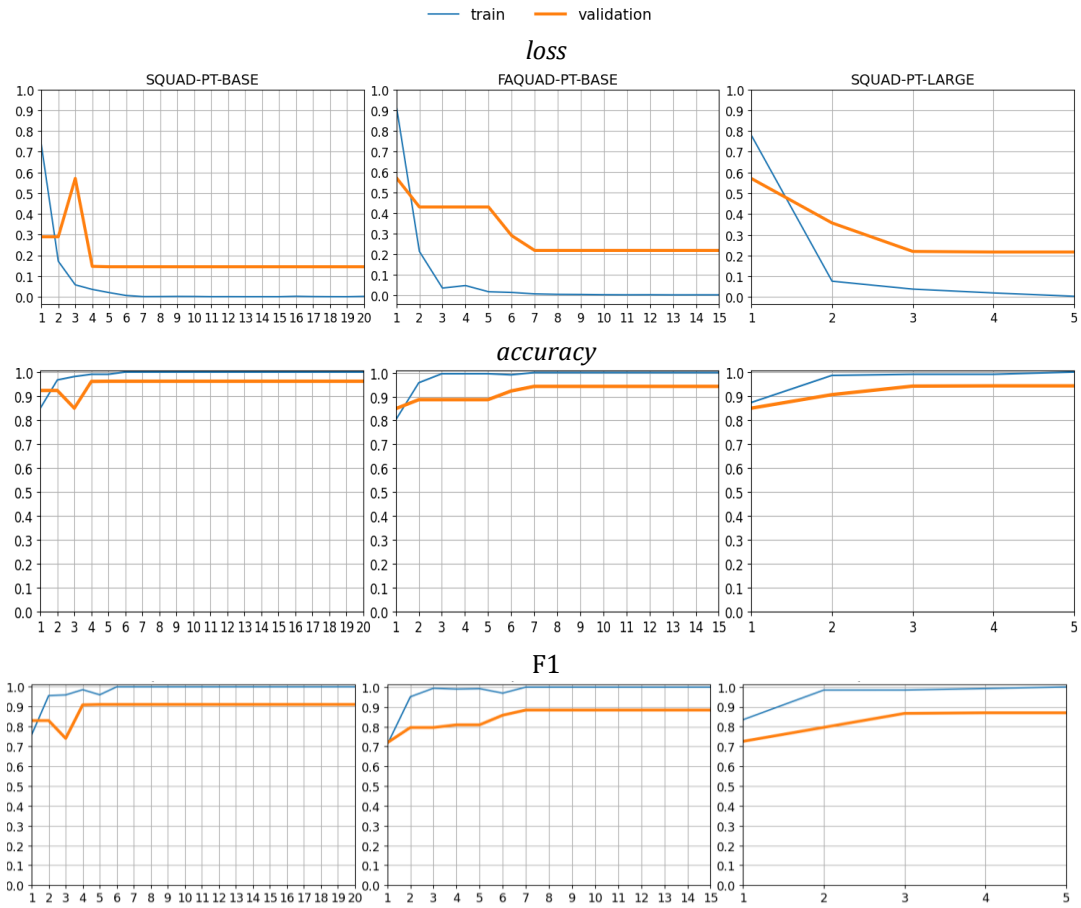
CAPÍTULO V - DISCUSSÃO DOS RESULTADOS OBTIDOS

Neste capítulo, pretende-se discutir e comparar os treinos realizados no Capítulo IV para validar a capacidade dos modelos em tarefas de Pergunta/Resposta. Os resultados obtidos no melhor modelo, serão ainda comparados com os resultados de outros modelos *Transformers* que se encontram publicamente disponíveis para este tipo de tarefa de PLN em Língua Portuguesa.

5.1 Comparação das Curvas de Aprendizagem Obtidas nos Treinos

A Figura 25 apresenta as curvas de aprendizagem obtidas nas métricas *loss, accuracy* e F1 para a combinação de hiperparâmetros que conduziu aos melhores resultados de cada um dos modelos, conforme descrito na Secção 4.1, 4.2 e 4.3. A linha azul representa a curva de treino e a laranja a curva de validação.

Figura 25 - Curvas de aprendizagem para as métricas *loss*, *accuracy* e F1 para os melhores resultados de cada modelo



Fonte: Elaboração própria

O modelo SQUAD-PT-BASE (primeira coluna) apresenta curvas de aprendizagem de treino e validação bastante próximas ao longo do processo, sendo que, este é o modelo com as curvas mais próximas do valor ideal para as três métricas de validação. Contudo, foi possível observar que, na terceira época existe um pico na curva de validação, que é mais evidente na métrica *loss* (primeira linha), o que sugere alguma dificuldade do modelo na validação dos dados. No entanto, na quarta época as curvas de treino e validação ficaram próximas o que demonstra um ajuste adequado das curvas de aprendizagem.

O modelo FAQUAD-PT-BASE (segunda coluna), possui as curvas de aprendizagem de treino e validação um pouco afastadas ao longo do treino, principalmente na métrica *loss*. No entanto, as curvas melhoram os valores gradativamente, o que mostra um bom ajuste das curvas. Este modelo demorou sete épocas a conseguir alcançar o seu melhor valor o que mostra a robustez do modelo e a capacidade de generalizar os dados.

O modelo SQUAD-PT-LARGE (terceira coluna) demonstra as curvas de aprendizagem de treino e de validação bastante próximas ao longo de todo o treino. Na figura, é possível observar que, as curvas evoluem gradualmente até ser possível o alcance dos melhores resultados. Este modelo é maior que os restantes, já que, foi pré-treinado com um *dataset* maior, comparativamente aos modelos SQUAD-PT-BASE e FAQUAD-PT-BASE. Deve-se notar que, o modelo SQUAD-PT-LARGE conseguiu alcançar os melhores resultados com um treino bastante curto, o que revela que, o modelo aprende rapidamente um *dataset* curto em português. Nas métricas *accuracy* e F1, é possível notar as curvas muito próximas o que demonstra a exatidão e precisão do modelo na previsão das respostas.

De maneira resumida, o modelo SQUAD-PT-BASE destaca-se por apresentar as curvas de aprendizagem mais próximas do ideal, em comparação com os modelos FAQUAD-PT-BASE e SQUAD-PT-LARGE. Contudo, existiu alguma dificuldade na interpretação do comportamento das curvas de aprendizagem, principalmente, no modelo SQUAD-PT-BASE, que obteve um pico na terceira época, mas posteriormente conseguiu melhorar os valores obtidos, aproximando as curvas.

5.2 Comparação dos Resultados Obtidos com Outros Treinos

Com o objetivo de identificar as capacidades dos modelos *Transformers* na execução da tarefa de Pergunta/Resposta, decidiu-se procurar por outros modelos capazes de desempenhar o mesmo tipo de tarefa.

Na secção 3.1 deste trabalho, identificaram-se os modelos bert-base-cased-squad-v1.1-portuguese[23] e bert-large-cased-squad-v1.1-portuguese[26] que foram pré-treinados com o *dataset* SQuAD v1.1 em português, e o modelo faquad-bert-base-portuguese-cased[24] que foi treinado com o *dataset* FaQuAD[25] (Guillou, 2021a, 2021b). Estes modelos serviram como base para este trabalho.

Para além destes modelos, também considerámos, para comparação, o modelo AlbertinaPT, que é baseado no modelo DeBERTa (referido na secção 2.6.2) (He et al., 2020). O AlbertinaPT foi treinado com dados exclusivamente em português, incluindo informação do *dataset* OSCAR[27], que é um grande corpus que incluí textos em várias línguas, incluindo em português de diversas páginas da *web*. Este modelo é capaz de executar diversas tarefas de Processamento de Linguagem Natural, desde a análise de sentimentos até a tarefas de Pergunta/Resposta (Santos et al., 2024).

Considerámos ainda nesta comparação, dois modelos baseados no modelo bert-base-cased-squad-v1.1-portuguese e bert-large-cased-squad-v1.1-portuguese, em que, foram aplicados outros hiperparâmetros e a técnica de *Overlapping Window* com o objetivo de aumentar o limite de átomos, ajustar a taxa de aprendizagem e conduzir a melhores resultados. A técnica de *Overlapping Window* divide um texto em várias partes para garantir que as informações importantes não sejam perdidas (Silva et al., 2022). Foi feito o ajuste fino destes modelos com o *dataset* SQuAD-BR, que é uma versão do SQuAD v.1.1 traduzido para português do Brasil pelo grupo de investigação Deep Learning Brazil[28].

A Tabela 15 contém os resultados obtidos nas métricas de avaliação *exact match* e F1, para que se possa comparar os vários valores obtidos dos modelos considerados.

