João Filipe Mendes Castilho

Validação Inteligente de Documentos

Validação Inteligente de Documentos utilizando Processamento de Linguagem Natural e Visão Computacional

1 2 9 0

UNIVERSIDADE D COIMBRA

João Filipe Mendes Castilho

Validação Inteligente de Documentos

Validação Inteligente de Documentos utilizando Processamento de Linguagem Natural e Visão Computacional

Dissertação no âmbito do Mestrado em Engenharia Informática, especialização em Sistemas Inteligentes orientada pelo Professor Doutor Jorge Henriques (DEI-UC) e Doutor Tiago Baptista (CRITICAL Software) e apresentado à Faculdade de Ciências e Tecnologia / Departamento de Engenharia Informática.

Junho de 2020

Faculdade de Ciências e Tecnologia

Departamento de Engenharia Informática

Validação Inteligente de Documentos

Validação Inteligente de Documentos utilizando Processamento de Linguagem Natural e Visão Computacional

João Filipe Mendes Castilho

Dissertação no âmbito do Mestrado em Engenharia Informática, especialização em Sistemas Inteligentes orientado pelo/a Professor/a Doutor/a Jorge Henriques (DEI-UC) e Doutor Tiago Baptista (CRITICAL Software) e apresentado à Faculdade de Ciências e Tecnologia / Departamento de Engenharia Informática.

Junho 2020

UNIVERSIDADE D
COIMBRA

Abstract

In recent years, more and more information has been produced that needs to be easily available and accessible. This need has led to an increasing digitalization of information in order to meet the demands of users. This process can be carried out through manual work, or by scanning files. The first option provides better results, since the information becomes "editable", however, it is a time-consuming, repetitive process and it's susceptible to failures. The second option is much faster, however, the information turns out to be in image format, which makes its management and organization difficult. Even if this second approach uses an Optical Character Recognition (OCR) tool, the information is not structured, which makes the research and organization process quite complex.

CRITICAL Software (CSW) is developing a document analysis platform, which offers numerous services, including the document classification service and the information extraction and analysis service.

The internship carried out at the CSW company, with the duration of an academic year, focuses on the extraction and analysis of information from documents. Its main objective is to implement a proof of concept of an algorithm for extracting and analyzing information from documents on the CSW platform, Intelligent Document Validation (IDV). The internship took place in three stages. In the first stage, research articles on state-of-the-art algorithms for document information analysis were explored and *CloudScan*[52], *CUTIE*[74], *Chargrid*[33] and *BERTgrid*[7] algorithms were selected. In the second stage, the algorithms selected in the previous step and respective variants were implemented in a total of 9 algorithms. In the third stage, an optimization analysis and performance comparison of the implemented algorithms was performed. These algorithms were tested on a data set composed of 1210 invoices and it was concluded that the algorithm with the best performance results from a variant that combines the *Chargrid* and *BERTgrid* algorithms, whose overall performance was 85.89% with the F1-Score metric . As a proof of concept, this version was successfully implemented on CSW's IDV platform.

Keywords

Machine Learning, Natural Language Processing, Computer Vision, Deep Learning, Image Processing

Resumo

Nos últimos anos, produz-se cada vez mais informação que necessita de estar facilmente disponível e acessível. Esta necessidade levou a uma crescente digitalização da informação de forma a satisfazer as exigências dos utilizadores. Este processo pode ser realizado através de trabalho manual, ou através de *scanning* de ficheiros. A primeira opção fornece melhores resultados, visto que a informação passa a ser "editável", no entanto, é um processo moroso, repetitivo e suscetível a falhas. A segunda opção é bastante mais rápida, no entanto, a informação acaba por estar em formato de imagem, o que torna difícil a sua gestão e organização. Mesmo que nesta segunda abordagem se utilize uma ferramenta de Reconhecimento de Caracteres Óptico (RCO), a informação não fica estruturada, o que torna bastante complexo o processo de pesquisa e organização.

A CRITICAL Software (CSW) está a desenvolver uma plataforma de análise de documentos, que oferece inúmeros serviços, onde se destacam o serviço de classificação de documentos e o serviço de extração e análise de informação.

O estágio realizado na empresa CSW, com a duração de um ano letivo, foca-se no serviço de extração e análise da informação de documentos. Tem como principal objetivo implementar uma prova de conceito de um algoritmo de extração e análise de informação de documentos na plataforma da CSW, o Intelligent Document Validation (IDV). O estágio desenvolvou-se em três etapas. Na primeira etapa, foram explorados artigos de investigação de algoritmos do estado da arte da análise de informação de documentos e foram selecionados os algoritmos *CloudScan*[52], *CUTIE*[74], *Chargrid*[33] e *BERTGrid*[7]. Na segunda etapa, foram implementados os algoritmos selecionados na etapa anterior e respetivas variantes num total de 9 algoritmos. Na terceira etapa, foi realizada a análise de optimização e comparação do desempenho dos algoritmos implementados. Estes algoritmos foram testados num conjunto de dados composto por 1210 faturas e, concluiu-se que o algoritmo com melhor desempenho resulta de uma variante que combina os algoritmos do *Chargrid* com *BERTgrid*, cujo desempenho geral foi de 85.89% com a métrica *F1-Score*. Como prova de conceito, esta versão foi implementada com sucesso na plataforma IDV da CSW.

Palavras-Chave

Aprendizagem Computacional, Processamento de Linguagem Natural, Visão Computacional, *Deep Learning*, Processamento de Imagens

Agradecimentos

Aos meus pais, irmãos e Raquel pela presença constante de apoio e confiança que me deram nos momentos mais difíceis.

Aos meus orientadores Jorge Henriques e Tiago Baptista por todo o acompanhamento e incentivos dados ao longo do estágio.

À empresa CRITICAL Software, por me ter proporcionado esta experência em ambiente empresarial e pelas condições que me ofereceram, inclusivé durante o confinamento imposto.

Por fim, um agradecimento a toda a equipa de Inteligência Artificial, por toda a disponibilidade e abertura que tiveram para comigo.

Índice

Acrónimos

AC Aprendizagem Computacional. 3, 4, 7, 12, 17, 22, 23, 45, 85

AWS *Amazon Web Services*. 34

CLN Compreensão de Linguagem Natural. 13

CSW CRITICAL Software. iii, v, 1–3, 39, 42–45, 60, 79, 80, 83, 86

IA Inteligência Artificial. 2, 4, 7, 12, 13, 33, 45, 85, 86

IDV Intelligent Document Validation. iii, v, xiii, 2, 3, 5, 39, 40, 45, 55, 75–77, 83, 86

IPA Interface de Programação de Aplicações. 39

LSTM Long Short Term Memory. 24, 28, 49, 50, 80

MA *Microsoft Azure*. xiii, 34, 37

PaaS *Platform as a Service*. 33, 34

PDF *Portable Document Format*. 19, 28, 50, 52, 55

PLN Processamento de Linguagem Natural. xiii, 2–4, 7, 11, 13–17, 22, 23, 30, 32, 40, 42, 85

RCO Reconhecimento de Caracteres Óptico. v, 2, 4, 7, 13, 18–21, 23, 28, 30, 34, 38, 40, 45, 50, 52, 53, 55, 56, 58, 59, 77

RNA Redes Neuronais Artificiais. 7, 8, 10–12, 21, 24

RNC Rede Neuronal Convolucional. 9, 11, 24, 29, 31, 51

RNN Rede Neuronal Recorrente. 11, 24

RNP Redes Neuronais Profundas. 9–11

RNS Redes Neuronais Superficiais. 10

SBE Sistemas Baseados em Exemplos. xv, 23, 24, 27, 32

SBR Sistemas Baseados em Regras. 2, 23, 24, 26, 32, 40, 77

UBL Universal Business Language. 28

VC Visão Computacional. xiii, 2–4, 7, 11, 17–19, 22, 23, 30, 32, 42, 45

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Este documento espelha o trabalho desenvolvido durante o estágio curricular com duração de um ano letivo integrado na área de Sistemas Inteligentes do Mestrado em Engenharia Informática, do Departamento em Engenharia Informática da Universidade de Coimbra.

O estágio curricular decorreu na empresa CSW e foi supervisionado pelo orientador Doutor Jorge Henriques (docente da Universidade de Coimbra) e pelo Doutor Tiago Baptista por parte da CSW.

Este capítulo elabora um resumo geral deste relatório. A secção 1.1 detalha o contexto e as motivações que levam à realização do estágio, na secção 1.2 são explicitados os objetivos que se pretendem com este estágio e as estratégias a desenvolver para a sua concretização e, por fim, na secção 1.3 é descrita a estrutura geral do documento.

1.1 Contexto e Motivações

A evolução tecnológica que tem marcado o desenvolvimento mundial nos últimos anos e a necessidade de poder utilizar todo o tipo de informação de forma praticamente instantânea, torna imprescindível a sua disponibilidade e fácil acesso. A possibilidade de digitalizar todo o tipo de informação veio agilizar muito este processo. Contudo, a linguagem utilizada pelos seres humanos, expressa em muito documentos, é pouco estruturada. Embora siga regras gramaticais, a linguagem humana frequentemente é rica em símbolos e sequências que podem ser interpretadas de múltiplas formas, em diferentes contextos, podendo tornar o discurso ambíguo.

Os sistemas informáticos não têm a capacidade de, por si só, conseguir compreeender a informação não estruturada. Para simplificar o processo de gestão e pesquisa de informação, por sistemas informáticos, há a necessidade de estruturar esse tipo de informação.

A gestão e organização de documentos é dependente de alguns processos, entre eles: a classificação de documentos e a extração de dados. Na classificação, é efetuada a sua categorização com base em propriedades visuais ou textuais do documento. Na extração de dados, é recolhida a informação mais relevante de forma estruturada e classificada. Para além disso, o processo de validação de documentos assegura que os processos anteriores foram bem efetuados.

Inicialmente, os processos de gestão de documentos eram efetuados por operadores humanos, no entanto, estes processos, frequentemente com falhas, tornavam-se muito morosos e repetitivos, implicando elevados custos nas empresas. Estas dificuldades desafiaram as organizações a criarem processos alternativos e levaram ao desenvolvimento de processos automáticos de classificação e extração. Embora hoje em dia algumas organizações mantenham esforço humano nestes processos, não dispensam o sistema automático de forma a aumentar a confiança na extração e reduzir a suscetibilidade a falhas.

O crescimento exponencial de documentos guardados em sistemas digitais tornou crítico o desenvolvimento de sistemas de categorização dos diferentes tipos de documento. Prevê-se que num futuro próximo, este seja o tipo de dados predominante em sistemas online [19]. É propósito deste estágio contribuir para o desenvolvimento do serviço de extração e análise de informação de documentos, que melhorem o seu processo de gestão.

1.2 Enquadramento e Objetivos

A CSW está, neste momento, a desenvolver uma solução que visa automatizar os processos de classificação e extração de documentos. O sistema IDV classifica o documento numa categoria pré-definida e extrai os dados mais relevantes. O objetivo deste sistema é aliviar o trabalho humano neste processo e tornar a extração de dados e classificação de documentos mais eficiente e menos suscetível a erros.

Os documentos que são tratados pelos serviços da plataforma IDV são, sobretudo, documentos estruturados, onde a informação se apresenta numa forma concisa, geralmente em formato de formulário, como é o caso dos certificados de nascimento, faturas, cartões de cidadão, passaportes, entre outros. Apesar de esse ser o foco principal, esta plataforma também é capaz de trabalhar com documentos menos estruturados, com maior informação textual e até manuscriptos, utilizando para isso várias técnicas de Processamento de Linguagem Natural (PLN).

Neste estágio, irei integrar a equipa de desenvolvimento da área de Inteligência Artificial (IA) da CSW responsável pela plataforma do IDV. O foco do meu estágio irá incidir, sobretudo, sobre o **serviço de extração** de informação de documentos. O objetivo principal é desenvolver uma prova de conceito de métodos de extração e compreensão automática de campos de documentos estruturados a partir de técnicas de Visão Computacional (VC) e PLN.

De forma mais aprofundada, o serviço atual de extração é um sistema com suporte em diferentes técnicas, mais concretamente, contém padrões e regras (Sistemas Baseados em Regras (SBR)), técnicas de PLN e Reconhecimento de Caracteres Óptico (RCO), entre outros. Embora este serviço tenha resultados aceitáveis, tem algumas limitações patentes, nomeadamente, a necessidade de efetuar uma implementação *ad-hoc* para cada tipo de documento e estrutura, o que reflete na falta de escalabilidade de adaptação para diferentes estruturas de documentos.

Na Figura 1.1 podemos visualizar o exemplo de um passaporte falso submetido na plataforma IDV.

Field	Extracted Value
Given name	STEVEN
Surname	LEE
Document number	N988888
Nationality	PRT
Date of birth	1994-01-26
Date of issue	2015-10-29
Expiration date	2020-10-29
Gender	M

Extra field	Extracted Value
MRZ - Document type	P
MRZ - Document subtype	<
MRZ - Issuing country	PRT
MRZ - Given name	STEVEN
MRZ - Surname	LEE
MRZ - Document number	N988888
MRZ - Nationality	PRT
MRZ - Date of birth	1994-01-26
MRZ - Expiration date	2020-10-29
MRZ - Personal number	13778777
Photo	–
Estimated minimum age	19

Figura 1.1: *Front-end* da plataforma IDV

Como é possível visualizar, do lado direito está apresentado o documento submetido, enquanto do lado esquerdo, estão os campos que foram extraídos do documento pelo serviço de extração da plataforma IDV.

O objetivo da CSW é tornar o serviço de extração mais genérico e a implementação automática, independentemente da estrutura e tipo de documento. Nesse ponto de vista, é necessário implementar métodos menos restritos e mais abrangentes de forma a poder lidar com documentos novos e efetuar a extração e compreensão de informação de forma independente à sua estrutura. São objetivos deste estágio:

- Explorar técnicas de PLN, de VC e de Aprendizagem Computacional (AC) no domínio do problema
- Explorar, implementar e testar algoritmos, com base no estado da arte, a partir de artigos de investigação (não é fornecido o código fonte).
- Implementar uma prova de conceito de um sistema baseado em um (ou mais) dos algoritmos testados.

De realçar, que embora a prova de conceito a ser implementada seja integrada na arquitetura da plataforma IDV, não é objetivo deste estágio integrar a o sistema proposto na versão de produção da plataforma. Como foi enunciado anteriormente, o estágio tem como finalidade desenvolver uma prova de conceito de um dos algoritmos de compreensão de documentos estruturados. Esta pretende verificar se:

- Os algoritmos baseados em exemplos conseguem ter desempenhos aceitáveis em ambiente de negócio.

- O desempenho do algoritmo escolhido para a prova de conceito consegue ser superior ou pelo menos não muito inferior ao serviço já implementado — se o desempenho não for muito inferior ao serviço já implementado, pode ser interessante para a empresa considerar este algoritmo, visto que traz inúmeros benefícios, sobretudo na adaptação para novos documentos, que com o serviço atual baseado em regras necessita de muito esforço humano na implementação.

- É possível tornar o serviço mais genérico, isto é, se o algoritmo é versátil e é capaz de ter desempenhos aceitáveis com diferentes documentos.

Sendo desejável que a prova de conceito evidencie um bom desempenho na extração e compreensão de dados de documentos, o sucesso deste estágio não depende desse resultado. Este estágio contém três componentes, todos de grande importância, que são a exploração de algoritmos do estado da arte através da análise dos seus artigos de investigação, implementação dos algoritmos e investigação e análise do seu desempenho. Não sendo um estágio puramente de Engenharia de *Software*, nem de exploração e investigação, as condições de sucesso devem incidir sobre a concretização dos três objetivos definidos para o estágio, pelo que o estágio deverá ser bem sucedido se permitir:

- A exploração, identificação e análise correta dos principais algoritmos referenciados no estado da arte para extração e compreensão de documentos.

- A implementação dos algoritmos selecionados a partir de artigos de investigação, mesmo que estes não contenham toda a informação necessária para a sua replicação.

- A análise do desempenho dos algoritmos implementados e identificação dos seus pontos fortes e fragilidades.

1.3 Estrutura do Documento

Este relatório está dividido em 6 capítulos. Os tópicos abordados em cada um dos restantes capítulos são explicitados de seguida:

- **Capítulo 2 - Fundamentação Teórica e Estado da Arte**

  Neste capítulo são apresentados alguns fundamentos teóricos da área que se pretende explorar e é analisado o estado da arte atual. Para melhor compreensão do tema em estudo, optou-se por iniciar este capítulo com a conceptualização dos principais temas de IA abordados no estado da arte. Na secção 2.1 aborda-se a área da AC, na secção 2.2 irá ser abordada a área de PLN, na secção 2.3 irá ser abordada a área de VC, na seção 2.4 irá ser abordada a área de RCO, sendo que na na secção 2.5 irão ser abordadas as técnicas utilizadas na extração e classificação de documentos. Por fim, as secções 2.6 e 2.7, serão dedicadas à revisão da literatura de métodos de extração e compreensão de documentos e a testes efetuados a serviços idênticos, respetivamente.

- **Capítulo 3 - Especificação do Sistema**

  Neste capítulo é abordada a especificação do sistema. Na introdução irei fazer uma especificação de alto nível do sistema da plataforma do IDV, focando mais concretamente no serviço que pretendo explorar. Na secção 3.1 irei fazer a análise dos riscos e respetivas estratégias de mitigação, na secção 3.2 irei explicitar as metodologias e ferramentas a serem utilizadas no decorrer do projeto. Na secção 3.3, irei especificar o planeamento do estágio, sendo que na secção 3.4, irei detalhar os algoritmos que irei implementar e na secção 3.5 irei explicar como irei validar os algoritmos descritos na secção anterior.

- **Capítulo 4 - Testes e Resultados**

  Neste capítulo serão apresentados os testes realizados nos vários algoritmos e os respetivos resultados. Na secção 4.1, detalham-se os testes de optimização que serão efetuados a alguns algoritmos e na secção 4.2, serão apresentados os testes de desempenho de todos os algoritmos implementados.

- **Capítulo 5 - Análise e Discussão**

  Neste capítulo, apresenta-se a discussão dos resultados obtidos e efetua-se uma análise aos algoritmos implementados.

- **Capítulo 6 - Conclusão**

  Neste último capítulo, apresenta-se uma síntese do trabalho desenvolvido em cada uma das etapas do estágio e as principais conclusões.

Capítulo 2

Fundamentação Teórica e Estado da Arte

Este capítulo irá incidir sobre o estado da arte dos sistemas de validação inteligentes de documentos. Em primeiro lugar, irei falar sobre Aprendizagem Computacional (secção 2.1). Existem muitos técnicas de AC que são utilizados em projetos semelhantes, nomeadamente algoritmos de Redes Neuronais Artificiais (RNA) que serão utilizados nos métodos de extração que irei implementar. A secção 2.2 irá incidir sobre a área de PLN. Algumas técnicas de PLN são incluídas em alguns algoritmos de extração e classificação de documentos. Principalmente, em cenários onde os documentos estão em formato de texto corrido, a utilização de técnicas de PLN são cruciais para o bom desempenho do sistema. A secção 2.3 irá abordar a área de VC. As técnicas de VC são também muito utilizadas nos métodos de extração e classificação, nomeadamente, na preparação de documentos muito estruturados e em formato de formulário (como é o caso de faturas) a leitura espacial do documento é crucial para o desempenho do sistema. Na secção 2.4, irei fazer uma pequena introdução na área de RCO. As técnicas de RCO são utilizadas numa fase inicial do sistema, na leitura dos caracteres dos documentos que estão digitalizados. Por último, na secção 2.5 irei abordar os métodos de extração e classificação de documentos e analisar o estado da arte atual deste tipo de sistemas. No final desta última secção, apresento os resultados dos testes que efetuei a serviços relacionados com o pretendido para este estágio.

2.1 Aprendizagem Computacional

A AC é um subcampo da área da IA que se foca na aprendizagem automática através da experiência. Uma das técnicas de AC são as RNAs. As RNAs são um conjunto de algoritmos que permitem fazer o reconhecimento de padrões e previsões. As RNAs foram inspiradas no modelo biológico humano, mais concretamente pelo cérebro humano, e são utilizadas para resolver problemas computacionais que não conseguem ser solucionados com técnicas tradicionais, como são os casos onde é complicado explicitar o conhecimento necessário ou as relações entre os dados. Atualmente são aplicadas em praticamente todas as áreas, desde a agricultura até à medicina [2]. O conceito de RNAs foi abordado pela primeira vez em 1943

quando o fisiólogo *Warren McCulloch* e o matemático *Walter Pitts* quiseram demonstrar como os neurónios se relacionavam entre si nos sistemas biológicos. Para esse feito, conseguiram implementar um modelo de uma rede neuronal com recurso a circuitos elétricos. No entanto, a primeira aplicação das RNAs para previsão remonta apenas ao ano de 1964. Hu (1964) tentou fazer a previsão meteorológica com recurso a uma rede linear adaptativa de Widrow. Ainda assim, os resultados foram pouco entusiasmantes, devido sobretudo à falta de algoritmos e dados de treino. No ano de 1986, pela primeira vez se introduziu o algoritmo de retropropagação o que desencadeou uma revolução na forma como as RNAs aprendiam. As RNAs passaram então a ser metodologias viáveis na previsão e classificação [73].

Redirecionando para a arquitetura das RNAs, estas são compostas por uma camada de entrada (*input layer*), uma ou várias camadas escondidas (*hidden layers*) e uma camada de saída (*output layer*), como podemos observar na Figura 1.

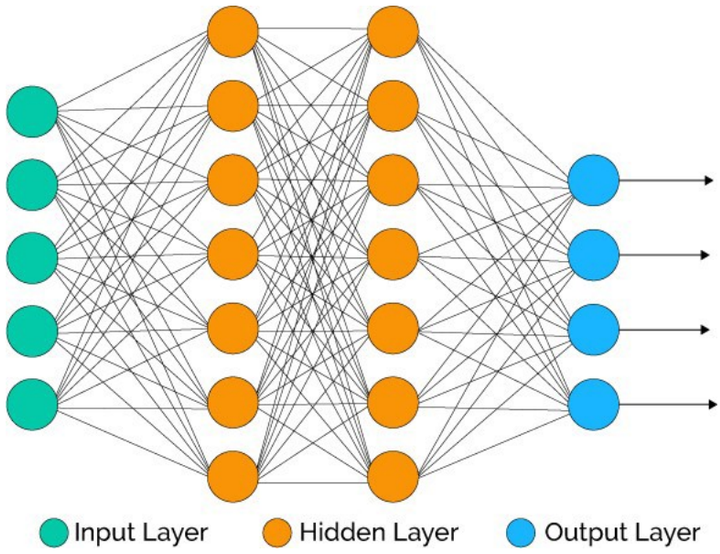


Figura 2.1: Rede Neuronal Artificial [4]

Cada camada é composta por diversos nós denominados por "neurónios". Os nós de cada camada vão se ligar a cada um dos nós da camada posterior, sendo que os valores de saída da camada anterior irão ser os valores de entrada da camada seguinte. Em cada ligação é atribuido um peso sináptico, que corresponde à força que cada valor de saída da camada anterior vai ter em cada nó da camada seguinte. Assim que um nó recebe os valores de entrada (V_e), efetua um cálculo linear, dando origem a um valor intermédio. Esse valor é depois embutido numa função de ativação que retorna o valor de saída do nó (V_s). O valor intermédio é obtido pela seguinte fórmula:

$$V_{int} = \sum_{i=0}^{m} w^{(i)} * V_e^{(i)} \ + b$$

sendo que m é o número de ligações entre os nós da camada anterior com a camada atual, o $w^{(i)}$ corresponde ao valor do peso sináptico da ligação i e o b corresponde ao valor da bias, uma constante de cada camada cujo objetivo é mover a função de ativação para a esquerda ou direita e tem um grande impacto no treino destas redes. O V_{int} não pode ser o valor de saída do nó porque o resultado que obtido é resultado de uma função linear e uma

"função linear é um polinómio de primeiro grau [...] eles são limitados na sua complexidade e têm menos poder para aprender mapeamentos funcionais complexos através dos dados" [67]. Foi necessário criar funções de ativação para introduzir propriedades não lineares ao nó. As funções de ativação são regras matemáticas que transformam os valores em sinais de saída. Na sua grande maioria, as funções de ativação não lineares normalizam os dados num intervalo fechado, geralmente entre 0 e 1, ou entre -1 e 1. Há diversas funções de ativação, mas as mais conhecidas estão apresentadas na Figura 2.2.

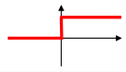
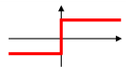
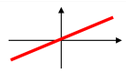
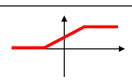
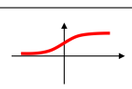
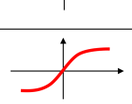
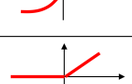
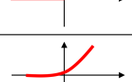
Activation function	Equation	Example	1D Graph
Unit step (Heaviside)	$\phi(z) = \begin{cases} 0, & z < 0, \\ 0.5, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Sign (Signum)	$\phi(z) = \begin{cases} -1, & z < 0, \\ 0, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Linear	$\phi(z) = z$	Adaline, linear regression	
Piece-wise linear	$\phi(z) = \begin{cases} 1, & z \geq \frac{1}{2}, \\ z + \frac{1}{2}, & -\frac{1}{2} < z < \frac{1}{2}, \\ 0, & z \leq -\frac{1}{2}, \end{cases}$	Support vector machine	
Logistic (sigmoid)	$\phi(z) = \frac{1}{1 + e^{-z}}$	Logistic regression, Multi-layer NN	
Hyperbolic tangent	$\phi(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$	Multi-layer Neural Networks	
Rectifier, ReLU (Rectified Linear Unit)	$\phi(z) = max(0, z)$	Multi-layer Neural Networks	
Rectifier, softplus	$\phi(z) = \ln(1 + e^{z})$	Multi-layer Neural Networks	

Figura 2.2: Funções de Ativação [58]

As funções de ativação podem ser divididas em dois tipos: lineares e não lineares. Na linear, o valor de saída não está confinado entre nenhum intervalo. Dentro das funções não lineares, eles diferem pela sua curvatura e intervalos abrangidos.As funções de ativação *Signum* e *Heaviside* em forma de escada, cuja diferença é que na *Signum* o valor mínimo é -1 e no *Heaviside* o mínimo é o valor 0. Ambas as funções têm um valor máximo de 1. Na função de ativação linear, não há qualquer operação sobre o valor de saída. A função sigmoidal abrange o intervalo entre 0 e 1 e a sua curvatura é em formato de S. Esta função é muito usada quando se pretende prever o valor de probabilidade de uma classe. A tangente hiperbólica também tem uma curvatura em formato de S, no entanto o intervalo abrangido é entre -1 e 1. Esta função é maioritariamente utilizada na classificação entre duas classes. a função *ReLU* assume o valor de saída 0, quando o valor de entrada é negativo, assumindo um valor linear de saída, quando o valor de entrada é positivo. Esta função de ativação é muito usadas em Redes Neuronais Profundas (RNP), nomeadamente em Rede Neuronal Convolucional (RNC). A softplus é uma função de ativação que suaviza a curvatura do *ReLU*. Esta função de ativação tende a assumir um valor de saída nulo (valor positivo muito reduzido), quando o valor de entrada é negativo, começando a assumir valores de saída próximos de valores lineares assim que os valores de entrada são positivos.

Quando uma RNA é criada, na maioria das vezes, os pesos das ligações dos nós são inicializados com valores aleatórios. Quando são efetuados os cálculos acima descritos com pesos aleatórios, na grande maioria dos casos não é possível alcançar os valores desejados. É então preciso arranjar uma forma de corrigir os valores desses pesos para uma melhor eficiência da RNA. Essa correção é efetuada no treino da RNA. O treino da RNA é efetuado para corrigir os valores dos pesos e das *biases*, recorrendo a diversas técnicas. Uma das técnicas mais populares é a técnica de retropropagação. Esta técnica é dividida em dois passos: o primeiro passo denomina-se por *"feedforward"* em que são introduzidos os valores de entrada e os valores de saída esperados. O valor é calculado desde as primeiras camadas até à camada de saída. Assim que é obtido o valor de saída, inicia-se o 2º passo. No segundo passo, é efetuada a diferença entre o valor obtido e o valor esperado que se dá por "erro" e a rede propaga a correção do erro para todas as camadas no sentido inverso. Neste processo, é permitido que sejam efetuadas correções nos pesos sinápticos através de uma equação, cujo o novo valor do peso passa a ser igual à multiplicação do coeficiente de aprendizagem (definido na implementação da rede neuronal), com o gradiente local e o sinal de entrada [16].

As redes neuronais podem ser divididas em diferentes tipos, umas deles são: as Redes Neuronais Superficiaiss (RNSs), ou *Shallow Neural Networks*, e as RNPs, ou *Deep Neural Networks*. As RNSs são as redes mais simples e contém poucas camadas escondidas (*hidden layers*) de nós. Estas podem ser *Feed Forward*, isto é, a informação é transmitida apenas no sentido frontal das primeiras camadas para as últimas ou podem ser *Feed Backward*, redes que conseguem guardar informação recorrendo ao uso de memória interna. As RNPs são mais complexas contendo várias camadas de nós e ligações complexas entre elas. Nos primeiros anos, as únicas RNAs utilizadas eram as RNSs. Estas aprendiam rápido e não precisavam de muitos dados de treino para conseguir uma performance aceitável. Por outro lado, durante o treino, a sua performance chegava a um ponto em que estagnava independentemente se lhe fossem dados mais casos de treinos. Nessa altura, as RNPs tinham um desempenho fraco comparativamente com outras técnicas de aprendizagem computacional. Com o virar do século, a necessidade de ter todo o tipo de informação disponibilizada virtualmente juntamente com o crescente poder dos computadores, permitiram que o peso das RNPs começasse a aumentar. Com o aumento de dados de treino, verificou-se um aumento de performance por parte destas redes e ao contrário das RNSs, estas conseguem ser alimentadas e melhoradas com muito mais dados de treino atingindo desempenhos muito superiores. Na figura 2.3, é possível comparar o desempenho das diferentes técnicas de aprendizagem computacional em função dos dados de treino disponíveis. No início da era de *Big Data* as técnicas tradicionais de aprendizagem computacional, tais como, as *Support Vector Machines*, as *Random Forest* e *Logistic Regression*, conseguiam com poucos dados melhorar o seu desempenho significativamente, no entanto, estas técnicas tendiam a estabilizar em valores não muito altos (comparativamente com as redes neuronais) independentemente do número de dados que lhes era atribuído. As redes neuronais de pequenas (*shallow*) e médias dimensões, não tinham um tão bom desempenho inicial, mas se lhes fossem dados um grande

conjunto de dados, o desempenho conseguia ser superior, no entanto, também estabilizavam num certo valor, independentemente dos dados inseridos. Ultimamente, tem se investigado muitos em RNP que, apesar de necessitarem de muitos dados e inicialmente terem resultados pouco satisfatórios, conseguiam atingir resultados muito superiores comparativamente com as outras técnicas.

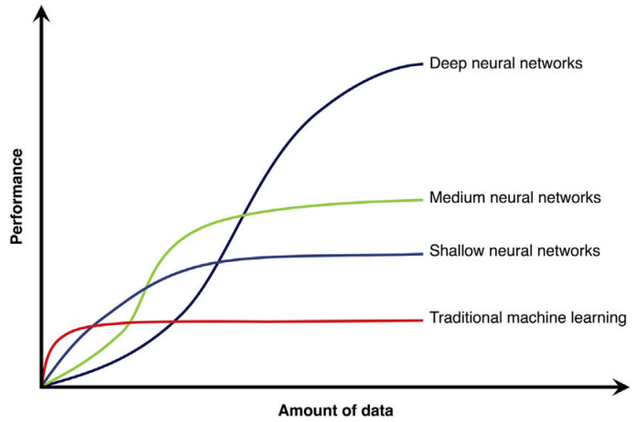


Figura 2.3: Performance das Redes Neuronais vs Quantidade de dados de treino. [65]

Dentro das RNPs, existem dois tipos de redes que irei aprofundar neste estágio: a Rede Neuronal Convolucional (RNC) e a Rede Neuronal Recorrente (RNN). Atualmente, os métodos de *Deep Learning* têm tido impacto nas áreas de reconhecimento de voz, de PLN, de VC, reconhecimento de padrões e classificação de imagens.

As técnicas de aprendizagem computacional são divididas em dois grupos: *Supervised Learning* e *Unsupervised Learning*. Quando um professor efetua a correção de um exame, é esperado que os alunos aprendam como responder às questões para que no próximo exame tenham uma melhor classificação. As técnicas de *Supervised Learning* baseia-se no exemplo anterior, sendo assim, ao treinar a RNA, esta tem acesso à resposta correta e assim consegue ajustar os valores dos pesos sinápticos e da bias para uma melhor classificação. O *Supervised Learning* permite otimizar mais eficientemente o critério de aceitação recorrendo a experiências anteriores e soluciona diversos tipos de problemas computacionais reais. No entanto, nem sempre o acesso a uma resposta correta é possível ou sequer faz sentido. Recorrendo a outro exemplo, imaginando uma família com um bebé em fase de aprendizagem. Esse bebé viu pela primeira vez uvas na cesta da sua casa. Passado uns tempos, na mesma cesta vê pela primeira vez bananas na sua casa. Não foi preciso dizer ao bebé que as duas frutas são diferentes para ele conseguir distinguir. Através das características das frutas, como cor, formato e textura, ele conseguiu identificar elementos que as distingam e assim, conseguiu identificar como frutas diferentes. As técnicas de *Unsupervised Learning* baseiam-se no exemplo anterior. Apesar de não haver nenhuma variável de saída que indique o valor correto, a partir do reconhecimento de padrões de exemplos anteriores, estas conseguem identificar corretamente. Há algumas vantagens no uso de técnicas de *Unsupervised Learning*. Os métodos existentes permitem identificar características importantes para classificar ou categorizar — características essas que muitas vezes, nós humanos não identificamos — e é mais fácil obter dados não organizados, como imagens ou áudios, do que dados organizados, como tabelas

de características, porque estas últimas necessitam de intervenção humana [Guru99].

