UNIVERSITÀ DEGLI STUDI FIRENZE

Università degli Studi di Firenze
Dipartimento di Ingegneria dell'Informazione (DINFO)
Corso di Dottorato in Ingegneria dell'Informazione

Curriculum: Smart Computing

Connecting the *DOCS*: a graph-based approach to document understanding

Candidate
Andrea Gemelli

Supervisor
Prof. Simone Marinai

PhD Coordinator
Prof. Stefano Berretti

ciclo XXXVI, 2020-2023

Università degli Studi di Firenze, Dipartimento di Ingegneria dell'Informazione (DINFO).

Thesis submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Information Engineering. Copyright © 2024 by Andrea Gemelli.

A Sara, la mia vera casa

Acknowledgments

Of all the pages in this little book, this was the most difficult to write. For two reasons, at least. The first is that, since it is not the first time writing the acknowledgments, the probability of sounding redundant is greater than zero. The second, however, is that words seem to be too tight clothes for me who, in three such important years, have become another person: more mature, more capable and above all happier. I began this incredible journey, I must admit, without knowing precisely which path I wanted to pursue, accompanied daily by the classic internal doubt "have I made the right choice?". Today, after three years, I would make the same choice a thousand times, but I would not have been able to find the answer within myself: because the answers lie outside of us, among the people who remind us of who we are if we give them the opportunity to enter into our fragility . This is why a thesis work, notoriously and usually solitary, cannot finish without an acknowledgments page: without the people and stories I was able to meet, I would most likely never have been able to complete my journey.

I'll start by thanking professor Simone Marinai, who has always supported my irrepressible desire not to sit still. I owe him every experience I made. I was not an overly "easy" student, often ready with the last word, but our discussions were always fruitful and produced excellent works. Thanks again, really.

From my first home, Florence, I would like to thank "mum and dad" Dasara and Daniele, a lighthouse to me, who managed to listen hours and hours of my complaints about everything. Their patience and empathy are like a warm blanket in winter. It must be said, however, that if it weren't for the snacks that I certainly scrounged a bit, I would be on par with all of Daniele's complaints that I (and Dasara, obviusly) also listened to. I would also like to thank all the colleagues and students of the AI lab and more

Francesco, Curzio, Luca, Amin, Lorenzo, Simone, Lisa, Davide and the old collegue Manzueto who brought a bit of color and laughs among into the gray smarta. Last, but not least, a special thanks to my bro Emanuele, or more simply ”Ema”: we could have been ”the next big thing” together, but you decided to leave me. Jokes aside, you managed to make me rediscover a confidence in myself that I thought was lost and, beyond the relationship between colleagues, you have become an irreplaceable friend.

There is a second home that will forever remain in my heart. It is located in Barcelona and, more precisely, in an unspecified place between Carrer de Balmes and the Computer Vision Center. Here I thank professors Joseph Lladós and Dimosthenis Karatzas who welcomed me into their research center for a year, finding me a place in their very crowded basement, as one of their students: thank you very much for this magnificent opportunity. Then there are all the incredible ”mofos” that I was able to meet and with whom I shared incredible moments. I thank Andrea, Laura, Moha, Sergi, Pietro, Carlos, Giuseppe, Hector, Rubèn, Ayan, Khan, Pau, Soumya, Aarya, Ehaa, Manisha, Dipam, George, and Kevin. It was amazing meeting you all guys, wishing the best for you and hoping to see you again somewhere around!. In particular I would like to spend a few words for my italian-spanish friends Enrico and Marco: you are such amazing people that I have loved spending my time with, in and out the lab. I would also thanks the three most brilliant and at the same time stupidest human being I have ever met in my life: Andres, Ali, and Sanket. It is incredible how much I love you from the bottom of my heart, we had such a great time with our talks that the moment we lived together are the best gift someone could ever wish for. In particular you Ali taught me what competition really is and literally saved my ass for the thesis with your inspiring example. Again, last but not least, my thank to Albin: you have always been there when I needed a friend. Let’s keep in touch, please! There is no price for all of this and I feel the luckiest person in the world. Thank you again, all of you, CVC people!

A special thanks goes of course to my sisters Numidia and Maria. It is impossible to describe the bound we could build the three of us, I miss you so much. Every time I used to come back home I always felt welcomed and safe. We shared our struggles and our successes, the sun on the terrace, the vermut, the music and the singing on the sofa and other countless adventures. You will always have a special place in my heart. I also want to mention the other incredible friends I made in this beautiful city. Thank you Sheyla for

our talks, Anderson for your underground teachings, Mario for our countless stupid jokes, Ali for your chillness, Eleni for the limitless days at the beach and Maria Chiara and Luigi for your craziness and creativity. Thank you chicos!

A third home, this one very special, it is the one of the Monkeys. I don't have a place in the world where I can pin you, because you move as much as I do. But you are a constant in my daily life, with you I have never felt alone and you have always brought happiness in my darkest days. I know that everywhere we will be we will forever bounded. I love you Ventu and Giunta, do not ever change!

I'm not a person who likes to sit still very much, and the people who have always been there know this well. You are my roots, the place to always return to, and for you the thanks may never really end. I thank my mum Tamara for her trust and support in my choices. I thank the arpenaz Matteo, Lorenzo and Gennaro who, even if I made you crazy, will always welcome me every time I come back. I thank Vale who, even if we unfortunately update each other rarely, every time it is as if we had never lost sight of each other even for a day. I thank the entire extended PlayOff family: Elena, Eleonora, Matilde, Beppe, Riccardo, Aleberto, Sore, fabbro, Margherita, Beatrice, Giro, Sere, Vane, Tofi, Puccio, I love you all. In particular for this last year, a special thank goes to the hospitality of "Gallera's Tana" and to all the Aperol crazy people, Galle, Buffo, Delba, Baso and the mythological Fatberto. You are peculiar, but I love you too.

Of the many homes and families that I have had, frequented and for which I am eternally grateful, there is one that has been the most important of all, the one that I feel is my true home. Always, wherever I am or go. I love you Sara: we have a special complicity, which I have cultivated with love and continue to cherish with sweetness. Thank you for always being there, both for my successes and for my most difficult periods.

Ringraziamenti

Fra tutte le pagine di questo libricciolo, questa è stata la più difficile da scrivere. Per due motivi, almeno. Il primo è che, non essendo la prima volta che scrivo dei ringraziamenti, la probabilità di suonare scontato è non nulla. Il secondo, invece, è che le parole mi sembrano vestiti stretti per me che, in tre anni così importanti, sono diventato un'altra persona: più matura, più capace e sopratutto più felice. Ho iniziato questo incredibile viaggio, devo ammetterlo, senza sapere precisamente quale strada avrei voluto percorrere, accompagnato giornalmente dal classico dubbio interiore "avrò fatto la scelta giusta?". Oggi, dopo tre anni, rifarei la stessa scelta mille volte, ma non avrei potuto trovare la risposta dentro di me: perché le risposte stanno fuori di noi, tra le persone che ci ricordano chi siamo se gli diamo la possibilità di entrare nelle nostre fragilità. È per questo che un lavoro di tesi, notoriamente e solitamente solitario, non può non avere dei ringraziamenti: senza le persone e le storie che ho potuto incontrare, molto probabilmente non sarei mai riuscito a completare il mio percorso.

Parto con il ringraziare il professore Simone Marinai, il quale ha sempre assecondato la mia irrefrenabile voglia di non stare fermo. Ogni esperienza, per me preziosissime, che ho potuto fare nella mia carriera di studente le devo a lui. Non sono stato uno studente troppo "facile", spesso pronto con l'ultima parola, ma le nostre discussioni sono state sempre fruttuose e hanno saputo produrre ottimi lavori. Grazie ancora, davvero.

Della mia casa fiorentina ci tengo a ringraziare "mamma e babbo" Dasara e Daniele, miei fari luce in quel di Smart Computing, riusciti a sopportare ore e ore di mie lamentele sul tutto. La loro pazienza ed empatia sono come una copertina calda d'inverno. C'è da dire, comunque, che se non fosse per le merendine che sicuramente ho un po' scroccato, sarei in pari per tutte le lamentele di Daniele che anche io (e Dasara, ovviamente) ci siamo ascoltati.

Vi voglio bene. Ci tengo poi a ringraziare tutti i colleghi e studenti del lab AI e non solo, Francesco, Curzio, Luca, Amin, Lorenzo, Simone, Lisa, Davide e il mio collega di lunga data Manzueto che hanno portato un po' di colore nella grigia smarta e qualche risata fra gli spifferi dei fantasmi (finestre, ndr). Ultimo, ma non per importanza, un ringraziamento speciale al mio bro Emanuele, o più semplicemente "Ema": saremmo potuti essere "the next big thing" insieme, ma tu hai deciso di lasciarmi nel più bello. A parte gli scherzi, sei riuscito a farmi riscoprire una fiducia in me stesso che pensavo ormai persa e, al di là del rapporto tra colleghi, sei diventato un amico insostituibile.

C'è una seconda casa che rimarrà per sempre nel mio cuore. Si trova a Barcellona e, più precisamente, in un luogo imprecisato tra Carrer de Balmes e il Computer Vision Center. Qua ringrazio i professori Joseph Lladós e Dimosthenis Karatzas che mi hanno accolto nel loro centro di ricerca per un anno, trovandomi un posto nell'affollatissimo basement, come un loro studente: grazie infinte per questa magnifica opportunità. Poi ci sono tutti gli incredibili "mofos" che ho potuto incontrare là dentro e con i quali ho condiviso giornate incredibili. Ringrazio Andrea, Laura, Moha, Sergi, Pietro, Carlos, Giuseppe, Hector, Rubèn, Ayan, Khan, Pau, Soumya, Aarya, Ehaa, Manisha, Dipam, George, e Kevin. È stato fantastico incontrarvi tutti ragazzi, vi auguro il meglio e spero di rivedervi da qualche parte! In particolare vorrei spendere qualche parola per i miei amici italo-spagnoli Enrico e Marco: siete persone fantastiche con cui ho amato passare il mio tempo, dentro e fuori dal laboratorio. Vorrei anche ringraziare i tre esseri umani più brillanti e allo stesso tempo più stupidi che abbia mai incontrato in vita mia: Andres, Ali e Sanket. È incredibile quanto vi ami dal profondo del mio cuore, ci siamo divertiti così tanto con le nostre chiacchierate che i momenti che abbiamo vissuto insieme sono stati un regalo meraviglioso. In particolare tu Ali mi hai insegnato veramente cosa significhi competizione, quella sana e vera, e mi hai letteralmente salvato per la tesi con il tuo esempio ispiratore. Ancora una volta, ultimo ma non meno importante, il mio ringraziamento va ad Albin: sei sempre stato lì quando avevo bisogno di un amico. Rimaniamo in contatto, per favore! Non c'è prezzo per tutto questo e mi sento la persona più fortunata del mondo. Grazie ancora a tutti voi, amici del CVC!

Un ringraziamento speciale va ovviamente alle mie sorelle Numidia e Maria. È impossibile descrivere il legame che abiamo costruito tra noi, mi mancate molto! Ogni volta che tornavo a casa mi sentivo sempre accolto e al sicuro. Abbiamo condiviso le nostre fatiche e i nostri successi, il sole sulla terrazza, il vermut, la musica e le canzoni sul divano, oltre a innumerevoli avventure. Avrete sempre un posto speciale nel mio cuore. Voglio anche ricordare gli altri incredibili amici che ho conosciuto in questa bellissima città. Grazie Sheyla per le nostre chiacchiere, Anderson per i tuoi insegnamenti underground, Mario per le nostre innumerevoli stupide battute, Ali per la tue vibes chill, Eleni per le giornate al mare e Maria Chiara e Luigi per la vostra pazzia e creatività.

Una terza casa, questa un po' particolare, è quella delle Scimmie. Non ho un posto dove collocarvi nel mondo, perché avete il brutto vizio di spostarvi quanto me. Ma siete una costante nelle mie giornate, con voi non mi sono mai sentito solo e avete sempre portato un'aria di spensieratezza nelle giornate più pesanti. So che ovunque saremo non ci divideremo mai. Vi voglio bene Ventu e Giunta, non cambiate mai!

Non sono una persona a cui piace stare molto fermo, e questo le persone che ci sono da sempre lo sanno bene. Siete le mie radici, il posto a cui sempre tornare, e per voi i ringraziamenti potrebbero non esaurirsi davvero mai. Ringrazio mamma Tamara per la fiducia e il sostegno da sempre nelle mie scelte. Ringrazio gli arpenaz Matteo, Lorenzo e Gennaro che, anche se faccio penare, mi accoglieranno sempre tutte le volte che tornerò. Ringrazio la Vale che, anche se ci sentiamo così poco, purtroppo, tutte le volte è come non ci fossimo mai persi di vista neanche per un giorno. Ringrazio tutta la famiglia allargata di PlayOff: Elena, Eleonora, Matilde, Beppe, Riccardo, Aleberto, Sore, fabbro, Margherita, Beatrice, Giro, Sere, Vane, Tofi, Puccio, vi voglio bene. In particolare per questo ultimo anno, un grazie speciale va all'accoglienza della tana del gallera e a tutti i bolliti degli Aperol, Galle, Buffo, Delba, Baso e il mitologico Fatberto. Siete particolari, ma voglio bene anche a voi.

Di tante case e famiglie che ho avuto, frequentato e di cui sono eternamente grato, ce n'è una che è stata la più importante di tutte, quella che sento la mia vera casa. Sempre, ovunque io sia o vada. Ti voglio bene Sara: abbiamo una complicità speciale, che ho coltivato con amore e continuo a costudire con dolcezza. Grazie di esserci stata sempre, sia per i miei successi che per i miei periodi più difficili.

Abstract

Graphs are a natural representation of the patterns we glimpse in the world as we perceive it. The *data* as we receive it from *nature* is not only a set of objects but also a group of informative interactions within them. Thanks to this remarkable expressiveness, graphs have achieved ubiquitous prominence beyond mathematics and have permeated various scientific domains, including Document Understanding. Documents have a precise and rather complex structure: the *objects* found within them can take on different meanings depending on their *positioning* and/or their *mutual relationships.* These reasons elected graphs as an adequate framework for leveraging structural information from documents, due to their inherent representational power to codify the object components (or semantic entities) and their pairwise relationships. The recent success of Geometric Deep Learning as well as Graph Neural Networks has enabled the development of state-of-the-art methods based on these architectures, which have made it possible to fill the gap between theoretical foundations and practical applications. Through a graph-based approach, our work tackled several Document Understanding tasks, trying to meet some of the limitations we found in this context, contributing with novel frameworks, data collections and augmentation techniques. The aim of this dissertation has been an attempt to *connect* our publications under one consistent narrative to support our hypotheses and, in particular, to *connect* graphs and documents under a common intersecting definition we referred as *graph document representation.* Starting from a general overview of *how documents met graph theory*, we delve into more specific details about the implementations of our research questions, both for structured objects, such as tables, and whole document pages.

***Keywords* -** Document Understanding, Geometric Deep Learning, Table Extraction, Document Layout Analysis, Key Information Extraction, Data Augmentation, Pattern Recognition

Contents

Contents **xv**

List of Figures **xix**

List of Tables **xxiii**

1 Introduction **1**
1.1 Outline, Research Questions and Contributions 5
1.2 Compendium of Publications 8

2 How Documents met Graph Theory **9**
2.1 Graph Theory in Pattern Recognition 10
2.2 Graph Neural Networks 13
2.3 Document structure interpreted as a graph 15

I Structured Document Objects **21**

3 A graph-based method for Table Extraction **23**
3.1 Introduction 23
3.2 Related Works 25
3.3 Problem Formulation 26
3.4 Method 27
3.4.1 Converting PDF pages to graphs 28
3.4.2 Representation embedding 29
3.4.3 Custom message passing 32
3.5 Experiments 33
3.5.1 Ablations 33

3.5.2 Results 33
3.6 Conclusions and Future Works 35

4 A dataset and augmentation approach for graph representation of tables 37
4.1 Introduction 38
4.2 Related Works 39
4.3 A dataset for CTE 42
4.3.1 Problem Formulation 42
4.3.2 Subset of PubLayNet and PubTables-1M 43
4.3.3 Annotation procedure 44
4.3.4 Dataset structure and format 44
4.3.5 Limitations 45
4.4 Data Augmentation on Graphs 46
4.4.1 Method 46
4.4.2 Preparing the data 48
4.4.3 Experiments 50
4.4.4 Limitations 52
4.5 Conclusions and Future Works 53

II Document Objects as Page Graph Nodes 55

5 A task agnostic document understanding framework based on GNNs 57
5.1 Introduction 57
5.2 Related works 58
5.3 Method 59
5.3.1 Documents graph structure 59
5.3.2 Node and edge features 60
5.3.3 Architecture 61
5.4 Results 63
5.4.1 Proposed model 63
5.4.2 FUNSD 65
5.4.3 RVL-CDIP Invoices 68
5.5 Conclusion and Future Works 70

6 Automatic generation of scientific papers 73
6.1 Introduction . . . 73
6.2 Related work . . . 76
6.3 Semi-automatic annotation . . . 78
6.3.1 Automatic annotation . . . 78
6.3.2 Annotation correction . . . 79
6.4 Document generation . . . 81
6.4.1 Layout Transformer training . . . 81
6.4.2 Layout generation . . . 82
6.4.3 Content generation . . . 83
6.5 Experiments . . . 84
6.5.1 Training and test data . . . 84
6.5.2 Object detector for DLA . . . 85
6.5.3 Results . . . 86
6.6 Conclusions and Future Works . . . 87

7 Conclusions and Future Works 91

A List of contributions 95

B List of repositories 97

Bibliography 99

List of Figures

1.1 Document Understanding challenges. (images taken from RVL-CDIP [60], XFUND [156] and DocLayNet [110]. 2

1.2 Document Understanding (DU) applications and Geometric Deep Learning (GDL) tasks. Bold text and bold lines highlight the ones we focused on in this thesis and how both applications and tasks relate to each other. 4

2.1 Geometric Deep Learning Blueprint (taken from [21]). 11

2.2 Architectures based on GDL principles (taken from [21]). . . 12

2.3 How graph-like document representations evolved. From left to right (images taken from original papers): hierarchical tree [101], Voronoi diagram [77], graph document layout [88], table graph [2] and one of ours graph representation proposal [49]. 16

2.4 A general schema to document graph representation extraction pipelines. 17

3.1 Overview of the proposed method. 27

3.2 Graph of a portion of page. Different types of nodes have different colors. (red: text, green: title, pink: table cell, orange: table header, light blue: caption, grey: other) 28

3.3 Node features: positions and representations have a fixed length; Spacy or SciBERT embeddings have a length of 300 and 760, respectively. 30

3.4 Example of table to illustrate representation embedding. . . . 32

3.5 Three examples of model inference per row at token-level (left), clustering (center) and post-processing (right). Different colors represent different types of nodes or blocks (red: text, green: title, pink: table cell, orange: table header, light blue: caption, grey: other, dark blue: list, yellow: table). . . . 36

4.1 Example page (content is intentionally concealed in 'Original') and corresponding CTE annotations. Objects represent the layout regions. Tokens contain the word tokens labeled according to the class in the top of the figure. Acronyms for table annotations are: THEAD (table headers), TSPAN (table sub-headers spanning along different columns), TGRID (table cells), TCOLS and TROWS (respectively columns and rows of the tables). 41
4.2 Recognition of columns. Group of 1s in the projected vector indicate different columns. 47
4.3 Recognition of rows. The blue bounding box is detected belonging to a new row since its x coordinate is lower than the previous green block. 48
4.4 Words and bounding boxes extracted from one PDF paper using PyMuPDF. Nodes are connected through a visibility directed (from green to red sides) graph. 49
4.5 Different types of tables with their classes in Tab2Know (taken from [79]). 51

5.1 Our proposed Doc2Graph framework. For visualisation purposes, the architecture shows the perspective of one node (the blue one in Doc2Graph). 61
5.2 Image taken from [141]: the document on the right is the revised version of the document on the left, where some answers (green) are mislabeled as question (blue), and some questions (blue) are mislabeled as headers (yellow) 65
5.3 Blue boxes are FUNSD entities ground truth, green boxes are the correct detected one (with IoU $> 0.25/0.50$), while red boxes are the false positive ones. 66
5.4 RVL-CDIP Invoices benchmark in [121]. There are 6 regions: supplier (pink), invoice_info (brown), receiver (green), table (orange), total (light blue), other (gray). 68

5.5 Entity Linking on FUNSD. Differently from other apporaces, we make use of directed edges improving explainability: green and red dots mean source and destination nodes, respectively. 71
5.6 Layout Analysis on RVLCDIP Invoices. Inference over two documents from RVL-CDIP Invoices, showing both: **1. Layout Analysis** (a,c) for six different classes - supplier (pink), invoice info (brown), receiver (green), table (orange), total (light blue), other (gray); and **2. Table Detection** (b,d) - in the images our extracted table (green) is shown in contrast with the label table (orange). 72

6.1 Different pipelines to create labelled data depend on the combination of three main factors: the sources from where to crawl the data (left), which data are available (center), and which annotation procedures to involve (right). Synthetic documents do not require to following any specific annotation procedure since labels are given within the generated data. . 74
6.2 The proposed pipeline: first, data are collected, annotated in a semi-automatic fashion, and finally manually corrected (red boxes); then they are used to train a LayoutTransformer generator model that in turn generates synthesized layouts (green boxes) that are filled with content (blue box). We obtain a dataset of documents belonging to the domain of the initial data. Labels in the diagram are as follows: 1: Formulas, titles, tables; 2: Abstract, authors, keywords; 3: Text; 4: Figures; LT: LayoutTransformer. 76
6.3 Examples of automatically annotated pages. From GROBID and PDFMiner we extract nine categories: title (blue), authors (light green), abstract (cyan), keywords (red), subtitles (purple), text (pink), images (yellow), formulas (black) and tables (green). Captions and references are added by hand. . 80
6.4 Examples of training examples (top) and generated pages before manual correction (bottom). Double column layout in the left, single column layout on the right. Region colors are the same as those used in Fig. 6.3 apart from the orange box in the last example that corresponds to References. 82
6.5 Example of post-processing layout corrections: overlapping bounding boxes in the left are correctly separated in the right. 83

6.6 Single column layouts: in the first row we show the training samples, while in the second one the generated document regions. Region colors are the same as the ones used in 6.3. . . 85
6.7 Examples of generated documents: double-column on the upper side, single column on the bottom one. 89

List of Tables

3.1 Comparison of tasks performed by different methods. (*future extension of our proposed method). 26

3.2 Comparison of original and merged datasets (Document Layout Analysis (DLA), Table Detection (TD), Table Structure Recognition (TSR), and Table Functional Analysis (TFA)). . 27

3.3 Evaluations are conducted with accuracy on all classes and F1 score on table cells and headers. Each model (A, B, ..) has been tested with different combination of features. 34

4.1 Comparison of CTE with related datasets: ♣ denotes the datasets used to generate the new annotations. A dataset is $S4G$ (suitable for graphs) if a graph can be constructed directly with no further preprocessing. DLA (Document Layout Analysis), TD (Table Detection), TSR (Table Structure Recognition), and TFA (Table Functional Analysis) show which tasks models can be trained for. 39

4.2 Annotation Format: each line contains different information in case of Objects, Tokens, or Links. 45

4.3 Results without data augmentation (*No Aug.*); Data Augmentation with Row and Column inversion (R/C); Data Augmentation with Row and Column inversion and random removal of nodes and edges (All). P, R and $F1$ correspond to Precision, Recall and F1 score. 52

4.4 Summary of F1 scores for different data augmentation approaches. *No Aug.* indicates no Data Augmentation technique was applied, *R/C* indicates row and column inversion technique and *All* indicates row and column inversion technique and random removal of nodes and arcs. 53

5.1 Ablation studies of Doc2Graph model. EP Inner dim and IP FC dim show edge predictor layer input dimension and the input projector fully connected layers output dimension, respectively. AUC-PR refers to the key-value edge class. The # Params refers to Doc2Graph trainable parameters solely. . 64
5.2 Entity detection results. YOLOv5 [71]-small performance on the entity detection task. 66
5.3 Results on FUNSD. The results have been shown for both semantic entity labeling and entity linking tasks with their corresponding metrics. 67
5.4 Layout analysis results on RVL-CDIP Invoices. Layout analysis accuracy scores depicted in terms of node classification task. 69
5.5 Table Detection in terms of F1 score. A table is considered correctly detected if its IoU is greater than 0.50. Threshold values refers to the scores an edges has to have in order to do not be cut: in our case is set to 0.50 by the softmax in use. . 70

6.1 Statistics of training and test data. 86
6.2 Summary of experimental results: comparison of AP measures for different ResNeXt models trained on different data. 86

Chapter 1

Introduction

Wise as you will have become, so full of experience,
you'll have understood by then what these Ithakas mean.

C. P. Cavafy

Automated document analysis has always captured the interest of researchers and companies. The former is because they are attracted to the multi-faced challenge introduced by this topic, involving the use of computer vision and natural language processing techniques. The latter, on the other hand, because massive production of documents, whether born digital or not, requires huge human supervision that is very expensive both in terms of time and money, as well as prone to errors. That is why, over the years, research interest regarding **Document Understanding** (DU) has advanced steadily, by proposing increasingly cutting-edge solutions. By using the term DU we would like to encompass in this dissertation the various terminologies, applications and solutions associated with the use of computer vision, machine learning and deep learning techniques for the automatic analysis of documents, changed through the years. The term Document Analysis and Recognition (DAR) [95] has been the most widely used, including the application of heuristics and/or statistical machine learning techniques to applications such as document segmentation, reading order detection, and Optical Character Recognition (OCR). Now that research has advanced especially for understanding visually-rich documents exploiting deep learning methods, the term Document AI [32] started to be adopted, mainly referring

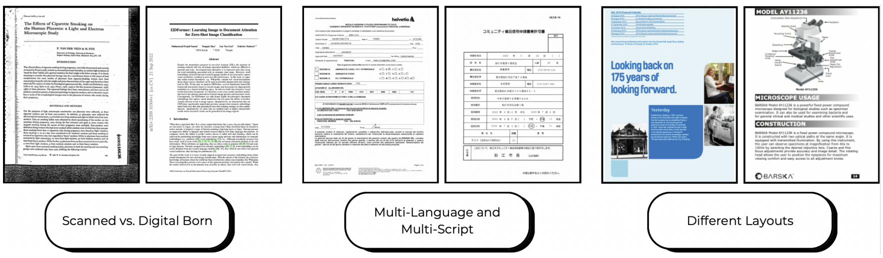


Figure 1.1: Document Understanding challenges. (images taken from RVL-CDIP [60], XFUND [156] and DocLayNet [110].

state-of-the-art advances in tasks such as Document Layout Analysis (DLA), Key Information Extraction (KIE), Document Visual Question Answering (DocVQA), Table Extraction (TE) and Document Classification (DC). Finally, as outlined by two recent benchmarks [19, 137], the term DU emerged as an umbrella term to comprehensively include all the different aspects of artificial and document intelligence.

DU introduces several challenges for different reasons. First of all, documents are not unique entities, but they are an arrangement of objects, such as paragraphs, images and tables, which are related to each other and can present themselves in potentially infinite ways. The high variability in documents can be found in the following aspects, as summarized in Figure 1.1: (i) *data quality*, e.g. scanned and digital-born documents require different preprocessing pipelines which may highly affect the whole DU workflow; (ii) *contents*, due to different languages and/or scripts; (iii) *different document layouts*, e.g. magazines, scientific papers and invoices arrange their content in total different ways. Moreover, benchmarks for DU often do not cover important real scenarios, due to privacy issues on business, medical or legal records or multi-page documents, making the development of state-of-the-art methods weak for scenarios where the distribution of data is different from the training one [47, 137] and in-the-wild applications. The majority of datasets available for the community, given the success of deep learning techniques and SSL [9], contain a very large number of samples automatically labelled (mainly composed by scientific papers), as opposed to small-scale manually annotated collections. Following what is outlined in [136], a solution to fill the gap between expensive annotation procedures and large automatically labelled collections, as well as coverage of as many document types as possible, could be the generation of synthetic data, that is still nowadays an open problem as important as it is challenging.

