Pontifícia Universidade Católica
do Rio de Janeiro

Matheus Telles Werner

Extracting Section Structure from Resumes in Brazilian Portuguese

Tese de Doutorado

Thesis presented to the Programa de Pós–graduação em Informática, do Departamento de Informática of PUC-Rio in partial fulfillment of the requirements for the degree of Doutor em Informática.

Advisor: Prof. Eduardo Sany Laber

Rio de Janeiro
September 2023

Pontifícia Universidade Católica
do Rio de Janeiro

Matheus Telles Werner

Extracting Section Structure from Resumes in Brazilian Portuguese

Thesis presented to the Programa de Pós–graduação em Informática of PUC-Rio in partial fulfillment of the requirements for the degree of Doutor em Informática. Approved by the Examination Committee:

Prof. Eduardo Sany Laber
Advisor
Departamento de Informática – PUC-Rio

Prof. Sergio Colcher
Departamento de Informática – PUC-Rio

Prof. Alberto Barbosa Raposo
Departamento de Informática – PUC-Rio

Dr. Alexandre Roberto Renteria
Jobzi

Prof. Julio Cesar Duarte
IME

Rio de Janeiro, September 27th, 2023

Matheus Telles Werner

Bachelor's in Computer Engineering (2016) at the Pontifical Catholic University of Rio de Janeiro (PUC-Rio). Master's in Informatics (2019) at the Pontifical Catholic University of Rio de Janeiro (PUC-Rio).

Bibliographic data

Werner, Matheus Telles

Extracting Section Structure from Resumes in Brazilian Portuguese / Matheus Telles Werner; advisor: Eduardo Sany Laber. – 2023.

85 f: il. color. ; 30 cm

Tese (doutorado) - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática, 2023.

Inclui bibliografia

1. Engenharia de Computação – Teses. 2. Analisador de Currículo. 3. Processamento de Linguagem Natural. 4. Extração de Informação. 5. Classificação de Imagem. 6. Segmentação de Texto. 7. Recursos Humanos. I. Laber, Eduardo Sany. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. III. Título.

CDD: 004

Acknowledgments

To my advisor Eduardo Sany Laber, for all your guidance and support over these four and a half years. Your professionalism, sincerity, and dedication have been a constant source of inspiration.

To my parents Rudi and Roberta, and my brothers Lucas and Natan, for your unconditional love and support during all this time and for understanding my decision to pursue my Ph.D.

To Louise, your love and support throughout this entire journey, particularly during this final stretch, mean a lot to me. Your presence, affection, and encouragement helped me finally get here.

To my friends and colleagues, for all these years of friendship and all the support and encouragement.

To my psychologist Cristiane Diniz, for all the invaluable assistance you have provided from the COVID pandemic up to this very moment. You have my eternal gratitude.

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

Abstract

Werner, Matheus Telles; Laber, Eduardo Sany (Advisor). **Extracting Section Structure from Resumes in Brazilian Portuguese** . Rio de Janeiro, 2023. 85p. Tese de Doutorado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

This thesis presents a novel resume parser designed to effectively reorganize the textual content of any resume into its original section structure. Our work addresses two practical challenges overlooked by the existing literature: (i) ensuring the correct reading order of text retrieved from resume files and (ii) extracting individually all sections, as well as work experience and education subsections. By taking into account the observation that most resumes adhere to basic document templates, we reframe the reading order problem as a template identification task. Our experiments suggest that even a widely-used small model like EfficientNet-B0 can accurately identify common templates. Additionally, we propose a sequence tagging approach that simultaneously identifies all resume sections and some subsections. We implement and compare two solutions based on the well-known CRF and BERT models. Our evaluation provides strong evidence that the CRF can serve as a practical alternative to BERT, depending on hardware and budget constraints. They yield comparable results in terms of identifying resume sections, while BERT displays a substantial advantage when identifying education and work experience subsections.

Keywords

Resume Parser; Natural Language Processing; Information Extraction; Image Classification; Text Segmentation; Human Resources.

Resumo

Werner, Matheus Telles; Laber, Eduardo Sany. **Extraindo a Estrutura de Seção de Currículos em Português** . Rio de Janeiro, 2023. 85p. Tese de Doutorado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Esta tese apresenta um novo analisador de currículos projetado para reorganizar o conteúdo textual de qualquer currículo em sua estrutura de seção original. Nosso trabalho aborda dois desafios práticos negligenciados pela literatura existente: (i) garantir a ordem de leitura correta do texto recuperado do arquivo de currículo e (ii) extrair individualmente todas as seções, bem como as subseções de experiências de trabalho e educação. Levando em consideração a observação de que a maioria dos currículos adere a modelos básicos de documentos, reformulamos o problema da ordem de leitura como uma tarefa de identificação de modelos de documento. Nossos experimentos sugerem que mesmo um pequeno modelo amplamente utilizado como o EfficientNet-B0 pode identificar com precisão modelos de documento comuns. Além disso, propomos uma abordagem de rotulação de sequências que identifica simultaneamente todas as seções do currículo e algumas subseções. Implementamos e comparamos duas soluções baseados nos conhecidos modelos CRF e BERT. Nossa avaliação fornece fortes evidências de que o CRF pode servir como uma alternativa prática ao BERT, dependendo do hardware e das restrições orçamentárias. Eles produzem resultados comparáveis em termos de identificação de seções de currículo, enquanto o BERT demonstra uma vantagem substancial ao identificar as subseções de educação e experiências de trabalho.

Palavras-chave

Analisador de Currículo; Processamento de Linguagem Natural; Extração de Informação; Classificação de Imagem; Segmentação de Texto; Recursos Humanos.

Table of contents

List of figures

List of tables

List of Abreviations

BERT – Bidirectional Encoder Representations from Transformers

Bi-LSTM – Bidirectional Long-Short-Term-Memory

CV – Curriculum Vitae

CNN – Convolutional Neural Network

CRF – Conditional Random Field

HMM – Hidden Markov Model

IOB – Inside-Outside-Beginning

IOBES – Inside-Outside-Beginning-End-Single

LSTM – Long Short Term Memory

NLP – Natural Language Processing

PDF – Portable Document Format

ResNets – Residual Networks

RNN – Recurrent Neural Networks

Seq2Seq – Sequence-to-Sequence

XML – Extensible Markup Language

I love deadlines. I love the whooshing noise they make as they go by.

Douglas Adams, *The Salmon of Doubt.*

1
Introduction

A resume, or curriculum vitae (CV), accounts for an individual's professional qualifications and educational background. It systematically organizes this information into distinct sections and presents it through various formats, including plain text, itemized lists, or further subdivisions. As an example, the *Summary* section provides an overview of an individual's noteworthy professional accomplishments in a narrative format while the *Work Experiences* section presents a compilation of previous positions held by the individual, typically accompanied by details such as the company name, employment duration, and a concise description of job responsibilities. Despite these well-defined structural elements, the content and presentation of information within a resume can exhibit significant variations influenced by personal preferences, industry conventions, and the job role being pursued.

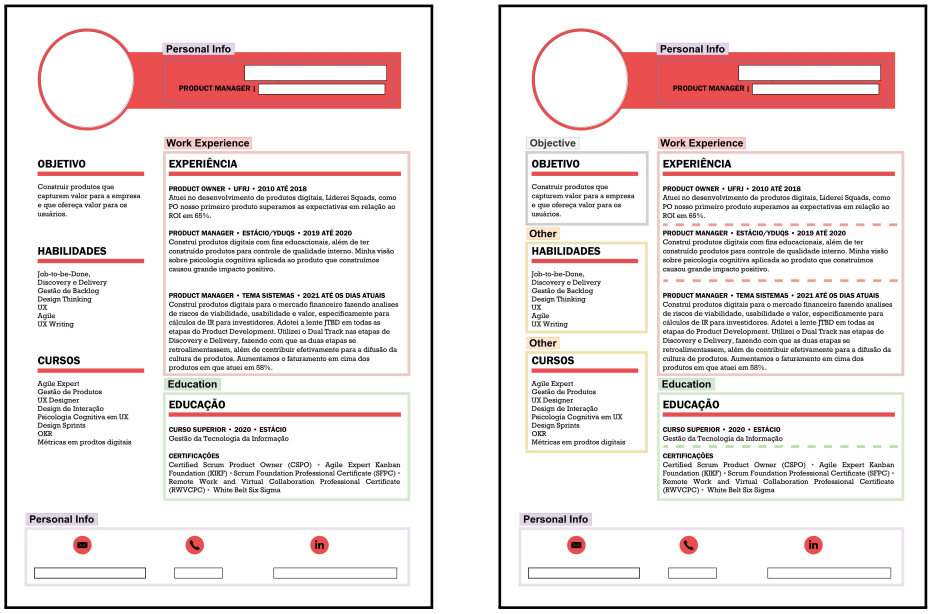
For Recruitment and Selection processes, developing a resume parser capable of accurately identifying all sections within any resume and extracting the pertinent information holds tremendous value for recruiters and companies. Such a tool would streamline various processes, including candidate selection during recruitment and user registration on job portals through resume uploads. Additionally, it can potentially support related NLP tasks, such as Job Matching [4, 5] and Skill Recommendation [6].

However, existing literature primarily focuses on entity extraction from the main resume sections [7, 8, 9, 10]. This involves identifying text segments corresponding to *Personal Info*, *Work Experiences*, and *Education* and, then, extracting entities from them such as *Contact Information*, *Job Titles*, *Company names*, and *Educational Institutions*. While obtaining these entities is a crucial aspect of resume parsing, it represents only part of the functionality required in practical scenarios. Equally important is extracting the remaining sections and establishing relationships between extracted entities, aspects covered only by commercial resume parsers such as Affinda[1] and Nanotecs[2].

Obtaining the complete section structure could help to effectively accomplish this objective, as it would enable more coherent and meaningful associ-

[1]https://affinda.com/
[2]https://nanonets.com/

ations between the information contained within the resumes. For instance, as demonstrated in Figure 1.1, extracting all resume sections and subsections is essential to attain a more comprehensive and precise representation of the resume's content. This enhanced representation could simplify processes such as grouping extracted entities related to the same work experiences, given that they would already be inside the same subsections.



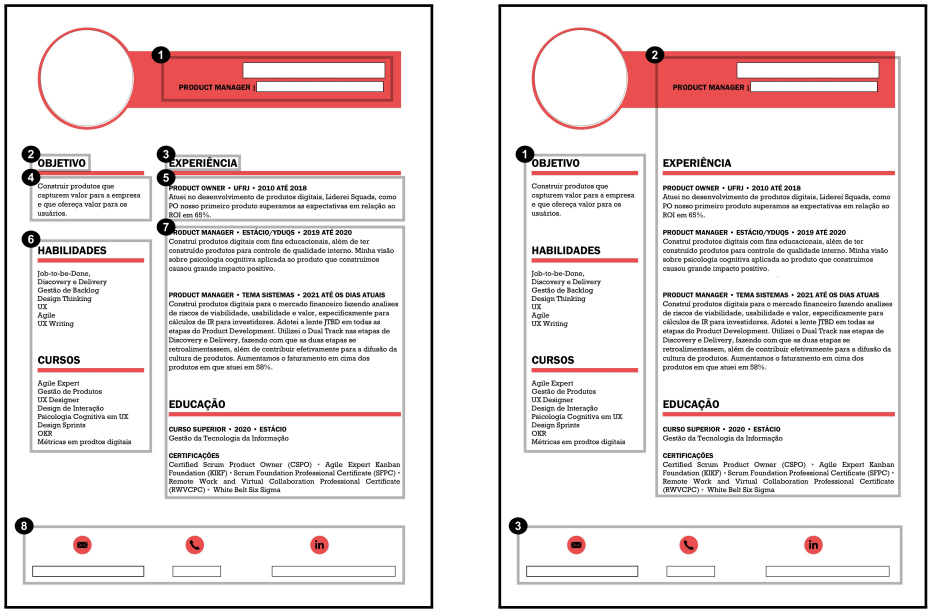
(a) Typical (b) Ideal

Figure 1.1: Figure 1.1a displays the sections typically identified by the resume parsers described in the literature. In contrast, Figure 1.1b presents an enhanced identification that includes additional sections and introduces subsections (dashed lines) for *Work Experiences* and *Education*.

Another often overlooked aspect in the literature is the significant influence of the chosen file format to store resume information on the process of information extraction. Arguably, PDF [11] stands out as the predominant document format for resumes, necessitating the initial step of text extraction through tools like PDFBox [12] to identify pertinent entities within the document. However, a significant challenge arises due to the inherent design of PDFs, which do not prioritize storing text in an order that is reasonable for human readers. This lack of adherence to a coherent reading order can result in extracted text being arranged in an unconventional manner, and this, in turn, has a direct influence on the effectiveness of resume parsers.

Figure 1.2 provides a visual representation of this issue. As we can observe, the resume text extracted by a PDF parser fragmented the sections *Objective* and *Work Experiences*, scattering their content across different portions of the text. It is worth emphasizing that the optimal reading order is

subjective and may vary among different readers. Nonetheless, in the context of resume parsing, it is fundamental to ensure the cohesion of each section.



(a) Actual (b) Expected

Figure 1.2: Figure 1.2a provides a visual representation of the reading order derived from text extracted by a PDF parser, with text boxes in the figure displaying numbers in its top left corner to represent this order. Ideally, the parser should aim to generate a reading order similar to that illustrated in Figure 1.2b.

Lastly, to our knowledge, existing literature have primarily targeted resumes in English [8, 9, 13], Chinese [7, 14], and European languages [10, 13]. This has resulted in a notable void in the availability of effective resume parsing tools tailored specifically for the Brazilian market.

1.1 Our Contributions

Our main contribution lies in developing a novel resume parser for Brazilian Portuguese that addresses two practical aspects that were overlooked by the proposals available in the literature: (i) ensuring the accurate reading order of text extracted from resume files and (ii) extracting the entire section structure within a resume. In other words, we propose a resume parser pipeline that given a pdf file systematically identifies and delineates all sections within a resume. More precisely, in the case of the *Education* and *Work Experiences* sections, we extend our approach to the individual identification of each subsection within these sections. This emphasis on *Education* and *Work Experiences* stems from their key role as arguably the main sections

of a resume, containing numerous entities that require association with other entities (e.g., *Job Titles* and *Company Names*).

To restore the correct reading order of the text, we first devised a metric to distinguish different levels of complexity in document layouts. We employ this metric to classify the resumes according to their complexity and then we identify the prevalent classes. The majority of resumes found in our study has low layout complexity and, thus, we focus on them. In such cases, deducing the correct reading order proves relatively straightforward. By exploiting this convention, we reframe the text reading order problem into a template identification task, which can be effectively addressed through Image Classification techniques. Our experiments suggest that even a widely-used small model like EfficientNet-B0 [2] can accurately identify common templates with a satisfactory accuracy above 85%. Additionally, we demonstrate that identifying page templates using an anonymized version of the pages (highlighting character-containing regions only) yields competitive results, ensuring data privacy while allowing us to make our anonymized dataset publicly available.

To extract the sections and subsections, we design a sequence tagging approach to simultaneously identify both types of information. Using our proposed approach, we implement two solutions based on the well-known CRF [15] and BERT [16] models. We aim to investigate the performance disparity between these models and evaluate their practical implications, particularly considering their significant computational cost differences. Our experiments demonstrate good results with regard to the sequences of segments that are generated. In terms of section identification, both CRF and BERT exhibited comparable performance, with CRF achieving an F1-Score of 83.5% and BERT achieving 84.9%. However, when identifying education and work experience subsections, BERT demonstrated a significant advantage over CRF: the latter achieved an F1-Score of 72.9%, while the former achieved 82.4%. Nonetheless, these results suggest that the CRF can be a feasible alternative to BERT, depending on hardware and budget limitations.

In summary, our contributions encompass the following:

1. Development of a resume parser for Brazilian Portuguese designed to structure the entire resume;

2. A simple yet effective approach to ensure the correct text reading order based on layout information;

3. Two segmentation models for extracting the sections and subsections simultaneously;

4. An in-depth analysis of employing BERT- and CRF-based models for resume segmentation;

5. An anonymized dataset for page template identification, promoting data accessibility while upholding privacy considerations.

It is worth mentioning that our study primarily concentrates on resumes in PDF format. Our choice is motivated by its widespread popularity and extensive usage [3]. Additionally, less common formats such as DOC and DOCX can easily be converted to PDF for compatibility.

1.2 Thesis Structure

The thesis is organized as follows. In Section 2, we introduce the fundamental concepts and methods required to comprehend the remainder of this work. Then, in Section 3, we provide a brief overview of existing academic and commercial approaches to resume parsing and other related information extraction tasks. Section 4 presents the pipeline of our resume parser, outlining its key components and functionality. In Section 5, we describe the datasets developed to evaluate our approach's performance and present our experimental study's results, focusing on page template identification using EfficientNet-B0 and resume segmentation using both CRF and BERT models. Finally, in Section 6, we conclude our research and discuss future directions for enhancing the capabilities and applications of our resume parser.

1.3 Disclaimer

Most of the resume information displayed throughout this paper was generated by us or taken from Microsoft Create[4]. Information from a real resume was only displayed when strictly needed. However, in those cases, we did not present any personal information that would make it simple to identify the author of the resume.

[3] https://pdfa.org/pdfs-popularity-online/
[4] `https://create.microsoft.com/en-us/templates/resumes-and-cover-letters`

2
Background

This chapter overviews the fundamental concepts and methods required to comprehend the remainder of this work. We begin by properly introducing the PDF format in Section 2.1, emphasizing its pertinent characteristics in our specific context. Proceeding to Section 2.2, we delve into the labeling tools we chose to assist in annotating the data collected for evaluating our proposed methodology. Lastly, Sections 2.3 and 2.4 provide a formal presentation of the machine learning tasks central to this work, accompanied by a comprehensive exposition of the corresponding models that will be employed in subsequent segments of this thesis.

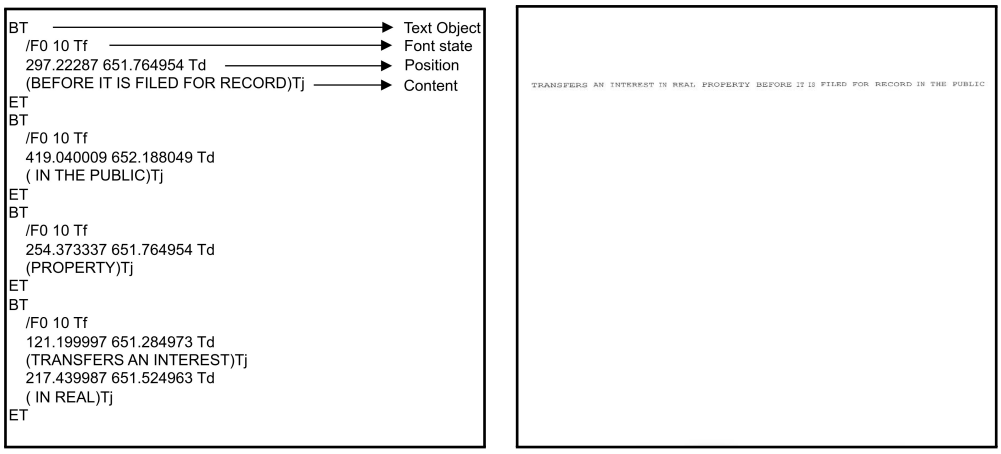
2.1
Portable Document Format

A Portable Document Format (PDF) [11] is a format file developed by Adobe to display a document in the same manner, regardless of the hardware and software used. Its simplest arrangement consists of storing a sequence of graphical objects describing where, how, and what will be displayed visually on the page. Figure 2.1 illustrates a toy example.

Because the purpose of a PDF is to ensure visual consistency and flexibility, there are no requirements regarding the objects' storage order [17]. The availability of an internally well-structured file and a human-readable order is optional and depends on the writing software used. On this subject, the files including this additional information are named Tagged PDF.