Nesta secção, foram abordadas diferentes técnicas de AC — com especial foco nas RNA —, nas técnicas de aprendizagem e os diferentes desempenhos que estas normalmente conseguem obter. No entanto, não basta treinar um modelo de IA para o desempenho ser automaticamente calculado, até porque não existe nenhuma fórmula única de calcular o desempenho. Não faz sentido que o desempenho de uma IA que identifica modelos de carros numa imagem seja calculado da mesma maneira que uma IA que faz transcrição de voz para texto. Por isso, existe uma solução que se chama métricas. As métricas são fórmulas que permitem calcular o desempenho de uma IA. Tal como há inúmeras áreas de ação de uma IA, também há inúmeras métricas que permitem calcular o desempenho desses modelos. Algumas delas são:

- **Precisão -** De todos os resultados previstos positivos (Verdadeiros Positivos e Falsos Positivos), a precisão indica qual é o rácio de acerto (Verdadeiros Positivos). Por exemplo, "de todos os valores reconhecidos como número de fatura pelo nosso serviço, qual é o rácio de acerto no reconhecimento?". Esta fórmula é utilizada quando os falsos positivos têm um impacto significativo. A fórmula é a seguinte:

$$Precisao = \frac{\#[VerdadeirosPositivos]}{\#[VerdadeirosPositivos] + \#[FalsosPositivos]}, \tag{2.1}$$

- ***Recall* -** Também conhecida como sensibilidade, de todo o conjunto de dados reais positivos (Verdadeiros Positivos e Falsos Negativos), o *recall* indica o rácio de acerto (Verdadeiros Positivos). Por exemplo, "de todos os números de fatura presentes nos documentos, qual é a proporção de reconhecimento bem efetuado pelo nosso serviço?" Esta fórmula é utilizada quando os falsos negativos têm um impacto significativo. A fórmula é a seguinte:

$$Recall = \frac{\#[VerdadeirosPositivos]}{\#[VerdadeirosPositivos] + \#[FalsosNegativos]}, \tag{2.2}$$

- ***F1 Score* -** Métrica de avaliação que considera tanto a precisão do modelo com a sua sensibilidade. O resultado do F1 Score é melhor quando existe um balanceamento da precisão e da sensibilidade. Por outro lado, quando um dos valores aumenta em detrimento do outro, o *F1 Score* tende a ter piores resultados. A fórmula é a seguinte:

$$F1Score = 2 * \frac{[Precisao] * [Recall]}{[Precisao] + [Recall]}, \tag{2.3}$$

- ***Intersection-Over-Union* -** Nesta métrica são considerados dois tipos de variáveis: a *bounding box* prevista pelo sistema e a *bounding box* verdadeira (bounding box etiquetada manualmente). Esta consiste em medir o rácio da área intersectada pelas duas *bounding boxes (overlapped area)* com a área formada pela união das duas *bounding boxes (union area)*. Esta fórmula é utilizada quando se pretende medir a precisão de uma área prevista (quer seja de um objeto ou de um campo relevante). A fórmula é a seguinte:

$$IoU = \frac{[Area\_of\_Overlap]}{[Area\_of\_Union]}, \tag{2.4}$$

- ***Katti's Formula (Eq. 2.5)* -** Métrica de avaliação que, segundo Katti et al. [33], permite medir quanto trabalho poderá ser poupado na utilização de um sistema automático de extração comparativamente a uma extração manual. Esta métrica pode resultar em valores negativos, o que nesse caso, significaria que era preferível efetuar a extração manual. Para cada palavra extraída, esta métrica calcula o número de inserções, remoções e modificações da palavra original. A palavra original é obtida do resultado do RCO, para que erros de RCO não sejam contabilizados nesta métrica. A fórmula, já descrita anteriormente (Eq. 2.5), é a seguinte:

$$1 - \frac{\#[insercoes] + \#[eliminacoes] + \#[modificacoes]}{N}, \tag{2.5}$$

  sendo N o número total de palavras.

- ***Average Precision* -** Idêntica à metrica de precisão, esta métrica calcula a precisão de acerto ao nível da classe, sendo que apenas considera a classe como certa, se todas as palavras contidas nela estiverem totalmente corretas. A fórmula é igual à da precisão.

2.2 Processamento de Linguagem Natural

O PLN, ou *Natural Language Processing*, é uma área da IA que tem princípios da área Linguística e que oferece "uma gama de técnicas computacionais que analisa e representa textos que ocorrem naturalmente em um ou mais níveis de análise linguística, com o objetivo de obter processamento de linguagem semelhante ao humano [...] [os textos] podem ser em qualquer linguagem [...] oral ou escrito [...] usada por humanos para comunicar uns com os outros" [39]. Nos primórdios da IA, o PLN era designado por Compreensão de Linguagem Natural (CLN), ou *Natural Language Understanding*, tendo a designação sido alterada por ainda não ser possível atingir a compreensão total de um texto introduzido [39]. Segundo Liddy (2001), para um sistema ser considerado CLN, tem de ser capaz de parafrasear um texto introduzido, traduzir o texto para uma diferente linguagem, ter a capacidade de responder a questões e poder retirar conclusões sobre o mesmo, independentemente do tipo de texto ou o tópico em questão.

O PLN aborda diversas áreas, tais como — a área linguística onde se foca nos modelos formais e estruturados das linguagens humanas, a psicologia cognitiva onde aborda os processos cognitivos dos seres humanos quando comunicam e a ciência da computação onde se foca sobretudo no processamento das estruturas das linguagens. O PLN pode ser aplicado em diversas situações do quotidiano, isto é, com esta tecnologia é possível efetuar:

- **Extração de Informação:** Retirar a informação principal de um texto e transformá-la numa representação estruturada

- **Sugestão de Literatura:** Fornecer uma lista de documentos interessantes sobre o tema abordado na questão ou comentário efetuado/a pelo utilizador

- **Resposta a Questões:** Idêntico ao fornecimento de literatura, no entanto, neste caso a resposta/explicação é fornecida diretamente pelo próprio sistema. Já diversos motores de pesquisa adotaram esta funcionalidade (ex: Google).

- **Resumo:** Abreviação da representação narrativa de um texto.

- **Classificação de texto:** Categorização de um texto ou excertos de texto. A maioria dos serviços de *Webmail* já utilizam esta funcionalidade para categorizar os *e-mails* (publicidade, spam,...)

- **Tradução:** Tradução de um texto para uma outra linguagem. O serviço mais conhecido que utiliza esta funcionalidade é a *Google Translate*.

- **Sistemas de *chat* e diálogo:** Implementação de um *chatbot* ou de um assistente pessoal. Os serviços mais conhecidos que utilizam esta funcionalidade são a *Siri* e a *Google Assistant*.

O PLN pode ser decomposta em duas abordagens: o processamento de linguagem natural onde o sistema analisa a linguagem e produz uma representação compreensiva do texto (semelhante a um leitor ou ouvinte), e a geração de linguagem natural onde o sistema é capaz de gerar texto natural, isto é, interpretável por um ser humano, a partir de uma representação compreensiva (semelhante a um escritor ou orador). Para o sistema de PLN conseguir compreender o texto introduzido, ele tem de passar por diversas etapas/camadas, umas vez que o processamento da linguagem exige uma abordagem dinâmica, dado que o significado do texto pode variar em função de diversos fatores [39]. Se um documento lido pelo sistema de PLN for um documento tecnológico, as palavras que podem assumir vários significados devem ser interpretadas pelo seu sentido tecnológico. Por exemplo, a palavra *"cache"* tem vários significados, podendo ser um esconderijo/lugar escondido onde se guardam coisas ou, no sentido tecnológico, ser um dispositivo de acesso rápido que está a servir de intermediário entre um processador e um dispositivo de armazenamento. Cada camada do PLN incide sobre um fator diferente que influencia o significado das frases. De acordo com Liddy (2001), as camadas são as seguintes:

- **Camada Fonológica:** O processo de identificação e produção de partes da linguagem oral faz parte da área da consciência fonológica do ser humano. Um fonema é a menor unidade sonora de uma linguagem e que combinados entre si dão origem a sílabas e palavras. Na língua portuguesa, existem cerca de 31 fonemas, 12 correspondentes às vogais e os outros 19 correspondentes às consoantes (por exemplo, /ó/, /ô/ e /lh/ são todos fonemas diferentes). Esta camada do PLN é responsável pela interpretação dos sons de cada palavra assim como das ligações entre as palavras.

- **Camada Morfológica:** Uma palavra é composta por morfemas, que são os elementos mais primitivos da linguagem e que representam a unidade mais simples de significado. Dando um exemplo, a palavra "deslealdade" pode ser dividida morfológicamente em

três componentes: o prefixo "des" (que significa ação contrária), o radical "leal" e o sufixo "dade" (formação de substantivo abstrato). Ao contrário das palavras, os morfemas têm apenas um significado e a partir deles o ser humano pode compreender o significado da palavra mesmo que não conheça a palavra. De forma análoga, através do conhecimento dos morfemas, o sistema PLN é capaz de retirar conclusões sobre o significado de uma palavra mesmo que não a conheça, desde que tenha conhecimento do significado do radical desta.

- **Camada Lexical:** Esta camada é responsável pela interpretação ao nível individual da palavra e o seu *part-of-speech*. Nesta camada de processamento, é atribuída uma *Tag* de *part-of-speech* a cada palavra, sendo que nas palavras que contém várias *tags*, é atribuída a que tem maior probabilidade. Esta camada de processamento linguístico utiliza um léxico de uma linguagem humana, composto por múltiplos lexemas, que são a menor unidade de significado. Quando queremos procurar o significado da palavras "estava", pesquisamos pela palavra "estar". Isto deve-se porque as palavras "estava" e "estar" pertencem ao mesmo lexema. Neste nível, as palavras que contém apenas um significado são substituídas pela representação lexical do seu significado, como por exemplo, se é um nome ou um verbo.

- **Camada Sintática:** Nesta camada é efetuada a análise das relações entre palavras da mesma frase. Para isso é necessário a gramática e analisador da linguagem em questão. Na maioria das linguagens, a sintaxe das frases fornece o seu sentido a partir da ordem das palavras. Por exemplo, as frases "o cão mordeu o homem" e "o homem mordeu o cão" a nível lexical são idênticas, no entanto a nível sintático fornecem significados diferentes. O resultado desta camada é obter uma representação da estrutura da frase, através das funções e/ou relações de dependência entre as palavras.

- **Camada Semântica:** Esta camada é responsável por identificar os significados possíveis da frase através da interação das palavras. Nesta camada, as palavras com múltiplos significados são desambiguadas semânticamente. Esta desambiguação é efetuada na camada semântica, ao invés da camada lexical, sempre que a informação das palavras necessita do resto da frase para ser interpretada. Nesta desambiguação é selecionado apenas um dos sentidos das palavras polissémicas e incluí-los na representação semântica da frase. Recorrendo a um exemplo, a palavra "andar" pode ter o significado de "dar passos" ("Ele foi andar a pé"), como pode ter o significado de desenvolvimento ("O projeto está a andar para a frente"). É esperado que esta camada tenha a capacidade de determinar o sentido correto da palavra a partir do tópico da frase. Na extração de informação, o processo de pesquisa e correspondência pode ser efetuado a um nível conceptual, em vez de se utilizarem termos simples. Assim, permite que o sistema tenha um maior desempenho nesta tarefa.

- **Camada de Discurso:** Ao contrário da camada semântica que efetua a análise ao

nível da frase, a camada de discurso analisa o texto como o seu todo, fazendo conexões entre as frases. Nesta camada pode ocorrer dois níveis de processamento:

  - **Resolução Anáfora:** Palavras que têm significado semântico vago, como pronomes, são substituídos pela entidade na qual eles se estão a referir. Por exemplo, nas frases "O jogador não me passa a bola. Ele é um bocado egoísta.", o pronome "Ele" deve ser substituído pela palavra "jogador".
  - **Reconhecimento da estrutura do texto:** Neste nível, é determinada a função de uma frase, consoante a secção da estrutura do texto no qual ela se insere, dando assim, maior significado ao próprio. Por exemplo, um artigo ou um texto dissertativo, é geralmente composto por uma introdução, desenvolvimento e conclusão.

- **Camada Pragmática:** Esta camada é responsável por interpretar a intenção do uso da linguagem para obter o significado real do texto. Para isso, este necessita de obter o contexto do mundo real, ou seja conter informação que não está incluída no texto, para conseguir compreender a verdeira intenção do texto. Por exemplo, nas frases "Os membros do governo proíbiram os transportadores de fazerem greve porque eles receavam que houvesse um caos" e "Os membros do governo proibiram os transportadores de fazerem greve porque eles têm um dever a cumprir", o pronome "eles" é ambíguo e só pode ser resolvido com recurso a conhecimento exterior.

Alguns sistemas de PLN incluem um pós-processamento para ajudar na tarefa de extração de dados textuais. Uma dessas técnicas é o N-grama. O N-grama é uma sequência de n-caracteres,n-fonemas ou n-palavras. Há alguns pares (ou conjuntos maiores) que aparecem probabílisticamente de forma mais frequente comparativamente a outros. Na língua portuguesa, a letra 'ç' é sempre seguida das vogais 'a', 'o' ou 'u', o par 'lh' também é constantemente seguido por qualquer uma vogal. O N-grama é baseado no modelo de multi-ordem de Markov e incide sobre a probabilidade da N-letra depender das N-1 letras anteriores. As sequências N-grama são muito utilizadas em sugestões de auto-completação de palavras e frases (a Google utiliza nos seus produtos), em correções ortográficas e reconhecimento de voz (nas palavras homófonas, como fato e facto) [49].

Uma outra técnica também muito utilizada na área de PLN é o *embedding* de palavras. Esta técnica permite estabelecer relações semânticas entre palavras, representando cada palavra num vetor numérico [23]. O objetivo do *embedding* é fazer com que as palavras que provenham de um contexto idêntico ocupem uma posição espacial próxima. Uma forma de verificar se duas palavras diferentes estão relacionadas entre si é calcular a similaridade do cosseno dos vetores produzidos no *embedding* das palavras [32]. Existem alguns algoritmos que se têm destacado nesta técnica, tais como o Word2Vec [46], o fastText [31] e o GloVe [54]. Em 2019, surgiu o modelo de linguagem BERT [8], também ele capaz de fazer *embedding* de palavras. Este algoritmo tem vindo a revolucionar a área de PLN pela inovação que trouxe na leitura e modelação textual. Ao contrário dos algoritmos anteriores, que aprendem

a ler o texto como uma sequência da esquerda para a direita ou como uma combinação das sequências esquerda-direita e direita-esquerda, o algoritmo BERT consegue ler a sequência de texto de uma só vez, permitindo que as palavras ganhem contexto com as restantes palavras à sua volta [25]. Considera-se este tipo de leitura de texto como não-direcional. O BERT faz recurso de mecanismos de atenção — uma componente que quantifica a dependência entre as palavras de entrada no modelo (auto-atenção) e, entre as palavras de entrada e saída do modelo (atenção geral) —, mais concretamente, de *Transformers* [66] e, neste momento, é considerado o estado da arte em 11 tarefas de PLN [8].

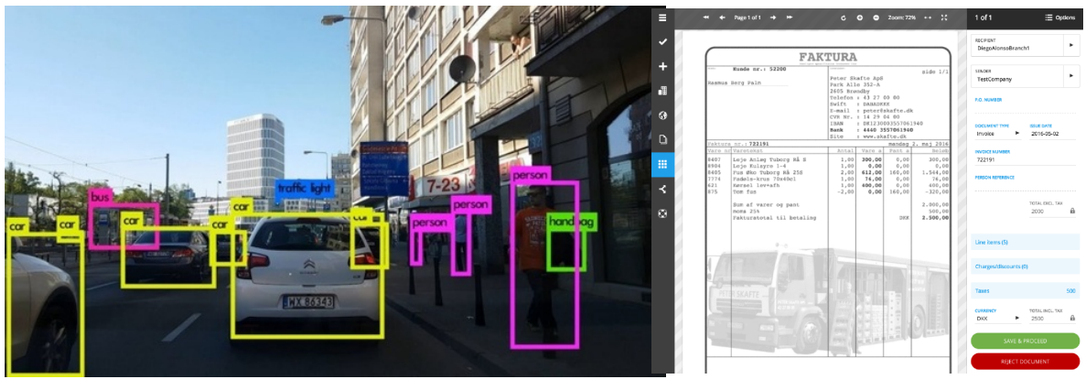
A área no qual o meu estágio se insere exige alguns conhecimentos de PLN. Esta tecnologia é usada em alguns algoritmos do estado da arte da extração de documentos. Embora a utilização exclusiva de técnicas de PLN não permita obter resultados satisfatórios (comparativamente com outro tipo de técnicas) em documentos de identificação (por exemplo, cartões de cidadão e passaportes) e comerciais (por exemplo, faturas), documentos esses em que a informação está muito estruturada e é classificável, a utilização desta tecnologia juntamente com técnicas de VC, permitiu obter resultados muito satisfatórios [74, 33, 7, 51]. Para além disso, em documentos com maior informação textual (por exemplo, alguns certificados de nascimentos têm um formato de texto expositivo/descritivo ao invés do tipo formulário), os princípios de PLN são muito úteis na retirada de informação relevante. Esses algoritmos aproveitam as relações semânticas, extraídas através do *embbeding* das palavras com recursos a modelos de linguagem, dos excertos de texto presente nos documentos para efetuar uma melhor categorização desses mesmos excertos (por exemplo, numa fatura identificar o campo "João Castilho" como o nome do cliente ou do vendedor). Mais à frente, na **secção 2.5** irei detalhar melhor a utilidade do PLN no que pretendo desenvolver e falar de alguns trabalhos relacionados que utilizam recursos desta tecnologia.

2.3 Visão Computacional

A visão computacional, ou Computer Vision, é "a ciência que desenvolve a base teórica e algorítmica na qual informações úteis sobre um objeto ou cena podem ser automaticamente extraídas e analisadas a partir de uma imagem observada, conjunto de imagens ou sequência de imagens" [17]. São inúmeras as razões que levaram ao crescimento exponencial da VC nos últimos anos. O crescente poder de processamento, da memória e da capacidade dos computadores, que juntamente com a melhoria das técnicas de AC, contribuíram fortemente para que estes sistemas se tornassem mais robustos [56]. No entanto, o estado de desenvolvimento tecnológico atual da VC ainda está longe das capacidades da visão humana.

Os seres humanos têm facilidade na capacidade de perceção da estrutura tridimensional do mundo que os rodeia. Quando estamos na rua, facilmente conseguimos identificar vários objetos (como carros, pessoas, sinais de trânsito...) conseguindo distinguir-los do fundo da imagem. Apesar de atualmente os sistemas de VC já serem capazes de identificar objetos numa imagem, de rastrear pessoas ou carros numa cena de vídeo e até desenvolver modelos

tridimensionais a partir da sobreposição de imagens, fazem-no de forma ainda rudimentar e de acordo com Szeliski(2010), "Poderá demorar vários anos até que os computadores sejam capazes de delinear e identificar objetos numa fotografia com a mesma capacidade que uma criança de 2 anos". O sistema de visão é extremamente complexo, embora esta complexidade tenda a ser subvalorizada pelas pessoas que não trabalham nesta área. Os seres humanos conseguem complementar a falta de informação patente numa imagem para conseguir retirar uma conclusão e isto ainda não está completamente conseguido na visão computacional, por exemplo, quando numa imagem duas pessoas estão sobrepostas, se apenas o pé de uma pessoa for visível, facilmente conseguimos concluir que se trata de uma pessoa. Embora ainda se reconheçam algumas limitações na VC, é inegável o desenvolvimento ocorrido nos últimos anos e a sua utilidade. A análise das figuras 2.4a e 2.4b, permite-nos observar que embora esta tecnologia seja muito complexa e exija muitos dados de treino, os resultados que hoje em dia já se conseguem atingir, permite-nos utilizá-la com qualidade em situações do dia a dia.



(a) Identificação de objetos [14]

(b) Extração de dados de uma fatura [52]

Figura 2.4: Exemplos da utilização prática de técnicas de Visão Computacional

Na figura 2.4a, podemos visualizar uma simulação da deteção de objetos por parte de um carro autónomo. Nesta imagem, podemos visualizar que para além de detetar a presença de um objeto, também consegue identificá-lo. Na figura 2.4b, observa-se uma captação de um software de extração de dados de faturas, o CloudScan [52], onde a visão computacional é util

A VC é atualmente usada em muitas áreas, tais como em sistemas biométricos (reconhecimento facial ou de impressão digital)[69], sistemas de vigilâncias (controlo de tráfico, deteção de intrusos...)[68], em veículos autónomos (deteção de obstáculos)[30], sistemas médicos (deteção de anomalias humanas)[47], em retalhos (reconhecimento de código de barras ou *QR codes*)[38], inspeção de máquinas (deteção de defeitos)[29], em sistemas de RCO (reconhecimento de caracteres óticos)[64] ou em sistemas de digitalização do documentos (reconhecimento da estrutura dos documentos, de forma a torná-la editável, e dos caracteres que o constituem)[33, 7, 74, 41].

Os algoritmos de extração que irei explorar durante o meu estágio contém princípios de VC. A estrutura de um documento indica muito sobre o próprio documento. Se observarmos

a estrutura de uma fatura, facilmente reconhecemos que esta contém poucas diferenças entre diferentes exemplares. Por exemplo, na fatura, os dados da empresa responsável pela venda geralmente são a primeira informação a ser mostrada, pelo que por norma fica situada no canto superior esquerdo. Ainda nas faturas, se esta contiver uma lista de itens, a probabilidade de ser uma lista de produtos ou serviços vendidos é grande. Geralmente, a seguir a uma lista, há um campo que indica um valor total a pagar. Posto isto, é possível reconhecer que a localização dos campos revela informação sobre o próprio campo. A análise da estrutura de documentos por via de técnicas de VC permite efetuar uma maior extração de informação dos documentos. Mais à frente, na **Secção 2.5** irei detalhar melhor o enquadramento destas técnicas no processo de extração de informação de documentos. Também irei detalhar alguns trabalhos relacionados que utilizam técnicas de VC neste processo.

2.4 Reconhecimento Óptico de Caracteres

"A maneira tradicional de introduzir dados para um computador é através do teclado. Contudo, não é a melhor solução nem a mais eficiente."[11] O Reconhecimento Óptico de Caracteres ou em inglês, Optical Character Recognition (OCR), é uma sub-área de VC e, como o nome indica, permite fazer o reconhecimento de caracteres de forma automática. De forma análoga a um sistema biológico humano, o RCO é como uma combinação dos olhos com o cérebro. O RCO permite através de imagens (geralmente documentos digitalizados) transformar dados de texto em formatos não estruturados em dados legíveis e editáveis [53]. O RCO contém algoritmos de reconhecimento de padrões no qual o seu princípio é ensinar a uma máquina quais são os padrões (letras, números, entre outros símbolos), divididos em classes, e respetivas características que espera encontrar. Este ensinamento é efetuado na fase de treino, através da visualização de vários exemplos de símbolos das diferentes classes, no qual posteriormente, a máquina é capaz de desenhar um protótipo com as características principais de cada classe. Durante a fase de reconhecimento, os símbolos a serem reconhecidos são comparados com os exemplos disponibilizados anteriormente e atribuídos à classe com características mais semelhantes [11].

A tecnologia de RCO nos dias de hoje consegue efetuar o reconhecimento tanto de caracteres manuscriptos como de caracteres impressos, no entanto o sucesso do seu reconhecimento é altamente dependente da qualidade e da complexidade estrutural do documento submetido, assim como artifactos introduzidos no processo de scanning [1]. O RCO é capaz de reconhecer documentos digitalizados ou capturados por fotografia, imagens e documentos *Portable Document Format* (PDF). Os casos em que o documento é capturado por fotografia (imagem) diferem dos que são digitalizados, isto porque a imagem pode estar um pouco desfocada, o documento não ter a orientação correta e a intensidade da luminosidade não ser a mais adequada. Apesar dos enormes avanços nesta área, ainda nenhum algoritmo de RCO é capaz de reconhecer corretamente 100% dos caracteres (há sempre caracteres que não são reconhecidos ou mal reconhecidos). O rácio médio de erro de reconhecimento ao nível de

palavra por ferramentas de RCO ronda os 1 a 10% [1]. De uma forma geral, quando ocorrem incertezas no reconhecimento, ou o algoritmo considera o seu melhor palpite em todos os caracteres, ou este alerta para os caracteres incertos, sendo necessário nesse caso um operador humano para confirmar.

A primeira abordagem funciona bem para documentos com informação contextual (artigos, cartas,...), mas tende a falhar para documentos mais estruturados (formulários, tabelas,...). Para estes documentos mais formatados, a segunda abordagem é a mais apropriada [59].

Passemos então a descrever como a tecnologia de RCO geralmente funciona. O primeiro passo para a utilização do RCO passa por digitalizar os documentos em formato de imagem. Em alguns sistemas, ocorre um passo adicional no qual esses documentos são convertidos para uma imagem binária, com o objetivo de poupar memória e esforço computacional. Depois dos documentos estarem em formato digital, são localizadas as regiões contendo informação textual através de processos de segmentação. Geralmente, a segmentação de páginas de documentos segue duas abordagens: *bottom-up* e *top-down*. A abordagem *bottom-up* deteta, em primeiro lugar, os caracteres tendo por base diversas características (por exemplo, relação de pixéis brancos e pretos e espaçamento), sendo agrupada sequencialmente em palavras, linhas e parágrafos. No outro sentido, a abordagem *top-down*, parte da divisão da página em parágrafos e consequentemente em linhas, palavras e caracteres [70]. O processo seguinte passa pela extração dos símbolos, efetuando um pré-processamento que visa a facilitar a identificação dos caracteres. A identificação de cada caracter é efetuada a partir comparação da extração de features obtidas na fase de treino. Assim que o caracter é identificado, é convertido para o código ASCII e é convertido num caracter "legível e editável". Por fim, a reconstrução das palavras é efetuada a partir da informação contextual [11].

Tem sido observado que diferentes sistemas de RCO cometem erros diferentes, ou porque o sistema não consegue detetar a presença de um caracter, ou porque consegue detetar a presença do caracter, mas não consegue identificar a que caracter corresponde, o que acaba por fazer com que diferentes sistemas tenham resultados muito divergentes entre eles. Isto deriva do facto dos diferentes sistemas usarem abordagens distintas para classificar caracteres [1]. Abdulkader e Casey (2009) propuseram um novo algoritmo que corrigia o resultado dos sistemas de RCO. Na abordagem deles, propuseram uma ferramenta de RCO principal (com melhores resultados) e diversas ferramentas secundárias. A premissa deles era que os erros no reconhecimento da RCO principal estavam altamente correlacionados com as divergências com as outras ferramentas. Assim sendo, a abordagem seguia a seguinte lógica: Eram introduzidos os documentos em todas as ferramentas (principal e secundárias), de seguida havia uma fase de estimativa de erro que consistia na comparação dos resultados (palavras que eram reconhecidas em praticamente todas as ferramentas eram consideradas corretas). As palavras cujos resultados diferiam entre si e cujo a estimativa de erro excedia uma *treshold* pré-definida eram consideradas suspeitas. A fase seguinte consistia no *clustering* de palavras suspeitas, e o *clustering* baseava-se no agrupamento de palavras com resultados idênticos na RCO principal, sendo que cada *cluster* é representado por um resultado principal. As

palavras suspeitas são então enviadas para um operador humano que garante a escolha de uma resposta duma lista possível de respostas das diferentes ferramentas ou escreve uma resposta nova. A opção tomada pelo operador é espalhada pelas restantes palavras do mesmo *cluster*. Por fim, baseado nas respostas dos operadores, os documentos estruturados de RCO são então gerados. A RNA responsável pela estimativa de erro é uma RNA simples com apenas um *hidden layer* composta por oito nós. O input dessa RNA são as seguintes *features*: os valores de confiança de todas as ferramentas de RCO, os tamanhos das *strings* de todos os resultados das ferramentas, as confianças mútuas dos resultados primários e secundários das ferramentas (caso a ferramenta permita calcular a confiança de palavras arbitrárias), *rankings* dos resultados primários e secundários das ferramentas (caso a ferramenta tenha esta componente), um valor binário que indica se o resultado é uma palavra válida do dicionário e por fim, uma pontuação do modelo de n-gramas calculado para cada um dos resultados. Os estudos que Abdulkader e Casey (2009) permitiram retirar a conclusão que o *clustering* de palavras suspeitas melhora significantemente a eficiência do sistema de correção. sendo que a redução de esforço humano chega atingir os 80% em alguns documentos. Em relação à ação do operador, os estudos concluem que o operador tende a precisar de 3 segundos para escolher a palavra correta, sendo que precisa de 6,5 segundos caso tenha a necessidade de escrever a palavra. A opção de escrever a palavra correta apenas é usada entre 8 a 33% do tempo.

Umas das maiores e mais conhecidas ferramentas de RCO é o *Tesseract*, que para além de efetuar o reconhecimento de caracteres, contém diversas funcionalidades, incluindo a identificação de mais de 100 linguagens humanas. Essa ferramenta é utilizada no projeto no qual eu me insiro. A ferramenta *Tesseract* é uma ferramenta que foi desenvolvida inicialmente pela HP (*Hewlett-Packard*) desde 1984 até 1994. Em 2005 foi disponibilizada como *open-source* e atualmente é suportada pela *Google*.

Patel (2012) decidiu testar a eficiência do *Tesseract* e as variações do desempenho entre imagens coloridas e imagens binárias. Para isso, usou um dataset de 20 exemplares de matrículas onde foram comparadas as imagens originais e as images convertidas para binário. Os resultados obtidos permitem concluir que o *Tesseract* tenha um desempenho de 61% para as imagens originais e um desempenho de 70% se estas fossem convertidas para binário e que a média do tempo de processamento necessário nas imagens originais fosse de 1 segundo, comparativamente com 0,82 segundos nas imagens binárias. De seguida, para o mesmo dataset, efetuou a comparação do *Tesseract* com o *Transym OCR*, uma ferramenta de RCO proprietária. Como o *Transym OCR* por si só, já faz a conversão para imagem binária, não houve a necessidade de fazer essa conversão manualmente. Sendo assim, os resultados provam que o *Transym OCR* tem um desempenho de 47% e uma média de tempo de processamento de 6,75 segundos.

As técnicas de RCO são usadas em praticamente todas as áreas, desde o setor educacional até ao setor governamental. Por exemplo, no setor bancário, o processamento de transações via cheques é efetuada maioritariamente usando abordagens de RCO. Nalguns casos, em que

os cheques são manuscriptos, pode requerer a confirmação manual de um funcionário. Esta tecnologia também é muito usada na indústria, onde a necessidade de economizar espaço e facilitar o processo de pesquisa criou a necessidade de ter os documentos digitalizados [48].

2.5 Validação Inteligente de Documentos

O crescimento exponencial de documentos guardados em sistemas digitais tornou crítico o desenvolvimento de sistemas de estruturação e categorização dos diferentes tipos de documentos. A maioria dos documentos comerciais e de identificação são documentos estruturados em formatos de formulários e tabelas. Este tipo de documentos é usado frequentemente, pois a sua representação é concisa e facilmente entendida. Foi necessário criar diversos processos que contribuíssem para a organização da informação deste tipo de ficheiros. Neste documentos, iremo-nos focar em 3 desses processos:

- **Classificação de documentos -** Este processo efetua a categorização dos documentos. A classificação pode ser binária (é uma fatura / não é uma fatura) ou pode considerar múltiplas classes (fatura / cartão de cidadão / certificado de nascimento / passaporte).

- **Extração de dados -** Este processo extrai a informação mais relevante dos documentos. Por exemplo, no caso de um passaporte, os dados mais relevantes devem ser o nome, a nacionalidade, a idade e o género.

- **Validação de documentos -** Este processo efetua a verificação do desempenho dos processos anteriores. Em alguns sistemas, este processo pode estar relacionado com a aprovação da autenticidade de um documento. Nesse caso, dependendo do documento submetido, o processo de validação busca anomalias no documento de forma a verificar se o documento é original.

Hoje em dia, em muitas empresas estes processos ainda são assegurados de forma manual por via de um operador humano. No entanto, são processos que exigem muito tempo dedicado, o que implicam um elevado custo na empresa, e os mesmos são suscetíveis a falhas humanas por erro ou mesmo fraude. As lacunas destes processos levaram à necessidade de se desenvolver processos automáticos. O recurso a técnicas de AC, de PLN e VC permitiu a criação de sistemas de validação de documentos inteligentes.

Apesar de haver uma grande evolução destas técnicas, nos dias de hoje, a categorização dos documentos ainda é um desafio, muito devido à diversidade dos documentos. Cada documento contém características específicas, o que dificulta a sua classificação (por exemplo, cada país tem o seu próprio cartão de cidadão, que embora seja o mesmo tipo de documento, difere de país para país). Ainda assim, estes sistemas estão cada vez mais a ser adoptados pelas empresas, não apenas para automatizar o processo de estruturar os documentos, tarefa essa que ainda tem algumas falhas pelo que ainda não é viável utilizá-la em alguns docu-

mentos mais sensíveis, mas sobretudo para ajudar e fornecer mais confiança ao processo de extração manual.

O processo de extração de informação de documentos pode ser dividido em duas abordagens principais: Sistemas Baseados em Regras (SBR) e Sistemas Baseados em Exemplos (SBE). Os SBR são sistemas que contém um conjunto de regras pré-definidas baseadas no conhecimento prévio humano. Este tipo de sistemas são fáceis de compreender e flexíveis. Por outro lado, estes sistemas contém uma natureza heurística, isto é, são sistemas que não são robustos em situações excepcionais, o que nesses casos, na generalidade, os resultados acabam por ser pouco satisfatórios. Os SBE são sistemas que fazem recurso de técnicas de AC para efetuar uma extração automática dos dados. Esta é uma abordagem mais recente comparativamente com a anterior e ainda se encontra em fase de maturação, no entanto é mais elegante, no sentido em que contrariamente ao sistema anterior, em que é necessário ter regras para cada tipo de documento — por vezes, as regras até divergem dentro do mesmo tipo de documento — por norma, é necessário apenas um algoritmo para diferentes tipos de documento, e nalguns casos já é possível obter melhores resultados comparativamente com os SBR. Estes sistemas, como acabam por lidar com muitos dados de treino, são sistemas robustos a pequenas variações e ruído e acabam por encontrar padrões complexos, que eventualmente não seriam descobertos por seres humanos. Para além disso, neste tipo de sistemas, como não é necessário a intervenção humana no processo de extração e categorização, acabam por aliviar o trabalho no design de padrões e regras, necessário nos SBR. Por outro lado, são sistemas que necessitam um enorme esforço na anotação dos dados de treino, isto é, na fase de treino dos algoritmos, é necessário indicar para cada documento quais são os campos a serem extraídos e onde estes estão localizados no documento. Para além disso, como são sistemas opacos, isto é, como não é fácil ter uma percepção visual de como eles funcionam, acaba por se tornar mais complexo melhorar os seus resultados [50].

Os SBE focam-se sobretudo em duas metodologias: de PLN e de VC. A tecnologia de PLN opera sobre matrizes unidimensionais (sequências de texto corrido), depois de ter sido efetuado o processo de extração de RCO, e consegue ter resultados bastante satisfatórios em documentos não estruturados[34, 71], como é o caso de artigos, livros ou jornais. No entanto, em documentos mais estruturados (formulários, tabelas), como é o caso de cartões de cidadão ou faturas, que não contém tanta informação textual e por outro lado, contém uma grande informação estrutural, a utilização exclusiva de técnicas de PLN tende a obter resultados pouco satisfatórios[33]. Por outro lado, em sentido inverso, as técnicas de VC têm a capacidade de explorar melhor a estrutura de documentos formatados, no entanto a sua abordagem, como é baseada em aspetos visuais ao invés de aspetos textuais, tende a ter resultados pouco satisfatórios quando é necessário entender a semântica do documento. Nos últimos anos têm surgido diversos métodos que aglomeram propriedades de ambas as metodologias [33, 7, 74]. A utilização de ambas metodologias permitiu obter melhores resultados na extração de informação dos documentos e tornar os sistemas mais robustos a diferentes estruturas de documentos.