DU is challenging also because it can be tackled on different aspects, which results in many tasks that modern deep learning systems have to face and solve. DC is usually the first step among the majority of DU pipelines since the recognition of the type of documents can give important information for their subsequent analysis. RVL-CDIP [60] is the most important benchmark for this task, which allowed the development and comparison of state-of-the-art methods [81]. KIE includes understanding the relevant information of forms [69, 156], receipts [67] and invoices [53]: e.g. dates, addresses and names of companies and customers, total paid in transactions of a certain type, and so on. In addition, question answering [99, 130] has emerged as an extension of the KIE task principles, where a natural language question replaces a property name. In this case Transformers [135, 138] are holding the most competitive results. TE is also a very important task related to those already listed, yet of such importance that it can be considered separately. The literature has expressed itself with many nomenclatures to refer to tasks addressing tables [61], but recently [131] proposed a unified definition and a new benchmark for TE. One of the first and most important tasks regarding documents has been definitely DLA: it aims at automatically finding regions of interest, such as text or figures, and, if needed, recognizing and classifying them, e.g. discriminating two blocks of text into title or paragraph. Physically speaking, the objective is to identify homogeneous contents (regions) in terms of coordinates, for the majority of bounding boxes, through different page layouts, such as rectangular, Manhattan, non-Manhattan, Multi-column Manhattan, Arbitrary Complex, and overlapping (horizontally and diagonally) [14, 76]. Ranging from the early '90s up to nowadays, it is possible to broadly divide the different techniques into three groups: heuristics, statistical machine learning, and deep learning methods. Marinai largely describes the first two groups in [97], dividing the different approaches depending on two criteria. The first one refers to *how* the document is analyzed, either using bottom-up [77, 103, 142], top-down [101] or hybrid techniques, depending on if the analyses start from basic document constituent elements, such as connected components, or the whole document. The second categorization criterium discriminate the different approaches based on *what* will be analyzed, either the physical or logical document layout [132, 150]. On the contrary deep learning, also thanks to

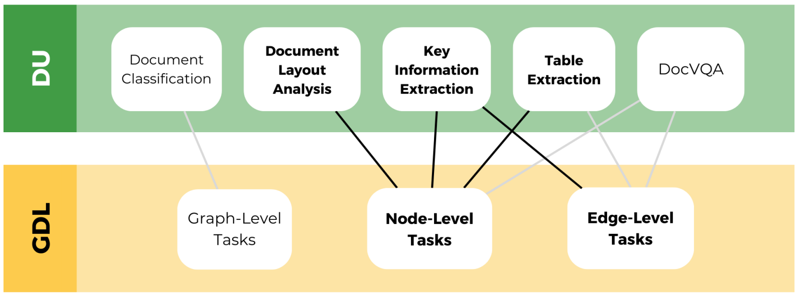


Figure 1.2: Document Understanding (DU) applications and Geometric Deep Learning (GDL) tasks. Bold text and bold lines highlight the ones we focused on in this thesis and how both applications and tasks relate to each other.

larger available document collections [85,164], started to be involved for DLA including Convolutional Neural Networks [63,120], multi-modal transformer-based architectures [66,154,155] and, after the recent success of **Geometric Deep Learning** (GDL) [21], Graph Neural Networks (GNNs).

Referring to the latter kind of architecture, with graphs being an extremely powerful and general representation of data, **DU can be backed by graph representations** as well. These structures robustly represent objects and relations and introduce three general types of prediction tasks at graph, node, and edge levels [126]. Graph reasoning for document parsing involves manipulating structured representations of semantically meaningful document objects (titles, tables, figures) and relations, using compositional rules. Customarily, graphs have been selected as an adequate framework for leveraging structural information from documents, due to their inherent representational power to codify the object components (or semantic entities) and their pairwise relationships. Moreover, having a graph offers accessibility to many granularities, at node, edge and graph levels, and the ability to solve numerous downstream tasks using a unique data representation. In this context, recently graph neural networks (GNNs) have emerged as a powerful tool to tackle the problems of KIE [27,158], DLA which includes well-studied subtasks like table detection [121,122,148], table structure recognition [92,157] and TE [49], Visual Question Answering VQA [86,89], synthetic document generation [17] and so on.

Figure 1.2 summarises and illustrates the many applications that can be handled in DU and the various tasks that are taken into account in GDL. The subjects covered in this thesis are represented in bold. As will be described in details in the following, the aim of this dissertation is two-fold: to *connect* a selection of our scholarly publications about DU under the same narrative strand and to *connect* graphs and documents under a common intersecting definition we refer as *graph document representation.*

1.1 Outline, Research Questions and Contributions

In this section we give an overview of the structure of this dissertation, along with research questions (RQs) that motivated our research and a summary of each chapter. In addition to the second one that introduces the pivotal topics of this thesis, we have divided our work into two main parts: **Part I. Structured Document Objects** - where we focus our interests on structured objects of the document, such as tables; and **II. Document Objects as Page Graph Nodes** - where we broaden the definition of *"document graph representation"* to the whole document, modelling the objects within it and the relationships between them as nodes and arcs, respectively, of its graph representation. Each part, in turn, is approached from two different perspectives, from both methodological and data aspects.

Chapter 2 - How Documents met Graph Theory

DU has been the leading topic of our research, but GNNs and graph theory played a central role. For this reason, we believe that an in-depth study dedicated to this topic would have enriched this thesis. In particular, we define a general schema shared across methods of DU to extract the *document graph representation.*

Chapter 3 - A graph-based method for Table Extraction

RQ 1: How can we build a graph on top of a PDF scientific paper, with a particular attention for tabular object?
RQ 2: Is it possible to tackle TE at once working on a graph representation?

We are aware that tables in scientific papers are essential to sum up novel discoveries and make research comparable. Being able to automatically extract them, minimising structural and textual misinterpretations, is crucial. Since the majority of publications are shared using PDFs, we propose a graphical representation directly on them, enriching node features vectors with suitable representation embeddings for numerical values. Moreover, differently from other methods, we are able to tackle in this way more tasks at once, including table detection, functional analysis and layout analysis.

Chapter 4 - A new dataset and a data augmentation approach about graph representation of tables

RQ 1: How can we build a new collection for Contextualized TE, that is an extended version of TE?
RQ 2: As for Natural Language Processing and Computer Vision, how can we define a data augmentation technique directly on a graph structure?

Tables are usually surrounded by other useful information and including them in the learning process would result in an enrichment of the TE task. That is why we collected a new annotated collection of data, subset of two popular datasets for TE (PubLayNet [164] and PubTables-1M [131]), to have multiple information altogether. The collection does not require any further preprocessing since the information are organized in a structured way to directly tackle Contextualized TE, an extended version of TE suited for graph-based methods. Moreover, we propose a simple yet effective data augmentation technique directly on graph structures to meet real case scenarios such as class unbalancing and scarcity of data. To do so, we created an automatic approach to generate different layouts of table through random removal of nodes and edges and inversion of rows and columns for the task of table type classification.

Chapter 5 - A task agnostic document understanding framework based on GNNs

RQ 1: How can we define a general framework to transform documents to graphs?
RQ 2: Is it possible to use such representation to tackle different DU tasks?
RQ 3: Can we reduce the problem dimensionality while retaining competitive performances?

Having already created a graph representation for a special case such as TE for scientific publications and PDF documents, we decided to extend this definition by creating Doc2Graph. This framework is able to extract a graph representation from *any* document, proposing suited nodes and edge features based on multiple modalities such as vision, language and layout. We tested Doc2Graph for four different tasks on two different benchmarks, obtaining competitive results and greatly reducing the number of parameters required to train the proposed algorithm.

Chapter 6 - Automatic generation of scientific papers

RQ 1: How can we deal with the high variability in content and layout when tackling any DU-related task?

The research community usually proposes new benchmarks and tasks to encourage the development of novel algorithms. These latter, after huge pretrainings, are then fine-tuned for downstream tasks and applied in-the-wild. However, being nearly impossible to represent at training time all the possible document representations, these systems still struggle with out-of-distribution data. An alternative solution is to propose generative methods, still an open problem and yet very promising. In this sense, we proposed a customizable semi-automatic pipeline to easily annotate small collections of documents and on top of that we use a generative layout transformer architecture to generate multiple document pages. We tested our solution for a DLA task, showing that a CNN-like architecture can improve final results with our generated data.

1.2 Compendium of Publications

This thesis is structured as a collection and re-elaboration of publications. Thus, each chapter is connected to one or two conference and journal articles:

- **Chapter 3: A graph-based method for Table Extraction**
  - **Andrea Gemelli**, E. Vivoli, S. Marinai: "Graph Neural Networks and Representation Embedding for Table Extraction in PDF Documents", in *International Conference on Pattern Recognition* (ICPR), Montréal (Canada), 2022
- **Chapter 4: A new dataset and a data augmentation approach about graph representation of tables**
  - **Andrea Gemelli**, E. Vivoli, S. Marinai: "CTE: A Dataset for Contextualised Table Extraction", in *Italian Research Conference on Digital Libraries* (IRCDL), Bari (Italy), 2023
  - D. Del Bimbo, **Andrea Gemelli**, S. Marinai: "Data Augmentation On Graphs for Table Type Classification", in *IAPR Joint International Workshops on Statistical Techniques in Pattern Recognition and Structural and Syntactic Pattern Recognition* (SSPR 2022), Montréal (Canada), 2022
- **Chapter 5: A task agnostic document understanding framework based on GNNs**
  - **Andrea Gemelli**, S. Biswas, E. Civitelli, S. Marinai. J. L. Canet: "Doc2Graph: a Task Agnostic Document Understanding Framework Based on Graph Neural Network", in *Text in Everything European Conference of Computer Vision* (ECCV Workshops), Tel Aviv (Israel), 2022
- **Chapter 6: Automatic generation of scientific papers**
  - L. Pisaneschi, **Andrea Gemelli**, S. Marinai. "Automatic Generation of Scientific Papers for Data Augmentation in Document Layout Analysis", *Pattern Recognition Letters*, vol. 167, March 2023, pp. 38-44. (Advances and New challenges in Document Analysis, processing and Recognition at the Dematerialization Age)
  - **Andrea Gemelli**, S. Marinai, L. Pisaneschi, F. Santoni. "Datasets and Annotations for Layout Analysis of Scientific Articles", *International Journal on Document Analysis and Recognition (IJDAR)*, 2024

Chapter 2

How Documents met Graph Theory

The image of the world around us, which we carry in our head, is just a model. Nobody in his head imagines all the world, government or country. He has only selected concepts, and relationships between them, and uses those to represent the real system

J. W. Forrester

Documents have a precise and rather complex structure: the *objects* found within them can take on different meanings depending on their *positioning* and/or their *mutual relationships*. Furthermore, since there are practically infinite types of layouts, considering the structure when processing the documents themselves is a crucial step for any algorithm trying to tackle DU related tasks. In this chapter we present *"how documents met graph theory"*, that is, how documents started to be represented as graphs and new solutions have been proposed to operate directly on them. To do so, we briefly introduce graph theory and how it has influenced deep learning, and then we report the latest approaches and state-of-the-art methods based on GNNs for DU.

2.1 Graph Theory in Pattern Recognition

The origins of graph theory coincide with Leonhard Euler's work on *geometria situs* ("geometry of location") [40] of 1736, and its problem of the *Seven Bridges of Königsberg*, historically known in mathematics and topology. The term *graph* appeared for the first time later on in 1878, in a Nature paper about *"chemistry and algebra"* [133]; but only in 1936 graph theory had its foundation with its very first textbook [13].

From a general definition, a graph is an ordered pair of *objects* and *relations*. Formally:

Definition 2.1.1. *A pair $G = (V, E)$ is called a* ***graph****, where:*

- *V is a finite set of singular elements called* ***vertices****, also known as nodes or points;*
- *E is a set of unordered pairs of vertices called* ***edges****, also known as links, i.e. $E = \{\{u, v\} \mid u, v \in V\}$*

Graphs are a natural representation of the patterns we glimpse in the world as we perceive it. The *data* as we receive it from *nature* is not only a set of objects but also a group of informative interactions within them. Thanks to this remarkable expressiveness, graphs have achieved ubiquitous prominence beyond mathematics and have permeated various scientific domains [126]. In the field of chemistry, for instance, a molecule can be depicted as a *chemical graph*, where vertices correspond to the constituent atoms of the compound and edges symbolize the chemical bonds connecting them. Similarly, in the domain of social networks, interactions such as "follow" and "like" among users can be construed as a *dynamic heterogeneous graph*, subject to evolution over time. When it comes to navigation tools like Google Maps, *weighted graphs* are used to determine the optimal path between a source and a target location.

In particular, in Structural Pattern Recognition, graphs have been extensively used as a powerful tool for representing and classifying visual patterns. The biological and geometrical structure of the brain, recognized to be very close to graph theory [23], influenced the experiments that were conducted for pattern recognition, like for the pioneering *Perceptron* [124]. Neurons in the visual cortex have a multi-layer structure with local spatial connectivity and cells, also called *receptive fields* [68], that respond only when stimulated:

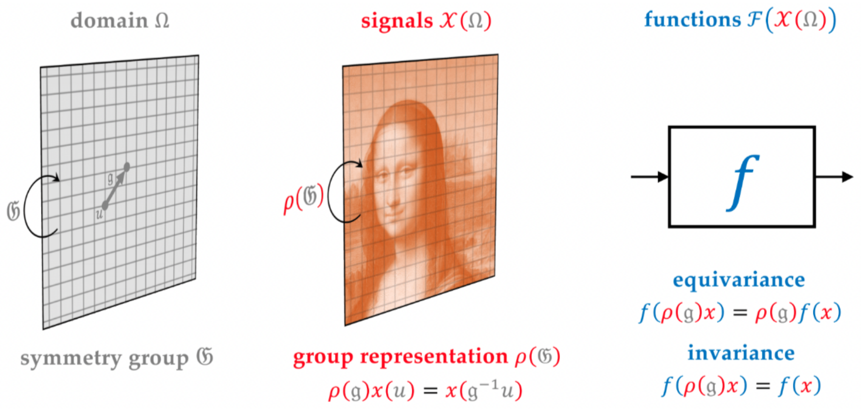

domain Ω
signals $\mathcal{X}(\Omega)$
functions $\mathcal{F}(\mathcal{X}(\Omega))$
$\mathfrak{G}$
$\mathfrak{g}$
u
$\rho(\mathfrak{G})$
f
equivariance
$f(\rho(\mathfrak{g})x) = \rho(\mathfrak{g})f(x)$
symmetry group $\mathfrak{G}$
group representation $\rho(\mathfrak{G})$
$\rho(\mathfrak{g})x(u) = x(\mathfrak{g}^{-1}u)$
invariance
$f(\rho(\mathfrak{g})x) = f(x)$

Figure 2.1: Geometric Deep Learning Blueprint (taken from [21]).

simpler cells are activated for simple and local inputs, then aggregated in more complex structures to recognise more complex patterns. These first insights impacted computer vision, creating one of the first attempt of geometric neural network called *Neocognitron* [44], an early version of modern Convolutional Neural Networks (CNNs) [82]: the major difference was that, at the time, it was trained without backprop [125]. That preliminary work was further extended by the famous 5-layer deep CNN called *LeNet-5* [83], tested on the famous MNIST dataset for the classification of digits. When the new challenging benchmark ImageNet [37] came out, an urgent need for more complex convolutional neural networks emerged to solve such a complex task. One of the first architectures developed and trained with a GPU was AlexNet [78], which officially started the "era" of deep learning. In the last ten years this branch of research has seen rapid growth and several new architectures have been proposed, such as Region Proposal Networks (RPN) [63, 120], Transformers [138], and Generative Adversarial Networks (GANs) [54], to name a few.

Given their common geometrical structures, the *zoo* of all these deep learning methods have been recently reconciled under the same terminology and mathematical priors with the term of Geometric Deep Learning (GDL) [21]. To formalise the *GDL Blueprint* (Figure 2.1) the authors are using group theory to model:

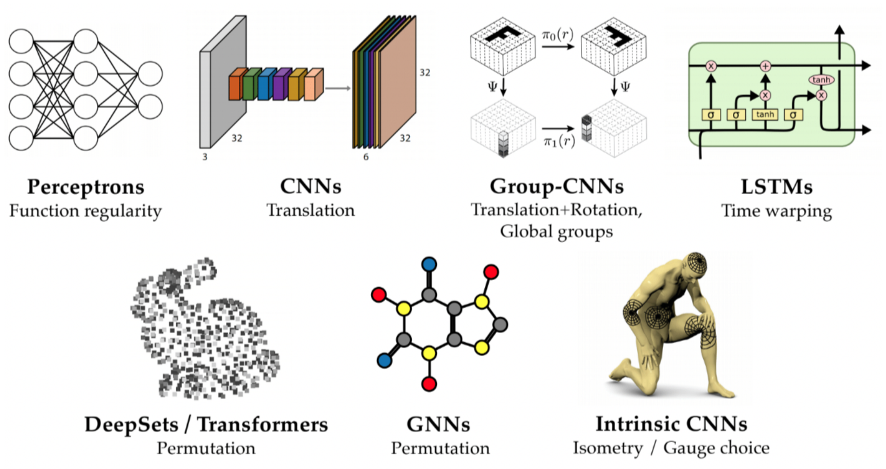


Figure 2.2: Architectures based on GDL principles (taken from [21]).

- the symmetry structure (group) of the domain Ω (e.g. translation group over $\mathbb{R}^2$ or permutation group over a graph)
- its representation ρ that will act on signals (e.g. images or node features)
- functions that incorporate properties of equivariance or invariance (e.g. convolutional or message passing layers)

Together these principles provide a very general design, recognisable in every popular deep neural network architecture, using a combination of equivariant (preserving the structure of the domain) and invariant layers that aggregate everything in a single output, e.g. image classification. These blueprints are applied to different domains or geometric structures called the *"5G"* of GDL: Grids, Groups, Graphs, Geometric Graphs and Gauges. The implementation of these principles gives rise to inductive biases that lead to some of the most popular architecture mentioned so far (Figure 2.2).

Among all the possible *functions* f that can be defined and/or *exploited* to work with different *signals* ρ, the emphasis of this thesis has been posed on graphs and Graph Neural Networks, which we introduce in the next section.

2.2 Graph Neural Networks

The two mathematicians A. Lehman and B. Weisfeiler, inspired by *chemical cyphers*, developed an algorithm also known as the WL-test [149], to determine when two graphs are *isomorphic*, i.e. structural equivalent and with the same connectivity despite of the order of nodes. Unfortunately, at the time the algorithm was threaded as unfeasible, leaving this problem unsolved and forgotten. A notable interest came back in the early 2000s with two works that introduced the term *Graph Neural Networks* for the first time [55, 128], as a generalisation of neural networks to graph structures briefly later improved and aligned to more recent deep learning techniques [87].

A Graph Neural Network (GNN) is an "optimizable transformation on all attributes of the graph (nodes, edges, global-context) that preserves graph symmetries (permutation invariances)" [126]. To pass graph-like data to the machine learning model, the definition 2.1.1 needs to be updated into a machine-readable format.

Definition 2.2.1. ***Machine-readable graph***. *Given $x_u \in \mathbb{R}^{\mathrm{m}}$ a feature vector associated with each node $u \in V$, a machine-readable graph is composed by:*

- *a **node feature matrix** $X \in \mathbb{R}^{|\mathrm{V}| \times \mathrm{m}}$, representing the set of nodes V in the m-dimensional space of study;*
- *an **adjacency matrix** $A \in \mathbb{R}^{|\mathrm{V}| \times |\mathrm{V}|}$, representing the set of edges E (with $a_{uv} = 1$ if $\{u, v\} \in E$, $a_{uv} = 0$ otherwise).*

Given this representation, a GNN is a function ϕ that maps each feature node x_u to an hidden and learnable representation h_u given its neighbours nodes' features vectors X_{N_u} (being $N_u = \{v \in V \mid \{u, v\} \in E\}$):

$$h_u = \phi(x_u, X_{N_u})$$

Formally, this approach has been generally named *message passing* [51], which is quite interestingly nothing but a strong resemblance of the "graph recolouring" heuristic approach proposed by Weisfeler and Lehman [153]. That being said, there are other possible strategies to create a GNN and, in a recent survey [139], the author listed concisely and formally the three main kinds of categories: (i) *convolutional*, the first attempts to generalize the success of convolutional layers over graph structures, (ii) message passing and (iii) *attentional*, nowadays the leading approaches to GDL. Despite their slightly different definition, any GNN layer is composed of a *permutation-invariant aggregator*, such as sum, mean or max, and an *update function*, e.g. a linear transformation of nodes' features and a ReLU activation. The nearly infinite possibilities to create different GNNs led to the foundations of a varied ecosystem of models [165]. Among them, some architectures have become of remarkable importance, not only for GDL but also for document understanding pipelines. The first generalisations of convolution relied on *spectral* approaches [22, 35]: using graph's adjacency and Laplacian matrices and applying a *graph* Fourier transform. These methods could project signals into the spectral domain where they could be processed using linear operations. The first simplified and scalable Graph Convolutional Network (GCN) [75] has been proposed for semi-supervised node classification, nearly equivalent to a GraphSAGE [58] with mean aggregator, in an inductive setting. These two methods are still nowadays among the leading architectures dealing with deep learning on graphs, even if attentional methods are taking over. The Graph Attention Network (GAT) [140], inspired by the famous transformer architecture [138], changed how hidden representations of each node in the graph are computed, attending over its neighbours and following a self-attention strategy. The original architecture has recently been refined in an extensive evaluation named GATv2 [20]: the previous *static* attention mechanism, has been fixed to be *dynamic* by simply applying the attention evaluation after the nonlinearity. Finally, it is worth mentioning also Dynamic Graph CNN (DGCNN) [147], differently from other approaches, is capable of dynamically updating the set graph's interactions between each layer.

In general, the recent success of GNNs and their rapid growth has to be associated with at least three important factors (as also described through the chapters in this thesis):

- **structure matters**: either by using GNNs alone or in combination with vision and language in a multi-modal fashion, structural and layout features have become irreplaceable for any model, as well as for document understanding [148, 155, 160];

- **a cheap architecture, yet effective**: Transformers have opened both new frontiers in machine learning and the need for scalable architectures for edge devices and real-time scenarios [34]; GNNs require a reduced number of parameters, while capable of retaining good inference results. We particularly address this point in Chapter 5, presenting the Doc2Graph framework [46];

- **versatility**: a graph allows different levels of *granularity*, opening to the possibility of solving multiple downstream tasks with a unique representation of the data. An example about table extraction is described in Chapter 3.

In particular, the last point is due to the specific tasks that can be performed on any graphical structure, which can be formally defined as follows:

Definition 2.2.2. ***Graph-related tasks***. *After the application of l subsequent GNN layers, the hidden representations $h_u^{l+1} (\forall u \in V)$ can be used at:*

- ***node-level****: to perform any classification / regression task, e.g. using an MLP and a softmax / sigmoid activation function;*
- ***edge-level****: aggregating nodes' pair feature vectors, to perform any classification / regression task;*
- ***graph-level****: aggregating all the hidden representations using a readout function, e.g. summation, average, maximum or minimum over all node and/or edge features, to perform any classification / regression task.*

As further described in the next section, having a graph representation of a document is extremely powerful: for instance, it allows to perform document classification at the graph level, document layout analysis at the node level, and table structure recognition at the edge level.

2.3 Document structure interpreted as a graph

Statistical and structural approaches have always been the two main ways of representing objects for pattern recognition and, in particular, document analysis. While the first generally makes use of vectors, the second one prefers strings, trees and graphs, the latter being the most general form of data structure [24]. Documents have been always seen and produced as a precise and logical arrangement of objects in relation to each other, naturally following a geometrical structure [59] (the following is summarized in Figure

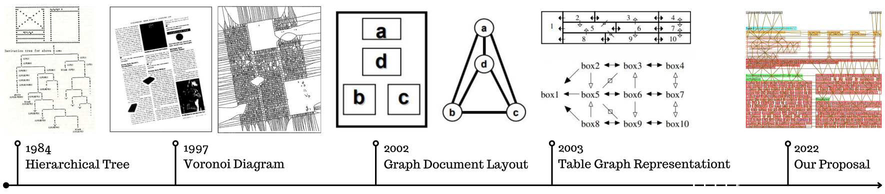


Figure 2.3: How graph-like document representations evolved. From left to right (images taken from original papers): hierarchical tree [101], Voronoi diagram [77], graph document layout [88], table graph [2] and one of ours graph representation proposal [49].

2.3). In 1984 Nagy proposed an X-Y cut algorithm in which the content of an optical scanned document is hierarchically represented as a tree [101]. A few years later, a Voronoi diagram is used for page segmentation [77], where each region is delimited by Voronoi *edges* and *points*. A graph representation of a document page is proposed in [88] to label document regions through graph matching algorithms: quite interestingly the way the graph is obtained is close to modern approaches. An OCR is used to extract document content: nodes are enriched with positional and heuristic features while edges use distance metrics and mutual positions. Another graph representation is proposed by [2], focusing on tables: each cell is a potential node of the graph while edges are built following precise heuristics similar to a visibility graph. Despite their flexibility, being able to encode multiple kinds of relationships within objects and not being constrained to a fixed structure, techniques for pattern recognition and document analysis did not obtain the same success as other statistical competitor approaches. A lack of algorithmic tools for graphs has been observed for years, mainly due to their high complexity and lack of a mathematical structure, until the recent widespread of GNNs. Differently from other domains where the structure is "already given", such as in chemistry or social network applications, applying GNNs for DU also needs to define the document graph. Each recent approach proposes its pipeline to extract **document graph representation**. Nevertheless, it is possible to recognise a common procedure and define a general pipeline to transform documents into graphs (Figure 2.4). The following are the steps shared among different approaches.

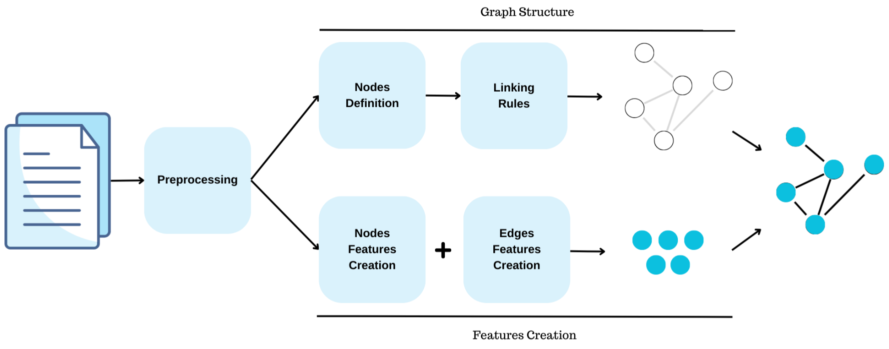


Figure 2.4: A general schema to document graph representation extraction pipelines.

Nodes definition: the first step to build a graph is to define *what* are its constituent objects. To do so heuristic or intelligent system and tools are exploited to perform a *document preprocessing*, usually involving OCRs (e.g. EasyOCR, Tesseract) [121, 161], PDF parsers (e.g. PDF Miner, PyMuPDF) [49, 143] or machine learning models depending on the data type. According to the downstream task, the nodes can be words, lines of text or *entities*, i.e. groups of words that acquire a definite meaning together. The outputs of this step are 2D coordinates of bounding boxes enclosing the aforementioned defined nodes and the recognised text within if needed. As a special case to table extraction-related tasks, nodes can also be table cells [29, 116].

Linking rules: the layout information from the previous step is then used to create the graph edges, following predefined linking rules. This is a very crucial step since the learning algorithms are highly affected by the *quality* of the structure and the *quantity* of the connections. That is why, on this point, the literature has not yet found a common ground, and each work proposes its own solution, differing in minor ways. To formalise this important step without excluding the many nuances in this research area, the two most important linking rules remain:

- **k-nearest neighbours** [27, 29]: based on Euclidean distances, this approach is as simple as it is effective. Several works differ on how these distances are evaluated, i.e. from the centres of bounding boxes or within the 4 or 8 surrounding quadrants identified by the sides of the rectangles; as a particular case, if k is equal to the number of nodes we can refer to this technique also as *fully connected* [46].

- **visibility graph** [122]: within a fixed range two bounding boxes are connected if they are on ”line-of-sights”, i.e. a line can be traced within them with no other intersecting rectangles within them. An interesting alternative has been proposed by [145], called β-skeleton, by changing the linking rule into “balls-of-sight”. In such a graph, two boxes are connected if they can both touch a circle that does not intersect with any other rectangle, providing a good balance between connectivity and sparsity.

In alternative to these approaches, [34] used a trained *edge proposal network* to initialise the graph before applying a multi-layer GCN, which is simply composed of two linear networks with ReLU activation in between.