[27] https://oscar-project.org/
[28] http://www.deeplearningbrasil.com.br/

Tabela 15 - Resultados de métricas obtidos na avaliação de modelos de Pergunta/Resposta

Referência	Modelo	Exact match	F1
(Guillou, 2021b)	bert-base-cased-squad-v1.1-portuguese[23]	0,7049	0,8250
(Sayama et al., 2019)	faquad-bert-base-portuguese-cased[24]	0,7453	0,8309
(Guillou, 2021a)	bert-large-cased-squad-v1.1-portuguese[26]	0,7268	0,8443
(Junqueira et al., 2023)	Albertina PT-BR Base	0,4512	0,5700
(Junqueira et al., 2023)	Albertina PT-BR Large	0,4730	0,3200
(Silva et al., 2022)	bert-base-cased-squad-v1.1-portuguese melhorado	0,7109	0,8291
(Silva et al., 2022)	bert-large-cased-squad-v1.1-portuguese melhorado	0,7312	0,8474
Nosso melhor modelo	SQUAD-PT-BASE	**0,8286**	**0,8490**

Fonte: Elaboração própria

Como se observa pela tabela, os modelos que fazem parte da família BERT (bert-base-cased-squad-v1.1-portuguese, faquad-bert-base-portuguese-cased, bert-base-cased-squad-v1.1-portuguese, bert-large-cased-squad-v1.1-portuguese e SQUAD-PT-BASE) apresentaram melhores resultados. O modelo bert-large-cased-squad-v1.1-portuguese obteve o seu melhor valor quando foram implementadas melhorias, alcançando 0,7312 de F1 e 0,8474 de *exact match*. Para além disso, o modelo faquad-bert-base-portuguese-cased demonstrou um desempenho semelhante com valores de 0,7453 de *exact match* e 0,8309 de F1, o que evidencia os resultados já obtidos para modelos da família BERT.

Por outro lado, os modelos que são baseados no DeBERTa demonstraram resultados bastante inferiores. Nomeadamente, o Albertina PT-BR Base que obteve 0,4512 de *exact match* e 0,5700 como F1 e o modelo Albertina PT-BR Large que alcançou 0,4730 de *exact match* e 0,3200 de F1 o que demonstra um baixo desempenho destes modelos na tarefa de pergunta resposta.

O modelo SQUAD-PT-BASE, treinado neste trabalho com o *dataset* FNSE, foi o que se destacou na tabela pelos resultados que conseguiu obter, conseguindo 0,8286 de *exact match* e 0,8490 de F1. Estes valores demonstram que, este modelo foi capaz de alcançar um desempenho superior aos restantes na tarefa de Pergunta/Resposta, já que, ele consegue retornar respostas muito precisas e corretas.

CAPÍTULO VI - CONCLUSÃO

Este capítulo apresenta a conclusão do trabalho, começando por ser feita uma síntese da motivação que levou à sua realização, sendo apresentadas, de seguida as principais contribuições, as conclusões relativas aos resultados obtidos, e, por fim, são apresentadas algumas considerações relativas ao trabalho futuro.

6.1 Síntese e Contribuições

Os *Transformers* são extremamente importantes pela sua capacidade de lidar com dependências de longo alcance em dados sequenciais. Eles utilizam mecanismos de atenção que permitem que o modelo capture relações contextuais entre palavras, de forma mais eficiente do que as Redes Neuronais Recorrentes (RNN), como as *Long Short-Term Memory* (LSTM) (Hochreiter, 1997) e as *Gated Recurrent Units* (GRU) (Cho, Bahdanau, et al., 2014) , o que os torna especialmente úteis em tarefas de PLN, sendo utilizados nos mais diversos contextos desde a tradução, resumo de texto, análise de sentimentos ou tarefas de Pergunta/Resposta.

Neste trabalho, efetuou-se o ajuste fino de três modelos de *Transformers* pré-treinados disponíveis no Hugging Face[22], escolhidos de acordo com a sua performance em tarefas de Pergunta/Resposta em português. O objetivo foi avaliar e comparar os resultados do ajuste fino efetuado nestes modelos, para se verificar a capacidade de resposta a perguntas com base num domínio de conhecimento específico, a FNSE.

As principais contribuições deste trabalho são:

1. Criação, a partir do conteúdo disponível do sítio *web* da Fundação Nossa Senhora da Esperança (FNSE), de um *dataset* de Perguntas/Respostas no formato SQuAD;
2. O treino de três modelos de Pergunta/Respostas com o *dataset* criado com informação da FNSE, entre eles o SQUAD-PT-BASE, FAQUAD-PT-BASE e SQUAD-PT-LARGE, que conseguem responder a perguntas acerca da Fundação da Nossa Senhora da Esperança de maneira adequada;
3. Está ainda em preparação um artigo a submeter a uma revista.

6.2 Conclusões

Neste trabalho, foi feito o ajuste fino de três modelos *Transformers* de Pergunta/Resposta: o bert-base-cased-squad-v1.1-portuguese, o faquad-bert-base-portuguese-cased e o bert-large-cased-squad-v1.1-portuguese, os quais são baseados no modelo de linguagem BERTimbau (Souza et al., 2020).

Estes modelos, já tinham sido pré-treinados anteriormente com dados em português, tendo obtido um F1 de 0,8250, 0,831 e 0,8443 e um *exact match* de 0,7049, 0,745 e 0,7268 respetivamente. Foram realizados diversos treinos, com diferentes hiperparâmetros, com base no *dataset* criado, o que resultou em métricas com resultados de 0, 91111 de F1 e uma *accuracy,* de cerca de 0,96154, com o modelo SQUAD-PT-BASE (obtido pelo treino com o modelo bert-base-cased-squad-v1.1-portuguese). Estes resultados são bastante próximos do ideal para as métricas, o que mostra que, o modelo SQUAD-PT-BASE consegue responder adequadamente a perguntas em português acerca da FNSE.

Além disso, o modelo SQUAD-PT-BASE foi avaliado com perguntas que não faziam parte do *dataset* original, para se verificar a capacidade de generalização de informação. Os resultados obtidos foram de 0,8490 de F1, 0,8286 de *exact match* e 0,8306 de *score* médio. Estes resultados indicam que o modelo SQUAD-PT-BASE consegue generalizar os dados suficientemente para responder a perguntas que não faziam parte do *dataset* de treino e validação.

6.3 Trabalho Futuro

De seguida, são apresentadas algumas sugestões de trabalho futuro para complementar o que já foi realizado:

1. Acrescentar mais perguntas ao *dataset* que foi criado, a partir de informação adicional que não seja apenas a existente na página *web* da fundação, de forma a dotar os modelos de mais conhecimento relativo à FNSE;
2. Integrar os modelos *Transformers* treinados de Pergunta/Resposta em métodos de *Retrieval-Augmented Generation* (RAG). O RAG integra técnicas de recuperação de informação com a capacidade dos modelos *Transformers* de gerar respostas baseadas tanto no *input* do utilizador, como na informação

recuperada (Lewis et al., 2020). Vários estudos demonstram as vantagens da utilização de RAG, nomeadamente, na geração de respostas mais precisas e fundamentadas, particularmente, em tarefas de Pergunta/Resposta de domínio aberto, onde o conhecimento do modelo pré-treinado pode não ser suficiente (Izacard & Grave, 2021; Lewis et al., 2020). A arquitetura RAG, é assim constituída por dois componentes principais:

- **Recuperador**: que utiliza técnicas de recuperação de informação, como por exemplo, a *Dense Passage Retriever* (DPR) (Karpukhin et al., 2020) a Query2Doc (Karpukhin et al., 2020), a *Hybrid Dense Retrieval* (HyDE) (Izacard & Grave, 2021) ou a *Learning to Filter Context for Retrieval-Augmented Generation* (FILCO) (Wang et al., 2023), as quais localizam fontes de informação relevantes que irão servir como contexto, e alimentar o gerador evitando-se assim, o fornecimento explícito do contexto;
- **Gerador**: que corresponde a um modelo de linguagem pré-treinado (ou ao qual foi feito o ajuste fino, tal como os treinados neste trabalho), que recebe as informações relevantes obtidas pelo recuperador (contextos), bem como a pergunta do utilizador de forma a gerar uma resposta.

3. Integrar os modelos desenvolvidos no sistema de visitas guiadas e de árvores dialogantes do Museu de Tiflologia que a FNSE está a desenvolver.

REFERÊNCIAS BIBLIOGRÁFICA

Abdi, A., Idris, N., & Ahmad, Z. (2018). QAPD: An Ontology-based Question Answering System in the Physics Domain. *Soft Computing*, *22*(1). https://doi.org/10.1007/s00500-016-2328-2

Aizawa, A. (2003). An information-theoretic perspective of tf–idf measures. *Information Processing & Management*, *39*(1), 45–65. https://doi.org/10.1016/S0306-4573(02)00021-3

Allahyari, M., Pouriyeh, S., Assefi, M., Safaei, S., Trippe, E. D., Gutierrez, J. B., & Kochut, K. (2017). *Text Summarization Techniques: A Brief Survey*. https://doi.org/10.48550/arXiv.1707.02268

Amidi, A., & Amidi, S. (2018, September 9). *Machine Learning tips and tricks cheatsheet*. https://stanford.edu/~shervine/teaching/cs-229/cheatsheet-machine-learning-tips-and-tricks#classification-metrics

Anzanello, M. J., & Fogliatto, F. S. (2011). Learning curve models and applications: Literature review and research directions. *International Journal of Industrial Ergonomics*, *41*(5), 573–583. https://doi.org/10.1016/j.ergon.2011.05.001

Bafna, P., Pramod, D., & Vaidya, A. (2016). Document clustering: TF-IDF approach. *International Conference on Electrical, Electronics, and Optimization Techniques, ICEEOT 2016*, 61–66. https://doi.org/10.1109/ICEEOT.2016.7754750

Berry, M. W., Dumais, S. T., & O'Brien, G. W. (1995). Using linear algebra for intelligent information retrieval. *SIAM Review*, *37*(4), 573–595.