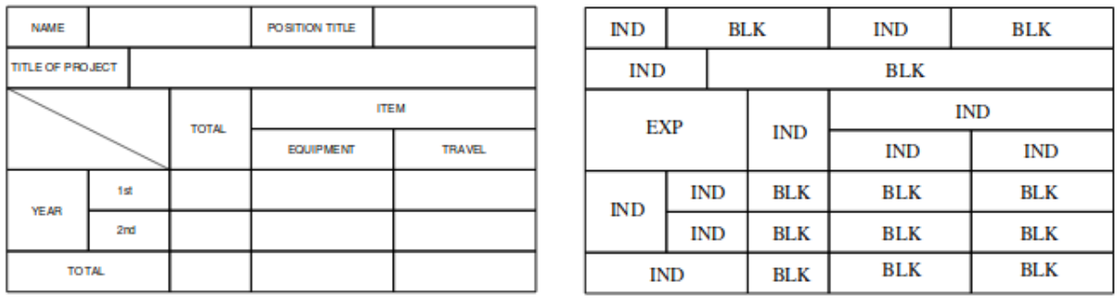
2.6 Revisão da Literatura

Existindo grande variedade de técnicas de extração de informação de documentos e múltiplas formas de as implementar, nos últimos anos têm sido desenvolvidos estudos diversificados de processos de extração de informação relevante de documentos que serão apresentados em seguida. Resumindo os artigos lidos, é possível dividi-los em 2 tipos de abordagem de extração de informação: os SBR e os SBE, ambos já explicitados anteriormente. Posto isto, as propostas de SBR são: a abordagem proposta por Amano e Asada [3] e a abordagem proposta por Esser et Al [12]. Já como SBE são propostos: o sistema de Yang [70], o CloudScan [52], o "Attend, Copy, Parse" [51], o Chargrid [33], o BERTgrid [7], e os sistemas propostos por Liu et al. [41] e Zhao et al [74]. Fazendo uma leitura temporal dos artigos, consegue-se evidenciar que a tendência inicial dos sistemas implementados eram basicamente SBR e que estes sistemas geralmente não tinham capacidade de generalizar para todo o tipo de layouts. Por outro lado, nos últimos anos têm surgido diversos SBE, que para além de terem a capacidade de generalizar para *layouts* nunca vistos pelos modelos, também tendem a apresentar melhores resultados, comparativamente com os sistemas anteriores. Dentro destes últimos sistemas, as RNA utilizadas por todas as abordagens são as RNC, redes neuronais geralmente usadas para compreensão de imagens e informação espacial, sendo que nalgumas também são usadas Rede Neuronal Recorrente (RNN), mais concretamente o Long Short Term Memory (LSTM), redes geralmente usadas para compreensão textual. Tendo presente a evolução sofrida ao longo do tempo, apresentam-se individualmente cada método enunciado anteriormente, iniciando com os algoritmos mais antigos, progredindo para a apresentação dos algoritmos mais recentes.

Em 2003, Amano e Asada [3] apresentaram uma análise de documentos estruturados em formulários e tabelas baseada em representações gramaticais. Nesta análise, o formulário é dividido em *boxes*, sendo que cada *box* é composta por um identificador único, um *label*, a sua posição e o seu tamanho. Um *label* pode ser decomposto em quatro tipos: branca (BLK), inserção (INS), indicação (IND) e explicação (EXP). As *labels* BLK e INS são células de entrada que são preenchidas pelo utilizador (BLK se a célula ainda não foi preenchida, caso contrário é INS). A *label* de indicação é a *label* que indica o que deve ser escrito nas *labels* de entrada, enquanto a *label* de explicação é a que dá uma explicação detalhada do que é pretendido. Podemos ver um exemplo desta estrutura nas Figuras 2.5a e 2.5b:

A *label* de indicação é uma componente forte na leitura deste tipo de estrutura, visto que as *labels* de entrada são determinadas pelas *labels* de indicação adjacentes (à esquerda ou em cima), e a sua relação é estabelecida se ambas tiverem o mesmo comprimento no lado em que estão ligadas. A relação entre a *label* de indicação e a respetiva *label* de entrada é considerada como uma *box* composta. Uma *box* composta é definida em quatro padrões: simples, múltipla, hierárquica e bidirecional. Um exemplo destes padrões pode ser vista na Figura 2.6:

A *box* composta simples é um padrão de combinação de um para um, em que a *label*



(a) Exemplo de um formulário [3]

(b) Resultado da classificação das *boxes* [3]

Figura 2.5: Exemplo de um formulário e respetiva classificação das *boxes*

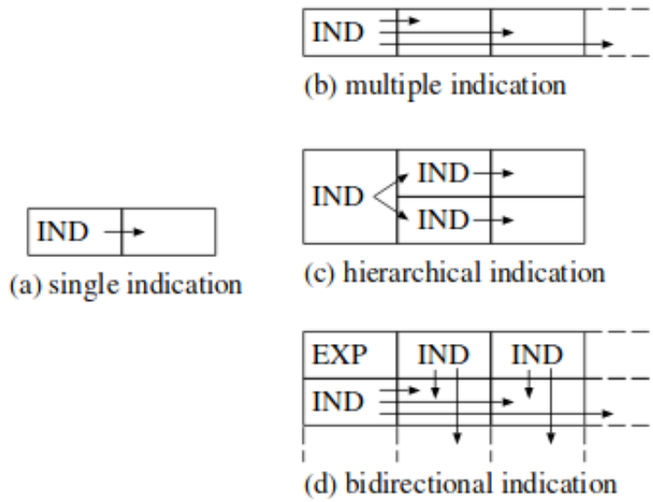


Figura 2.6: Padrões da Box Composta [3]

de indicação é associada ao valor correspondente. A *box* composta múltipla é um padrão de combinação em formato de lista em que múltiplos valores são associados a uma *label* de indicação. A *box* composta hierárquica é um padrão de combinação em formato de árvore e a *box* composta bidirecional é um padrão de combinação em formato de tabela em que cada valor de entrada é associado às *labels* de indicação horizontal e vertical correspondente.

Amano e Asada implementaram uma abordagem de representação de formulários em formato de grafo de forma a conseguir representar o conhecimento estrutural deste tipo de documentos. Para isso, foram utilizadas as *boxes* e respetivas adjacências. No grafo, cada *box* era representada como um nó e a ligação às *boxes* adjacentes eram representadas como arestas. Na Figura 2.7, podemos visualizar os diferentes tipos de relações entre duas *boxes* adjacentes.

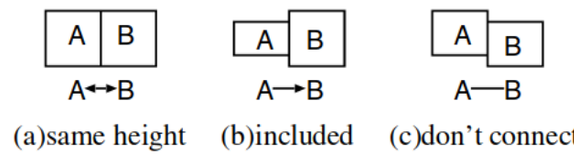


Figura 2.7: Relações entre boxes adjacentes [3]

Se duas *boxes* adjacentes tiverem o mesmo comprimento no lado em que são adjacentes,

são consideradas equalitárias ou que têm o mesmo peso. Se por outro lado, uma das *boxes* for mais pequena do que a outra, é considerado que a mais pequena é incluída na *box* maior. Caso haja um desfasamento entre as duas *boxes*, é considerado que não existe qualquer tipo de relação entre as *boxes*. Podemos observar a representação da tabela em formato de grafo na Figura 2.8.

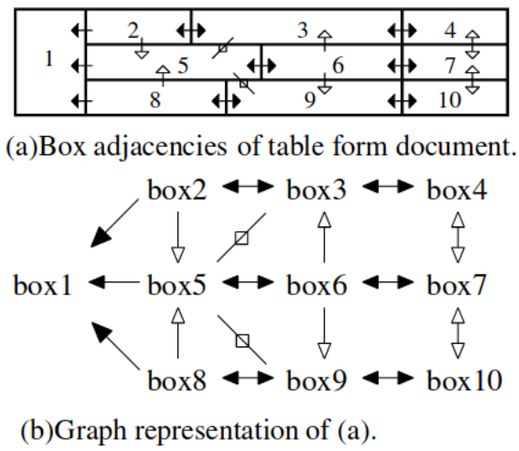


Figura 2.8: Representação da tabela em formato de grafo [3]

Em 2012, Esser et al. [12] implementou um SBR com uma abordagem *layout-based* que efetua a extração de campos de documentos de negócio. Apesar de ser um SBR, este método requere dados com as posições das palavras anotadas que serão utilizados na afinação dos parâmetros dos padrões. O fluxo de trabalho deste método pode ser visualizado na figura 2.9.

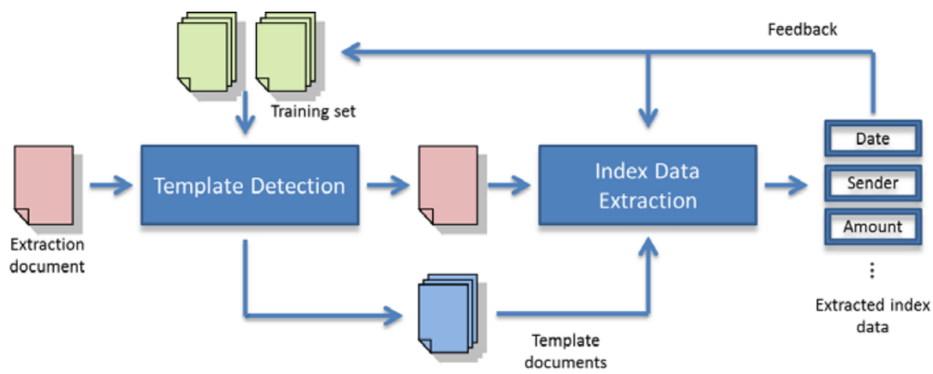


Figura 2.9: Fluxo de trabalho do método proposto por Esser et al. [12]

O fluxo de trabalho deste método é composto por 3 etapas: i) deteção de estruturas, onde são identificados documentos já anotados com estruturas idênticas ao documento submetido, ii) geração e cálculo de regras, tendo em conta os padrões dos documentos identificados na etapa anterior. Nesta etapa, é também efetuada a extração, sendo que são combinadas as posições dos dados de documentos anteriores com o documento submetido. É atribuída uma pontuação a cada palavra tendo em conta a distância e cobertura entre o padrão e a palavra. As palavras cujas pontuações estão acima de uma *threshold* são extraídas. iii) *feedback* do utilizador - os dados extraídos são apresentados ao utilizador, sendo que este tem a possibilidade de corrigir os dados e re-integrar via feedback. Este processo irá afetar a

extração de documentos futuros, sendo que o documento passará a fazer parte dos dados de treino. Este método foi avaliado em cerca de 3346 documentos de faturação, sendo compostos por 56 *templates* diferentes e 11 campos de extração. A métrica utilizada foi o F1-Score, atingindo uma média de resultados por volta dos 95%.

Em 2017, Yang et al. [70] apresentou uma abordagem para classificação semântica da estrutura do documento que exerce a segmentação do documento em regiões e, posteriormente, classifica esses segmentos (listas, tabelas, figuras, parágrafos, etc...). Embora não seja um método de extração, esta abordagem utiliza propriedades de SBE. Neste método os dados necessitam de estar anotados ao nível do pixel, sendo que cada pixel é anotado à classe correspondente. Como é difícil arranjar dados anotados a esse nível, os autores implementaram um processo que gera documentos sintéticos. Os documentos são submetidos a uma rede neuronal do tipo CNN, *Multimodal Fully Convolutional Network*. Este modelo é composto por: i) um codificador que aprende a hierarquia das representações de *features*, ii) um descodificador que retorna a máscara de segmentação e iii) um descodificador auxiliar que apenas está presente na fase de treino e efetua a reconstrução do documento. A visualização do modelo pode ser vista na figura 2.10.

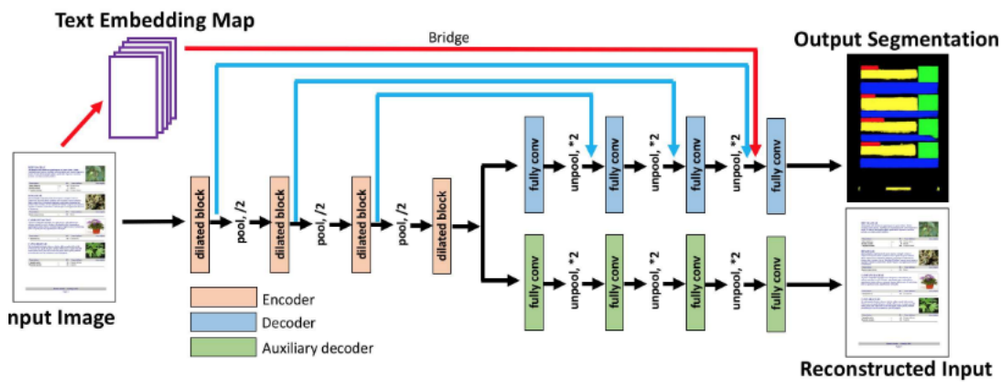


Figura 2.10: Fully convolution neural network [70]

Para além disso, foi implementado um mapa de *embedding* de texto, que fornece um contexto semântico das palavras ao método. O *embedding* é considerado ao nível da frase. Para cada píxel de uma frase, é usado o *embedding* da frase correspondente. Este mapa servirá também como entrada da rede neuronal, juntamente com a imagem do documento. O modelo foi treinado com recurso aos dados sintéticos anotados, sendo que posteriormente, efetuou-se um *fine-tuning* em dados reais não anotados. As métricas utilizadas na validação deste método foram as métricas *Intersection-Over-Union* e *F1-Score*. Este método foi validado com recurso aos conjuntos de dados ICDAR2015 (70 páginas) e SectLabel (347 páginas). No conjunto de dados ICDAR2015, foi utilizada a métrica de IoU e foram efetuados dois tipos de teste, sendo que no primeiro foi considerado duas classes: textual e não textual. Neste teste, o método proposto teve um resultado de 94.5% e 91%, respetivamente, comparativamente aos 84.7% e 86% de Bloomerg et al. [5] e aos 90.6% e 90.3% de Bukhari et al. [6]. O outro teste teve em conta as classes: figuras e testes. Neste, os resultados do método proposto

foram respetivamente 77-1% e 91%, comparativamente aos 70.1% e 85.8% de Fernandez et al. [13]. O teste efetuado no conjunto de SectLabel utilizou a métrica de F1-Score e teve em conta as classes: secção, descrição, lista e parágrafo. Neste teste os resultados foram, respetivamente, 91.9%, 89.3%, 79.3% e 96.9%, comparativamente aos resultados 91.6%, 78.1%, 71.2% e 96.9% de Luong et al. [42].

Em 2017, Palm et al. [52] apresentou o *CloudScan*, um sistema de produção que efetua a análise de faturas. Esta implementação não tem noções de *template*, ao invés disso, faz recurso de componentes textuais. No entanto, este sistema fornece a extração apenas a campos fixos pré-determinados, sendo que o modelo na fase atual apenas é capaz de suportar faturas. Este sistema recebe como input um documento em formato PDF, sendo que usa um serviço externo de RCO caso a informação textual não esteja disponibilizada. De seguida, há 2 abordagens que podem ser seguidas. Uma primeira onde são criados N-gramas de palavras na mesma linha, com um tamanho que varia de 1 a 4 palavras. Para estes N-gramas são calculadas diversas *features* e é classificado através de um modelo de regressão logística. A outra abordagem utiliza um classificador LSTM. Os autores defendem que neste classificador é preferível usar como entrada as palavras ao invés de N-gramas, pois permite ao classificador obter uma melhor leitura do documento. Na fase de pós-processamento, em cada campo, alguns n-gramas são filtrados por causa da sua sintaxe não corresponder corretamente ao pretendido. Na fase final do processo, é construída uma fatura em formato Universal Business Language (UBL). Esta implementação está preparada para estar numa constante aprendizagem, sendo que cada documento submetido é utilizado para otimizar os modelos. Na validação do método foram utilizadas as métricas de *F1 Score*, precisão e sensibilidade. Em primeiro lugar, foi efetuada uma *ceiling analysis* de forma a calcular o desempenho máximo teórico possível. De seguida, foram efetuados dois testes: o primeiro tendo em conta o desempenho geral e o segundo tendo em conta o desempenho para estruturas de documentos não vistas anteriormente. Nestes testes, foi testado o desempenho para 8 classes, correspondentes a campos de faturas. Estes testes servirão de comparação da rede neuronal LSTM e um modelo *baseline*, um classificador de regressão logística. No primeiro teste, a LSTM teve uma média de 89.1% comparativamente aos 88.7% do *baseline*. No segundo teste, a LSTM teve uma média de resultados de 84%, comparativamente aos 78.8% do *baseline*.

Mais tarde, em 2018, Palm et al. [51] apresentou um método de extração ponta-a-ponta, no sentido em que não é necessário lidar com dados de treino anotados, isto é, não é necessário indicar que campos devem ser extraídos e onde eles se situam. Em vez disso, assim que os dados são extraídos, o utilizador tem possibilidade de corrigir, sendo que esse *feedback* é utilizado para otimizar os modelos. Com este método, denominado por *"Attend, Copy, Parse"*, os autores esperam poupar o esforço humano extensivo na catalogação dos campos dos documentos de treino. A primeira etapa consiste na construção um banco de memória externo do mesmo tamanho do documento original, e que contém as palavras codificadas como

uma sequência de caracteres localizadas na posição de memória correspondente à posição das palavras no documento original. Nesta etapa, são criados N-gramas ao nível da palavra, sendo que posteriormente é efetuado um *1-hot encoding* ao nível do caracter. Este método é composto por 3 módulos principais. O módulo *Attend* é responsável por calcular a distribuição de *attention*, efetuar um *embbeding* das palavras e construir uma representação do documento que servirá como entrada para uma RNC, o modulo *Copy* é responsável por calcular a soma da distribuição de *attention* para cada N-grama presente no banco de memória e o módulo *Parse* é responsável por analisar se as palavras estão no formato standard (por exemplo, "23 de Julho" para 02/07, ou 23.0 para 23) e efetuar as devidas alterações. Este método foi treinado em cerca de 1.181.990 faturas de 43000 fornecedores, sendo que foram consideradas 7 classes presentes nos documentos. Em primeiro lugar, foi efetuada uma *ceiling analysis*, sendo que posteriormente, os testes foram comparados com o método *CloudScan*. A métrica utilizada foi o *F1-Score*, tendo este método atingido uma média de resultados de 83%, comparativamente aos 81% do CloudScan.

Em 2018, Katti et al. [33] introduziu uma nova abordagem para o processamento e compreensão de documentos em formato de imagem. Essa abordagem, que foi denominada por *Chargrid*, representa o documento como uma grelha bidimensional de caracteres (de forma a manter a estrutura espacial do documento) e consiste na construção de *bounding boxes* que rodeiam cada caracter, sendo atribuído posteriormente um valor constante $\mathrm{E}(\mathrm{c}_k)$ a cada área coberta pelo caracter, considerando c_k como o caracter selecionado. Na Figura 2.11, podemos observar o resultado esperado.

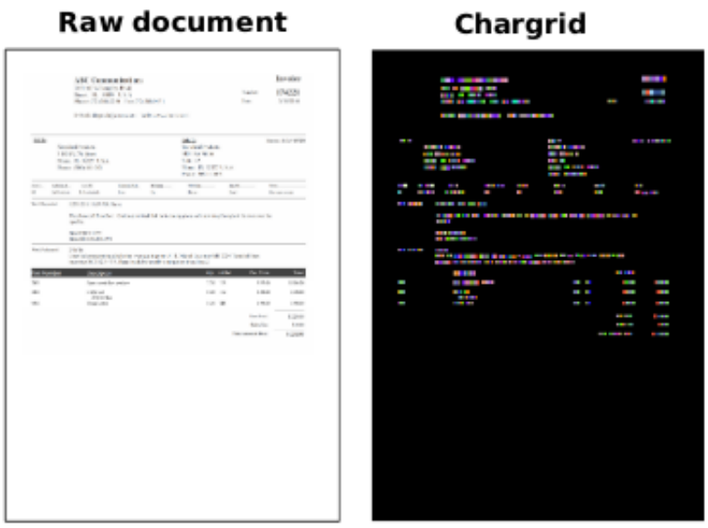


Figura 2.11: Representação *Chargrid* [33]

Segundo os autores, as vantagens desta representação são que cada caracter é codificado apenas com um único valor escalar, o que facilita a posterior análise do documento, e como todos os pixéis de cada caracter são mapeados com o mesmo valor, é possível reduzir o tamanho da representação (pode-se reduzir pelo factor do tamanho da área do menor caracter), o que reduz substancialmente no tempo computacional do processamento. Depois do

documento ser reduzido e ter sido efetuado um *1-hot encoding* da representação, esta servirá como entrada para uma rede neuronal convolucional. No pós-processamento, os caracteres são agrupados e categorizados. Apesar dos autores referirem que este método é genérica, a validação foi apenas efetuada em faturas. Foram consideradas 9 classes presentes nos documentos e a métrica utilizada pode ser visualizada na fórmula 2.5:

$$1 - \frac{\#[insercoes] + \#[eliminacoes] + \#[modificacoes]}{N}, \tag{2.5}$$

sendo que as inserções, eliminações e modificações são relativas aos dados previstas comparativamente com os dados verdadeiros e N é o número total de palavras nos dados verdadeiros. O valor da métrica pode ser negativo, sendo que esses casos são indicativos que seria preferível efetuar a extração manual dos dados. Na Figura 2.12, podemos visualizar os resultados obtidos:

Model/Field	Invoice Number	Invoice Amount	Invoice Date	Vendor Name	Vendor Address	Line-item Description	Line-item Quantity	Line-item Amount
sequential	80.98%	79.13%	83.98%	28.97%	16.94%	-0.01%	-0.18%	0.22%
image-only	47.79%	68.91%	45.67%	19.68%	13.99%	49.50%	46.79%	63.49%
chargrid-net	80.48%	80.74%	83.78%	36.00%	39.13%	52.80%	65.20%	65.57%
chargrid-hybrid-C32	74.85%	77.93%	80.40%	32.00%	31.48%	46.27%	64.04%	63.25%
chargrid-hybrid-C64	82.49%	80.14%	84.28%	34.27%	36.83%	48.81%	64.59%	64.53%

Figura 2.12: Comparação de resultados entre modelos sequenciais (PLN), modelos baseados em imagens (VC) e o *Chargrid* [33]

Na análise dos resultados, podemos observar que o *Chargrid* tende a igualar os melhores resultados de cada tipo de label, igualando o modelo sequencial (*CloudScan*) em dados de instâncias singulares (*label* - valor) e igualando o modelo baseado em imagens em dados tabulados (como é o caso dos detalhes de produtos).

Em 2019, Denk e Reisswig [7] aproveitaram as contribuições do *Chargrid* para implementar uma representação de grelha ao nível da palavra, o *BERTgrid*. Inicialmente, o documento em formato de imagem é submetido a um serviço de RCO que obtém as palavras e as respetivas posições. De seguida, os dados são serializados e posteriormente embutidos no modelo de linguagem BERT ([8]) que fornece um vetor contextual para cada palavra. O modelo usado para a extração de informação é o mesmo modelo *fully convolutional encoder-decoder* e as mesmas tarefas de aprendizagem de segmentação semântica e regressão de *bounding boxes* usadas por Katti et Al (2018) no *Chargrid*. As diferenças entre os métodos são que a representação é ao nível da palavra, ao invés de ser ao nível do caracter, e é efetuado um *embbeding* das palavras antes da representação do documento ser submetida na rede convolucional. As restantes etapas são idênticas às do método *Chargrid*. Este método também foi validado em faturas e as classes consideradas foram as mesmas que no método anterior, sendo que a combinação dos modelos do *Chargrid* e os modelos do *BERTgrid* permitiram melhorar os resultados em cerca de 4% (61.76% $\pm$0.72 para 65.48% $\pm$ 0.58).

Também em 2019, Liu et al. [41] implementou um sistema de extração de informação de documentos estruturados usando convolução de grafos. Este sistema combina a informação textual e visual dos documentos para efetuar uma melhor extração da informação. Neste método foram utilizados dois tipos de *embbeding*: o *embbeding* do grafo e do texto. No grafo, os nós contém os segmentos de textos, com o texto e respetiva posição, e as arestas contém as dependências visuais entre os segmentos de texto. O *embbeding* do texto é efetuado com recurso ao *Word2Vec*. Estes dois tipos de *embbeding* servirão como entrada para uma rede *Bi-LSTM*, passando posteriormente por um modelo *Fully Connected* e uma camada *CRF*. Este método foi validado nos conjuntos de dados VATI e IPR, com recurso à metrica *F1-Score*. No primeiro conjunto, composto por 3000 faturas de estrutura fixa e 16 classes, o método atingiu uma média de resultados de 87.3%. No segundo, composto por 1500 recibos e 4 classes, a média de resultados foi de 83.6%.

Zhao et al. [74] apresentou um novo método que representa o documento como uma grelha de palavras. A abordagem começa por efetuar um processo simples de *tokenize* das palavras. Posteriormente, esta abordagem divide o texto em palavras, que são posteriormente colocadas numa posição da grelha. A posição da grelha é calculada a partir da distância relativa entre palavras. Desta forma, é possível prevalecer a informação textual e espacial do documento original. Esta grelha de palavras servirá como entrada para uma RNC. Este método foi validado com o conjunto de dados ICDAR2019SROIE, composto por 1000 recibos e 4 classes etiquetadas. A métrica principal utilizada foi a *Average Precision*, sendo que este método foi comparado com os métodos *CloudScan* [52] e *BERT for NER* [8] (utilização do modelo BERT para a tarefa de reconhecimento de classes). O conjunto de dados foi dividido em 3 grupos de entidades distintas: recibos de taxi, refeições e hóteis. O método conseguiu uma média de resultados de, respetivamente, 94%, 81.5% e 74.6%, comparativamente aos resultados do *CloudScan* que foram 82%, 64% e 60% respetivamente e aos resultados do *BERT for NER* que foram respetivamente 88.1%, 80.1% e 71.7%.

Numa análise global da revisão da literatura, verifica-se que em todos os artigos, os algoritmos foram validados com recurso apenas a documentos do tipo fatura. Esta opção pode ser explicada: i) pela facilidade de arranjar conjunto de dados deste tipo de documentos comparativamente com outros tipos de documentos, como é o caso de cartões de cidadão ou passaportes (geralmente, só entidades governamentais é que conseguem colecionar este tipo de documentos de identificação) e ii) porque a grande maioria das entidades de negócio necessita sobretudo de organizar os seus documentos de faturação.

Como poderá ser visualizado na Tabela 2.1, os algoritmos podem ser organizados pelos seus objetivos e pela abordagem que utilizam

Relativamente aos objetivos, estes podem ser de 3 tipos: Representação/Leitura do documento, Segmentação do Documento e Extração dos campos de documentos. Como representação, temos o método proposto por Amano e Asada, na segmentação temos o método

Algoritmos	Objetivo	Abordagem
Amano and Asada [3]	Representação	SBR
Yang et al. [70]	Segmentação	SBE
Esser et al. [12]	Extração	SBR
Palm et al. [52]	Extração	SBE
Palm et al. [51]	Extração	SBE
Katti et al. [33]	Extração	SBE
Denk and Reisswig [7]	Extração	SBE
Liu et al. [41]	Extração	SBE
Zhao et al. [74]	Extração	SBE

Tabela 2.1: Objetivos e Abordagens dos algoritmos presentes no Estado da Arte

proposto por Yang et al., sendo os restantes métodos direcionados para a extração dos campos. Relativamente à abordagem que utilizaram, os métodos propostos por Amano e Asada e Esser et al. são SBR, sendo que os restantes métodos utilizaram propriedades dos métodos de SBE.

Considerando agora apenas os métodos de extração baseados em exemplos, na Tabela 2.2 podemos visualizar algumas características destes métodos.

	[52]	[51]	[33]	[7]	[41]	[74]
Etiquetagem dos dados			✓	✓	✓	✓
***Feedback* do Utilizador**	✓	✓				
VC		✓	✓	✓	✓	✓
PLN	✓	✓	✓	✓	✓	✓
Aprendizagem Supervisionada	✓	✓	✓	✓	✓	✓
Aprendizagem Não Supervisionada			✓	✓		
Extração de Campos Recursivos			✓	✓	?	?

Tabela 2.2: Características dos SBE de extração da Literatura

Relativamente à abordagem dos dados, os métodos propostos por Palm et al. não necessitam de etiquetar os dados, sendo que a aprendizagem é efetuada através do *feedback* do utilizador. Excepto o primeiro, todos os restantes métodos utilizam propriedades de VC, nomeadamente com a representação estrutural do documento e submissão a redes neuronais convolucionais. Para além disso, todos os métodos usam pelo menos uma propriedade de PLN. Os métodos propostos por Palm et al. utilizam a técnica de N-gramas, o *embbeding* das palavras também é uma técnica usada pelos métodos *"Attend, Copy, Parse"*, *BERTgrid* e Liu et al e CUTIE. A atribuição de um valor a cada caracter/palavra é uma técnica com propriedades textuais utilizada pelos métodos *Chargrid* e *BERTgrid*. Todos os métodos utilizam

aprendizagem supervisionada nos seus modelos, sendo que os métodos *Chargrid* e *BERTgrid* utilizam também uma aprendizagem não supervisionada na reconstituição do documento original. Relativamente à extração de campos recursivos, como é o caso de tabelas, apenas os métodos *Chargrid* e *BERTgrid* explicitam nos seus artigos que estes campos são suportados. Os métodos CUTIE e o proposto por Liu et al., não indicam nenhuma referência a este tipo de campos, o que não torna possível comprovar a sua eficiência. Por fim, embora praticamente todos os métodos (excepto o *CloudScan* se autoentitulem como métodos genéricos para diferentes documentos, a validação dos métodos apenas foi efetuada em conjuntos de dados de faturação.

A análise efetuada aos algoritmos do estado da arte permitiu identificar os pontos fortes e vulnerabilidades de cada algoritmo. O desenvolvimento conseguido nas diferentes soluções propostas ao longo do tempo, evidenciam uma preocupação em aperfeiçoar sistemas tecnológicos, que melhorem a capacidade de extração e análise de informação de documentos. É evidente o caminho já percorrido, sendo também notório que o desenvolvimento destes sistemas continua a ser um desafio na área da IA. No presente estágio, pretende-se dar algum contributo nesta vertente, efetuando uma análise comparativa dos algoritmos propostos. Da análise efetuada aos diferentes artigos de investigação dos algoritmos de extração e análise de informação de documentos, foram selecionados para serem implementados e testados, os algoritmos Cloudscan de Palm et al.[52], o CUTIE de Zhao et al.[74], o Chargrid de Katti et al.[33] e o BERTGrid de Denk e Reisswig[7]. Estes algoritmos serão descritos de forma mais detalhada na secção 3.4. Optou-se por não utilizar o algoritmo proposto por Attend, Copy, Parse de Palm et al.[51], pois a necessidade de ser implementado um modelo para cada campo a ser extraído, levantou dúvidas quanto à eficiência deste algoritmo e quanto à sua escalabilidade para documentos que contenham muitos campos a serem extraídos. Para além disso, também não será considerado o algoritmo proposto por Liu et al.[41], pois apesar de reconhecer que tem elevado potencial, a informação disponibilizada no artigo de investigação é escassa e falta alguma informação crucial, o que impossibilita a sua replicação.

De seguida, irei explicitar os testes que foram efetuados a serviços *Platform as a Service* (PaaS) idênticos ao serviço que irá ser implementado neste estágio.

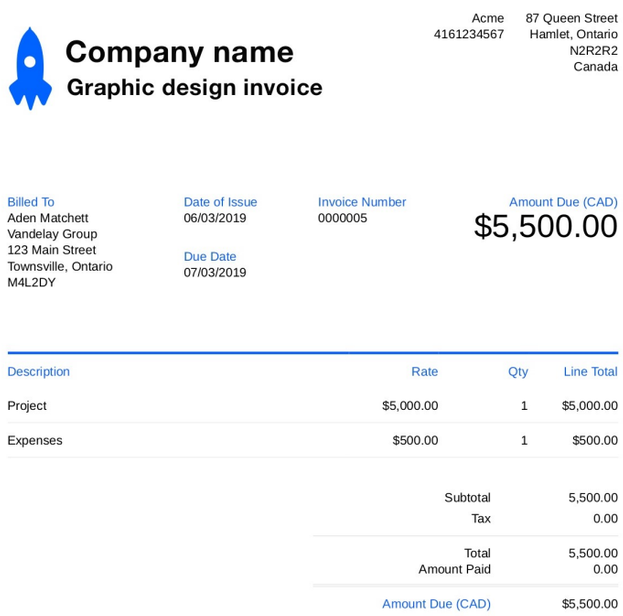
2.7 Análise de Serviços Idênticos

A classificação de documentos tem suscitado um grande interesse nas grandes empresas do setor tecnológico, tais como a *Microsoft*, *Amazon* e *Google*, que quiseram começar a explorar este mercado. Estes sistemas de validação permitiram que o processo fosse muito mais eficiente quer em termos temporais, quer em termos de desempenho. No entanto, embora tenha havido uma melhoria no processo, atualmente, as soluções existentes ainda têm algumas limitações, constituíndo simultaneamente um desafio e uma oportunidade para o seu

desenvolvimento. Para oferecer um melhor contexto a este problema, seria interessante testar as *Platform as a Service* (PaaS) de análise e extração de documentos disponibilizados em *Cloud* das grandes companhias tecnológicas. Um PaaS é um sistema sediado em ambiente *Cloud*, no qual é disponibilizado ao utilizador/cliente o *hardware* e *software* necessário para o serviço que se pretende. Os serviços que providenciam as funcionalidades descritas são, o serviço *Textract* da *Amazon Web Services* (AWS), o *Form Recognizer* da MA e o *Document Understanding* da *Google Vision*. O interesse de investigar estes serviços é, sobretudo, porque:

- São serviços idênticos ao que pretenderei implementar
- Investigação dos pontos fracos e fortes destes serviços
- Comparação de resultados com o nosso sistema
- Estão disponíveis para o utilizador comum (estes serviços têm uma *tier* gratuita limitada para o serviço deles serem testados)

Assim sendo, os serviços a serem analisados são o *Textract* da AWS e o *Form Recognizer* da MA. Na Figura 2.13 podemos visualizar um exemplo de uma fatura a ser submetida neste serviço.



Company name
Graphic design invoice

Acme
4161234567
87 Queen Street
Hamlet, Ontario
N2R2R2
Canada

Billed To
Aden Matchett
Vandelay Group
123 Main Street
Townsville, Ontario
M4L2DY

Date of Issue
06/03/2019

Due Date
07/03/2019

Invoice Number
0000005

Amount Due (CAD)
$5,500.00

Description	Rate	Qty	Line Total
Project	$5,000.00	1	$5,000.00
Expenses	$500.00	1	$500.00

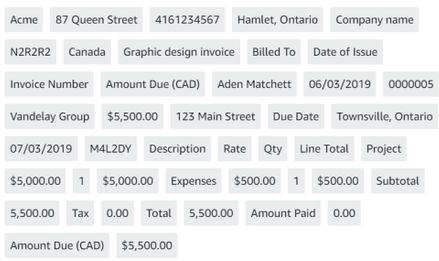
Subtotal	5,500.00
Tax	0.00
Total	5,500.00
Amount Paid	0.00
Amount Due (CAD)	$5,500.00

Figura 2.13: Exemplo de uma fatura falsa [28]

Começando inicialmente pelo serviço *Textract* da AWS, este serviço fornece-nos a extração de informação de qualquer tipo de documento impresso. Para cada documento submetido, o serviço oferece-nos 3 funcionalidades: a extração de *Raw Text*, de *Forms* ou de *Tables*.

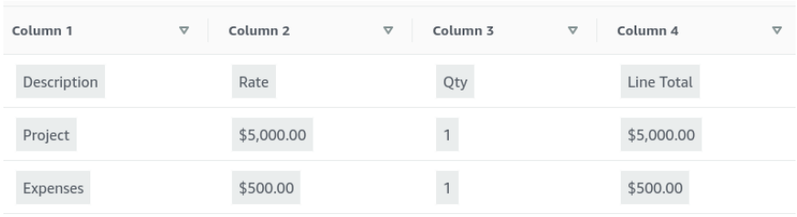
A extração de *Raw Text* é uma extração de RCO, onde apenas é visualizado o reconhecimento de cada palavra. Um exemplo do resultado desta funcionalidade pode ser visualizado na Figura 2.14a. Esta funcionalidade é interessante para analisarmos se o reconhecimento das *labels* e dos valores foi bem efetuado. Uma *label* é um campo descritivo associado a um

ou mais valores. Por exemplo, neste caso, uma *label* pode ser o *"Invoice Number"*, *"Total"* ou *"Description"*. Um valor é o campo com a informação que pretendemos extrair. Neste caso, um exemplo de valor poderia ser "0000005", "5,500.00" ou *"Project"*.



(a) Resultado da funcionalidade *Raw Text*

(b) Resultado da funcionalidade *Forms*



(c) Resultado da funcionalidade *Tables*

Figura 2.14: Resultados do Textract para o documento da Figura 2.13

A funcionalidade *Forms* associa a *label* ao valor correspondente. Esta funcionalidade apenas funciona para os casos em que uma *label* corresponde apenas a um valor (em tabelas, é necessário um pedido diferente). É possível visualizar um resultado desta funcionalidade na Figura 2.14b.