Node feature vector creation: as outlined in Definition 2.2.1 each node require a feature vector. Depending on the different downstream tasks, this information can be obtained by a concatenation of different modalities, choosing between:

- **visual features**: pre-trained (or trained end-to-end) visual backbones, e.g ResNet [64] or U-Net [123], plus ROIAlign layers are usually involved to scale hidden feature maps to bounding box regions, to encode font type, styles, colors (if any) and other discriminative visual patterns;

- **layout features**: scaled coordinates of bounding box regions, represented by all (or any) terms of this set: $(x_{min}, y_{min}, x_{max}, y_{max}, x_{ctr}, y_{ctr}, w, h)$ representing left-bottom and right-upper corners and rectangle center, width and height, respectively; usually this information is encoded using fully connected layers or 2D positional embedding [155];

- **textual features**: pre-trained (or trained end-to-end) word vectors and/or transformer-based architectures, e.g. SpaCy [42] or SciBert [12], are usually involved to encode document extracted text. [122] uses a histogram to count the number of numeric, alphabet or symbol elements to keep the content anonymous after the OCR application. [116] uses only the length of the word.

- **typographic features**: additional information about text. These features include font type, size and other related kinds of information, usually found and only available in PDF documents.

Edge feature vector creation (optional): some approaches include additional information on edges, which may include reading order, edge direction, node distance and polar coordinates, to name a few.

A GNN for document understanding was first introduced for table structure recognition by classifying edges for cells, rows, and columns [116] and for table detection in invoice documents [121]. In particular the latter, using a GAT in a recent extension in [122], proposes a language-independent approach for privacy issues related to business documents. As a natural extension, a similar approach has been used for form understating [27], to solve the word grouping, labelling and entity linking tasks proposed by FUNSD [69]. It is interesting how a second method called FUDGE [34] could be able to drastically reduce the number of parameters required by the GAT simply by applying a three-layer depth GCN on top of a ResNet50 and still achieve an improvement in results for the same task. Finally, GNNs have been applied also to the broader task of DLA. Either combining them in a multi-modal fashion along with visual and semantic features [160] or using a DGCNN on top of a novel graph sampling method [148], graph-based techniques hold the state of the art for DLA benchmarks [110, 164], achieving better results than traditional deep learning methods.

This chapter has primarily aimed to give the reader a general overview of how the paths of graph theory and document understanding have intersected in the world of research and, in general terms, a definition of *document graph representation* that will be recalled throughout the following chapters. This thesis will now go on to describe more specifically what our proposals have been for structured objects, such as tables, and document pages using graph-based methods.

Part I

Structured Document Objects

Chapter 3

A graph-based method for Table Extraction

Tables are widely used in several types of documents since they can represent important information in a structured way. In scientific papers, tables sum up novel discoveries and summarize experimental results, making the research comparable and easily understandable by scholars. Several methods perform Table Extraction working at image-level, losing useful information during the conversion from PDF files. The application of OCR tools can be prone to recognition errors, making table structure and textual content misinterpreted. Moreover, these approaches are considering tables alone or require multiple models trained separately to perform different tasks on tables. Exploiting a Graph Neural Network on top of PDF parsing tools, overcomes all the aforementioned limitations. A graph-based approach is able to perform Table Extraction at once, also taking into consideration the surrounding document regions. This chapter is mainly based on this work [49].

3.1 Introduction

Nowadays documents are usually produced and shared as PDF files, due to their capability of rendering the document content in a faithful way on different platforms and devices. Developing systems able to correctly extract information from them is crucial for any document understanding pipeline. Intelligent systems need lots of data to be trained and to generalise on different distributions; moreover, PDF documents need to be accessible and labelled. Since there exist several datasets with these features and they are mainly composed of scientific articles [47], we focused our attention on these kind of documents. Despite the wide use of PDF files, 95.5% of published articles in PDF format are not semantically tagged [102]. Therefore, extracting information from these documents remains a difficult problem. This is particularly important for tables that are generated taking into account the semantic information (e.g. from LaTeX or MSWord) that is lost in the PDF. As a consequence TE, meaning detecting tables, recognising their structure, and analysing their contents, remains a challenging task as demonstrated among competitions on table analysis to deal with both born-digital and scanned documents [45, 52]. Nevertheless tables are commonly used as a compact and efficient way for describing statistical and relational information [146] and it is essential to efficiently extract and analyze them, e.g. to compare SOTA results in scientific papers [72]. So far, most of the works consider image-based approaches; as shown in [61] recent methods often exploit Computer Vision and Natural Language Processing (NLP) techniques to deal with tables and OCR tools are employed whenever text is needed. Since scientific papers are currently distributed as born-digital PDF files it is appropriate, in our view, to look at the information in the PDF file without relying on error-prone OCR tools [18] [19]. However, extracting information from PDF documents is not an easy task. The first techniques for extracting information from tables explored solutions for PDF documents taking into account heuristics object-based approaches. These methods analyze both textual and positional information often relying on heuristics [62, 98]. Differently, various PDF parsers with different features can be used. Some of them only extract basic PDF elements such as text, images, and graphical items. Others are able to extract additional information specifically for scientific papers, such as title, authors, and abstract [94] [1] [96]. In other cases, it is possible to extract more complex items (e.g. tables) [25], but these tools require an accurate location of the table, otherwise they may fail when tables are surrounded by text [5, 73]. The latter techniques often rely on low-level tools for PDF parsing (e.g. PDFMiner [107] MuPDF [7], or PDFBox [3]). Among others, we preferred PyMuPDF [115] being reliable and well-documented, providing token-level objects needed to build the graph.

Taken into considerations the aforementioned challenges and limitations, the main contributions of our work are:

1. **the redefinition of the TE task** as a node classification one, addressed by a GNN. Graph nodes are composed of basic PDF objects while edges are computed considering relationships and mutual distances between nodes. Our experiments show that GNNs are a well suited solution for TE;

2. the graph nodes are augmented using a **novel textual embedding** of different representations for numerical and non-numerical values. These embeddings are learned over table cells elements taking into account the PubTables-1M dataset [131]. Our experiments demonstrate the efficiency of the proposed representation embeddings in conjunction with the node positional information. Ablation studies are conducted exploring also other word embeddings;

3. **a new dataset** is collected by merging the ground-truths of two widely-used datasets for DLA [164] and for TE [131]. This novel dataset allows us to perform both tasks at the same time.

3.2 Related Works

Table Understanding (TU) consists of three steps [61]: Table Detection (TD), Table Structural Segmentation (TSS), and Table Recognition (TR). TSS is referred to as Table Structure Recognition (TSR) in [131] where the recognition of column and projected row headers is defined as Table Functional Analysis (TFA). To perform TD table boundary coordinates are detected. This task is often performed in the image domain [28] and recently approached with object detection techniques. Usually, models like Faster-RCNN and Mask-RCNN [164] [84] are used. On the other hand, at the token level, NLP-based methods are involved in using both textual and visual features, such as LayoutLM [155] in [85]. Once tables are found, their structure is recognized by identifying their rows, columns, and cell positions. A Cascade Mask R-CNN (CascadeTabNet) is used in [114] to detect tables and body cells, arranging them in columns and rows based on their positions.

Table 3.1: Comparison of tasks performed by different methods. (*future extension of our proposed method).

Methods	**Tasks**			
	DLA	TD	TSR	TFA
LayoutLM [85]	✓	✓		
VSR [160]	✓	✓		
Riba et al. [121]		✓		
CascadeTabNet [114]		✓	✓	
LGPMA [117]			✓	
TGRNet [157]			✓	
Ours	✓	✓	(✓*)	✓

However, none of these methods take into account the document structure and require separated subsequent steps to deal with tables. On the contrary, as outlined in Chapter 2, the graph representation of a document allows to solve several tasks at once. In this scenario, GNNs can tackle TE bringing advantages beyond solely relying on visual and/or language features.

3.3 Problem Formulation

The term TE appears back in 2003 [111], aiming at labeling each line of a document with a tag and describing its function relative to tables. A more recent work [131] proposes another meaning for TE, providing TD, TSR, and TFA annotations. With TE, we refer to the task of detecting tables and extracting the meaning of their content at once, at the token level. To this purpose, we adopt a GNN to tackle TE as a node classification problem. To the best of our knowledge, our method is the only one addressing TE and DLA at once. In Table 3.1 we compare the sub-tasks performed by some methods described in Section 3.2 and Chapter 2.

In order to perform TE, we collected a new dataset as described in the following section. To do so, we merged the data and the annotations given by the PubLayNet and PubTables-1M datasets, both based on PubMed Central publications. The merged dataset contains 13 different classes adding to the regions annotated in PubLayNet, the table annotations described in PubTables-1M (*row, column, table header, projected header, table cell*, and

Table 3.2: Comparison of original and merged datasets (Document Layout Analysis (DLA), Table Detection (TD), Table Structure Recognition (TSR), and Table Functional Analysis (TFA)).

Datasets	**# Pages** (train / val / test)	**# Tables**	**Tasks**	**# Classes**
PubLayNet	336k / 11k / 11k	107k	DLA, TD	5
PubTables-1M	460k / 57k / 57k	948k	TD, TSR, TFA	7
Merged	67k / 1.5k / 1.5k	27k	DLA, TD, TSR, TFA	13

grid cell). Moreover, we add the two classes *caption* and *other*, the latter being all the remaining not-labeled text-regions (e.g. page headers and page numbers). Details are summarized in Table 3.2. We further expanded the dataset and the task itself, giving it the name of Contextualized Table Extraction as extensively described in Chapter 4.

3.4 Method

In the next sections we illustrate how a PDF paper is handled to build its graph, adding explanations about node and edge features and representation embeddings and the strategies used to handle class imbalance during training. Finally, we describe the message passing algorithm applied to train the GNN. The whole pipeline, from documents to layout inference, is summarized in Figure 3.1.

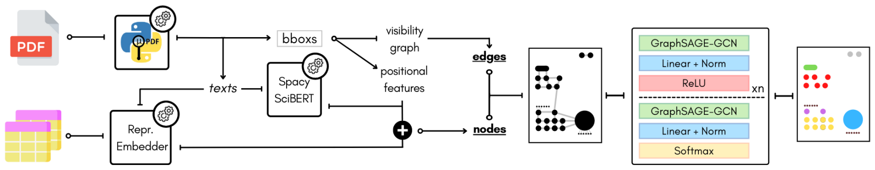
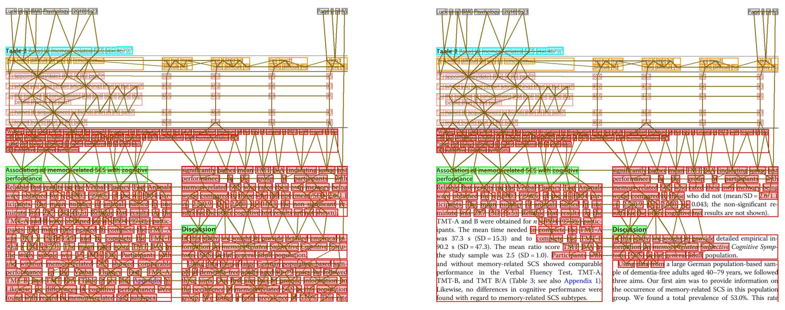


Figure 3.1: Overview of the proposed method.



(a) Whole graph (b) Sub-graph for class balancing

Figure 3.2: Graph of a portion of page. Different types of nodes have different colors. (red: text, green: title, pink: table cell, orange: table header, light blue: caption, grey: other)

3.4.1 Converting PDF pages to graphs

Graphs are generated from PDF files in three steps:

1. information about basic items (tokens) in PDFs are extracted by using PyMuPDF;

2. each node is connected to its nearest visible nodes according to the visibility graph [121];

3. features are added to each node and edge.

Following the general schema provided in Chapter 2, we enrich our graph nodes with positional and textual features. We use new representation embedding features (Section 3.4.2) that help the model to better discriminate table cells and headers from the rest, performing also TFA. Inspired by [121] we make use of edge weights and enrich graph edges with token boxes distances, letting closer elements contribute more to the message passing algorithm (Section 3.4.3). Node features are a combination of geometrical and textual information as shown in Figure 3.3. The geometrical features are $< x_1, y_1, x_2, y_2, w, h, x_c, y_c, A >$, where x_1, y_1, x_2, y_2 are the corners of the bounding box having width w, height h, center x_c, y_c and area A. Other node features describe the textual content from different perspectives: a) inspired by [121] we add three values <% of characters, % of digits, % of symbols> to

Algorithm 1 Edge weight w_e

```
Require: $u, v \in V, e = (u, v) \in E$
  if $(u$ above $v) \vee (u$ below $v)$ then
    $d_e \leftarrow \max(u.y_1, v.y_1) - \min(u.y_2, v.y_2)$
  else if $(u$ left of $v) \vee (u$ right of $v)$ then
    $d_e \leftarrow \max(u.x_1, v.x_1) - \min(u.x_2, v.x_2)$
  end if
    $w_e = 1 - \frac{d_e}{\max_{e \in E}\{d_e\}}$
```

better distinguish items in tables from other page contents. The values are the percentage of characters, digits, and symbols in the node, respectively; b) we add a boolean value for *images* that identifies images recognized by PyMuPDF; c) tokens are described with the proposed representation embedding: as described in Section 3.4.2 they are more informative if containing digits or symbols; d) static NLP-based embeddings (SciBERT [12] and Spacy [42]) are also considered among node features. The feature of edge (u, v) is the distance between bounding boxes of u and v as defined in Algorithm 1. To handle the graph, we use the DGL open-source library [144]. An example of the graph corresponding to a portion of a page is shown in Figure 3.2a.

In scientific papers, most of the nodes belong to paragraphs and are labeled as “text” and correspond to more than 80% of the whole dataset. During training, we deal with this class imbalance by excluding pages without tables and by discarding some “text” nodes in the remaining pages. A “text” node v is discarded if there is a path with more than k edges from v to any other node u with a different label in the original graph. Discarded nodes are called ”islands”. By removing islands it is possible to reduce the number of nodes surrounded by others of the same class. In this way, the message passing algorithm aggregates more messages coming from different sources helping the method to discriminate objects. In Figure 3.2b we show the graph obtained by removing islands in the whole page graph.

3.4.2 Representation embedding

Even though tables in documents can be recognized by only using the layout information of document objects, textual information can help for this task. Common methods for representing textual information rely on word embeddings. Static embeddings represent isolated words (i.e. Word2Vec [100], GloVe [108]) while contextualized embeddings provide different word representations according to different contexts (i.e. Elmo [109], BERT [38]). Word embeddings are extremely useful in several NLP tasks; however, it is difficult to represent numbers, formulas, and intervals that often appear in tables. Recently, [70] defined a new Word2Vec approach for enhancing numerical embeddings. Word embeddings can hardly learn numerals as there are an infinite number of them and their individual appearances in training corpora are rare. Two different representations are provided for words and numerals: the latter are represented by prototypes obtained by clustering numerals with SOM or GMM. Numerals are represented either by the closest prototype or by a weighted average of the closest prototypes.

Inspired by [70] we propose a representation embedding to handle formulas and intervals in addition to words and numerals as described in Section 3.4.2. Table headers are often either words or word-numeral combinations, while table cells mainly contain numerals or a combination of numerals with other symbols (e.g. numbers and intervals). In some cases (e.g. in tables comparing SOTA papers) all the cells contain words, but this is not very frequent in our dataset. For each token corresponding to a graph node, we obtain the representation embedding by first mapping the token into a standardized representation and then embedding the representation in a dense vector. A representation is a combination of symbols (e.g. '±', '+', '°') and the x and w characters that correspond to sequences of digits and words, respectively. Examples of representations of tokens are: "*Precision-Recall*" → "*w-w*"; "12.5" → "$x.x$", "+3.1(2.5± 1.0)"→ "$+x.x(x.x \pm x.x)$". Algorithm 2 describes the function *word2repr* that maps a token ($word_i$) to its representation ($repr_i$). In PubTables-1M there are more than 50,000 different representations that provide an overview of the various contents of a scientific table. To embed the representations, we induce a set P of prototypes obtained by clustering the l = 2000 most frequent representations

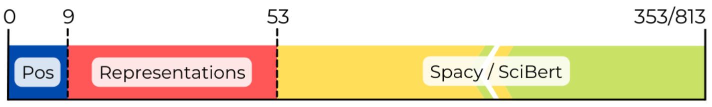


Figure 3.3: Node features: positions and representations have a fixed length; Spacy or SciBERT embeddings have a length of 300 and 760, respectively.

Algorithm 2 Word to Representation

```
Require: $len(word_i) > 0$
Ensure: $repr_i = word2repr(word_i)$
  $repr_i \leftarrow word_i$
  $repr_i \leftarrow repr_i.replace(/[A - Za - z]/g, "\text{w}")$
  $repr_i \leftarrow repr_i.replace(/[0 - 9]/g, "\text{x}")$
  $repr_i \leftarrow repr_i.sub(r"(.)\backslash 1+", r"\backslash 1")$
```

in the dataset. The clustering is obtained by computing with the Levenshtein distance [43] the distance matrix $D_{l \times l}$ (corresponding to distances between representations). The Affinity propagation algorithm is then applied to the $D_{l \times l}$ matrix to compute the P prototypes (in our experiments we have $P = 47$). Following [70], we define a similar process to embed token representations by training a Word2Vect model with SkipGram negative sampling. To train the Word2Vect over the training set T of tables, we consider three ways to *visit* table cells and define context elements and the target one. Given a table $t \in T$, and considering a sliding window of size $w = 5$, we extract for each cell $c^t_{i,j}$ (i= row, j= column of table t) a list of neighboring cells arranged following one of three patterns:

Headers: $c^t_{i,j} \Rightarrow [c^t_{i,0}, c^t_{i,j-1}, c^t_{i,j}, c^t_{i-1,j}, c^t_{0,j}]$
Rhombus: $c^t_{i,j} \Rightarrow [c^t_{i,j-1}, c^t_{i-1,j}, c^t_{i,j}, c^t_{i+1,j}, c^t_{i+1,j+1}]$
Linear : $c^t_{i,j} \Rightarrow [c^t_{i,j-2}, c^t_{i,j-1}, c^t_{i,j}, c^t_{i,j+1}, c^t_{i,j+2}]$

For instance, for the table in Figure 3.4, taking $c^t_{3,2} = v_9$ we obtain the following patterns:

Headers: $c^t_{3,2} = v_9 \Rightarrow [r_c, v_8, v_9, v_5, h_b]$
Rhombus: $c^t_{3,2} = v_9 \Rightarrow [v_8, v_5, v_9, v_{13}, v_{10}]$
Linear : $c^t_{3,2} = v_9 \Rightarrow [r_c, v_8, v_9, v_{10}, v_{11}]$

In the current system we use the Rhombus method since it provides better results and most likely reflects the graph structure for a center node. Once we have the embeddings for the P prototypes, the ones for other representations can be computed by associating each of them to its closest prototype embedding or by computing a weighted average of all the prototypes. In

	h_a	h_b	h_c	h_d
r_a	v_0	v_1	v_2	v_3
r_b	v_4	v_5	v_6	v_7
r_c	v_8	v_9	v_{10}	v_{11}
r_d	v_{12}	v_{13}	v_{14}	v_{15}

Figure 3.4: Example of table to illustrate representation embedding.

some preliminary tests, we found that the first approach provides more informative vectors. The resulting representation embeddings are employed in the node feature vectors.

3.4.3 Custom message passing

In this work we apply a variant of the GraphSAGE algorithm [58]. The information flows through the graph aggregating node features from neighbors. As this process iterates, nodes incrementally gain more information from farther ones. Given a graph $G = (V, E)$ each node $v \in V$ has its own feature vector h_v (Figure 3.3) and it collects information from neighbors $N(v)$, whose vectors are called *messages*. Most algorithms copy the *messages* into the so-called *mailbox*, but in our case we scale them by an edge weight w_e computed in Algorithm 1. The weight w_e is the normalized spatial distance of nodes u and v, it reaches the maximum value ($w_e = 1$) on touching nodes. In so doing, nearest nodes contribute more to the information flow under the hypothesis that local nodes often belong to the same class. At step k, each feature vector in the neighborhood of v is collected in its mailbox $m_v^k = \{w_e h_u^{k-1} \mid \forall u \in N(v)\}$. Then each node aggregates messages using a permutation-invariant differentiable function, such as pooling, mean, or sum. We sum messages to compute a weighted average of feature vectors of neighboring nodes: $h_{N(v)}^k = \frac{\sum_{m \in m_v^k} m}{|N(v)|}$ We then concatenate the current node feature vector h_v^{k-1} with the aggregation of neighboring nodes $h_{N(v)}^k$ and each node updates itself: $h_v^k = \sigma(W \times CONCAT(h_v^{k-1}, h_{N(v)}^k))$ where W is the weights matrix of a fully connected layer, applied to learn different patterns in feature vectors. Other approaches add h_v^{k-1} directly to messages or just update it with $h_{N(v)}^k$.

3.5 Experiments

In this section, we discuss the performed experiments. At training time, only pages containing tables are used (around 27k) reserving 5% of them for validation. The test set contains 1.5k pages including also pages without tables. To evaluate the performance of the proposed method, model accuracy and F1 scores are considered. The F1 metric is used for table cells and table header classes. In this section, we discuss the methods and the corresponding results reported in Table 3.3.

3.5.1 Ablations

Each method differs from the others varying the hidden layer dimensions h_{dim} and the number of parameters p_{no}. After fixing the number of GNN layers ($l_{no} = 4$), three different methods are employed. *Base*, the first one, fixes the hidden dimension $h_{dim} = 1000$ for all input features size in_{dim}. By doing so, the network size changes with the different sizes of the input elements. In the method called *Padding*, instead, the size of the network is fixed by padding the input size and forcing it to have dimension: $in_{dim} = (bbox + repr + max(Spacy, SciBERT)) = 861$. Finally *Scaled* fixes the number of network parameters $p_{no} = 100k$ by reshaping the hidden size h_{dim} on the basis of the input size, according to the following second degree equation: $p_{no} = (l_{no} - 2) * h^2_{dim} + (in_{dim} + out_{dim}) * h_{dim}$. The main difference between *Padding* and *Scaled* is related to the first layer parameters which are not completely used by *Padding* since some input values are always set to zero.

3.5.2 Results

Experimental results (summarized in Table 3.3) are designed to answer two main questions. Firstly, whether the proposed representation embedding improves the performance on table detection and discrimination of cells and headers. Secondly, if the representation embedding is a good alternative to language models. Concerning the first question, we can notice that in general by adding the representation embedding we have better performance for the detection of table cells (cell F1). Model B adds representation embeddings to the positional information of model A and, in this case, the cell F1 score increases for all methods (e.g. for Padding by 5.4 %). cell-h F1 scores are lower

Table 3.3: Evaluations are conducted with accuracy on all classes and F1 score on table cells and headers. Each model (A, B, ..) has been tested with different combination of features.

Model	Features	Metrics	Methods		
			Base	*Padding*	*Scaled*
A	bbox	accuracy	0.873	0.841	0.866
		cell F1	0.798	0.765	0.799
		cell-h F1	0.659	0.651	0.642
B	bbox + repr	accuracy	0.876	**0.875**	0.873
		cell F1	0.821	0.819	0.816
		cell-h F1	0.653	0.649	0.648
C	bbox + Spacy	accuracy	0.859	0.847	0.868
		cell F1	0.767	0.773	0.781
		cell-h F1	0.685	0.675	0.660
D	bbox + repr + Spacy	accuracy	0.865	0.860	0.809
		cell F1	0.780	0.776	0.811
		cell-h F1	**0.689**	0.675	0.644
E	bbox + SciBERT	accuracy	**0.882**	0.843	**0.879**
		cell F1	0.838	0.816	**0.846**
		cell-h F1	0.688	**0.699**	**0.686**
F	bbox + repr + SciBERT	accuracy	0.709	0.787	0.870
		cell F1	**0.855**	**0.832**	0.777
		cell-h F1	0.668	0.636	0.671

because headers often contain words that are not well modeled by representation embeddings. Furthermore, if we add the SciBERT word embeddings to representations, we notice that model F outperforms all the others in the detection of table cells (using the *Base* method). With respect to model E the cell F1 score increases by 1.7%. Regarding the second question, we can notice that representation embedding alone achieves good results. For cell F1 score, model B obtains almost the same results compared to C (where Spacy replaces the representation embedding) and E (where SciBERT replaces the representation embedding). About the accuracy, model B outperforms C over all the methods and E with padding.

The results can also be verified qualitatively by looking at inferences of three pages in Figure 3.5. For each page, we present the graph node predictions (pages with tokens), clusters of nodes belonging to the same class and the grouped entities (pages with object boundaries). The majority of table headers and cell nodes are well recognized. Some errors are present mainly near region boundaries: e.g., at the top of table bodies, some table cells are misclassified as headers. As a first step towards page objects identification, we apply a post-processing phase based on PyMuPDF *blocks*. The library detects groups of entities that are then labeled with the majority class among them. Even if this approach works most of the times, relying on simple rules is prone to errors. In second row, it is possible to see that PyMuPDF fails (f) because there is too much space withing the table and it outputs overlapping boxes.

3.6 Conclusions and Future Works

We presented a pipeline to perform DLA and TE in PDFs of scientific papers. We propose to represent PDF papers as graphs and redefine the problem of TE as a node classification problem by means of GNNs. We enhance node features using novel representation embeddings that we empirically prove to be effective to discriminate table elements from other classes. Even if the post-processing provides promising results, to improve them future works will investigate and include edge classification in the GNN model to group elements belonging to same entities. In this way it will be possible to compare this approach with both TD and TSR methods. We also aim to further investigate the representation embedding by studying its properties.

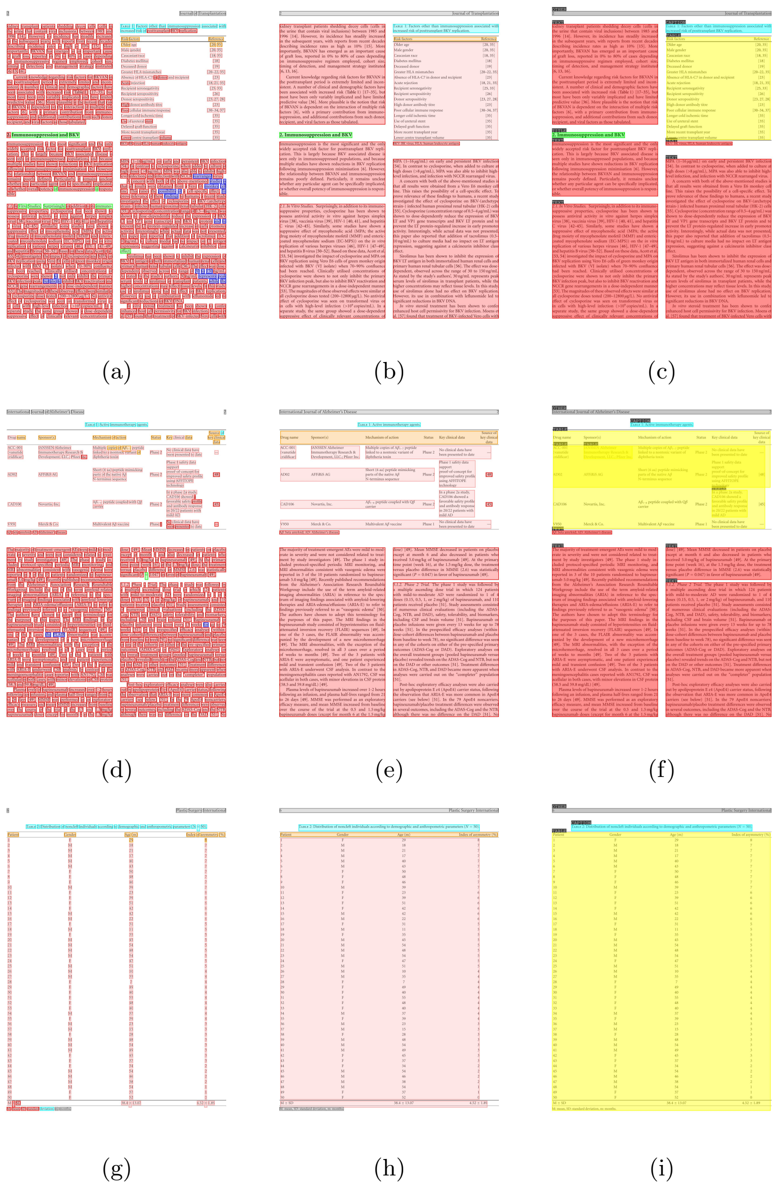


Figure 3.5: Three examples of model inference per row at token-level (left), clustering (center) and post-processing (right). Different colors represent different types of nodes or blocks (red: text, green: title, pink: table cell, orange: table header, light blue: caption, grey: other, dark blue: list, yellow: table).