Bick, E. (2000). *The Parsing System "Palavras": Automatic Grammatical Analysis of Portuguese in a Constraint Grammar Framework*.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer New York, NY.

Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2017). Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics*, *5*, 135–146. https://doi.org/10.1162/tacl_a_00051

Branco, A., Castro, S., Silva, J., & Costa, F. (2011). *CINTIL DepBank Handbook: Design options for the representation of grammatical dependencies*. University of Lisbon.

Branco, A., & Silva, J. (2004). Evaluating Solutions for the Rapid Development of State-of-the-Art POS Taggers for Portuguese. In M. T. Lino, M. F. Xavier, F. Ferreira, R. Costa, & R. Silva (Eds.), *Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC'04)* (pp. 507–510).

Breiman, L. (2001). Random Forests. *Machine Learning*, *45*(1), 5–32. https://doi.org/10.1023/A:1010933404324

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., … Amodei, D. (2020).

Language Models are Few-Shot Learners. *Advances in Neural Information Processing Systems*, *2020-December*. https://doi.org/10.48550/arxiv.2005.14165

Brownlee, J. (2019, August 6). *How to use Learning Curves to Diagnose Machine Learning Model Performance*. https://machinelearningmastery.com/learning-curves-for-diagnosing-machine-learning-model-performance/

Brum, H., & Nunes, M. das G. V. (2017, November 24). Building a Sentiment Corpus of Tweets in Brazilian Portuguese. *20th Language Resources and Evaluation Conference in Miyazaki*. https://doi.org/10.48550/arXiv.1712.08917

Cardoso, N. (2008). *Novos rumos para a recuperação de informação geográfica em português* (L. Costa & D. Santos, Eds.; pp. 71–85).

Cavnar, W. B., & Trenkle, J. M. (1994). N-Gram-Based Text Categorization. *Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval*.

Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785–794. https://doi.org/10.1145/2939672

Cho, K., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). *Learning phrase representations using RNN encoder-decoder for statistical machine translation*. https://doi.org/10.48550/arXiv.1406.1078

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning Phrase Representations using RNN encoder-decoder for Statistical Machine Translation. *EMNLP 2014 - 2014 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*. https://doi.org/10.3115/v1/d14-1179

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., & Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, *41*(6), 391–407. https://doi.org/10.1002/(SICI)1097-4571(199009)41:6<391::AID-ASI1>3.0.CO;2-9

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. http://arxiv.org/abs/1810.04805

Dhillon, I. S., & Modha, D. S. (2001). Concept Decompositions for Large Sparse Text Data Using Clustering. *Machine Learning*, *42*(1/2), 143–175. https://doi.org/10.1023/A:1007612920971

Friedman, J. H. (2001). Greedy Function Approximation: A Gradient Boosting Machine. *Annals of Statistics*, *29*(5). https://doi.org/10.1214/aos/1013203451

Grandini, M., Bagli, E., & Visani, G. (2020). *Metrics for Multi-Class Classification: an Overview*. https://doi.org/10.48550/arXiv.2008.05756

Grilo, S., Bolrinha, M., Silva, J., Vaz, R., & Branco, A. (2020). The BDCamões Collection of Portuguese Literary Documents: a Research Resource for Language Technology and Digital Humanities. *Proceedings of the 12th Conference on Language Resources and Evaluation (LREC 2020)*, 849–854.

Guillou, P. (2021a). *NLP: Como treinar um modelo de Question Answering em qualquer linguagem baseado no BERT large*. https://medium.com/@pierre_guillou/nlp-como-treinar-um-modelo-de-question-answering-em-qualquer-linguagem-baseado-no-bert-large-1c899262dd96

Guillou, P. (2021b). *NLP: Modelo de Question Answering em qualquer idioma baseado no BERT base (Estudo de caso em português)*. https://medium.com/@pierre_guillou/nlp-modelo-de-question-answering-em-qualquer-idioma-baseado-no-bert-base-estudo-de-caso-em-12093d385e78

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., … Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2

He, P., Liu, X., Gao, J., & Chen, W. (2020). *DeBERTa: Decoding-enhanced BERT with Disentangled Attention*. https://doi.org/10.48550/arXiv.2006.03654

Hochreiter, S. (1997). Long Short-term Memory. *Neural Computation MIT-Press*. https://www.bioinf.jku.at/publications/older/2604.pdf

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, *9*(8). https://doi.org/10.1162/neco.1997.9.8.1735

Howard, J., & Ruder, S. (2018). *Universal Language Model Fine-tuning for Text Classification*. https://doi.org/10.48550/arXiv.1801.06146

Hunter, J. D. (2007). Matplotlib: A 2D Graphics Environment. *Computing in Science & Engineering*, *9*(3), 90–95. https://doi.org/10.1109/MCSE.2007.55

Izacard, G., & Grave, E. (2021). *Leveraging Passage Retrieval with Generative Models for Open Domain Question Answering*. https://arxiv.org/abs/2007.01282

Jabbar, H. K., & Khan, R. Z. (2015). Methods to Avoid Over-Fitting And Under-Fitting in Supervised Machine Learning (Comparative Study). *Computer Science, Communication and Instrumentation Devices 70.10.3850*, 978–981.

Jain, A. K., Murty, M. N., & Flynn, P. J. (1999). Data clustering: A review. *ACM Computing Surveys*, *31*(3). https://doi.org/10.1145/331499.331504

Joachims, T. (1998). *Text Categorization with Support Vector Machines: Learning with Many Relevant Features*.

Jolliffe, I. T., & Cadima, J. (2016). Principal component analysis: a review and recent developments. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, *374*(2065), 20150202. https://doi.org/10.1098/rsta.2015.0202

Junqueira, J. da R., Luis Junior, C., Silva, F. L. V., Côrrea, U. B., & de Freitas, L. A. (2023). Albertina in Action: An Investigation of its Abilities in Aspect Extraction, Hate Speech Detection, Irony Detection, and Question-Answering. *Anais Do XIV Simpósio Brasileiro de Tecnologia Da Informação e Da Linguagem Humana (STIL 2023)*, 146–155. https://doi.org/10.5753/stil.2023.234159

Jurafsky, D., & Martin, J. H. (2023). *Speech and Language Processing*.

Karpukhin, V., Oğuz, B., Min, S., Lewis, P., Wu, L., Edunov, S., Chen, D., & Yih, W. (2020). *Dense Passage Retrieval for Open-Domain Question Answering*. https://arxiv.org/abs/2004.04906

Karunasingha, D. S. K. (2022). Root mean square error or mean absolute error? Use their ratio as well. *Information Sciences*, *585*, 609–629. https://doi.org/10.1016/j.ins.2021.11.036

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., & Liu, T.-Y. (2017). Lightgbm: A highly efficient gradient boosting decision tree. *Advances in Neural Information Processing Systems*, *30*.

Khan, S., Naseer, M., Hayat, M., Zamir, S. W., Khan, F. S., & Shah, M. (2022). Transformers in Vision: A Survey. *ACM Computing Surveys*, *54*(10s), 1–41. https://doi.org/10.1145/3505244

Kowsari, Jafari Meimandi, Heidarysafa, Mendu, Barnes, & Brown. (2019). Text Classification Algorithms: A Survey. *Information*, *10*(4), 150. https://doi.org/10.3390/info10040150

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W., Rocktäschel, T., Riedel, S., & Kiela, D. (2020). *Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks*. https://doi.org/10.48550/arXiv.2005.11401

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., & Stoyanov, V. (2019). *RoBERTa: A Robustly Optimized BERT Pretraining Approach*.

Lopez, A. (2008). Statistical Machine Translation. *ACM Computing Surveys*, *40*(3), 1–49. https://doi.org/10.1145/1380584.1380586

Maynard, D., & Funk, A. (2012). *Automatic Detection of Political Opinions in Tweets* (pp. 88–99). https://doi.org/10.1007/978-3-642-25953-1_8

McCallum, A., & Nigam, K. (1998). A Comparison of Event Models for Naive Bayes Text Classification. *AAAI/ICML-98 Workshop on Learning for Text Categorization*. https://doi.org/10.1.1.46.1529

McKinney, W. (2010). *Data Structures for Statistical Computing in Python*. 56–61. https://doi.org/10.25080/Majora-92bf1922-00a

Medhat, W., Hassan, A., & Korashy, H. (2014). Sentiment analysis algorithms and applications: A survey. *Ain Shams Engineering Journal*, *5*(4), 1093–1113. https://doi.org/10.1016/j.asej.2014.04.011

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *1st International Conference on Learning Representations, ICLR 2013 - Workshop Track Proceedings*. https://doi.org/https://doi.org/10.48550/arXiv.1301.3781

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. *Advances in Neural Information Processing Systems*, *26*.