The format also includes other accessibility options that facilitate the navigation of the content. The most relevant for our work is the availability of a Table of Contents, which could reflect the resume's sections.

Despite this, many documents do not store this metadata. The reasons are many. For example, the writing software used may simply not have implemented these functionalities, or it may be impossible to generate them automatically due to the flexibility of the document's layout allowed by the software. Furthermore, besides the writing software, there is also reading software (e.g., PDF reader, Python package), which may ignore the metadata



(a) Internal Structure (b) Representation

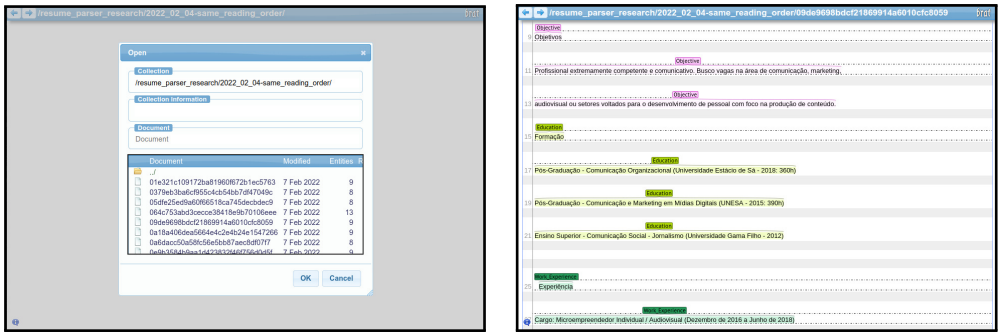
Figure 2.1: Toy example of the same PDF file represented from different perspectives. Figure 2.1a illustrates a segment of the PDF's internal structure, delineating the instructions for rendering textual objects on the screen. Figure 2.1b showcases how these identical objects appear when rendered by PDF reader software. Notably, these objects' internal storage sequence does not necessarily mirror the sequential manner in which a human would read the displayed content.

stored and display the document's reading order in the wrong order, even though it is available internally.

2.2 Annotation Tools

Annotation tools play a crucial role in the process of building datasets for supervised machine learning and natural language processing (NLP) tasks. They provide environments that offer tools to assist in all stages of the annotation process of the collected data — making the whole process more straightforward and faster. In the context of this thesis, two distinct annotation tools were employed: BRAT [18] and Label Studio [19]. These tools were chosen for text and image annotation, respectively.

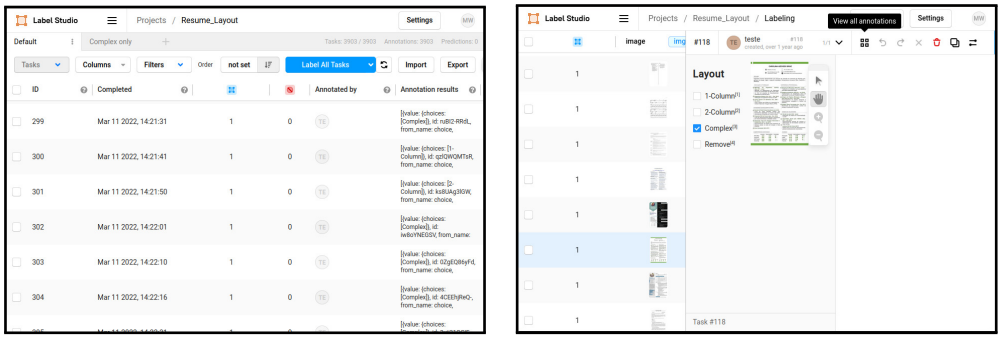
BRAT A specialized open-source software designed for text annotation tasks. It enables users to highlight and categorize entities and relationships within textual data, facilitating tasks such as named entity recognition and relationship extraction.



(a) Data Repository (b) Annotation Task

Figure 2.2: Screenshoots of the annotation tool BRAT.

Label Studio A versatile open-source annotation tool tailored for various data types, including text, images, audio, and video. It offers customizable annotation schemas, role-based collaboration, and active learning integration, empowering annotators to create accurate and diverse labeled datasets for machine learning and multi-modal analysis tasks.



(a) Data Repository (b) Annotation Task

Figure 2.3: Screenshoots of the annotation tool Label Studio.

While these tools were developed with different purposes in mind, it is noteworthy that BRAT and Label Studio share many essential features that make them reliable:

1. **Open-Source Nature:** Both BRAT and Label Studio are open-source solutions with strong communities and easy-to-set-up tools.

2. **User-Friendly Interfaces:** Each tool features a user-friendly interface, making them approachable and accessible to newcomers;

3. **Account-Based Access Control:** BRAT and Label Studio incorporate account-based systems that block access to the data by third parties and facilitate collaborative annotation;

4. **Web-Server Compatibility:** Both tools are designed to be deployed on web servers, allowing one to annotate data through any internet browser.

2.3 Document Image Classification

Document Image Classification is a task within the field of computer vision, focusing on categorizing images of documents into predefined classes based on their visual content. The goal of this task is to automatically assign labels to document images, enabling efficient organization, retrieval, and management of large collections of digitized documents. Figure 2.4 displays examples of different document images from the RVL-CDIP dataset [20].

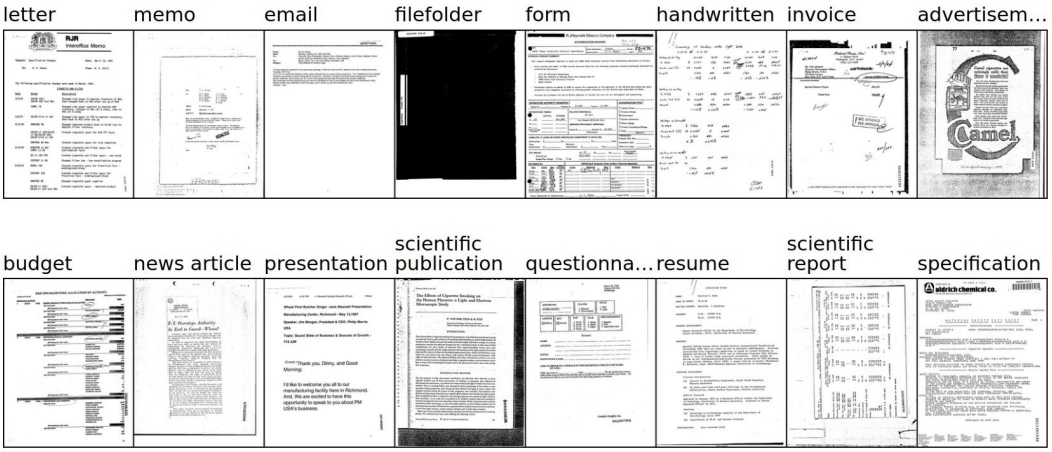


Figure 2.4: Examples of document images from different document types. Image retrieved from `https://adamharley.com/rvl-cdip/`

Formally, let tensor $m \in \mathbb{R}^{H \times W \times C}$ represent the input document image, where H is the height in pixels, W is the width in pixels, and C is the number of channels (e.g., grayscale or RGB channels). Additionally, assume we have a predefined document type set $d = \{d_1, d_2, ..., d_D\}$, signifying the D different document types that can be associated with a document image. The document image classification aims to predict a label $l \in d$ to the input image. Its objective is to find the label $l^* \in d$ that maximizes the conditional probability $p(l^*|m)$, as expressed by the equation: $\text{argmax}_t\, p(l|m)$. In simpler terms, the document image classification goal is to find the document type that has the highest probability of being correct given the observed input image, considering the visual patterns encountered within neighboring pixels.

In fact, image classification is one of the earliest tasks of computer vision, and as such, numerous well-established computer vision techniques naturally address it. In the context of this work, we employed EfficientNets [2], one of the most employed models for this task. In the remainder of this section, we briefly

explain how it works, starting from its base architecture, the Convolutional Neural Network.

2.3.1 Convolutional Neural Network

Convolutional Neural Network (CNN) is a deep learning architecture specially designed for image analysis and processing. It excels in tasks such as image classification due to its ability to automatically learn hierarchical features directly from pixel values. The fundamental concept of CNNs is first to detect local image patterns, such as edges, and then progressively combine these basic features to form more complex ones, such as shapes. To achieve this, CNNs incorporate the Convolutional and Pooling layers. Figure 2.5 illustrates these concepts.

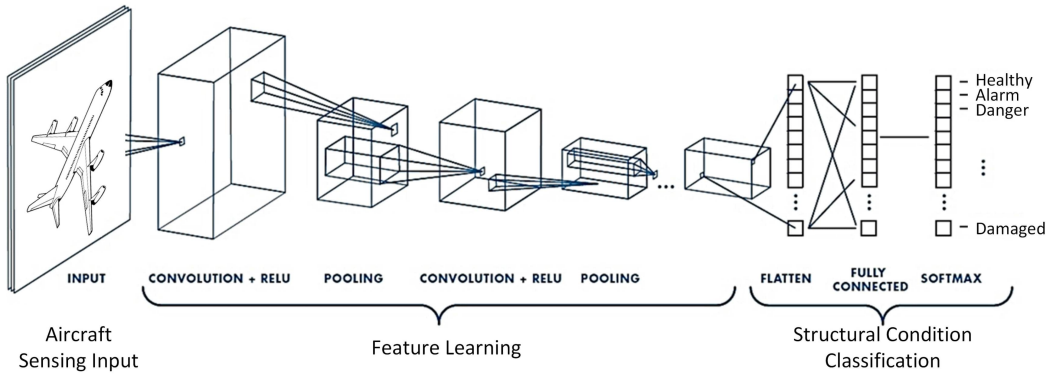


Figure 2.5: A simple CNN architecture [1].

The Convolutional Layer is responsible for detecting local patterns in images by sliding learnable filters over the input, generating feature maps representing specific features such as edges and textures. As layers deepen, the network captures increasingly complex features.

The Pooling Layer downsamples the generated feature maps by aggregating values within small adjacent regions. This downsizing not only enhances the model's computational efficiency but also improves its robustness against slight variations in the input.

After repeating this process multiple times, the model generates representative image features. These features are then flattened and input into a linear classifier to determine the image's category.

2.3.2 EfficientNet

While neural network models find extensive applications, they still struggle to determine optimal input size, the number of layers, and other hyperpa-

rameters. Assigning these values typically involves a trial-and-error approach or reliance on heuristics, which can prove time-intensive. Within CNN-based architectures, EfficientNet [2] addresses this issue by introducing a methodology that systematically scales the architecture based on available computational resources.

EfficientNet starts with a baseline model, EfficientNet-B0, and defines a set of operations that can be applied to it (such as adding more layers). They also establish a maximum limit on mathematical operations, representing the computational resources. By progressively increasing computational resources, they search for the most effective combinations of operations that fit within the augmented resources while yielding optimal performance on the widely-used ImageNet dataset [21] for image classification. This systematic methodology results in a family of eight models, ranging from EfficientNet-B0 to B7, each tailored to different resource constraints and demonstrating improved performance in various tasks.

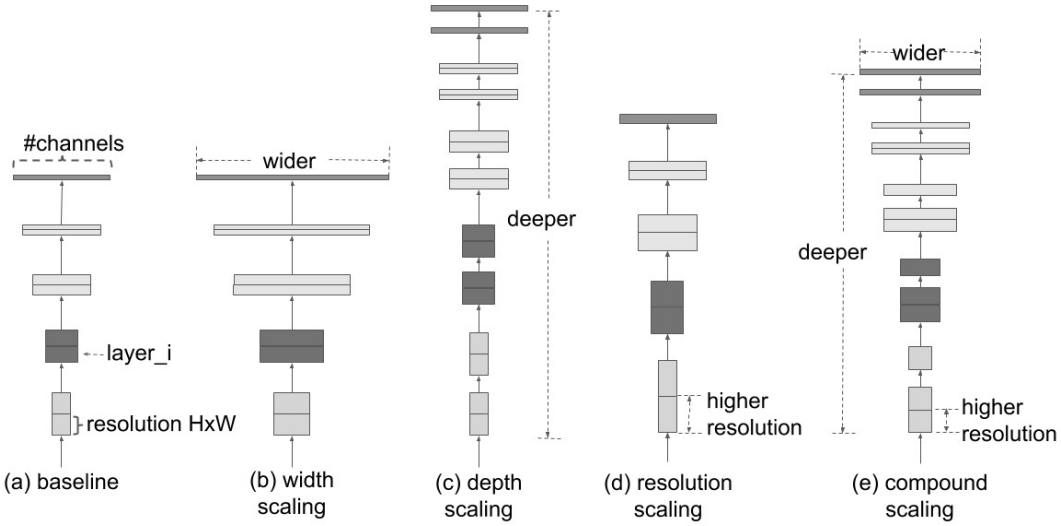


Figure 2.6: EfficientNet operations [2].

2.4 Sequence Tagging

Sequence tagging is a fundamental task widely used in NLP applications, involving the assignment of tags to a sequence of elements (e.g., words, lines). For example, Part-of-Speech (POS) tagging aids in grammatical analysis by categorizing each word's syntactic role, while Named Entity Recognition (NER) identifies and classifies entities like names of people, locations, and organizations. Additionally, the use of sequence tagging extends to more complex tasks like document segmentation, where it helps in outlining the section structures.

For these last applications, it is common practice to use a tagging scheme to represent entities and sections spanning multiple words accurately. For example, *IOB2* (Inside-Outside-Beginning) scheme [22] adds a prefix B to the first word of an entity while adding a prefix I to the remaining words. For the location "São Paulo", the word "São" would be tagged *B-Location* while "Paulo" would be tagged *I-Location*. This scheme also includes the tag O to be assigned to all words unrelated to any entity.

Formally, let $x = \{x_1, x_2, ..., x_N\}$ represent the input sequence consisting of N elements. Additionally, assume we have a predefined tag set $t = \{t_1, t_2, ..., t_T\}$, denoting the T potential tags that can be assigned to these elements. Sequence Tagging aims to predict a tag for each element in the input sequence, creating a corresponding tag sequence $y = \{y_1, y_2, ..., y_N\}$, where each $y_i \in t$. The objective is to find the optimal tag sequence y^* that maximizes the conditional probability $p(y^*|x)$, as expressed by the equation: $y^* = \text{argmax}_y\, p(y|x)$. In simpler terms, the sequence tagging goal is to find the tag sequence that has the highest probability of being correct given the observed input sequence, considering the contextual dependencies and relationships between neighboring elements.

Many popular NLP methods are used to address sequence tagging tasks. In the context of this work, we employed CRF [15] and BERT [16], arguably two of the most employed models for this task. In the remainder of this section, we briefly explain how each of these models works.

2.4.1 Conditional Random Fields

Conditional Random Fields (CRF) [15] are a type of probabilistic graphical model used for structured prediction tasks, where the goal is to make predictions about structured outputs. CRFs are particularly effective in scenarios where there are dependencies between the output tags. In the context of sequence tagging, Linear-chain CRF is a subtype of CRF specifically designed for dealing with sequences, where the dependencies between tags are constrained to their neighboring tags.

For modeling the $p(y|x)$, the Liner-chain CRF defines a set of feature functions $f = (f_1, f_2, ..., f_K)$ that capture the relationships between the sequence's elements and tags. Each feature function $f_k \in f$ computes a score based on the strength of a certain feature in each element $x_i \in x$, taking into account the tags y_i and y_{i-1} assigned to the current and previous elements.

Then, the scores of all feature functions are combined using:

$$score(y|x) = \sum_{i=1}^{N} \sum_{j=1}^{K} \lambda_j f_j(y_i, y_{i-1}, x, i).$$

Finally, the resulting score is transformed into a probability distribution over all possible tag sequences using a normalization function like the softmax.

Selecting useful feature functions plays a critical role in the CRF's performance. Thus, it is crucial to take time and define as many functions as possible that seem aligned with the task at hand. For example, considering the document segmentation application, we could employ indicative functions that verify whether a word is commonly used as a section heading or is uppercase.

2.4.2 Bidirecional Encoder Representations from Transformers

Bidirectional Encoder Representations from Transformers (BERT) [16] is a transformer-based model [23] that learns contextualized word embeddings [24] by simultaneously considering both the left and right neighboring contexts of each word in a sentence. These embeddings encode comprehensive information about word meanings in the context of a sequence, effectively extracting high-quality features directly from raw text and removing the need for manual feature engineering, as seen in the CRF model.

2.4.2.1 Model Architecture

It is out of the scope of this work to explain the intricacies behind a BERT model and how it models the $p(y|x)$ for the sequence tagging task. Yet, for a basic understanding of its functionality, we provide a simplified explanation of how BERT operates, emphasizing its two foundational components: subword tokenization [25] and the attention mechanism [26]. We also include a visual representation of the simplified BERT architecture in Figure 2.7 to enhance comprehension.

At first, BERT receives a sequence of words and breaks them into a sequence of subwords by employing the WordPiece tokenization [25]. The justification is that subwords enable the model to handle words not present in its initial vocabulary and effectively represent morphologically rich languages. For example, the word "segmentation" could be split into "segment" and "##ation", where "##" denotes a subword continuation. In the context of sequence tagging, to avoid introducing excessive complexity, the model could

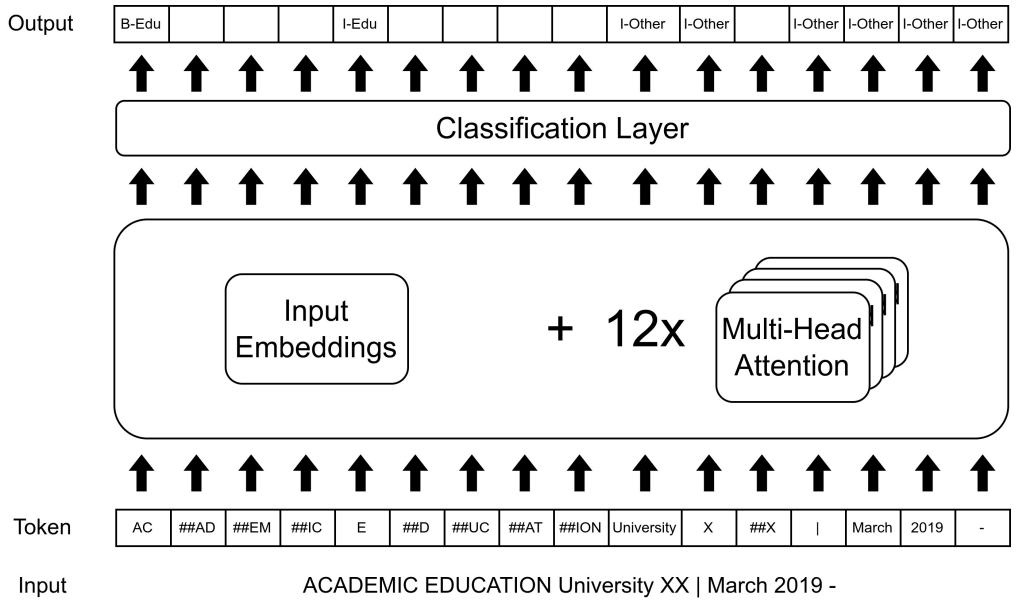


Figure 2.7: A simplified representation of the BERT architecture. WordPiece tokenization effectively balances the segmentation of words into subword units while preserving common words' integrity; this maintains words like "University" as complete units while breaking down words like "ACADEMIC". Multi-head attention employs several distinct attention mechanisms to simultaneously address diverse aspects of input text. The BERT architecture iteratively refines input embeddings through twelve layers before transmitting them to the classification layer. Empty entries in the output layer indicate that we ignored the predicted tag.

ignore subword continuations and assign the first token's output tag as the word's tag.