Chapter 4

A dataset and augmentation approach for graph representation of tables

As shown in the previous chapter, extracting tables from documents, and particularly from scientific articles, is as important as it is challenging. Usually, Table Extraction is tackled using traditional Computer Vision or Natural Language Processing methods that totally or partially disregard the important structural information. This is also partly due to datasets that do not explicitly expose this feature in their annotations. Consequently, the application of graph-based methods, although attracting increasing interest, is limited by the scarcity of data with structural information. For this reason, we have created a new collection and task, called Contextualized Table Extraction, to meet these limitations. Moreover, we proposed a new augmentation approach directly on the graph representation of tables, tested for the task of table type classification. This chapter is mainly based on these two works [36, 50].

4.1 Introduction

Nowadays, large collections of documents require a huge amount of human work to annotate documents and extract important information. In the last thirty years, the DAR community tried to overcome this challenge, exploiting suitable algorithms and artificial intelligence techniques to automatize the analysis of documents and reduce its costs. Among others, DC, DLA and TU more broadly attracted the interest of researchers and companies. Moreover, collections of scientific papers such as arXiv and PubMed opened to the possibility of accessing a large number of documents along with their structural information represented in standard formats such as LaTeX and XML. That is why scientific literature parsing and scientific table analysis rapidly became one of the most prominent areas of research in DAR: large datasets have been released [131, 164], allowing the community to develop deep learning models. Unfortunately, as we will describe in the next sections, these datasets come with partial information that forces the experimentation of DLA and TE. From this identified lack, we define Contextualized Table Extraction (CTE), a broader task that comes along with novel annotations for a collection of 75k scientific pages containing more than 35k tables, encouraging the development of new systems capable of tackling a multitude of tasks at once. CTE is formulated as a token and link classification task, encouraging the usage of graph-based methods which are widely used in applications where the structure and layout in documents matter.

This chapter introduces the following contributions:

- We define the task of **Contextualized Table Extraction** (CTE), an extended version of TE as defined in [131] that adds layout information and encourages the development of end-to-end systems that can tackle multiple tasks at once;

- Novel annotations are created by merging subset of [131, 164] and extending the collections presentend in [49]. The **new collection** includes 75k scientific pages and more than 35k tables. Tokens at the basis of annotations correspond to words extracted from PDFs using PyMuPDF and labeled according to the region they belong to; table structure information is encoded as links between tokens;

- The dataset **encourages the use and development of graph methods on documents**, providing to the community a new set of labeled

Table 4.1: Comparison of CTE with related datasets: ♣ denotes the datasets used to generate the new annotations. A dataset is *S4G* (suitable for graphs) if a graph can be constructed directly with no further preprocessing. DLA (Document Layout Analysis), TD (Table Detection), TSR (Table Structure Recognition), and TFA (Table Functional Analysis) show which tasks models can be trained for.

Dataset	#pages	#tables	#classes	DLA	TD	TSR	TFA	S4G
PubLayNet (♣)	358k	107k	5	✗	✓	✗	✗	✗
PubTables-1M (♣)	574k	948k	7	✗	✓	✓	✓	✗
DocBank	500k	417k*	12	✓	✓	✓*	✗	✓**
SciTSR	0	15k	-	✗	✗	✓	✗	✓
CTE	75k	35k	13	✓	✓	✓	✓	✓

*DocBank is an extension of TableBank, from which we gathered these information

**If tokens used as graph nodes, no information on edges

data to experiment with GNN-based techniques. The annotations do not require any further processing (either in labels or data themselves) to construct a graph over the scientific pages.

- We also propose **a new augmentation technique for table structured data**, directly on their graph representation. The method has been tested on the Tab2Know dataset for table classification.

4.2 Related Works

Despite the advances in the field, several challenges strongly limited the generalization of methods developed until a few years ago, as outlined in Chapter 1. To address these challenges a large number of data need to be collected in order to fully exploit the power of Deep Learning models that achieve the state-of-the-art for the aforementioned tasks. Unfortunately, creating such datasets is nothing but trivial since accurate annotations come at a high cost in terms of time and human effort [110, 129]. On the other hand, automatic annotation techniques are not always applicable since they require a large number of documents shared together with their source files in standard formats such as LaTeX, XML, or HTML [85, 164]. Additionally, these techniques usually generate weakly labeled collections and are more error-prone than manually annotated ones. A third solution could be using data augmentation techniques, either based on heuristic or generative approaches, in scenarios where few labeled samples are available and/or more variability in layout and contents its needed. These points are discussed in more detail in Chapter 6, where the aforementioned limits are met by a proposed generative method.

Other document collections. We compare our collection with some of the most important datasets proposed for DLA nad TE in in Table 4.1. PubLayNet and DocBank have been widely used to train object detectors [63,120] and transformers [155] for DLA. Overall, these datasets contain around half a million pages labeled into five and twelve different classes, respectively. PubLayNet has been constructed merging the information extracted from PDFMiner (bounding box regions) and the XML files shared by the publishers (containing the region labels). DocBank is built gathering the LaTeXsource files and assigning labels taking into account the section tags. For the TE task, a recent dataset has been released (PubTables-1M) which counts nearly one million tables, labeled to perform not only TD and TSR but also Table Functional Analysis (TFA) that provides additional information on table cells like table headers. Even if it is smaller, SciTSR [29] introduced a collection of 15k tables generated from LaTeX to perform TSR, mainly using a GNN. As it is possible to notice in Table 4.1, all these datasets lack a comprehensive and broader set of annotations, forcing the community to develop multiple systems that, in application scenarios, would lead to heavy and large pipelines.

Data Augmentation techniques. In object detection, DA techniques involve color operations (contrast, brightness), geometric operations (translations, rotations), and bounding box operations [166]. None of these can be used in our case since we are considering graphs to represent the tables and augmentation operations commonly used in vision and language have no analogs for graphs [162]. Similarly to what we did for trees [8] and inspired by [74], we applied some of their augmentations on table examples directly in the graph structure). Operations that can be performed on tables are random deletion of rows, row replication, column deletion and column replication. Instead of working directly on images, we therefore extract the table structure and then apply DA on their graph representation, by means of node deletion, edge deletion and inversion of node contents.

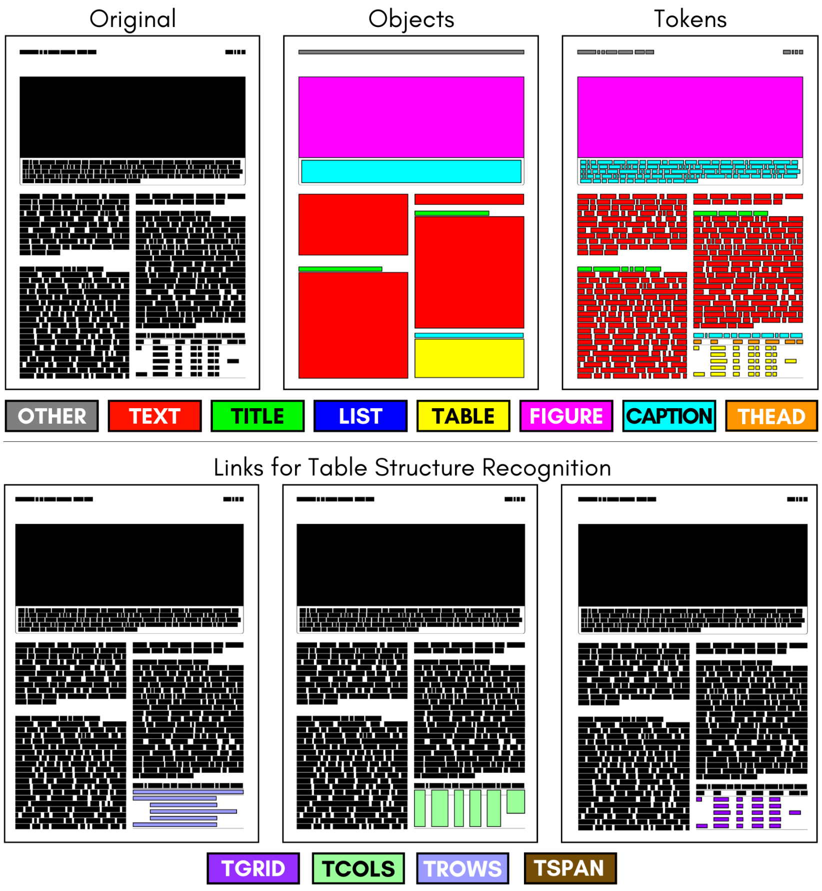


Figure 4.1: Example page (content is intentionally concealed in 'Original') and corresponding CTE annotations. Objects represent the layout regions. Tokens contain the word tokens labeled according to the class in the top of the figure. Acronyms for table annotations are: THEAD (table headers), TSPAN (table sub-headers spanning along different columns), TGRID (table cells), TCOLS and TROWS (respectively columns and rows of the tables).

4.3 A dataset for CTE

The proposed dataset for CTE is obtained by merging data and annotations given by PubLayNet and PubTables-1M datasets, both based on PubMed Central publications. As depicted in the next sections, firstly we identify the pages of scientific papers annotated in both datasets, then we merge the information and add two novel classes (captions and page information) and finally use PyMuPDF to extract text and position of tokens. We used a preliminary small version of this collection in [49], applying a GNN to tackle CTE. After the release of PubLayNet test set we updated the version of CTE dataset, now containing more annotated data.

4.3.1 Problem Formulation

CTE is the broader task of extracting tables (meaning their detection) recognizing their structure and performing functional analysis, along with other page layout information. To do so, CTE is formulated as token and link classification tasks, similarly to [85], since fine-grained objects like tokens permits to tackle multiple tasks at once. For instance, recognizing the table headers and grid cells allows to detect the tables (grouping tokens together through links) and add functional information. In addition, through token and link classification the need for more components would be reduced since a method capable of successfully solving CTE would require us to train only one model, extracting more information at once. Given Precision and Recall for token and link classification, namely Token Precision (TP), Token Recall (TR), Link Precision (LP), and Link Recall (LR). We can define the $F1_{CTE}$ metric as follows:

$$F1_{CTE} = \frac{F1_{Token} + F1_{Link}}{2} = \frac{TP \cdot TR}{TP + TR} + \frac{LP \cdot LR}{LP + LR}. \tag{4.1}$$

Token classification. The first step required to tackle CTE is the classification of tokens, extracted from PDF pages using PyMuPDF. Tokens contain textual and positional information, along with class information inherited from the larger region they belong to (details in Table 4.2, token annotations). This subtask exposes these properties:

1. Through token classification it is possible to achieve DLA, TD, and TFA at once.

2. If tackled along with link classification to achieve CTE the $F1_{CTE}$ metric (Eq. 4.1) should be used. If tackled alone the metric proposed in [85] can be used as well.

Link Classification. In order to group together tokens belonging to tables into columns, rows, or grid cells, additional information on links among pairs of tokens is added. This subtask exposes these properties:

1. Through link classification it is possible to perform TSR.

2. Similarly to token classification, F1 is preferred to evaluate link classification if tackled alone.

3. Links connecting non-tables items should be considered as an additional class '*none*'.

Object Recognition. Even if not required to do CTE, the annotations include area information of different regions in the paper (as common for object detection). Grouping together tokens belonging to the same class via edges can be exploited to find such areas, e.g. extracting sub-graphs from the whole document. A recent paper [145] exploited GNN to perform post-OCR paragraph recognition by grouping together similar items in the pages.

4.3.2 Subset of PubLayNet and PubTables-1M

PubLayNet is a collection of 358,353 PDF pages with five types of regions annotated (*title, text, list, table, image*) [164]. PubTables-1M [131] is a collection of 947,642 fully annotated tables, including information for table detection, recognition, and functional analysis (such as identifying *column headers, projected rows*, and *table cells*). The datasets are built to address different tasks, as summarized in Table 4.1.

To merge the datasets, we first identify the papers belonging to both collections. From this subset, we keep pages with tables fully annotated in PubTables-1M and pages without tables: this filters out even more pages, since we found some PubTables-1M annotations to have only one annotated table in pages containing two or more tables. Following this step, we obtain approximately 75k pages. The resulting merged dataset contains objects labeled into 13 different classes, having in addition to the regions annotated in PubLayNet the table annotations described in PubTables-1M (*row, column, table header, projected header, table cell*, and *grid cell*). Moreover, we added two classes: *caption* and *other*. *Captions* are heuristically found taking into account the proximity with images and tables, while the *other* class contains all the remaining not-labeled text regions (e.g. page headers and page numbers). The remaining PubLayNet train, validation and test documents kept the same split in our own collection.

4.3.3 Annotation procedure

Once a complete annotated list of pages is selected from the two datasets, we leverage an external tool to extract page tokens. After comparing several tools, we opted for PyMuPDF [115] which is a Python open-source library backed by a large community and constantly maintained. Each element, visible or not visible, present in the PDF page is extracted and annotated based on the annotation bounding-box it appears in, as depicted in Figure 4.1: tokens are labeled according to their enclosing labeled region (upper part); links, instead, are presented as groups of tokens for visualization purposes (bottom part), but encoded as couples as described in details in the next Section and in Table 4.2. By doing so, the resulting page is composed by extracting page tokens along with their position (bounding boxes coordinates) and their textual content (mostly single words). This process heavily depends on original versions of the PDF files: even if the document name is the same along the two datasets annotations (PubLayNet and PubTables-1M) the PDF version of PubLayNet documents could differ. This is due to the two year gap between the datasets' release dates. To obtain reliable information, in our approach we discard all the pages (and tables) in which the content of the two sources does not correspond anymore.

4.3.4 Dataset structure and format

After the merging procedure, we end up with three JSON files (subset of the original PubLayNet one) splitting the data into train, val, and test. Each one contains information regarding tokens extracted by PyMuPDF, their links and the regions that group them (larger objects). Tokens have these information: *token id, bounding box coordinates, text, class id*, and *object id* (larger region to which it belongs). Links between tokens (belonging to the same row, column or grid cell) have information such as *link id, class id*, and *token id* (list of tokens linked together). Finally, objects contain information such as *object id, bounding box coordinates* and *class id.* A representation of

Table 4.2: Annotation Format: each line contains different information in case of Objects, Tokens, or Links.

Object annotations

Index	**0**	**1,0**	**1,1**	**1,2**	**1,3**	**2**
Content	object id	x0	y0	x1	y1	class id

Token annotations

Index	**0**	**1,0**	**1,1**	**1,2**	**1,3**	**2**	**3**	**4**
Content	token id	x0	y0	x1	y1	text	class id	parent object id

Link annotations

Index	**0**	**1**	**2,0**	**2,1**	**2,n-1**
Content	link id	class id	1st token id	2nd token id	n-th token id

the aforementioned annotation format is represented in Tables 4.2.

4.3.5 Limitations

We are aware that the proposed dataset, even if it is proposing a new benchmark to tackle CTE, has room for improvement. As such, we list the limitations of the dataset:

1. There is a small quantity of data and tables compared to other datasets. Considering that adding more annotated data would be nothing but trivial, we believe this point could be addressed in two ways: i) as a starting pool of data to train generative models and getting new samples automatically labeled (e.g. using techniques similar to [112]); ii) using the CTE collection as a challenging benchmark to compare lightweight models, such as GNNs, along with state-of-the-art transformers (notably anger of huge amount of data).

2. The heuristics used for the the classes *caption* and *other* could affect the generalization of trained models, highly dependent on the paper format used in PubMed Central. On the other hand, we are enriching information about tables by recognizing captions, that contain valuable table descriptions and that otherwise would be discarded.

3. We still lack additional information such as author, keywords, and equations. We are going to add these additional labels in the near future, considering Grobid [56] in the annotation procedure, since it

is a machine learning library for extracting technical information from scientific publications, from PDF to XML/TEI structured documents.

4. The first attempts to define a baseline is reported in [49], in which the task of TE and DLA are treated end-to-end. This chapter aims at sharing the CTE dataset in a way that the scientific community can further propose baselines on this work.

While we acknowledge that CTE has some limitations, we believe that it represents a significant step towards a more comprehensive solution for TE in documents. In our previous work [49], we investigated different ways to achieve CTE through ablation studies, so as to analyze the impact of different components on the system's performances. In this chapter, instead, we define the F1-based metric ($F1_{CTE}$) for the updated dataset regarding CTE. As the combination of two metrics, namely Token F1 and Link F1, they can be used to evaluate the performance of the system.

4.4 Data Augmentation on Graphs

For DA, new graphs can be obtained by modifying their structure and the information associated with the nodes and edges. Since the embedding for each table is evaluated through a message passing algorithm that strongly relies on the table structure and content, removing elements of the graph and changing node features helps to generate more variability of examples for each class. This not only improves the generalization capability of the model, but can help to reduce class imbalance.

4.4.1 Method

Our method consists of two main steps: random removal of nodes and edges and inversion of rows and columns. Although we applied simple heuristics, the results reported on Tab2Know [79] show that they were effective, as well as easy to implement and reproduce.

Random removal of nodes and edges. In these operations, a random sample of nodes or arcs within the table is removed from the graph. By doing so, it is possible to generate a new graph similar to the initial one, but with different information.

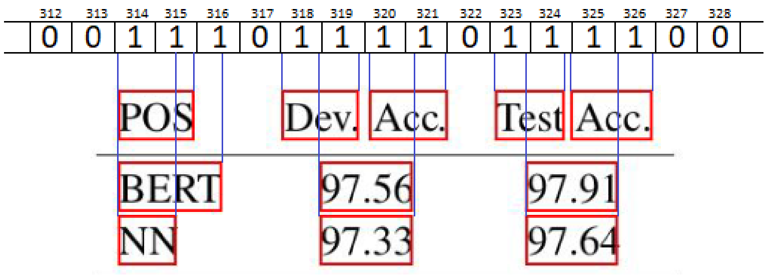


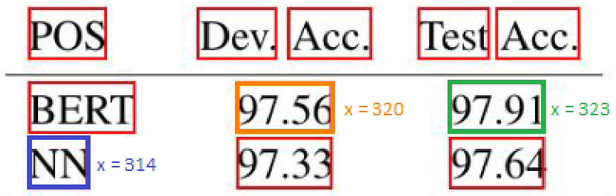
Figure 4.2: Recognition of columns. Group of 1s in the projected vector indicate different columns.

- nodes removal: a random subset of node indexes is removed. The size of the sample depends on the number of nodes in the graph, a random number between 1% and 20% of the total number of nodes.
- edges removal: a random subset of edge indexes is removed. The size of the sample depends on the number of edges in the graph, a random number between 1% and 20% of the total number of edges.

The amount of randomly removed nodes/edges is an arbitrary choice. We did not want to: (i) discard too much information and (ii) introduce any bias in the decision.

Inversion of rows and columns The row and column inversion technique is more complex, due to the fact that the internal structure of the tables is not known. Therefore, it is necessary to define an approach to approximate this structure. Once identified, rows or columns can be inverted, by means of swapping their node features.

- Column inversion: table column identification is made with a projection-profile based approach which defines a vector of size equal to the width of the table region. Each element of the vector is initialized to 0. Then, for each word, the coordinates x_1 and x_2 of the corresponding bounding box are extracted and projected, setting to 1 the vector values whose indices correspond to these coordinates. The obtained result is shown in figure 4.2: adjacent 0s should identify column boundaries, while adjacent 1s the coordinates of each column. Thus, two columns can be inverted by swapping their contents, that is, the features of the nodes whose center of the bounding box belongs to those columns. The



POS	Dev. Acc.	Test Acc.
BERT	97.56	97.91
NN	97.33	97.64

Figure 4.3: Recognition of rows. The blue bounding box is detected belonging to a new row since its x coordinate is lower than the previous green block.

limitation of this technique is visible whenever there is a space between words belonging to the same column.

- Row inversion: To reverse rows, it is necessary to compare the positions of "successive" bounding boxes. PyMuPDF reads and orders the content from left to right and from top to bottom. So, when a bounding box appears positioned ahead of the next one, it means that the latter is on a new row. In Fig. 4.3 the orange, green and blue bounding boxes are successive ones: the last one is on a new row since its x coordinate is lower than the green one. Once the structure of the rows has been identified, they can be reversed by swapping the features of the nodes belonging to them. The limitation of this technique is visible in the case of multi-row tables.

4.4.2 Preparing the data

The first step to apply a GNN for table classification is the conversion of tables in PDF papers into graphs as outlined in the previous chapter. The library is used to extract words and their bounding boxes; by using the positions of the tables in the annotations, only the words within them can be considered (Fig. 4.4). One graph for each table is built, where words correspond to nodes. Each node is connected to its nearest visible node when their bounding boxes intersect horizontally or vertically. Each graph, representing a table, is associated with the annotation corresponding to its type.

The Tab2Know dataset contains information regarding tables extracted from scientific papers in the Semantic Scholar Open Research Corpus. Tables are extracted using PDFFigures [30], a tool that finds figures, tables,

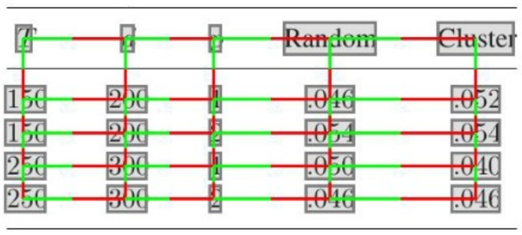


Figure 4.4: Words and bounding boxes extracted from one PDF paper using PyMuPDF. Nodes are connected through a visibility directed (from green to red sides) graph.

and captions within PDF documents, and Tabula[1], that outputs a CSV per each table reflecting its structure and content. After the conversion, each table is saved as an RDF triple addressable by an unique URI. Each CSV is then analyzed to recognize headers, type of table and columns type. The authors define an ontology of 27 different classes, 4 of which are defined as ”root” ones (Example, Input, Observation and Other): the others are given depending on the type of columns found inside each table (e.g. Recall is a subclass of Metric that is a subclass of Observation). Their training corpus is composed of 73k tables, labeled using Snorkel [119] and starting from a small pre-labeled set of tables obtained through human supervision using SPARQL queries. Human annotators then looked at 400 of them, checking their labeling correctness and, after resolving their conflicts when disagreeing, used this subset as the test set. To extract and group information on tables from Tab2Know, we built a conversion system to derive a JSON object for each available table. The information is the table numbering in the document, the page number where the table is located, the number of rows that make up the header, the document URL, the table class definition, and the caption text. We also added some information not represented in the RDF graph, such as the position of the table and the location of the caption (the latter information is obtained using PDFFigures and Tabula). Then we downloaded the PDFs of papers containing corresponding tables, accessed from the Semantic Scholar Open Research Corpus. From each paper, the pages containing the tables are extracted. Unfortunately it is not possible to use the whole Tab2Know dataset. For instance, some papers are no longer available or an

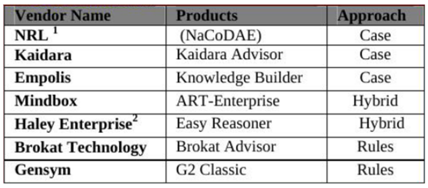
[1]https://github.com/tabulapdf/tabula

updated version does not match the annotations provided anymore. From the total, only the data whose annotations match are used, discarding the others. We obtained a subset containing 33,069 tables extracted from 11,800 scientific documents (45% of the original one). In addition, this dataset is very unbalanced (80% Observation, 10% Input, 7% Other, 3% Example) and it contains several missing or wrong annotations (55% of column classes have been labeled as 'others', across 22 different classes). For these reasons, we only use in this preliminary work the test set that was manually classified and corrected by humans. Specifically, this dataset contains 361 tables extracted from 253 scientific papers. The distribution of tables according to the class is as follows: 235 *Observation*, 43 *Input*, 13 *Example*, 29 *Other* (41 were 'unclassified', and we do not consider them during training). We retain 20% of this subset as training (randomly sampled keeping the same class occurrences) and, through the data augmentation techniques described before, we evaluated the generalization capabilities of the proposed model. The Tab2Know dataset can be used for performing table classification. However, the manually labeled subset is small and therefore we need to implement suitable Data Augmentation (DA) techniques. DA is widely used in machine learning in order to make models generalize better on unseen samples and unbalanced datasets.

Tables without annotation and those from which a graph cannot be built are discarded. At the end we obtain 320 graphs split into four classes: Observation (235), Input (43), Example (13), and Other (29). Examples of classes can be seen in Fig. 4.5. Each node in the graph corresponds to a feature vector. In addition to the geometric features of the nodes, such as position and size, textual content embeddings are added using spaCy. In particular, two spaCy models are used and compared: *en_core_web_lg* and *en_core_sci_lg*. The first one is the largest English vocabulary which associates each word with a numerical vector of 300 values; the other model, trained on a biomedical corpus, associates each word with a numerical vector of 200 values. The results obtained using the two models are compared in the experiments.

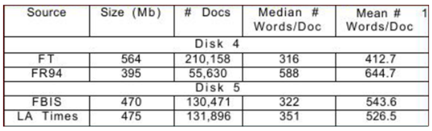
4.4.3 Experiments

The main experiments performed are summarized in Table 4.3 that compares results obtained by applying different Data Augmentation techniques and spaCy models `en_core_web_lg` and `en_core_sci_lg` with baseline results. In bold we highlight the most significant results of the F1 score for each



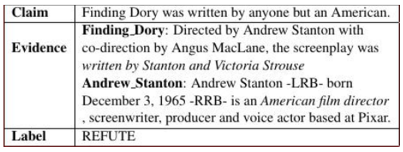
Vendor Name	Products	Approach
NRL [1]	(NaCoDAE)	Case
Kaidara	Kaidara Advisor	Case
Empolis	Knowledge Builder	Case
Mindbox	ART-Enterprise	Hybrid
Haley Enterprise[2]	Easy Reasoner	Hybrid
Brokat Technology	Brokat Advisor	Rules
Gensym	G2 Classic	Rules

Observation



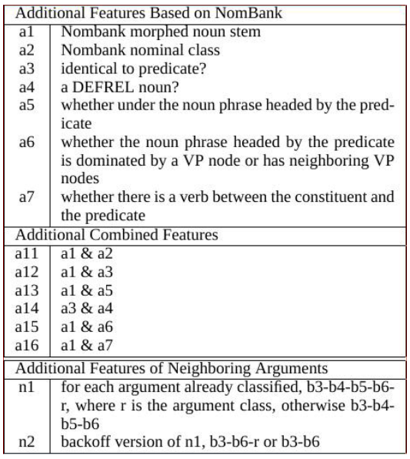
Source	Size (Mb)	# Docs	Median # Words/Doc	Mean # Words/Doc 1
		Disk 4		
FT	564	210,158	316	412.7
FR94	395	55,630	588	644.7
		Disk 5		
FBIS	470	130,471	322	543.6
LA Times	475	131,896	351	526.5

Input



Claim	Finding Dory was written by anyone but an American.
Evidence	**Finding_Dory**: Directed by Andrew Stanton with co-direction by Angus MacLane, the screenplay was *written by Stanton and Victoria Strouse* **Andrew_Stanton**: Andrew Stanton -LRB- born December 3, 1965 -RRB- is an *American film director* , screenwriter, producer and voice actor based at Pixar.
Label	REFUTE

Example



Additional Features Based on NomBank	
a1	Nombank morphed noun stem
a2	Nombank nominal class
a3	identical to predicate?
a4	a DEFREL noun?
a5	whether under the noun phrase headed by the predicate
a6	whether the noun phrase headed by the predicate is dominated by a VP node or has neighboring VP nodes
a7	whether there is a verb between the constituent and the predicate
Additional Combined Features	
a11	a1 & a2
a12	a1 & a3
a13	a1 & a5
a14	a3 & a4
a15	a1 & a6
a16	a1 & a7
Additional Features of Neighboring Arguments	
n1	for each argument already classified, b3-b4-b5-b6-r, where r is the argument class, otherwise b3-b4-b5-b6
n2	backoff version of n1, b3-b6-r or b3-b6

Other

Figure 4.5: Different types of tables with their classes in Tab2Know (taken from [79]).

technique applied. These values are also summarized in Table 4.4 to discuss the outcomes of the experiment. Table 4.4 summarizes the best F1 score values obtained considering some DA combinations. We can observe that the models appear to be rather inaccurate. This is mainly caused by the dataset itself, that is unbalanced toward the Observation class and small in size. It can also be seen that models using the `en_core_sci_lg` embedding show better results than those using `en_core_web_lg`, since the first one is a biomedical-based embedding that is most likely capable of appropriately characterizing and recognizing terms present in the tables extracted from scientific documents. In particular, models that exploit `en_core_web_lg` and do not use data augmentation techniques turn out to be less accurate and fail to recognize any table of class Other. In general, models that employ data augmentation result in higher F1 score values. Furthermore, observing Table 4.4, it can be seen that better values are obtained for the models in which data augmentation techniques are applied: particularly among these,

Table 4.3: Results without data augmentation (*No Aug.*); Data Augmentation with Row and Column inversion (*R/C*); Data Augmentation with Row and Column inversion and random removal of nodes and edges (*All*). *P*, *R* and *F*1 correspond to Precision, Recall and F1 score.