Nassiri, K., & Akhloufi, M. (2023). Transformer Models Used for Text-based Question Answering Systems. *Applied Intelligence*, *53*(9), 10602–10635. https://doi.org/10.1007/s10489-022-04052-8

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., … Chintala, S. (2019). *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. https://medium.com/@pierre_guillou/nlp-modelo-de-question-answering-em-qualquer-idioma-baseado-no-bert-base-estudo-de-caso-em-12093d385e7

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Müller, A., Nothman, J., Louppe, G., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, É. (2012). *Scikit-learn: Machine Learning in Python*.

Pennington, J., Socher, R., & Manning, C. D. (2014). GloVe: Global Vectors for Word Representation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1532–1543.

Porter, M. F. (1980). An algorithm for suffix stripping. *Program: Electronic Library and Information Systems*, *14*(3), 130–137. https://doi.org/10.1108/eb046814

Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A. V., & Gulin, A. (2018). CatBoost: unbiased boosting with categorical features. *NIPS 2018: Proceedings of the 32nd International Conference on Neural Information Processing Systems*, 6639–6649. https://doi.org/10.5555/3327757.3327770

Qader, W. A., Ameen, M. M., & Ahmed, B. I. (2019). An Overview of Bag of Words;Importance, Implementation, Applications, and Challenges. *2019 International Engineering Conference (IEC)*, 200–204. https://doi.org/10.1109/IEC47844.2019.8950616

Quaresma, P. (2023). *Sentiment Analysis - Portulan Clarin*. https://portulanclarin.net/workbench/uevora-sentiment-analysis/

Quinlan, J. R. (1986). Induction of Decision Trees. *Machine Learning*, *1*(1). https://doi.org/10.1023/A:1022643204877

Radford, A., Wu, J., Child, R., Amodei, D., & Sutskever, I. (2019). Language Models are Unsupervised Multitask Learners. *OpenAI Blog*, 1–8.

Rajpurkar, P., Jia, R., & Liang, P. (2018). Know what you don't know: Unanswerable questions for SQuAD. *ACL 2018 - 56th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference (Long Papers)*, *2*, 784–789. https://doi.org/10.18653/v1/p18-2124

Rajpurkar, P., Zhang, J., Lopyrev, K., & Liang, P. (2016). *SQuAD: 100,000+ Questions for Machine Comprehension of Text*. https://doi.org/10.48550/arXiv.1606.05250

Rocha, P., & Santos, D. (2000). CETEMPúblico: Um corpus de grandes dimensões de linguagem jornalística portuguesa. In M. das Graças Volpe Nunes Nunes (Ed.), *Actas do V Encontro para o processamento computacional da língua portuguesa escrita e falada (PROPOR'2000)* (pp. 131–140).

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning Representations by Back-propagating Errors. *Nature*, *323*(6088). https://doi.org/10.1038/323533a0

Santos, D., & Rocha, P. (2005). The key to the first CLEF with Portuguee: Topics, Questions and Answers in CHAVE. *Multilingual Information Access for Text, Speech and Images, 5th Workshop of the Cross-Language Evaluation Forum, CLEF 2004*, 821–832.

Santos, R., Rodrigues, J., Gomes, L., Silva, J., Branco, A., Cardoso, H. L., Osório, T. F., & Leite, B. (2024). *Fostering the Ecosystem of Open Neural Encoders for Portuguese with Albertina PT* Family*.

Sayama, H. F., Araujo, A. V., & Fernandes, E. R. (2019). FaQuAD: Reading Comprehension Dataset in the Domain of Brazilian Higher Education. *Proceedings - 2019 Brazilian Conference on Intelligent Systems, BRACIS 2019*. https://doi.org/10.1109/BRACIS.2019.00084

Shah, F. P., & Patel, V. (2016). A review on feature selection and feature extraction for text classification. *2016 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*, 2264–2268. https://doi.org/10.1109/WiSPNET.2016.7566545

Silva, E. H., Laterza, J., & Faleiros, T. P. (2022). New State-of-the-Art for Question Answering on Portuguese SQuAD v1. 1. *Anais Do X Symposium on Knowledge Discovery, Mining and Learning*, 98–105. https://doi.org/10.5753/kdmile.2022.227787

Silva, J., & Branco, A. (2010). Out-of-the-Box Robust Parsing of Portuguese. *Proceedings of the 9th International Conference on the Computational Processing of Portuguese (PROPOR2010)*, 75–85.

Souza, F., Nogueira, R., & Lotufo, R. (2020). BERTimbau: Pretrained BERT Models for Brazilian Portuguese. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *12319 LNAI*. https://doi.org/10.1007/978-3-030-61377-8_28

Jones, K. S. (1972). A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, *28*(1), 11–21. https://doi.org/10.1108/eb026526

The pandas development team. (2020, February). *pandas-dev/pandas: Pandas*. https://doi.org/10.5281/zenodo.3509134

Vapnik, V. N., & Chervonnkis, A. Y. (1964). A Class of Algorithms for Pattern Recognition Learning. *Avtomat. i Telemekh*, *25*(6), 937–945.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). *Attention Is All You Need*. http://arxiv.org/abs/1706.03762

Wang, Q., Ma, Y., Zhao, K., & Tian, Y. (2022). A Comprehensive Survey of Loss Functions in Machine Learning. *Annals of Data Science*, *9*(2), 187–212. https://doi.org/10.1007/s40745-020-00253-5

Wang, W., & Lu, Y. (2018). Analysis of the Mean Absolute Error (MAE) and the Root Mean Square Error (RMSE) in Assessing Rounding Model. *IOP Conference Series: Materials Science and Engineering*, *324*, 012049. https://doi.org/10.1088/1757-899X/324/1/012049

Wang, W., Yan, M., & Wu, C. (2018). *Multi-granularity hierarchical attention fusion networks for reading comprehension and question answering*. https://doi.org/10.18653/v1/P18-1158

Wang, Z., Araki, J., Jiang, Z., Parvez, M. R., & Neubig, G. (2023). *Learning to Filter Context for Retrieval-Augmented Generation*. https://doi.org/10.48550/arXiv.2311.08377

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Scao, T. Le, Gugger, S., … Rush, A. M. (2019). *HuggingFace's Transformers: State-of-the-art Natural Language Processing*. https://doi.org/10.48550/arXiv.1910.03771

Zellers, R., Bisk, Y., Schwartz, R., & Choi, Y. (2018). *SWAG: A Large-Scale Adversarial Dataset for Grounded Commonsense Inference*. https://doi.org/10.48550/arXiv.1808.05326

ANEXOS

ANEXO 1 – Serviços de análise de frases (Lx-Parser, Lx-DepParser e Tags usadas)

Lx-Suite[1] é um serviço online e gratuito que permite o processamento superficial de frases em português. Este serviço é constituído por um conjunto de ferramentas como o *LX Sentencer Splitter*[29], que é capaz de delimitar os limites de uma frase e de um parágrafo, o *LX-Tokenizer*[30] que segmenta o texto em átomos, o *LX-Tagger*[31] que atribui a cada átomo uma tag disponível na Tabela 16 e Tabela 17, o *LX-Featurizer* que atribui valores de característica de flexão a palavras com categorias nominais e o *LX-Lemmatizer* que atribui um lema às palavras (Branco & Silva, 2004)

Tabela 16 - Tags lexicais e sintagmáticas

Tag	Descrição
ADJ	Adjetivo
ADV	Advérbio
CARD	Cardinal
CJ	Conjunção
CL	Clítico
CN	Nome Comum
DA	Artigo Definido
DEM	Demonstrativo
DFR	Fracionário
DGTR	Numeração Romana
DGT	Dígito
DM	Marcador Discursivo
EADR	Endereço Eletrónico
EOE	Fim da enumeração

[29] https://portulanclarin.net/workbench/lx-sentencesplitter/
[30] https://portulanclarin.net/workbench/lx-tokenizer/
[31] https://portulanclarin.net/workbench/lx-tagger/

EXC	Exclamativa
GER	Gerúndio
GERAUX	Gerúndio "ter"/"haver" em tempos compostos
IA	Artigo Definido
IND	Indefinido
INF	Infinitivo
INFAUX	Infinitivo "ter" /" haver" em tempos compostos
INT	Interrogativa
ITJ	Interjeição
LTR	Letra
MGT	Classe de magnitude
MHT	Mês
NP	Sintagma nominal
ORD	Ordinal
PADR	Parte de endereço
PNM	Parte de nome
PNT	Pontuação
POSS	Possessivo
PPA	Particípio passado
PP	Sintagma preposicional
PPT	Particípio Passado em tempo composto
PREP	Preposição
PRS	Pronome Pessoal
QNT	Quantificador
REL	Pronome Relativo
STT	Forma de Tratamento
SYB	Símbolo
TERMN	Terminação

UM	"um" ou "uma"
UNIT	Símbolo de unidade de medida
VAUX	Finito "ter"/"haver" em tempos compostos
V	Verbo (diferente de PPA, PPT, INF, GER)
WD	Dia da semana

Fonte: (Branco & Silva, 2004)

Tabela 17 - Tags Multipalavra

Tag	Descrição
LADV1…LADVn	Advérbio multipalavra
LCJ1…LCJn	Conjunção multipalavra
LDEM1…LDEMn	Demonstrativo multipalavra
LDFR1…LDFRn	Expressão fracional multipalavra
LDM1…LDMn	Marcador discursivo multipalavra
litj1…litjn	Interjeição multipalavra
lprs1…lprsn	Pronome pessoal multipalavra
lprep1…lprepn	Preposição multipalavra
lqd1…lqdn	Quantificador multipalavra
lrel1…lreln	Pronome relativo multipalavra

Fonte: (Branco & Silva, 2004)

Lx-Parser[2] é um serviço disponível online de forma gratuita, desenvolvido pela Universidade de Lisboa, que é utilizado para análise de frases no idioma português. Este serviço é capaz de analisar sintaticamente as frases, contruindo as árvores de constituência. O analisador faz uso de gramáticas probabilísticas e é capaz de alcançar um *F-Score* de 0,89. Este serviço classifica as palavras consoante as tags da Tabela 18 (J. Silva & Branco, 2010) .