A funcionalidade *Tables*, como o nome indica, efetua a extração de tabelas. Como é óbvio, esta funcionalidade só funciona se o documento submetido contiver pelo menos uma tabela. Um exemplo de um resultado desta funcionalidade pode ser visualizada na Figura 2.14c.

Para além destas funcionalidades, o serviço retorna-nos a *bounding box* e pontuação de confiança de cada valor, de cada *form* e de cada tabela.

Por outro lado, o serviço tem algumas restrições entre as quais:

- Suportar apenas a língua inglesa
- Não suportar documentos manuscriptos
- Suportar apenas documentos com no máximo 10% de rotação

Infelizmente, na fase atual, o *Amazon Textract* ainda não providencia a previsão e compreensão de campos de documentos, isto é, é capaz de retornar os valores e *labels* e também consegue fazer a ligação da *label* ao valor respetivo, no entanto, o serviço não é capaz de indicar que o *label* e valor correspondem a uma determinada entidade, como Total ou Nº Fatura. Talvez fosse possível contornar este problema através do campo do *label*, ao fazer *parsing* deste campo e com isso tentar atribuir a uma entidade. No entanto, pela variedade

de palavras e nomes que uma *label* poderia assumir, necessitaria de muito processamento extra. Por esse mesmo motivo, de forma a ser justo na análise, não irei comparar este serviço com os algoritmos que serão implementados.

Também não será possível efetuar testes ao serviço *Document Understanding* da *Google Vision*, pois este ainda está numa versão *Beta* e é necessário uma subscrição especial para usufruir o serviço deles, o qual não me foi possível obter. Ainda assim, é interessante verificar o estado atual do serviço através da sua plataforma de demonstração.

Este serviço, está divido em 2 tipos de processamento de documentos, um geral e outro exclusivamente de faturas. Segundo a *Google*, o processamento geral aceita qualquer tipo de documentos até um máximo de 5 páginas ou 20 MB. Este é composto por três funcionalidades: a funcionalidade de texto *OCR*, de pares *label*/valor e de tabelas, ambos idênticos às funcionalidades respetivas do *Raw Text*, *Forms* e *Tables* do serviço *Amazon Textract*. O serviço de processamento de faturas, para além das funcionalidades anteriores, tem uma funcionalidade extra de previsão de entidades na fatura, no qual eles denominam por *Invoice Schema*. Esta funcionalidade, que pode ser vista na Figura 2.15, é composta por 12 entidades fixas que são elas: Nome e Morada do emissor, Nº e Data da Fatura, Nome e Morada do Recetor, Data de Emissão, Termos de Pagamento, Ordem de Compra, Imposto, Total e Por Receber. Para além disso, contém mais 4 entidades que podem aparecer múltiplas vezes, tendo em conta o número de itens comprados e são elas: a Quantidade, Preço Unitário, Descrição e Preço Total.

2.13.

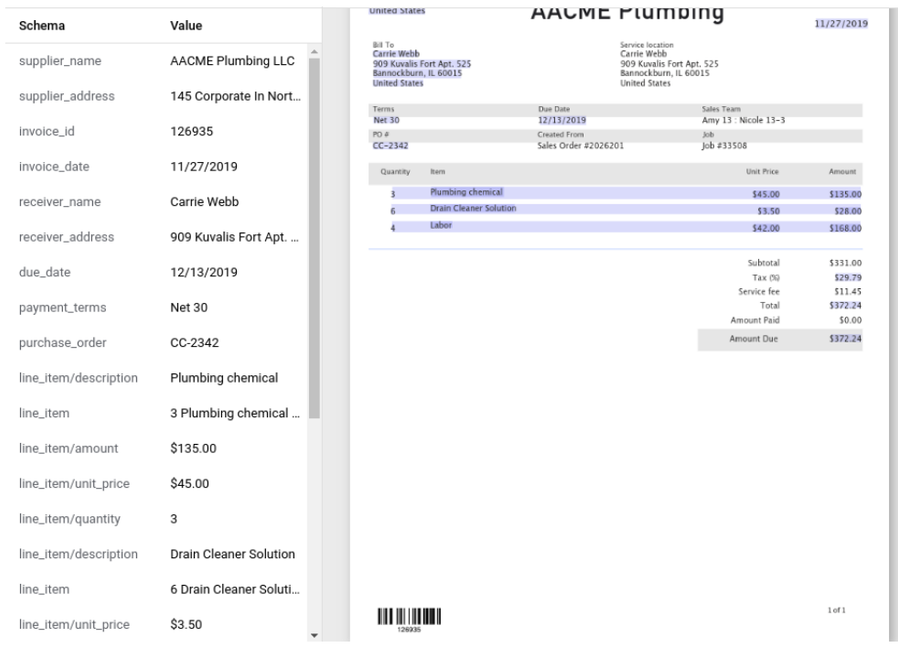


Figura 2.15: Funcionalidade do processamento de faturas da *Google Document Understanding*

Infelizmente, é pena não ser possível ainda testar este serviço, pois pelo pouco que é possível analisar do desempenho deste, parece ser bastante aceitável.

Analisando agora o serviço Form Recognizer do MA, tem que se realçar que o serviço Form Recognizer ainda está em versão Beta pelo que neste momento apenas contém modelos que estão otimizados para fornecer suporte a ficheiros do tipo recibo, podendo também funcionar para ficheiros de faturas. Para além disso, também é possível treinar um modelo customizável fazendo upload de 5 casos de treino idênticos. O funcionamento do Form Recognizer difere um pouco do serviço anterior (Textract), pois este não fornece a funcionalidade de fazer a ligação das *labels* aos valores, no entanto, consegue fazer a previsão de entidades pré-determinadas, o que vai de encontro ao que pretendemos também fazer nos algoritmos a implementar. As entidades pré-definidas pelo serviço são: "Sub-total", "Taxa/Imposto", "Total", "Nome do Vendedor", "Morada do Vendedor", "Número de telefone do vendedor", "Data da Fatura" e "Hora da Fatura". Na Figura 2.16, é possível visualizar o resultado do serviço Form Recognizer para a fatura da Figura 2.13.


```
"understandingResults": [{
  "pages": [1],
  "fields": {
    "Subtotal": {
      "valueType": "numberValue",
      "value": 500.0,
      "text": "5,500.00",
      "elements": [{
        "$ref": "#/recognitionResults/0/lines/35/words/0"
      }]
    },
    "Total": {
      "valueType": "numberValue",
      "value": 500.0,
      "text": "5,500.00",
      "elements": [{
        "$ref": "#/recognitionResults/0/lines/39/words/0"
      }]
    },
    "Tax": {
      "valueType": "numberValue",
      "value": 0.0,
      "text": "0.00",
      "elements": [{
        "$ref": "#/recognitionResults/0/lines/37/words/0"
      }]
    },
    "MerchantAddress": null,
    "MerchantName": {
      "valueType": "stringValue",
      "value": "Hamlet, Ontario",
      "text": "Hamlet, Ontario",
      "elements": [{
        "$ref": "#/recognitionResults/0/lines/3/words/0"
      }, {
        "$ref": "#/recognitionResults/0/lines/3/words/1"
      }]
    },
    "MerchantPhoneNumber": null,
    "TransactionDate": {
      "valueType": "stringValue",
      "value": "2019-06-03",
      "text": "06/03/2019",
      "elements": [{
        "$ref": "#/recognitionResults/0/lines/13/words/0"
      }]
    },
    "TransactionTime": null
  }
```


Figura 2.16: Resultados do Form Recognizer da MA para o documento da Figura 2.13

Como pode ser visto na Figura, para cada uma das entidades pré-definidas, o resultado

retorna o campo do tipo de valor, que pode ser classificado como valor numérico ou textual, o campo do valor onde retorna o valor da entidade já pré-formatado, o campo texto que retorna o valor da entidade no formato original do documento. Para além disso, retorna um campo de elementos, onde faz referência à posição RCO das palavras que percentem a essa entidade.

Ao contrário dos serviços anteriores, é possível e faz sentido testar este serviço. Apesar de este serviço estar otimizado para recibos, também é capaz de desempenhar a previsão em faturas, pelo que é interessante comparar o desempenho deste serviço com os algoritmos que irão ser implementados. Os resultados destes testes podem ser vistos na Secção 4.2.

Capítulo 3

Especificação do Sistema

A CSW está neste momento a desenvolver uma plataforma — Intelligent Document Validation (IDV) — em que se pretende fornecer serviços relacionados com a análise de documentos. A plataforma do IDV providencia uma *API REST* que utiza uma *framework Flask Python*. Com esta Interface de Programação de Aplicações (IPA), os serviços podem ser integrados em múltiplos sistemas, quer *backend* ou *frontend*. O objetivo desta plataforma é facilitar o processo de organização e de gestão de documentos, cujo processos ainda hoje são feitos maioritariamente por humanos. Para atingir esse objetivo e forma a que esta plataforma consiga ser versátil, é necessário torná-la o mais genérica possível, independentemente do formato ou tipo de documento. Com isto pretendemos que seja possível usar os serviços disponíveis, independentemente do documento ser em manuscrito ou impresso, se está digitalizado ou em formato "editável", ou se está em formato de formulário ou de texto.

Na figura 3.1 é possível visualizar os serviços disponibilizados por esta plataforma.

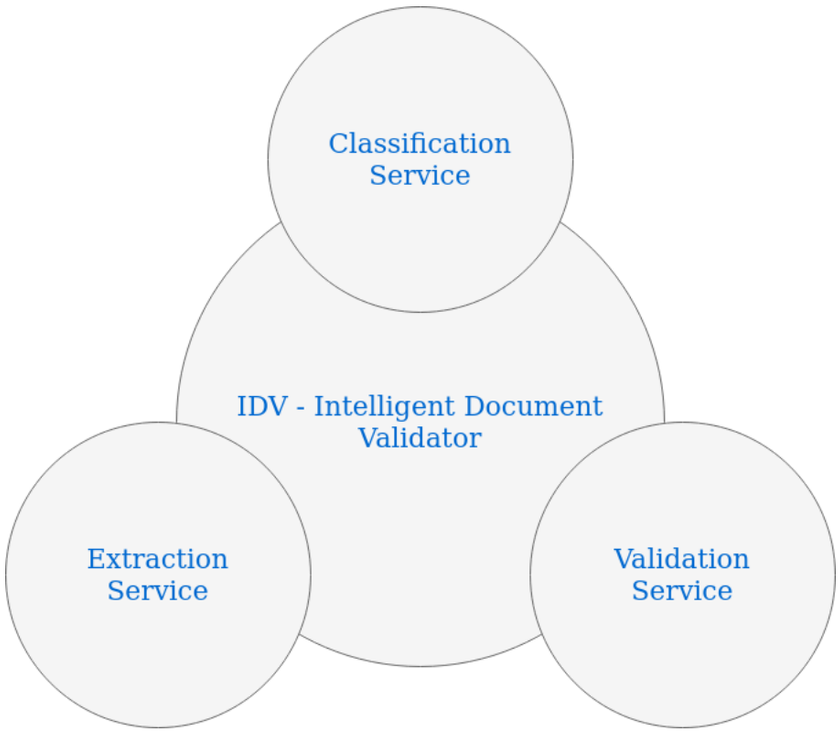


Figura 3.1: Serviços da plataforma IDV

De seguida irei explicar, de forma breve, em que consiste cada um dos serviços:

- **Classificação -** Efetua a categorização do tipo de documento submetido. Para além disso, indica a confiança do modelo na sua previsão. Nesta etapa é possível verificar se o modelo é manuscrito ou impresso ou se está em formato de formulário ou texto. A classificação pode ser binária, isto é, verificar se é um tipo de documento ou não (fatura ou não-fatura), ou multi-classe, isto é, dadas as categorias de documentos (por exemplo, fatura, cartão de cidadão, licença de condução) é escolhida a categoria com melhor resultado da previsão. Na fase atual, o serviço de classificação contém suporte interno para classificadores de imagem baseados em *Keras* e *Tensorflow*, um classificador de texto baseado na biblioteca *spaCy*, classificadores baseados em regras e classificadores compostos. Para além disso, este serviço pode ser ajustado de forma a suportar outros tipos de classificadores.

- **Extração -** Efetua a extração dos campos relevantes para cada tipo de documento. Este processo, na fase atual, é suportado por diferentes técnicas que são usadas tendo em conta o contexto. Essas técnicas são sobretudo: i) a definição de padrões e regras (como nos SBR) — isto é, para cada tipo de documento existe um conjunto de regras, que indicam a localização e efetuam a extração dos valores relevantes, sendo que essas regras são definidas com base em conhecimento prévio, ii) técnicas de PLN e RCO — estas técnicas são sobretudo utilizadas em documentos com muita informação textual (documentos de texto corrido e manuscriptos). Além disso, este serviço fornece diversos recursos de processamento de imagens e processamento de linguagem natural utilizados no processo de extração.

- **Validação -** Efetua a validação dos serviços de classificação e extração. Neste processo, os serviços anteriores são analisados relativamente ao seu desempenho. Neste momento, o serviço de validação implementado na plataforma é apenas uma processo de validação por igualdade, isto é, no serviço de classificação é verificado se a categoria prevista é correta, enquanto no serviço de extração, é verificado se todos os campos relevantes do documento foram extraídos com sucesso. Ainda assim, a arquitetura da plataforma está desenhada para receber outros tipos de validação.

Dentro dos serviços disponíveis na plataforma, o estágio curricular no qual me encontro a realizar foca-se **unicamente** no serviço de extração. Como foi enunciado anteriormente, o objetivo principal deste estágio é desenvolver uma prova de conceito de um dos algoritmos de extração e análise da informação de documentos estruturados. O serviço atual de extração já implementado na plataforma do IDV é, sobretudo, baseado em regras (também contém técnicas de RCO e PLN), sendo o objetivo deste trabalho implementar uma prova de conceito de um ou vários algoritmos baseados em exemplos, efetuando previamente a implementação e análise de diversos algoritmos. Para alcançar este objetivo é necessário:

- Verificar se os algoritmos baseados em exemplos conseguem ter desempenhos aceitáveis em ambiente de negócio.
- O desempenho do algoritmo escolhido para a prova de conceito consegue ser superior ou pelo menos não muito inferior ao serviço já implementado — se o desempenho não for muito inferior ao serviço já implementado, pode ser interessante para a empresa considerar este algoritmo, visto que traz inúmeros benefícios, sobretudo na adaptação para novos documentos, que com o serviço atual baseado em regras necessita de muito esforço humano na implementação.
- É possível tornar o serviço mais genérico, isto é, se o algoritmo é versátil e é capaz de ter desempenhos aceitáveis com diferentes documentos.

Embora os objetivos em cima estejam delineados para analisar o sucesso do algoritmo da prova de conceito, a não concretização destes objetivos, não implica que o estágio seja mal sucedido. Pode acontecer o caso dos algoritmos não obterem um bom desempenho e isso não se dever à má implementação destes, pelo que o sucesso do estágio não se deve analisar exclusivamente pelos resultados obtidos. O estágio no qual estou a frequentar é um estágio que contém três componentes, todos de grande importância, que são a exploração de algoritmos do estado da arte através da análise dos seus artigos de investigação, implementação dos algoritmos e investigação e análise do desempenho dos algoritmos. Não sendo um estágio puramente de Engenharia de *Software*, nem de exploração e investigação, as condições de sucesso devem incidir sobre as três componentes principais do estágio, pelo que o estágio deverá ser bem sucedido se for capaz de:

- Explorar, identificar e analisar corretamente algoritmos que sejam considerados o estado da arte da extração e análise da informação de documentos.
- Implementar algoritmos do estado da arte a partir de artigos de investigação, mesmo que estes apenas contenham informação parcial ou dúbia.
- Efetuar uma análise correta do desempenho, sendo capaz de identificar pontos fortes e fragilidades nos algorimos analisados

Relativamente aos requisitos do sistema, não sendo este estágio um projeto de engenharia de *software*, não faz muito sentido definir previamente os requisitos do sistema, pois há algumas variáveis que apenas irão ficar bem definidas no decorrer da análise dos algoritmos. Ainda assim, seria interessante analisar potenciais riscos que poderão acontecer no decorrer do estágio.

3.1 Análise de Riscos

A identificação de potenciais riscos que poderão, eventualmente, acontecer no decorrer do projeto é importante para o sucesso do mesmo. Ter uma estratégia bem definida, para

quando um dos riscos acontecer, saber que plano de mitigação acionar, é uma boa forma de reduzir o impacto que estes podem ter no sucesso do projeto. Tendo em atenção o contexto do projeto, há alguns riscos que poderão eventualmente ocorrer, nomeadamente:

- ***Dataset* Disponibilizado -** O *dataset* disponibilizado para a validação dos modelos a serem implementados pode ser de fraca qualidade ou ter uma quantidade insuficiente para obter bons resultados nos modelos. A estratégia de mitigação do risco para este cenário irá depender da origem do mesmo. Se porventura o *dataset* for de fraca qualidade, então a estratégia passará por fazer uma seleção mais precisa dos documentos a comporem o conjunto de dados a partir da identificação dos documentos de menor qualidade e inserir novos dados no dataset proposto (em caso de necessidade e, relativamente aos documentos das faturas, a CSW poderá disponbilizar mais dados caso seja necessário). Se por outro lado, não houver um conjunto de dados suficiente para obter uma boa validação, então a estratégia passará por incluir mais dados no *dataset*, ou caso isto seja impossível, recorrer a técnicas de *Data Augmentation* (técnicas que permitem criar novos dados no dataset a partir de modificações do dataset atual, como por exemplo, efetuar rotações dos documentos, ou efetuar cópias com condições de luz diferentes, etc...). Este risco tem uma probabilidade média de acontecer e, caso aconteça, o impacto que poderá ter é grande.

- **Pouca familiarização com as tecnologias a serem usadas -** A pouca experiência prática em técnicas de *Deep Learning*, VC ou PLN, pode levar a atrasos no desenvolvimento do projeto. Neste caso, a estratégia de mitigação será ler a máxima documentação possível sobre o assunto, realizar alguns tutoriais nesta área e consultar alguns exemplos *online* e os colegas de equipa com mais experiência nesta área. A probabilidade deste risco ocorrer é grande, sendo que também pode ter um grande impacto para o sucesso do projeto.

- **Informação incompleta por parte dos algoritmos a explorar -** A informação providenciada pelos autores na publicação dos artigos de investigação dos algoritmos que consistem no estado da arte, pode não conter informação suficiente para que a replicação dos modelos seja possível. A estratégia de mitigação do risco dependerá do nível de falta de informação. Se a informação em falta não for crucial na implementação dos algoritmos e poder ser colamtada com experimentação e testes de optimização por parte do aluno, então será essa a opção a tomar por parte do aluno. Caso contrário, se a informação em falta tiver um impacto crucial na implementação do algoritmo, então a estratégia de mitigação consistirá na escolha de outro algoritmo de extração e análise de documentos. Este risco tem uma grande probabilidade de acontecer e, caso acontença, o impacto que poderá ter é grande.

3.2 Metodologias e Ferramentas

Esta secção irá descrever as metodologias e ferramentas utilizadas no decorrer do estágio curricular.

A metodologia a ser seguida por este projeto é baseada no SCRUM. O SCRUM é uma metodologia de desenvolvimento que está a ser adoptada pela CSW e que é utilizada pela equipa de desenvolvimento do projeto no qual estou inserido.

O SCRUM é uma metodologia iterativa e incremental de desenvolvimento de projetos cujo o objetivo é agilizar o processo de gestão e ser responsivo a alterações e imprevistos que possam ocorrer durante o projeto. Esta metodologia parte do princípio que a análise, design e processos de desenvolvimento são imprevisíveis no início dos projetos. Em muitos projetos, o problema não está muito bem definido na fase inicial, ou porque o cliente não tem a certeza daquilo que pretende, ou porque podem ocorrer imprevistos durante o desenvolvimento e o SCRUM permite reavaliar e fazer os ajustes necessários no decorrer do projeto. [61].

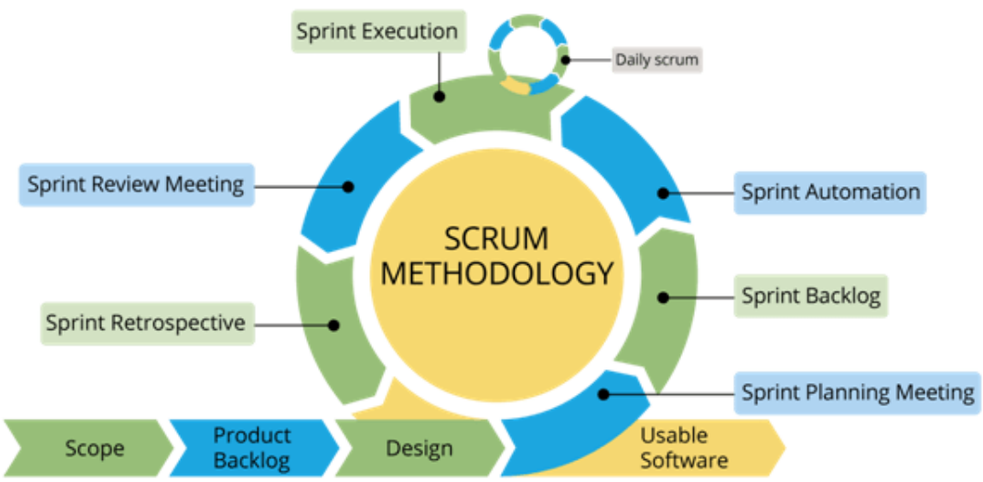


Figura 3.2: Modelo geral do workflow do SCRUM [62]

O Scrum está divido em 3 fases: a fase de planeamento e arquitetura, a fase de desenvolvimento e a fase de encerramento. A fase de planeamento e arquitetura consiste no desenvolvimento do *Product Backlog* — a lista de tarefas a serem desenvolvidas durante o projeto — na identificação dos riscos e respetivos métodos de mitigação e no design da arquitetura do projeto. A fase de desenvolvimento consiste num ciclo iterativo de desenvolvimento, denominado por *Sprint*. No início de cada *Sprint*, há uma reunião de planeamento onde são discutidas as tarefas concluídas e as dificuldades encontradas na *Sprint* anterior e são definidas as tarefas que irão ser incluídas na nova *Sprint*, também conhecido como *Sprint Backlog*. No nosso projeto, cada *Sprint* tem uma duração de duas semanas e cada reunião dura em média 30 minutos. Para além disso, no nosso projeto em cada semana existe uma reunião onde é discutido o progresso efetuado até ao momento. Na fase de encerramento é efetuada a preparação para o lançamento do produto ou serviço, o que inclui toda a documentação necessária e testes efetuados.

No SCRUM existem apenas três cargos distintos: o *Product Owner*, o *Scrum Master* e a

equipa de desenvolvimento.

- ***Product Owner* -** Pessoa responsável por efetuar a ligação entre a equipa de desenvolvimento e o cliente. Para além disso, é a pessoa responsável pelo *Product Backlog* e é ele que define a data de lançamento e conteúdo do produto. No nosso projeto, este papel é efetuado pelo Doutor Tiago Baptista.

- ***Scrum Master* -** Pessoa responsável por manter a equipa focada no desenvolvimento do projeto. Este protege a equipa de distrações internas e externas e resolve impedimentos que possam surgir e impossibilitem ou dificultem o fluxo normal de desenvolvimento. No nosso projeto, este papel é efetuado pelo Doutor Tiago Baptista.

- **Equipa de Desenvolvimento -** Equipa responsável pelo desenvolvimento do projeto. No nosso projeto, a equipa é composta pelo lider técnico Doutor Rui Lopes e pelos Engenheiros Ana Guarino, Diana Mendes, João Castilho e Rodrigo Pinto.

As ferramentas que são utilizadas como auxiliares na gestão de projeto são:

- ***Jira*[1] -** *Software* de gestão de projectos ágeis que permite fazer o acompanhamento de tarefas de desenvolvimento. Esta ferramente é utilizada pela maioria das equipas da CSW para gestão de projetos.

- ***BitBucket*[2] -** Ferramenta de controlo de versões de repositórios Git. Esta ferramente é utilizada pela maioria das equipas da CSW para gestão de projetos.

- ***Confluence*[3] -** Ferramenta de gestão de documentação de projetos. Esta ferramente é utilizada pela maioria das equipas da CSW para gestão de projetos.

- ***Docker*[4] -** Ferramenta que permite correr aplicações em *containers*. Esta ferramenta contém um ambiente virtual de desenvolvimento que utiliza isolamento de recursos, o que permite aos desenvolvedores desenvolverem e correrem as suas aplicações mais facilmente.

- ***Teams*[5] -** Ferramenta utilizada pelas equipas da empresa *CSW* para comunicação interna. Devido ao regime de teletrabalho imposto pelo Governo no decorrer do estágio, esta ferramenta teve uma preponderância enorme no contacto com o meu tutor e com a restante equipa.

Relativamente às ferramentas técnicas a serem utilizadas no projeto são:

- ***Python*[6] -** Linguagem de programação de alto nível e interpretada. Esta linguagem é muito utilizada em projetos de Inteligência Artificial devido às bibliotecas de *Data*

[1] https://www.atlassian.com/software/jira
[2] https://bitbucket.org/
[3] https://www.atlassian.com/software/confluence
[4] https://www.docker.com/
[5] https://www.microsoft.com/pt-pt/microsoft-365/microsoft-teams/group-chat-software
[6] https://www.python.org/

Science integradas. Esta linguagem é utilizada pela equipa de IA da CSW, mais precisamente, na plataforma onde o meu estágio se insere.

- ***TensorFlow & Keras***[7] **-** O *Tensorflow* e o *Keras* são bibliotecas *open-source* de *Python*, conhecidas pela sua vasta biblioteca de técnicas de AC, de fácil integração e modificação. Estas bibliotecas são as mais utilizadas no mundo da IA na linguagem *Python* e são utilizadas pela equipa de inteligência artificial da CSW, mais precisamente, no plataforma onde se insere o meu estágio

- ***OpenCV-Python***[8] **-** O *OpenCV*, ou *Open Source Computer Vision Library*, é uma biblioteca *open-source* de métodos de VC e AC. Esta biblioteca é composta por mais de 2500 algoritmos optimizados e é muita utilizada em projetos que lidam com processamento de imagens e vídeos. A equipa de IA da CSW utiliza esta biblioteca nos seus projetos, mais precisamente, na plataforma onde o meu estágio se insere.

- ***Tesseract OCR***[9] **e** ***Google OCR***[10] **-** A ferramenta *Tesseract* é uma ferramenta *open-source* que efetua o reconhecimento de caracteres ópticos. Já a ferramenta *Google OCR* é uma ferramenta proprietária também ela de reconhecimento de caracteres ópticos. Ambas são atualmente suportadas pela *Google*. Estas ferramentas são utilizadas pela equipa de IA da CSW nos seus projetos, mais precisamente, na plataforma onde o meu estágio se insere.

- ***Dataturks***[11] **-** O *Dataturks* é uma ferramenta que permite fazer anotações de classes em imagens a partir da criação de *Bounding Boxes*. A empresa da CSW detém uma licença de utilização deste *Software* proprietário, sendo esse o principal motivo da escolha desta ferramenta.

3.3 Planeamento de Estágio

Esta secção irá descrever o planeamento do trabalho efetuado durante o 1º semestre e do que pretendo realizar no decorrer do 2º semestre.

Relativamente ao que foi efetuado durante o **1º Semestre**. Na primeira *Sprint* tive a tarefa de definir o foco e objetivos do meu estágio. Na segunda e terceira *Sprint* estive a estudar a plataforma atual do IDV. Durante o estudo, foram-me dadas duas tarefas, sendo que a primeira foi testar se os ficheiros com o sistema de cores CMYK estaria a funcionar corretamente na nossa plataforma e a segunda tarefa foi implementar um método de extração de RCO com recurso ao serviço *Batch Read File* da *Microsoft Azure*. Esta segunda tarefa foi aproveitada pela equipa de desenvolvimento do IDV para ser integrada na plataforma. Da 4ª *Sprint* à 6ª *Sprint*, dediquei-me ao estudo do conhecimento teórico necessário e da

[7] `https://www.tensorflow.org/`
[8] `https://opencv.org/`
[9] `https://github.com/tesseract-ocr/`
[10] `https://cloud.google.com/vision/docs/ocr`
[11] `dataturks.com`

análise do estado da arte das áreas que irei necessitar para a realização das minhas futuras tarefas. Ainda no seguimento deste tarefa, foram efetuados testes a serviços idênticos ao que pretenderei explorar que servirão como base de comparação com os resultados de validação do(s) modelo(s) que irei implementar. Da 4ª *Sprint* à 9ª *Sprint* estive a escrever o meu relatório de estágio. Por fim, antes da entrega intermédia, da *Sprint* 7 à *Sprint* 9, foi criada a especificação técnica do meu estágio. Na semana a seguir à entrega intermédia, correspondente à 10ª *Sprint*, irei preparar a minha apresentação para a defesa intermédia. Na figura 3.3, podemos observar o diagrama de *Gantt* relativamente ao planeamento do 1º semestre.

Já relativamente ao planeamento do trabalho para o **2º semestre**, no que resta da 10ª *Sprint* irei configurar o ambiente de pesquisa e desenvolvimento. Na 11ª e 12ª *Sprint*, irei explorar o primeiro modelo e validar os resultados. Na 13ª e 14ª *Sprint*, explorar-se-á o segundo modelo e fazer a validação dos respetivos resultados. Na 15ª e 16ª *Sprint*, de forma análoga, irei explorar o terceiro modelo e validar os seus resultados. Na 17ª e 18ª *Sprint*, farei a análise dos modelos e implementar a prova de conceito do modelo escolhido. Na 19ª *Sprint* irei dedicar-me aos testes e *benchmarking* do modelo. Simultaneamente, da 19ª à 21ª *Sprint*, irei finalizar a redação do relatório final de estágio. Na semana que se segue à entrega final, prepararei a apresentação para a defesa final. Na figura 3.4, é possível observar o diagrama de *Gantt* relativo ao plano para o 2º semestre.

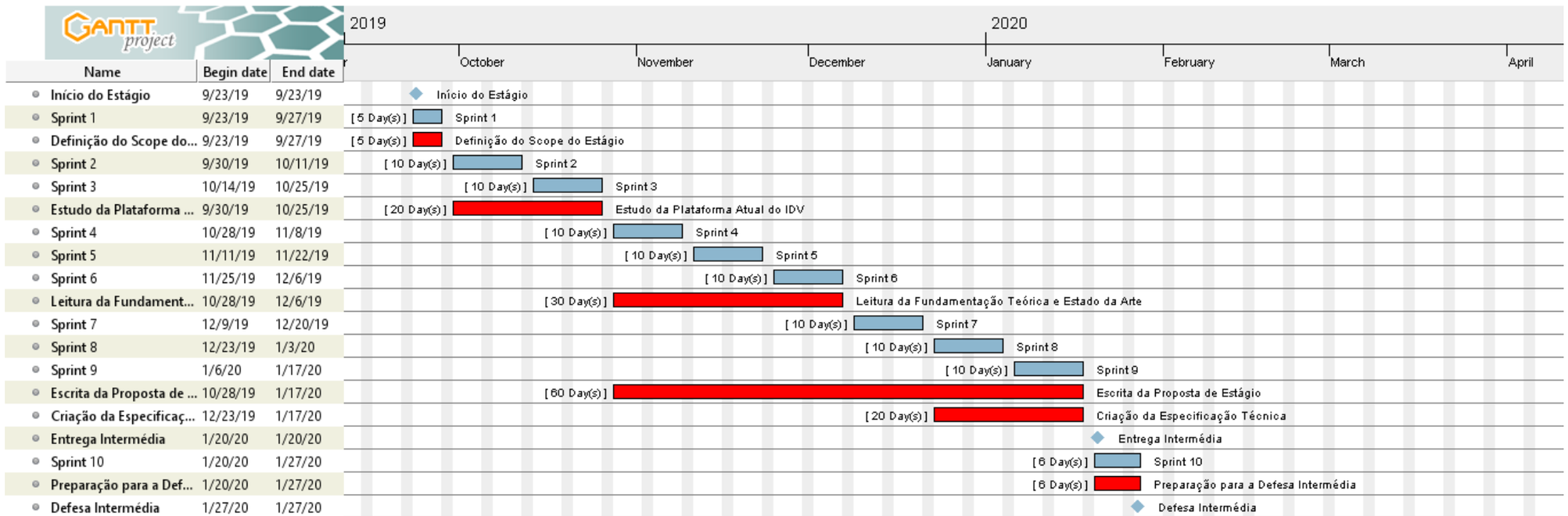


Figura 3.3: Planeamento do trabalho do primeiro semestre

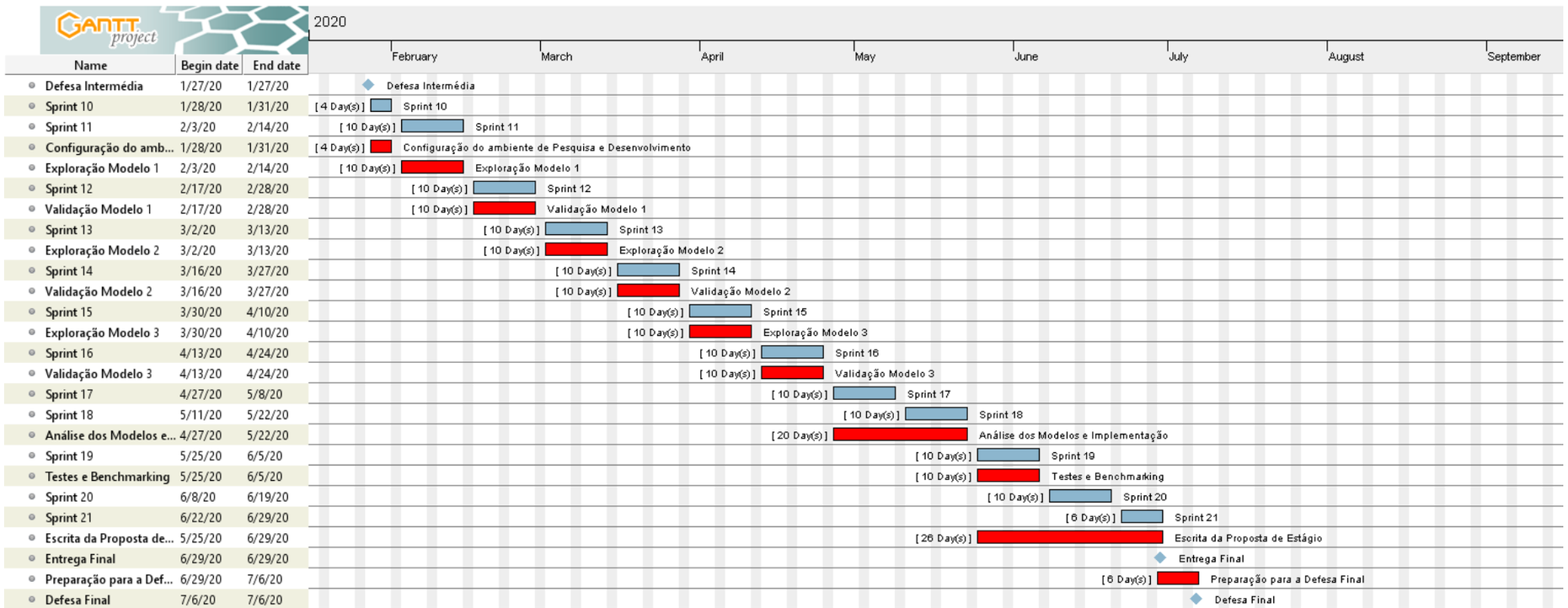


Figura 3.4: Planeamento do trabalho para o segundo semestre

3.4 Implementação dos Algoritmos

Apresenta-se nesta secção uma análise detalhada de todos os algoritmos a implementar no decorrer do estágio. Como foi referido na Secção 2.6, apenas iremos considerar para implementação os algoritmos de extração e análise de informação de documento baseados em exemplos. Na revisão da literatura, foram analisados 6 artigos referentes a algoritmos que cumprem este critério. Analisados os seus pontos fortes e fracos, optou-se pela implementação dos algoritmos *CloudScan* de Palm et al.[52], o *CUTIE* de Zhao et al.[74], o *Chargrid* de Katti et al.[33], com a adicionante de também se implementar o algoritmo *BERTgrid* de Denk and Reisswig[7].