	No Aug.					
	train size: 63					
	web_lg			**sci_lg**		
Classes (#)	*P*	*R*	*F1*	*P*	*R*	*F1*
Observation (185)	0.85	0.84	0.84	0.87	0.92	0.89
Input (35)	0.37	0.49	0.42	0.47	0.54	0.51
Example (10)	0.42	0.5	0.45	0.67	0.2	0.31
Other (23)	0.00	0.00	0.00	0.43	0.26	0.32
All (253)	0.41	0.46	0.43	0.61	0.48	**0.51**

	R/C											
	train size: 200						train size: 400					
	web_lg			**sci_lg**			**web_lg**			**sci_lg**		
Classes (#)	*P*	*R*	*F1*	*P*	*R*	*F1*	*P*	*R*	*F1*	*P*	*R*	*F1*
Observation (185)	0.82	0.84	0.83	0.85	0.92	0.89	0.82	0.84	0.84	0.84	0.94	0.88
Input (35)	0.37	0.43	0.39	0.52	0.46	0.48	0.34	0.34	0.34	0.52	0.40	0.45
Example (10)	0.80	0.40	0.53	0.60	0.30	0.40	0.57	0.40	0.47	0.50	0.30	0.37
Other (23)	0.06	0.04	0.05	0.41	0.30	0.35	0.05	0.04	0.05	0.50	0.30	0.38
All (253)	0.51	0.43	0.45	0.60	0.50	**0.53**	0.45	0.41	0.42	0.59	0.48	0.52

	All											
	train size: 200						train size: 400					
	web_lg			**sci_lg**			**web_lg**			**sci_lg**		
Classes (#)	*P*	*R*	*F1*	*P*	*R*	*F1*	*P*	*R*	*F1*	*P*	*R*	*F1*
Observation (185)	0.81	0.83	0.82	0.85	0.94	0.89	0.81	0.82	0.81	0.84	0.93	0.88
Input (35)	0.32	0.37	0.34	0.48	0.40	0.44	0.33	0.40	0.36	0.52	0.43	0.47
Example (10)	0.80	0.40	0.53	0.67	0.40	0.50	0.60	0.30	0.40	0.50	0.30	0.37
Other (23)	0.06	0.04	0.05	0.57	0.35	0.43	0.05	0.04	0.05	0.36	0.22	0.27
All (253)	0.50	0.41	0.44	0.64	0.52	**0.56**	0.45	0.39	**0.41**	0.55	0.47	0.50

the one obtained by alternating the inversion of rows and columns with random removal of nodes and arcs is preferable.

4.4.4 Limitations

The proposed solution has some aspects that could be deepened or improved to further develop the work started. First, it might be useful to test the implemented Data Augmentation techniques on other datasets to analyze their potential and efficiency. In addition, other Data Augmentation techniques could be implemented, such as adding or removing rows and columns to a table, or improving the techniques already implemented. For example, the

Table 4.4: Summary of F1 scores for different data augmentation approaches. *No Aug.* indicates no Data Augmentation technique was applied, *R/C* indicates row and column inversion technique and *All* indicates row and column inversion technique and random removal of nodes and arcs.

	No Aug.	**R/C**		**All**	
train size	64	200	400	200	400
`en_core_web_lg`	0.43	**0.45**	0.44	**0.49**	0.41
`en_core_sci_lg`	0.52	**0.53**	0.52	**0.56**	0.51

row and column recognition techniques could be improved, especially in the case of multi-row tables.

4.5 Conclusions and Future Works

In this Chapter we presented a new dataset to tackle the task of Contextualized Table Extraction. Usually, TE pipelines involve several components to perform different tasks on tables, without considering other important information present in the document such as captions. Based on these limitations, the proposed collection of data aims at developing models capable of tackling more tasks at once, resulting in CTE. We are looking to extend the dataset by adding more information such as authors, keywords, and equations. In addition to providing a new dataset for contextualized TE, the CTE task can also serve as a basis for future research, such as: investigating the effectiveness of using GNNs versus transformer architectures or combining table structure information with external additional knowledge base for a DocVQA setting on scientific papers.

Furthermore we presented a new data augmentation technique on the table graph structure. The results achieved are promising, even if the method proposed is limited by the simple heuristics proposed. The use of Data Augmentation techniques made it possible to improve the results obtained by an increase in the F1-Score measure in the ablation studies presented. We conclude by noting how such Data Augmentation techniques applied directly to graphs could prove to be an interesting clue for the application of GNNs in the presence of resource-limited datasets, a very common situation in many application domains.

Part II

Document Objects as Page Graph Nodes

Chapter 5

A task agnostic document understanding framework based on GNNs

The application of Graph Neural Networks has become crucial in various Document Understanding tasks beyond Table Extraction since they can unravel important structural patterns, fundamental in Key Information Extraction processes. Previous works in the literature propose task-driven models and do not take into account the full power of graphs. In this chapter we present Doc2Graph, a task-agnostic document understanding framework based on a graph-based model, to solve different tasks given different types of documents. We evaluated our approach on two challenging datasets for Key Information Extraction in forms and invoices. This chapter is mainly based on this work [46].

5.1 Introduction

The current state-of-the-art practice in the document understanding community is to utilize the power of huge pre-trained vision-language models [4,154, 155] that learn whether the visual, textual and layout cues of the document are correlated. Despite achieving superior performance on most document understanding tasks, large-scale document pre-training comes with a high computational cost both in terms of memory and training time. We present a solution that does not rely on huge vision-language model pre-training modules, but rather recognizes the semantic text entities and their relationships from documents exploiting graphs. The solution experiments on two challenging benchmarks for forms [69] and invoices [53] with a very small amount of labeled training data. Inspired by some prior works [34, 121, 122], we introduce *Doc2Graph*, a novel task-agnostic framework to exploit graph-based representations for document understanding. The proposed model is validated in three different challenges, namely KIE in form understanding, invoice layout analysis and table detection. A graph representation module is proposed to organize the document objects. The graph nodes represent words or the semantic entities while edges the pairwise relationships between them. Finding the optimal set of edges to create the graph is anything but trivial: usually in literature heuristics are applied, e.g. using a visibility graph [121]. In this work, we do not make any assumption a priori on the connectivity: rather we attempt to build a fully connected graph representation over documents and let the network learn by itself what is relevant.

In summary, the primary contributions of this work can be summarized as follows:

- **Doc2Graph**, the first task-agnostic GNN-based document understanding framework, evaluated on two challenging benchmarks (form and invoice understanding) for three significant tasks, without any requirement of huge pre-training data;
- We propose a **general graph representation module for documents**, that do not rely on heuristics to build pairwise relationships between words or entities;
- **A novel GNN architectural pipeline** with node and edge aggregation functions suited for documents, that exploits the relative positioning of document objects through polar coordinates.

5.2 Related works

Current state-of-the-art approaches [4, 65, 113, 154, 155] on document understanding tasks have utilized the power of large pre-trained language models, relying on language more than the visual and geometrical information in a document and also end up using hundreds of millions of parameters in the process. Moreover, most of these models are trained with a huge transformer pipeline, which requires an immense amount of data during pre-training. In this regard, Davis et al. [33] and Sarkar et al. [127] proposed language-agnostic models. In [33] they focused on the entity relationship detection problem in forms [69] using a simple CNN as a text line detector and then detecting key-value relationship pairs using a heuristic based on each relationship candidate score generated from the model. Sarkar et al. [127] rather focused on extracting the form structure by reformulating the problem as a semantic segmentation (pixel labeling) task. They used a U-Net based architectural pipeline, predicting all levels of the document hierarchy in parallel, making it quite efficient. The FUDGE [34] framework was then developed for form understanding as an extension of [33] to greatly improve the state-of-the-art on both the semantic entity labeling and entity linking tasks by proposing relationship pairs using the same detection CNN as in [33]. Then a GCN was deployed with plugged visual features from the CNN so that semantic labels for the text entities were predicted jointly with the key-value relationship pairs, as they are quite related tasks. Inspired by this influential prior work [34], we aim to propose a task-agnostic GNN-based framework called *Doc2Graph* that adapts a similar joint prediction of both the tasks, semantic entity labeling and entity linking using a node classification and edge classification module respectively. Doc2Graph is established to tackle multiple challenges ranging from KIE for form understanding to layout analysis and table detection for invoice understanding, without needing any kind of huge data pre-training and being lightweight and efficient.

5.3 Method

In this section, we present the proposed approach. First, we describe the pre-processing step that converts document images into graphs. Then, we describe the GNN model designed to tackle different kinds of tasks.

5.3.1 Documents graph structure

A graph is a structure made of nodes and edges. A graph can be seen as a language model representing a document in terms of its segments (text units) and relationships. A preprocessing step is required. Depending on the task, different levels of granularity have to be considered for defining the constituent objects of a document. They can be single words or entities, that is, groups of words that share a certain property (e.g., the name of a company). In our work we try both as the starting point of the pipeline: we apply an OCR to recognize words, while a pre-trained object detection model for detecting entities. The chosen objects, once found, constitute the nodes of the graph.

At this point, nodes need to be connected through edges. Finding the optimal set of edges to create the graph is anything but trivial: usually in literature heuristics are applied, e.g. using a visibility graph [121]. These approaches: (i) do not generalise well on different layouts; (ii) strongly rely on the previous node detection processes, which are often prone to errors; (iii) generate noise in the connections, since bounding box of objects could cut out important relations or allow unwanted ones; (iv) exclude in advance sets of solutions, e.g. answers far from questions. To avoid those behaviours, we do not make any assumption a priori on the connectivity: we build a fully connected graph and we let the network learn by itself what relations are relevant.

5.3.2 Node and edge features

In order to learn, suitable features should be associated to nodes and edges of the graph. In documents, this information can be extracted from sources of different modalities, such as visual, language and layout ones. Different methods can be applied to encode a node (either word or entity) to enrich its representation. In our pipeline, with the aim to possibly keep it lightweight, we include:

- a language model to encode the text. We use the spaCy large English model to get word vector representations of words and entities;

- a visual encoder to represent style and formatting. We pretrain a U-Net [123] on FUNSD for entities segmentation. Since U-Net uses feature maps at different encoder's layers to segment the images, we decide to use all these information as visual features. Moreover, it is important to highlight that, for each features map, we used a RoI Alignment layer to extract the features relative to each entities bounding box;

- the absolute normalized positions of objects inside a document; layout

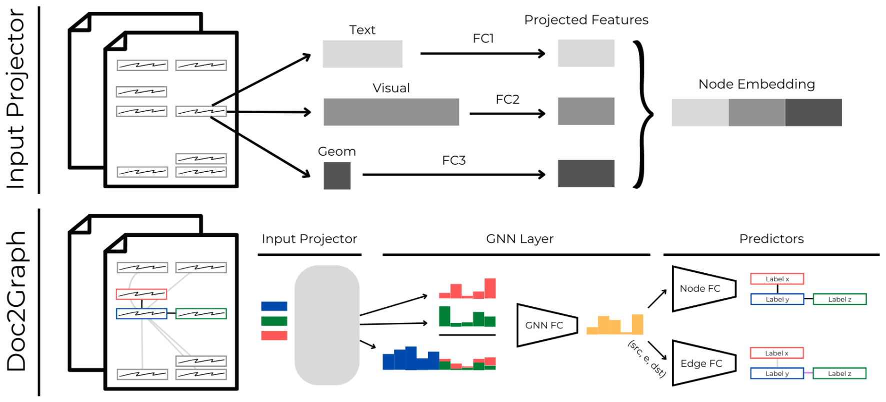


Figure 5.1: Our proposed Doc2Graph framework. For visualisation purposes, the architecture shows the perspective of one node (the blue one in Doc2Graph).

and structure are meaningful features to include in industrial documents, e.g. for key-value associations.

As for the edges, to the best of our knowledge, we propose two new sets of features to help both the node and the edge classification tasks:

- a normalized Euclidean distance between nodes, by means of the minimum distance between bounding boxes. Since we are using a fully connected graph this is crucial for the aggregation node function in use to keep locality property during the message passing algorithm;
- relative positioning of nodes using polar coordinates. Each source node is considered to be in the center of a Cartesian plane and all its neighbors are encoded by means of distance and angle. We discretize the space into bins (one-hot encoded), whose number can be chosen, instead of using normalized angles: a continuous representation of the angle is challenging because, for instance, two points at the same distance with angles 360° and 0° would be encoded differently.

5.3.3 Architecture

Each node feature vector passes through our proposed architecture (Fig. 5.1, visualization of GNN layer inspired by "A Gentle Introduction to GNNs"):

the connectivity defines the neighborhood for the message passing, while the weight learnable matrices are shared across all nodes and edges, respectively. We make use of four different components:

- Input Projector: this module applies as many fully connected (FC) layers as there are different modalities in use, to project each of their representations into inner spaces of the same dimension; e.g., we found it to be not very informative combine low dimensional geometrical features with high dimensional visual ones, as they are;

- GNN Layer: we make use of a slightly different version of GraphSAGE [58]. Using a fully connected graph, we redefine the aggregation strategy (eq. 5.2);

- Node Predictor: this is a FC layer, that maps the representation of each node into the number of target classes;

- Edge Predictor: this is a two-FC layer, that assigns a label to each edge. To do so, we propose a novel aggregation on edges (eq. 5.3).

GNN Layer

Our version of GraphSAGE slightly differs in the neighborhood aggregation. At layer l given a node i, h_i its inner representation and $N(i)$ its set of neighbors, the aggregation is defined as:

$$h_{N(i)}^{l+1} = aggregate(\{h_j^l, \forall j \in N(i)\}) \tag{5.1}$$

where *aggregate* can be any permutation invariant operation, e.g. sum or mean. Usually, in other domains, the graph structure is naturally given by the data itself but, as already stated, in documents this can be challenging (sec. 5.3.1). Then, given a document, we redefine equation 5.1 as:

$$h_{N(i)}^{l+1} = \frac{c}{|\Upsilon(i)|} \sum_{j \in \Upsilon(i)} h_j^l \tag{5.2}$$

where $\Upsilon(i) = \{j \in N(i) : |i-j| < threshold\}$, $|i-j|$ is the Euclidean distance of nodes i and j saved (normalized between 0 and 1) on their connecting edge, and c is a constant scale factor.

Edge Predictor

We consider each edge as a triplet (src, e, dst): e is the edge connecting the source (src) and destination (dst) node. The edge representation h_e to feed into the two-FC classifier is defined as:

$$h_e = h_{src} \parallel h_{dst} \parallel cls_{src} \parallel cls_{dst} \parallel e_{polar} \tag{5.3}$$

where h_{src} and h_{dst} are the node embeddings output of the last GNN layer, cls_{scr} and cls_{dst} are the softmax of the output logits of the previous node predictor layer, e_{polar} are the polar coordinates described in sec 5.3.2 and $\parallel$ is the concatenation operator. These choices have been made because: (i) relative positioning on edges is stronger compared to absolute positioning on nodes: the local property introduced by means of polar coordinates can be extended to different data, e.g. documents of different sizes or orientations; (ii) if the considered task comprise also the classification of nodes, their classes may help in the classification of edges, e.g. in forms it should not possible to find an answer connected to another answer.

Given the task, graphs can be either undirected or directed: both are represented with two or one directed edge between nodes, respectively. In the first case, the order does not matter and so the above formula can be redefined as:

$$h_e = (h_{src} + h_{dst}) \parallel cls_{src} \parallel cls_{dst} \parallel e_{polar} \tag{5.4}$$

5.4 Results

In this chapter we present experiments of our method on two different datasets, FUNSD and RVL-CDIP invoices, to tackle three tasks: entity linking, layout analysis and table detection. We also discuss results compared to other methods.

5.4.1 Proposed model

We performed ablation studies on our proposed model for entity linking on FUNSD without contribution and classification of nodes (Fig. 5.1), since we found it to be the most challenging task. In Tab. 5.1 we report different combinations of features and hyperparameters. Geometrical and textual features make the largest contribution, while visual features bring almost three points more to the Key-Value F1 score by an important increase in terms of

Table 5.1: Ablation studies of Doc2Graph model. EP Inner dim and IP FC dim show edge predictor layer input dimension and the input projector fully connected layers output dimension, respectively. AUC-PR refers to the key-value edge class. The # Params refers to Doc2Graph trainable parameters solely.

Features					*F_1 per classes* (↑)			
Geometric	**Text**	**Visual**	**EP Inner dim**	**IP FC dim**	**None**	**Key-Value**	**AUC-PR** (↑)	**# Params** $\times 10^6$ (↓)
✓	✗	✗	20	100	0.9587	0.1507	0.6301	0.025
✗	✓	✗	20	100	0.9893	0.1981	0.5605	0.054
✓	✓	✗	20	100	0.9941	0.4305	0.7002	0.120
✓	✓	✗	300	300	0.9961	0.5606	0.7733	1.18
✓	✓	✓	300	300	**0.9964**	**0.5895**	**0.7903**	2.68

network parameters (2.3 times more). Textual and geometrical features remain crucial for the task at hand, and their combination increase by a large amount both of their scores when used in isolation. This may be due to two facts: (i) our U-Net has not been included during the GNN training time (as done in [34]), unable to adjust the representation for spotting key-value relationship pairs; (ii) the segmentation task used to train the backbone do not yield useful features for that goal (as shown in Tab. 5.1). The hyper-parameters shown in the table refer to the edge predictor (EP) inner layer input dimension and the input projector fully connected (IP FC) layers (per each modality) output dimension, respectively. A larger EP is much more informative for the classification of links into 'none' (cut edges, meaning no relationship) or 'key-value', while more dimensions for the projected modalities helped the model to better learn the importance of their contributions. These changes bring an improvement of 13 points on the key-value F1 scores, between the third and fourth line of the table where we keep the features fixed. We do not report the score relative to others network settings since their changes only brought a decrease in overall metrics. We use a learning rate of 10^{-3} and a weight decay of 10^{-4}, with a dropout of 0.2 over the last FC layer. The threshold over neighbor nodes and their contribution scale factor (sec. 5.3.3) are fixed to 0.9 and 0.1, respectively. The bins to discretize the space for angles (sec. 5.3.3) are 8. We apply one GNN layer before the node and edge predictors.

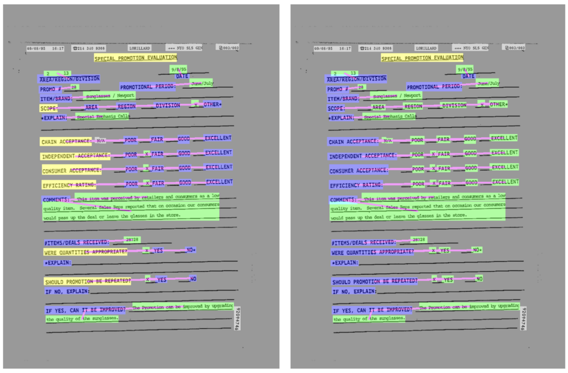
5.4.2 FUNSD

Dataset

The dataset [69] comprises 199 real, fully annotated, scanned forms. The documents are selected as a subset of the larger RVL-CDIP [60] dataset, a collection of 400,000 grayscale images of various documents. The authors define the Form Understanding (FoUn) challenge into three different tasks: word grouping, semantic entity labeling and entity linking. A recent work [141] found some inconsistency in the original labeling, which impeded its applicability to the key-value extraction problem. In this work, we are using the revised version of FUNSD.

Entity Detection

Our focus is on the GNN performances but, for comparison reasons, we used a YOLOv5 small [71] to detect entities (pretrained on COCO [90]). In [69] the word grouping task is evaluated using the ARI metric: since we are not using words, we evaluated the entity detection with F1 score using two different IoU thresholds (Tab. 5.2). For the semantic entity labeling and entity linking tasks we use IoU $>$ 0.50 as done in [34]: we did not perform any optimization on the detector model, which introduces a high drop rate



09/08/95 16:17 ☎214 340 9308 LORILLARD → NYO SLS GEN 003/003

SPECIAL PROMOTION EVALUATION

2 13 AREA/REGION/DIVISION 9/8/95 DATE

PROMO # 28 PROMOTIONAL PERIOD: June/July

ITEM/BRAND: Sunglasses / Newport

SCOPE: AREA REGION DIVISION x OTHER*

*EXPLAIN: Special Emphasis Calls

CHAIN ACCEPTANCE: N/A POOR FAIR GOOD EXCELLENT

INDEPENDENT ACCEPTANCE: POOR x FAIR GOOD EXCELLENT

CONSUMER ACCEPTANCE: POOR x FAIR GOOD EXCELLENT

EFFICIENCY RATING: POOR x FAIR GOOD EXCELLENT

COMMENTS: This item was perceived by retailers and consumers as a low quality item. Several Sales Reps reported that on occasion our consumers would pass up the deal or leave the glasses in the store.

#ITEMS/DEALS RECEIVED: 28728

WERE QUANTITIES APPROPRIATE? x YES NO*

*EXPLAIN:

SHOULD PROMOTION BE REPEATED? x YES NO

IF NO, EXPLAIN:

IF YES, CAN IT BE IMPROVED? The Promotion can be improved by upgrading the quality of the sunglasses.

92094746

Figure 5.2: Image taken from [141]: the document on the right is the revised version of the document on the left, where some answers (green) are mislabeled as question (blue), and some questions (blue) are mislabeled as headers (yellow)

Table 5.2: Entity detection results. YOLOv5 [71]-small performance on the entity detection task.

	Metrics (↑)			*% Drop Rate* (↓)	
IoU	**Precision**	**Recall**	**F_1**	**Entity**	**Link**
0.25	0.8728	0.8712	0.8720	12.72	16.63
0.50	0.8132	0.8109	0.8121	18.67	25.93

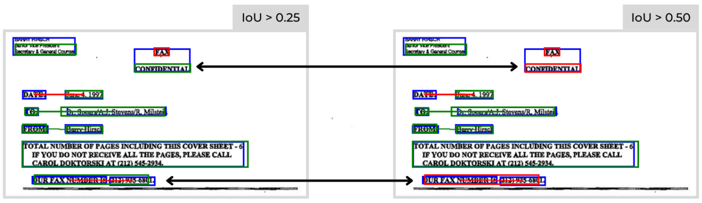


Figure 5.3: Blue boxes are FUNSD entities ground truth, green boxes are the correct detected one (with IoU $>$ 0.25/0.50), while red boxes are the false positive ones.

for both entities and links. We create the graphs on top of YOLO detections, linking the ground truth accordingly (Fig. 5.3): false positive entities (red boxes) are labeled as class 'other', while false negative entities cause some key-value pairs to be lost (red links). The new connections created as a consequence of wrong detections are considered false positives and labeled as 'none'.

Numerical results

We trained our architecture (sec. 5.3.3) with a 10-fold cross validation. Since we found high variance in the results, we report both mean and variance over the 10 best models chosen over their respective validation sets. The objective function in use (L) is based on both node (L_n) and edge (L_e) classification tasks: $L = L_n + L_e$. In Tab. 5.3 we report the performance of our model Doc2Graph compared to other language models [65, 155] and graph-based techniques [27, 34]. The number of parameters # Params refer to the trainable Doc2Graph pipeline (that includes the U-Net and YOLO backbones); for the spaCy word-embedding details, refer to their documentation. Using

YOLO our network outperforms [27] for semantic entity labeling and meets their model on entity linking, using just 13.5 parameters. We could not do better than FUDGE, which still outperforms our scores. Their backbone is trained for both tasks along with the GCN (GCN that adds just minor improvements). The gap, especially on entity linking, is mainly due to the low contributions given by our visual features (Tab. 5.1) and the detector in use (Tab. 5.3). We also report the results of our model initialized with ground truth (GT) entities, to show how it would perform in the best case scenario. Entity linking remains a harder task compared to semantic entity labeling and only complex language models seem to be able to solve it. Moreover, for the sake of completeness, we highlight that, with good entity representations, our model outperforms all the considered architectures for the Semantic Entity Labeling task. Finally, we want to further stress that the main contribution of a graph-based method is to yield a simpler but more lightweight solution.

Qualitative results

The order matters for detecting key-value relationship, since the direction of a link induce a property for the destination entity that enriches its meaning. Differently from FUDGE [34] we do make use of directed edges, which led to a better understanding of the document having interpretable results. In Fig. 5.5 we show our qualitative results using Doc2Graph on groundtruth: green and red dots mean source and destination nodes, respectively. As

Table 5.3: Results on FUNSD. The results have been shown for both semantic entity labeling and entity linking tasks with their corresponding metrics.

		F_1 (↑)		
Method	**GNN**	**Semantic Entity Labeling**	**Entity Linking**	**# Params** $\times 10^6$ (↓)
BROS [65]	✗	**0.8121**	**0.6696**	138
LayoutLM [65, 155]	✗	0.7895	0.4281	343
FUNSD [69]	✓	0.5700	0.0400	-
Carbonell et al. [27]	✓	0.6400	0.3900	201
FUDGE w/o GCN [34]	✗	0.6507	0.5241	12
FUDGE [34]	✓	0.6652	0.5662	17
Doc2Graph + YOLO	✓	0.6581 ± 0.006	0.3882 ± 0.028	13.5
Doc2Graph + GT	✓	$\underline{0.8225}$ ± 0.005	0.5336 ± 0.036	**6.2**

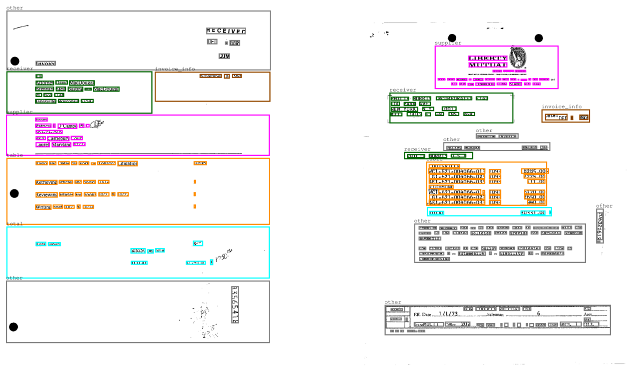


Figure 5.4: RVL-CDIP Invoices benchmark in [121]. There are 6 regions: supplier (pink), invoice_info (brown), receiver (green), table (orange), total (light blue), other (gray).

shown in the different example cases, Fig. 5.5a and 5.5b resemble a simple structured form layout with directed one-to-one key-value association pairs and Doc2Graph manages to extract them. On the contrary, where the layout appears to be more complex as in Fig. 5.5d, Doc2Graph fails to generalize the concept of one-to-many key-value relationship pairs. This may be due to the small number of trainable samples we had in our training data and the fact that header-cells usually present different positioning and semantic meaning. In the future we will integrate a table structure recognition path into our pipeline, hoping to improve the extraction of all kinds of key-value relationships in such more complex layout scenarios.

5.4.3 RVL-CDIP Invoices

Dataset

In the work of Riba et al. [121] another subset of RVL-CDIP has been released. The authors selected 518 documents from the invoices classes, annotating 6 different regions (two examples of annotations are shown in Fig. 5.4). The task that can be performed are layout analysis, in terms of node classification, and table detection, in terms of bounding box (IoU $>$ 50).

Numerical results

As done previously, we perform a k-fold cross validation keeping, for each fold, the same amount of test (104), val (52) and training documents (362). This time we applied an OCR to build the graph. There are two tasks: layout analysis, in terms of accuracy, and table detection, using F1 score and $IoU > 0.50$ for table regions. Our model outperforms [121] in both tasks, as shown in tables 5.4 and 5.5. In particular, for table detection, we extracted the subgraph induced by the edge classified as 'table' (two nodes are linked if they are in the same table) to extract the target region. Riba et al. [121] formulated the problem as a binary classification: we report, for brevity, in Tab. 5.5 the threshold on confidence score they use to cut out edges, that in our multi-class setting ('none' or 'table') is implicitly set to 0.50 by the softmax.