Tabela 18 – Lx-Parser tagger

Tag	Descrição
A	Adjetivo Sub-Constituinte da frase
AP	Frase Adjetiva
ADV	Advérbio Sub-constituinte da frase
ADVP	Frase do Advérbio
C	Complemento
CL	Clítico
CP	Frase Complemento
CARD	Cardinal Sub-constituinte da frase
CONJ	Conjunção Sub-constituinte da frase
CONJP	Frase de conjunção
D	Determinante
DEM	Demonstrativo
N	Substantivo
NP	Locução Substantiva
O	Ordinal
P	Preposição
PP	Frase de Preposição
PPA	Particípio
POSS	Possessivo
PRS	Pronome Pessoal
QNT	Quantificador
REL	Pronome Relativo
S	Frase
V	Verbo Sub-constituinte da frase
VP	Frase Verbal

Fonte: (J. Silva & Branco, 2010)

Lx-DepParser[3] é um serviço de análise de frases em português, que considera as funções gramaticais das frases pela construção de árvores de dependências. Este serviço foi treinado com as frases de *CINTIL-DependencyBank*[32] (Branco et al., 2011). O LX-DepParser classifica os constituintes da frase segundo as tags das Tabela 19, Tabela 20, Tabela 21 e Tabela 22 (Branco et al., 2011).

Tabela 19 – Conjunto de tags de função gramatical

Tag	Descrição
C	Complemento
CARD	Cardinal
COORD	Coordenação
CONJ	Conjunção
DEP	Dependência
DO	Complemento Direto
IO	Complemento Indireto
M	Modificador
N	Nome
OBL	Complemento Oblíquo
PRD	Predicado
PUNCT	Ponto
ROOT	Raiz da Frase
SJ	Sujeito anticausativo
SJac	Sujeito simples
SJcp	Sujeito complexo
SP	Especificador

Fonte: (Branco et al., 2011)

[32] https://hdl.handle.net/21.11129/0000-000B-D31C-8

Tabela 20 - Conjunto de tags para categorias nominais

Tag	Descrição
M	Masculino
F	Feminino
G	Género Indeterminado
S	Singular
P	Plural
N	Número indeterminado
Dim	Diminutivo
Sim	Aumentativo
Comp	Comparativo

Fonte: (Branco et al., 2011)

Tabela 21 - Conjunto de tags para verbos

Tag	Descrição
1	Primeira pessoa
2	Segunda Pessoa
3	Terceira Pessoa
Pi	Presente do Indicativo
Ppi	Pretérito Perfeito do Indicativo
Ii	Pretérito Imperfeito do indicativo
Mpi	Pretérito Mais que Perfeito do Indicativo
Fi	Futuro do Indicativo
C	Condicional
Pc	Presente do Conjuntivo
Ic	Pretérito Imperfeito do Conjuntivo
Fc	Futuro do Conjuntivo
Imp	Imperativo

Fonte: (Branco et al., 2011)

Tabela 22 - Conjunto de Tags para verbos Infinitivos

Tags	Descrição
Ifl	Flexionado
Nifl	Não Flexionado

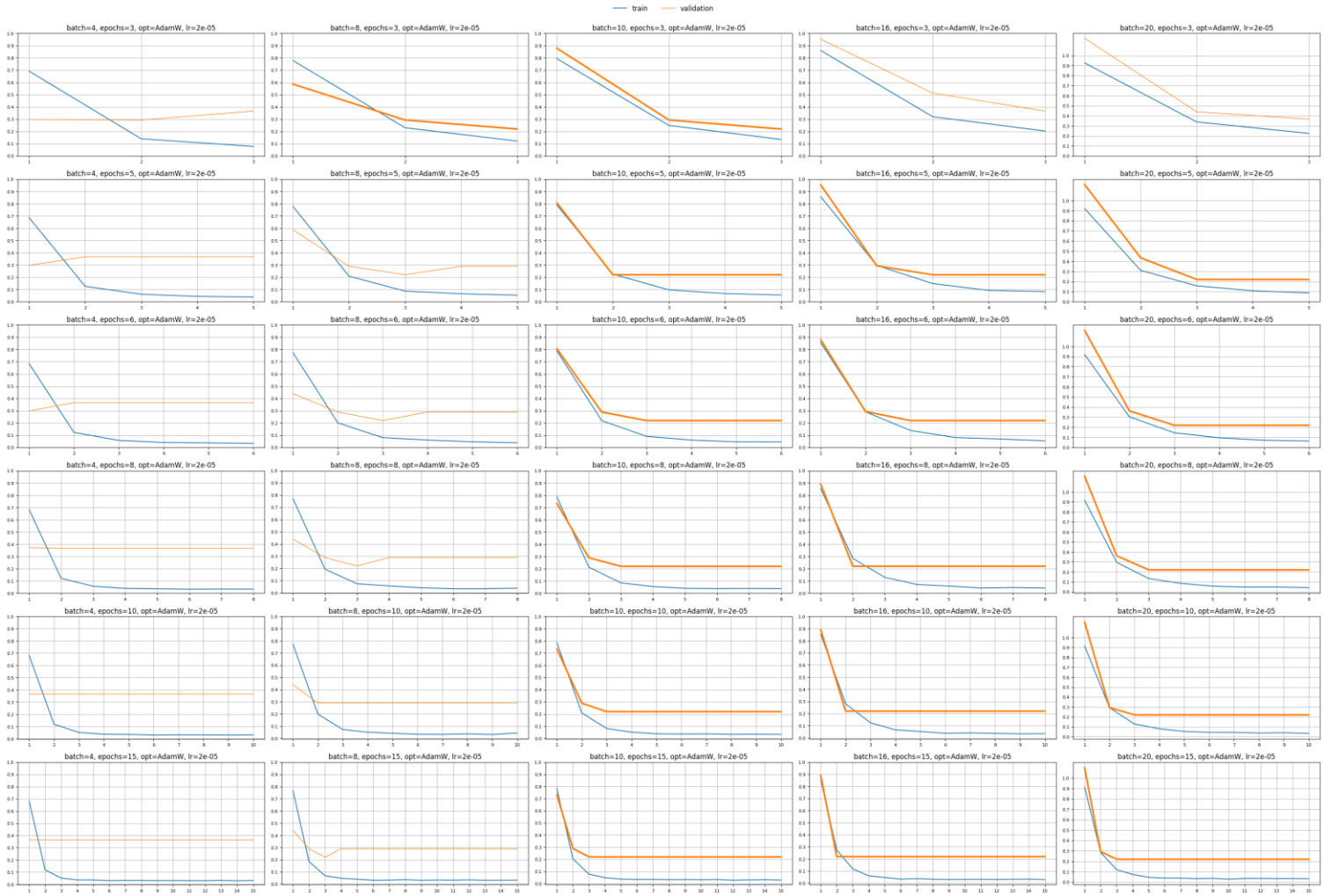
Fonte: (Branco et al., 2011)

O Cintil Trebank[33] é um corpus que contém árvores de constituência construídas com base de textos em Língua Portuguesa. Este corpus contém 10 mil frases e 110 mil átomos retirados de várias fontes (como notícias e romances). O Cintil DependencyBank é um corpus constituído por árvores de dependências e está alinhado ao CINTIL Treebank. Este corpus é composto por 3 000 frases retiradas de artigos de jornais portugueses.