Once subword tokenization is complete, BERT assigns an embedding to represent each subword in a continuous vector space. These embeddings are then refined iteratively through the attention mechanism [26], which helps the model analyze how subwords relate to one another. For instance, in the sentence "John works as a developer at Google", the attention mechanism could enhance the representation of "developer" by incorporating its connections to "John" and "Google". By repeatedly refining these embeddings, BERT encodes contextual relationships between words within the sequence, culminating in what is known as Contextual Word Embedding.

Finally, BERT feeds these high-quality representations to some linear classifier in order to predict the outputs related to the NLP task at hand.

At first glance, this explanation of how BERT operates may resemble that of the Recurrent Neural Network (RNN) architecture with the Attention Mechanism [26]. However, while there are similarities between them when looking at these architectures from a simplified perspective, RNNs and Transformers differ significantly in terms of parallelism, scalability, and input size

handling. RNNs process sequential data step by step, lacking true parallel processing abilities. In contrast, Transformers like BERT process all data simultaneously, albeit constrained by a 512-token limit. This parallel processing capability accelerates training, empowering Transformers to attain larger dimensions and undergo training on more extensive datasets compared to RNNs, vastly improving its performance on the same NLP tasks.

2.4.2.2 Pre-trained Models

Furthermore, BERT offers the advantage of pre-trained models that encode intricate contextual relationships between words. This is achieved through prior training, where these models learn to predict missing (masked-out) words within sentences using vast amounts of unlabeled data. For example, in the sentence "What are [masked] doing today?" the expected prediction might include words like "you", "they", or "we".

This pre-training ensures the model already possesses contextual understanding, requiring downstream tasks to just fine-tune it on specific datasets to better focus on the task at hand. This accessibility to pre-trained models significantly streamlines the creation of highly effective models, eliminating the necessity of training from scratch.

3
Related Work

Within this chapter, our focus is directed toward a comprehensive survey of the pre-existing approaches to resume parsing. These efforts are divided into three distinct categories: (i) Traditional, encompassing literature predating the emergence of Contextual Neural Networks; (ii) Modern, encapsulating recent strides within the field, which primarily emphasize the integration of Transformer models into resume parsing; (iii) Enterprise Solutions, encapsulating commercially available resume parsing tools that are shrouded in proprietary details. Furthermore, the subsequent sections of this chapter provide succinct reviews of pertinent works that have surfaced in the domains of Reading Order Detection and Text Segmentation tasks.

3.1
Resume Parsing

Comparing the various resume parsers proposed in the literature poses challenges due to the absence of a standardized benchmark dataset. Each tool adopts its own perspective on the relevant information and guidelines for data annotation, resulting in datasets with distinct characteristics. Furthermore, we have not found any publicly available dataset or source code related to the works in the literature, possibly due to privacy and intellectual property concerns. As a result, rather than directly comparing metrics, it becomes more relevant to understand how the available approaches handle document data, and metadata and adapt existing models from related problems.

3.1.1
Traditional Approaches

As previously mentioned, most works [7, 8, 9, 27] focus on extracting key sections and entities from resumes using a two-step pipeline as illustrated in Figure 3.1. The first step involves identifying the sections of interest, while the second step consists of extracting specific entities within them.

These two steps are typically formulated as sequence tagging tasks, and because of that, they utilize similar algorithms and attributes. Algorithms such as Hidden Markov models (HMM) [7], Conditional Random Fields

(CRF) [8, 9, 27], Convolutional Neural Networks (CNN) [9], and Bidirectional Long-Short-Term-Memory - CNN (Bi-LSTM-CNN) [9] have been employed. Regarding attributes, neural-based models [9] rely on Word Embeddings exclusively. In contrast, other models [8, 27] utilize ontologies that encompass the most frequent headings and terms across different sections, visual metadata (e.g., font size, bold formatting), and textual elements (e.g., "Is it a number?", "Is it all caps?") within the documents.

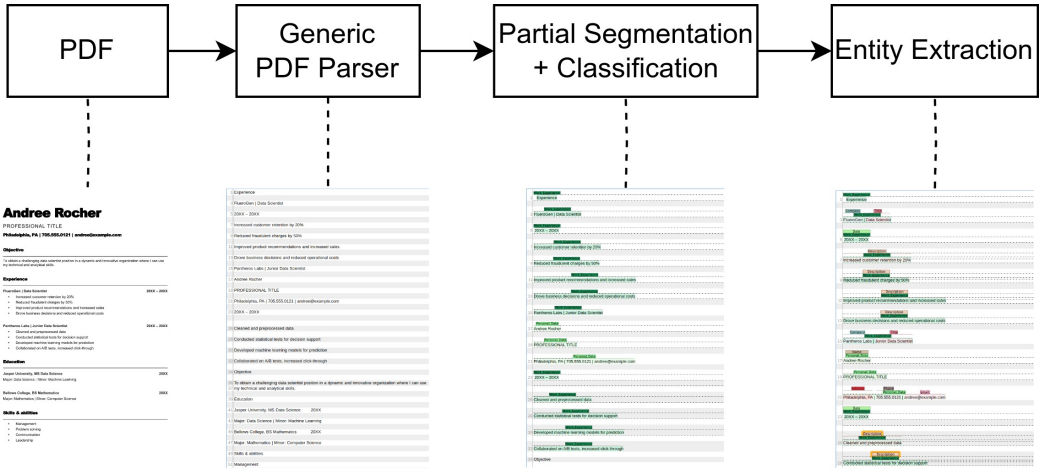


Figure 3.1: Example of a basic pipeline for parsing resumes. The primary objective of most of the proposed approaches is to extract the relevant entities. Section identification serves as a means to filter the text portions that the models should scan for improving precision.

3.1.2 Modern Approaches

Recent works [10, 13, 28, 29] focus on expanding the extracted information from resumes, primarily leveraging Transformers [23] like BERT [16].

Vukadin et al. [13] propose a single multi-objective and multilingual BERT model capable of extracting main sections and entities in five languages simultaneously. They also propose a second multilingual BERT model to extract the competency level associated with skill entities.

The work that is closest to ours is by Pawar et al. [28], where the goal is to group entities associated with the same work experience (or education). They develop an ensemble model that combines rule-based, CRF-based, and neural-based models to extract entities and determine lines associated with the same item. However, this approach has limitations, including the lack of detailed information on the manually extracted features employed. Additionally, the study focuses primarily on assessing the number of related entities identified, with no comprehensive evaluation of segmentation quality. Notably,

they remain the only study employing traditional Word Embeddings instead of contextual Word Embeddings, citing the significant computational costs associated with using models like BERT in practical scenarios.

On the other hand, the works of Gaur et al. [29] and Barducci et al. [10] are complementary to ours. They use proprietary tools to obtain the *Education*, *Work Experiences*, and *Skills* sections and concentrate on a more comprehensive extraction of information within these sections.

It is noteworthy to mention that Barducci et al. [10] were the first to highlight the challenges of extracting information from PDF resumes with "complex" templates such as *2-Column*. They specifically cite the issue of non-sequential text in these resumes.

3.1.3 Enterprise Solutions

Despite these research efforts, several practical aspects still require attention, such as handling documents with complex layouts and accurately identifying individual education and work experience items. While academia has yet to explore these areas extensively, commercial applications developed by companies like Affinda and Nanotecs address these points. However, the details of these solutions and their associated metrics remain proprietary.

Based on the brief descriptions of the technologies employed [30, 31], it is evident that their architectures seem more complex than the one illustrated in Figure 3.1. For example, they introduce an initial component for Document Layout Analysis [32] to decompose the original document into blocks and arrange them in the most appropriate reading order. They also enrich components already explored in the literature, covering more information while employing Contextual Word Embeddings [16].

3.2 Reading Order Detection

The challenge of determining the reading order in PDF documents is not new. Meunier et al. [3] proposed one of its first solutions by segmenting document pages into rectangular blocks and optimizing their grouping based on geometric attributes to establish the reading order.

In recent years, Wang et al. [33] introduced ReadingBank, a dataset with reading order, text, and layout information for $500,000$ DOCX documents in English, facilitating the utilization of deep neural networks for reading order detection. Their proposed model, LayoutReader, utilizes a seq2seq architecture [34] with transformers for accurate reading order prediction. One limitation of

this approach is that it relies on DOCX documents which typically impose layout constraints. Another limitation is that it does not guarantee that all tokens will be ordered since it uses a seq2seq architecture.

Furthermore, to our knowledge, Gu et al. [35] stands as the only work that evaluates the influence of reading order on subsequent tasks, specifically, the extraction of entities and semantic relations from scanned forms. In contrast to [33], they adopt a heuristic approach similar to [3], aiming to group close textual elements to enhance contextual clarity while extracting entities and relationships of interest.

3.3 Text Segmentation

The literature offers several approaches [36, 37, 38, 39] to address text segmentation (and classification). These methodologies typically start with a sentence encoder, which derives a vector representation of each line based on word embeddings. To accomplish this, they employ techniques such as weighted averaging [36], RNNs [37, 39], or transformers [38]. Subsequently, the encoded sentences undergo further processing through other RNN [36, 37] or Transformer [38, 39] models to obtain refined sentence representations that take into account the surrounding context. Finally, with a list of enriched sentences in hand, each approach presents a unique strategy on how to segment the text (and classify the segments obtained).

One noteworthy approach by Arnold et al. [36] closely resembles the strategy applied in resume parsing. It involves assigning tags to each line, and text segments are identified by tracking tag changes between consecutive lines. In contrast, Barrow et al. [37] segment the text first before performing classification. Finally, both Lo et al. [38] and Bai et al. [39] propose a simultaneous approach where the models assign a topic to each line and detect whether the same is a segment boundary.

An interesting aspect to highlight is that, unlike resume parsing, there exists a standardized dataset [36] based on the Wikipedia section structure, which serves as a common benchmark for evaluating these approaches. Notably, this dataset intentionally omits section headings to challenge the segmentation of text. Moreover, in addition to employing classification metrics, these studies utilize the p_k segmentation metric [40] to assess the alignment between predicted segment breaks and annotated segmentation.

4
Our Proposal

This chapter presents the proposed resume parser designed to organize the unstructured text extracted from a resume file into a semi-structured object that reflects its original section structure. We provide a brief overview of the parser's pipeline. Then, we detail the techniques it employs to ensure the accurate reading order of the text and identify the sections and subsections within a resume.

4.1
Pipeline

Figure 4.1 illustrates our resume parser pipeline, which consists of three core components. For the PDF parser component, we simply use PDFBox [12], a popular open-source library for manipulating PDF files under the Apache Software Foundation. In contrast to other resume parsers, we add a pre-processing component between the PDF parser and the section segmenter to ensure the correct reading order of the text retrieved; this is detailed in Section 4.2. In addition, we modify the segmenter component, already found in other approaches, to simultaneously handle the segmentation of sections and subsections; this is detailed in Section 4.3. To avoid overuse of some terms and better distinguish between these two types of information, we use first- and second-level information to refer to the sections (e.g., Personal Information, Work Experience) and items (individual education and work experience subsections).

Our parser does not identify entity-level information, such as degrees and job titles. That said, we understand that our components can be easily coupled with other well-established approaches, given the modular design commonly adopted by most resume parsers. After segmenting a resume, we could use the resulting sections as inputs to the models proposed in [10, 29] since they are specialized in extracting entities from one section at a time. Alternatively, it is also doable to create a single model that simultaneously extracts all levels of information by combining our tagging schema, detailed in Section 4.3.2, with the tagging schema proposed in [13]. However, this combination is limited to Transform-based models, given their multi-objective optimization capabilities.

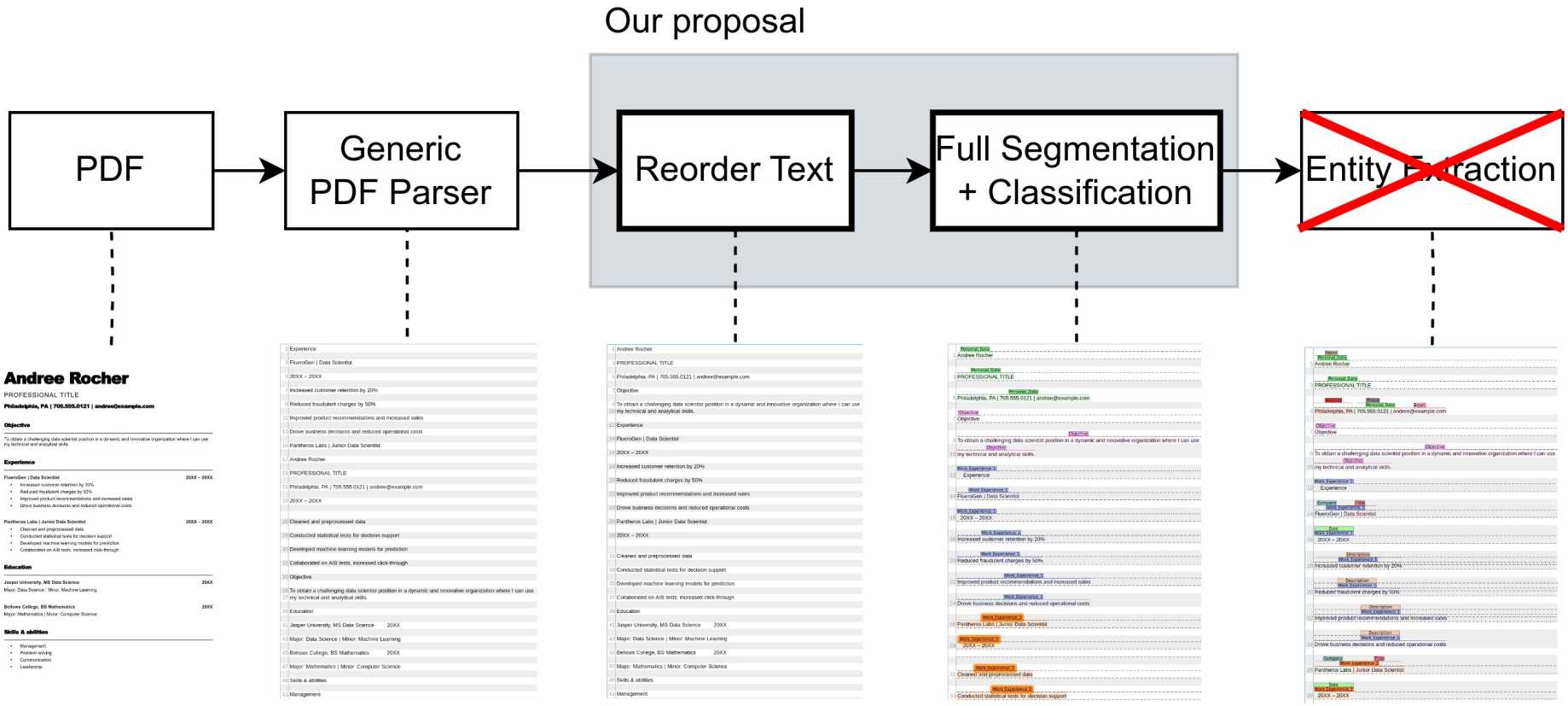


Figure 4.1: Our resume parser pipeline. The idea is to organize the unstructured text extracted from a resume file into a semi-structured object that reflects its original section structure. Our parser does not identify entity-level information, such as degrees and job titles. That said, we understand that our components can be easily coupled with other well-established approaches, given the modular design commonly adopted by most resume parsers.

4.2 Reading Order

Correctly retrieving the text from PDF documents can be challenging because the format was primarily designed to preserve the visual position of the document's elements (e.g., text and image), which means that documents are not required to store their elements in a logical order [17]. As a result, document parsers implement rules-based algorithms [41, 42] to infer the most appropriate reading order of the text extracted based on its layout. However, the failure rate of these heuristics seems to increase with the complexity of the document's layout.

As displayed at the beginning of this thesis, Figure 1.2a shows a document where the parser failed to extract the text correctly. Although the change in the reading order seems small, it can have significant consequences when extracting information. The document in question, for example, may end up not extracting the section *Objective*.

To mitigate this problem, we propose a template identifier for resume pages. We model the problem of identifying a template as an *document image classification task* [20] where each page template is considered a different document type. We work with the assumption that most resume layouts derive from a few well-defined templates with easy-to-extract reading orders. Therefore, if we map the most used templates and identify which one a page uses, we can employ a heuristic to reorder its elements to ensure they follow the correct reading order.

4.2.1 Layout Complexity

The problem of determining the reading order of documents can be comprehended as first identifying textual elements whose reading orders are known (e.g., paragraphs) and then ordering them in a way that is reasonable for human readers. Different layouts use different amounts of textual elements, and it is plausible to assume that the more they are present on a page, the more difficult it becomes to organize them. Hence, a layout's number of textual elements can indicate its complexity.

To quantify this concept, we just consider resumes that consist exclusively of textual blocks – rectangular-shaped textual elements whose content is structured to be read from left to right and top to bottom. By imposing this constraint, we can distinguish each textual block's location through a series of vertical and horizontal cuts in the blank spaces across the page. For instance, in a page with two columns, inserting a vertical line between them allows

us to differentiate between blocks in the left and right columns, resulting in two sub-regions that can be further subdivided through additional cuts. The application of a sequence of n such cuts generates a list of $n+1$ textual blocks.

It is important to note that our assumptions do not account for all resume layouts. Specifically, it cannot assess pages that employ "L-shaped" content organization, as shown in Figure Figure 4.2. However, we argue that such elaborate visual arrangements are uncommon in resumes, and our approach is designed to cover the majority of resume layouts effectively.

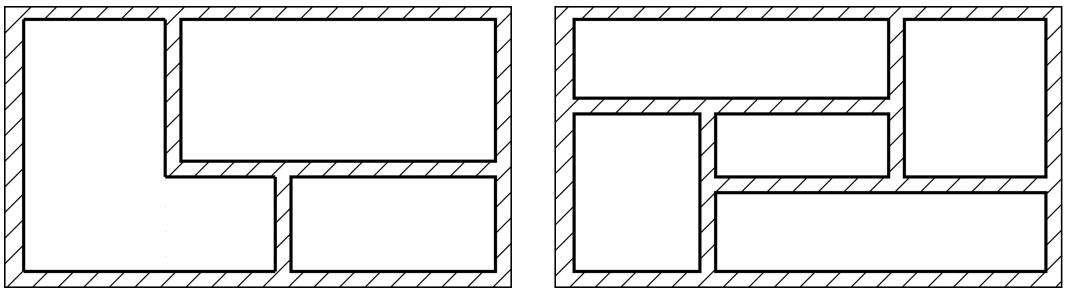
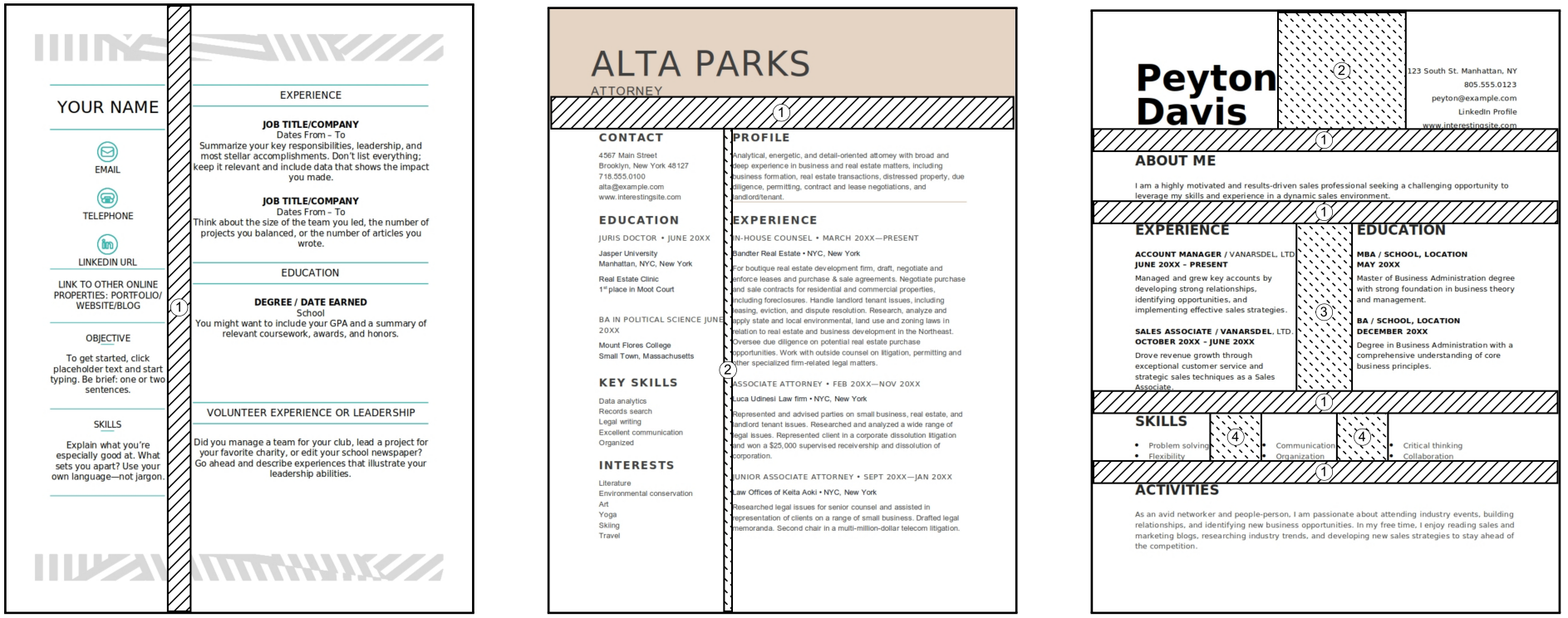


Figure 4.2: Examples of text elements organized in a L-shape format [3].