Qualitative results

In Fig. 5.6 we show the qualitative results. The two documents are duplicated to better visualize the two tasks. For layout analysis, the greater bounding boxes colors indicate the true label that the word inside should have (the colors reflects classes as shown in Fig. 5.4). For the table detection we use a simple heuristic: we take the enclosing rectangle (green) of the nodes connected by 'table' edges, then we evaluate the IoU with target regions (orange). This heuristic is effective but simple and so error-prone: if a false positive is found outside table regions this could lead to a poor detection result, e.g. a bounding box including also 'sender item' entity or 'receiver item' entity. In addition, as inferred from Figs. 5.6a and 5.6b, 'total' regions could be taken out. In the future, we will refine this behaviour

Table 5.4: Layout analysis results on RVL-CDIP Invoices. Layout analysis accuracy scores depicted in terms of node classification task.

	Accuracy (↑)	
Method	**Max**	**Mean**
Riba et al. [121]	62.30	-
Doc2Graph + OCR	**69.80**	**67.80** ± 1.1

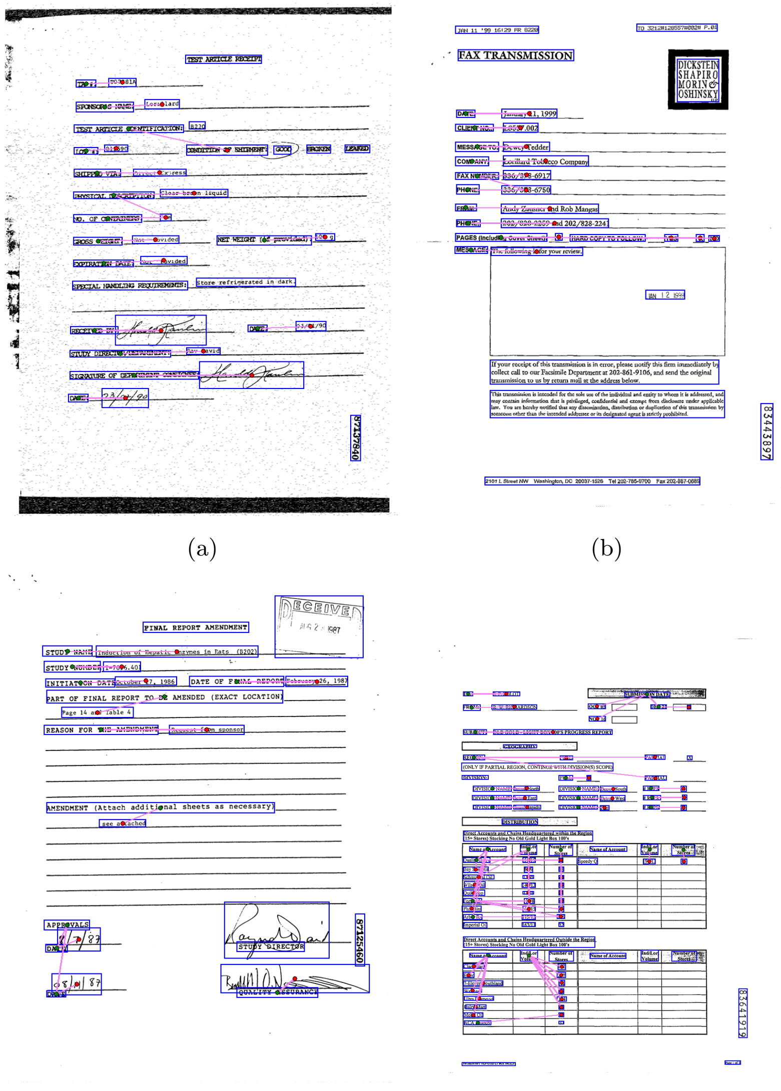
Table 5.5: Table Detection in terms of F1 score. A table is considered correctly detected if its IoU is greater than 0.50. Threshold values refers to the scores an edges has to have in order to do not be cut: in our case is set to 0.50 by the softmax in use.

		Metrics (↑)		
Method	**Threshold**	**Precision**	**Recall**	$\mathbf{F_1}$
Riba et al. [121]	0.1	0.2520	**0.3960**	0.3080
Riba et al. [121]	0.5	0.1520	0.3650	0.2150
Doc2Graph + OCR	0.5	**0.3786** ± 0.07	0.3723 ± 0.07	**0.3754** ± 0.07

by both boosting the node classification task and including 'total' as a table region for the training of edges.

5.5 Conclusion and Future Works

In this chapter, we have presented a task-agnostic document understanding framework based on a GNN. We proposed a general representation of documents as graphs, exploiting full connectivity between document objects and letting the network automatically learn meaningful pairwise relationships. Node and edge aggregation functions are defined by taking into account the relative positioning of document objects. We evaluated our model on two challenging benchmarks for three different tasks: entity linking on forms, layout analysis on invoices and table detection. Our preliminary results show that our model can achieve promising results, keeping the network dimensionality considerably small. For future works, we will extend our framework to other documents and tasks, to deeper investigate the generalization property of the GNN. We would like also to explore more extensively the contribution of different source features and how to combine them in more meaningful and learnable ways.



(a) (b) (c) (d)

Figure 5.5: Entity Linking on FUNSD. Differently from other apporaces, we make use of directed edges improving explainability: green and red dots mean source and destination nodes, respectively.

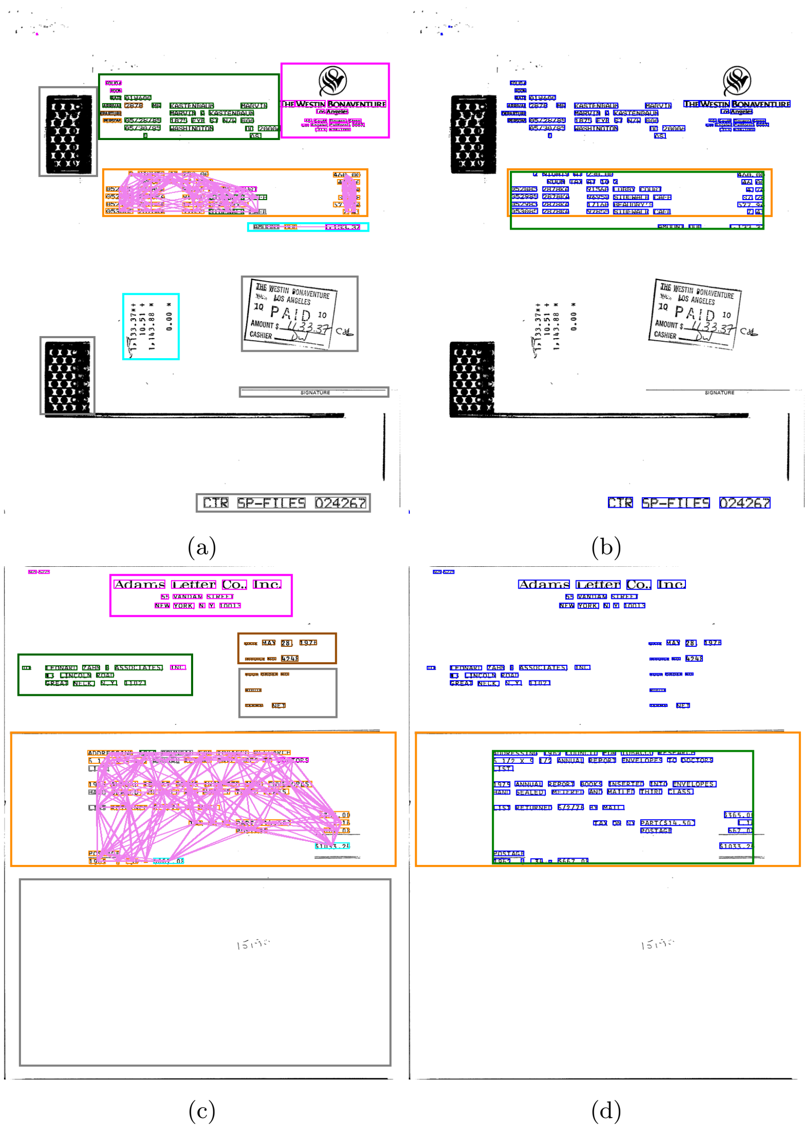

THE WESTIN BONAVENTURE
Los Angeles
THE WESTIN BONAVENTURE LOS ANGELES
PAID
AMOUNT $ 433.37
CASHIER
SIGNATURE
CTR SP-FILES 024267
(a)
(b)
Adams Letter Co. Inc.
(c)
(d)

Figure 5.6: Layout Analysis on RVLCDIP Invoices. Inference over two documents from RVL-CDIP Invoices, showing both: **1. Layout Analysis** (a,c) for six different classes - supplier (pink), invoice info (brown), receiver (green), table (orange), total (light blue), other (gray); and **2. Table Detection** (b,d) - in the images our extracted table (green) is shown in contrast with the label table (orange).

Chapter 6

Automatic generation of scientific papers

As already introduced in Chapter 4, datasets of scientific articles have played a key role in the development of modern deep learning systems for Document Understanding. Usually, these are either manually or automatically labelled, introducing a tradeoff between the quantity and quality of the annotated data, as well as the costs required to create them. A third solution is to consider synthetic document generation. The latter is not only of considerable help in application scenarios where there is little data available to perform model training but also allows for the creation of a potentially infinite number of diverse examples along with their annotations. However, as effective as it is, document generation is still a task far from being solved. In this scenario we propose a customizable pipeline to generate high-quality pages of scientific papers, demonstrating their effectiveness on a custom benchmark for Document Layout Analysis. This chapter is mainly based on these works [47, 112].

6.1 Introduction

Recently an increasing demand for larger datasets for deep learning methods has started to open new challenges on how to annotate such collections.

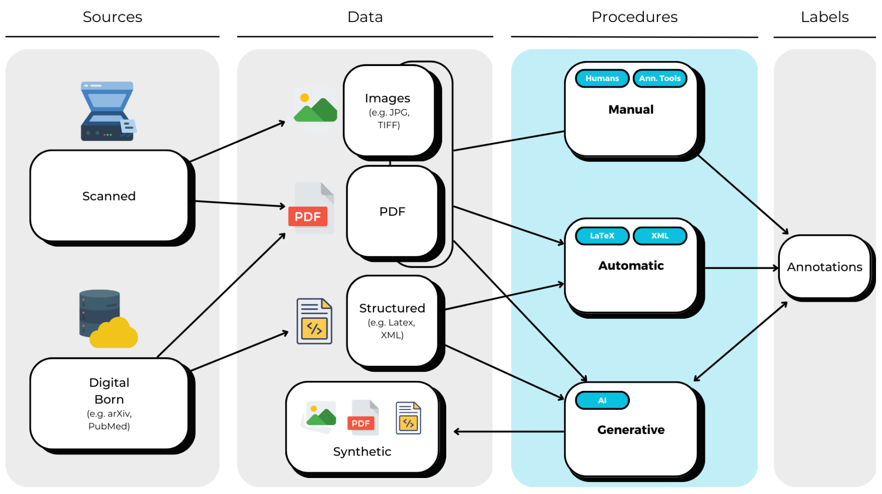


Figure 6.1: Different pipelines to create labelled data depend on the combination of three main factors: the sources from where to crawl the data (left), which data are available (center), and which annotation procedures to involve (right). Synthetic documents do not require to following any specific annotation procedure since labels are given within the generated data.

Throughout the years, procedures for annotating documents have been proposed trying to maximise two different measures, usually inversely proportional: quality and amount of annotations, taking into consideration also time, costs and data variability. As a matter of fact, available datasets, such as PubLayNet and DocBank [85, 164], exhibit limited variability in content and layout: their creators collected papers from arXiv or PubMed Central, mostly including double column layouts. As a consequence, when trained models are applied to different layouts (e.g. single-page proceedings) the results can be worse than expected. On the opposite, another recent dataset called DocLayNet [110] came out, proposing a new collection of manual annotated documents varied in layout, content and document domain. The dataset consists of documents belonging to six distinct domains (Financial, Scientific, Patents, Manuals, Laws, and Tenders), counts 80,863 PDF pages with annotated bounding boxes belonging to 11 different classes. To ensure homogeneity among different annotators, a 100 pages annotation guideline has been written and administered to 32 selected experts. The whole annotation process lasted 6 months.

In general, we could summarize the dataset creation procedures as *manual* or *automatic*. The first ones generally do not scale well with the dataset size. In addition, guidelines need to be defined so that different annotators follow the same rules to produce coherent and homogeneous annotations among data. This approach has been the foundation of most datasets across different machine learning fields but today it is often not preferred due to its high cost both in terms of money and time. However, researchers continue to propose manually annotated data since the human supervision is capable of yielding qualitative annotations and gather important information that are not easily available through automatic annotations. On the contrary, automatic approaches often rely on rule-based scripts to annotate large collections of data, such as PDF scientific articles shared as PDF files. Since it is not trivial to access information such as text, tables, and titles and to detect them easily in the page layout starting from a PDF, additional structured information, e.g. contained in LaTeX and XML, need to be used for the automatic annotations of this data, which is the primary reason that limits the application of these techniques on different documents.

An alternative solution is the generation of synthetic documents together with their annotations. Using generative methods it is possible to automatically create annotations for any arbitrary amount of data, with layout and content variations. That is why we designed a synthetic data generator that can be easily customized to generate documents of a desired specific layout, to augment the training data and potentially boost the performances of a pre-trained object detector model.

A summary of the three possible ways to collect annotated collections of documents, as described so far, is depicted in Figure 6.1. Our contribution in this chapter belong to the third block of the *procedures* column, and the three main aspects of our work are the following:

- We propose a **semi-automatic annotation approach to obtain high-quality annotations** for a small set of scientific papers; annotation errors are manually fixed and 11 different regions (title, authors, abstract, keywords, subtitle, text, image, table, caption, formula, reference) are labeled; these annotations are used for data augmentation;

- We propose a **data generation pipeline** that starts from the annotated pages to train a generative model (based on LayoutTransformer [57]) and produce large collections of pages with variable layout

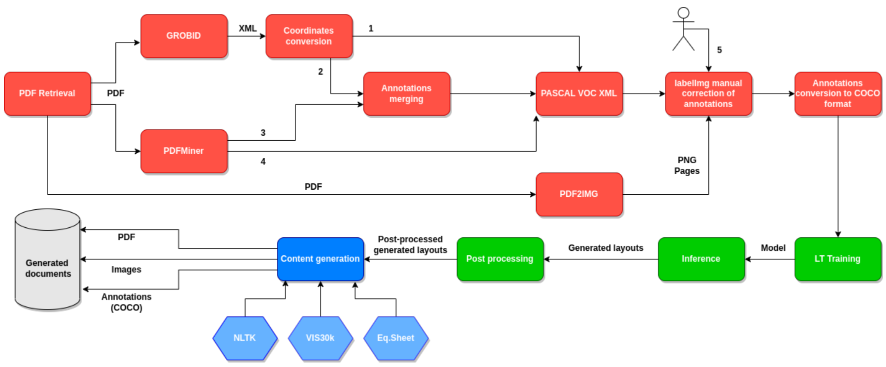


Figure 6.2: The proposed pipeline: first, data are collected, annotated in a semi-automatic fashion, and finally manually corrected (red boxes); then they are used to train a LayoutTransformer generator model that in turn generates synthesized layouts (green boxes) that are filled with content (blue box). We obtain a dataset of documents belonging to the domain of the initial data. Labels in the diagram are as follows: 1: Formulas, titles, tables; 2: Abstract, authors, keywords; 3: Text; 4: Figures; LT: LayoutTransformer.

whose regions are populated with synthetically generated content (e.g. text, images, tables). These pages can then be used to train DLA engines;

The overall approach is layout-agnostic. Our experiments show that it can work for both double and single-column scientific papers and we demonstrate that it is possible to improve the results of DLA by using a small semi-automatically annotated dataset.

6.2 Related work

The generation of synthetic data for boosting the recognition performance of trainable models has been extensively used in the last years and research on DLA is not an exception. Some methods require the user to carefully design rules that describe the distribution of regions in the page layout. For instance, in [11] a generator of semi-structured documents is proposed. This tool generates samples of administrative documents requiring the user to design by hand the general structure (defines the different types of information needed). The tool then modifies the position of items in the page by taking into account suitable random variables. Structured historical documents are generated, starting from a hand-designed general organization of the document in [26]. In [118] a generative process that treats every physical component of a document as a random variable and models their intrinsic dependencies using a Bayesian Network graph is proposed. The Bayesian Network defines the synthetic document generation process. The primitive units, style attributes, and layout elements are all treated as random variables and represented as nodes in a graph. Several sub-networks are designed to model different parts of a scientific paper (e.g. document, section, table, and figure). User-defined layout and font styles are considered as starting information in Document Domain Randomization (DDR [91]) to render simulated pages of scientific papers by modeling randomized textual and non-textual contents of interest that are downloaded from online repositories.

More recently, several generative models for document layout generation have been proposed. In [16], the authors try to learn layouts from a set of document images, leveraging a GAN (Generative Adversarial Network) architecture. In [104] an architecture based on VAE (Variational Auto-Encoder) and RNN (Recurrent Neural Network) is described. It can convert training page layouts into an embedded compact representation, whose space is represented by a Gaussian distribution. New document layouts can be created by sampling novel values from the distribution. A GAN-based method to generate a document layout given images, keywords and category of the document is described in [163]. One problem with this approach is that a large amount of training data is needed. Also transformers have been used to generate layouts. In [6] the authors model the distribution of objects in example layouts and a self-attention mechanism is used to detect high-level object relationships. LayoutTransformer, proposed in [57] is a framework that leverages self-attention to learn contextual relationships between layout elements and generates new layouts in the given domain. The authors exploit it to generate scientific papers, but only rely on PubLayNet for training.

Motivated by the transformers' achievements for document generation, we make use of the LayoutTransformer generative model [57] to perform data augmentation by first creating synthetic layouts starting from a small number of annotated pages and then populating the regions with techniques inspired by DDR [91]. In addition, in Section 6.4.1 we explain how the graph structure of the document layout was taken into account to create new variations that kept logical sense.

6.3 Semi-automatic annotation

The pipeline of the proposed approach is summarized in Fig. 6.2. Given a few papers from the target publication (journal or conference proceedings) some preliminary annotations are obtained by integrating the output of two PDF parsing tools (Sec. 6.3.1). These annotations are then refined by users (Sec. 6.3.2) and clean data are used to train the generative model as described in Section 6.4.

6.3.1 Automatic annotation

Since PDF files faithfully represent the layout and typographic information of documents, but not their semantics, we parse the PDF papers with two tools: the GROBID [56] machine learning library, focused on extracting bibliographic information from scientific papers, and the PDFMiner library [106] that extracts layout elements in the pages. The GROBID output is a TEI XML file [31] that represents the metadata of the paper. In particular, the TEI tags describe the class type and the bounding box for titles, sub-titles, tables, and formulas. However, in the case of authors, abstract, and keywords the position information is missing. The PDF files are then processed by PDFMiner which extracts all the text blocks and images with their bounding box coordinates. The information extracted with the two tools is then merged and nine object classes are identified as follows.

- *Titles, subtitles, tables* and *formulas* (arrow 1 in Fig. 6.2) are identified in the GROBID output using beautifulsoup4 [10] to parse the TEI XML files.

- GROBID also extracts *abstract, authors*, and *keywords* (arrow 2 in Fig. 6.2), in this case, but there is no bounding box information in the TEI tags. Since all the text-boxes are extracted by PDFMiner it is possible to find position information by using text matching between GROBID and PDFMiner results. The text matching is computed with *SequenceMatcher* (from Python difflib [39]) considering two strings as similar if the score is greater than a threshold set to 0.7 (the threshold is a balance between too much noise with low values and too many false negatives with high ones).

- The *text* regions extracted by PDFMiner (arrow 3 in Fig. 6.2) that are not matched with the output of GROBID are first filtered with some heuristics to exclude false positives and then saved as *text*. For instance, text regions smaller than the font size are removed. The other objects extracted by PDFMiner are *figures* (arrow 4 in Fig. 6.2).

In the subsequent manual correction, the labels of text regions corresponding to *captions* and *references* are fixed obtaining regions for the 11 classes that are used by the generative model. Annotation examples for double and single columns are shown in Fig. 6.3. After performing the previous steps the annotation information is converted to a format suitable for manual correction. For each processed paper we create page images (using the pdf2image python library [105]) and convert the annotations to XML files in PASCAL VOC format [41].

6.3.2 Annotation correction

Not surprisingly, there are some errors in the automatic annotation. We therefore manually check the annotations using labelImg [80] (arrow 5 in Fig. 6.2), an interactive tool that reads PASCAL VOC annotations, supports the user to fix errors, and saves the updated annotations.

Corrections performed include the addition of missing objects (e.g. tables, images, captions, and page numbers) and the separation of overlapping objects that we want to avoid. Since the proposed content generation (Section 6.4.3) cannot easily handle overlapping regions, we deal with the overlaps by removing the shared area and reducing the overlapping regions accordingly. Given simple annotation guidelines, for each page, the annotators needed a time ranging from 30 seconds to two minutes depending on the complexity of the page (i.e. number of different regions in the page).

By correcting the annotations we expect to help the generative model to produce better training samples: considering that we started with very few training examples, it is crucial to work with high-quality annotations to produce plausible results. We will show how this step improves also the final results by a large margin, in the experiments presented in Section 6.5.3. The last step is to prepare the data to use them to train the generative model (LayoutTransformer) that accepts data in COCO format [90].

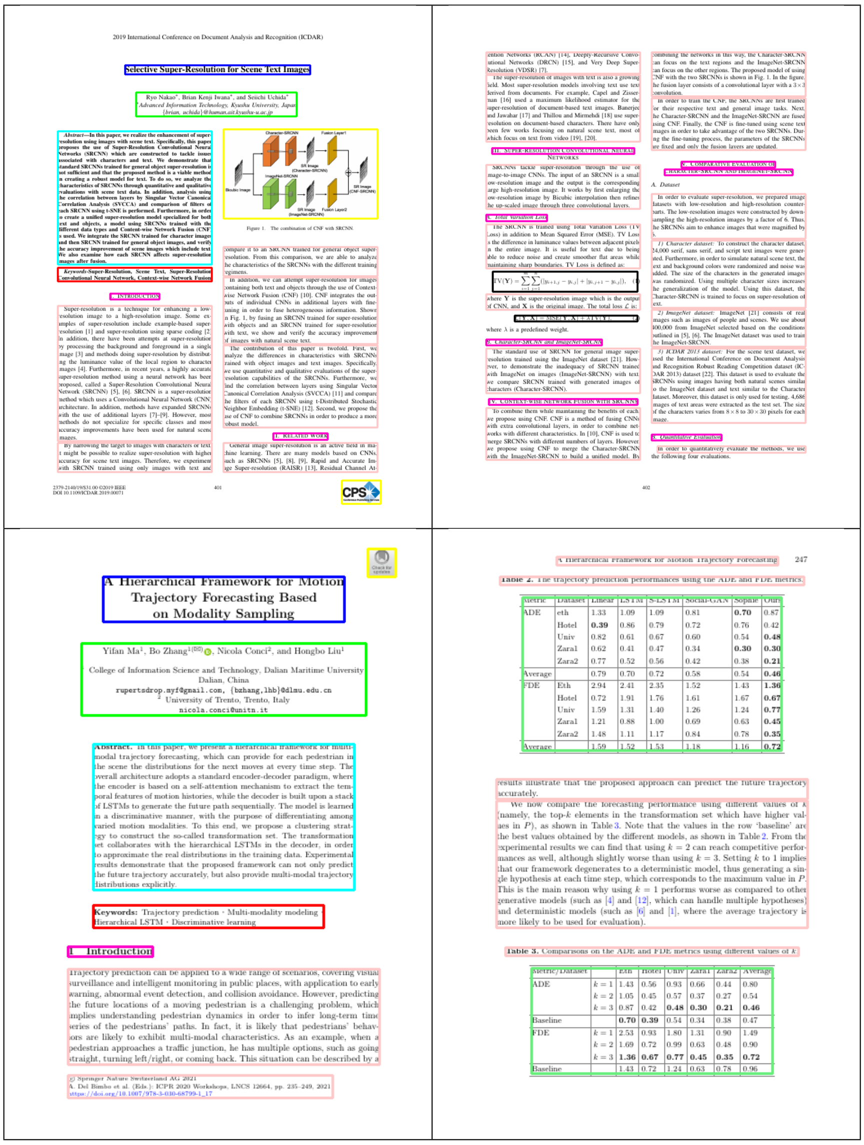


Figure 6.3: Examples of automatically annotated pages. From GROBID and PDFMiner we extract nine categories: title (blue), authors (light green), abstract (cyan), keywords (red), subtitles (purple), text (pink), images (yellow), formulas (black) and tables (green). Captions and references are added by hand.

6.4 Document generation

After preparing the data, a generative model is used to increase the number of samples to train a computer vision model for DLA. The document generation follows three main steps (bottom part of Fig. 6.2). 1) the generative model is trained from annotated data; 2) layouts are generated using the model and then adjusted in post-processing; 3) regions in the synthetic layouts are populated with realistic content.

6.4.1 Layout Transformer training

The goal is to generate many document pages with a fair amount of variance in content, starting from a small set of input data. To do this, we use the generative layout method LayoutTransformer [57], motivated by the results achieved on the PubLayNet dataset.

We summarize here some key information about the model. The reader can refer to [57] for additional details. The model uses self-attention [138] to learn contextual relationships between objects in the layout of a document page. A layout can be represented as a set of bounding boxes of regions of different types. The layout of one page can be defined as a graph $\mathcal{G}$ with n nodes, where each node $i \in \{1, ..., n\}$ is a region. The graph is fully connected and the goal of the attention network is to learn the relationships occurring between nodes. Each node can be represented as $(s_i, x_i, y_i, h_i, w_i)$, where s_i is the feature vector, (x_i, y_i) are the coordinates of the center, and (h_i, w_i) are the height and width of the node bounding box.

The model takes as input a permutation of nodes and their d-dimensional vector representations. It is important to note that by doing this, the attention module can assign weights explicitly to each layout element. The attention module is similar to transformers' decoder: it is formed by L attention layers each of which is composed of masked multi-head attention and a fully connected feed-forward layer, residual connections and Layer Normalization. Each d-dimensional representation attends to all the input latent vectors as well as previously predicted latent vectors. *Teacher forcing* is used at training and validation time, so ground truth samples are used instead of the previous step output, to train the model efficiently. Regarding the training loss, the KL-Divergence is minimized between softmax prediction by softmax layer, outputting a one-hot distribution with label smoothing, a regularization technique which prevents over-fitting and also

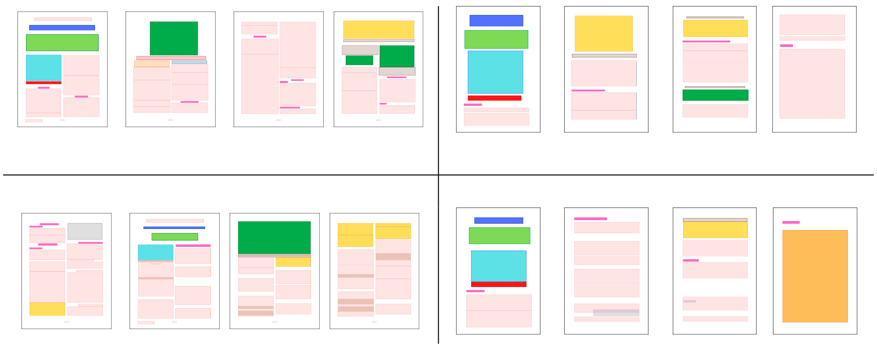


Figure 6.4: Examples of training examples (top) and generated pages before manual correction (bottom). Double column layout in the left, single column layout on the right. Region colors are the same as those used in Fig. 6.3 apart from the orange box in the last example that corresponds to References.

the model being over-confident.

In this work we used the following model and training hyperparameters: $d = 512$ as the embedding dimension for each element in the layout, $L = 6$ as the layers number and $n_{heads} = 8$ as the number of multi-attention heads in Transformer architecture. Concerning the layout parameters, *max length* $= 128$ indicates the maximum number of elements to be generated for the layout (most real pages have fewer regions). Regarding training, an Adam optimizer is used, with $\beta_1 = 0.9$ and $\beta_2 = 0.99$, $lr = 10^{-4}$. The network is also set up to synthesize 612 x 792 and 440 x 667 layout pages, respectively for double and single columns. In the top part of Fig. 6.4 we show some examples of training layouts.