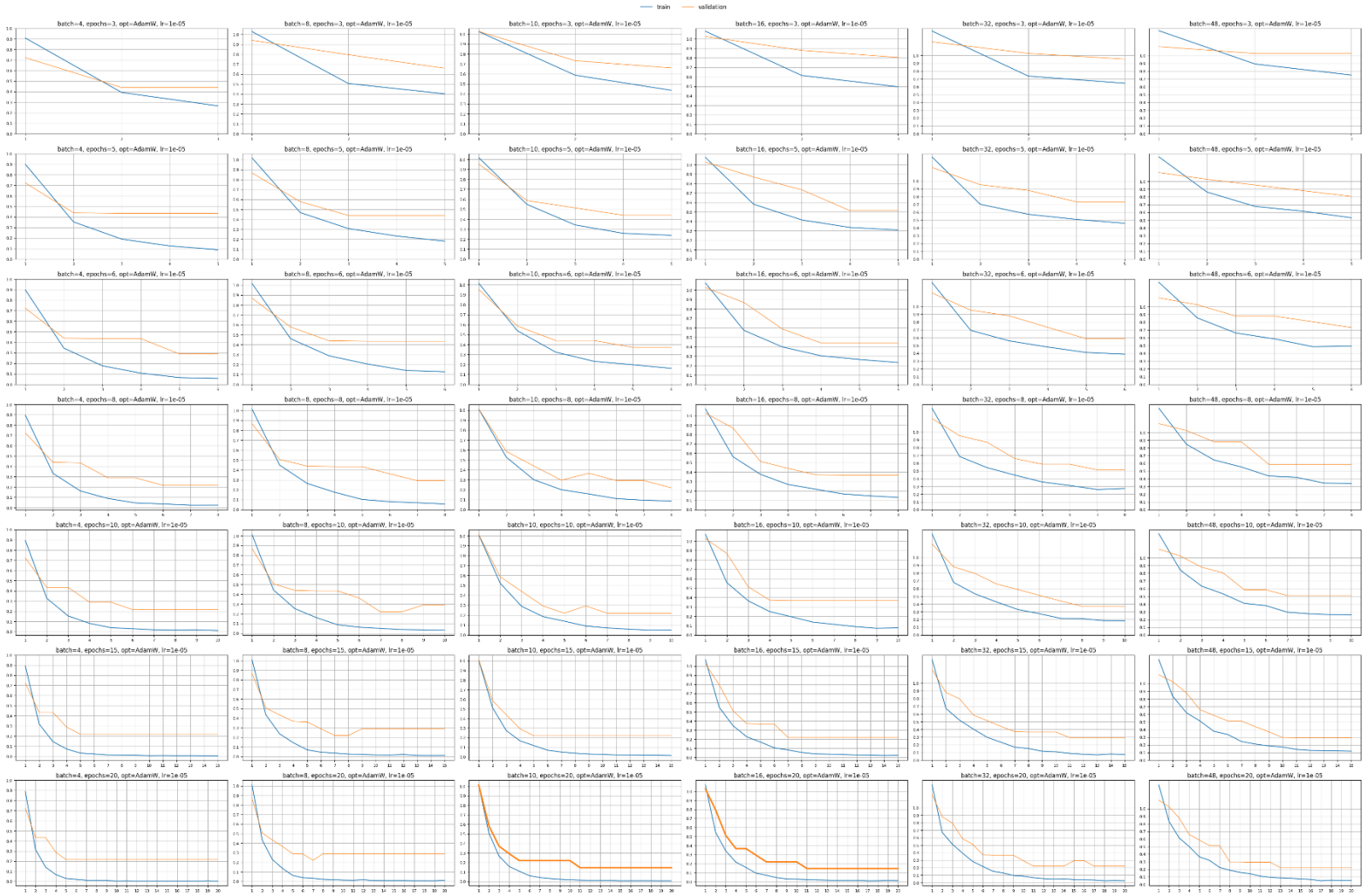
VISTA@UE-SA[34] é um serviço online desenvolvido por Paulo Quaresma que é utilizado para análise de sentimentos de frases curtas em português. Este serviço é baseado em aprendizagem máquina e foi treinado por uma coleção de frases curtas previamente classificadas (Brum & Nunes, 2017; Quaresma, 2023)

[33] https://hdl.handle.net/21.11129/0000-000B-D2FE-A
[34] https://portulanclarin.net/workbench/uevora-sentiment-analysis/

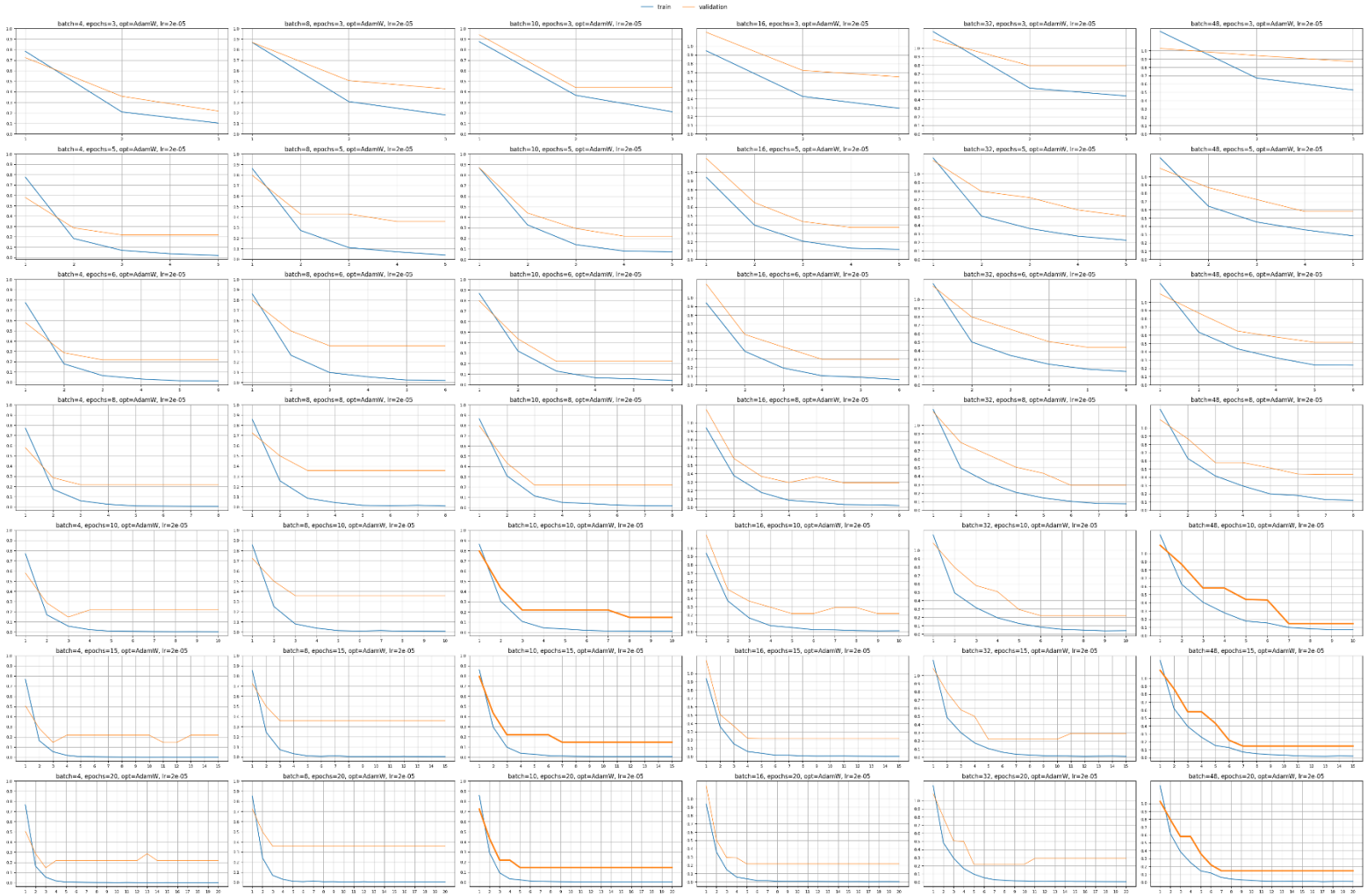
ANEXO 2 – Curvas de Aprendizagem (SQUAD-PT-BASE, learning_rate=1e-04)