In this context, we define layout complexity as the minimum number of cuts required to obtain the fewest number of text blocks on a page. The complexity level of a page corresponds to the number of cuts required +1. Figure 4.3 provides a visual representation of different levels of layout complexity based on this definition.



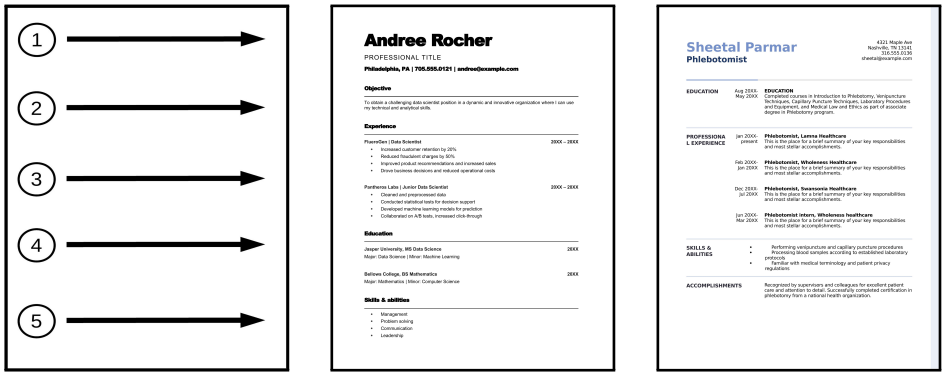
(a) Level 2 (b) Level 3 (c) Level 9

Figure 4.3: As illustrated in Figures 4.3a, 4.3b, and 4.3c, the reading order complexity of a page can be associated with the minimum number of cuts required by their respective layouts to allocate each text block present in a distinct page region. Here the complexity level of the layout is equivalent to number of cuts + 1. Level 1 is omitted, given it would be just the entire page as a single text block.

4.2.2 Template Types

Based on our previous discussion, in this thesis, we focus on identifying the most common templates used on resumes: 1-*Column* and 2-*Column*, which correspond to all page layouts of complexity levels 1 and 2. For clarity, we define a given page as:

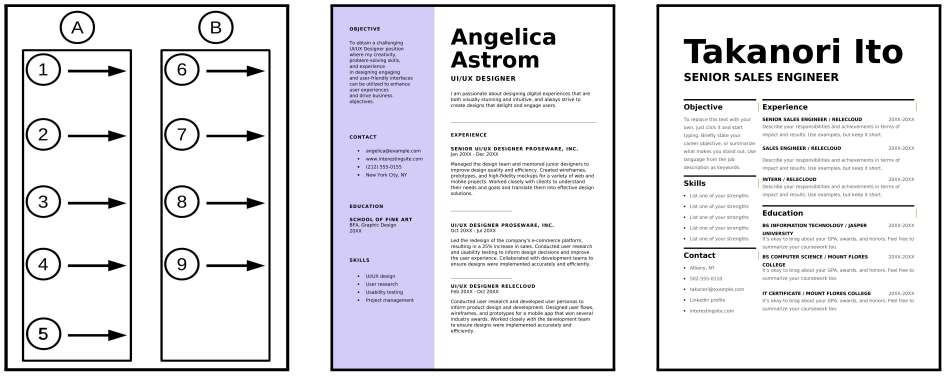
– 1-***Column*:** If we can correctly read the **entire content** of the page starting from the top left corner and, from there, following from left to right and from top to bottom. Figure 4.4 illustrates this procedure;



(a) Reading order (b) Correct (c) Wrong

Figure 4.4: Figure 4.4a depicts the reading order that defines a *1-Column* template. Figure 4.4b is an example that follows this reading order. Figure 4.4c is an example that does not follow it since the Name and Surname would be read separately given the other contacts in the upper right corner.

– 2-***Column*:** If we can find an empty block vertically crossing the **entire page**, splitting the existing text into two independent columns, in which we can apply the procedure described for 1-Column. Figure 4.5 illustrates this procedure;



(a) Reading order (b) Correct (c) Wrong

Figure 4.5: Figure 4.5a depicts the reading order that defines the 2-*Column* template. Figure 4.5b is an example that follows this ordering while Figure 4.5c is an example that does not, as the page header makes it impossible to divide the text into two columns.

– ***Complex*:** If a page does not fit into the above categories.

As a quick note, it is worthwhile mentioning that we assume the text is represented by a sequence of words. Hence, we are considering a word-by-word reading to define these templates. However, the same approach can also be used when representing the text as a sequence of any other textual block type, such as sentences, paragraphs, or text segments obtained through a *document layout analysis task* [32]. The only critical aspect is that the chosen type follows the reading patterns defined.

4.2.3 Reconstruction Step

The "reconstruction" step replicates the visual reading order in the data reading order. In the 1-*Column* case, this means reordering the words according to their coordinates on the y and x axes, respectively. The 2-*Column* case is analogous with an additional pre-processing step where the **widest** vertical empty block is detected. The content is then split into two blocks: one on the left of the empty block and the other on the right. During reordering, words on the left precede those on the right. We did not develop a heuristic for pages classified as *Complex* since this class represents all unmapped reading orders.

4.2.4 Impact of Templates in the Reading Order

Here we discuss how robust the text reading order retrieved by popular parsers is. For that, we perform a simple experiment exploiting our defined

templates and the expected number of y-axis line inversions occurring on each one of them.

Given a line sequence $(l_1, l_2, \ldots, l_n)$, obtained as an output of some PDF parser, we define a y-axis inversion as any instance where $y_i > y_{i+1}$ for two consecutive lines l_i, l_{i+1} [1]. Note that we are assuming the origin $(0, 0)$ is at the top-left corner of the page Thus, a *1-Column* template is expected to have zero inversions, while a *2-Column* template is expected to have only one inversion - the column break. Thus, the idea for our experiment is to analyze how well a parser handles *1-Column* and *2-Column* pages by extracting the texts of several pages with these templates and comparing the number of inversions obtained with the ones expected. For this experiment, we employed one of the datasets created in this work, detailed in Section 5.2.2, which consists of $3,638$ pages manually annotated according to the defined templates. Notably, we only consider lines containing at least 5 characters to mitigate possible inversions caused by noisy lines in the resumes.

Table 4.1: Number of y-axis inversions for two commonly used document parsers (in %). PDFBox returns the lines in the order they were stored, while PDFMiner applies a heuristic to reorder them based on their spatial locations.

Parser	PDFBox			PDFMiner		
Inversions	1-Column	2-Column	Complex	1-Column	2-Column	Complex
0	70.60	0.17	7.88	54.77	1.56	9.24
1	20.94	39.24	29.57	24.98	40.45	32.20
2	4.72	16.67	20.14	9.73	25.52	20.91
3+	3.74	43.92	42.41	10.52	32.47	37.65

Table 4.1 shows the results for two well-established parsers, namely PDFBox [12] and PDFMiner [43]. As an example, PDFBox reconstructed 70.6% of the resumes annotated as *1-Column* without inverting any line while only 39.2% of the resumes annotated as *2-Column* presented one inversion. We observed that many pages had more inversions than they should have for both tools. By manually inspecting some of them, we noticed that many inversions seem negligible from a practical point of view. These are cases like the page's footer going to the middle of the text or semantically independent chunks changing order, such as *Name* and *Telephone* lines. Nonetheless, there are also critical cases where the parsers positioned all section headings at the end of the page or moved one work experience into another section.

[1] With a slight abuse of notation, we reuse the symbols l and y from Chapter 2 to denote a line and its y-coordinate, respectively.

Of course, with this approach, it is impossible to accurately measure the proportion of resumes whose texts have been altered so significantly as to jeopardize the understanding of their content. Our main intention is to convey that potential mistakes during the section segmentation and information extraction steps could be due to problems in the input text rather than the respective models and that assuring the correct reading order of the text retrieved is desirable.

4.3 Resume Segmentation

Once the proper reading order of the text has been ensured, the next step is segmenting the text regarding both the first and second levels of information. Our proposed methodology treats each individual work experience and education item as an independent section, allowing us to formulate the problem as a conventional *sequence tagging task*.

We deviate a little from a standard implementation by introducing a minor modification in the tagging scheme to enable the model to distinguish the initial item in each section from the subsequent ones; this adjustment is necessary as the initial item is encountered in a distinct context and aids in a clear differentiation between the first and second levels. Using this strategy, we can retrieve the entire *Work Experiences* section by concatenating the adjacent work experience items. For the *Education* section, the process is analogous.

We implemented CRF- and BERT-based approaches using this methodology to analyze the performance differences between them. Our motivation is to verify whether BERT is justifiable for our task, despite its significantly higher computational requirements compared to CRF; or if the CRF can be a practical and viable alternative when computational constraints are a concern.

4.3.1 Section Types

Our resume parser extracts all sections written in the resume. However, because it is laborious and arguably infeasible to have the names of all possible sections, our models explicitly tag only *Personal Information*, *Objective*, *Summary*, *Education*, and *Work Experiences* sections in this study. This choice has to do with the high frequency of these sections in resumes. Any other identified section is named as *Other*. For the sake of clarity, we explain the expected content of our chosen sections:

- ***Personal Info*:** Basic information about the candidate, such as their full name, contact details (email, phone, and web page), and address.

Matheus Silva
Brazilian, Single, 30 years old
Rio de Janeiro, RJ
Phone: +55 (21) 98888-7777
E-mail: matheus@example.com

– ***Objective*:** Outlines the candidate's aspirations, career goals, and the job position they seek.

Intended Position: Work on Digital Marketing activities: content marketing, inbound marketing, and metrics analysis.

– ***Summary*:** Highlights the candidate's relevant skills, experience, and accomplishments;

About Me
I have an IT Support Professional Certificate. Experience as a Computer Technician working with software and hardware installation, computer assembly, and hardware recommendation.

– ***Work Experience*:** Refers to a candidate's past employment. A job typically includes the job title, company name, employment dates, and a description of the candidate's responsibilities and accomplishments;

Professional Experiences:

Financial Analyst, Company XX
Sep/2017 - Jan/2018
* Analyze current and past financial data and performance.
* Prepare reports and projections based on analysis.

– ***Education*:** Refers to a candidate's academic background. An education typically includes the degree earned, the institution name, and the graduation date;

ACADEMIC EDUCATION
University XX | March 2019 - December 2022
Bachelor's degree in Business Administration

– ***Other*:** Any additional information the candidate wants to include in their resume that did not fit into the previous sections. That might include volunteer experience, certifications, awards, or references.

Complementary Courses and Knowledge
* English – Intermediary
* Spanish – Intermediary
* Excel / VBA and MACRO – Advanced

4.3.2 Tagging Scheme

Using a naive approach, we need at least two models to obtain a resume's first and second levels of information. The first model extracts all the resume sections; the second model receives a single section and splits it into individual items. The first model requires classifying each line as part of a specific section, so we can later extract each section. In contrast, the second model only requires determining where each item begins through a binary classification. A *True* tag indicates the beginning of an item, while a *False* indicates the continuation of the last item.

To avoid implementing these models, we can apply a minor tweak to our tag encoding to have a single model that simulates the same core behavior. As already stated, the first step to do that is to classify each item of a section and the other sections simultaneously by considering each item as a section in itself. After that, when we encode our annotations, we have to differentiate the first item of the section from the remaining ones.

We initially encode all annotations using the standard *IOB2 scheme* [22], meaning the annotation's first line adds a prefix *B* to the section type while the remaining lines add a prefix *I*. We do not use the tag *O* because our annotation cover all resume. Then, we append an *Item* suffix (inspired by the *BILOU scheme* [44]) to all items other than the first to differentiate them, which should be equivalent to the *True* tag of the binary classification. Table 4.2 illustrates and differentiates our tagging scheme from the standard scheme.

Table 4.2: A comparison of the standard *IOB2 scheme* and our modified implementation.

Line	Standard	Ours
skills to drive innovation.	I-Summary	I-Summary
Education	B-Education	B-Education
2020 - 2022 XXX University	I-Education	I-Education
M.Sc.in Computer Science	I-Education	I-Education
2016 - 2020 YYY University	B-Education	B-Education-Item
B.S. in Computer Science	I-Education	I-Education
Skills	B-Other	B-Other

4.3.3 Segmentation Models

To segment resumes, the CRF and BERT models take a sequence of words as input and assign tags to them as output, thus generating the segmentation.

However, we enforce that both models assign a single tag type per line to ensure that no section changes occur inside a line since it is reasonable to assume that these shifts only occur when moving from one line to the next.

In the remainder of this chapter, we describe the specific adaptation required by each model to handle the resume segmentation task.

CRF Architecture

For the CRF-based approach, we convert the sequence of words into a sequence of lines and use them as input to the model. Working with lines is more practical since we must manually extract attributes from the input to feed the model. It allows us to select higher-level attributes that better align with our segmentation task. When merging the words into a line, we must also "merge" their metadata. For simplicity, we use the majority rule for all properties.

Table 4.3 describes the attributes selected, which can be divided into four categories: Ontology, Text, Visual, and Spatial. Each category covers a type of information encoded in the resumes that can be exploited to help us detect and differentiate the sections within them. It is noteworthy to highlight that most of our attributes are geared toward identifying section breaks. The identification of the section type is given solely by the attributes of the Ontology category.

Ontology category As the name suggests, represents the attributes based on an ontology, examining whether any token in the line matches some typical section heading or tokens commonly presented in specific sections. Most sections contain particular terms that are only present in them. For example, the *Work Experience* section typically has "Professional Experience" (or simply "Experiences") as the heading of the section, while "position", "company", and "job activities" often precede prominent information in it.

We implemented a normalization step to improve token matching between our ontology and the text. These included lowercasing all text to ensure case-insensitive matching, stemming to reduce words to their root forms for better recognition of word variations, replacing all numerical values with "0" to standardize numerical representations, and removing stopwords and punctuation marks. As a result, a sentence like "WORK EXPERIENCES: Developer from 2018 to 2020" would be transformed into "work experi develop 0000 0000".

To obtain the Resume Headings vocabulary, we examined 100 resumes[2]

[2]It is necessary to note that none of the resumes used in this examination are present in the dataset described in section 5.3.1 to evaluate the model's performance.

Table 4.3: The manually extracted features used by the CRF model. The term *default* refers to each resume's most frequently found value for the given attribute.

	Ontology (*for each section type)
1	Percentage of section header tokens
2	List of common tokens in section segment
	Text
3	Percentage of uppercase tokens
4	Percentage of lowercase tokens
5	Percentage of titlecase tokens
6	Has year
7	Has number
8	Has punctuation
9	Has punctuation at end
10	Has colon
11	Has colon at end
12	$\lvert$Tokens$\rvert \leq 2$
	Visual
13	Is Bold
14	Is Italic
15	color = default
16	fontsize > default
17	fontsize < default
18	fontsize = default
19	Is same style as previous line
20	Is same style as next line
	Spatial (*w.r.t previous / next line)
21	Has indentation level changed
22	$y^{gap} - \mu_{y^{gap}} > \sigma_{y^{gap}}$
23	$y^{gap} - \mu_{y^{gap}} < -\sigma_{y^{gap}}$
24	$\lvert y^{gap} - \mu_{y^{gap}} \rvert < \sigma_{y^{gap}}$
25	$y^{gap} >$ default
26	$y^{gap} <$ default
27	$y^{gap} =$ default

and carefully selected the most prevalent headings for each section type. The selected headings are listed in Appendix A.

On the other hand, we opted to employ an automated approach to obtain the section body vocabulary due to its extensive and diverse nature compared to the prior vocabulary. To achieve this, we leveraged the χ^2 test [45], a widely used method in machine learning for feature selection. In simple terms, the χ^2 test helps identify the terms (unigrams and bigrams only) that better represent a specific section by considering their occurrence patterns; they should appear prominently only in that particular section.

Text category Textual attributes consisting of common text patterns. Lines containing section headings tend to be uppercase and short if we ignore stopwords. On the opposite side, lines containing inner section content are mostly lowercase and long. Detecting date periods and punctuation is also relevant to identifying second-level information since each education and work experience usually specifies a date period representing its start and end.

Visual category Visual attributes embedded into the words, identifying elements such as bold, italic, and font size. It is common for section headings and other remarkable information on resumes to have distinct visual attributes to make them stand out. However, for cases such as font size, it is essential that we first identify the size being used as default to determine which lines deviate from it.

Spatial category Spatial attributes that measure the distance between the current and previous/following lines regarding the y and x axes, as illustrated in Figure 4.6. It is reasonably common for resumes to have a large gap between the end of a section and the start of a new one. The same reasoning can be applied when starting a new education or work experience. Like the strategy employed for the font sizes, we must first identify the most common gap size between one line and another to determine which gaps deviate from it.

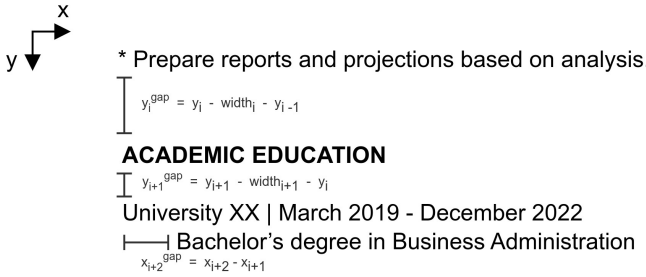


Figure 4.6: How to compute the y and x axes gap between consecutive lines.

BERT Architecture

The BERT-based approach does not require manual attribute selection. However, it requires pre-processing and post-processing steps to allow BERT to deal with lengthy texts, given its 512 input token limitation. On top of that, it also requires a second post-processing step to deal with the lack of cohesion between adjacent predictions. In the CRF-based approach, the prediction of line i depends on the prediction of line $i-1$. Therefore, the model naturally restricts a line to be classified as *I-Education* only if the previous line was classified as either *I-Education* or *B-Education*. This does not occur for BERT because the lines are all processed in parallel.