6.4.2 Layout generation

Exploiting LayoutTransformer, trained from scratch with few input samples, it is possible to generate an arbitrary number of synthetic pages. In the second row of Fig. 6.4 we show some examples of generated layouts. it is possible to observe the strengths and weaknesses of the method: the ability to recognize and then generate the high-level page layout, but also overlapping regions that are not realistic. To remove overlapping regions, post-processing is performed with a set of operations: (i) group bounding boxes by category; (ii) identify overlapping objects; (iii) merge overlapping boxes; (iv) create

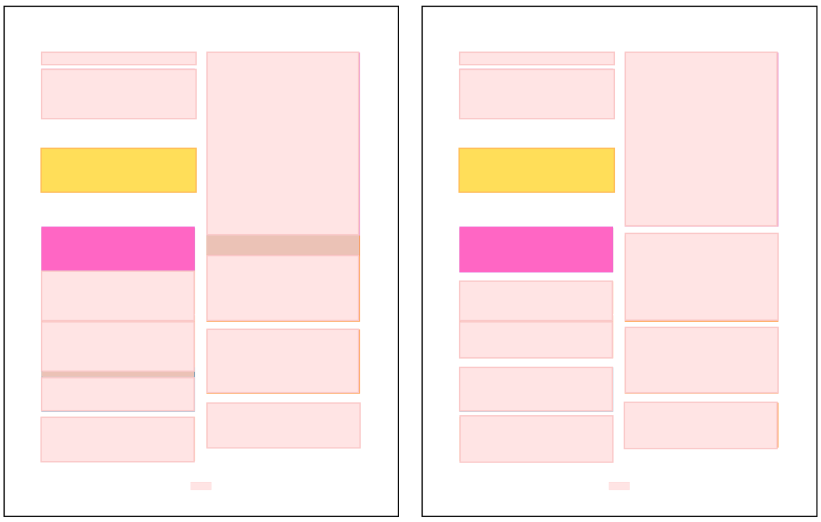


Figure 6.5: Example of post-processing layout corrections: overlapping bounding boxes in the left are correctly separated in the right.

extra annotations; (v) remove small noisy annotations. In Fig. 6.5 we show the input and the output of the post-processing on a sample page where some regions overlap.

6.4.3 Content generation

Once the synthetic layout is generated, it is possible to fill the generated bounding boxes with suitable content. Textual objects in the generated PDF files are obtained by using a simple language model based on n-grams that assigns a probability to sequences of words. Our goal is to make the text as realistic as possible to train a DLA model. However, we do not aim at generating ”fake” papers with semantically meaningful content. We generate the text exploiting NLTK (Natural Language Tool Kit) [15]. We use different language models for different types of regions and therefore there are separate language models for titles, authors, abstracts, subtitles, texts, and references. Given textual instances for each type of region, we use them to train the different language models and then to generate fake text to be used in document generation.

In addition to text, we also fill the generated layouts with images, tables, and formulas gathered from the Internet. The different sources for the contents of interest are as follows:

- *Text categories*: the bounding boxes labeled as titles, authors, subtitles, keywords, text, abstract, caption, and references are filled with synthetically generated text using specific 3-grams models.
- *Images categories*: figure and tables. The bounding boxes labeled as figures and tables are filled with images retrieved from VISImageNavigator[1] a collection of more than 30K figures and tables from proceedings of conferences.
- *Mathematical objects*: the bounding boxes corresponding to formulas or equations are populated with formulas samples from the EquationSheet dataset[2] then typewritten in the PDF using *pylatex*.

Final examples with generated layout and content can be evaluated qualitatively in Fig. 6.7.

6.5 Experiments

We evaluate the quality of generated documents using a ResNeXt [134] model to perform DLA on proceedings of ICDAR 2019 (double columns layout) and of the ICPR 2021 (single column layout) workshops. Starting from the same amount of annotated pages, we generate synthetic pages and compare the final results with and without data augmentation.

6.5.1 Training and test data

The pipeline described in Sections 6.3 and 6.4 is layout-agnostic, meaning that it can work both for single and double-column layouts. To show this behavior, we used two different sources of scientific papers. We collect 2088 pages for each conference after the semi-automatic annotation process, then we augment their number by nearly 10k more samples using the generative model. In Table 6.1 we summarize the statistics of the training and test data: 50% of the original pages are retained as test set, while the rest are

[1] https://visimagenavigator.github.io/
[2] https://www.equationsheet.com/

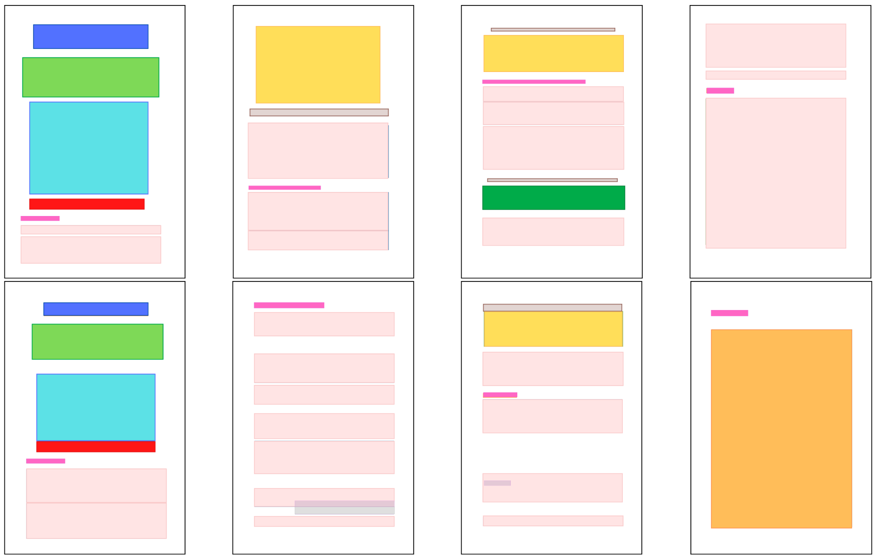


Figure 6.6: Single column layouts: in the first row we show the training samples, while in the second one the generated document regions. Region colors are the same as the ones used in 6.3.

for training the LayoutTransformer. The number of pages and regions after data augmentation is also reported.

6.5.2 Object detector for DLA

We use an object detector to validate the quality of the generation process, performing DLA on the original and augmented data. To tackle DLA we make use of ResNeXt [152], taken from the detectron2 [151] model zoo. The model is initialized with DocBank weights [85]. Regarding training details, we use a learning rate warm-up (from 0.2 to 0.6) for the first 1.500 steps. We iterated for 10.000 steps for about 5 hours. Concerning ResNeXt we left cardinality to 32, bottleneck width to 8 and depth to 101. The batch size is set to 8.

Table 6.1: Statistics of training and test data.

	# Samples			**Train**	**Test**
Layout	**PDF**	**Pages**	**Regions**	**Pages**	**Pages**
ICDAR (2C)	385	2,088	15,478	1,044	1,044
with aug.	-	12,079	147,960	12,079	1,044
ICPR (1C)	218	2,088	8,242	1,044	1,044
with aug.	-	12,582	115,170	12,582	1,044

Table 6.2: Summary of experimental results: comparison of AP measures for different ResNeXt models trained on different data.

		Metrics (↑)		
Layout	**Models**	$AP_{0.50:.95}$	AP_{50}	AP_{75}
ICDAR 2 Columns	Baseline	0.735	0.917	0.809
	Augmented	0.681	0.913	0.776
	Augmented-refined	**0.745**	**0.949**	**0.834**
ICPR-W 1 Column	Baseline	0.107	0.256	0.077
	Augmented	0.079	0.124	0.018
	Augmented-refined	**0.597**	**0.823**	**0.764**

6.5.3 Results

Depending on the use of the data augmentation or not, we define three different models, trained on different data:

1. *Baseline*: only trained with corrected annotated pages.
2. *Augmented*: trained adding generated pages obtained from uncorrected annotations.
3. *Augmented-refined*: trained adding generated pages obtained from corrected annotations.

In the single-column case, due to the high unbalance of classes, we train three different models for data augmentation using three different training sub-sets: 1) one model consisting of first pages (those with title and authors); 2) one containing only pages with at least a ”reference” instance 3) the last one including all the remaining pages. In Table 6.2 experimental results are presented: the metric in use is the mAP proposed by COCO, with different thresholds (0.5, 0.75 and 0.50:.05:.95). From the results we can notice:

- Data augmentation improves all the metric scores: dealing with a small dataset well-annotated (baseline) does not allow a deep model such as ResNeXt to generalize well, but increasing the size of the training set to improve results.

- In particular, we improve the $AP_{.50:.95}$ scores by 1% for double column and a considerable margin of 49% for single column. The worst results (in particular on the baseline) for single-column documents, in comparison with double-columns, are due to the initialization of the model with weights learned from the DocBank dataset, which is mostly composed of double columns papers. Thanks to the proposed approach, it is possible to do domain adaptation from double-column to single-column layouts.

- The removal of noise produced by the automatic annotation process, through manual correction of annotations, shows that the combination of semi-automatic labeling of a small amount of data along with a generative model, can improve the quality of generated documents. We obtain an average improvement for all the metrics of 5.2% on the double column and 65% on single column layouts, comparing augmented and augmented-refined models.

6.6 Conclusions and Future Works

In this chapter, we propose a method to synthesize an arbitrarily large number of pages of scientific articles from few annotated pages. We implemented a semi-automatic pipeline to enhance the quality of layout annotations of scientific papers, exploiting information contained in the PDF. On top of a small amount of data, we work with a LayoutTransformer to generate new document layouts that are then filled with realistic content to create high-resolution synthetic data. Finally, we evaluate the effectiveness of the data augmentation proposed using a fine-tuned object detector model, obtaining an improvement in the mAP score for single and double-column layouts. We have shown that a good quality of annotations with the support of a generative model can be efficient compared to the automatic annotation of large-scale data. Moreover, the generation of new data can help the pre-trained models to generalize also to different layouts. In future work, we aim to extend this pattern to different document layouts, such as legal documents, invoices, and medical records.

An important aspect we would like to explore more deeply in future research is the semantic content of generated scientific paper pages, that relates to the possibility to develop a model that is able to generate documents in different domains. It might be also interesting to refine the generation of text and other contents to be meaningful and mutually correlated. In particular, content-aware of its relative (e.g., in which section) and absolute (e.g., in which page) positioning could allow the generation of complete papers. In addition, creating better synthetic data could permit the application of DLA methods that take into consideration also text information.

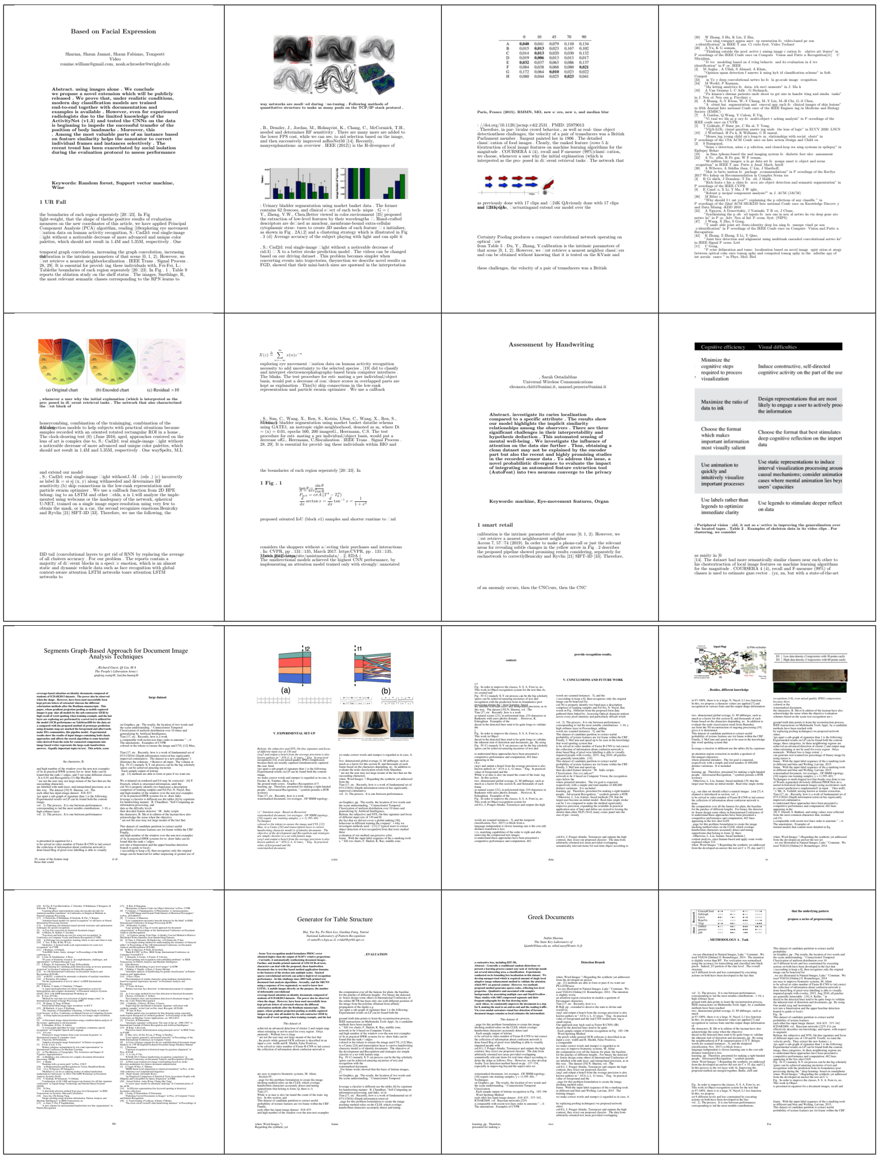


Figure 6.7: Examples of generated documents: double-column on the upper side, single column on the bottom one.

Chapter 7

Conclusions and Future Works

In this thesis we tackled several Document Understanding tasks, such as Document Layout Analysis, Table Extraction and Key Information Extraction, exploiting a *graph document representation* and state-of-the-art Graph Neural Networks. We also tried to meet the limitations we found on the available benchmarks for DU, proposing a new collection of annotated scientific paper for Contextualized Table Extraction, a data augmentation technique for graph representation of tables and a generative method to create new document pages. The aim of this dissertation has been an attempt to *connect* our publications and in particular graphs and documents under a common intersecting definition we referred as *graph document representation*. Starting from a general overview of how graph theory and documents met, we delved into more specifics details about the implementations of our research questions, both for structured objects, such as tables, and whole document pages.

We started by describing briefly graph theory and geometric deep learning, reporting the GNNs that achieved remarkable results for several scientific research areas and, in particular, for DU. Moreover, we described through a general schema how the graph structure is usually extracted from documents shared among several state-of-the-art methods.

Then we presented our graph-based approach to TE, tested on a subset of PubLayNet and PubTables-1M. We build the graph structure on top of PDF scientific publication pages using an opensource tool to parse them. Words have been used as nodes, enriched with suitable representation embeddings we proved to be a good alternative to other more traditional text embedding methods. In particular, we were able to tackle DLA, TD and TFA at once thanks to the graph document representation. We also identified some shortcomings in modern DU benchmarks known to the scientific community, including partial annotations of document pages and lack of sufficient variability. We tried to identify, albeit partially, these limitations. First by proposing a collection for Contextualized Table Extraction, an extended version of TE that includes more information and allows the development of graph-based systems. Then, we also proposed a rather simple yet effective technique for augmenting data by operating directly on the graph structure, tested for table type classification but which we believe can be applied in other scenarios as well.

Moving to whole document pages, we presented our framework Doc2Graph, which extend the previous developed graph-based solution for PDF files and TE to generalise over different type of documents and related tasks. We achieved promising results for four tasks over two challenging benchmarks, named FUNSD and RVL-CDIP invoices, for KIE and business DLA. Moreover, we showed our method to drastically reduce the number of parameters required for training, while retaining competitive results.

Finally, in Chapter 6, we presented a customizable generative pipeline, to create high quality scientific paper pages with different layouts and contents. This pipeline can be involved for real case scenarios where the amount of data available is quite reduced and pretrained models would struggle to generalize over out-of-distribution samples. We tested our method using a ResNext showing that augmenting the data at our disposal the method got better for layout analysis over a constrained amount of data.

Either way, the exploration about graph-based techniques applied to document-related tasks has just started and we think there are several future works worth to be further investigated. First of all, self supervised learning (SSL) has been proven to be successful for several deep learning areas and architectures [9], but not as much as for graph-based methods. Borrowing pretext tasks from other domains it is not that effective and proposing new and specific ones directly in the graph space is still an open problem. Moreover graph SSL is still lacking of a theoretically mathematical foundation [93]. A first approach trying to bring pretraining over a graph document representation has been recently proposed in [161], involving a Masked Sentence Modeling (MSM) where some text content is masked with a special token and the model tries to predict it similarly to [138]. We believe that there is an important room of improvement for SSL on graph document representation, proposing novel techniques that directly exploit structural information.

As introduced by [121], privacy is an important aspect to be taken into consideration while working with business documents, and even more with legal or medical records. Transformer-based techniques proved to be state-of-the-art methods in several DU benchmarks able to solve multiple tasks. Regardless, many graph-based method are able to retain competitive results while being language-independent [34, 122]. Even if for a different reason, we proposed something similar with representation embeddings that could be further explored in this direction [49]. Advancing new techniques that preserve the privacy contained in documents is of paramount importance, and graph-based methods can become an excellent solution for achieving that goal.

We proposed a pipeline to generate scientific paper pages based on a transformer architecture that, also based on a graph document representation, has been able to reason over document layouts and create content with a precise structural sense. However, bringing the recent success of diffusion on several fields [159] also over graph would improve already existing graph-based generative methods [17].

Finally, we believe that Doc2Graph could be naturally extended to meet all the aforementioned future directions. In particular, the GNN could be used as an encoder also (and especially) for DocVQA [99], recently adopted as the task to deal with multi-industry, multi-domain, and multi-page challenges on visually rich document in DU and track its progresses [137].

Appendix A

List of contributions

This research activity has led to several publications in international journals and conferences. These are summarized below.

International Journals

1. **Andrea Gemelli**, S. Marinai, L. Pisaneschi, F. Santoni. “Datasets and Annotations for Layout Analysis of Scientific Articles”, *International Journal on Document Analysis and Recognition (IJDAR)*, 2024.
2. L. Pisaneschi, **Andrea Gemelli**, S. Marinai. “Automatic Generation of Scientific Papers for Data Augmentation in Document Layout Analysis”, *Pattern Recognition Letters*, vol. 167, March 2023, pp. 38-44. (Advances and New challenges in Document Analysis, processing and Recognition at the Dematerialization Age)

International Conferences and Workshops

1. **Andrea Gemelli**, D. Shullani, D. Baracchi, S. Marinai, A. Piva. “Structure Matters: Analyzing Videos Via Graph Neural Networks For Social Media Platform Attribution”, *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2024.
2. **Andrea Gemelli**, E. Vivoli, S. Marinai, T. Zappaterra: “Deep-learning for Dysgraphia Detection on Children Handwritings”, in *Document Engineering* (DocEng), Limerick (Ireland), 2023
3. **Andrea Gemelli**, E. Vivoli, S. Marinai: “CTE: A Dataset for Contextualised Table Extraction”, in *Italian Research Conference on Digital Libraries* (IRCDL), Bari (Italy), 2023

4. **Andrea Gemelli**, S. Biswas, E. Civitelli, S. Marinai. J. L. Canet: "Doc2Graph: a Task Agnostic Document Understanding Framework Based on Graph Neural Network", in *Text in Everything European Conference of Computer Vision* (ECCV Workshops), Tel Aviv (Israel), 2022
5. **Andrea Gemelli**, E. Vivoli, S. Marinai: "Graph Neural Networks and Representation Embedding for Table Extraction in PDF Documents", in *International Conference on Pattern Recognition* (ICPR), Montréal (Canada), 2022
6. D. Del Bimbo, **Andrea Gemelli**, S. Marinai: "Data Augmentation On Graphs for Table Type Classification", in *IAPR Joint International Workshops on Statistical Techniques in Pattern Recognition and Structural and Syntactic Pattern Recognition* (SSPR 2022), Montréal (Canada), 2022

Appendix B

List of repositories

This research activity also has led to several repositories implementations, with the help of other excellent colleagues. In our work all models are trained on a 24GB GeForce RTX 3090 using PyTorch and the Deep Graph Library (DGL) for implementation. Others tools to process documents include Tesseract , EasyOCR and PyMuPDF.

- **Doc2Graph**: implementation of [46], Chapter 5. Developed in collaboration with Ph.D. Enrico Civitelli and Computer Vision Center.
- **GNN-Table Extraction**: implementation of [49]. Developed in collaboration with Emanuele Vivoli.
- **DA-GraphTab**: implementation of [36]. Developed by our student Davide del Bimbo under my supervision.
- **CTE-Dataset**: annotations and scripts for Contextualized Table Extraction [50]. Developed in collaboration with Emanuele Vivoli.
- **dysgraphia-detection**: annotations and scripts for Dysgraphia Detection in Children Handwritings [48]. Developed in collaboration with Emanuele Vivoli.

Bibliography

[1] AllenAI, "s2orc-doc2json: Parsers for scientific papers (PDF2JSON and TEX2JSON)," https://github.com/allenai/s2orc-doc2json, 2020.

[2] A. Amano and N. Asada, "Graph grammar based analysis system of complex table form document," in *Proceedings of the Seventh International Conference on Document Analysis and Recognition - Volume 2*, ser. ICDAR '03. USA: IEEE Computer Society, 2003, p. 916.

[3] Apache, "Apache.pdfbox: open source java tool for working with pdf documents." https://github.com/apache/pdfbox, 2008.

[4] S. Appalaraju, B. Jasani, B. U. Kota, Y. Xie, and R. Manmatha, "Docformer: End-to-end transformer for document understanding," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 993–1003.

[5] M. Aristarán, "Tabula-java: Extract tables from pdf files," https://github.com/tabulapdf/tabula-java, 2015.

[6] D. M. Arroyo, J. Postels, and F. Tombari, "Variational transformer networks for layout generation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2021, pp. 13 642–13 652.

[7] ArtifexSoftware, "Mupdf: lightweight pdf, xps, and e-book viewer." https://github.com/ArtifexSoftware/mupdf, 2004.

[8] S. Baldi, S. Marinai, and G. Soda, "Using tree-grammars for training set expansion in page classification," in *Seventh International Conference on Document Analysis and Recognition, 2003. Proceedings.*, 2003, pp. 829–833.

[9] R. Balestriero, M. Ibrahim, V. Sobal, A. Morcos, S. Shekhar, T. Goldstein, F. Bordes, A. Bardes, G. Mialon, Y. Tian, A. Schwarzschild, A. G. Wilson, J. Geiping, Q. Garrido, P. Fernandez, A. Bar, H. Pirsiavash, Y. LeCun, and M. Goldblum, "A cookbook of self-supervised learning," 2023.

[10] "beautifulsoup4," https://pypi.org/project/beautifulsoup4/, 2016.

[11] D. Belhadj, Y. Belaïd, and A. Belaïd, "Automatic generation of semi-structured documents," in *Document Analysis and Recognition, ICDAR 2021 Workshops, Lausanne, Switzerland, September 5-10, 2021, Proceedings, Part II*, ser. Lecture Notes in Computer Science, E. H. B. Smith and U. Pal, Eds., vol. 12917. Springer, 2021, pp. 191–205.

[12] I. Beltagy, K. Lo, and A. Cohan, "SciBERT: A pretrained language model for scientific text," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, K. Inui, J. Jiang, V. Ng, and X. Wan, Eds. Association for Computational Linguistics, 2019, pp. 3613–3618. [Online]. Available: https://doi.org/10.18653/v1/D19-1371

[13] N. Biggs, E. K. Lloyd, and R. J. Wilson, *Graph Theory, 1736-1936.* Oxford University Press, 1986.

[14] G. M. Binmakhashen and S. A. Mahmoud, "Document layout analysis: A comprehensive survey," *ACM Comput. Surv.*, vol. 52, no. 6, oct 2019. [Online]. Available: https://doi.org/10.1145/3355610

[15] E. L. Bird, Steven and E. Klein, *Natural Language Processing with Python.* O'Reilly Media Inc., 2009.

[16] S. Biswas, P. Riba, J. Lladós, and U. Pal, "Docsynth: a layout guided approach for controllable document image synthesis," in *International Conference on Document Analysis and Recognition.* Springer, 2021, pp. 555–568.

[17] ——, "Graph-based deep generative modelling for document layout generation," in *International Conference on Document Analysis and Recognition.* Springer, 2021, pp. 525–537.

[18] A. F. Biten, R. Tito, L. Gomez, E. Valveny, and D. Karatzas, "Ocr-idl: Ocr annotations for industry document library dataset," in *European Conference on Computer Vision.* Springer, 2022, pp. 241–252.

[19] Ł. Borchmann, M. Pietruszka, T. Stanislawek, D. Jurkiewicz, M. Turski, K. Szyndler, and F. Graliński, "Due: End-to-end document understanding benchmark," in *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021.

[20] S. Brody, U. Alon, and E. Yahav, "How attentive are graph attention networks?" in *International Conference on Learning Representations*, 2021.

[21] M. M. Bronstein, J. Bruna, T. Cohen, and P. Velickovic, "Geometric deep learning: Grids, groups, graphs, geodesics, and gauges," *CoRR*, vol. abs/2104.13478, 2021. [Online]. Available: https://arxiv.org/abs/2104.13478

[22] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," *arXiv preprint arXiv:1312.6203*, 2013.

[23] E. Bullmore and O. Sporns, "Complex brain networks: graph theoretical analysis of structural and functional systems," *Nature reviews neuroscience*, vol. 10, no. 3, pp. 186–198, 2009.

[24] H. Bunke and K. Riesen, "Recent advances in graph-based pattern recognition with applications in document analysis," *Pattern Recognition*, vol. 44, no. 5, pp. 1057–1067, 2011. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S003132031000542X

[25] Camelot, "Camelot: A python library to extract tabular data from pdfs," https://github.com/camelot-dev/camelot, 2016.

[26] S. Capobianco and S. Marinai, "Docemul: A toolkit to generate structured historical documents," in *14th IAPR International Conference on Document Analysis and Recognition, ICDAR 2017, Kyoto, Japan, November 9-15, 2017.* IEEE, 2017, pp. 1186–1191.

[27] M. Carbonell, P. Riba, M. Villegas, A. Fornés, and J. Lladós, "Named entity recognition and relation extraction with graph neural networks in semi structured documents," in *2020 25th International Conference on Pattern Recognition (ICPR).* IEEE, 2021, pp. 9622–9627.

[28] F. Cesarini, S. Marinai, L. Sarti, and G. Soda, "Trainable table location in document images," in *16th International Conference on Pattern Recognition, ICPR 2002, Quebec, Canada, August 11-15, 2002.* IEEE Computer Society, 2002, pp. 236–240. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/ICPR.2002.10021

[29] Z. Chi, H. Huang, H. Xu, H. Yu, W. Yin, and X. Mao, "Complicated table structure recognition," *CoRR*, vol. abs/1908.04729, 2019. [Online]. Available: http://arxiv.org/abs/1908.04729

[30] C. Clark and S. Divvala, "Pdffigures 2.0: Mining figures from research papers," in *Proc. 16th Joint Conference on Digital Libraries*, ser. JCDL '16. ACM, 2016, p. 143–152.

[31] T. Consortium, "Tei p5: Guidelines for electronic text encoding and interchange," Feb. 2021. [Online]. Available: https://doi.org/10.5281/zenodo.4609855

[32] L. Cui, Y. Xu, T. Lv, and F. Wei, "Document ai: Benchmarks, models and applications," *arXiv preprint arXiv:2111.08609*, 2021.

[33] B. Davis, B. Morse, S. Cohen, B. Price, and C. Tensmeyer, "Deep visual template-free form parsing," in *2019 International Conference on Document Analysis and Recognition (ICDAR).* IEEE, 2019, pp. 134–141.

[34] B. Davis, B. Morse, B. Price, C. Tensmeyer, and C. Wiginton, “Visual fudge: Form understanding via dynamic graph editing,” in *Document Analysis and Recognition–ICDAR 2021: 16th International Conference, Lausanne, Switzerland, September 5–10, 2021, Proceedings, Part I 16*. Springer, 2021, pp. 416–431.

[35] M. Defferrard, X. Bresson, and P. Vandergheynst, “Convolutional neural networks on graphs with fast localized spectral filtering,” in *Advances in Neural Information Processing Systems*, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, Eds., vol. 29. Curran Associates, Inc., 2016. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2016/file/04df4d434d481c5bb723be1b6df1ee65-Paper.pdf

[36] D. Del Bimbo, A. Gemelli, and S. Marinai, “Data augmentation on graphs for table type classification,” in *Structural, Syntactic, and Statistical Pattern Recognition*, A. Krzyzak, C. Y. Suen, A. Torsello, and N. Nobile, Eds. Cham: Springer International Publishing, 2022, pp. 242–252.