De seguida, serão explicitadas as arquiteturas de cada um dos algoritmos a implementar. Na implementação dos algoritmos, tem-se em consideração a informação disponibilizada pelos autores nos artigos consultados e, sempre que esta se revelar insuficiente, será colmatada com as minhas competências técnicas.

3.4.1 *CloudScan*

O *CloudScan* [52] é um serviço de compreensão e extração de informação de faturas providenciado pela empresa *Tradeshift*. Este método alia algumas propriedades de engenharia, nomeadamente na construção de *features*, juntamente com propriedades de Aprendizagem Profunda, ou *Deep Learning*. No seu formato original, foram comparadas duas abordagens: uma primeira onde é utilizado um classificador de regressão logística onde são usados n-gramas e respetivas features para cada n-grama, e uma segunda abordagem que utiliza um modelo LSTM. Nesta última abordagem, não são utilizados N-gramas, sendo que as features são calculadas para cada palavra. Os autores referiram que neste caso era preferível modelar palavras ao invés de n-gramas, de forma a que o modelo leia o texto de forma mais natural seguindo uma ordem da esquerda para a direita. Na implementação que irá ser efetuada neste estágio, apenas irá ser seguida a segunda abordagem. Esta escolha foi tomada com base no desempenho de ambas as abordagens.

Na Figura 3.5, podemos visualizar o *Workflow* da implementação que irá ser efetuada.

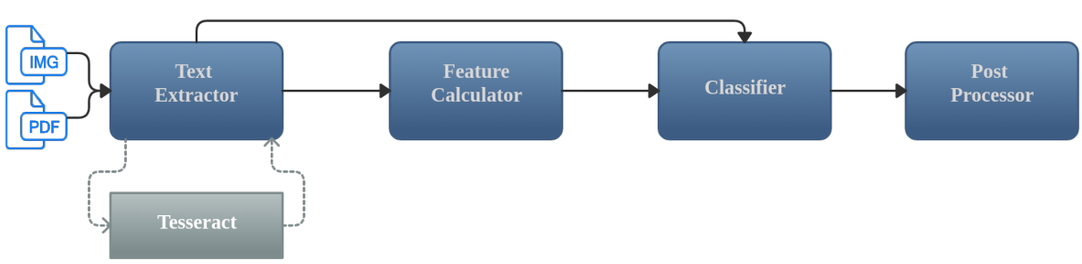


Figura 3.5: CloudScan Workflow

Na Figura 3.5, podemos dividir o algoritmo *CloudScan* em 4 grandes etapas. São elas a etapa de extração de texto, do cálculo de *Features*, do classificador e por fim, dos pós-processamento. De seguida, irá ser explicado o que consiste cada uma das etapas.

Extração de Texto

Nesta etapa é efetuada a extração do texto do documento. Caso o documento esteja em formato de PDF com texto embutido, o texto é extraído, caso contrário terá que ser usado um serviço de RCO. No método *CloudScan*, a ferramenta de RCO a ser utilizada é a ferramenta *open-source Tesseract*. Apesar do desempenho ser inferior à ferramenta de RCO *Google OCR* [64], foi escolhida esta ferramenta pela forma como dispõe os resultados — a camada mais baixa do Tesseract (excluíndo a camada palavra) é a camada linha, enquanto no *Google OCR* (excluíndo as camadas símbolo e palavra) é a camada parágrafo. Ainda assim, apesar de necessitar um esforço adicional na implementação, há formas de contornar a mudança de linha no *Google OCR* e caso este algoritmo seja escolhido para a prova de conceito, a ferramenta *Google OCR* será integrada. O resultado desta etapa é uma matriz composta por linhas e colunas. Para cada palavra, é extraída a sua *Bounding Box* com a posição relativa da palavra. A *Bounding Box* tem o seguinte formato: *(x-left-top,y-left-top, x-right-bottom,y-right-bottom)*. Cada elemento da matriz contém a palavra e respetiva *Bounding Box*. Cada palavra é adicionada à matriz final respeitando a linha original no documento e a sua posição horizontal relativa às outras palavras na mesma linha.

Cálculo de *Features*

Nesta etapa são calculadas as *features* para cada uma das palavras. Na abordagem que foi seguida na implementação, as *features* poderão ter propriedades numéricas e booleanas (verdadeiro ou falso) num total de 24 *features* (Apêndice A). O resultado desta etapa é um vetor de *features* para cada palavra.

Na fase de treino da rede neuronal, é nesta etapa que é construído o *target* da previsão do documento. Visto que cada uma das classes pode conter múltiplas palavras, as palavras são anotadas seguindo o esquema de *IoB Labelling* [57]. Neste esquema, a primeira palavra de uma classe é anotada com a *tag* B (*Begin*), as restantes palavras da mesma classe são anotadas com a *tag* I (*Inside*). Todas as palavras que não pertencem a nenhuma das classes são anotadas com a *tag* O (*Outside*). Isto faz com que as 15 classes originais (14 + a classe *Unknown/Background*), transformem-se em 29 classes finais — cada classe passa a ser representada em 2 classes - *Begin* e *Inside*.

Classificador LSTM

Antes das palavras do documento entrarem dentro do classificador, é efetuado um *Hash* das mesmas. Segundo os autores, o hash da palavra resulta num vetor binário de tamanho 2^{18}. Devido a algumas dificuldades que surgiram na integração do vetor binário no *layer Embbeding* — o *layer Embbeding* considera que cada palavra que entra esteja representada como um número e não numa lista de valores binários — foi optado por fazer *hashing* da palavra representado-a como um número inteiro. Depois de ser efetuado o *embbeding* de cada palavra, ocorre um *Dropout* de 50%. De seguida, o resultado do *Dropout* é concatenado com

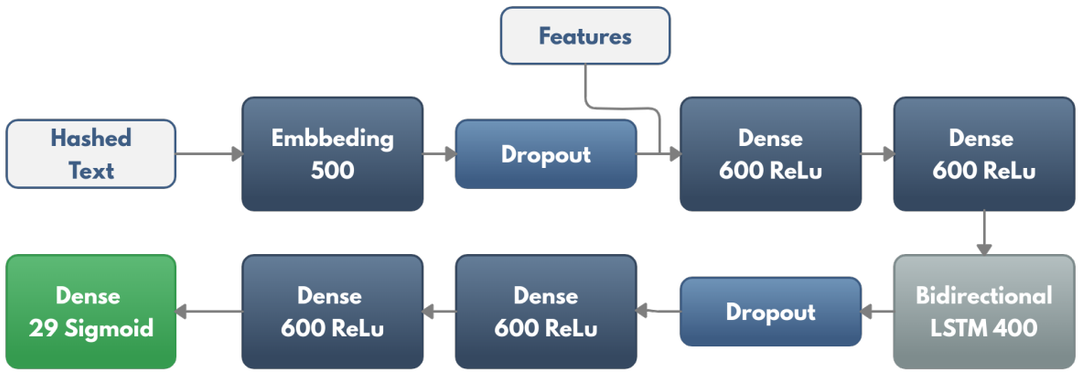


Figura 3.6: Classificador LSTM do CloudScan [52]

a lista de *features* da palavra. Depois de concatenados, passam por dois *Dense Layers* com 600 unidades e ativação *ReLu* e entram na layer *LSTM Bidirectional* com 400 unidades. Posteriormente, ocorre novamente um *Dropout* de 50%, e voltam a percorrer outra vez duas *layers* de 600 unidades com ativação *ReLu*. Por fim, a *layer* final é também ela *Dense*, sendo que as unidades nesta são 29 — correspondente ao número de classes *Target* —, sendo que é utilizada o método de ativação sigmoidal. O resultado desta classificador é uma lista de probabilidades de cada palavra ser pertencente a uma das classes.

Pós-Processamento

A fase de pós-processamento do algoritmo *CloudScan* é relativamente simples. Nesta fase, cada palavra é associada à classe com maior probabilidade. Posteriormente, as sub-classes *Begin* and *Inside* da classe original são agrupadas. No artigo indicam que as classes que contêm palavras na sub-classe *Inside*, mas nenhuma na sub-classe *Begin* são ignoradas, no entanto, neste caso optou-se por não ignorar visto que o desempenho diminuia consideravelmente nesse caso — por vezes, o classificador prevê todas as palavras de uma classe corretamente, mas indica a primeira palavra dessa classe como *Inside* incorretamente. Apesar de incorreto, no agrupamento das classes as palavras estão pela ordem correta e bem previstas, pelo que foi considerado não ignorar nesses casos.

3.4.2 CUTIE

O método CUTIE [74] é um método proposto por Zhao et al. para compreensão e extração de documentos estruturados. Este método representa o documento como uma grelha de palavras, cuja representação servirá como entrada numa RNC onde as palavras são embutidas como *features* vetorizadas com conotações semânticas.

O artigo de investigação do método de *CUTIE* tinha alguma falta de informação, que foi colmatada pela disponibilização do código-fonte da rede neuronal por parte dos autores[12]. Ainda assim, o código disponibilizado, no período da sua consulta (Março/Abril 2020) não estava na sua versão final, sendo que havia muitas variações de redes neuronal e nenhuma

[12]https://github.com/vsymbol/CUTIE

correspondia à versão relatada no artigo. Ainda assim, foi possível juntamente com o artigo complementar a informação discrepante, sendo que a implementação de algumas pequenas etapas dependeu da minha interpretação, pelo que a implementação final poderá não corresponder completamente à versão original dos autores.

Na Figura 3.7, podemos visualizar a sequência de passos que o algoritmo *CUTIE* contém.

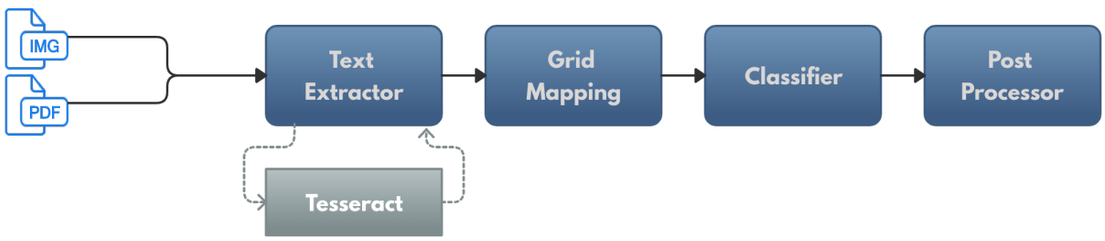


Figura 3.7: CUTIE Workflow

O algoritmo *CUTIE* é composto por 4 etapas, sendo elas, a extração de texto, o mapeamento da representação, a classificação e o pós-processamento. De seguida, irei explicar melhor o que consiste cada uma delas.

Extração de Texto

Nesta etapa é efetuada a extração do texto do documento. Caso o documento esteja em formato de PDF com texto embutido, o texto é extraído, caso contrário terá que ser usado um serviço de RCO. No método *CUTIE*, a ferramenta de RCO a ser utilizada é a ferramenta *open-source Tesseract*. Analogamente aos motivos de escolha desta ferramento por parte do *CloudScan*, o facto da camada mais baixa por parte do *Tesseract* ser a linha, enquanto no *Google OCR* é o parágrafo, facilita a escolha da primeira pela forma como a representação é mapeada — neste caso é preferível as palavras estarem organizadas por linhas e colunas, em vez de outras organizações mais complexas, isto porque na mesmo espaço vertical, podem estar 2 blocos (um à esquerda e outro à direita), o que faz com que não sejam interpretados como a mesma linha por parte do *Google OCR*. Há formas de contornar a mudança de linha no *Google OCR* e caso este algoritmo seja escolhido para a prova de conceito, a ferramenta *Google OCR* será integrada. O resultado desta etapa é uma array composta por linhas e colunas. Para cada palavra, é extraída a sua *Bounding Box* com a posição relativa da palavra. A *Bounding Box* tem o seguinte formato: *(x-left-top,y-left-top, x-right-bottom,y-right-bottom)*. Cada palavra é adicionada à array final respeitando a linha original no documento e a sua posição horizontal relativa às outras palavras na mesma linha.

Mapeamento da Representação

O primeiro passo do mapeamento da representação é o processo de *word tokenizer*. Neste processo simples, todos os caracteres alfabéticos são convertidos para minúsculas e os caracteres numéricos são convertidos para 0's. Estas palavras são depois convertidas num código numérico utilizando um dicionário pré-definido (com todas as *tokenized words* dos dados de

treino). Paralelamente a este processo, são guardadas as posições da palavras que se situam numa posição mais extremada (esquerda, direita, cima e baixo), o número de linhas que o documento tem e a linha com o maior número de palavras. Estes dados servirão para, respetivamente, definir a *Bounding Box* do documento e inicializar a grelha de representação.

Na fase de construção da grelha de representação, começa-se por inicializar a grelha. Posteriormente, para cada palavra presente no RCO, é calculada a sua posição na grelha. Se a posição correspondente já estiver ocupada com outra palavra (na verdade, está ocupada com o código númerico da palavra), é efetuado um deslocamento das palavras na mesma linha. Caso as colunas nessa linha estejam todas ocupadas, é necessário adicionar uma nova coluna a todas as linhas. Na fase final da representação, são removidas todas as linhas e colunas sem nenhuma palavra. Por fim, é efetuado um redimensionamento da grelha para um tamanho alvo de 60 por 60. Na Figura 3.8, podemos ver um exemplo da grelha de representação do método *CUTIE*.

(a)

	BIEVENIDO	AL	SERVICIO	
TAXI	DEL	A.P.C	DE	MADRID
EUGENIO		DOMINGUEZ		GALLEG
N°	LICENCIA:			14670
		...		
TOTAL	(I/IVA):	12.45		€
**	I.V.A.		INCLUTIDO	**
DIST.	SERVICIO:		5.9	km
TARIFAS	TR:			1
HORA	INICIO:			08:59
HORA	FINAL:			09:23
-DATOS	CLIENTE:			
-ORIGEN:				
-DESTINO:				

Figura 3.8: Grelha de Representação do método *CUTIE*

Apesar de não estar presente na versão do artigo, foi decidido testar a utilização do método *CUTIE* juntamente com o modelo de linguagem *BERT*. Para isso, nesta fase em vez de se utilizar o processo *tokenizer words*, é usado o *embbeding* proveniente do modelo *BERT*.

Classificador

No artigo de investigação do método *CUTIE*, foram apresentadas duas redes neuronais, as redes *CUTIE-A* e *CUTIE-B*. Neste estágio só se irá implementar uma das redes e optou-se pela rede *CUTIE-B* por apresentar melhores resultados. Na Figura 3.9, podemos visualizar a arquitetura dessa rede neuronal.

A rede neuronal proposta é composta por 6 blocos. No primeiro bloco, o *input* é introduzido num *layer* de embbeding com dimensão 128. Depois das palavras serem vetorizadas, ocorre um Dropout de 10%. O segundo bloco é composto por 4 *layers* convolucionais com um *kernel* de tamanho 3 por 5 e com 256 canais. O terceiro bloco também é ele composto por 4 *layers* convolucionais com *kernel* de 3 por 5 e 256 canais, mas neste caso com um rácio

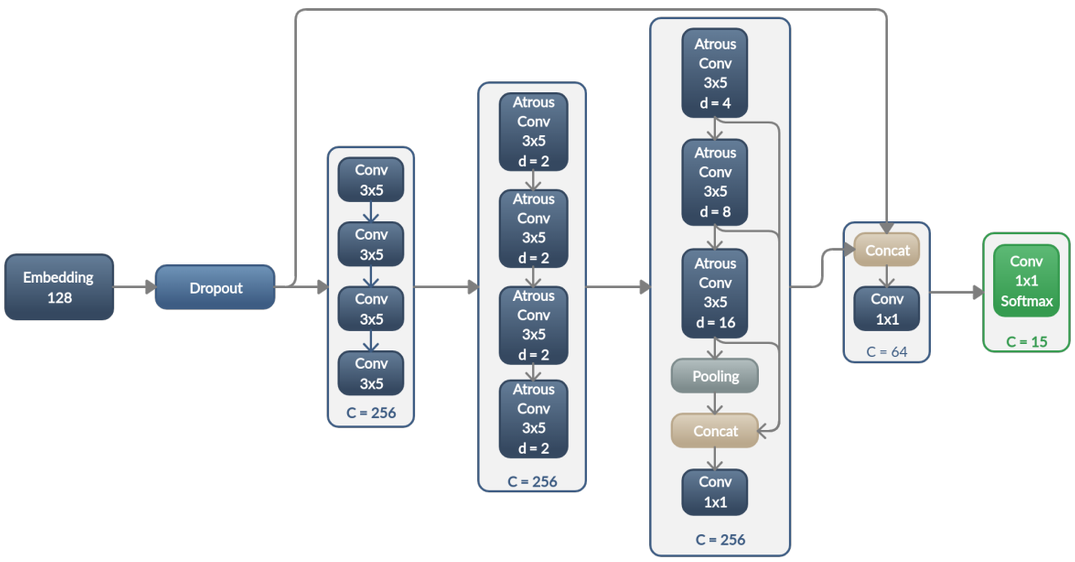


Figura 3.9: Arquitetura da Rede Neuronal CUTIE

de dilatação de 2. O terceiro bloco é composto inicialmente por 3 *layers* convolucionais com *kernel* de 3 por 5, sendo que a dilatação é de respetivamente, 4, 8 e 16. De seguida é passado por um *Pooling layer*. Posteriormente ao *Pooling*, há uma concatenação de todos os *layers* anteriores pertencentes ao mesmo bloco. Há ainda por fim neste bloco um último *layer* convolucional com kernel de 1 por 1. Neste bloco, todos os *layers* continham 256 canais. No penúltimo bloco, há uma concatenação do resultado do *Dropout* com o resultado do bloco anterior, passando posteriormente por um *layer* convolucional de kernel 1 por 1 e 64 canais. Por fim, o último bloco é composto por um único *layer* convolucional com *kernel* de 1 por 1, com o método de ativação *Softmax*. Neste último *layer*, o número de canais corresponde ao número de classes.

No cenário da junção do método *CUTIE* com o *BERT*, descartou-se o *layer* inicial de *Embedding*, visto que o *BERT* já providencia a vetorização das palavras.

Pós-processamento

Na fase de pós-processamento do método *CUTIE*, é iterada cada palavra da representação original, sendo que é atribuída à classe com maior probabilidade prevista pelo classificador. No final, todas as palavras da mesma classe são agrupadas.

3.4.3 *Chargrid/BERTgrid*

O *Chargrid* [33] é um método de extração de documentos estruturados. Este método representa o documento como uma grelha de caracteres o que preserva a estrutura original do documento. No seu formato original, este algoritmo é capaz de prever a máscara de segmentação semântica, cuja finalidade é classificar corretamente os caracteres pertencentes à mesma classe, e também consegue prever as *bounding boxes* de classes pertencentes à mesma entidade — como é o caso das tabelas de produtos/serviços, que pode conter classes como descrição, quantidade e custo do produto/serviço—utilizando regressão para

esse efeito. Na implementação que irá ser efetuada, não iremos considerar a regressão das *Bounding Boxes*. Esta opção deve-se pelo facto de não considerarmos a extração dos campos de produtos/serviços essencial nesta fase, pelo que se optou por não incluir essas classes no nosso conjunto de dados.

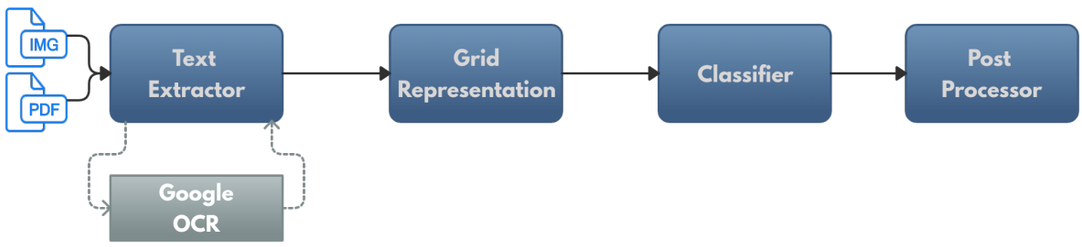


Figura 3.10: Chargrid Workflow

Na Figura 3.10, podemos dividir o algoritmo *Chargrid* em 4 etapas principais. São elas a etapa de extração de texto, da construção da grelha de representação do *Chargrid*, do classificador e, por fim, do pós-processamento. De seguida irá ser explicado o que consiste cada uma das etapas.

Extração de Texto

Nesta etapa é efetuada a extração do texto do documento. Caso o documento esteja em formato de PDF com texto embutido, o texto é extraído, caso contrário terá que ser usado um serviço de RCO. No método *Chargrid*, a ferramenta de RCO a ser utilizada é a ferramenta *Google OCR*. Neste algoritmo, optou-se pela escolha do *Google OCR*, pois não era fundamental os resultados estarem apresentados por ordem cima-baixo da esquerda para a direita (ao contrário do *CloudScan*). Para além disso, o *Google OCR* tem um desempenho muito superior quando a extração é efetuada ao nível do caracter [64]. Por esse mesmo motivo, escolheu-se a ferramenta que é uma das utilizadas na plataforma IDV e também a que tem melhor desempenho.

O resultado desta etapa é uma lista composta por tuplos, em que cada um contém o caracter e respetiva *Bounding Box*. Para cada caracter, é extraído a sua *Bounding Box* com a posição relativa do caracter. A *Bounding Box* tem o seguinte formato: *(x-left-top,y-left-top, x-right-bottom,y-right-bottom)*.

Representação *Chargrid*

Na etapa da representação *Chargrid*, o primeiro passo é atribuir um valor escalar para cada caracter ($a \rightarrow 1, b \rightarrow 2$, etc...). Por uma questão de desempenho, apenas foram considerados os 60 caracteres mais frequentes, sendo que os restantes foram catalogados como desconhecidos.

Segundo os autores, é criada uma representação vectorial do tamanho $\mathbb{R}^{H,W,N_c}$, sendo que H corresponde à altura, W à largura e N_c ao número de caracteres. A área ocupada por um caracter é ocupada pelo valor escalar correspondente, sendo o restante (caracteres

Figura 3.11: Chargrid Representation Workflow

desconhecidos e *background*) mapeado a 0. Posto isto, começou-se por inicializar uma matriz tridimensional de zeros (Etapa 1 da Figura 3.11), sendo as primeiras duas dimensões referentes à altura e largura e a terceira dimensão com um tamanho igual a 2. O passo seguinte foi a partir do resultado do RCO obtido no passo anterior, obter as linhas e colunas ocupadas pela área de cada caracter. De forma a garantir, que o mesmo pixel não contenha mais do que um caracter, é verificado se as linhas e colunas da matriz pertencentes ao caracter já estavam previamente ocupadas. Em caso afirmativo, é calculada a distância até ao centro da área de cada um dos caracteres, sendo que o pixel é atribuído ao caracter com menor distância. É por este motivo que a terceira dimensão da matriz inicialmente implementada tem tamanho 2 — o primeiro elemento corresponde ao valor escalar do caracter, o segundo elemento corresponde ao índice do caracter no RCO. Paralelamente à atualização da matriz com os novos caracteres, são guardadas algumas estatísticas, tais como a menor largura e altura dos caracteres presentes. Finalizada a atualização da grelha com todos os caracteres do documento, removeu-se a segunda coluna da terceira dimensão, sendo que a matriz passou a ser representada bidimensionalmente.

Na Etapa 2 da Figura 3.11, é efetuado o primeiro *downsampling* da representação chargrid. Segundo os autores, o primeiro *downsampling* é reduzido até duas vezes a menor resolução encontrada. Então, o primeiro passo foi obter o caracter com menor comprimento (quer em altura ou largura). De seguida, é calculado o factor de redução tendo em conta que o menor caracter contenha pelo menos 2 pixeis, garantindo assim que não há perda de informação. Por fim, é efetuado o *downsampling* da representação utilizando a interpolação por *Nearest Neightbor*.

A terceira etapa é apenas considerada na fase de treino, onde é efetuado um *padding* aleatório de zeros em cada direção até um máximo de 16 elementos/pixéis de forma a criar maior variação dentro do conjunto de dados.

Na quarta etapa é efetuado um *1-Hot Encoding*, cuja a representação que antes deste passo se encontrava em formato bidimensional (altura por largura), passa a representar-se como uma matriz tridimensional, sendo que a terceira dimensão corresponde aos caracteres. Por fim, na última etapa é efetuada o *downsampling* final para uma resolução alvo de 355 por 255. Os autores consideram a resolução alvo como 356 por 256, mas no processo de *Encoder* e *Decoder* da rede neuronal, a resolução final era diferente à inicial — as operações contidas na rede neuronal faziam com que a resolução final tivesse uma dimensão ímpar, ficando com dimensão 355 por 255, por isso de forma a facilitar a comparação entre o *input* e o resultado optou-se por diminuir a resolução inicial para 355 por 255.

Um exemplo da transformação do doucmento original para a representação do *Chargrid*,

pode ser visualizada na Figura 3.12.

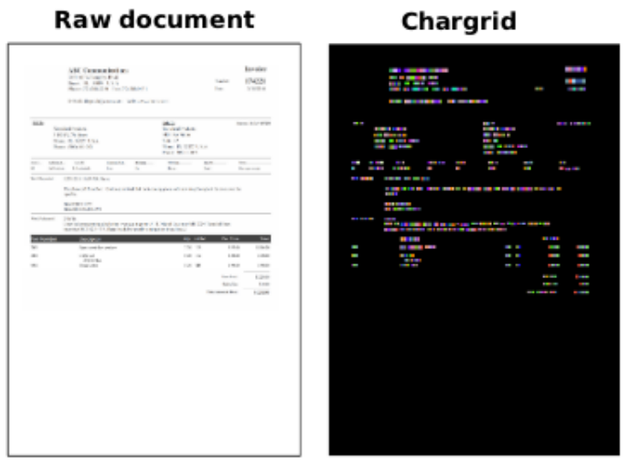


Figura 3.12: Documento Original vs Representação *Chargrid*

Classificador

A rede neuronal do *Chargrid* é uma rede convolucional que é composta por 2 fases: a fase de codificação e a fase de descodificação. Na Figura 3.13 é possível visualizar a arquitetura da rede neuronal convolucional do *Chargrid*.

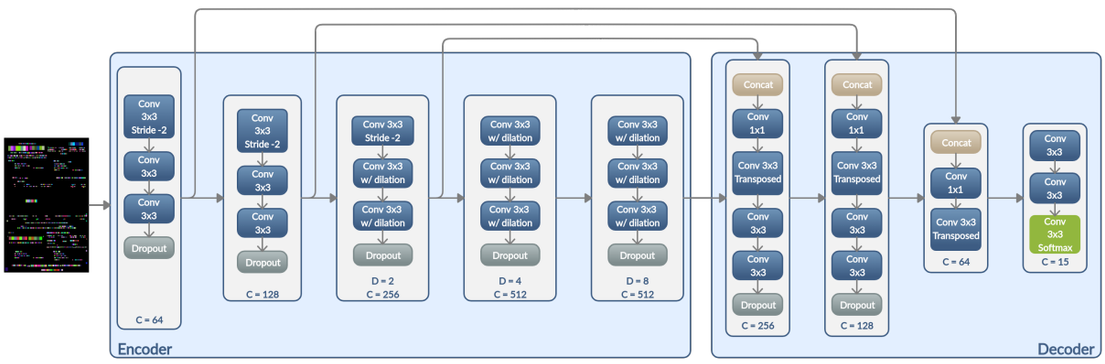


Figura 3.13: Rede Neuronal *Chargrid*

A fase de codificação é composta por cinco blocos em que o número de canais é dobrado em cada bloco começando em 64. Todos os *layers* são seguidos de uma *Batch Normalization* e do método de ativação *ReLu* com a exceção da última *layer*.

Exceptuando no número de canais, os primeiros dois blocos do *Encoder* são iguais, compostos por uma *layer* convolucional bidimensional com *strides* de 2 em 2 e um *kernel* bidimensional de tamanho 3, sendo seguido por duas *layers* convolucionais com *kernel* de tamanho 3 e um *Dropout* espacial com 10% de probabilidade. O terceiro bloco é composto por 3 *layers* convolucionais com *kernel* de tamanho 3, sendo que o primeiro *layer* contém *strides* de 2 em 2 e os restantes contém um rácio de dilatação igual a 2. O quarto e quinto bloco, também são eles idênticos com a exceção do rácio de dilatação, sendo compostos por 3 *layers* convolucionais com dilatação 4 e 8 respetivamente, sendo seguidos por um *Dropout* espacial com probabilidade de 10%.

Em relação ao *Decoder*, os 3 primeiros blocos concatenam o resultado do bloco anterior com o resultado de blocos do *Encoder* com o mesmo número de canais, sendo respetivamente, o terceiro, segundo e primeiro bloco do *Encoder*. Os primeiros blocos são idênticos com a exceção do número de canais, sendo compostos por 4 *layers* convolucionais, sendo que o primeiro *layer* contém um *kernel* de tamanho 1 e os restantes de 3, e o segundo *layer* é transposto, seguidos posteriormente por um *Dropout* Espacial de 10%. O terceiro bloco contém um *layer* convolucional com *kernel* de tamanho 1 e um segundo *layer* com um *kernel* de tamanho 3 e transposto. O último bloco do *Decoder* é composto por 3 *layers* convolucionais com *kernel* de tamanho 3, sendo que o último *layer* um método de ativação *Softmax* e contém 15 canais, correspondentes ao número de classes.

O algoritmo de *Chargrid* utiliza como método de inicialização de pesos o método de He[21], com um decay de 0.0001. O modelo é treinado com recurso ao *Optimizer SGD* com *Momentum* de 0.9 e um *Learning Rate* de 0.05. Para além disso, é utilizado um método agressivo de balanceamento de classes denominado por *Focal Loss* dinâmico[40] com uma constante C de 1.04.

Pós-Processamento

.

Segundo os autores, na fase de pós-procesamento os caracteres pertencentes à mesma classe são agrupados. No entanto, como o resultado do classificador tem um tamanho de $355x255x15$ — as primeiras duas dimensões referentes à altura e largura e a última dimensão referente a cada uma das classes —, não é possível identificar o caracter correspondente utilizando apenas o resultado do classificador. Posto isto, foi necessário utilizar o RCO do documento para identificar qual o caracter correspondente a cada pixel. Sendo assim, para cada caracter presente no RCO, calculou-se a área que este ocupava no documento final, e calculou-se a soma das probabilidades de cada classe nessa área. O caracter era então atribuído à classe com maior probabilidade na área ocupada.

BERTGrid

O *BERTgrid* é uma continuação por parte de Denk e Reisswig do trabalho efetuado por Katti et al. O *BERTgrid* difere do *Chargrid* na medida em que, em vez de a grelha ser ao nível do caracter, a representação é efetuada ao nível da palavra. Para além disso, é utilizado o modelo de linguagem BERT [8] que permite oferecer um maior contexto textual. Para a codificação das palavras foi utilizado o serviço *BERT Serving*[13]. Foram utilizados dois modelos *BERT*, um treinado com palavras de múltiplas línguas[14] e outro com palavras portuguesas[15]. O desempenho do *BERTgrid* com os diferentes modelos pode ser visto na secção 4.2

[13] `https://github.com/hanxiao/bert-as-service`

[14] `https://storage.googleapis.com/bert_models/2018_11_23/multi_cased_L-12_H-768_A-12.zip`

[15] https://github.com/neuralmind-ai/portuguese-bert

Comparativamente com o método de *Chargrid*, a grande diferença está na etapa da criação da representação, sendo que nas restantes apenas tiveram que se ajustar pequenas modificações de forma a garantir a conexão correta entre as etapas. Sendo assim, analogamente ao *Chargrid*, começou-se por inicializar uma matriz bidimensional de zeros do tamanho do documento original. Para cada palavra do RCO é verificado a área que ocupa e os pixéis são atribuídos de acordo com o *Chargrid*. Como nesta representação, não há *1-Hot Encoding* é efetuado logo o *downsampling* para a resolução final de $355x255xEncodingDimension$, sendo a dimensão do Encoding igual a 768 no caso do *BERT* multi-linguas e 1024 no *BERT* português. Por fim, já com a resolução alvo, cada linha de palavras são codificadas com o modelo *BERT*, sendo que o vector de representação obtido para cada palavra é atribuído à área do documento correspondente — poderia-se codificar logo na fase inicial as palavras e atribuir à àrea correspondente na matriz no seu tamanho original,no entanto, o processo tornava-se muito complexo (as dimensões são muito altas), pelo que operações de redimensionamento e procura tornavam-se muito lentas ou até inexequíveis.

Na Figura 3.14 podemos visualizar as representações do *Chargrid* e *BERTgrid*, através da atribuição de uma cor para cada código. Apesar de nesta visualização da representação não ser visivel a contextualização que é fornecida na representação *BERTgrid*, as representações dos dois algoritmos permitem-nos evidenciar que a codificação no *Chargrid* é efetuada ao nível do caracter, enquanto na representação do *BERTgrid* a codificação é efetuada ao nível da palavra.

(a) Representação *Chargrid* (b) Representação *BERTgrid*

Figura 3.14: Comparação das representações *Chargrid* e *BERTgrid*

3.5 Validação dos Algoritmos

A qualidade dos dados utilizados para validar um algoritmo tem um impacto muito significativo no desempenho destes. Um bom conjunto de dados permite aos algoritmos descobrir heurísticas e padrões que poderão ser utilizados em dados nunca antes vistos. s

Um dos objetivos que se pretende com este estágio é implementar uma prova de conceito

de um sistema baseado em exemplos que consiga atingir resultados satisfatórios e, se possível, se consiga ser versátil para diferentes documentos. Para verificar a versatilidade, o conjunto de dados não poderá ser composto por apenas documentos do mesmo tipo. Não é fácil arranjar um conjunto de dados de qualidade e que seja muito variado. Ainda assim, a CSW está consciente deste problema e aceitou disponibilizar os seguintes conjuntos de dados:

- **Faturas -** São disponibilizados cerca de 17000 documentos de faturação da própria CSW.

- **Certificados de Nascimento e Passaportes -** São também disponibilizados um conjunto pequeno de documentos de certificados de nascimento e passaportes. Neste caso, estes documentos são fornecidos por um cliente da CSW. De forma a manter a confidencialidade dos dados, não é possível revelar mais nenhum tipo de informação relativamente a este conjunto de dados.

Infelizmente, devido à situação que afligiu o mundo no ano de 2020 e que nos obrigou a entrar num confinamento obrigatório e a trabalhar em regime de teletrabalho, levantou algumas burocracias e dificultou o acesso a alguns conjuntos de dados mais confidenciais, nomedamente dos certificados de nascimento e passaportes, que, numa fase inicial foram considerados para exercerem o teste de desempenho dos algoritmos implementados.

Numa fase inicial, estes serão os conjuntos de dados que irão servir de validação para os algoritmos implementados. Numa fase mais adiantada, caso seja possível, poderá haver um conjunto de dados maior e mais diversificado.

Relativamente às métricas de validação, foram analisadas as métricas utilizadas para validar os trabalhos relacionados com o projeto. O resultado dessa análise pode ser visualizada na tabela 3.1.