Pre-processing We first tokenize the input word sequence using the WordPiece tokenizer [25]. If the number of generated tokens exceeds 512, we partition the token sequence into independent subsequences. We apply a 128-token overlay between the previous subsequence's end and the next subsequence's beginning to improve context awareness between adjacent subsequences. This overlay does not undergo classification to avoid bias during training. As a result, the first subsequence can classify up to 512 tokens, whereas the subsequent ones can classify up to 384 tokens.

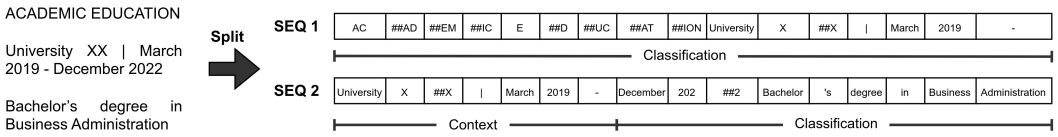


Figure 4.7: The WordPiece tokenizer converts the input text containing 15 "words" into a sequence of 25 tokens. Suppose we employ a toy BERT model with a token limit of 16 and consider a context size of 7 tokens. In that case, we must partition the token sequence into two minor subsequences and process them separately. The first sequence processes the first 16 tokens, while the second processes the remaining 9 tokens. However, we append 7 context tokens to the beginning of the later subsequence to improve context awareness.

Classification BERT processes those input subsequences separately and outputs tags for each one of them. To ensure that the model only outputs one tag type per line, inspired by Vukadin et al. [13], we introduce a *New Line* index that sets to true only the first token of each line, excluding those in the contextual area. This index ensures that the model only takes into account those tokens' outputs during classification and only backpropagates them to update the model weights during the training phase. All remaining

tokens output the same tag as the first in the line and are ignored during backpropagation.

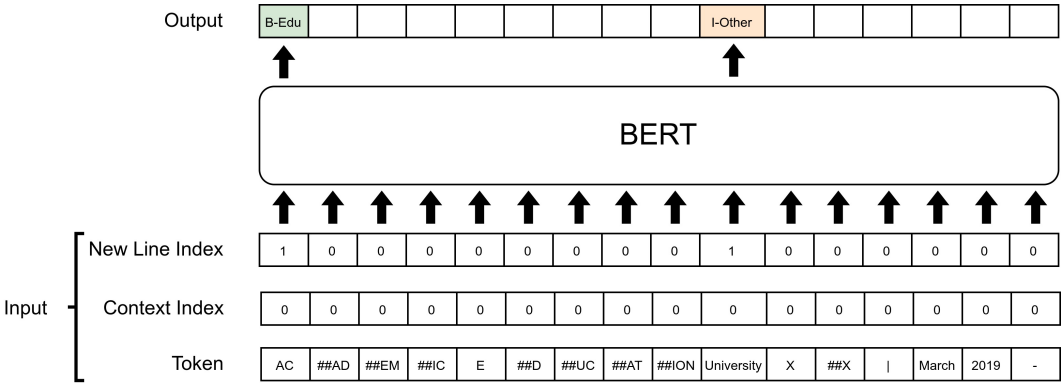


Figure 4.8: In addition to the token sequence, we add newline and context index sequences as input to BERT. This additional information allows us to control which outputs to consider during the training and inference phases.

Post-processing If there is more than one subsequence, we drop all subsequences' contextual windows and concatenate them to obtain a unique sequence once again. Finally, the last step is to ensure the output segmentation is valid. Since BERT outputs are independent, it can output a tag *B-Education* followed by a tag *I-Other*, which is undesirable. To circumvent that, we employed the following heuristic: (i) For each token tag, recover its prefix (*B* or *I*) and section type (*Other*, *Work Experience*, ...); (ii) Split the output sequence into a list of segments based on the prefix *B*; (iii) For each segment, assign its section type as the most frequent one in number of lines.

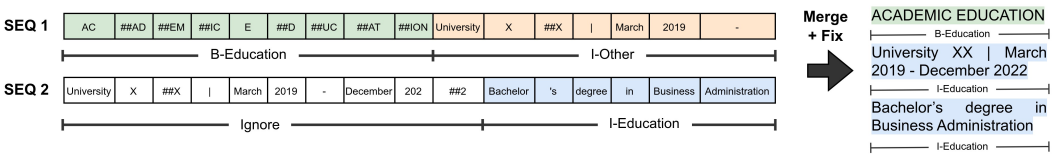


Figure 4.9: After processing each subsequence, we translate the outputs generated back to the original format. First, we drop the **SEQ 2**'s contextual window and then attach it to the end of **SEQ 1**. The output of the second line present in **SEQ 2** is also ignored in this process because the line was already classified in **SEQ 1**. Finally, once we have the original sequence, we apply post-processing to ensure the tag sequence is consistent.

5
Experiments

In this section, we present the results of the experiments conducted for the template classification and resume segmentation tasks related to our proposed resume parser. For both of these tasks, we provide details about the dataset used, the experimental setup, and a comprehensive analysis of the performance observed in our developed models.

5.1
Initial Considerations

Likely due to privacy concerns, we have not found publicly available datasets for resume parsing. All resumes used in our study were provided by an HR Tech company based in Brazil and, subsequently, manually annotated by us to construct datasets tailored to each task. Because of this, the section related to each task includes a brief description of the creation process of the dataset used.

We performed the experiments for both tasks five times, randomly dividing the datasets into training, validation, and testing sets with a ratio of 60/20/20 for each iteration. The tables in this section report the average results of the test sets accompanied by their corresponding standard deviations.

All models were implemented in PyTorch [46] and executed on a single core of an Intel(R) Core(TM) i5-12400 CPU @ 4.40GHz, equipped with 32 GB of RAM and a GeForce RTX 4070 Ti with 12GB of VRAM. Our source code is available at `https://github.com/matwerner/resume_parser/`.

5.2
Template Classification Task

To evaluate our approach, we employed EfficientNets [2], a family of Convolutional Networks designed to efficiently grow in size as more computational resources are made available. The great advantage of this architecture is that it achieves results that are competitive with those obtained by much larger networks (e.g., ResNets [47]) in image classification tasks. Specifically, we employed only the EfficientNet-B0, the smallest model of the family. Preliminary

results showed no significant gain in using larger EfficientNet versions. The same happened with our tests with ResNets.

The EfficientNet-B0 is tested under three different training configurations described in Section 5.2.3. Since we are dealing with image classification, we compared these configurations in terms of standard classification metrics such as Accuracy, Precision, Recall, and F1-Score. We also point out that we take into account the resumes themselves when partitioning the dataset. We do this to prevent the same PDF document from having pages in different partitions.

5.2.1 Comparative Method

We also evaluate a heuristic (as a baseline) inspired by the concepts introduced in Section 4.2.1 regarding the complexity level of page layouts and their correlation with the templates under consideration in this thesis.

The core assumption is that the set of cuts representing each template is performed in the largest empty spaces of the page. Thus, the heuristic's objective is to identify these spaces, obtain the set of cuts ζ characterizing the page, and then classify its template as:

$$Template(\zeta) = \begin{cases} \textit{1-Column} & \text{if } \zeta \text{ is empty;} \\ \textit{2-Column} & \text{if } \zeta \text{ contains a vertical cut only;} \\ \textit{Complex} & \text{otherwise.} \end{cases} \tag{5-1}$$

For that purpose, the method applies a top-to-bottom page segmentation strategy for extracting the cuts, inspired by other approaches available in the literature [3, 48]. At each step, it performs horizontal or vertical cuts in all rectangular empty spaces traversing the entire page. Then, for each region created, it repeats the same procedure recursively until it cannot find any gap larger than a threshold. When the recursion ends, the method discards all horizontal cuts not followed by vertical cuts. A horizontal cut alone is redundant since the two regions created can still be read in top-to-bottom and left-to-right order. Finally, the resulting set of cuts is used to classify the page template based on Equation 5-1.

When horizontal and vertical cuts are feasible, the heuristic prioritizes the former over the latter because our primary objective is distinguishing *1-Column* and *2-Column* templates. All gaps dividing two regions must have a width $\geq \theta_x$ and a height $\geq \theta_y$. Furthermore, we require all regions to have at least θ_t tokens to avoid introducing tiny regions due to a few strangely positioned tokens.

In more detail, considering a page as a list of tokens, the described procedure for performing vertical cuts is as follows:

1. Sort the tokens in the x-axis;
2. Enumerate all empty spaces between two consecutive sorted tokens with width $\geq \theta_x$. Their height is the same as the page;
3. Perform a cut in each empty space, dividing the page into regions. Each token is transferred to its assigned region;
4. For each region created, check whether it has at least θ_t tokens. If false, merge the region with the following one;
5. Append all valid cuts to ζ;

The procedure is analogous to performing horizontal cuts.

A crucial implementation detail we must address is that the thresholds θ_x, θ_y, and θ_t vary with each page. The method associates statistics extracted from the resume to each threshold and multiplies them by the global factors α_x, α_y, and α_t. In the case of θ_t, this is just the total number of tokens on the page. Thus, for example, considering $\alpha_t = 0.01$ and a resume containing $1,000$ tokens, each region in it must have at least $\theta_t = 10$ tokens. On the other hand, θ_x and θ_y are multiplied by the median x-axis gap between consecutive tokens in the same line and the median y-axis gap between consecutive lines. For that to work, the method sorts all tokens with respect to the y and x axes beforehand. A line is formed by all tokens with the same y-coordinate.

5.2.2 Dataset

This task requires a dataset consisting of resume page images and their annotated template types. To prepare the data for annotation, all resumes were split into individual pages and then converted into images in their default display resolutions (usually A4 paper size, 72 dpi). The resulting collection is then uploaded to Label Studio [19] and annotated following the guidelines presented further in this section.

An alternative dataset with anonymized images is also created. In this version, images only highlight parts of the resume containing characters and conceal any other graphic element present, as illustrated in Figure 5.1. Our motivation is to have a new dataset with the following properties: (i) It does not contain sensitive information, so that we can make it available along with our code and (ii) it retains the characteristics of the original dataset that are

relevant for the template identification task. As hinted by the results in [49], a template classifier should rely mainly on the positional distribution of the tokens on the page, not on its meaning.

(a) Original (b) Anonymized

Figure 5.1: The original and anonymized versions of the same resume page.

Annotation Guideline The annotation process is straightforward; basically, each page is labeled according to the categories described in Section 4.2.2. Nonetheless, we shall mention that we prioritize the top-to-bottom left-to-right reading pattern when more than one reading order is possible. And also, this work focuses solely on text, meaning that figures, photos, and other graphic elements are ignored for the purpose of labeling a page.

Statistics The annotated dataset comprises 1,953 resumes spanning 3,638 pages. After completing the annotation process, we categorized 2,034 pages as *1-Column*, 576 pages as *2-Column*, and 1,028 pages as *Complex*. Table 5.1 shows the resume distribution over the template types used.

Table 5.1: The number of resumes using different combinations of template types.

Template Types	Count
1-Column only	629
2-Column only	355
Complex only	524
1-Column + 2-Column	114
1-Column + Complex	297
2-Column + Complex	30
1-Column + 2-Column + Complex	4

We examined the presence of a Logical Structure embedded in the PDFs of annotated resumes, indicating that the software tool established a reading order. Among the $1,953$ selected resumes, $1,005$ contained this information, accounting for approximately 51.5% of the files. Perhaps not surprisingly, most of them were created by Microsoft Word. It is important to note that the presence of such a structure does not necessarily guarantee that the reading order is correct [50]. For detailed distributions of this and other metadata attributes based on the software tool used, please refer to Appendix C.1.

5.2.3 Experimental Setup

Our experimental setting follows Kornblith et al. [51], where it is presented an in-depth analysis of the impacts of employing ImageNet [21] pre-trained models on many image classification tasks. Similarly to them, we analyze the impact of using the EfficientNet-B0 model over our original and anonymized datasets in three different training settings: as fixed feature extractors, fine-tuned from ImageNet initialization, and trained from random initialization. We used the pre-trained EffiecientNet-B0 model available at the Pytorch website[1].

When using the model as a feature extractor, we converted all images into feature vectors and trained a logistic regression with regularization *L2* and penalty $\lambda \in [10^{-6}, 10^{5}]$. For the fine-tuning setting, we trained only the classifier layer for 30 epochs using the SGD optimizer with Nesterov moment $m = 0.9$ and learning rate $\alpha = 0.01$; next, we unfroze the remaining layers (except for the batch normalization layers) and trained for another 20 epochs with $\alpha \in [10^{-2}, 10^{-4}]$ and $\lambda \in \{10^{-4}, 10^{-5}, 0\}$. Finally, for the randomly initialized weights setting, we trained the model for 50 epochs with $\alpha \in [10^{-1}, 10^{-3}]$ and $\lambda \in \{10^{-3}, 10^{-4}, 0\}$. In all these settings, we used Adam optimizer [52] with a cosine decay and batch size of 64.

We obtained the optimal parameters through a grid search on top of the accuracy score on a validation set for all these training settings. All models were trained up to their fixed number of epochs or until the accuracy on a validation set decreased through five consecutive epochs.

In the case of the heuristic method, we also employed a grid search over the parameters σ_x, σ_y, and σ_t to obtain their optimal values. We tested $\sigma_x \in \{1.0, 2.0, 3.0, 4.0, 5.0\}$, $\sigma_y \in \{1.0, 2.0, 3.0, 4.0, 5.0\}$ and $\sigma_t \in \{0.01, 0.03, 0.05, 0.07, 0.1\}$.

[1] `https://pytorch.org/vision/stable/models.html`

Pre-processing Since we only employed EfficientNet-B0, all images were reduced to 224×224 pixels and normalized by subtracting the average pixel and dividing by the standard deviation of each channel. If we were training from scratch, we used the statistics from the training images. Otherwise, we used the same statistics applied in the pre-trained model.

We also applied data augmentation to increase the number of training instances by applying small shifts and flips on the vertical and horizontal axes, inverting the colors, and randomly resizing the original images.

5.2.4 Results

Table 5.2 shows the accuracies obtained by all models and configurations tested on top of the datasets with the original and anonymized resume pages.

Table 5.2: Accuracy (in %) for all models trained. The heuristic method (baseline) obtained an accuracy of 74.93% ± 0.82%.

Dataset	Feature-Extraction	Fine-Tuning	Scratch
Original	75.92 ± 2.02	84.87 ± 1.45	80.83 ± 2.18
Anonymized	77.34 ± 1.66	85.45 ± 1.56	**85.50 ± 1.47**

The heuristic method already yields an accuracy of 74.9%, well above the majority class predictor, which would have an accuracy of about 55.9%. When examining metrics related to each template type, we observed it obtained a reasonable F1-Score for both *1-Column* and *2-Column*, both around 80%. On the other hand, it only obtained an F1-Score of 48.1% for *Complex* due to low recall. This outcome can be attributed to the method's tendency to minimize the number of cuts, favoring the classification as *1-Column* or *2-Column*. In a real-world scenario, however, the opposite would be preferable. Recall that pages classified as anything other than *Complex* undergo a "reconstruction" phase, emphasizing the importance of high precision in those cases.

Moving to the EfficientNet-B0 models trained under different settings, we observe that the behavior of the results is in line with those seen in other studies [51, 53]. The feature extraction setting has the lowest accuracy among the tested settings, only 75.9%. Even so, this is 1.0% above that obtained by the heuristic, driven mainly by the improvement of *Complex*. Meanwhile, as expected, the fine-tuning setting yields the best model with an accuracy of 84.9%. However, perhaps surprisingly, given the size of the dataset, even a model trained from scratch achieves a reasonable accuracy of 80.8%. By

examining once again the metrics related to each template type, we noticed that these differences in accuracy are primarily derived from the ability of the models to correctly identify complex pages, as all of them seem capable of distinguishing *1-Column* and *2-Column* from each other.

Replacing the original dataset with the anonymized one improved around 1% the accuracies of the models trained under the feature extraction and fine-tuning settings and around 4% the model trained from scratch, matching its performance to the fine-tuning one. The accuracy should not change for the heuristic because both versions use the tokens' bounding boxes only.

These results support the initial hypothesis that simplifying the images maintains the relevant properties of the image for this task. On the other hand, the difference in the improvement rates indicates that this transformation facilitated detecting patterns already embedded in the pre-trained model without enabling the capture of more complex or less frequent patterns of template types with fewer examples.

5.2.5 Error Analysis

To better understand each model's strengths and weaknesses, we analyzed the errors made in the test set of one of the dataset splits by the heuristic method and the fine-tuned EfficientNet-B0 model. To this end, we looked at other classification metrics and visually inspected misclassified pages to identify possible common error patterns.

Tables 5.3 and 5.4 show the confusion matrix and the classification metrics for each template type for the heuristic method. As we can see, the largest number of errors is due to classifying *Complex* as *1-Column* and *2-Column*, which significantly impairs both types' precision.

Table 5.3: Confusion matrix per template type for the heuristic method.

		Predicted		
		1-Column	**2-Column**	**Complex**
True	**1-Column**	401	15	30
	2-Column	3	95	5
	Complex	107	26	61

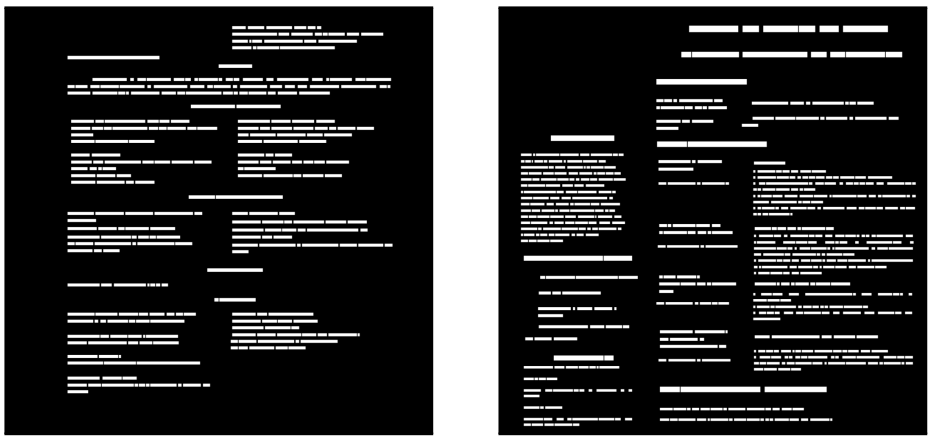
We found that most of those errors were caused by the heuristic not performing cuts between distinct text blocks separated by a narrow gap. As we suspected earlier, the method appears to cut the page only when there is a clear division between the pieces of information. The justification is that

Table 5.4: Classification metrics (in %) per template type for the heuristic method.

Template	Precision	Recall	F1-Score
1-Column	78.47	89.91	83.80
2-Column	69.85	92.23	79.49
Complex	63.54	31.44	42.06
Average	70.62	71.19	68.45

there are endless possibilities of how to cut a page if it does not take into consideration the gap size, especially in the horizontal direction. Nevertheless, allowing cuts in narrower gaps in a practical scenario may be more helpful since the method would classify as *1-Column* or *2-Column* only those pages in which it could find no empty spaces or only one vertical space. As already discussed, the precision of these templates must be high as they will undergo a "reconstruction" phase afterward.

Figures 5.2a and 5.2b illustrate the error discussed. The first is an example of a *Complex* page classified as *1-Column*. To be correctly classified, the method would require first to perform a horizontal cut between the section headings (white lines centered in the middle of the page) and their content. Then, perform a vertical cut separating the left and right text blocks. The issue is that the gap between the heading and its content is too narrow. The second is an example of a *Complex* page classified as *2-Column*. The method correctly performed the vertical cut dividing the text into two columns, and even performed horizontal cuts afterward — however, not in the gaps that would have enabled the method to perform even more vertical lines.