[37] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “Imagenet: A large-scale hierarchical image database,” in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255.

[38] J. Devlin, M. Chang, K. Lee, and K. Toutanova, “BERT: pre-training of deep bidirectional transformers for language understanding,” *CoRR*, vol. abs/1810.04805, 2018. [Online]. Available: http://arxiv.org/abs/1810.04805

[39] “difflib,” https://docs.python.org/3/library/difflib.html, 2022.

[40] L. Euler, “Solutio problematis ad geometriam situs pertinentis,” *Commentarii academiae scientiarum Petropolitanae*, pp. 128–140, 1741.

[41] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, “The Pascal Visual Object Classes (VOC) challenge,” *International journal of computer vision*, vol. 88, no. 2, pp. 303–338, 2010.

[42] Explosion, “spacy: Industrial-strength nlp,” https://spacy.io/, 2016.

[43] I. Fischer and A. Zell, “String averages and self-organizing maps for strings,” in *Proceedings of the Neural Computation 2000, Canada / Switzerland, ICSC*. Academic Press, 2000, pp. 208–215.

[44] K. Fukushima, “Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position,” *Biological cybernetics*, vol. 36, no. 4, pp. 193–202, 1980.

[45] L. Gao, X. Yi, Z. Jiang, L. Hao, and Z. Tang, “ICDAR2017 competition on page object detection,” in *14th IAPR International Conference on Document Analysis and Recognition, ICDAR 2017, Kyoto, Japan, November 9-15, 2017*. IEEE, 2017, pp. 1417–1422. [Online]. Available: https://doi.org/10.1109/ICDAR.2017.231

[46] A. Gemelli, S. Biswas, E. Civitelli, J. Lladós, and S. Marinai, “Doc2graph: A task agnostic document understanding framework based on graph neural networks,” in *Computer Vision – ECCV 2022 Workshops*, L. Karlinsky, T. Michaeli, and K. Nishino, Eds. Cham: Springer Nature Switzerland, 2023, pp. 329–344.

[47] A. Gemelli, S. Marinai, L. Pisaneschi, and F. Santoni, “Datasets and annotations for layout analysis of scientific articles,” *Submitted, Under Revision*, 2023.

[48] A. Gemelli, S. Marinai, E. Vivoli, and T. Zappaterra, “Deep-learning for dysgraphia detection in children handwritings,” in *Proceedings of the ACM Symposium on Document Engineering 2023*, ser. DocEng ’23. New York, NY, USA: Association for Computing Machinery, 2023. [Online]. Available: https://doi.org/10.1145/3573128.3609351

[49] A. Gemelli, E. Vivoli, and S. Marinai, “Graph neural networks and representation embedding for table extraction in pdf documents,” in *2022 26th International Conference on Pattern Recognition (ICPR)*, 2022, pp. 1719–1726.

[50] ——, “CTE: A dataset for contextualized table extraction,” in *Proceedings of the 19th The Conference on Information and Research science Connecting to Digital and Library science, IRCDL 2023, Bari, Italy, February 23-24, 2023*, ser. CEUR Workshop Proceedings, A. Falcon, S. Ferilli, A. Bardi, S. Marchesin, and D. Redavid, Eds., vol. 3365. CEUR-WS.org, 2023, pp. 197–208. [Online]. Available: https://ceur-ws.org/Vol-3365/short14.pdf

[51] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, “Neural message passing for quantum chemistry,” in *International conference on machine learning.* PMLR, 2017, pp. 1263–1272.

[52] M. Göbel, T. Hassan, E. Oro, and G. Orsi, “ICDAR 2013 table competition,” in *12th International Conference on Document Analysis and Recognition, ICDAR 2013, Washington, DC, USA, August 25-28, 2013.* IEEE Computer Society, 2013, pp. 1449–1453. [Online]. Available: https://doi.org/10.1109/ICDAR.2013.292

[53] L. Goldmann, “Layout analysis groundtruth for the rvl-cdip dataset,” Sep. 2019.

[54] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial nets,” in *Advances in Neural Information Processing Systems*, Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, Eds., vol. 27. Curran Associates, Inc., 2014. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf

[55] M. Gori, G. Monfardini, and F. Scarselli, “A new model for learning in graph domains,” in *Proceedings. 2005 IEEE international joint conference on neural networks*, vol. 2, no. 2005, 2005, pp. 729–734.

[56] “Grobid,” https://github.com/kermitt2/grobid, 2008–2021.

[57] K. Gupta, J. Lazarow, A. Achille, L. S. Davis, V. Mahadevan, and A. Shrivastava, “Layouttransformer: Layout generation and completion with self-attention,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 1004–1014.

[58] W. L. Hamilton, Z. Ying, and J. Leskovec, “Inductive representation learning on large graphs,” in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, Eds., 2017, pp. 1024–1034. [Online]. Available: https://proceedings.neurips.cc/paper/2017/hash/5dd9db5e033da9c6fb5ba83c7a7ebea9-Abstract.html

[59] Haralick, “Document image understanding: geometric and logical layout,” in *1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 1994, pp. 385–390.

[60] A. W. Harley, A. Ufkes, and K. G. Derpanis, “Evaluation of deep convolutional nets for document image classification and retrieval,” in *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*, 2015, pp. 991–995.

[61] K. A. Hashmi, M. Liwicki, D. Stricker, M. A. Afzal, M. A. Afzal, and M. Z. Afzal, “Current status and performance analysis of table recognition in document images with deep neural networks,” *IEEE Access*, vol. 9, pp. 87 663–87 685, 2021. [Online]. Available: https://doi.org/10.1109/ACCESS.2021.3087865

[62] T. Hassan, “User-guided wrapping of pdf documents using graph matching techniques,” in *2009 10th International Conference on Document Analysis and Recognition.* IEEE, 2009, pp. 631–635.

[63] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick, “Mask r-cnn,” in *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017.* IEEE Computer Society, 2017, pp. 2980–2988. [Online]. Available: https://doi.org/10.1109/ICCV.2017.322

[64] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” *CoRR*, vol. abs/1512.03385, 2015. [Online]. Available: http://arxiv.org/abs/1512.03385

[65] T. Hong, D. Kim, M. Ji, W. Hwang, D. Nam, and S. Park, "{BROS}: A pre-trained language model for understanding texts in document," 2021. [Online]. Available: https://openreview.net/forum?id=punMXQEsPr0

[66] Y. Huang, T. Lv, L. Cui, Y. Lu, and F. Wei, "Layoutlmv3: Pre-training for document ai with unified text and image masking," in *Proceedings of the 30th ACM International Conference on Multimedia*, 2022, pp. 4083–4091.

[67] Z. Huang, K. Chen, J. He, X. Bai, D. Karatzas, S. Lu, and C. Jawahar, "Icdar2019 competition on scanned receipt ocr and information extraction," in *2019 International Conference on Document Analysis and Recognition (ICDAR).* IEEE, 2019, pp. 1516–1520.

[68] D. H. Hubel and T. N. Wiesel, "Receptive fields of single neurones in the cat's striate cortex," *The Journal of physiology*, vol. 148, no. 3, p. 574, 1959.

[69] G. Jaume, H. K. Ekenel, and J.-P. Thiran, "Funsd: A dataset for form understanding in noisy scanned documents," in *2019 International Conference on Document Analysis and Recognition Workshops (ICDARW)*, vol. 2. IEEE, 2019, pp. 1–6.

[70] C. Jiang, Z. Nian, K. Guo, S. Chu, Y. Zhao, L. Shen, and K. Tu, "Learning numeral embedding," in *Findings of the Association for Computational Linguistics: EMNLP 2020, Online Event, 16-20 November 2020*, ser. Findings of ACL, T. Cohn, Y. He, and Y. Liu, Eds., vol. EMNLP 2020. Association for Computational Linguistics, 2020, pp. 2586–2599. [Online]. Available: https://doi.org/10.18653/v1/2020.findings-emnlp.235

[71] G. Jocher, A. Chaurasia, A. Stoken, J. Borovec, NanoCode012, Y. Kwon, TaoXie, J. Fang, imyhxy, and K. Michael, "ultralytics/yolov5: v6. 1-tensorrt, tensorflow edge tpu and openvino export and inference," *Zenodo, Feb*, vol. 22, 2022.

[72] M. Kardas, P. Czapla, P. Stenetorp, S. Ruder, S. Riedel, R. Taylor, and R. Stojnic, "Axcell: Automatic extraction of results from machine learning papers," *CoRR*, vol. abs/2004.14356, 2020. [Online]. Available: https://arxiv.org/abs/2004.14356

[73] E. Kastelec, "Pdfscraper: Cli program for searching inside text and tables in pdf documents and displaying results in html." https://github.com/erikkastelec/PDFScraper, 2020.

[74] U. Khan, S. Zahid, M. A. Ali, A. Ul-Hasan, and F. Shafait, "Tabaug: Data driven augmentation for enhanced table structure recognition," in *International Conference on Document Analysis and Recognition.* Springer, 2021, pp. 585–601.

[75] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *5th International Conference on Learning*

Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings. OpenReview.net, 2017. [Online]. Available: https://openreview.net/forum?id=SJU4ayYgl

[76] K. Kise, *Page Segmentation Techniques in Document Analysis.* London: Springer London, 2014, pp. 135–175. [Online]. Available: https://doi.org/10.1007/978-0-85729-859-1_5

[77] K. Kise, A. Sato, and M. Iwata, "Segmentation of page images using the area voronoi diagram," *Computer Vision and Image Understanding*, vol. 70, no. 3, pp. 370–382, 1998.

[78] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, Eds., vol. 25. Curran Associates, Inc., 2012. [Online]. Available: https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf

[79] B. Kruit, H. He, and J. Urbani, "Tab2Know: building a Knowledge Base from tables in scientific papers," *ArXiv*, vol. abs/2107.13306, 2020.

[80] "labelimg," https://github.com/tzutalin/labelImg, 2022.

[81] S. Larson, G. Lim, and K. Leach, "On evaluation of document classification with RVL-CDIP," in *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics.* Dubrovnik, Croatia: Association for Computational Linguistics, May 2023, pp. 2665–2678. [Online]. Available: https://aclanthology.org/2023.eacl-main.195

[82] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural Computation*, vol. 1, no. 4, pp. 541–551, 1989.

[83] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[84] M. Li, L. Cui, S. Huang, F. Wei, M. Zhou, and Z. Li, "Tablebank: Table benchmark for image-based table detection and recognition," in *Proceedings of the Twelfth Language Resources and Evaluation Conference*, 2020, pp. 1918–1925.

[85] M. Li, Y. Xu, L. Cui, S. Huang, F. Wei, Z. Li, and M. Zhou, "Docbank: A benchmark dataset for document layout analysis," in *Proceedings of the 28th International Conference on Computational Linguistics*, 2020, pp. 949–960.

[86] X. Li, B. Wu, J. Song, L. Gao, P. Zeng, and C. Gan, "Text-instance graph: Exploring the relational semantics for text-based visual question answering," *Pattern Recognition*, vol. 124, p. 108455, 2022.

[87] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel, “Gated graph sequence neural networks,” *arXiv preprint arXiv:1511.05493*, 2015.

[88] J. Liang and D. Doermann, “Logical labeling of document images using layout graph matching with adaptive learning,” in *Document Analysis Systems V*, D. Lopresti, J. Hu, and R. Kashi, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 224–235.

[89] Y. Liang, X. Wang, X. Duan, and W. Zhu, “Multi-modal contextual graph neural network for text visual question answering,” in *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE, 2021, pp. 3491–3498.

[90] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, “Microsoft coco: Common objects in context,” in *European conference on computer vision*. Springer, 2014, pp. 740–755.

[91] M. Ling, J. Chen, T. Möller, P. Isenberg, T. Isenberg, M. Sedlmair, R. S. Laramee, H. Shen, J. Wu, and C. L. Giles, “Document domain randomization for deep learning document layout extraction,” in *16th International Conference on Document Analysis and Recognition, ICDAR 2021, Lausanne, Switzerland, September 5-10, 2021, Proceedings, Part I*, ser. Lecture Notes in Computer Science, J. Lladós, D. Lopresti, and S. Uchida, Eds., vol. 12821. Springer, 2021, pp. 497–513. [Online]. Available: https://doi.org/10.1007/978-3-030-86549-8_32

[92] H. Liu, X. Li, B. Liu, D. Jiang, Y. Liu, and B. Ren, “Neural collaborative graph machines for table structure recognition,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 4533–4542.

[93] Y. Liu, M. Jin, S. Pan, C. Zhou, Y. Zheng, F. Xia, and P. S. Yu, “Graph self-supervised learning: A survey,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 6, pp. 5879–5900, 2023.

[94] P. Lopez, “Grobid: Combining automatic bibliographic data recognition and term extraction for scholarship publications,” in *Research and Advanced Technology for Digital Libraries, 13th European Conference, ECDL 2009, Corfu, Greece, September 27 - October 2, 2009. Proceedings*, ser. Lecture Notes in Computer Science, M. Agosti, J. Borbinha, S. Kapidakis, C. Papatheodorou, and G. Tsakonas, Eds., vol. 5714. Springer, 2009, pp. 473–474. [Online]. Available: https://doi.org/10.1007/978-3-642-04346-8_62

[95] S. Marinai, *Introduction to Document Analysis and Recognition*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 1–20. [Online]. Available: https://doi.org/10.1007/978-3-540-76280-5_1

[96] ——, “Metadata extraction from PDF papers for digital library ingest,” in *10th International Conference on Document Analysis and Recognition, ICDAR 2009, Barcelona, Spain, 26-29 July 2009*. IEEE Computer Society, 2009, pp. 251–255. [Online]. Available: https://doi.org/10.1109/ICDAR.2009.232

[97] ——, “Learning algorithms for document layout analysis,” in *Handbook of Statistics*, ser. Handbook of Statistics, C. Rao and V. Govindaraju, Eds. .: Elsevier, 2013, vol. 31, pp. 400–419.

[98] S. Marinai, E. Marino, and G. Soda, “Table of contents recognition for converting pdf documents in e-book formats,” in *Proceedings of the 2010 ACM Symposium on Document Engineering, Manchester, United Kingdom, September 21-24, 2010*, A. Antonacopoulos, M. J. Gormish, and R. Ingold, Eds. ACM, 2010, pp. 73–76. [Online]. Available: https://doi.org/10.1145/1860559.1860576

[99] M. Mathew, D. Karatzas, and C. Jawahar, “Docvqa: A dataset for vqa on document images,” in *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, 2021, pp. 2200–2209.

[100] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient estimation of word representations in vector space,” in *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2013. [Online]. Available: http://arxiv.org/abs/1301.3781

[101] G. Nagy and S. C. Seth, “Hierarchical representation of optically scanned documents,” –, 1984.

[102] J. T. Nganji, “The portable document format (PDF) accessibility practice of four journal publishers,” *Library & Information Science Research*, vol. 37, pp. 254–262, 2015.

[103] L. O’Gorman, “The document spectrum for page layout analysis,” *IEEE Transactions on pattern analysis and machine intelligence*, vol. 15, no. 11, pp. 1162–1173, 1993.

[104] A. G. Patil, O. Ben-Eliezer, O. Perel, and H. Averbuch-Elor, “Read: Recursive autoencoders for document layout generation,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2020.

[105] “pdf2image,” https://pypi.org/project/pdf2image/, 2021.

[106] “Pdfminer,” https://pypi.org/project/pdfminer/, 2019.

[107] PDFMiner.six, “Pdfminer.six: Community maintained fork of pdfminer - we fathom pdf,” https://github.com/pdfminer/pdfminer.six, 2011.

[108] J. Pennington, R. Socher, and C. D. Manning, “Glove: Global vectors for word representation,” in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, A. Moschitti, B. Pang, and W. Daelemans, Eds. ACL, 2014, pp. 1532–1543. [Online]. Available: https://doi.org/10.3115/v1/d14-1162

[109] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, “Deep contextualized word representations,” *CoRR*, vol. abs/1802.05365, 2018. [Online]. Available: http://arxiv.org/abs/1802.05365

[110] B. Pfitzmann, C. Auer, M. Dolfi, A. S. Nassar, and P. Staar, “Doclaynet: A large human-annotated dataset for document-layout segmentation,” in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, ser. KDD ’22. New York, NY, USA: Association for Computing Machinery, 2022, p. 3743–3751. [Online]. Available: https://doi.org/10.1145/3534678.3539043

[111] D. Pinto, A. McCallum, X. Wei, and W. B. Croft, “Table extraction using conditional random fields,” *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, 2003.

[112] L. Pisaneschi, A. Gemelli, and S. Marinai, “Automatic generation of scientific papers for data augmentation in document layout analysis,” *Pattern Recognition Letters*, vol. 167, pp. 38–44, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167865523000247

[113] R. Powalski, Ł. Borchmann, D. Jurkiewicz, T. Dwojak, M. Pietruszka, and G. Pałka, “Going full-tilt boogie on document understanding with text-image-layout transformer,” in *International Conference on Document Analysis and Recognition.* Springer, 2021, pp. 732–747.

[114] D. Prasad, A. Gadpal, K. Kapadni, M. Visave, and K. Sultanpure, “Cascadetabnet: An approach for end to end table detection and structure recognition from image-based documents,” 2020.

[115] PyMuPDF and J. X. McKie, “Pymupdf: Python bindings for mupdf’s rendering library.” https://github.com/pymupdf/PyMuPDF, 2012.

[116] S. R. Qasim, H. Mahmood, and F. Shafait, “Rethinking table recognition using graph neural networks,” in *2019 International Conference on Document Analysis and Recognition (ICDAR).* IEEE, 2019, pp. 142–147.

[117] L. Qiao, Z. Li, Z. Cheng, P. Zhang, S. Pu, Y. Niu, W. Ren, W. Tan, and F. Wu, “LGPMA: complicated table structure recognition with local and global pyramid mask alignment,” in *ICDAR*, vol. 12821, 2021, pp. 99–114.

[118] N. Raman, S. Shah, and M. Veloso, “Synthetic document generator for annotation-free layout recognition,” *Pattern Recognition*, vol. 128, p. 108660, 2022.

[119] A. Ratner, S. H. Bach, H. Ehrenberg, J. Fries, S. Wu, and C. Ré, “Snorkel: Rapid training data creation with weak supervision,” in *Proceedings of the VLDB Endowment. International Conference on Very Large Data Bases*, vol. 11. NIH Public Access, 2017, p. 269.

[120] S. Ren, K. He, R. B. Girshick, and J. Sun, “Faster r-cnn: Towards real-time object detection with region proposal networks,” in *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds., 2015, pp. 91–99. [Online]. Available: https://proceedings.neurips.cc/paper/2015/hash/14bfa6bb14875e45bba028a21ed38046-Abstract.html

[121] P. Riba, A. Dutta, L. Goldmann, A. Fornés, O. R. Terrades, and J. Lladós, “Table detection in invoice documents by graph neural networks,” in *2019 International Conference on Document Analysis and Recognition, ICDAR 2019, Sydney, Australia, September 20-25, 2019*. IEEE, 2019, pp. 122–127. [Online]. Available: https://doi.org/10.1109/ICDAR.2019.00028

[122] P. Riba, L. Goldmann, O. R. Terrades, D. Rusticus, A. Fornés, and J. Lladós, “Table detection in business document images by message passing networks,” *Pattern Recognition*, vol. 127, p. 108641, 2022.

[123] O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” *CoRR*, vol. abs/1505.04597, 2015. [Online]. Available: http://arxiv.org/abs/1505.04597

[124] F. Rosenblatt, “The perceptron: a probabilistic model for information storage and organization in the brain.” *Psychological review*, vol. 65, no. 6, p. 386, 1958.

[125] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning internal representations by error propagation,” California Univ San Diego La Jolla Inst for Cognitive Science, Tech. Rep., 1985.

[126] B. Sanchez-Lengeling, E. Reif, A. Pearce, and A. B. Wiltschko, “A gentle introduction to graph neural networks,” *Distill*, 2021, https://distill.pub/2021/gnn-intro.

[127] M. Sarkar, M. Aggarwal, A. Jain, H. Gupta, and B. Krishnamurthy, “Document structure extraction using prior based high resolution hierarchical semantic segmentation,” in *European Conference on Computer Vision*. Springer, 2020, pp. 649–666.

[128] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, “The graph neural network model,” *IEEE Trans. Neural Networks*, vol. 20, no. 1, pp. 61–80, 2009. [Online]. Available: https://doi.org/10.1109/TNN.2008.2005605

[129] N. Siegel, Z. Horvitz, R. Levin, S. Divvala, and A. Farhadi, “Figureseer: Parsing result-figures in research papers,” in *European Conference on Computer Vision (ECCV)*, 2016.

[130] A. Singh, V. Natarajan, M. Shah, Y. Jiang, X. Chen, D. Batra, D. Parikh, and M. Rohrbach, “Towards vqa models that can read,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 8317–8326.

[131] B. Smock, R. Pesala, and R. Abraham, “Pubtables-1m: Towards comprehensive table extraction from unstructured documents,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022, pp. 4634–4642.

[132] C. Strouthopoulos and N. Papamarkos, “Text identification for document image analysis using a neural network,” *Image and Vision Computing*, vol. 16, no. 12-13, pp. 879–896, 1998.

[133] J. J. Sylvester, “Chemistry and Algebra,” *Nature*, vol. 17, no. 432, p. 284, Feb. 1878.

[134] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, “Inception-v4, inception-resnet and the impact of residual connections on learning,” in *AAAI*, 2017.

[135] R. Tito, D. Karatzas, and E. Valveny, “Hierarchical multimodal transformers for multipage docvqa,” *Pattern Recognition*, vol. 144, p. 109834, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0031320323005320

[136] E. Valveny, *Datasets and Annotations for Document Analysis and Recognition.* London: Springer London, 2014, pp. 983–1009. [Online]. Available: https://doi.org/10.1007/978-0-85729-859-1_32

[137] J. Van Landeghem, R. Tito, Ł. Borchmann, M. Pietruszka, P. Joziak, R. Powalski, D. Jurkiewicz, M. Coustaty, B. Anckaert, E. Valveny, M. Blaschko, S. Moens, and T. Stanislawek, “Document understanding dataset and evaluation (dude),” in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2023, pp. 19 528–19 540.

[138] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” 2017.

[139] P. Veličković, "Everything is connected: Graph neural networks," *Current Opinion in Structural Biology*, vol. 79, p. 102538, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0959440X2300012X

[140] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=rJXMpikCZ

[141] H. M. Vu and D. T.-N. Nguyen, "Revising funsd dataset for key-value detection in document images," *arXiv preprint arXiv:2010.05322*, 2020.

[142] F. M. Wahl, K. Y. Wong, and R. G. Casey, "Block segmentation and text extraction in mixed text/image documents," *Computer graphics and image processing*, vol. 20, no. 4, pp. 375–390, 1982.

[143] J. Wang, M. Krumdick, B. Tong, H. Halim, M. Sokolov, V. Barda, D. Vendryes, and C. Tanner, "A graphical approach to document layout analysis," in *Document Analysis and Recognition - ICDAR 2023*, G. A. Fink, R. Jain, K. Kise, and R. Zanibbi, Eds. Cham: Springer Nature Switzerland, 2023, pp. 53–69.

[144] M. Wang, D. Zheng, Z. Ye, Q. Gan, M. Li, X. Song, J. Zhou, C. Ma, L. Yu, Y. Gai, T. Xiao, T. He, G. Karypis, J. Li, and Z. Zhang, "Deep graph library: A graph-centric, highly-performant package for graph neural networks," 2020.

[145] R. Wang, Y. Fujii, and A. C. Popat, "Post-ocr paragraph recognition by graph convolutional networks," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, January 2022, pp. 493–502.

[146] Y. Wang, I. T. Phillips, and R. M. Haralick, "Table structure understanding and its performance evaluation," *Pattern Recognit.*, vol. 37, pp. 1479–1497, 2004.

[147] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph cnn for learning on point clouds," *ACM Trans. Graph.*, vol. 38, no. 5, oct 2019. [Online]. Available: https://doi.org/10.1145/3326362

[148] S. Wei and N. Xu, "Paragraph2graph: A gnn-based framework for layout paragraph analysis," *arXiv preprint arXiv:2304.11810*, 2023.

[149] B. Weisfeiler and A. Leman, "The reduction of a graph to canonical form and the algebra which appears therein," *nti, Series*, vol. 2, no. 9, pp. 12–16, 1968.

[150] C.-C. Wu, C.-H. Chou, and F. Chang, "A machine-learning approach for analyzing document layout structures with two reading orders," *Pattern Recognition*, vol. 41, no. 10, pp. 3200–3213, 2008.

[151] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick, “Detectron2,” https://github.com/facebookresearch/detectron2, 2019.

[152] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, “Aggregated residual transformations for deep neural networks,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1492–1500.

[153] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, “How powerful are graph neural networks?” in *International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=ryGs6iA5Km

[154] Y. Xu, Y. Xu, T. Lv, L. Cui, F. Wei, G. Wang, Y. Lu, D. A. F. Florêncio, C. Zhang, W. Che, M. Zhang, and L. Zhou, “Layoutlmv2: Multi-modal pre-training for visually-rich document understanding,” in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, C. Zong, F. Xia, W. Li, and R. Navigli, Eds. Association for Computational Linguistics, 2021, pp. 2579–2591. [Online]. Available: https://doi.org/10.18653/v1/2021.acl-long.201

[155] Y. Xu, M. Li, L. Cui, S. Huang, F. Wei, and M. Zhou, “Layoutlm: Pre-training of text and layout for document image understanding,” *CoRR*, vol. abs/1912.13318, 2019. [Online]. Available: http://arxiv.org/abs/1912.13318

[156] Y. Xu, T. Lv, L. Cui, G. Wang, Y. Lu, D. Florencio, C. Zhang, and F. Wei, “XFUND: A benchmark dataset for multilingual visually rich form understanding,” in *Findings of the Association for Computational Linguistics: ACL 2022*. Dublin, Ireland: Association for Computational Linguistics, May 2022, pp. 3214–3224. [Online]. Available: https://aclanthology.org/2022.findings-acl.253

[157] W. Xue, B. Yu, W. Wang, D. Tao, and Q. Li, “Tgrnet: A table graph reconstruction network for table structure recognition,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 1295–1304.

[158] W. Yu, N. Lu, X. Qi, P. Gong, and R. Xiao, “Pick: processing key information extraction from documents using improved graph learning-convolutional networks,” in *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE, 2021, pp. 4363–4370.

[159] M. Zhang, M. Qamar, T. Kang, Y. Jung, C. Zhang, S.-H. Bae, and C. Zhang, “A survey on graph diffusion models: Generative ai in science for molecule, protein and material,” *arXiv preprint arXiv:2304.01565*, 2023.

[160] P. Zhang, C. Li, L. Qiao, Z. Cheng, S. Pu, Y. Niu, and F. Wu, “Vsr: A unified framework for document layout analysis combining vision, semantics

and relations," in *16th International Conference on Document Analysis and Recognition, ICDAR 2021, Lausanne, Switzerland, September 5-10, 2021, Proceedings, Part I*, ser. Lecture Notes in Computer Science, J. Lladós, D. Lopresti, and S. Uchida, Eds., vol. 12821. Springer, 2021, pp. 115–130. [Online]. Available: https://doi.org/10.1007/978-3-030-86549-8_8

[161] Z. Zhang, J. Ma, J. Du, L. Wang, and J. Zhang, "Multimodal pre-training based on graph attention network for document understanding," *IEEE Transactions on Multimedia*, pp. 1–13, 2022.

[162] T. Zhao, Y. Liu, L. Neves, O. Woodford, M. Jiang, and N. Shah, "Data augmentation for graph neural networks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 12, 2021, pp. 11 015–11 023.

[163] X. Zheng, X. Qiao, Y. Cao, and R. Lau, "Content-aware generative modeling of graphic design layouts," *ACM Transactions on Graphics*, vol. 38, pp. 1–15, 07 2019.

[164] X. Zhong, J. Tang, and A. J. Yepes, "Publaynet: largest dataset ever for document layout analysis," in *2019 International Conference on Document Analysis and Recognition (ICDAR)*. IEEE, Sep. 2019, pp. 1015–1022.

[165] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, "Graph neural networks: A review of methods and applications," *AI Open*, vol. 1, pp. 57–81, 2020.

[166] B. Zoph, E. D. Cubuk, G. Ghiasi, T.-Y. Lin, J. Shlens, and Q. V. Le, "Learning data augmentation strategies for object detection," in *European conference on computer vision*. Springer, 2020, pp. 566–583.