Métricas \ Métodos	Esser et al. [12]	Yang et al. [70]	CloudScan [52]	Attend, Copy, Parse [51]	Chargrid [33]	BERTgrid [7]	Liu et al. [41]	CUTIE [74]
Precision	✓		✓	✓			✓	✓
Recall	✓		✓	✓			✓	
F1 Score	✓		✓	✓			✓	
Intersection-Over-Union		✓						
Personalized Formula (2.5)					✓	✓		
Average Precision								✓

Tabela 3.1: Métricas utilizadas na validação dos métodos relacionados

Efetuada a análise das métricas utilizadas por projetos idênticos, as métricas que serão utilizadas para validar os modelos a implementar serão a *F1 Score* e, consequentemente, Precisão e *Recall*, embora estas últimas serão tidas em contas como métricas auxiliar e não serão apresentadas neste relatório, para não tornar os resultados muito extensos e confusos. No nosso serviço de extração pretendemos evitar ao máximo os casos de falsos positivos (situações em que identifica erradamente um campo numa classe), sendo que também é desejável evitar os cenários de falsos negativos (a não identificação de uma classe que estava presente no documento), embora tenham menos impacto no desempenho global. Posto isto, visto que o *F1 Score* avalia equilibradamente os casos de falsos positivos e falsos negativos,

irá ser a métrica principal na validação do serviço que será implementado. Estas métricas serão utilizadas ao nível da palavra, sendo que uma palavra só é considerada como verdadeira positiva se corresponder na sua totalidade á original. Recorrendo a um exemplo, considera-se um campo original da morada como "Rua da Terceira Esquina". Se o algoritmo prever esse campo como "Rua da Terceira Total", são contabilizados 3 verdadeiros positivos, "Rua da Terceira", 1 falso positivo, "Total", e 1 falso negativo, "Esquina". Com estes resultado, obtínhamos um desempenho de 75% na métrica Precisão, 75% no *Recall* e também 75% no *F1-Score*. Podia-se ter considerado a utilização das métricas ao nível da classe, o que neste caso obtínhamos 0% em todas as métricas, mas considero que a métrica ao nível da palavra oferece maior informação do que realmente está a acontecer na previsão, do que simplesmente considerar uma classe totalmente correta ou incorreta.

3.5.1 Conjunto de Dados

O conjunto de dados que irá ser utilizado no treino, validação e teste dos algoritmos irá ser composto apenas por faturas e recibos. Infelizmente, devido a algumas questões burocráticas, não foi possível obter alguns dos conjuntos de dados anteriormente referidos.

Foi disponibilizado inicialmente um conjunto de 17528 faturas não anotadas. Foi do âmbito do meu estágio proceder à anotação desses mesmos dados, contando com uma ajuda inicial por parte de colegas de equipa. O processo de anotação foi efetuado através da ferramenta *Dataturks*[16]. Nesta, para cada conjunto de dados, são definidas N classes, sendo que a cada uma lhes é atribuída um nome. Posteriormente, na fase de anotação de cada imagem — correspondente a cada documento —, para cada uma das classes é desenhada uma *bounding box* em formato retangular na área da imagem que se pretende atribuir à respetiva classe. No caso da anotação das faturas, a área é sobreposta sobre as palavras que se pretendem incluir na classe correspondente. Na fase final, é retornado um ficheiro em formato JSON, contendo todas as anotações efetuadas, sendo que em cada linha contém o nome do ficheiro da imagem anotada e respetivas anotações das classes. Para cada uma das classes anotadas, é disponibilizada a posição dos quatro vértice da *Bounding Box*.

Devido à morosidade no processo de anotação de documentos — era necessário cerca de 2 minutos para a anotação de cada documento —, foram anotadas um total de 1210 faturas.

Neste conjunto de dados, foram consideradas um total de 15 classes, sendo elas:

- ***Label Data* -** Corresponde à *label* da data da emissão do documento. Na Figura 3.15, o texto pertencente a esta classe é "Invoice Date".
- ***Valor Data* -** Corresponde à data de emissão do documento. Na Figura 3.15, o texto pertencente a esta classe é "Jan 4th, 2018".
- ***Label Nº Fatura* -** Corresponde à *label* do número da fatura. Na Figura 3.15, o texto pertencente a esta classe é "Invoice No".

[16] `https://dataturks.com/`

[17] `https://freeinvoicebuilder.com/`

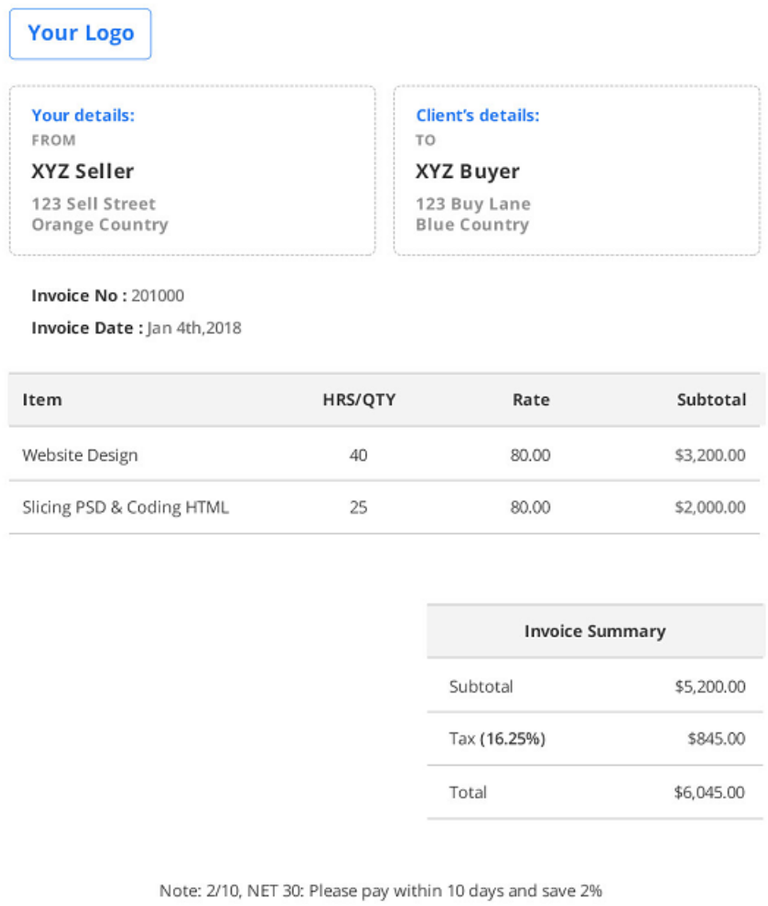


Figura 3.15: Exemplo de uma fatura falsa retirada da plataforma Online Invoice Generator[17]

- ***Valor Nº Fatura*** **-** Corresponde ao número da fatura. Na Figura 3.15, o texto pertencente a esta classe é "201000".

- ***Nome Emissor*** **-** Corresponde à entidade responsável pela emissão da fatura. Na Figura 3.15, o texto pertencente a esta classe é "XYZ Seller".

- ***Nome Recetor*** **-** Corresponde à entidade destinatária da fatura. Na Figura 3.15, o texto pertencente a esta classe é "XYZ Buyer".

- ***Label Morada Emissor*** **-** Corresponde à *label* da morada da entidade responsável pela emissão da fatura. Na figura não se encontra nenhum exemplo desta label, mas um exemplo possível desta classe poderia ser "Morada".

- ***Valor Morada Emissor*** **-** Corresponde à morada da entidade responsável pela emissão da fatura. Na Figura 3.15, o texto pertencente a esta classe é "123 Sell Street Orange Country".

- ***Label Morada Recetor*** **-** Corresponde à *label* da morada da entidade destinatária da fatura. Tal como na classe *Label Morada Emissor*, na figura não se encontra nenhum exemplo desta label.

- ***Valor Morada Recetor* -** Corresponde à morada da entidade destinatária da fatura. Na Figura 3.15, o texto pertencente a esta classe é "123 Buy Lane Blue Country".

- ***Label Total* -** Corresponde à *label* do total da fatura. Na Figura 3.15, o texto pertencente a esta classe é "Total".

- ***Valor Total* -** Corresponde ao total da fatura. Na Figura 3.15, o texto pertencente a esta classe é "6,045.00".

- ***Label Moeda* -** Corresponde à *label* da moeda no qual os valores da fatura estão expostos. Na Figura 3.15, não se encontra nenhum exemplo desta classe, mas um exemplo possível poderia ser "Moeda".

- ***Valor Moeda* -** Corresponde à moeda no qual os valores da fatura estão expostos. Na Figura 3.15, o texto pertencente a esta classe é "$". Nesta classe, é priveligiado o nome da moeda por extenso (como Euro ou Dólar), considerando-se apenas o símbolo em documentos cujo os primeiros não estão presentes.

- ***Background/Unknown* -** Corresponde a toda a área da imagem que não pertence a nenhuma das classes anteriores.

Como é possível observar na Figura 3.16, a grande maioria das classes está presente em mais de 85% do conjunto total de dados.

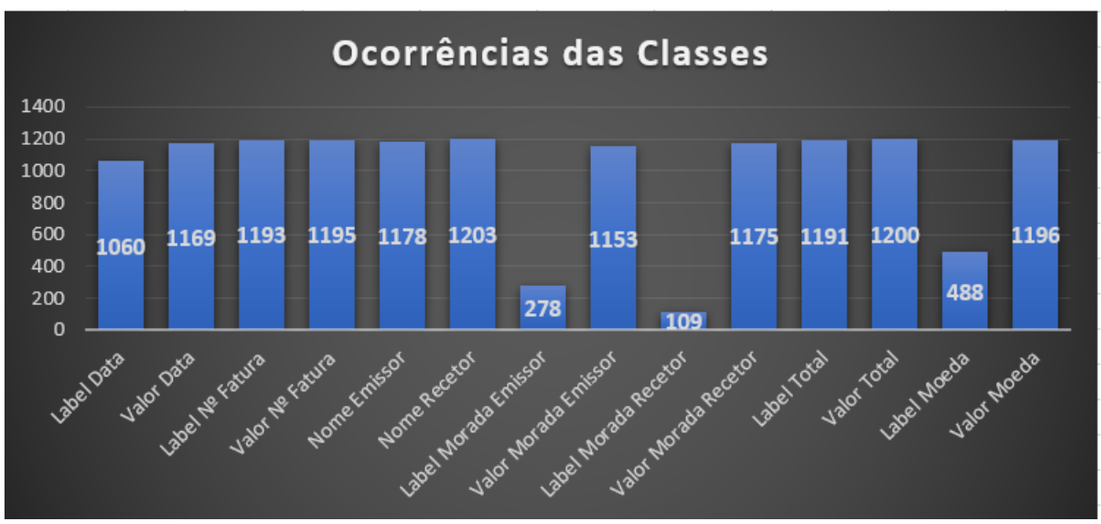


Figura 3.16: Número de ocorrências de cada uma das classes no conjunto total de dados de faturas

De todas estas classes, apenas as classes *Label Morada Emissor*, *Label Morada Recetor* e *Label Moeda* estão presentes em poucos exemplares. Isto é facilmente explicado pelo facto de não ser muito comum aparecer este tipo de *labels* em faturas. Ainda assim, a fraca presença destas classes não é crítica, visto que a informação por estas providenciada não é essencial para efetuar a leitura correta do documento.

Na Figura 3.17, visualizamos o número médio de palavras por cada classe.

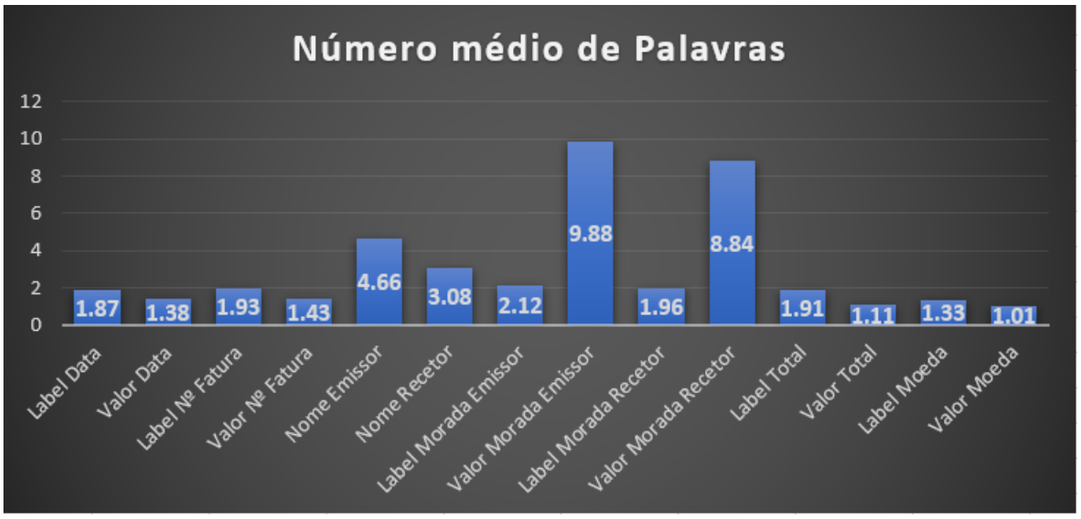


Figura 3.17: Número médio de palavras por classe no conjunto total de dados de faturas

Como seria de esperar, as classes mais textuais, como é o caso da morada do emissor e recetor, são as classes que contém um maior número de palavras. O nome da entidade emissora e recetora, geralmente, também contém múltiplas palavras. De realçar que o nome do recetor é sempre constante no nosso conjunto total de dados, isto pelo facto das faturas utilizadas terem como destinário a "CRITICAL Software, SA". A ligeira diferença do número de palavras com o real (3.08 para 3), pode ser explicado por algum erro de OCR (do qual os algoritmos implementados são alheios) ou por erro de formatação da fatura — embora muito raro, nalgumas faturas é efetuada a mudança de linha a meio da palavra, continuando o restante na linha seguinte. Ora, consequentemente, o OCR interpreta como 2 palavras diferentes.

No conjunto de testes a realizar, as cerca de 1210 faturas irão ser repartidas em treino, validação e teste, sendo respetivamente dividas percentualmente em cerca de 70%, 15% e 15%, sendo compostas respetivamente por 846, 182 e 182 faturas. De forma a poder analisar com maior credibilidade, os diferentes algoritmos irão utilizar os mesmos dados para treino, para validação e para teste.

Capítulo 4

Testes e Resultados

Neste capítulo, serão apresentados os testes de optimização dos algoritmos (Secção 4.1), mais precisamente dos algoritmos *CloudScan* e *CUTIE*. Na secção seguinte (Secção 4.2), serão apresentados os resultados dos testes de desempenho de todos os algoritmos implementados.

4.1 Optimização dos Algoritmos

Na fase de implementação dos algoritmos, verificou-se que a informação disponibilizada nos artigos de investigação que os descreviam, nomeadamente alguns valores de hiper parâmetros, não era suficiente para os poder replicar na totalidade, situação observada particularmente nos artigos que descrevem os algoritmos *Cloudscan* e *CUTIE*.

Os artigos que descrevem o algoritmo *Chargrid* e a sua variação *BERTgrid*, embora contenham alguns passos pouco claros, disponibilizam a informação suficiente para a sua implementação sem a exigência de testes suplementares.

Face ao exposto, optou-se por proceder a testes de optimização dos algoritmos *CloudScan* e *CUTIE*, de forma a tentar identificar os valores ideiais para cada um desses hiper parâmetros em falta.

4.1.1 *CloudScan*

Tal como foi dito anteriormente, no artigo do CloudScan[52] não estava explicado na sua totalidade que parâmetros de optimização do modelo foram utilizados e quais os seus valores. Sendo assim, o primeiro passo passou por aglomerar os valores dos parâmetros disponibilizados no artigo e selecionar valores padrão para os parâmetros que estariam em falta. Na Tabela 4.1, podemos visualizar os valores que irão ser considerados como valores padrão.

Como é possível observar, o *Optimizer Adam*, o *Dropout Rate* de 50% e um *Batch Size* de 96 foram disponibilizados pelos autores do artigo, pelo que fazia sentido considerá-los como parte dos valores padrão. Nos parâmetros *Learning Rate*, *Weight Decay*, *Weight Initialization* e balanceamento de classes, assumiram-se os valores padrão da biblioteca *Tensorflow Keras*, assumindo respetivamente os valores 0.001, None, Glorot Uniform e None.

Parâmetros	Disponibilizado pelo artigo	Valores
Optimizer	✓	Adam
Dropout Rate	✓	50%
Batch Size	✓	96
Learning Rate		0.001
Weights Decay		None
Weights Initilization		Glorot Uniform
Class Balancing		None

Tabela 4.1: Valoes padrão do algoritmo CloudScan

De forma a perceber se a variação dos parâmetros faz aumentar o desempenho do algoritmo CloudScan, decidiu-se efetuar um teste inicial com os valores padrão. Na Tabela 4.2 podemos visualizar os resultados desse mesmo teste.

Desempenho do algoritmo *CloudScan* com os hiper parâmetros padrão				
Label Data	**Valor Data**	***Label* Nº Fatura**	**Valor Nº Fatura**	**Nome Emissor**
66.04%	51.57%	70.45 %	57.25%	65.74 %
Label Morada Emissor	**Valor Morada Emissor**	**Label Morada Recetor**	**Valor Morada Recetor**	**Nome Recetor**
75.19%	43.17%	21.33%	49.32%	74.95%
Label Total	**Valor Total**	**Label Moeda**	**Valor Moeda**	**Desempenho Geral**
65.79%	56.09%	75.72%	55.71%	59.51%

Tabela 4.2: Resultados do algoritmo CloudScan com os parâmetros padrão utilizando a métrica F1-Score

Efetuando uma análise geral aos resultados deste teste inicial, observa-se que algoritmo *CloudScan* conseguiu obter um desempenho aceitável na previsão da classes da ***Label* da Moeda** e da **Morada do Emissor**, assumindo valores, respetivamente, de 75.72% e 75.19%. Por outro lado, o desempenho das classes referentes às moradas, ***Label* da Morada do Recetor** e **Valores das Moradas do Recetor e Emissor**, evidenciaram menor desempenho assumindo valores de, respetivamente, 21.33%, 49.32% e 43.17%. Para além disso, destaca-se a discrepância do desempenho do algoritmo na previsão das classes dos **Nomes do Emissor e Recetor**, assumindo respetivamente valores de 65.74% e 74.95%. Em suma, o desempenho consegue obter um desempenho bastante superior na previsão das *labels*, comparativamente aos valores, sendo que é refletido num desempenho geral de 59.51%.

Efetuado o teste inicial, irão ser procedidos os testes de optimização dos parâmetros. Estes testes irão ser divididos em 2 grandes grupos: um primeiro cujo objetivo é testar os parâmetros *Learning Rate* e *Optimizer* e o segundo conjunto de testes irá servir para testar a influência das diferentes abordagens sobre os pesos no desempenho do algoritmo, sendo que irão ser testados diferentes inicializações de pesos, o *decay* dos pesos e estratégias de balanceamento de classes.

Optimizer* e *Learning Rate

Começou-se inicialmente por testar os parâmetros de optimização Optimizers e Learning rate. Decidiu-se juntar estes dois parâmetros no mesmo teste, visto que ambos os valores estão correlacionados. Na Tabela 4.3 é possível visualizar os valores que irão ser testados para cada um dos parâmetros na realização dos testes.

Parâmetros					
Learning Rate	0.0005	0.001	0.005		
Optimizers					
SGD	Momentum	0	0.3	0.6	0.9
RMSprop [24]	Momentum	0	0.3	0.6	0.9
	Rho	0.8	0.9	0.95	0.99
Adagrad [10]					
Adadelta [72]	Rho	0.8	0.9	0.95	0.99
Adam [35]	Beta 1	0.8	0.85	0.9	
	Beta 2 Offset	0.02	0.05	0.09	0.099
Adamax [35]	Beta 1	0.8	0.85	0.9	
	Beta 2 Offset	0.02	0.05	0.09	0.099
Nadam [9]	Beta 1	0.8	0.85	0.9	
	Beta 2 Offset	0.02	0.05	0.09	0.099

Tabela 4.3: Valores a serem testados no conjunto de testes dos *Optimizers* e *Leaning Rate*

Neste conjunto de testes decidimos variar o *Learning Rate* e combinar os diferentes valores deste com os vários *Optimizers*. Os optimizers são o algoritmo gradiente descendente estocástico (*SGD*), o algoritmo *RMSprop*, o algoritmo *Adagrad* e a sua variação *Adadelta* e o algoritmo *Adam* e as suas variações *Adamax* e *Nadam*. Para estes *Optimizers*, decidiu-se variar algum dos seus parâmetros, como é o caso do *Momentum*, *Rho*, *Beta 1* e *Beta 2* — *Beta 2 = Beta 1 + Beta 2 Offset*. A combinação destes valores correspondeu a um total de 183 testes.

Efetuado o conjunto total de testes (Apêndice B), os parâmetros do algoritmo com melhor desempenho pode ser visto na Tabela 4.4.

Parâmetros			
Learning Rate	Optimizer	Beta 1	Beta 2
0.001	Adam	0.9	0.95

Tabela 4.4: Parâmetros do algoritmo com melhores resultados no teste aos *Optimizers* e *Learning Rate*

O algoritmo com melhor resultado não é muito diferente do algoritmo com os parâmetros padrão. Apenas se difere o valor Beta 2 que por *default* assume o valor 0.999 e neste caso

tem o valor de 0.95. Esta ligeira diferença fez também aumentar ligeiramente o desempenho do algoritmo, como podemos visualizar na Tabela 4.5.

Desempenho da melhor configuração dos Optimizers e *Learning Rate*				
Label Data	**Valor Data**	***Label* Nº Fatura**	**Valor Nº Fatura**	**Nome Emissor**
74.59%	56.14%	68.99 %	65.79%	59.54 %
Label Morada Emissor	**Valor Morada Emissor**	**Label Morada Recetor**	**Valor Morada Recetor**	**Nome Recetor**
78.57%	43.58%	54,54%	61.11%	80.67%
Label Total	**Valor Total**	**Label Moeda**	**Valor Moeda**	**Desempenho Geral**
70.31%	59.54%	73.04%	45.54%	61.95%

Tabela 4.5: Resultados detalhados do algoritmo com os parâmetros da Tabela 4.4, com recurso à métrica F1-Score

Analisando de forma geral a Tabela 4.5, o algoritmo *CloudScan* obteve um desempenho relativamente bom na previsão do **Nome do Recetor** e da ***Label* da Morada do Emissor**, obtendo respetivamente os resultados de 80.67% e 78.57%. Por outro lado, o desempenho foi menor na previsão dos **Valores da Moeda e Morada do Emissor**, obtendo resultados de, respetivamente, 45.54% e 43.58%. Para além disso, há a salientar a discrepância de valores dos dados do emissor e do recetor, havendo uma discrepância de desempenho acima de 10%. Em suma, comparando os resultados deste teste com os observados no teste com valores padrão, verifica-se que há um ligeiro acréscimo de desempenho, sobretudo nas classes dos valores, ainda que o desempenho nestas classes continue muito inferior comparativamente ao observado nas *labels*. Também no desempenho geral do algoritmo, se verifica que há um valor superior, obtendo agora um resultado de 61.95% ao invés dos 59.51% obtidos anteriormente.

De seguida, irá-se proceder os testes à influência do balanceamento das classes e na inicialização e regulação dos pesos no desempenho do algoritmo CloudScan.

Inicialização e Regularização dos Pesos e Balanceamento das Classes

Neste segundo conjunto de testes, optou-se por testar diferentes inicializações de pesos e diferentes *decays*. Para além disso, decidiu-se também testar uma abordagem de *Focal Loss* dinâmico[40] com diferentes valores da constante "c". Esta abordagem permite penalizar mais o erro nas classes menos presentes. Neste caso, o valor de "c" corresponde a um coeficiente de penalização que é utilizado num cálculo para a penalização de cada classe sempre que a previsão é incorreta. Desta forma, quanto menor a área que uma classe ocupar no documento, maior vai ser a penalização na função *loss*. O valor de c é inversamente proporcional à penalização do erro nas menores classes, sendo que quanto maior for o valor de c, maior equilíbro das penalizações e menor penalização das classes menos presentes. Esta abordagem é interessante em conjuntos de dados cujas as classes não estejam balanceadas e que não seja fácil (ou sequer faça sentido) eliminar elementos dessas classes para as balancear.

Na Tabela 4.6, podemos observar os valores dos hiper parâmetros que irão ser testados. Os restantes parâmetros não incluídos nesta tabela irão assumir o seu valor padrão.

Parâmetros					
Weight Initialization	Random Uniform	He Uniform	Glorot Uniform	Lecun Uniform	
Focal Loss	None	c=1.01	c=1.03	c=1.05	
Weights Decay	None	0.0005	0.001	0.005	0.01
Weight Decay Penalty	None	L1	L2		

Tabela 4.6: Parâmetros e respetivos valores utilizados nos testes à inialização, regularização e balanceamento dos pesos

Para a inicialização dos pesos, decidiu-se testar a inialação *Random Uniform*, *Glorot Uniform*[15], *LeCun Uniform* [37] e a *He Uniform*[21]. Ainda relativo aos pesos, foi decidido testar sem *decay* e com diferentes valores de *decay*, sendo que podem penalizar a soma absoluta dos valores dos pesos (*L1 Regularizer* ou sobre a soma quadrática dos valores dos pesos (*L2 Regularizer*). No balanceamento das classes, decidiu-se primeiro testar sem *Focal Loss*, utilizando a função de *loss Categorical Cross Entropy* e, posteriormente testar com *Focal Loss* para diferentes valores de C. As combinações dos diferentes valores de parâmetros consiste num conjunto total de 240 testes.

Efetuado o conjunto total de testes (Apêndice C), os parâmetros do algoritmo com melhor desempenho pode ser visto na tabela 4.7.

Parâmetros			
Weight Initialization	Focal Classes	Weights Decay	Weights Penalty
He Uniform	None	0.0005	L1

Tabela 4.7: Parâmetros do algoritmo com melhores resultados nos testes à inicialização, regularização e balanceamento dos pesos

O teste com melhor resultado utilizou uma inicialização de pesos *He Uniform* em vez de *Glorot Uniform*. A estratégia de focalização das classes não resultou num melhor desempenho. Admite-se que estes resultados sejam explicados por haver neste algoritmo maior balanceamento da presença das classes comparativamente com outros algoritmos que contêm uma grande componente visual — *CUTIE* e *Chargrid*. Para além disso, de forma a prevenir o *overfitting*, este teste utilizou um *decay* dos pesos de 0.005 sobre a soma absoluta dos pesos (L1).

Na Tabela 4.8, estão disponibilizados os resultados deste teste.

A análise que se pode fazer a estes resultados não difere muito das duas análises anteriores. Continua a haver uma discrepância entre os **dados do emissor e do recetor** — que será discutida no Capítulo 5 - Análise e Discussão —. Tal como na melhor configuração do conjunto de testes anteriores, o algoritmo consegue ter uma previsão relativamente boa nas classes das ***Labels* da Moeda e Morada do Emissor**, tendo um desempenho respetivo de 80.91% e 78,57%. Observa-se um desempenho menor na previsão dos **Valores da Moeda** e da **Morada do Emissor**, que assumem valores de, respetivamente, 39.56% e 52.15% —

Desempenho da melhor configuração dos Pesos e Balanceamento das Classes				
Label Data	**Valor Data**	***Label* Nº Fatura**	**Valor Nº Fatura**	**Nome Emissor**
68.27%	55.06%	63.70 %	60.87%	60.82 %
Label Morada Emissor	**Valor Morada Emissor**	**Label Morada Recetor**	**Valor Morada Recetor**	**Nome Recetor**
78.57%	52.15%	60.00%	64.72%	75.65%
Label Total	**Valor Total**	**Label Moeda**	**Valor Moeda**	**Desempenho Geral**
64.33%	60.40%	80.91%	39.56%	62.21%

Tabela 4.8: Resultados detalhados do algoritmo com os parâmetros da Tabela 4.7, com recurso à métrica F1-Score

Curiosamente, valores correspondentes às *labels* com melhor desempenho. Os resultados do desempenho deste teste, quando comparados com o teste dos valores padrão, evidenciam uma melhoria ligeira do desempenho geral, sendo respetivamente 62.21% e 59.51%. Salienta-se também uma melhoria considerável nos resultados observados nas classes das moradas.

De seguida, irão ser procedidos os testes de optimização ao algoritmo *CUTIE*.

4.1.2 *CUTIE*

De forma análoga ao que foi efetuado para o algoritmo *CloudScan*, também procedemos à aglomeração dos valores dos hiper parâmetros disponibilizados no artigo de investigação do método *CUTIE*[74], assim como a seleção de valores padrão para os parâmetros que se encontram em falta.

Na Tabela 4.9, podemos visualizar os valores que irão ser considerados como valores padrão.

Parâmetros	Disponibilizado pelo artigo	Valores
Optimizer	✓	Adam
Dropout Rate	✓	0.1
Batch Size	✓	32
Learning Rate	✓	0.001
Learning Rate Decay	✓	10% do LR anterior - 150 epochs
Embbeding Size	✓	128
Weights Decay		None
Weights Initilization		Glorot Uniform
Class Balancing		None

Tabela 4.9: Parâmetros disponibilizados no artigo do algoritmo CUTIE

Como é possível visualizar, dos hiper parâmetros disponibilizados no artigo, temos o *Optimizer* Adam (no qual, considerámos os valores padrão do *Beta-1* e *Beta-2* da biblioteca *Tensorflow Keras*), um *Dropout Rate* de 10%, um tamanho de *mini-batch* de 32 dados, um *Learning Rate* de 0.001 com um decay para 10% do valor anterior a cada 150 *epochs* e um tamanho de *embbeding* igual a 128. Nos parâmetros que não foram disponibilizados, foram

utilizados como valores padrão, os valores padrão da biblioteca *Tensorflow Keras*.

Na Tabela 4.10, podemos visualizar os resultados obtidos com estes valores padrão.

Desempenho do algoritmo *CUTIE* com os hiper parâmetros padrão				
Label Data	**Valor Data**	***Label* Nº Fatura**	**Valor Nº Fatura**	**Nome Emissor**
0.00%	0.00%	0.00 %	0.00%	0.00%
Label Morada Emissor	**Valor Morada Emissor**	**Label Morada Recetor**	**Valor Morada Recetor**	**Nome Recetor**
0.00%	0.00%	0.00%	0.00%	0.00%
Label Total	**Valor Total**	**Label Moeda**	**Valor Moeda**	**Desempenho Geral**
0.00%	0.00%	0.00%	0.00%	0.00%

Tabela 4.10: Resultados do algoritmo CUTIE com os parâmetros padrão, com recurso à métrica F1-Score

Verifica-se que utilizando apenas os valores padrão, os resultados são muito negativos, indiciando que a rede neuronal não consegue aprender a prever em documentos novos, identificando todas as palavras como desconhecidas. Essa má identificação fez com que todas as classes tivessem um resultado de F1-Score de 0.00%. A primeira hipótese considerada foi a falta de balanceamento dos dados. Como a maioria das palavras não são categorizadas nas classes alvo, pertencendo à classe ***Unknown/Background***, na fase de treino, a rede neuronal conseguia obter um boa *accuracy*, classificando todas as palavras como pertencentes a essa classe.

Sendo assim, o primeiro teste a ser efetuado é o teste ao balanceamento das classes.

Balanceamento das Classes

De forma análoga ao balanceamento das classes utilizado no método *CloudScan*, foi utilizado a fórmula de *Focal Loss*[40] para balancear as diferentes classes. Nos testes que se seguem, foi efetuado um teste sem balanceamento, seguido de diversos testes utilizando *Focal Loss*, com o coeficiente de c a variar entre 1.00 e 1.05, com variação de 0.01, num conjunto total de 6 testes.

Na Tabela 4.11, podemos visualizar o resultado do desempenho geral do algoritmo com os diferentes valores.

Class Balancing						
None	c = 1.00	c = 1.01	c = 1.02	c = 1.03	c = 1.04	c = 1.05
0.00%	74.97%	82.13%	81.86%	0.00%	0.00%	81.38%

Tabela 4.11: Resultados globais com métrica F1-Score utilizando diferentes valores de foco de classes no algoritmo CUTIE

Da análise que se obtém a partir dos valores da Tabela 4.11, é possível concluir que um dos problemas dos fracos resultados que se obtiveram com os valores padrão, estava no não balanceamento das classes. Na Tabela 4.11, consegue-se observar que quando o o valor c

assume resultados mais baixos (≤ 1.02) — o que significa uma maior penalização para o erro em classes de menor representação —, o algoritmo obtém melhor desempenho. Para além disso, há várias configurações (sem *Focal Loss*, ou com coeficientes de c igual a 1.03 e 1.04) em que o algoritmo não conseguiu prever corretamente nenhuma das classes alvo, identificando todos os elementos como desconhecidos. Em suma, o algoritmo *CUTIE* obtém um melhor desempenho quando este utiliza *Focal Loss* com um valor de c igual a 1.01.

Na Tabela 4.12, podemos visualizar os resultados do desempenho do algoritmo nas diferentes classes com *Focal Loss* e c igual a 1.02.

Desempenho da melhor configuração do Balanceamento das Classes				
Label Data	**Valor Data**	***Label* Nº Fatura**	**Valor Nº Fatura**	**Nome Emissor**
81.80%	71.30%	83.17%	78.54%	75.61%
Label Morada Emissor	**Valor Morada Emissor**	**Label Morada Recetor**	**Valor Morada Recetor**	**Nome Recetor**
72.73%	79.06%	27.85%	92.00%	89.91%
Label Total	**Valor Total**	**Label Moeda**	**Valor Moeda**	**Desempenho Geral**
74.53%	64.46%	87.57%	75.41%	82.13%

Tabela 4.12: Resultados do algoritmo CUTIE com o *Focal Loss* e valor de c = 1.01

Ao analisar a Tabela 4.12, verifica-se que o algoritmo tem um desempenho muito eficiente na previsão do **Nome e Morada do Recetor**, assumindo valores de 89.91% e 92.00%, respetivamente. No entanto, observa-se um desempenho muito baixo na ***Label* da Morada**, com um resultado de 27.85%. Há algum equilíbrio entre os valores e respetivas *labels*, ainda assim, a previsão das *labels* tende a assume resultados ligeiramente superiores aos respetivos valores — A exceção ocorre nos dados da morada. Em suma, o algoritmo evidencia um bom desempenho geral, com um resultado de 82.13%.

De seguida, apresentam-se os testes à influência da inicialização dos pesos no desempenho do CUTIE. Devido à influência que o *Focal Loss* tem no desempenho do algoritmo, foi decidido que, ao contrário do que foi realizado nos testes de optimização ao algoritmo *CloudScan*, os valores óptimos do conjunto de testes realizados anteriormente irão servir como base nos testes consequentes. Caso não se considerasse esta opção, o desempenho nos novos testes seria muito fraco, o que poderia condicionar a sua análise. Nos parâmetros que não foram testados anteriormente, serão considerados os valores padrão.

Inicialização de pesos

No conjunto de testes efetuado às diferentes inicializações de pesos, foram considerados os seguintes métodos: *Random Normal* e *Uniform*, *Truncated Normal*, *LeCun Normal* e *Uniform*, *Glorot Normal* e *Uniform* e *He Normal* e *Uniform*. Os resultados gerais destes testes podem ser visualizados na Tabela 4.13.

Na Tabela 4.13, conseguimos ver a influência que a inicialização dos pesos têm no desempenho geral do algoritmo, havendo muita discrepância nos resultados entre os métodos mais básicos e os métodos de *LeCun*[37], *Glorot*[15] e *He*[21]. O método de inicialização com

Parâmetros				
Random Normal	Random Uniform	Truncated Normal	LeCun Normal	LeCun Uniform
0.00%	50.39%	2.11%	81.28%	77.51%
Glorot Normal	Glorot Uniform	He Normal	He Uniform	
79.08%	81.35%	78.77%	75.31%	

Tabela 4.13: Resultados globais com métrica F1-Score dos métodos de inicialização de pesos no algoritmo CUTIE

melhor desepenho, acabou por ser o *Glorot Uniform*, o método que também já tinhamos selecionado como padrão e também ele usado como padrão na biblioteca *Tensorflow Keras*.