(a) 1-Column (b) 2-Column

Figure 5.2: The Figures 5.2a and 5.2b display *Complex* pages that the heuristic method classified as *1-Column* and *2-Column* respectively.

Tables 5.5 and 5.6 show the confusion matrix and the respective classification metrics for each template type for the fined-tuned EfficientNet-B0 model. The model does much better at identifying *Complex*, which improves *1-Column* and *2-Column* precisions.

Table 5.5: Confusion matrix per template type for the best model trained.

		Predicted		
		1-Column	**2-Column**	**Complex**
True	**1-Column**	405	2	39
	2-Column	0	98	5
	Complex	30	22	142

Table 5.6: Classification metrics (in %) per template type for the best model trained.

Template	Precision	Recall	F1-Score
1-Column	93.10	90.80	91.94
2-Column	80.32	95.14	87.11
Complex	76.34	73.19	74.73
Average	83.25	86.38	84.59

It is more challenging to interpret which image patterns misled the model because EfficientNet is a neural network model. However, we noticed that a reasonable number of the misclassified images were due to annotation errors during the dataset creation. The only consistent pattern we found through many images was the existence of two seemingly "aligned" text blocks – the top and bottom of the their bounding boxes start and end at approximately the same y-axis coordinates. This could suggest the model had difficulties understanding whether those blocks were distinct pieces of information or an aesthetic choice where part of the information is isolated (see Figure 5.3). Figures 5.4a and 5.4b illustrate the error discussed.

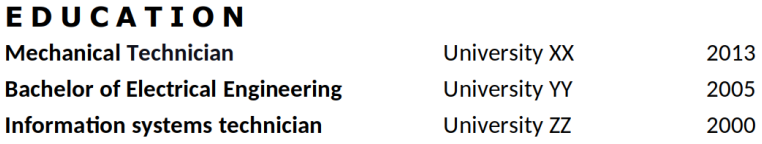
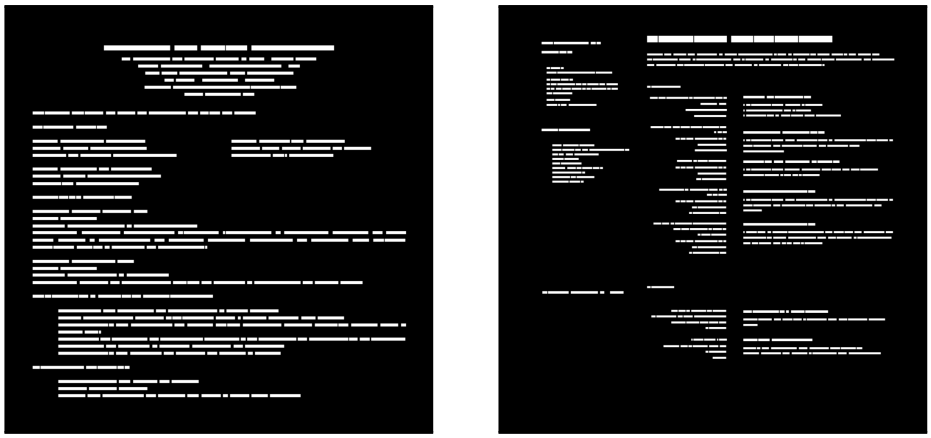


Figure 5.3: An *Education* section where each piece of information from an education is isolated from the others. Realistically, we read these educations line-by-line. However, assuming the reading should be column-by-column seems valid for the model. Even more so with the discretized images.



(a) 1-Column (b) 2-Column

Figure 5.4: The Figures 5.4a and 5.4b display *Complex* pages that the EfficientNet-B0 model classified as *1-Column* and *2-Column* respectively.

5.3 Resume Segmentation Task

To evaluate the First- and Second-level segmentation, we examined the quality of the segment sequences generated by CRF and BERT models under different training configurations with respect to both levels of information. When evaluating the first level, we concatenate the work experience and education items to retrieve their respective sections.

The true and predicted segment sequences are compared using the Micro F1-Score. Similar to Yu et al. [7] and Singh et al. [8], a predicted segment is only considered correct (True Positive) when it matches an annotated segment in terms of section type and the similarity between the predicted and annotated segments, measured by the Jaccard score, is at least 90% as illustrated in Figure 5.5. When testing alternative thresholds, the results observed were similar to those presented in section 5.3.3.

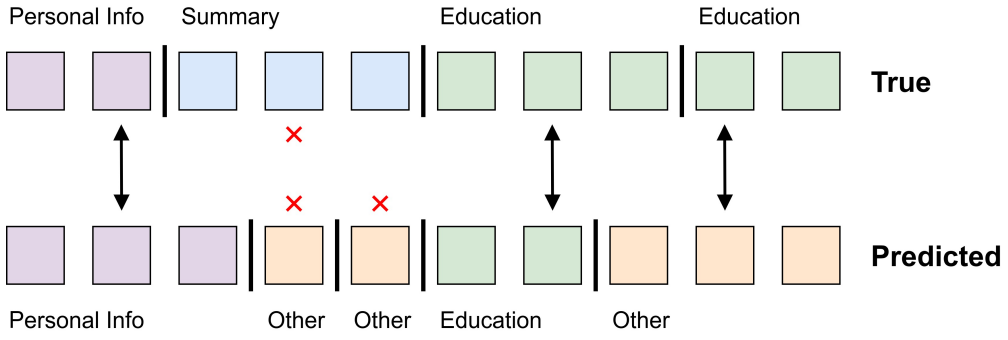


Figure 5.5: Matching (↕) between true and predicted segment sequences for classification evaluation with a Jaccard score threshold of at least 60%. In this simplified example, only two segments are considered correct.

We provide an additional experiment that specifically explores the impact of employing the item suffix strategy, as opposed to the conventional approach, in Appendix B.

5.3.1 Dataset

This task requires a dataset consisting of resumes with all their existing sections and some subsections annotated. To assess the quality of our segmentation component, we just add resumes with the correct reading order to this dataset. Recall that resumes with a wrong reading order should be handled by the first component of our proposed pipeline.

However, obtaining resumes with the correct reading order is challenging as it is not readily available. Therefore, as an approximation, we only use texts in which we have high confidence regarding the correctness of their reading orders. More precisely, we selected resumes in which each of its pages was labeled as *1-Column* or *2-Column* by the template identifier model and where the reading order remained the same as initially extracted after the reconstruction step. The resumes that met these criteria were uploaded to Brat [18] and annotated, following the guidelines presented further in this section.

We did not resort to any automated procedures to annotate resumes because it is unusual for them to contain an embedded structure within the PDF document, as commonly found in scientific papers in PubMed and arXiv repositories; There is also the fact that these repositories make the same file available both in PDF format and in standardized XML. The latter associate tags (e.g., Figure, Table, Text, Footnote) to all visual elements in the document. In this regard, studies [32, 54] building datasets on top of these repositories develop strategies to map the tags to the PDF layout.

Annotation Guideline We can break down the annotation process into two steps: segment identification and segment labeling. For illustration, Table 5.7 exemplifies an annotated resume.

Segment identification corresponds to identifying shifts in the resume section being referred to while reading the text, generating a list of segments. In the *IOB2 scheme*, the B prefix represents a section shift while the I prefix represents that we continue in the same section. For this step, we must annotate all resume section shifts, as our Resume Parser should be capable of identifying all sections described in a resume[2].

[2]Headers and Footers are not considered sections but part of the surrounding resume section, except when they contain personal information.

Segment labeling corresponds to mapping each found segment into one of the resume sections described in Section 4.3.1. For this step, we do not detail how it is decided which section a segment refers to, though the descriptions given in Section 4.3.1 should provide a good direction. We just note that when a segment can be placed in more than one section, we use a "majority" rule, which means we map it to the section most represented in that segment. For example, a segment describing formal education and information regarding language skills and certifications will be considered an Education section.

Table 5.7: An example of an annotated *2-Column* resume.

Line	Annotation
210 Stars Ave Berkeley, CA 78910	B-Personal Info
808.555.0118	I-Personal Info
dian@example.com	I-Personal Info
www.greatsiteaddress.com	I-Personal Info
Objective	B-Objective
Office Manager with 5 years of	I-Objective
experience in managing	I-Objective
administrative tasks, seeking a	I-Objective
challenging position to leverage	I-Objective
organizational, communication, and	I-Objective
leadership skills to streamline office	I-Objective
operations and support business	I-Objective
growth.	I-Objective
DIAN NUGRAHA	B-Personal Info
Experience	B-Work Experience
Dec 20XX–Jan 20XX	I-Work Experience
Office Manager • Northwind Traders	I-Work Experience
Feb 20XX–Dec 20XX	B-Work Experience
Administrative Assistant • Wide World Importers	I-Work Experience
Mar 20XX–Feb 20XX	B-Work Experience
Office Intern • Olson Harris, Ltd.	I-Work Experience
Developed and implemented office policies and procedures to	I-Work Experience
improve office efficiency and reduce costs.	I-Work Experience
Education	B-Education
Bellows College, Berkeley, CA	I-Education
• Bachelor of Science in Business Administration, 20XX	I-Education
Communication	B-Other
As an office manager, I have honed my communication skills	I-Other
through years of experience in verbal and written	I-Other
communication with clients, vendors, and team members. I	I-Other
have extensive experience in creating and delivering	I-Other
presentations, preparing and responding to business	I-Other
correspondence, and ensuring effective communication	I-Other
throughout the office.	I-Other
Leadership	B-Other
I have demonstrated strong leadership skills in managing a	I-Other
team of administrative staff and supervising daily office	I-Other
operations. I have experience in providing guidance and	I-Other
support to staff, setting performance expectations and	I-Other
providing feedback, and addressing issues as they arise.	I-Other
References	B-Other
Available upon request.	I-Other

Statistics To train and evaluate our segmenter, we annotated 1,081 resumes whose original reading orders matched the reading orders of our approach. Of the selected resumes, 940 exclusively include *1-Column* pages, 54 have only *2-*

Column pages, and 87 contain at least one page of each template type. We do not select resumes with *Complex* template, as we cannot verify their reading order. Table 5.8 shows some characteristics of the annotated sections.

Table 5.8: Basic statistics of the annotated sections.

Section Type	#Annotations	#Lines	#Words
Education	2,385	6,067	45,245
Objective	669	1,826	13,644
Other	2,386	15,223	124,246
Personal Info	1,210	6,888	56,766
Summary	528	5,170	56,601
Work Experience	4,772	36,171	370,264
All	11,950	71,345	666,766

We examined the presence of the Table of Contents embedded in the PDFs of annotated resumes, as it could potentially accelerate the segment identification step. Among the 1,081 selected resumes, only 75 contained this information, accounting for approximately 7% of the files. For detailed distributions of this and other metadata attributes based on the software tool used, please refer to Appendix C.2.

It is worth noting that each entry in this Table references the page and coordinates of a heading, but the specific region it encompasses is not defined. This means that an automated heuristic for segment extraction would only be feasible if the text content is correctly ordered.

5.3.2 Experimental Setup

For CRF, we inspected how the ontology, textual, visual, and spatial features detailed in Table 4.3 influence the overall results. To build each one of these models, we employed the L-BFGS algorithm [55] with l_1 and l_2 regularization, where $l_1, l_2 \in [10^{-3}, 10^2]$.

For BERT, similarly to the experiments done in Section 5.2, we analyzed the impact of using a pre-trained model as a feature extractor and fine-tuning it to our data following the experimental setups laid in [16, 56]. We used the pre-trained BERT Base model for Brazilian Portuguese named BERTimbau [56] available at the HuggingFace website[3].

When using the model as a feature extractor, we tested training the classifier layer for 10 epochs with a learning rate α set to 10^{-3} using as

[3] `https://huggingface.co/neuralmind/bert-base-portuguese-cased`

inputs only the last layer and the concatenation of the last four layers. For the fine-tuning setting, we trained only the classifier layer following the same setup used for the feature extractor approach; afterward, we unfroze the remaining layers and trained the model for another 10 epochs with learning rate $\alpha \in \{2 \cdot 10^{-5}, 1 \cdot 10^{-5}, 5 \cdot 10^{-6}\}$. In all these settings, we used a batch size of 8 and Adam optimizer [52] with a learning rate warm-up over the first 10% steps followed by linear decay of the learning rate over the remaining steps.

We obtained the optimal parameters in all these settings through a grid search on top of the Micro F1-Score on a validation set. All models were trained up to their fixed number of epochs or until the F1-Score on a validation set decreased through three consecutive epochs.

5.3.3 Results

Table 5.9 examines the performance of the CRF using different sets of features to understand how each one impacts the overall results.

Table 5.9: F1-Score (in %) for all CRF models trained with different sets of features. The "+" symbol indicates adding new features to those used in the column to its left. For example, "+ Visual" implies employing vocabulary, text, and visual features.

	Level	Ontology	+ Text	+ Visual	+ Spatial
Section	Education	87.48 ± 1.21	89.30 ± 1.56	89.83 ± 1.05	**90.53 ± 0.43**
	Objective	85.28 ± 1.42	**87.33 ± 1.26**	87.21 ± 1.93	86.79 ± 1.30
	Other	66.77 ± 2.46	70.97 ± 2.21	75.82 ± 2.41	**78.02 ± 1.90**
	Personal Info	83.41 ± 2.50	86.81 ± 1.86	88.26 ± 1.93	**88.86 ± 2.49**
	Summary	68.87 ± 1.81	70.96 ± 2.11	69.98 ± 1.72	**71.40 ± 0.90**
	Work Experience	82.58 ± 2.97	84.38 ± 1.69	84.69 ± 1.75	**85.13 ± 2.12**
	Average	79.07 ± 1.12	81.63 ± 0.79	82.63 ± 0.89	**83.46 ± 1.16**
Item	Education	47.22 ± 4.57	66.60 ± 1.83	69.59 ± 1.88	**71.69 ± 2.74**
	Work Experience	24.89 ± 2.92	55.79 ± 2.44	66.31 ± 3.12	**74.20 ± 2.91**
	Average	36.05 ± 1.67	61.19 ± 1.52	67.95 ± 1.91	**72.95 ± 2.64**

Regarding extracting first-level information, the use of vocabulary and text features already yields satisfactory results for most sections, with an average F1-Score of 81.2%. The *Personal Info*, *Objective*, *Education*, and *Work Experience* sections achieve the highest scores, all above 84.0%. These results can be explained by the large number of examples in the dataset corresponding to these sections, combined with the fact that each of them usually contains specific keywords with high discriminatory power. For example, the *Education* section should have a high concentration of words like "university", "school",

and "degree", which are unlikely to appear elsewhere in the text. On the other hand, the *Summary* and *Other* sections have lower scores, which can be linked to their more diverse content and intersection with the content of other sections. For the first level, the impact of introducing visual and spatial features is small for most of the sections. The exception is the *Other* section, which gets a 7.4% increase, presumably because these features help identify section boundaries between two consecutive *Other* sections.

However, the most significant impact of including the visual and spatial features can be observed in extracting second-level information. They improve the F1-Score of the education and work experience items by 5.9% and 18.0% compared to using only the textual and ontology attributes, respectively. The explanation is the same as the one suggested for the *Other* section's improvement previously. While these features do not directly provide noteworthy gains in identifying the section to which a line belongs, they play a crucial role in distinguishing section boundaries.

Table 5.10 illustrates the same evaluation for the BERT models. Here we examine the performance of employing BERT as a feature extractor and fine-tuning it to our data. In addition, we examine whether attaching the manually extracted features employed in the CRF to the classifier layer improves the overall performance over the default models.

Table 5.10: F1-Score (in %) for all BERT models trained with different sets of feature and training settings.

	Level	Feature Extraction		Fine-Tuning	
		Default	+ CRF Feat.	Default	+ CRF Feat.
Section	Education	75.65 ± 2.73	79.87 ± 1.15	**91.26 ± 0.85**	90.49 ± 1.30
	Objective	80.63 ± 3.08	83.59 ± 1.21	**88.09 ± 2.15**	87.75 ± 1.30
	Other	62.86 ± 2.21	69.83 ± 2.53	76.52 ± 2.17	**77.57 ± 1.50**
	Personal Info	87.36 ± 2.27	90.05 ± 1.41	**91.63 ± 1.80**	91.01 ± 0.69
	Summary	58.64 ± 3.63	62.05 ± 3.84	73.60 ± 3.08	**73.73 ± 3.86**
	Work Experience	64.98 ± 2.87	68.20 ± 3.24	**88.53 ± 1.67**	88.36 ± 2.28
	Average	71.69 ± 2.06	75.60 ± 1.44	**84.94 ± 1.52**	84.82 ± 0.75
Item	Education	65.26 ± 1.49	71.75 ± 1.78	**81.22 ± 2.61**	79.98 ± 3.08
	Work Experience	64.29 ± 2.51	73.07 ± 2.85	83.45 ± 1.86	**84.04 ± 2.15**
	Average	64.77 ± 1.52	72.41 ± 2.28	**82.33 ± 2.06**	82.01 ± 2.60

When utilizing BERT as a feature extractor, the results obtained for both the first and second levels were comparable, at best, to the CRF model with ontology and text features. However, considering the substantial difference in computational costs, these results are unsatisfactory. Nonetheless, we observed a notable improvement in the results by fine-tuning BERT with our specific

data. For first-level information, the average F1-Score was 1.5% higher than that of the best CRF model. The strong correlation between section extraction and the detection of section headings can explain this modest gain. Ontologies typically cover most headings since resumes tend to follow similar naming conventions, leaving little room for significant improvements at this level.

On the other hand, the second level exhibited a considerable performance gain. Both work experience and education items improved by approximately 9%. This indicates that BERT is better suited for identifying these items than the CRF model, potentially due to the greater variety of patterns employed to describe such items within resume sections.

Regarding the inclusion of the manually extracted features used in the CRF, we observed that the results remained stable when fine-tuning this extended version. This indifference in the results suggests that it is possible to detect most of the section boundaries by exploiting the resume layout's visual and spatial features or better understanding the textual content, rendering the CRF features unnecessary when using models with many learning parameters. However, incorporating these features improved the results when employed alongside BERT as a feature extractor. In conclusion, incorporating manually extracted features is beneficial only under constrained scenarios where the model's understanding of the resume content is limited.

5.3.4 Error Analysis

To better understand each model's strengths and weaknesses, we analyzed the errors made in the test set of one of the dataset splits by the best CRF and BERT models. To this end, we looked at their confusion matrices and visually inspected misclassified segments to identify possible error patterns. In these matrices, we added the tag $\varnothing$ to represent all segments that did not match any other segments with a Jaccard score of at least 90%.

Table 5.11 shows the confusion matrix of the CRF model. Evidently, most segments invalidated by our classification criteria fall under $\varnothing$. Thus, the most significant shortcoming of the model is incorrectly segmenting the text. This point can be clearly inferred when comparing the F1-Score of the *Education* and *Work Experiences* at the section and item levels displayed in Table 5.9. There is visible deterioration when moving from identifying sections to items, which indicates that the model is classifying lines in the right sections, but having difficulty identifying item breaks. The same logic can be applied to the case of the section *Other*.

Manually inspecting the segmentations obtained, we found the limited

Table 5.11: Confusion matrix for the best CRF model from one of the dataset splits. For clarity, ∅: Not Matched; **Edu**: Education; **Obj**: Objective; **Oth**: Other; **Per**: Personal Info; **Sum**: Summary; **Work**: Work Experience.

		Predicted						
		∅	**Edu**	**Obj**	**Oth**	**Per**	**Sum**	**Work**
True	∅	0	84	11	54	23	10	150
	Edu	132	322	0	5	0	1	1
	Obj	17	0	116	3	0	0	0
	Oth	72	3	0	345	0	9	10
	Per	27	0	0	2	207	0	0
	Sum	16	0	1	12	2	72	2
	Work	204	7	0	3	0	2	667

vocabulary to be the main responsible for the errors committed by the model. For example, most mistakes for the *Work Experience* were made on items starting with the company name. Correctly identifying company names is a complex task in itself. To give an idea, Brazil alone has around 20 million companies currently in operation [4]. Another number of mistakes seems to be caused by not recognizing unusual job titles such as "agricultor" ("Farmer") and "motorista categoria D" ("D-category driver"). This problem is analogous to *Education*. In this case, it is only necessary to replace the names of companies with universities and job titles with course names.