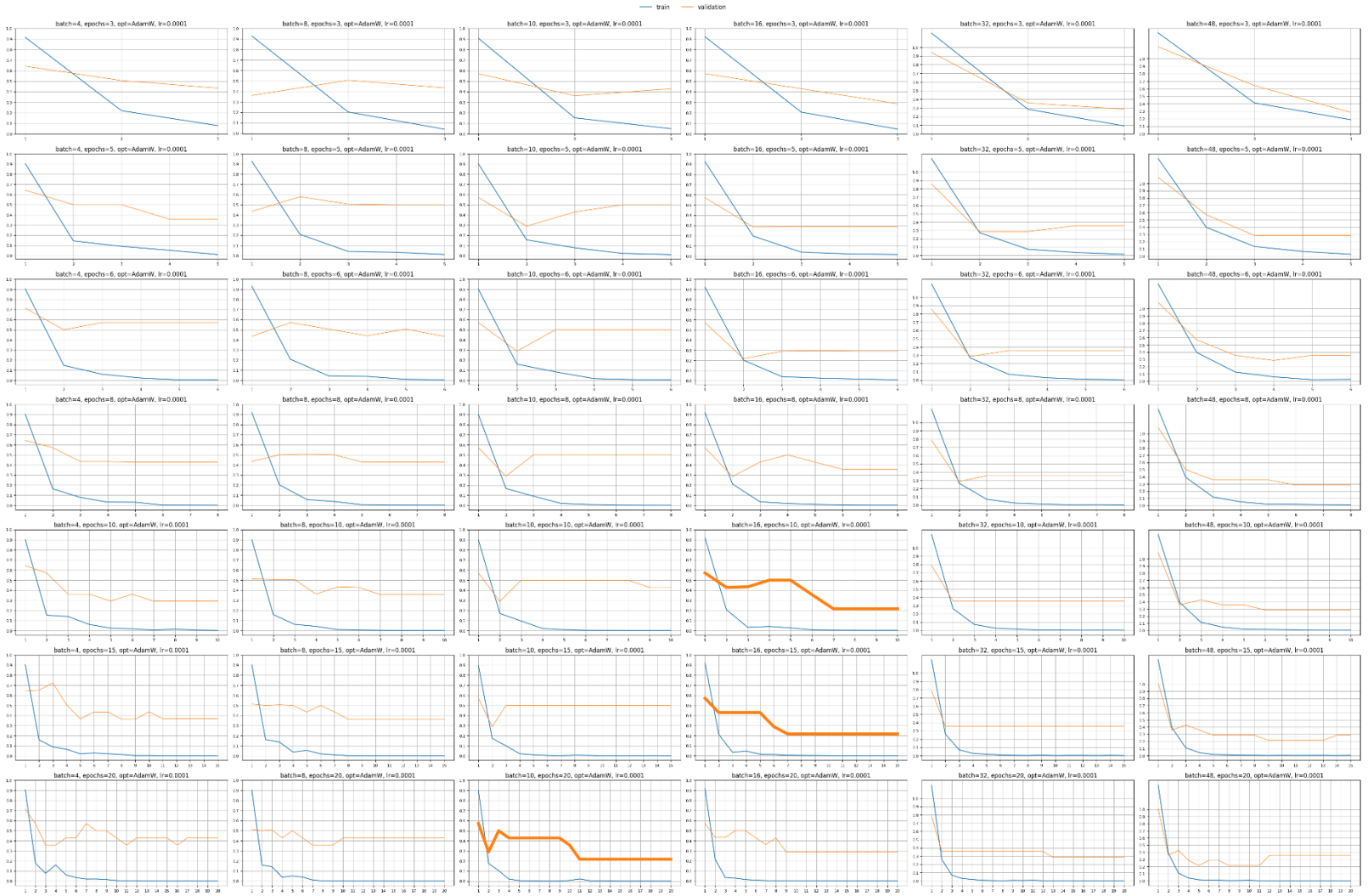
ANEXO 3 – Curvas de Aprendizagem (SQUAD-PT-BASE, learning_rate=1e-05)



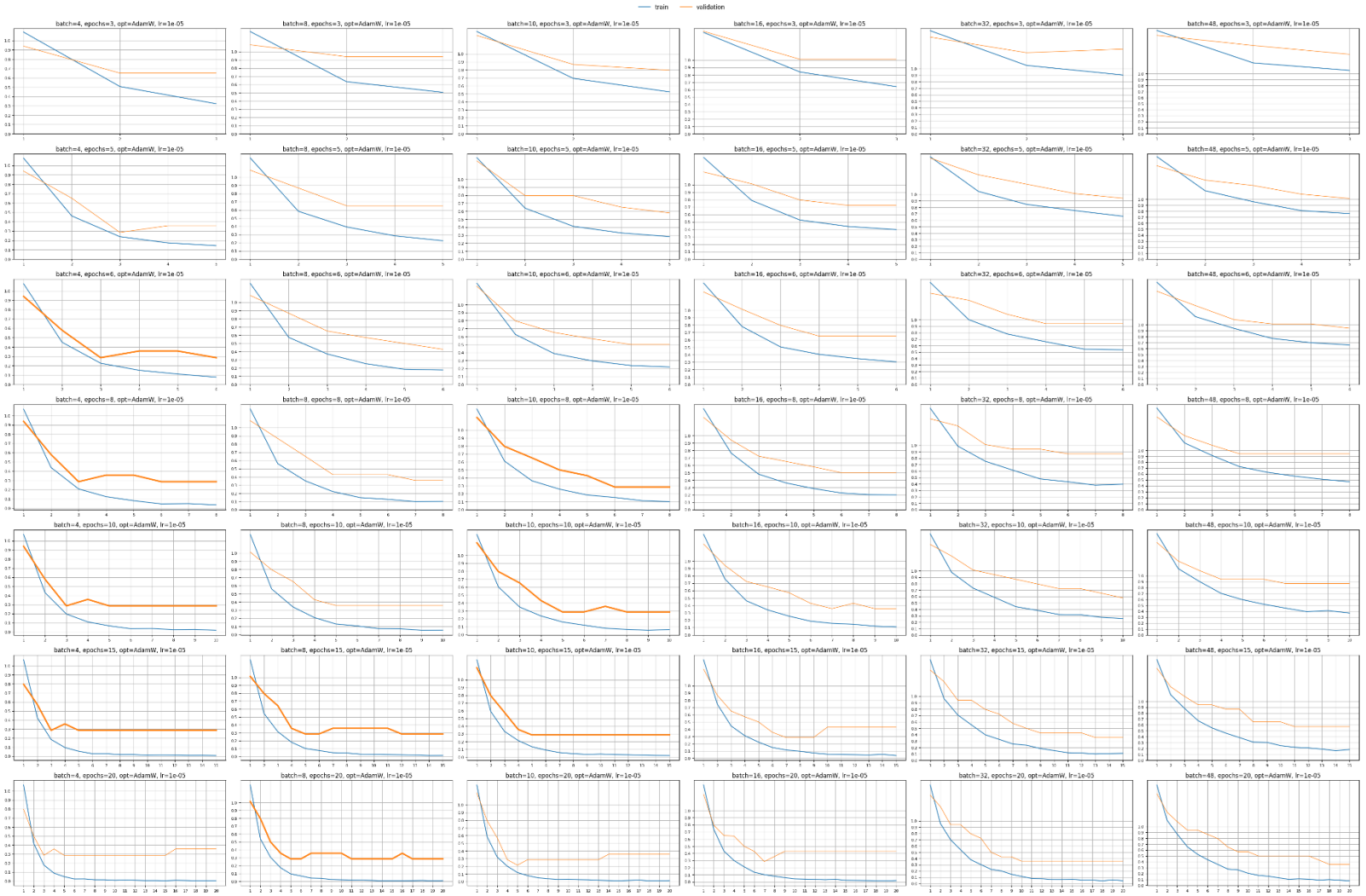
ANEXO 4 – Curvas de Aprendizagem (SQUAD-PT-BASE, learning_rate=2e-05)



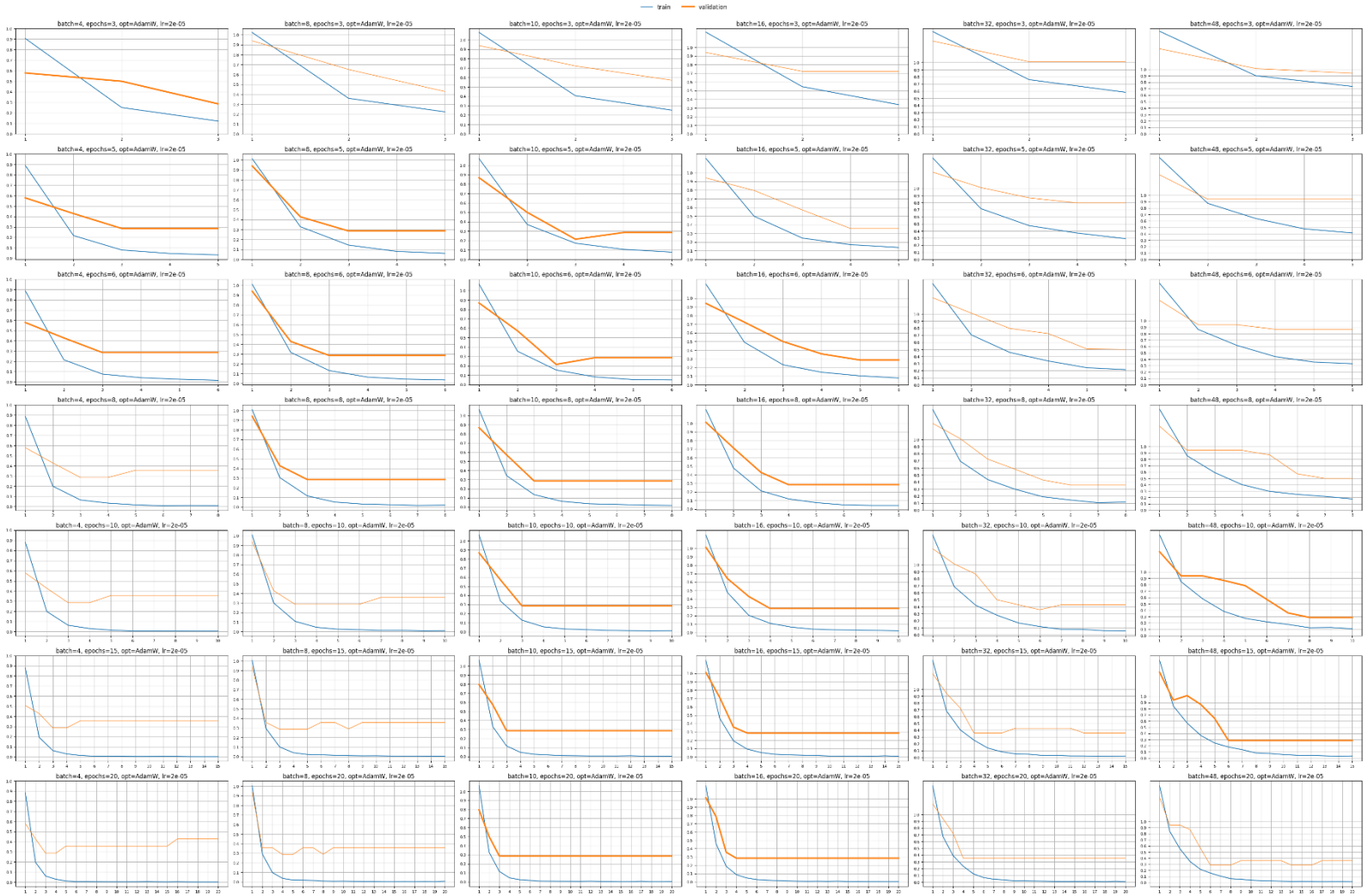
ANEXO 5 – Curvas de Aprendizagem (FAQUAD-PT-BASE, learning_rate=1e-04)