Na Tabela 4.14, podemos visualizar os resultados mais detalhados do teste ao CUTIE com este método de inicialização.

Desempenho da melhor configuração da Inicialização dos Pesos				
Label Data	**Valor Data**	***Label* Nº Fatura**	**Valor Nº Fatura**	**Nome Emissor**
84.68%	73.33%	83.44%	80.79%	74.62%
Label Morada Emissor	**Valor Morada Emissor**	**Label Morada Recetor**	**Valor Morada Recetor**	**Nome Recetor**
73.24%	77.58%	0.00%	92.23%	89.66%
Label Total	**Valor Total**	**Label Moeda**	**Valor Moeda**	**Desempenho Geral**
76.05%	66.67%	0.00%	80.52%	81.35%

Tabela 4.14: Resultados do algoritmo CUTIE com o método de inicialização de pesos *Glorot Uniform*

Analisando os resultados da Tabela 4.14, é possível identificar algumas diferenças entre estes resultados e os resultados da Tabela 4.12. Assumindo que o hiper parâmetro em teste é igual ao hiper parâmetro do teste anterior, não era espectável assistir a uma grande discrepância entre os resultados — devido à aleatoriedade conferida pelos métodos de inicialização de pesos, é normal que haja alguma variação, embora não muito significativa. De uma forma geral, os resultados são muito idênticos ao teste anterior, assumindo um desempenho muito eficiente na previsão das classes do **Nome e Morada do Recetor**, com valores de, respetivamente, 89.66% e 92.23%. A grande distinção relativamente aos resultados anteriores, é a falha total na previsão das classes da ***Label* da Moeda** e da **Morada do Recetor**. Esta falha total na previsão fez influenciar a previsão geral, diminuindo o desempenho para valores de 81.35%.

De seguida, irão ser efetuados os testes ao decay dos pesos.

Decay dos Pesos

Nos testes efetuado ao *decay* dos pesos, foram consideradas três abordagens principais, uma primeira sem qualquer tipo de *decay*, uma segunda em que os pesos são penalizados pelo seu valor absoluto (*L1 Regularization*) e uma última em que os pesos são penalizados pelo seu valor quadrático (*L2 Regularization*).

Na Tabela 4.15, podemos visualizar os valores assumidos pelo *decay* dos pesos nos testes efetuados e respetivos resultados gerais.

Valor do *Decay*								
None	**L1**				**L2**			
—	0.0001	0.0005	0.001	0.005	0.0001	0.0005	0.001	0.005
81.30%	76.53%	73.09%	74.80%	68.04%	78.48%	79.14%	77.60%	75.30%

Tabela 4.15: Resultados globais com métrica F1-Score dos métodos de inicialização de pesos no algoritmo CUTIE

Dos testes efetuados, podemos visualizar que o algoritmo tem um melhor desempenho quando não há *decay* dos pesos ou quando o *decay* é pequeno. Comparando as duas abordagens onde ocorre o *decay* dos pesos, é possível também visualizar, que independentemente do valor, o desempenho é superior quando os pesos são penalizados pelo seu valor quadrático (*L2*). Com este tipo de penalização, os pesos com um valor absoluto superior são mais penalizados comparativamente com os pequenos que assumem valores menores.

Na Tabela 4.16, é possível visualizarmos os resultados mais detalhados do teste efetuado ao algoritmo *CUTIE* sem *decay* de pesos.

Desempenho da melhor configuração do *Decay* de Pesos				
Label Data	**Valor Data**	***Label* Nº Fatura**	**Valor Nº Fatura**	**Nome Emissor**
83.99%	70.21%	0.00%	76.25%	74.37%
Label Morada Emissor	**Valor Morada Emissor**	**Label Morada Recetor**	**Valor Morada Recetor**	**Nome Recetor**
76.92%	77.97%	34.57%	91.55%	88.96%
Label Total	**Valor Total**	**Label Moeda**	**Valor Moeda**	**Desempenho Geral**
72.73%	68.56%	88.17%	78.96%	81.30%

Tabela 4.16: Resultados do algoritmo CUTIE sem o decay de pesos (= *None*)

No seguimento da análise dos conjuntos de teste anteriores, visto que, curiosamente, os hiper parâmetros são iguais nas melhores configurações dos diferentes conjuntos de teste, a análise que pode fazer também é idêntica. O algoritmo *CUTIE* com melhor configuração consegue obter melhor desempenho na previsão das classes do **Nome e Morada do Recetor**, obtendo valores de, respetivamente, 88.96% e 91.55%. Em sentido inverso, este teve um desempenho menor nas classes da ***Label* do Nº de Fatura** — cuja previsão nunca foi efetuada de forma correta — e da **Morada do Recetor**, com desempenho de 34.57%. Regra geral, não há uma diferença muito discrepante entre os valores da previsão das *Labels* e respetivos Valores, sendo que o algoritmo com esta configuração, neste conjunto de testes obteve um desempenho de 81.30%.

4.2 Testes de Desempenho Finais

Apresentam-se nesta secção, os testes de desempenho realizados aos algoritmos implementados com as melhores versões de cada um. Os resultados destes testes serão ainda comparados com o desempenho de outros sistemas, nomeadamente o *Azure Form Recognizer* e o serviço de extração de dados de faturas da plataforma IDV.

Como foi explicado na Secção 3.5.1, os testes que serão efetuados têm como base o mesmo conjunto de dados para treino, validação e teste, em todos os algoritmos. Serão nestes testes finais, os algoritmos *CloudScan*, o *CUTIE* e a suas versões com a integração do modelo de linguagem *BERT*, o *Chargrid* e a suas versões adaptadas com *BERT*, BERTgrid. Considera-se ainda uma combinação dos métodos de *Chargrid* e *BERTgrid*, cujo o classificador integra ambas as representações, sendo que no final do primeiro bloco de *layers* ocorre um *merge* de ambas — mais propriamente, uma adição dos valores das representações.

Na Tabela 4.17, podemos visualizar o desempenho de todos os algoritmos para cada uma das classes que se pretendem extrair.

Modelos	Data *Label*	Data Valor	Nº Fatura *Label*	Nº Fatura Valor	Nome Emissor	Morada Emissor *Label*	Morada Emissor Valor
CloudScan	79.60%	60.56%	74.46%	69.80%	68.87%	78.83%	46.89%
CUTIE	81.09%	72.42%	82.76%	76.89%	75.17%	0.00%	79.83%
CUTIE + BERT ML	83.22%	74.94%	81.55%	76.23%	75.51%	0.00%	91.58%
CUTIE + BERT PT	83.62%	77.45%	78.17%	80.68%	74.81%	**79.41%**	80.12%
Chargrid	88.72%	84.89%	86.90%	91.70%	76.18%	73.81%	78.53%
BERTGrid ML	87.45%	79.91%	86.53%	91.68%	76.74%	67.48%	80.97%
BERTGrid PT	88.80%	83.55%	**90.19%**	85.06%	76.05%	72.96%	81.26%
Char + BERT ML	**90.52%**	82.98%	87.01%	**93.28%**	78.63%	73.17%	79.55%
Char + BERT PT	88.48%	**85.31%**	85.81%	92.47%	**79.28%**	67.03%	**82.53%**
Modelos	**Nome Recetor**	**Morada Recetor *Label***	**Morada Recetor Valor**	**Total *Label***	**Total Valor**	**Moeda *Label***	**Moeda Valor**
CloudScan	83.72%	31.46%	54.18%	69.75%	60.65%	87.72%	58.82%
CUTIE	88.76%	27.78%	91.55%	74.20%	61.45%	87.43%	76.72%
CUTIE + BERT ML	89.41%	32.43%	91.58%	72.01%	61.18%	0.00%	73.43%
CUTIE + BERT PT	89.33%	22.22%	91.65%	74.63%	61.38%	85.11%	72.79%
Chargrid	89.79%	31.82%	93.30%	75.12%	81.28%	87.83%	80.86%
BERTGrid ML	**90.32%**	39.13%	**94.41%**	78.29%	**82.54%**	87.57%	79.22%
BERTGrid PT	89.06%	27.78%	94.12%	75.84%	81.40%	87.86%	79.17%
Char + BERT ML	89.91%	34.04%	93.31%	**80.06%**	81.41%	**89.66%**	84.72%
Char + BERT PT	**90.32%**	**67.03%**	94.01%	78.12%	83.77%	88.51%	**84.73%**

Tabela 4.17: Resultados finais do desempenho dos algoritmos para cada uma das classes

Visualizando os resultados na Tabela 4.17, depara-se que há 3 níveis de desempenho geral. O algoritmo *CloudScan* tem um desempenho inferior, comparativamente aos demais. na maioria das classes analisadas. O algoritmo *CUTIE* e suas variações obtiveram um bom desempenho na generalidade das classes, mas ainda assim, inferior aos algoritmos *Chargrid* e *BERTgrid*. A junção das representações do *Chargrid* e *BERTgrid*, na maioria das classes, obtêm o melhor desempenho da sua previsão. As classes que obtiveram melhor desempenho, de um modo geral, foram as classes relativas aos **dados do recetor**, enquanto as classes que obtiveram menor desempenho foram as classes do **nome do emissor e *labels* de ambas as moradas**.

A Tabela 4.18 permite visualizar o desempenho global de cada um dos algoritmos no teste anterior. Nesta Tabela, tanto numa análise global, como nas análises das *labels* e dos valores,

foram considerados dois tipos de médias, a global e a das classes. Na média das classes, é calculado o desempenho de cada classe individualmente e é efetuada a média desses valores. Com esta variável consegue-se identificar melhor o desempenho dos algoritmos na previsão das classes, independentemente do número de palavras de cada classe. Por outro lado, a média global calcula o desempenho geral dos algoritmos ao aglomerar todos os verdadeiros positivos, falsos positivos e falsos negativos. Desta forma, conseguimos ter uma noção mais real do desempenho dos algoritmos, considerando-se no entanto, que esta variável é mais influenciada pelas classes com maior número de palavras.

Modelos	Média Global	Média das Classes	Média Global *Labels*	Média Global Valores	Média Classes *Labels*	Média Classes Valores
CloudScan	64.30%	66.72%	73.96%	61.33%	71.07%	63.37%
CUTIE	82.13%	76.87%	79.25%	82.72%	74.24%	78.82%
CUTIE + BERT ML	80.67%	64.29%	69.76%	82.68%	45.90%	77.94%
CUTIE + BERT PT	81.88%	75.82%	77.12%	82.82%	71.96%	78.58%
Chargrid	84.11%	80.38%	80.85%	84.79%	74.65%	84.64%
BERTGrid ML	85.00%	80.40%	81.01%	85.83%	74.65%	84.45%
BERTGrid PT	84.86%	80.39%	80.74%	85.70%	74.77%	84.42%
Char + BERT ML	85.12%	81.54%	**82.77%**	85.63%	**76.12%**	85.59%
Char + BERT PT	**85.89%**	**81.78%**	80.99%	**86.94%**	75.35%	**86.60%**

Tabela 4.18: Resultados finais do desempenho geral dos algoritmos

Da análise da Tabela, conseguimos visualizar que, exceptuando no caso do *CloudScan*, o desempenho global é superior comparativamente ao desempenho médio das classes. Isto é consequência do facto de a maioria dos algoritmos obter um desempenho superior nas classes mais textuais, que contém um maior número de palavras. Se analisarmos separadamente o desempenho das *labels* e dos valores, verificamos que regra geral o desempenho é superior nos valores. Efetuando agora a análise dos algoritmos, é possível verificar que nos testes realizados, o modelo *BERT* não tem grande influência no desempenho do algoritmo *CUTIE*. Por outro lado, no caso do *Chargrid*, o desempenho foi superior aquando da utilização do modelo *BERT*, atingindo o melhor desempenho do conjunto de testes na junção dos algoritmos do *Chargrid* e *BERTgrid*.

De seguida, apresentam-se os resultados da comparação do algoritmo que obteve melhor resultado nos testes efetuados anteriormente, o *Chargrid + BERTgrid*, com dois sistemas externos. O primeiro sistema analisado é o *Azure Form Recognizer*, já apresentado na Secção 2.7. O segundo sistema utilizado na análise comparativa, é o extractor/processador de faturas que se encontra implementado na plataforma IDV.

Como foi referido anteriormente, o sistema *Azure Form Recognizer* ainda está em versão *Beta* e está otimizado para recibos, podendo contudo também prever faturas. Este contém oito entidades pré-definidas, sendo elas: **Sub-total**, **Taxa/Imposto**, **Total**, **Nome do Vendedor/Emissor**, **Morada do Vendedor/Emissor**, **Número de telefone do Vendedor/Emissor**, **Data da Fatura** e **Hora da Fatura**. Na comparação dos resultados deste sistema com o algoritmo *Chargrid + BERTgrid*, teve-se em consideração apenas as classes

utilizadas na validação dos algoritmos implementados. Desta forma, apenas iremos considerar as classes **Nome do Vendedor/Emissor**, **Morada do Vendedor/Emissor** e **Data da Emissão da Fatura**.

O extractor/processador de faturas implementado no IDV é composto por 5 entidades pré-definidas, nomeadamente: **Morada do Recetor**, **Data de Emissão**, **Total da Fatura** e **VAT do Emissor e Recetor**. Este extractor encontra-se numa versão descontinuada, pelo que não foi possível obter o acesso aos dasdos da classe **Morada do Recetor**. Sendo assim, serão consideradas na análise comparativa apenas as classes **Data de Emissão da Fatura** e **Total da Fatura**.

Dadas as características do sistema *Azure Form Recognizer* para compreensão de recibos — versão pré-treinada — e do sistema de extração e compreensão de dados de faturas implementado na plataforma *IDV* — Sistemas Baseados em Regras (SBR) —, não é necessário treinar previamente os sistemas externos utilizados. Nesse sentido, apenas foi considerado o conjunto de dados de teste utilizado nos testes anteriores.

Optou-se por não incluir os erros de Reconhecimento de Caracteres Óptico (RCO) na análise de desempenho dos algoritmos externos, uma vez que estes também não foram considerados nas análises anteriores. Para ultrapassar a não inclusão dos erros de RCO, realizou-se o teste de desempenho dos sistemas externos de forma manual, evitando a contabilização de erros originados numa má leitura textual do documento. Na Tabela 4.19, podemos visualizar os resultados destes testes.

Modelo	**Nome Emissor**			**Valor Data**			**Valor Total**		
	Precisão	*Recall*	*F1-Score*	Precisão	*Recall*	*F1-Score*	Precisão	*Recall*	*F1-Score*
Azure Form Recognizer	50.82%	12.30%	19.81%	78.38%	31.18%	44.62%	69.70%	27.38%	39.32%
IDV Invoice Extractor	—	—	—	90.48%	41.76%	57.14%	73.65%	67.58%	70.49%
Chargrid + BERTgrid PT	81.29%	77.36%	79.28%	84.13%	86.53%	85.31%	83.77%	83.77%	83.77%

Tabela 4.19: Comparação do desempenho do *Chargrid + BERTgrid* PT com algoritmos externos

A análise dos resultados da Tabela 4.19, permite salientar que o desempenho do algoritmo *Chargrid + Bertgrid* supera os resultados observados noutros sistemas em todos os campos considerados.

O *Azure Form Recognizer* evidencia nestes testes um desempenho muito baixo em todas as classes analisadas, sendo que na maioria das vezes não consegue detetar o campo ou deteta-o com frequentes incorreções.

O extractor de dados de faturas do IDV, evidencia muita dificuldade em identificar o **campo da Data**, com um *Recall* de 41.76%, mas quando o deteta, fá-lo na maioria das vezes de forma correta, obtendo uma precisão de 90.48% — Tem até uma maior precisão do que o *Chargrid + BERTgrid*, os resultados evidenciam que obtém uma precisão de 84.13% —. Já relativamente ao **campo Total**, os resultados indicam uma capacidade de previsão mais frequente, com um *Recall* de 67.58%, contudo a precisão da previsão deste campo é

menor, com um resultado de 73.65%.

Capítulo 5

Análise e Discussão

Neste capítulo apresenta-se a discussão dos resultados obtidos nos testes, procurando salientar os pontos fortes e fragilidades dos algoritmos implementados.

A implementação dos algoritmos foi validada num conjunto de dados constituído por 1210 faturas que são remetidas à CSW. Este conjunto de dados apresenta algumas características que lhes conferem vantagens, nomeadamente, a variedade de estruturas e formatos de apresentação da informação, permitindo testar a versatilidade dos algoritmos analisados. Por outro lado, o facto de estes documentos serem remetidos sempre à mesma entidade, reduz a variação de valores assumidos nos campos do recetor, o que pode influenciar os resultados da previsão nestes campos.

A análise global de todos os testes efetuados permite salientar que as classes com menor presença no conjunto de dados — **Morada Emissor Label** e **Morada Recetor Label** — são as classes que obtiveram pior desempenho na sua previsão. Verifica-se que a maioria dos documentos não expressam claramente estas *Labels*, dificultando o trabalho de previsão dos algoritmos. É expectável que num conjunto de dados mais alargado, em que estas classes tenham maior expressão, os resultados sejam diferentes, aumentando consideravelmente o desempenho da sua previsão.

Por outro lado, consegue-se identificar uma clara discrepância na capacidade de previsão entre os dados do recetor e os dados do emissor. Se por um lado é bom termos um conjunto de dados que são remetidos à CSW, em vez de serem faturas que a própria CSW envia — pela variedade de estruturas de faturas que a CSW recebe —, para o trabalho realizado, seria ideal que o conjunto de dados incluísse faturas que são remetidas e enviadas por entidades diferentes.

Verica-se também que em campos que aparecem múltiplas vezes na fatura, por exemplo o **Valor da Moeda**, os algoritmos por vezes prevêm corretamente o seu valor, no entanto, é classificado como errado. Dando o exemplo da moeda, esta é uma classe que aparece múltiplas vezes. Apesar de na anotação, se ter tido cuidado de garantir alguma coerência, os algoritmos por vezes detetam outros campos da moeda, igualmente certos na prática, mas que o analisador detetou como errado porque não terem sido os campos anotados. Por exemplo, no caso dos campos da moeda, quando há campos de *label*/valor, a anotação considera esses

campos. No caso em que a *label* esteja omissa, na anotação da moeda, prevalece o símbolo ou nome extensivo mais próximo dos campo do total da fatura. Esta situação permite admitir que os algoritmos possam ser ainda mais fiáveis do que os resultados demonstram.

Reconhecendo algumas limitações e características do conjunto de dados analisados, admitimos que uma amostra maior permitisse evidenciar um desempenho superior dos classificadores. Para se ter uma noção, os autores dos artigos *CloudScan*[52], *CUTIE*[74] e *Chargrid*[33] usaram um conjunto de dados total, respetivamente, de 326471, 4484 e 12000, valores muito superiores às 1210 faturas utilizadas no nosso conjunto de dados. Dada a morosidade do processo de anotação das faturas, e o tempo limitado que se tinha para a concretização de todos os objetivos de estágio, considerou-se que, não sendo um tamanho ideal da amostra, a dimensão permitia analisar com alguma confiança os algoritmos implementados.

Era intenção deste estágio, ter a oportunidade de analisar os algoritmos com outros conjuntos de dados de diferentes tipos de documentos, de forma a poder fazer uma análise da capacidade de generalização de cada um dos algoritmos. Infelizmente, o confinamento imposto durante este estágio, levantou algumas burocracias e dificultou o acesso a esses dados que, pertencendo a clientes da CSW, são muito sensíveis e confidenciais. Ainda assim, na fase final de estágio, o acesso a outro conjunto de dados composto por cerca de 400 passaportes foi possibilitado por parte da empresa, contudo, embora se reconheça o interesse em testar com vários conjuntos de documentos para efetuar a análise da versatilidade dos algoritmos, dada a escassez do tempo, optou-se por não a fazer neste momento. Ainda assim, mantém-se o interesse em analisar os algoritmos com outros conjuntos de dados num trabalho futuro.

Redirecionando agora a discussão para os algoritmos, inicia-se pelo algoritmo *CloudScan*. Este algoritmo de extração e compreensão de documentos é o mais simples — contém 9 milhões de parâmetros —, pelo que era expectável que tivesse um desempenho inferior aos demais. Contudo, como este algoritmo utiliza como classificador um modelo LSTM, frequentemente utilizado quando se pretende lidar com elementos textuais, esperava-se que nas classes mais textuais, como é o caso das **Moradas do Recetor e Emissor** das faturas, apresentasse um desempenho minimamente aceitável ou pelo menos idêntico às restantes classes. Não foi isso que se observou, vericando-se que nestas classes o desempenho foi significativamente inferior ao desempenho geral, assumindo nas referidas classes um desempenho de 46.89% e 54.18%, quando o desempenho geral é de 64.30%. Este decréscimo é ainda mais surpreendente, quando se compara com os resultados obtidos nestas classes pelos restantes algoritmos implementados, onde inclusivé se verifica um acréscimo de desempenho. O *CloudScan* já tinha sido comparado pelos autores dos artigos de investigação dos métodos *CUTIE*[74] e *Chargrid*[33], como foi referido na Secção 2.5. No artigo de *CUTIE*, os resultados obtidos por este algoritmo no desempenho geral são semelhantes ao que obtivémos nos nossos testes. Relativamente aos resultados evidenciados no artigo do *Chargrid*, estes não são comparáveis diretamente com os resultados deste estudo, uma vez que a métrica utilizada é

diferente. Contudo, também nesse artigo se verificou um decréscimo muito significativo no desempenho do algoritmo nas classes mais textuais, tendo-se observado nas classes do **Nome e Morada do Vendedor** um desempenho de 16.94% e 28.97%, respetivamente, enquanto nas classes do **Nº da Fatura**, **Quantia** e **Data** se obtiveram, respetivamente, os resultados de 80.98%, 79.13% e 83.98%.

Teria sido interessante verificar se a inclusão do modelo de linguagem *BERT* influencia o desempenho do algoritmo na previsão, no entanto, o tempo foi um factor limitante que fez excluir essa opção nesta fase. Ainda assim, poderá ser considerado num trabalho futuro.

Relativamente ao algoritmo *CUTIE*, de acordo com os autores que o propuseram[74], apesar de se tratar de um sistema pouco complexo, composto por cerca de 14 milhões de parâmetros, no entanto, consegue atingir resultados ao nível do estado da arte na extração e análise de dados de documentos. Nos testes que foram efetuados a este algoritmo, identificou-se a particularidade de começar a fazer *overfitting* dos dados de treino demasiado cedo, comparativamente aos restantes algoritmos (nos primeiros 30 *epochs*), apesar das diversas tentativas de o tentar reduzir, nomeadamente com a aplicação de *Dropout* e de estratégias de regularização dos pesos. O desempenho geral que o método *CUTIE* conseguiu obter no conjunto de testes efetuados, é bastante bom, atingindo valores de 82%. Ainda assim, nos testes efetuados, em alguns casos o algoritmo não conseguiu prever corretamente algumas classes, obtendo desempenhos nulos nessas previsões. Se a falha de previsão ocorresse sempre nas mesmas classes, poderia ser colmatado com uma focalização das classes mais agressiva, no entanto, a falha da previsão das classes não foi padronizada, podendo acontecer em várias classes. Não existindo uma explicação óbvia para este acontecimento, acredito que, no decorrer dos testes, o *overfitting* que ocorre no modelo esteja a influenciar a falha de previsão, visto que o algoritmo não fica tão bem treinado, comparativamente aos seus concorrentes. Num trabalho futuro, poderão ser implementadas novas propostas de combate ao *overfitting*, nomeadamente propor diferentes estratégias de *Data Augmentation* das representações e efetuar testes de optimização mais intensivos.

Para além disso, considerei que a implementação do método *CUTIE*, juntamente com o modelo de linguagem *BERT*, fosse melhorar o desempenho deste método, no entanto, como é demonstrado na Tabelas 4.17 e 4.18, isso não não se verificou, inclusivé tiveram um pior desempenho geral. Contudo, esta variação permitiu aumentar o desempenho nas classes mais textuais, como é o caso das moradas, com a inclusão do modelo de linguagem referido.

Relativamente ao algoritmo *Chargrid*, trata-se do algoritmo mais complexo, com cerca de 23 milhões de parâmetros. Este algoritmo preserva a estrutura visual do documento e os resultados obtidos no seu desempenho, quando comparados com os resultados dos algoritmos anteriores, demonstram ser mais eficientes e equilibrados nas diferentes classes. A inclusão do modelo *BERT* no *Chargrid*, denominado por *BERTgrid*[7], permitiu melhorar ligeiramente os resultados, com destaque para o aumento do desempenho em classes mais textuais, como

seria de esperar. No entanto, foi a junção das representações do *Chargrid* com o modelo *BERTgrid* que permitiu obter o melhor desempenho de todos os algoritmos considerados nos testes, superando o desempenho dos restantes em praticamente todas as classes.

Relativamente ao modelo de linguagem *BERT*, foram consideradas duas versões, uma treinada com palavras de múltiplas línguas[1] e outra com palavras em português do Brasil[2]. Neste estudo, o *BERT* em português foi o modelo de linguagem que permitiu obter melhor desempenho, não por ser melhor do que o múltiplas linguas, mas porque o nosso conjunto de dados está maioritariamente em português. Ainda assim, este modelo treinado em português não está optimizado para o nosso cenário. Apesar de conter muitas palavras portuguesas no seu vocabulário, não contém algumas palavras que neste contexto têm uma preponderância muito grande, como é o caso das palavras "Fatura", "Sub-total" ou "Moeda". Inicialmente, foi considerado no desenvolvimento deste trabalho, treinar um modelo *BERT* com linguagem técnica associada a faturas, no entanto, optámos por não o fazer devido a serem necessárias muitas faturas e muito tempo dedicado para o seu treino. Os autores do artigo do *BERTgrid*[7] também treinaram um modelo *BERT* com linguagem técnica de faturas, sendo que para isso foram utilizadas cerca de 800.000 faturas e de acordo com a *Google*, detentora deste modelo, com uma boa gráfica dedicada, são necessários 10 a 18 dias para o treino de um modelo de linguagem *BERT*. Estes argumentos condicionaram a nossa decisão de não treinar o *BERT* com linguagem técnica de faturas, considerando no entanto, que num trabalho futuro, seria interessante analisar o desempenho dos algoritmos com esta versão do modelo.

Em síntese, as análises efetuadas aos algoritmos, permitem considerar que o desempenho do *Chargrid* se revelou melhor do que os desempenhos observados no *CloudScan* e *CUTIE*. Considera-se que o *Chargrid* é mais versátil e apresenta maior capacidade de adaptabilidade a diferentes documentos estruturados pela preservação da estrutura visual dos documentos. Ainda que não se tenha tido oportunidade de analisar o desempenho em documentos com outras estruturas, admite-se que estas características permitem melhor adaptação a diferentes documentos estruturados. Se nestes argumentos, também incorporarmos o modelo de linguagem *BERT*, admite-se que com o contexto textual potenciado, melhorará o desempenho nas tarefas de extração e comprensão de dados de documentos estruturados. No caso do *CUTIE*, a análise efetuada, permite esperar que em documentos mais fixos, onde haja normas, como é o caso dos passaportes e cartões de cidadão, seja possível obter resultados superiores aos obtidos nos testes com documentos de faturas. Esta expetativa é fundamentada pela sua representação da estrutura do documento, que não respeitando as distâncias absolutas, tem em conta a distância relativa entre palavras.

Apesar dos bons desempenhos obtidos na análise efetuada em documentos estruturados, considera-se que nenhum destes algoritmos está otimizado para documentos menos estrutu-

[1] `https://github.com/google-research/bert`

[2] `https://github.com/neuralmind-ai/portuguese-bert`

rados, com uma maior componente textual, como é o caso de artigos ou alguns certificados de nascimento (sobretudo antigos, que ainda estão em manuscripto), pelo que provavelmente os resultados iriam ser bastantes inferiores aos que se apresentaram nos testes realizados. Considera-se que para se obter um serviço genérico utilizável nesse tipo de documentos, será necessário recorrer a outros métodos de extração, como por exemplo utilizar exclusivamente o modelo de linguagem para a função de *Named Entity Recognition*.

Como forma de validar os resultados obtidos nos algoritmos implementados, realizou-se uma análise comparativa do desempenho do melhor algoritmo obtido, o *Chargrid + BERTgrid PT*, com algoritmos externos aos da implementação efetuada neste estágio, como é o caso do *Azure Form Recognizer* e o extrator de faturas da plataforma IDV. Os resultados da análise comparativa, permitiu salientar que o *Chargrid + BERTgrid PT* teve um desempenho claramente superior aos restantes. Reconhece-se que nesta comparação, apenas foi possível considerar a análise comparativa em 2 ou 3 campos compatíveis, o que limitou o estudo realizado. Nesta análise comparativa, podem ser considerados alguns argumentos que justificam o menor desempenho dos algoritmos externos utilizados. O *Form Recognizer* foi inicialmente desenhado para reconhecer vários documentos estruturados, contudo, a versão atual, apesar de também ter sido demonstrada com faturas, está otimizada para extrair campos de recibos e ainda se encontra numa versão *Beta*. Relativamente ao extrator de faturas do IDV, utilizada pela empresa IDV, a sua utilização encontra-se descontinuada e apenas foi possível fazer a comparação de 2 campos. Ainda assim, foi interessante comparar estes modelos, pois permitiu obter uma noção do valor acrescentado do desempenho dos algoritmos implementados face ao que existe disponível neste momento para extração e compreensão de documentos estruturados. Admitindo que esta é uma área em desenvolvimento, reconheço o interesse de futuramente fazer a análise de desempenho dos algoritmos implementados com outros sistemas que estão em produção, nomeadamente o caso do *Google Document Understanding* e o *Amazon Textract*. Neste momento ainda não é possível, visto que os sistemas referidos ainda estão em fase de desenvolvimento e não se encontram disponíveis ao público.

No seguimento das análises aos algoritmos implementados, elegeu-se o modelo *Chargrid + BERTgrid PT* para ser implementado como prova de conceito. Assim, a fase final do estágio foi dedicada à implementação desse modelo na plataforma do IDV na CSW. Na fase atual, estão a ser efetuados testes para a compreensão de dados de faturas, uma vez que a versão implementada foi otimizada para este tipo de documentos. Ainda assim, admite-se a sua adaptabilidade para outro tipo de documentos, embora nesta fase, não esteja a ser considerada a sua utilização na versão de produção.

Capítulo 6

Conclusão

O estágio curricular que decorreu neste ano letivo e do qual resultou este relatório, pode ser dividido em 3 etapas.

A primeira etapa decorreu no primeiro semestre e foi dedicada à pesquisa bibliográfica. Na análise do estado da arte foram explorados e analisados diversos artigos de extração e compreensão de documentos estruturados, permitindo adquirir e consolidar conhecimentos na área da IA, mas sobretudo na área de AC e *Deep Learning*. Nesta etapa, para além do aprofundar de conhecimentos, foram selecionados algoritmos a implementar na etapa seguinte.

A segunda etapa foi dedicada à implementação dos algoritmos *CloudScan*, *CUTIE* e *Chargrid* e respetivas variações. A concretização desta etapa foi particularmente desafiante, exigindo grande esforço conceptual e tempo, na mobilização do conhecimentos adquiridos para a implementação dos diferentes algoritmos. Apesar de se tratarem de algoritmos já publicados, o facto de os ter implementado a partir de informação disponibilizada em artigos, exigiu o desenvolvimento de novas competências uma vez que estes não continham toda a informação necessária para a replicação dos algoritmos e, nem sempre estava expressa da forma mais clara. As lacunas na informação disponível nos artigos, suscitaram enormes desafios, que tiveram que ser colmatados por competências desenvolvidas, sobretudo ao nível da Engenharia e da tomada de decisão. Para além da implementação dos algoritmos inicialmente selecionados, foram implementadas várias variações do algoritmo *CUTIE* e *Chargrid*, que, apesar de não exigirem tanto esforço como uma implementação nova, exigiram um envolvimento considerável, pelo facto das representações com o modelo *BERT* serem de dimensões distintas, obrigando à reformulação de algumas etapas dos algoritmos, e exigindo a integração de novas tecnologias, como foi o caso da utilização do modelo de linguagem *BERT*, que é o estado da arte na área de PLN.

Dedicou-se a terceira etapa à investigação e análise de desempenho dos algoritmos implementados. Nesta etapa foi efetuada a anotação dos dados de faturas e foram realizados os testes de optimização e de desempenho dos vários algoritmos. As diferentes análise efetuadas, permitiram concluir que o algoritmo que obteve melhores resultados foi o algoritmo que juntou as representações de *Chargrid* e *BERTgrid*, com um desempenho geral de 85.89%.

Por fim, foi a realizada a prova de conceito com a implementação do algoritmo com

melhores resultados no desempenho na plataforma da CSW, o IDV.

Em suma, o trabalho desenvolvido ao longo do estágio permitiu a concretização dos objetivos inicialmente traçados, contribuindo decisivamente para o desenvolvimento de competências na área da IA, nomeadamente mecanismos de aprendizagem automática e construção de algoritmos com inspiração na natureza, mas também competências de exploração, de investigação e de trabalho de equipa.

Apesar de não ser finalidade essencial deste estágio, o algoritmo proposto e utilizado na prova de conceito, revelou bons níveis de desempenho e considera-se que poderá ser útil para uma utilização futura na CSW. A oportunidade de desenvolver este estágio na CSW, uma empresa de referência na área da tecnologia, permitiu-me um contacto de proximidade com o mundo empresarial, aspeto que considero muito relevante na preparação para o início da atividade profissional.

Para finalizar, sinto que cumpri com as expetativas delineadas para o estágio, pelo que creio que o estágio foi muito bem sucedido.

Referências Bibliográficas

[1] Abdulkader, A. and Casey, M. R. (2009). Low cost correction of ocr errors using learning in a multi-engine environment. In *2009 10th International Conference on Document Analysis and Recognition*, pages 576–580. IEEE.

[2] Abiodun, O. I., Jantan, A., Omolara, A. E., Dada, K. V., Mohamed, N. A., and Arshad, H. (2018). State-of-the-art in artificial neural network applications: A survey. *Heliyon*, 4(11):e00938.

[3] Amano, A. and Asada, N. (2003). Graph grammar based analysis system of complex table form document. In *Seventh International Conference on Document Analysis and Recognition, 2003. Proceedings.*, pages 916–920. Citeseer.

[4] Bhatia, R. (2018). When not to use neural networks. `https://medium.com/datadriveninvestor/when-not-to-use-neural-networks-89fb50622429`. Acedido em 8/10/2019.

[5] Bloomberg, D. S. and Vincent, L. (2007). Document image applications. *Morphologie Mathmatique*, 8.

[6] Bukhari, S. S., Shafait, F., and Breuel, T. M. (2011). Improved document image segmentation algorithm using multiresolution morphology. In *Document recognition and retrieval XVIII*, volume 7874, page 78740D. International Society for Optics and Photonics.

[7] Denk, T. I. and Reisswig, C. (2019). Bertgrid: Contextualized embedding for 2d document representation and understanding. *arXiv preprint arXiv:1909.04948*.

[8] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

[9] Dozat, T. (2016). Incorporating nesterov momentum into adam.

[10] Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(Jul):2121–2159.

[11] Eikvil, L. (1993). Optical character recognition. *citeseer. ist. psu. edu/142042. html*.

[12] Esser, D., Schuster, D., Muthmann, K., Berger, M., and Schill, A. (2012). Automatic indexing of scanned documents: a layout-based approach. In *Document Recognition and Retrieval XIX*, volume 8297, page 82970H. International Society for Optics and Photonics.