For the remaining sections, the problem is more related to the difference between the name of the section heading and its actual content, as illustrated in Table 5.12. Each heading is strongly associated with a section type, so it is difficult for the model to disentangle these correlations. The model also struggles to identify unusual headings, such as "Domínio de Softwares" ("Software Domain"), which is expected since it heavily depends on the quality of the ontology used.

Table 5.12: Example of a section misclassified by the CRF model. Commonly used headings such as "Skills" are strongly correlated to specific section types.

Line	Annotated	Predicted
Skills and Competencies	B-Objective	B-Other
I'm looking for a job to put into practice what I've learned ...	I-Objective	I-Other
companies. Such as Sales, Inventory, Cash, Repositor...	I-Objective	I-Other
assigned to me, always seeking knowledge to contribute ...	I-Objective	I-Other
...	...	..

[4] https://www.gov.br/economia/pt-br/assuntos/noticias/2022/julho/mais-de-1-3-milhao-de-empresas-sao-criadas-no-pais-em-quatro-meses

Table 5.13 shows the confusion matrix of the BERT model. The initial observations are the same as those already made during the CRF error analysis. Even the pre-trained model had difficulties finding the correct segmentation. Comparing the two confusion matrices, we observe that the most meaningful difference between them is in the number of correct education and work experience segments, which was already expected given the results displayed in Table 5.10. It is also noticeable that BERT considerably reduced the number of errors related to the names of companies and educational institutions.

Table 5.13: Confusion matrix for the best BERT model from one of the dataset splits. For clarity, ∅: Not Matched; **Edu**: Education; **Obj**: Objective; **Oth**: Other; **Per**: Personal Info; **Sum**: Summary; **Work**: Work Experience.

		Predicted						
		∅	**Edu**	**Obj**	**Oth**	**Per**	**Sum**	**Work**
True	∅	0	55	13	83	13	17	142
	Edu	72	382	0	6	0	1	0
	Obj	13	0	121	1	0	1	0
	Oth	68	1	3	351	0	13	3
	Per	13	0	0	1	222	0	0
	Sum	14	0	0	10	1	79	1
	Work	117	0	0	5	0	2	759

For *Work Experiences*, finding simple and frequent error patterns was difficult. The most apparent was single work experiences summarizing the entire professional trajectory within the company or the different projects to which the person was allocated, as illustrated in Table 5.14. We also noticed that this section is the one that contains the most considerable amount of data annotation errors. The explanation is simple, work experience items tend to be extensive. The amount of text combined with the lack of visual indications signaling the beginning/end of an item makes it difficult to annotate it within the annotation tool correctly.

For *Education*, some of the main reasons for the segmentation errors were: (i) education degrees using terms that are frequent in other contexts. For example, "Técnico em Turismo" ("Technician in Tourism") and "Gestão de Saúde" ("Health Management"); (ii) a single education spanning many lines - an unusual pattern for an education item; and (iii) the presence of additional information on an education item (e.g., main courses taken during undergrad).

Table 5.14: Example of a segment misclassified by the BERT model. The presence of a job title in a row is a strong indication that this is the beginning of a new work experience.

Line	Annotated	Predicted
Google Inc.	B-Work Experience	B-Work Experience
Occupation: Lead Data Scientist - Feb/2019 - Current	I-Work Experience	I-Work Experience
Occupation: Data Scientist - Mar/2014 - Feb/2019	I-Work Experience	B-Work Experience
Occupation: Intern - Oct/2013 - Mar/2014	I-Work Experience	B-Work Experience
Managed a team of data scientists, machine learning...	I-Work Experience	I-Work Experience
...	...	..

6
Conclusion

In this final chapter, we summarize our contributions to the domain of resume parsing, including a short discussion of the inherent limitations associated with our proposed methodology, and conclude by outlining potential future research directions.

6.1
Contributions

This thesis proposes a new resume parser whose main goal is to extract all sections from a given resume and also all items from its *Education* and *Work Experience* sections. The proposed parser consists of a pipeline with two core components: the first focuses on fixing the reading order of the text extracted from the PDF document, while the second extracts sections and items from the reordered text.

The first component employs EfficientNet-B0 to identify the most common page templates based on a layout complexity measure introduced in Section 4.2.1 and reorder the resume text according to the identified template on each page. This approach yields good results, achieving an average accuracy of 85.5% and an average F1-Score of 90.5% for *1-Column*, 86.8% for *2-Columns*, and 74.5% for *Complex* templates. The model successfully distinguishes *1-Column* and *2-Column* templates but struggles to differentiate them from the Complex template due to visual similarities.

Still related to the first component, we found out that identifying page templates using either the original or anonymized versions (highlighting character-containing regions only) produces very similar results. This finding allows us to provide an anonymized dataset while upholding privacy concerns.

The second component employs CRF- and BERT-based models to extract all resume sections and individual items from the *Education* and *Work Experience* sections. To this end, we considered each item as a separate section and slightly adjusted our tag encoding to distinguish the first item from the subsequent ones. As expected, BERT yields the best results for both the first and second levels of segmentation with an average F1-Score of 84.9% and 82.4%, respectively, demonstrating that the model is very good at segmenting

the text into sections. However, CRF also yields satisfactory results with an average F1-Score of 83.5% and 72.9%, respectively, which may be surprising given that it has significantly fewer parameters than BERT. The most significant difference between them lies in getting individual work experiences and education items, where we observe an average gap of about 9% in the F1-Score.

Nonetheless, the CRF can be a feasible alternative to BERT, depending on hardware and budget limitations. Indeed, its GPU-free requirement and potential for parallelization across multiple CPU cores make it an attractive option for resource-constrained scenarios.

6.2 Limitations

For the first component, the most obvious limitation is that it can only correctly order the text from documents using *1-Column* and *2-Column* templates. Other less frequent but still relevant templates are not mapped. An example is the *2-Column with Header* displayed in Figures 4.3b and 4.5c.

Despite that, we recommend our approach for fixing the reading order of resumes with simple templates only. The reason is that each mapped template is associated with a tailored heuristic. Hence, complex templates may require equally complex heuristics, which may be too cumbersome to develop accurately.

Finally, it is essential to remember that our proposal is based on the examination of resumes in the Brazilian job market. Our methodology may require adaptations for other markets, as each country has its own conventions. An example is regarding the implementation of a sub-component for identifying Tables. We encountered very few resumes that use tables to organize information. However, other countries may use them more frequently, requiring identifying them before reordering the text since they typically have their unique reading orders.

For the second component, a shortcoming of our approach is that we do not process the header and footnote of each page separately. That implies that these elements are present in the middle of the sections extracted, which is inappropriate from a pragmatic point of view. In addition, they may harm the capability of the models to segment the sections correctly.

Before employing our CRF approach as a segmenter, seeking a more robust method to enrich the vocabularies associated with the defined section types is advised since we currently use a naive approach. As mentioned in Section 4.3.3, the CRF relies exclusively on these vocabularies to associate each segmented section with its correct type.

For both components, the methodology used to build the datasets allowed us to annotate a reasonable number of resumes. On the other hand, it has limited the conclusions we can draw around the impact of reordering the text. We managed to demonstrate the shortcomings of some document parsers in handling *1-Column* and *2-Column* templates. However, we do not have a quantitative metric directly comparing the parsers' reading orders to those of our approach. It is also pending to verify how the reading order impacts the subsequent tasks of extracting entities and semantic relationships.

6.3 Future Work

As mentioned, one natural future work is ensuring that more templates are correctly ordered. For this purpose, we can employ the measure of complexity proposed in Section 4.2.1 to prioritize them. The idea is to start with simple templates, which are the easiest to develop heuristics and should arguably cover most resumes yet to be mapped.

In parallel, developing a more sophisticated methodology to extract the reading order of more complex layouts would be essential. One possibility is to work with larger textual elements to avoid creating excessive rules to ensure correct reading order. To obtain these elements, we could employ the Document Layout Analysis task [32]. Another alternative is to employ a LayoutReader-based model directly since it can handle any layout [33]. However, this approach would require training the model from scratch because the available pre-trained model [1] was trained using DOCX documents in English. It would also require developing post-processing steps to circumvent limitations imposed by the architecture, similar to what happened with BERT in Section 4.3.3.

Regarding resume segmentation, an evident future work is appropriately identifying more section types, such as Skills and Complementary Courses. It is reasonable to assume that they should perform well since we already indirectly identify them through the generic "Other" section.

Improving the ontology's quality and size is an urgent priority for the CRF-based segmentation model. One possibility would be to use a more robust approach, such as TaxoCon [57], to build the vocabularies associated with the various sections. The sections can be seen as part of a topic hierarchy, and the method's primary goal is to generate new topics on that hierarchy and assign them semantically related terms. This approach also opens the possibility of further organizing our ontology— for example, by generating sub-topics *Job Title* and *Company Name* under the *Work Experience* topic.

[1] `https://github.com/microsoft/unilm/tree/master/layoutreader`

Moving on to the transformers-based segmentation model, we can replace the BERT model with a more state-of-the-art model like LayoutLMs [49], an extension of BERT that considers the Layout information associated with tokens. Our primary concern is improving the identification of the *Other* and *Summary* sections since they have the worst results in our experiments.

Another relevant research direction would be investigating whether the core ideas discussed in this thesis could be helpful when extracting information from PDF documents in other domains. Some promising candidates are Legal, Medical, and Financial documents. A concrete example would be invoices, which contain a structure similar to a resume. They are electronic documents issued by numerous companies containing sections describing information about the seller, buyer, and goods transferred between the parties.

Finally, distancing ourselves from the scope of our work, we consider three other interesting research directions.

The first is to unite the ideas proposed in this thesis with those proposed in Vukadin et al. [13]. The union of both approaches would produce a model capable of simultaneously extracting any resume's sections, subsections, and entities.

The second work is to develop a Job Posting Parser, focusing on extracting the job responsibilities and skills described in the job posting. Ideally, the parser would also be capable of determining whether a given skill is optional or required. If the latter, the number of years of experience demanded. Building a high-quality dataset for this task should be trivial since millions of job postings are available daily on job portals.

The third work is adapting the LayoutReader [33] to work with larger textual elements. The original work suggests that this model does well even when only taking into account the layout information of the tokens. Therefore, it is natural to presume that the same must be valid when considering the layout information of paragraphs, for example. The most significant impact of this change is that the model would be language-agnostic and capable of handling any arbitrary visual element embedded in the files, such as photos, graphics, and tables.

Bibliography

[1] TABIAN, I.; FU, H. ; SHARIF KHODAEI, Z.. **A convolutional neural network for impact detection and characterization of complex composite structures**. Sensors, 19(22):4933, 2019.

[2] TAN, M.; LE, Q.. **Efficientnet: Rethinking model scaling for convolutional neural networks**. In: INTERNATIONAL CONFERENCE ON MACHINE LEARNING, p. 6105–6114. PMLR, 2019.

[3] MEUNIER, J.-L.. **Optimized xy-cut for determining a page reading order**. In: EIGHTH INTERNATIONAL CONFERENCE ON DOCUMENT ANALYSIS AND RECOGNITION (ICDAR'05), p. 347–351. IEEE, 2005.

[4] GUO, S.; ALAMUDUN, F. ; HAMMOND, T.. **Résumatcher: A personalized résumé-job matching system**. Expert Systems with Applications, 60:169–182, 2016.

[5] LI, C.; FISHER, E.; THOMAS, R.; PITTARD, S.; HERTZBERG, V. ; CHOI, J. D.. **Competence-level prediction and resume & job description matching using context-aware transformer models**. In: PROCEEDINGS OF THE 2020 CONFERENCE ON EMPIRICAL METHODS IN NATURAL LANGUAGE PROCESSING (EMNLP), p. 8456–8466, 2020.

[6] DAVE, V. S.; ZHANG, B.; AL HASAN, M.; ALJADDA, K. ; KORAYEM, M.. **A combined representation learning approach for better job and skill recommendation**. In: PROCEEDINGS OF THE 27TH ACM INTERNATIONAL CONFERENCE ON INFORMATION AND KNOWLEDGE MANAGEMENT, p. 1997–2005, 2018.

[7] YU, K.; GUAN, G. ; ZHOU, M.. **Resume information extraction with cascaded hybrid model**. In: PROCEEDINGS OF THE 43RD ANNUAL MEETING OF THE ASSOCIATION FOR COMPUTATIONAL LINGUISTICS (ACL'05), p. 499–506, 2005.

[8] SINGH, A.; ROSE, C.; VISWESWARIAH, K.; CHENTHAMARAKSHAN, V. ; KAMBHATLA, N.. **Prospect: a system for screening candidates for recruitment**. In: PROCEEDINGS OF THE 19TH ACM INTERNATIONAL

CONFERENCE ON INFORMATION AND KNOWLEDGE MANAGEMENT, p. 659–668, 2010.

[9] AYISHATHAHIRA, C.; SREEJITH, C. ; RASEEK, C.. **Combination of neural networks and conditional random fields for efficient resume parsing**. In: 2018 INTERNATIONAL CET CONFERENCE ON CONTROL, COMMUNICATION, AND COMPUTING (IC4), p. 388–393. IEEE, 2018.

[10] BARDUCCI, A.; IANNACCONE, S.; LA GATTA, V.; MOSCATO, V.; SPERLÌ, G. ; ZAVOTA, S.. **An end-to-end framework for information extraction from italian resumes**. Expert Systems With Applications, 210:118487, 2022.

[11] ADOBE SYSTEMS. **Document management — Portable document format — Part 1: PDF 1.7**. URL: `https://opensource.adobe.com/dc-acrobat-sdk-docs/pdfstandards/PDF32000_2008.pdf`, 2023. Last accessed on 30/08/23.

[12] APACHE SOFTWARE FOUNDATION. **Apache PDFBox**, 2020.

[13] VUKADIN, D.; KURDIJA, A. S.; DELAČ, G. ; ŠILIĆ, M.. **Information extraction from free-form cv documents in multiple languages**. IEEE Access, 9:84559–84575, 2021.

[14] CHUANG, Z.; MING, W.; GUANG, L. C.; BO, X. ; ZHI-QING, L.. **Resume parser: Semi-structured chinese document analysis**. In: 2009 WRI WORLD CONGRESS ON COMPUTER SCIENCE AND INFORMATION ENGINEERING, volumen 5, p. 12–16. IEEE, 2009.

[15] LAFFERTY, J. D.; MCCALLUM, A. ; PEREIRA, F. C. N.. **Conditional random fields: Probabilistic models for segmenting and labeling sequence data**. In: PROCEEDINGS OF THE EIGHTEENTH INTERNATIONAL CONFERENCE ON MACHINE LEARNING, p. 282–289, 2001.

[16] DEVLIN, J.; CHANG, M.; LEE, K. ; TOUTANOVA, K.. **BERT: pre-training of deep bidirectional transformers for language understanding**. In: PROCEEDINGS OF THE 2019 CONFERENCE OF THE NORTH AMERICAN CHAPTER OF THE ASSOCIATION FOR COMPUTATIONAL LINGUISTICS: HUMAN LANGUAGE TECHNOLOGIES, NAACL-HLT 2019, MINNEAPOLIS, MN, USA, JUNE 2-7, 2019, VOLUME 1 (LONG AND SHORT PAPERS), p. 4171–4186. Association for Computational Linguistics, 2019.

[17] ADOBE ACROBAT. **About tags, accessibility, reading order, and reflow**. URL: `https://helpx.adobe.com/acrobat/using/accessibility-features-pdfs.html`, 2023. Last accessed on 13/02/22.

[18] STENETORP, P.; PYYSALO, S.; TOPIĆ, G.; OHTA, T.; ANANIADOU, S. ; TSUJII, J.. **brat: a web-based tool for NLP-assisted text annotation**. In: PROCEEDINGS OF THE DEMONSTRATIONS SESSION AT EACL 2012, p. 102–107, Avignon, France, April 2012. Association for Computational Linguistics.

[19] TKACHENKO, M.; MALYUK, M.; HOLMANYUK, A. ; LIUBIMOV, N.. **Label Studio: Data labeling software**, 2020-2022. Open source software available from https://github.com/heartexlabs/label-studio.

[20] HARLEY, A. W.; UFKES, A. ; DERPANIS, K. G.. **Evaluation of deep convolutional nets for document image classification and retrieval**. In: 2015 13TH INTERNATIONAL CONFERENCE ON DOCUMENT ANALYSIS AND RECOGNITION (ICDAR), p. 991–995. IEEE, 2015.

[21] DENG, J.; DONG, W.; SOCHER, R.; LI, L.-J.; LI, K. ; FEI-FEI, L.. **Imagenet: A large-scale hierarchical image database**. In: 2009 IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION, p. 248–255. IEEE, 2009.

[22] RAMSHAW, L.; MARCUS, M.. **Text chunking using transformation-based learning**. In: THIRD WORKSHOP ON VERY LARGE CORPORA, p. 82–94, 1995.

[23] VASWANI, A.; SHAZEER, N.; PARMAR, N.; USZKOREIT, J.; JONES, L.; GOMEZ, A. N.; KAISER, Ł. ; POLOSUKHIN, I.. **Attention is all you need**. Advances in neural information processing systems, 30, 2017.

[24] MIKOLOV, T.; SUTSKEVER, I.; CHEN, K.; CORRADO, G. S. ; DEAN, J.. **Distributed representations of words and phrases and their compositionality**. Advances in neural information processing systems, 26, 2013.

[25] WU, Y.; SCHUSTER, M.; CHEN, Z.; LE, Q. V.; NOROUZI, M.; MACHEREY, W.; KRIKUN, M.; CAO, Y.; GAO, Q.; MACHEREY, K. ; OTHERS. **Google's neural machine translation system: Bridging the gap between human and machine translation**. arXiv preprint arXiv:1609.08144, 2016.

[26] BAHDANAU, D.; CHO, K. ; BENGIO, Y.. **Neural machine translation by jointly learning to align and translate**. In: 3RD INTERNATIONAL CONFERENCE ON LEARNING REPRESENTATIONS, 2015.

[27] CHEN, J.; GAO, L. ; TANG, Z.. **Information extraction from resume documents in pdf format**. Electronic Imaging, 2016(17):1–8, 2016.

[28] PAWAR, S.; THOSAR, D.; RAMRAKHIYANI, N.; PALSHIKAR, G. K.; SINHA, A. ; SRIVASTAVA, R.. **Extraction of complex semantic relations from resumes**. In: IJCAI-21 WORKSHOP ON APPLIED SEMANTICS EXTRACTION AND ANALYTICS (ASEA), 2021.

[29] GAUR, B.; SALUJA, G. S.; SIVAKUMAR, H. B. ; SINGH, S.. **Semi-supervised deep learning based named entity recognition model to parse education section of resumes**. Neural Computing and Applications, 33(11):5705–5718, 2021.

[30] AFFINDA. **What artificial intelligence technologies does Affinda use?** URL: `https://affinda.com/resume-parser`, 2023. Last accessed on 09/02/22.

[31] NANOTECS. **How to OCR Resumes using Intelligent Automation**. URL: `https://nanonets.com/blog/ocr-for-resume-parsing-deep-learning/`, 2023. Last accessed on 09/02/22.

[32] ZHONG, X.; TANG, J. ; YEPES, A. J.. **Publaynet: largest dataset ever for document layout analysis**. In: 2019 INTERNATIONAL CONFERENCE ON DOCUMENT ANALYSIS AND RECOGNITION (ICDAR), p. 1015–1022. IEEE, 2019.