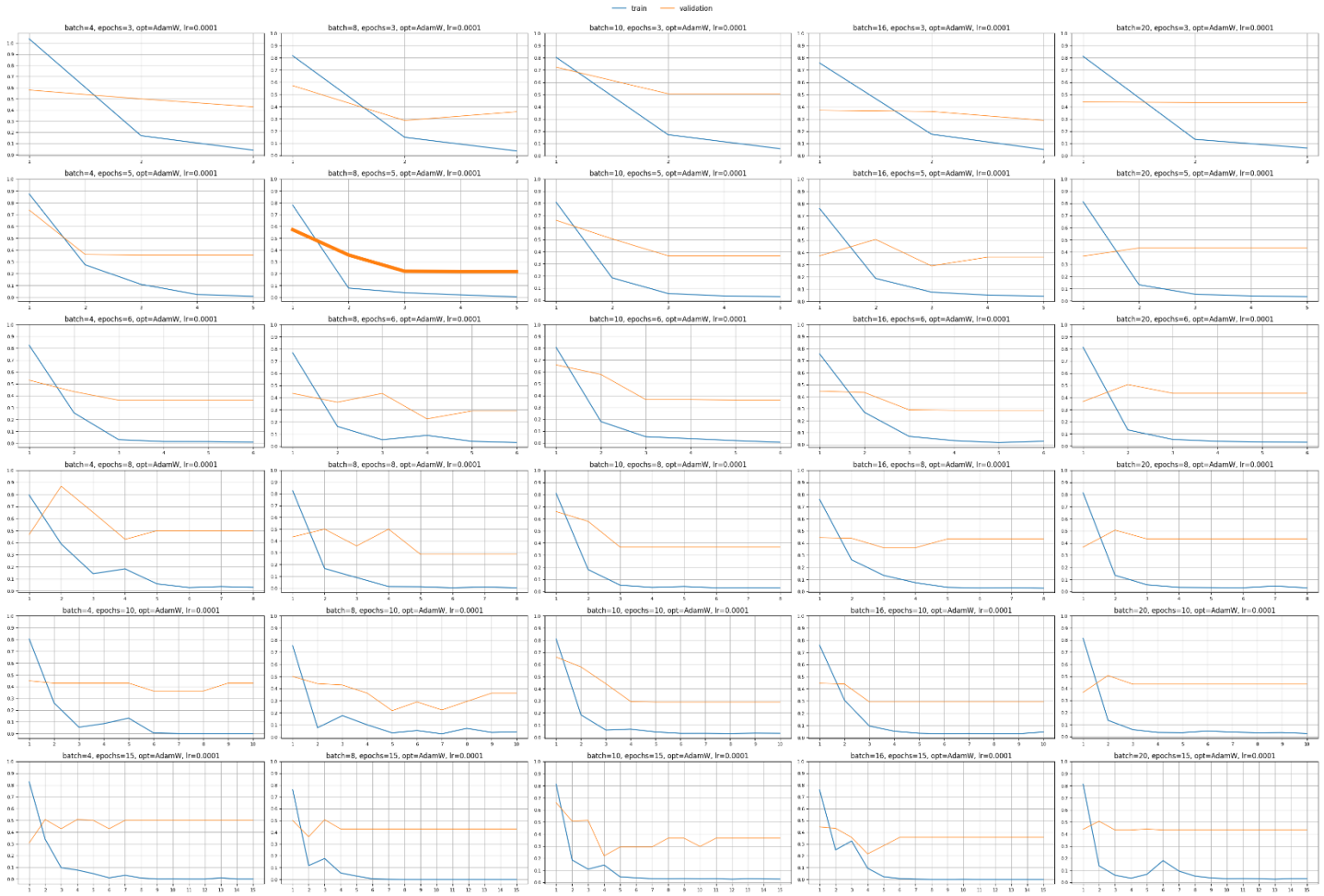
ANEXO 6 – Curvas de Aprendizagem (FAQUAD-PT-BASE, learning_rate=1e-05)



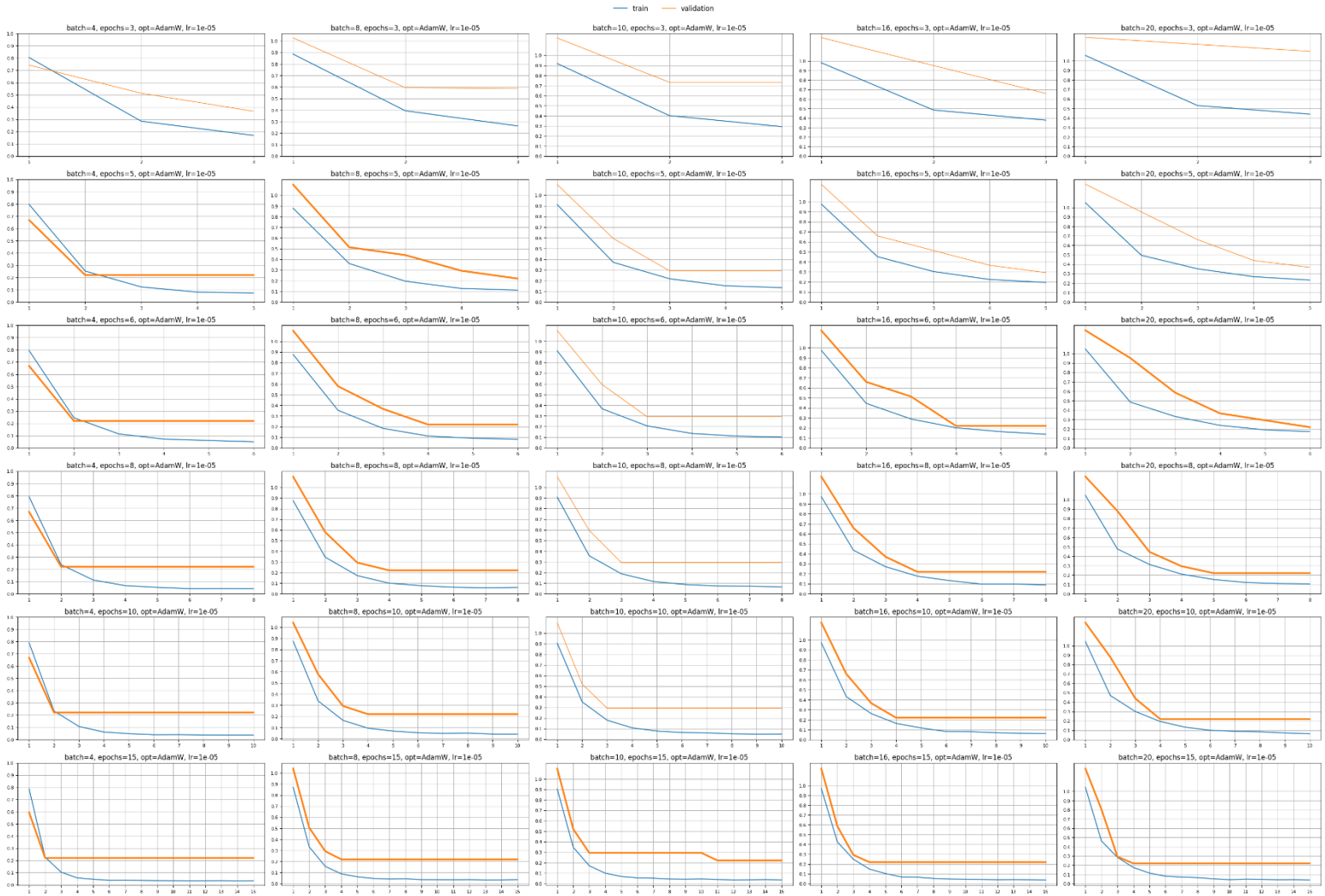
ANEXO 7 – Curvas de Aprendizagem (FAQUAD-PT-BASE, learning_rate=2e-05)



ANEXO 8 – Curvas de Aprendizagem (SQUAD-PT-LARGE, learning_rate=1e-04)



ANEXO 9 – Curvas de Aprendizagem (SQUAD-PT-LARGE, learning_rate=1e-05)



ANEXO 10 – Curvas de Aprendizagem (SQUAD-PT-LARGE, learning_rate=2e-05)