[13] Fernández, F. C. and Terrades, O. R. (2012). Document segmentation using relative location features. In *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*, pages 1562–1565. IEEE.

[14] Ganesh, M. (2015). What are some real world applications of computer vision? `https://www.quora.com/What-are-some-real-world-applications-of-computer-vision`. Acedido em 03/02/2020.

[15] Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256.

[16] Grübler, M. (2018). Entendendo o funcionamento de uma rede neural artificial. `https://medium.com/brasil-ai/entendendo-o-funcionamento-de-uma-rede-neural-artificial-4463fcf44dd0`. Acedido em 10/10/2019.

[17] Gunasekaran, S. (1996). Computer vision technology for food quality assurance. *Trends in Food Science & Technology*, 7(8):245–256.

[Guru99] Guru99. Supervised vs unsupervised learning: Key differences. `https://www.guru99.com/supervised-vs-unsupervised-learning.html`. Acedido em 8/10/2019.

[19] Han, E.-H. S. and Karypis, G. (2000). Centroid-based document classification: Analysis and experimental results. In *European conference on principles of data mining and knowledge discovery*, pages 424–431. Springer.

[20] Haque, M. M., Pervin, S., and Begum, Z. (2013). Literature review of automatic single document text summarization using nlp. *International Journal of Innovation and Applied Studies*, 3(3):857–865.

[21] He, K., Zhang, X., Ren, S., and Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034.

[22] Heaton, J. (2019). Non-mathematical introduction to using neural networks. `https://www.heatonresearch.com/content/non-mathematical-introduction-using-neural-networks`. Acedido em 07/10/2019.

[23] Hellrich, J. (2019). *Word embeddings: reliability & semantic change*, volume 347. IOS Press.

[24] Hinton, G., Srivastava, N., and Swersky, K. (2012). Neural networks for machine learning lecture 6a overview of mini-batch gradient descent. *Cited on*, 14(8).

[25] Horev, R. (2018). Bert explained: State of the art language model for nlp. `https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270`. Acedido em 17/04/2020.

[26] Hori, O. and Doermann, D. S. (1995). Robust table-form structure analysis based on box-driven reasoning. In *Proceedings of 3rd International Conference on Document Analysis and Recognition*, volume 1, pages 218–221. IEEE.

[27] HungarianAlgorithm.com (2013). The hungarian algorithm. `http://www.hungarianalgorithm.com/hungarianalgorithm.php`. Acedido em 27/12/2019.

[28] in 90 Seconds, D. L. (2018). The best graphic design invoice templates. `https://www.digital-invoice-template.com/graphic-design/`. Acedido em 30/12/2019.

[29] Ince, T., Kiranyaz, S., Eren, L., Askar, M., and Gabbouj, M. (2016). Real-time motor fault detection by 1-d convolutional neural networks. *IEEE Transactions on Industrial Electronics*, 63(11):7067–7075.

[30] Janai, J., Güney, F., Behl, A., and Geiger, A. (2017). Computer vision for autonomous vehicles: Problems, datasets and state of the art. *Arxiv*, pages arXiv–1704.

[31] Joulin, A., Grave, E., Bojanowski, P., and Mikolov, T. (2016). Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

[32] Karani, D. (2018). Introduction to word embedding and word2vec. `https://towardsdatascience.com/introduction-to-word-embedding-and-word2vec-652d0c2060fa`. Acedido em 16/04/2020.

[33] Katti, A. R., Reisswig, C., Guder, C., Brarda, S., Bickel, S., Höhne, J., and Faddoul, J. B. (2018). Chargrid: Towards understanding 2d documents. *arXiv preprint arXiv:1809.08799*.

[34] Kim, N., Patel, R., Poliak, A., Wang, A., Xia, P., McCoy, R. T., Tenney, I., Ross, A., Linzen, T., Van Durme, B., et al. (2019). Probing what different nlp tasks teach machines about function word comprehension. *arXiv preprint arXiv:1904.11544*.

[35] Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[36] Kise, K., Sato, A., and Iwata, M. (1998). Segmentation of page images using the area voronoi diagram. *Computer Vision and Image Understanding*, 70(3):370–382.

[37] LeCun, Y. A., Bottou, L., Orr, G. B., and Müller, K.-R. (2012). Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–48. Springer.

[38] Li, W., Duan, F., Chen, B., Yuan, J., Tan, J. T. C., and Xu, B. (2012). Mobile robot action based on qr code identification. In *2012 IEEE international conference on robotics and biomimetics (ROBIO)*, pages 860–865. IEEE.

[39] Liddy, E. D. (2001). Natural language processing.

[40] Lin, T.-Y., Goyal, P., Girshick, R., He, K., and Dollár, P. (2017). Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988.

[41] Liu, X., Gao, F., Zhang, Q., and Zhao, H. (2019). Graph convolution for multimodal information extraction from visually rich documents. *CoRR*, abs/1903.11279.

[42] Luong, M.-T., Nguyen, T. D., and Kan, M.-Y. (2012). Logical structure recovery in scholarly articles with rich document features. In *Multimedia Storage and Retrieval Innovations for Digital Library Systems*, pages 270–292. IGI Global.

[43] Mao, S., Rosenfeld, A., and Kanungo, T. (2003). Document structure analysis algorithms: a literature survey. In *Document Recognition and Retrieval X*, volume 5010, pages 197–207. International Society for Optics and Photonics.

[44] McHenry, C. A. and Burt, S. W. (2014). Electronic document classification. US Patent 8,745,091.

[45] Mihajlovic, I. (2019). Everything you ever wanted to know about computer vision. `https://towardsdatascience.com/everything-you-ever-wanted-to-know-about-computer-vision-heres-a-look-why-it-s-so-awesome-e8a58dfb641e`. Acedido em 04/11/2019.

[46] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

[47] Milletari, F., Navab, N., and Ahmadi, S.-A. (2016). V-net: Fully convolutional neural networks for volumetric medical image segmentation. In *2016 fourth international conference on 3D vision (3DV)*, pages 565–571. IEEE.

[48] Mithe, R., Indalkar, S., and Divekar, N. (2013). Optical character recognition. *International journal of recent technology and engineering (IJRTE)*, 2(1):72–75.

[49] Nadkarni, P. M., Ohno-Machado, L., and Chapman, W. W. (2011). Natural language processing: an introduction. *Journal of the American Medical Informatics Association*, 18(5):544–551.

[50] Palm, R. B. (2019). End-to-end information extraction from business documents.

[51] Palm, R. B., Laws, F., and Winther, O. (2018). Attend, copy, parse-end-to-end information extraction from documents. *arXiv preprint arXiv:1812.07248*.

[52] Palm, R. B., Winther, O., and Laws, F. (2017). Cloudscan-a configuration-free invoice analysis system using recurrent neural networks. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 1, pages 406–413. IEEE.

[53] Patel, C., Patel, A., and Patel, D. (2012). Optical character recognition by open source ocr tool tesseract: A case study. *International Journal of Computer Applications*, 55(10):50–56.

[54] Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

[Point] Point, T. Artificial neural network - basic concepts. `https://www.tutorialspoint.com/artificial_neural_network/artificial_neural_network_basic_concepts.htm`. Acedido em 10/10/2019.

[56] Prince, S. J. (2012). *Computer vision: models, learning, and inference*. Cambridge University Press.

[57] Ramshaw, L. A. and Marcus, M. P. (1999). Text chunking using transformation-based learning. In *Natural language processing using very large corpora*, pages 157–176. Springer.

[58] Raschka, S. (2014). Activation functions for artificial neural networks. `http://rasbt.github.io/mlxtend/user_guide/general_concepts/activation-functions/`. Acedido em 8/10/2019.

[59] Rudak, P. (1990). Method for identifying unrecognizable characters in optical character recognition machines. US Patent 4,914,709.

[60] Schuster, D., Muthmann, K., Esser, D., Schill, A., Berger, M., Weidling, C., Aliyev, K., and Hofmeier, A. (2013). Intellix–end-user trained information extraction for document archiving. In *2013 12th International Conference on Document Analysis and Recognition*, pages 101–105. IEEE.

[61] Schwaber, K. (1997). Scrum development process. In *Business object design and implementation*, pages 117–134. Springer.

[62] SRL, A. M. (2018). Curs scrum agile. `https://www.automarini.ro/curs-agile.php`. Acedido em 08/01/2020.

[63] Szeliski, R. (2010). *Computer vision: algorithms and applications*. Springer Science & Business Media.

[64] Tafti, A. P., Baghaie, A., Assefi, M., Arabnia, H. R., Yu, Z., and Peissig, P. (2016). Ocr as a service: an experimental evaluation of google docs ocr, tesseract, abbyy finereader, and transym. In *International Symposium on Visual Computing*, pages 735–746. Springer.

[65] Tang, A., Tam, R., Cadrin-Chênevert, A., Guest, W., Chong, J., Barfett, J., Chepelev, L., Cairns, R., Mitchell, J., Cicero, M., Gaudreau Poudrette, M., Jaremko, J., Md, C., Gallix, B., Gray, B., Geis, R., O'Connell, T., Babyn, P., Koff, D., and Shabana, W. (2018).

Canadian association of radiologists white paper on artificial intelligence in radiology. *Canadian Association of Radiologists Journal*, 69.

[66] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

[67] Walia, A. S. (2017). Activation functions and it's types-which is better? `https://towardsdatascience.com/activation-functions-and-its-types-which-is-better-a9a5310cc8f`. Acedido em 8/10/2019.

[68] Wang, X. (2013). Intelligent multi-camera video surveillance: A review. *Pattern recognition letters*, 34(1):3–19.

[69] Woodard, D. L. and Flynn, P. J. (2005). Finger surface as a biometric identifier. *Computer vision and image understanding*, 100(3):357–384.

[70] Yang, X., Yümer, M. E., Asente, P., Kraley, M., Kifer, D., and Giles, C. L. (2017). Learning to extract semantic structure from documents using multimodal fully convolutional neural network. *CoRR*, abs/1706.02337.

[71] Yin, W., Ebert, S., and Schütze, H. (2016). Attention-based convolutional neural network for machine comprehension. *arXiv preprint arXiv:1602.04341*.

[72] Zeiler, M. D. (2012). ADADELTA: an adaptive learning rate method. *CoRR*, abs/1212.5701.

[73] Zhang, G., Patuwo, B. E., and Hu, M. Y. (1998). Forecasting with artificial neural networks:: The state of the art. *International journal of forecasting*, 14(1):35–62.

[74] Zhao, X., Wu, Z., and Wang, X. (2019). CUTIE: learning to understand documents with convolutional universal text information extractor. *CoRR*, abs/1903.12363.

Apêndices

.1 Apêndice A

Features do CloudScan	
Nome da *Feature*	**Descrição**
Bottom Margin	Coordenada vertical da margem inferior da palavra normalizada pela altura do documento
Top Margin	Coordenada vertical da margem superior da palavra normalizada pela altura do documento
Left Margin	Coordenada horizontal da margem esquerda da palavra normalizada pela largura do documento
Right Margin	Coordenada horizontal da margem direita da palavra normalizada pela largura do documento
Bottom Margin Relative	Distância vertical à palavra mais próxima localizada abaixo normalizada pela altura do documento
Top Margin Relative	Distância vertical à palavra mais próxima localizada acima normalizada pela altura do documento
Left Margin Relative	Distância horizontal à palavra mais próxima localizada à esquerda normalizada pela largura do documento
Right Margin Relative	Distância horizontal à palavra mais próxima localizada à direita normalizada pela largura do documento
Horizontal Position	Distância horizontal entre a palavra e a palavra mais á esquerda normalizada pela distância horizontal entre a palavra à esquerda e a palavra à direita
Vertical Position	Distância vertical entre a palavra e a palavra acima normalizada pela distância vertical entre a palavra acima e a palavra abaixo
Has Digits	Variável booleana que indica se a palavra contém caracteres numéricos
Is Known City	Variável booleana que indica se a palavra está presente numa base de dados de cidades conhecidas
Is Known Country	Variável booleana que indica se a palavra está presente numa base de dados de países conhecidos
Is Known Zip	Variável booleana que indica se a palavra está presente numa base de dados de códigos-postais conhecidos
Left Alignment	Número de palavras no mesmo documento que têm uma margem esquerda idêntica (com tolerância de 5 pixels)
Length	Número de caracteres da palavra
Page Height	Altura do documento
Page Width	Largura do documento

Position On Line	Número de palavras à esquerda normalizada pela número total de palavras da mesma linha
Line Size	Número de palavra presentes na mesma linha
Line White Space	Área ocupada pelos espaços em branco na mesma linha normalizada pela área total da linha
Parses as Amount	Variável booleana que indica se a palavra consegue ser interpretada como quantia
Parses as Date	Variável booleana que indica se a palavra consegue ser interpretada como data
Parses as Number	Variável booleana que indica se a palavra consegue ser interpretada como um número inteiro

Tabela 1: *Features* consideradas para cada palavra na implementação da representação do *CloudScan*

.2 Apêndice B

Testes efetuado ao *Learning Rate* e *Optimizers* do método *CloudScan*						
Learning Rate	*Optimizer*	*Beta 1*	*Beta 2*	*Momentum*	*Rho*	Desempenho Geral
0.0005	SGD	—	—	0.0	—	0.00%
0.0005	RMSProp	—	—	0.0	0.8	55.97%
0.0005	RMSProp	—	—	0.0	0.95	56.47%
0.0005	RMSProp	—	—	0.0	0.99	57.96%
0.0005	Adagrad	—	—	—	—	0.00%
0.0005	Adadelta	—	—	—	0.8	0.00%
0.0005	Adadelta	—	—	—	0.9	0.00%
0.0005	Adadelta	—	—	—	0.95	0.00%
0.0005	Adadelta	—	—	—	0.99	0.00%
0.0005	Adam	0.8	0.82	—	—	56.62%
0.0005	Adamax	0.8	0.82	—	—	56.62%
0.0005	Nadam	0.8	0.82	—	—	58.06%
0.0005	SGD	—	—	0.3	—	0.00%
0.0005	SGD	—	—	0.6	—	0.00%
0.0005	SGD	—	—	0.9	—	0.00%
0.001	SGD	—	—	0.0	—	0.00%
0.001	RMSProp	—	—	0.0	0.8	59.27%
0.001	RMSProp	—	—	0.0	0.9	56.37%
0.001	RMSProp	—	—	0.0	0.95	57.54%
0.001	RMSProp	—	—	0.0	0.99	59.38%
0.001	Adagrad	—	—	—	—	0.00%
0.001	Adadelta	—	—	—	0.8	0.00%
0.001	Adadelta	—	—	—	0.9	0.00%
0.001	Adadelta	—	—	—	0.95	0.00%
0.001	Adadelta	—	—	—	0.99	0.00%
0.001	Adam	0.8	0.82	—	—	56.54%
0.001	Adamax	0.8	0.82	—	—	59.11%
0.001	Nadam	0.8	0.82	—	—	56.83%
0.001	SGD	—	—	0.3	—	0.00%
0.001	SGD	—	—	0.6	—	0.00%
0.001	SGD	—	—	0.9	—	0.00%
0.005	SGD	—	—	0.0	—	0.00%
0.005	RMSProp	—	—	0.0	0.8	0.00%
0.005	RMSProp	—	—	0.0	0.9	0.00%
0.005	RMSProp	—	—	0.0	0.95	0.00%

0.005	RMSProp	—	—	0.0	0.99	0.00%
0.005	Adagrad	—	—	—	—	0.00%
0.005	Adadelta	—	—	—	0.8	0.00%
0.005	Adadelta	—	—	—	0.9	0.00%
0.005	Adadelta	—	—	—	0.95	0.00%
0.005	Adadelta	—	—	—	0.99	0.00%
0.005	Adam	0.8	0.82	—	—	0.00%
0.005	Adamax	0.8	0.82	—	—	0.00%
0.005	Nadam	0.8	0.82	—	—	0.00%
0.005	SGD	—	—	0.3	—	0.00%
0.005	SGD	—	—	0.6	—	0.00%
0.005	SGD	—	—	0.9	—	0.00%
0.0005	Adam	0.8	0.85	—	—	58.12%
0.0005	Adamax	0.8	0.85	—	—	58.77%
0.0005	Nadam	0.8	0.85	—	—	59.23%
0.001	Adam	0.8	0.85	—	—	59.06%
0.001	Adamax	0.8	0.85	—	—	57.26%
0.001	Nadam	0.8	0.85	—	—	59.53%
0.005	Adam	0.8	0.85	—	—	58.70%
0.005	Adamax	0.8	0.85	—	—	58.62%
0.005	Nadam	0.8	0.85	—	—	58.01%
0.0005	Adam	0.8	0.89	—	—	55.98%
0.0005	Adamax	0.8	0.89	—	—	57.02%
0.0005	Nadam	0.8	0.89	—	—	58.10%
0.001	Adam	0.8	0.89	—	—	58.72%
0.001	Adamax	0.8	0.89	—	—	58.93%
0.001	Nadam	0.8	0.89	—	—	58.18%
0.005	Adam	0.8	0.89	—	—	0.00%
0.005	Adamax	0.8	0.89	—	—	58.85%
0.005	Nadam	0.8	0.89	—	—	58.47%
0.0005	Adam	0.8	0.90	—	—	58.87%
0.0005	Adamax	0.8	0.90	—	—	59.29%
0.0005	Nadam	0.8	0.90	—	—	58.03%
0.001	Adam	0.8	0.90	—	—	60.69%
0.001	Adamax	0.8	0.90	—	—	59.16%
0.001	Nadam	0.8	0.90	—	—	58.17%
0.005	Adam	0.8	0.90	—	—	56.09%
0.005	Adamax	0.8	0.90	—	—	59.88%

0.005	Nadam	0.8	0.90	—	—	0.00%
0.0005	Adam	0.85	0.87	—	—	58.42%
0.0005	Adamax	0.85	0.87	—	—	56.78%
0.0005	Nadam	0.85	0.87	—	—	58.40%
0.001	Adam	0.85	0.87	—	—	59.37%
0.001	Adamax	0.85	0.87	—	—	58.04%
0.001	Nadam	0.85	0.87	—	—	58.99%
0.005	Adam	0.85	0.87	—	—	0.00%
0.005	Adamax	0.85	0.87	—	—	0.00%
0.005	Nadam	0.85	0.87	—	—	60.50%
0.0005	Adam	0.85	0.90	—	—	60.09%
0.0005	Adamax	0.85	0.90	—	—	57.12%
0.0005	Nadam	0.85	0.90	—	—	56.63%
0.001	Adam	0.85	0.90	—	—	57.75%
0.001	Adamax	0.85	0.90	—	—	56.70%
0.001	Nadam	0.85	0.90	—	—	57.09%
0.005	Adam	0.85	0.90	—	—	0.00%
0.005	Adamax	0.85	0.90	—	—	59.05%
0.005	Nadam	0.85	0.90	—	—	0.00%
0.0005	Adam	0.85	0.94	—	—	58.44%
0.0005	Adamax	0.85	0.94	—	—	57.58%
0.0005	Nadam	0.85	0.94	—	—	55.89%
0.001	Adam	0.85	0.94	—	—	58.84%
0.001	Adamax	0.85	0.94	—	—	59.58%
0.001	Nadam	0.85	0.94	—	—	59.69%
0.005	Adam	0.85	0.94	—	—	0.00%
0.005	Adamax	0.85	0.94	—	—	56.95%
0.005	Nadam	0.85	0.94	—	—	56.98%
0.0005	Adam	0.85	0.95	—	—	56.64%
0.0005	Adamax	0.85	0.95	—	—	59.86%
0.0005	Nadam	0.85	0.95	—	—	55.15%
0.001	Adam	0.85	0.95	—	—	56.58%
0.001	Adamax	0.85	0.95	—	—	58.41%
0.001	Nadam	0.85	0.95	—	—	56.60%
0.005	Adam	0.85	0.95	—	—	0.00%
0.005	Adamax	0.85	0.95	—	—	58.52%
0.005	Nadam	0.85	0.95	—	—	0.00%
0.0005	Adam	0.9	0.92	—	—	57.86%

0.0005	Adamax	0.9	0.92	—	—	56.85%
0.0005	Nadam	0.9	0.92	—	—	57.30%
0.001	Adam	0.9	0.92	—	—	57.31%
0.001	Adamax	0.9	0.92	—	—	56.24%
0.001	Nadam	0.9	0.92	—	—	58.37%
0.005	Adam	0.9	0.92	—	—	0.00%
0.005	Adamax	0.9	0.92	—	—	56.90%
0.005	Nadam	0.9	0.92	—	—	0.00%
0.0005	Adam	0.9	0.95	—	—	57.03%
0.0005	Adamax	0.9	0.95	—	—	56.92%
0.0005	Nadam	0.9	0.95	—	—	54.05%
0.001	Adam	0.9	0.95	—	—	61.95%
0.001	Adamax	0.9	0.95	—	—	56.48%
0.001	Nadam	0.9	0.95	—	—	58.25%
0.005	Adam	0.9	0.95	—	—	0.00%
0.005	Adamax	0.9	0.95	—	—	0.00%
0.005	Nadam	0.9	0.95	—	—	0.00%
0.0005	Adam	0.9	0.99	—	—	58.40%
0.0005	Adamax	0.9	0.99	—	—	54.13%
0.0005	Nadam	0.9	0.99	—	—	57.43%
0.001	Adam	0.9	0.99	—	—	58.27%
0.001	Adamax	0.9	0.99	—	—	56.65%
0.001	Nadam	0.9	0.99	—	—	57.34%
0.005	Adam	0.9	0.99	—	—	0.00%
0.005	Adamax	0.9	0.99	—	—	58.20%
0.005	Nadam	0.9	0.99	—	—	0.00%
0.0005	Adam	0.9	1.00	—	—	56.02%
0.0005	Adamax	0.9	1.00	—	—	54.67%
0.0005	Nadam	0.9	1.00	—	—	58.04%
0.001	Adam	0.9	1.00	—	—	59.66%
0.001	Adamax	0.9	1.00	—	—	54.91%
0.001	Nadam	0.9	1.00	—	—	57.81%
0.005	Adam	0.9	1.00	—	—	0.77%
0.005	Adamax	0.9	1.00	—	—	0.00%
0.005	Nadam	0.9	1.00	—	—	0.00%

Tabela 2: Testes de optimização dos hiper parâmetros *Optimizer* e *Learning Rate* do algoritmo *CloudScan* com recurso à métrica *F1-Score*

.3 Apêndice C

Inicialização e Regulação dos Pesos e ao Balanceamento das Classes do *CloudScan*				
Método Inicialização	*Focal Loss*	*Weight Decay*	*Penalty Decay*	Desempenho Geral
Random Normal	None	0	None	56.41%
Random Normal	1.01	0	None	56.06%
Random Normal	1.03	0	None	52.63%
Random Normal	1.05	0	None	0.00%
Random Normal	None	0.0001	None	28.54%
Random Normal	1.01	0.0001	None	23.75%
Random Normal	1.03	0.0001	None	38.09%
Random Normal	1.05	0.0001	None	0.00%
Random Normal	None	0.0005	None	0.00%
Random Normal	1.01	0.0005	None	0.00%
Random Normal	1.03	0.0005	None	0.00%
Random Normal	1.05	0.0005	None	0.00%
Random Normal	None	0.001	None	0.00%
Random Normal	1.01	0.001	None	0.00%
Random Normal	1.03	0.001	None	0.00%
Random Normal	1.05	0.001	None	0.00%
Random Normal	None	0.005	None	0.00%
Random Normal	1.01	0.005	None	0.00%
Random Normal	1.03	0.005	None	0.00%
Random Normal	1.05	0.005	None	0.00%
Random Normal	None	0	L1	0.00%
Random Normal	1.01	0	L1	0.00%
Random Normal	1.03	0	L1	54.76%
Random Normal	1.05	0	L1	0.00%
Random Normal	None	0.0001	L1	46.08%
Random Normal	1.01	0.0001	L1	49.10%
Random Normal	1.03	0.0001	L1	45.21%
Random Normal	1.05	0.0001	L1	0.00%
Random Normal	None	0.0005	L1	31.91%
Random Normal	1.01	0.0005	L1	37.92%
Random Normal	1.03	0.0005	L1	44.12%
Random Normal	1.05	0.0005	L1	0.00%
Random Normal	None	0.001	L1	25.67%
Random Normal	1.01	0.001	L1	29.11%
Random Normal	1.03	0.001	L1	31.67%

Random Normal	1.05	0.001	L1	0.00%
Random Normal	None	0.005	L1	0.00%
Random Normal	1.01	0.005	L1	0.00%
Random Normal	1.03	0.005	L1	0.00%
Random Normal	1.05	0.005	L1	0.00%
Random Normal	None	0	L2	0.00%
Random Normal	1.01	0	L2	0.00%
Random Normal	1.03	0	L2	0.00%
Random Normal	1.05	0	L2	0.00%
Random Normal	None	0.0001	L2	0.00%
Random Normal	1.01	0.0001	L2	0.00%
Random Normal	1.03	0.0001	L2	0.00%
Random Normal	1.05	0.0001	L2	0.00%
Random Normal	None	0.0005	L2	0.00%
Random Normal	1.01	0.0005	L2	0.00%
Random Normal	1.03	0.0005	L2	0.00%
Random Normal	1.05	0.0005	L2	0.00%
Random Normal	None	0.001	L2	0.00%
Random Normal	1.01	0.001	L2	0.00%
Random Normal	1.03	0.001	L2	0.00%
Random Normal	1.05	0.001	L2	0.00%
Random Normal	None	0.005	L2	0.00%
Random Normal	1.01	0.005	L2	0.00%
Random Normal	1.03	0.005	L2	0.00%
Random Normal	1.05	0.005	L2	0.00%
He Uniform	None	0	None	0.00%
He Uniform	1.01	0	None	0.00%
He Uniform	1.03	0	None	0.00%
He Uniform	1.05	0	None	0.00%
He Uniform	None	0.0001	None	29.32%
He Uniform	1.01	0.0001	None	30.70%
He Uniform	1.03	0.0001	None	41.13%
He Uniform	1.05	0.0001	None	0.00%
He Uniform	None	0.0005	None	0.00%
He Uniform	1.01	0.0005	None	0.00%
He Uniform	1.03	0.0005	None	0.00%
He Uniform	1.05	0.0005	None	0.00%
He Uniform	None	0.001	None	0.00%

He Uniform	1.01	0.001	None	0.00%
He Uniform	1.03	0.001	None	0.00%
He Uniform	1.05	0.001	None	0.00%
He Uniform	None	0.005	None	0.00%
He Uniform	1.01	0.005	None	0.00%
He Uniform	1.03	0.005	None	0.00%
He Uniform	1.05	0.005	None	0.00%
He Uniform	None	0	L1	56.04%
He Uniform	1.01	0	L1	58.08%
He Uniform	1.03	0	L1	58.95%
He Uniform	1.05	0	L1	0.00%
He Uniform	None	0.0001	L1	46.61%
He Uniform	1.01	0.0001	L1	48.18%
He Uniform	1.03	0.0001	L1	50.20%
He Uniform	1.05	0.0001	L1	0.00%
He Uniform	None	0.0005	L1	62.21%
He Uniform	1.01	0.0005	L1	34.44%
He Uniform	1.03	0.0005	L1	40.17%
He Uniform	1.05	0.0005	L1	0.00%
He Uniform	None	0.001	L1	26.45%
He Uniform	1.01	0.001	L1	26.54%
He Uniform	1.03	0.001	L1	38.94%
He Uniform	1.05	0.001	L1	0.00%
He Uniform	None	0.005	L1	0.00%
He Uniform	1.01	0.005	L1	0.00%
He Uniform	1.03	0.005	L1	0.00%
He Uniform	1.05	0.005	L1	0.00%
He Uniform	None	0	L2	0.00%
He Uniform	1.01	0	L2	0.00%
He Uniform	1.03	0	L2	0.00%
He Uniform	1.05	0	L2	0.00%
He Uniform	None	0.0001	L2	0.00%
He Uniform	1.01	0.0001	L2	0.00%
He Uniform	1.03	0.0001	L2	0.00%
He Uniform	1.05	0.0001	L2	0.00%
He Uniform	None	0.0005	L2	0.00%
He Uniform	1.01	0.0005	L2	0.00%
He Uniform	1.03	0.0005	L2	0.00%

He Uniform	1.05	0.0005	L2	0.00%
He Uniform	None	0.001	L2	0.00%
He Uniform	1.01	0.001	L2	0.00%
He Uniform	1.03	0.001	L2	0.00%
He Uniform	1.05	0.001	L2	0.00%
He Uniform	None	0.005	L2	0.00%
He Uniform	1.01	0.005	L2	0.00%
He Uniform	1.03	0.005	L2	0.00%
He Uniform	1.05	0.005	L2	0.00%
Glorot Uniform	None	0	None	55.65%
Glorot Uniform	1.01	0	None	58.69%
Glorot Uniform	1.03	0	None	0.00%
Glorot Uniform	1.05	0	None	0.00%
Glorot Uniform	None	0.0001	None	25.88%
Glorot Uniform	1.01	0.0001	None	37.63%
Glorot Uniform	1.03	0.0001	None	49.55%
Glorot Uniform	1.05	0.0001	None	0.00%
Glorot Uniform	None	0.0005	None	0.00%
Glorot Uniform	1.01	0.0005	None	0.00%
Glorot Uniform	1.03	0.0005	None	0.00%
Glorot Uniform	1.05	0.0005	None	0.00%
Glorot Uniform	None	0.001	None	0.00%
Glorot Uniform	1.01	0.001	None	0.00%
Glorot Uniform	1.03	0.001	None	0.00%
Glorot Uniform	1.05	0.001	None	0.00%
Glorot Uniform	None	0.005	None	0.00%
Glorot Uniform	1.01	0.005	None	0.00%
Glorot Uniform	1.03	0.005	None	0.00%
Glorot Uniform	1.05	0.005	None	0.00%
Glorot Uniform	None	0	L1	56.82%
Glorot Uniform	1.01	0	L1	57.89%
Glorot Uniform	1.03	0	L1	0.00%
Glorot Uniform	1.05	0	L1	0.00%
Glorot Uniform	None	0.0001	L1	46.77%
Glorot Uniform	1.01	0.0001	L1	48.31%
Glorot Uniform	1.03	0.0001	L1	49.28%
Glorot Uniform	1.05	0.0001	L1	0.00%
Glorot Uniform	None	0.0005	L1	30.01%

Glorot Uniform	1.01	0.0005	L1	36.11%
Glorot Uniform	1.03	0.0005	L1	44.41%
Glorot Uniform	1.05	0.0005	L1	0.00%
Glorot Uniform	None	0.001	L1	26.27%
Glorot Uniform	1.01	0.001	L1	32.71%
Glorot Uniform	1.03	0.001	L1	40.03%
Glorot Uniform	1.05	0.001	L1	0.00%
Glorot Uniform	None	0.005	L1	0.00%
Glorot Uniform	1.01	0.005	L1	0.00%
Glorot Uniform	1.03	0.005	L1	0.00%
Glorot Uniform	1.05	0.005	L1	0.00%
Glorot Uniform	None	0	L2	0.00%
Glorot Uniform	1.01	0	L2	0.00%
Glorot Uniform	1.03	0	L2	0.00%
Glorot Uniform	1.05	0	L2	0.00%
Glorot Uniform	None	0.0001	L2	0.00%
Glorot Uniform	1.01	0.0001	L2	0.00%
Glorot Uniform	1.03	0.0001	L2	0.00%
Glorot Uniform	1.05	0.0001	L2	0.00%
Glorot Uniform	None	0.0005	L2	0.00%
Glorot Uniform	1.01	0.0005	L2	0.00%
Glorot Uniform	1.03	0.0005	L2	0.00%
Glorot Uniform	1.05	0.0005	L2	0.00%
Glorot Uniform	None	0.001	L2	0.00%
Glorot Uniform	1.01	0.001	L2	0.00%
Glorot Uniform	1.03	0.001	L2	0.00%
Glorot Uniform	1.05	0.001	L2	0.00%
Glorot Uniform	None	0.005	L2	0.00%
Glorot Uniform	1.01	0.005	L2	0.00%
Glorot Uniform	1.03	0.005	L2	0.00%
Glorot Uniform	1.05	0.005	L2	0.00%
Lecun Uniform	None	0	None	56.18%
Lecun Uniform	1.01	0	None	58.29%
Lecun Uniform	1.03	0	None	0.00%
Lecun Uniform	1.05	0	None	1.07%
Lecun Uniform	None	0.0001	None	23.78%
Lecun Uniform	1.01	0.0001	None	31.76%
Lecun Uniform	1.03	0.0001	None	36.98%

Lecun Uniform	1.05	0.0001	None	0.00%
Lecun Uniform	None	0.0005	None	0.00%
Lecun Uniform	1.01	0.0005	None	0.00%
Lecun Uniform	1.03	0.0005	None	0.00%
Lecun Uniform	1.05	0.0005	None	0.00%
Lecun Uniform	None	0.001	None	0.00%
Lecun Uniform	1.01	0.001	None	0.00%
Lecun Uniform	1.03	0.001	None	0.00%
Lecun Uniform	1.05	0.001	None	0.00%
Lecun Uniform	None	0.005	None	0.00%
Lecun Uniform	1.01	0.005	None	0.00%
Lecun Uniform	1.03	0.005	None	0.00%
Lecun Uniform	1.05	0.005	None	0.00%
Lecun Uniform	None	0	L1	56.24%
Lecun Uniform	1.01	0	L1	57.60%
Lecun Uniform	1.03	0	L1	0.00%
Lecun Uniform	1.05	0	L1	0.00%
Lecun Uniform	None	0.0001	L1	47.78%
Lecun Uniform	1.01	0.0001	L1	42.21%
Lecun Uniform	1.03	0.0001	L1	48.88%
Lecun Uniform	1.05	0.0001	L1	10.62%
Lecun Uniform	None	0.0005	L1	32.64%
Lecun Uniform	1.01	0.0005	L1	39.12%
Lecun Uniform	1.03	0.0005	L1	29.93%
Lecun Uniform	1.05	0.0005	L1	33.87%
Lecun Uniform	None	0.001	L1	27.78%
Lecun Uniform	1.01	0.001	L1	35.46%
Lecun Uniform	1.03	0.001	L1	40.34%
Lecun Uniform	1.05	0.001	L1	0.00%
Lecun Uniform	None	0.005	L1	0.00%
Lecun Uniform	1.01	0.005	L1	0.00%
Lecun Uniform	1.03	0.005	L1	0.00%
Lecun Uniform	1.05	0.005	L1	0.00%
Lecun Uniform	None	0	L2	0.00%
Lecun Uniform	1.01	0	L2	0.00%
Lecun Uniform	1.03	0	L2	0.00%
Lecun Uniform	1.05	0	L2	0.00%
Lecun Uniform	None	0.0001	L2	0.00%

Lecun Uniform	1.01	0.0001	L2	0.00%
Lecun Uniform	1.03	0.0001	L2	0.00%
Lecun Uniform	1.05	0.0001	L2	0.00%
Lecun Uniform	None	0.0005	L2	0.00%
Lecun Uniform	1.01	0.0005	L2	0.00%
Lecun Uniform	1.03	0.0005	L2	0.00%
Lecun Uniform	1.05	0.0005	L2	0.00%
Lecun Uniform	None	0.001	L2	0.00%
Lecun Uniform	1.01	0.001	L2	0.00%
Lecun Uniform	1.03	0.001	L2	0.00%
Lecun Uniform	1.05	0.001	L2	0.00%
Lecun Uniform	None	0.005	L2	0.00%
Lecun Uniform	1.01	0.005	L2	0.00%
Lecun Uniform	1.03	0.005	L2	0.00%
Lecun Uniform	1.05	0.005	L2	0.00%

Tabela 3: Testes de optimização dos hiper parâmetros da Inicialização e Regulação dos Pesos e também ao Balanceamento das Classes do algoritmo *CloudScan* com recurso à métrica *F1-Score*