[33] WANG, Z.; XU, Y.; CUI, L.; SHANG, J. ; WEI, F.. **Layoutreader: Pre-training of text and layout for reading order detection**. In: PROCEEDINGS OF THE 2021 CONFERENCE ON EMPIRICAL METHODS IN NATURAL LANGUAGE PROCESSING, p. 4735–4744, 2021.

[34] SUTSKEVER, I.; VINYALS, O. ; LE, Q. V.. **Sequence to sequence learning with neural networks**. Advances in neural information processing systems, 27, 2014.

[35] GU, Z.; MENG, C.; WANG, K.; LAN, J.; WANG, W.; GU, M. ; ZHANG, L.. **Xylayoutlm: Towards layout-aware multimodal networks for visually-rich document understanding**. In: PROCEEDINGS OF

THE IEEE/CVF CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION, p. 4583–4592, 2022.

[36] ARNOLD, S.; SCHNEIDER, R.; CUDRÉ-MAUROUX, P.; GERS, F. A. ; LÖSER, A.. **Sector: A neural model for coherent topic segmentation and classification**. Transactions of the Association for Computational Linguistics, 7:169–184, 2019.

[37] BARROW, J.; JAIN, R.; MORARIU, V.; MANJUNATHA, V.; OARD, D. W. ; RESNIK, P.. **A joint model for document segmentation and segment labeling**. In: PROCEEDINGS OF THE 58TH ANNUAL MEETING OF THE ASSOCIATION FOR COMPUTATIONAL LINGUISTICS, p. 313–322, 2020.

[38] LO, K.; JIN, Y.; TAN, W.; LIU, M.; DU, L. ; BUNTINE, W.. **Transformer over pre-trained transformer for neural text segmentation with enhanced topic coherence**. In: FINDINGS OF THE ASSOCIATION FOR COMPUTATIONAL LINGUISTICS: EMNLP 2021, p. 3334–3340, 2021.

[39] BAI, H.; WANG, P.; ZHANG, R. ; SU, Z.. **Segformer: A topic segmentation model with controllable range of attention**. In: PROCEEDINGS OF THE AAAI CONFERENCE ON ARTIFICIAL INTELLIGENCE, volumen 37, p. 12545–12552, 2023.

[40] BEEFERMAN, D.; BERGER, A. ; LAFFERTY, J.. **Statistical models for text segmentation**. Machine learning, 34:177–210, 1999.

[41] APACHE SOFTWARE FOUNDATION. **PDFBox FAQ**. URL: `https://pdfbox.apache.org/2.0/faq.html`, 2022. Last accessed on 08/08/22.

[42] SHINYAMA, Y.; GUGLIELMETTI, P. ; MARSMAN, P.. **PDFMiner Text Converter**. URL: `https://pdfminersix.readthedocs.io/en/latest/topic/converting_pdf_to_text.html`, 2022. Last accessed on 08/08/22.

[43] SHINYAMA, Y.; GUGLIELMETTI, P. ; MARSMAN, P.. **PDFMiner**, 2022.

[44] RATINOV, L.; ROTH, D.. **Design challenges and misconceptions in named entity recognition**. In: PROCEEDINGS OF THE THIRTEENTH CONFERENCE ON COMPUTATIONAL NATURAL LANGUAGE LEARNING (CONLL-2009), p. 147–155, 2009.

[45] LIU, H.; SETIONO, R.. **Chi2: Feature selection and discretization of numeric attributes**. In: PROCEEDINGS OF 7TH IEEE INTERNATIONAL

CONFERENCE ON TOOLS WITH ARTIFICIAL INTELLIGENCE, p. 388–391. IEEE, 1995.

[46] PASZKE, A.; GROSS, S.; MASSA, F.; LERER, A.; BRADBURY, J.; CHANAN, G.; KILLEEN, T.; LIN, Z.; GIMELSHEIN, N.; ANTIGA, L. ; OTHERS. **Pytorch: An imperative style, high-performance deep learning library**. Advances in neural information processing systems, 32, 2019.

[47] HE, K.; ZHANG, X.; REN, S. ; SUN, J.. **Deep residual learning for image recognition**. In: PROCEEDINGS OF THE IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION, p. 770–778, 2016.

[48] HU, J.; KASHI, R. ; WILFONG, G.. **Document image layout comparison and classification**. In: PROCEEDINGS OF THE FIFTH INTERNATIONAL CONFERENCE ON DOCUMENT ANALYSIS AND RECOGNITION. ICDAR'99 (CAT. NO. PR00318), p. 285–288. IEEE, 1999.

[49] XU, Y.; LI, M.; CUI, L.; HUANG, S.; WEI, F. ; ZHOU, M.. **Layoutlm: Pre-training of text and layout for document image understanding**. In: PROCEEDINGS OF THE 26TH ACM SIGKDD INTERNATIONAL CONFERENCE ON KNOWLEDGE DISCOVERY & DATA MINING, p. 1192–1200, 2020.

[50] OVERLEAF. **Got a PDF but is it accessible?** URL: `https://www.overleaf.com/learn/latex/An_introduction_to_tagged_PDF_files%3A_internals_and_the_challenges_of_accessibility`, 2023. Last accessed on 17/08/23.

[51] KORNBLITH, S.; SHLENS, J. ; LE, Q. V.. **Do better imagenet models transfer better?** In: PROCEEDINGS OF THE IEEE/CVF CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION, p. 2661–2671, 2019.

[52] KINGMA, D. P.; BA, J.. **Adam: A method for stochastic optimization**. 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings, 2015.

[53] AFZAL, M. Z.; KÖLSCH, A.; AHMED, S. ; LIWICKI, M.. **Cutting the error by half: Investigation of very deep cnn and advanced training strategies for document image classification**. In: 2017 14TH IAPR INTERNATIONAL CONFERENCE ON DOCUMENT ANALYSIS AND RECOGNITION (ICDAR), volumen 1, p. 883–888. IEEE, 2017.

[54] LI, M.; XU, Y.; CUI, L.; HUANG, S.; WEI, F.; LI, Z. ; ZHOU, M.. **Docbank: A benchmark dataset for document layout analysis**. In: PROCEEDINGS OF THE 28TH INTERNATIONAL CONFERENCE ON COMPUTATIONAL LINGUISTICS, p. 949–960, 2020.

[55] LIU, D. C.; NOCEDAL, J.. **On the limited memory bfgs method for large scale optimization**. Mathematical programming, 45(1-3):503–528, 1989.

[56] SOUZA, F.; NOGUEIRA, R. ; LOTUFO, R.. **Bertimbau: pretrained bert models for brazilian portuguese**. In: INTELLIGENT SYSTEMS: 9TH BRAZILIAN CONFERENCE, BRACIS 2020, RIO GRANDE, BRAZIL, OCTOBER 20–23, 2020, PROCEEDINGS, PART I 9, p. 403–417. Springer, 2020.

[57] LEE, D.; SHEN, J.; KANG, S.; YOON, S.; HAN, J. ; YU, H.. **Taxocom: Topic taxonomy completion with hierarchical discovery of novel topic clusters**. In: PROCEEDINGS OF THE ACM WEB CONFERENCE 2022, p. 2819–2829, 2022.

A
Common Resume Headings

Table A.1 displays the Resume Headings vocabulary. We examined approximately 100 resumes and carefully selected the most prevalent headings for each section type. In addition to Brazilian Portuguese, we included headings in English to account for cases where some resumes incorporate both languages.

Table A.1: Common resume headings by section type.

Section Type	Headings
Education	"Academic Background", "Dados Academicos", "Educacao", "Educacao Formacao", "Education", "Educational Background", "Escolaridade", "Formacao", "Formacao Academica", "Formacao E Cursos", "Formacao Educacional", "Formacao Escolar", "Formacoes Academicas", "Informacao Academica"
Objective	"Goal", "Interesses", "Objective", "Objetivo", "Objetivo Profissional", "Professional Goal"
Other	"Additional Information", "Adicionais", "Aptidoes Competencias", "Area De Interesse", "Artigos", "Atividade Adicional", "Atividade Complementar", "Atividades Adicionais", "Atividades Adicionais", "Atividades Complementares", "Atividades Extra Curriculares", "Atividades Extracurriculares", "Atividades Extras", "Awards", "Capacitacoes Certificacoes", "Cargo de Interesse", "Cargo Pretendido", "Certificacao", "Certificacoes", "Certificado", "Certificados", "Certifications", "Certifications And Licenses", "Competences", "Competencias", "Competencias Adicionais", "Competencias Comportamentais", "Competencias Habilidades", "Competencias Pessoais", "Competencias Profissionais", "Competencias Tecnicas", "Computer Skills", "Conhecimentos", "Conhecimentos Adicionais", "Conhecimentos Complementares", "Conhecimentos Em Informatica", "Conhecimentos Especificos", "Conhecimentos Extracurriculares", "Conhecimentos Gerais", "Conhecimentos Habilidades", "Conhecimentos Idiomas", "Conhecimentos Profissionais", "Conhecimentos Tecnicos", "Conquistas", "Cursos", "Cursos Adicionais", "Cursos Aperfeicoamento Profissional", "Cursos Certificacoes", "Cursos Complementares", "Cursos De Extensao", "Cursos Extracurriculares", "Cursos Extras", "Cursos Extras Curriculares", "Cursos Formacao Complementar", "Cursos Seminarios", "Dados Complementares", "Deficiencias", "Diploma", "Distincoes", "Experiencia Complementar", "Experiencias Internacionais", "Experiencias Profissionais Complementares", "Extracurricular Activities", "Ferramentas", "Formacao Academica Complementar", "Formacao Complementar", "Habilidade", "Habilidade Competencias", "Habilidades", "Habilidades Adicionais", "Habilidades Competencias", "Habilidades Conhecimentos", "Habilidades Especificas", "Habilidades Ferramentas", "Habilidades Idioma", "Habilidades Informatica", "Habilidades Pessoais", "Habilidades Profissionais", "Habilidades Qualificacoes", "Habilidades Softwares", "Habilidades Tecnicas", "Honors", "Honras", "Idiomas", "Idiomas Estrangeiros", "Idiomas Habilidades", "Idiomas Informatica", "Informacao Extra", "Informacoes Adicionais", "Informacoes Complementares", "Informacoes Extras", "Informatica", "Informatica Conhecimentos Habilidades", "International Trips", "Key Skills", "Languages", "Lingua Estrangeira", "Linguagens Programacao", "Linguas", "Linguas Estrangeiras", "Lista Competencias", "Outras Atividades", "Outras Certificacoes", "Outras Competencias", "Outras Ferramentas", "Outras Habilidades", "Outras Informacoes", "Outras Informacoes Relevantes", "Outras Linguas", "Outras Qualificacoes", "Outros Conhecimentos", "Outros Cursos", "Outros Idiomas", "Outros Objetivos", "Palestras", "Participacao Eventos", "Patentes", "Premios", "Premios Reconhecimentos", "Pretensao Salarial", "Principais Competencias", "Principais Habilidades", "Principais Habilidades Competencias", "Producao Bibliografica", "Publicacoes", "Publications", "Qualificacao Atividades Complementares", "Qualificacoes Adicionais", "Qualificacoes Atividades Complementares", "Qualificacoes Atividades Profissionais", "Qualificacoes Competencias", "Qualificacoes Complementares", "Qualificacoes Conhecimentos", "Qualificacoes Cursos Complementares", "Qualificacoes Extras", "Qualificacoes Habilidades", "Qualificacoes Informacoes Adicionais", "Reconhecimentos", "Referencias", "Referencias Pessoais", "Referencias Profissionais", "Resumo Competencias", "Resumo Habilidades", "Resumo Habilidades Competencias", "Scholarships", "Sintese Qualificacoes Competencias", "Skills", "Tecnologias", "Treinamentos", "Vivencia Internacional", "Volunteer Work"
Personal Info	"Dados Pessoais", "Informacoes Pessoais", "Personal Data", "Personal Details", "Personal Information", "Personal Profile"
Summary	"About", "About Me", "Caracteristicas Pessoais", "Perfil", "Perfil Pessoal", "Perfil Profissional", "Principais Qualificacoes", "Profile", "Qualificacao", "Qualificacao Profissional", "Qualificacoes Pessoais", "Qualifications", "Qualifications Summary", "Resumo", "Resumo Curriculo Profissional", "Resumo Profissional", "Resumo Qualificacoes", "Sintese", "Sintese Qualificacoes", "Sintese Profissional", "Sobre", "Sobre Mim", "Sumario", "Sumario Qualificacoes", "Summary", "Summary Of Qualifications"
Work Experience	"Atuacao Profissional", "Dados Profissionais", "Employment", "Employment History", "Experience", "Experiencia", "Experiencia Profissional", "Historico Profissional", "Informacao Profissional", "Positions Held", "Principais Experiencias Profissionais", "Professional Background", "Professional Experience", "Trajetoria Profissional", "Work Experience"

B
Regarding the Use of the Item Suffix Strategy

To complement the experiments conducted in 5.3, we devised an additional experiment comparing the model's performance when employing the item suffix strategy proposed in Section 4.3.2 and not. For that, we trained the CRF model with all manually selected attributes and fine-tuned the BERT model under these two settings. Table B.1 displays the results obtained.

Our strategy improved the metrics of both models by a modest amount. In most cases, the models handle the *Education* and *Work Experience* sections and items well, whether or not they employed the suffix. Its addition occasionally helped, given that it increased the F1-Score of these sections for both levels of information. Moreover, it did not harm any result related to other sections. Hence, it makes sense to use our proposed approach even though it is a marginal gain.

Table B.1: F1-Score (in %) for the CRF and BERT models with and without the item suffix strategy. **True**: The model used the item suffix. **False**: The model did not use the item suffix.

Level		CRF		BERT	
		False	True	False	True
Section	Education	88.74 ± 0.61	90.53 ± 0.43	90.75 ± 1.29	**91.26 ± 0.85**
	Objective	86.34 ± 2.10	86.79 ± 1.30	87.06 ± 2.73	**88.09 ± 2.15**
	Other	77.02 ± 1.21	**78.02 ± 1.90**	76.30 ± 1.10	76.52 ± 2.17
	Personal Info	89.24 ± 2.12	88.86 ± 2.49	89.64 ± 1.95	**91.63 ± 1.80**
	Summary	69.35 ± 2.09	71.40 ± 0.90	73.31 ± 4.55	**73.60 ± 3.08**
	Work Experience	82.24 ± 1.59	85.13 ± 2.12	87.58 ± 1.48	**88.53 ± 1.67**
	Average	82.16 ± 0.82	83.46 ± 1.16	84.11 ± 1.08	**84.94 ± 1.52**
Item	Education	70.53 ± 3.17	71.69 ± 2.74	80.52 ± 2.67	**81.22 ± 2.61**
	Work Experience	74.14 ± 2.43	74.20 ± 2.91	81.61 ± 2.73	**83.45 ± 1.86**
	Average	72.33 ± 2.71	72.95 ± 2.64	81.07 ± 2.39	**82.33 ± 2.06**

C
PDF Metadata Statistics

When we introduced the PDF format in Chapter 2.1, we commented on the format's support for optionally storing the Table of Contents and the Logical Structure of the rendered content. As filling in this metadata may facilitate the tasks worked on in this thesis, we counted their appearances in the created datasets to understand whether it would be worth exploiting them. We segment this information by the name of the writing software used. This distinction is relevant since storing (or not) this information should arguably be a software decision predominantly. It is reasonable to think that most users are unaware of these metadata and their impact on the automatic extraction of information.

We display the resultant distributions in Section C.1 for the Template Classifier dataset and Section C.2 for the Resume Segmentation dataset. According to the PDF documentation [11], the Table of Contents is located in the field *Outlines*, while the Logical Structure is located in the field *StructTreeRoot*. Furthermore, the name of the software used can be found in the field *Producer*.

C.1
Template Classifier Dataset

Table C.1 shows the presence of a Logical Structure and Table of Contents for the PDF files composing the Template Classifier dataset (see Section 5.2.2 for details). We segmented this information by the software tool used because, as is clarified in the table, each tool has a different policy regarding the existence of a logical structure and table of contents.

The software most frequently utilized comprises offerings from Microsoft and Google (SkiaPDF). The dominance of these companies was not surprising, given their provision of these tools free of charge, coupled with their standing as market leaders in the technology sector, serving hundreds of millions of users. Canva, another extensively employed software, owes its popularity to its no-cost nature. Unlike the aforementioned options, Canva permits more versatile document layouts, adding to its appeal.

Regarding the Logical Structure, it was only logical for Microsoft Word to

Table C.1: The number of PDF files composing the Template Classifier dataset encoding a Logical Structure and Table of Contents by the most used Software Tools. For clarity, we grouped all versions of the same software into one entry. The entry *<Empty>* means the *producer* property is empty in the file metadata.

Software Tool	Logical Structure		Table of Contents		total
	False	True	False	True	
Microsoft Word	33	485	518	0	518
Microsoft Word for Office 365	9	270	279	0	279
Canva	204	0	204	0	204
SkiaPDF	144	0	144	0	144
<Empty>	84	34	107	11	118
Microsoft Print To PDF	98	0	98	0	98
Apache FOP	1	82	83	0	83
Adobe PDF Library	44	19	58	5	63
Microsoft PowerPoint	0	44	44	0	44
Qt	44	0	28	16	44
...	287	71	299	59	358
total	948	1,005	1,862	91	1,953

incorporate this data, as the PDFs were generated from the standardized XML format, DOCX. An intriguing observation is that SkiaPDF does not retain this data despite its potential origin in a Microsoft Word-like application from Google. Meanwhile, Canva lacks this data due to its emphasis on providing flexible document layouts. Consequently, establishing an automatic logical structure among its elements becomes an intricate task.

Regarding the Table of Contents, only a minority of documents include such information. This scarcity can likely be justified by the requirement for users to define section headers (via header styles) to trigger the generation of this information. This demand for manual input seems somewhat unconventional, which might explain its limited prevalence.

C.2 Resume Segmentation Dataset

Table C.2 shows the presence of a Logical Structure and Table of Contents for the PDF files composing the Resume Segmentation dataset (see Section 5.3.1 for details). We segmented this information by the software tool used because, as is clarified in the table, each tool has a different policy regarding the existence of a logical structure and table of contents.

All the observations pertaining to the existence of the Table of Contents and the Logical Structure, as discussed for the Template Classification dataset

Table C.2: The number of PDF files composing the Resume Segmentation dataset encoding a Logical Structure and Table of Contents by the most used Software Tools. For clarity, we grouped all versions of the same software into one entry. The entry *<Empty>* means the *producer* property is empty in the file metadata.

Software Tool	Logical Structure		Table of Contents		total
	False	True	False	True	
Microsoft Word	14	296	310	0	310
Microsoft Word for Office 365	6	159	165	0	165
SkiaPDF	116	0	116	0	116
Apache FOP	11	94	105	0	105
<Empty>	42	57	85	14	99
Microsoft Print To PDF	56	0	56	0	56
Qt	55	0	10	45	55
LibreOffice	25	0	21	4	25
GPL Ghostscript	19	0	19	0	19
www.ilovepdf.com	0	17	13	4	17
...	90	24	106	8	114
total	434	647	1,006	75	1,081

in section C.1, hold true for the present dataset as well.

The sole additional matter that requires clarification pertains to the absence of the Canva software when comparing the data presented in Tables C.1 and C.2. The explanation behind this phenomenon is straightforward. This particular dataset exclusively consists of files categorized under the *1-Column* and *2-Columns* templates, both of which are characterized by their simplicity. Canva, on the other hand, is particularly renowned for enabling the creation of more intricate layouts, falling under our defined category of *Complex*. As a result, it is natural for Canva to be absent from this dataset, given its focus on designs that are beyond the scope of the current dataset's designated classifications.