UNIVERSITY OF TWENTE.

Faculty of Electrical Engineering, Mathematics & Computer Science

UNIVERSITY
OF TWENTE.

ELSEVIER

Logical Structure Extraction of Electronic Documents Using Contextual Information

M.Sc. Thesis

by

Semere Kiros Bitew

Department of Computer Science

Specialization of Data Science & Smart Services

August 2018

Supervisors:
Dr. M. Theune
Dr. ir. D. Hiemstra
Dr. S. Petridis (Elsevier)

Faculty of Electrical Engineering,
Mathematics and Computer Science
University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands

University Of Twente

Masters Thesis

Logical Structure Extraction of Electronic Documents using Contextual Information

Author:
Semere Kiros Bitew
semerebitew@gmail.com

Supervisors:
Dr. Mariet Theune
Dr.ir. Djoerd Hiemstra
Dr. Sergios Petridis

A thesis submitted in fulfillment of requirements for the degree of Master of Science

in

Computer Science
Data Science and Smart Services

Faculty of Electrical Engineering, Mathematics and Computer Science

August 31, 2018

Acknowledgement

I am thankful to many people for their support, contribution and friendship, which made two years of my Master's studies at the University of Twente very exciting.

I would like to thank my supervisors Mariet Theune and Djoerd Hiemstra for their guidance, critical feedback and time reading (and re-reading) my earlier versions of this master's thesis. I would like to thank you both for the countless meetings, email communications and constructive feedbacks, which helped me develop a research mindset. Mariet, I am grateful for helping me contact Elsevier to work on this interesting topic.

Next, I want to thank Sergios Petridis from Elsevier for the day to day supervision. His support, motivation and above all giving me freedom to pursue my own ideas; greatly helped me get a better practical understanding of NLP and machine learning. I also want to thank my colleagues in the Apollo project at Elsevier; Timos, Yi, Ehsan and Arash that made my graduation project pleasant to do. In particular, I learned a lot from Timos, who has constantly explained me how different machine learning models work.

Finally, I am very thankful to my family and friends for the love, support and encouragement.

Abstract

Logical document structure extraction refers to the process of coupling the semantic meanings (logical labels) such as title, authors, affiliation, etc., to physical sections in a document. For example, in scientific papers the first paragraph is usually a title. Logical document structure extraction is a challenging natural language processing problem.

Elsevier, as one of the biggest scientific publishers in the world, is working on recovering logical structure from article submissions in its project called the Apollo project. The current process in this project requires the involvement of human annotators to make sure logical entities in articles are labeled with correct tags, such as title, abstract, heading, reference-item and so on. This process can be more efficient in producing correct tags and in providing high quality and consistent publishable article papers if it is automated. A lot of research has been done to automatically extract the logical structure of documents.

In this thesis, a document is defined as a sequence of paragraphs and recovering the labels for each paragraph yields the logical structure of a document. For this purpose, we proposed a novel approach that combines random forests with conditional random fields (RF-CRFs) and long short-term memory with CRFs (LSTM-CRFs). Two variants of CRFs called linear-chain CRFs (LCRFs) and dynamic CRFs (DCRFs) are used in both of the proposed approaches. These approaches consider the label information of surrounding paragraphs when classifying paragraphs.

Three categories of features namely, textual, linguistic and markup features are extracted to build the RF-CRF models. A word embedding is used as an input to build the LSTM-CRF models.

Our models were evaluated for extracting reference-items on Elsevier's Apollo dataset of 146,333 paragraphs. Our results show that LSTM-CRF models trained on the dataset outperform the RF-CRF models and existing approaches. We show that the LSTM component efficiently uses past feature inputs within a paragraph. The CRF component is able to exploit the contextual information using the tag information of surrounding paragraphs.

It was observed that the feature categories are complementary. They produce the best performance when all the features are used. On the other hand, this manual feature extraction can be replaced with an LSTM, where no handcrafted features are used, achieving a better performance. Additionally, the inclusion of features generated for the previous and next paragraph as part of the feature vector for classifying the current paragraph improved the performance of all the models.

Contents

List of Figures

List of Tables

List Of Acronyms

CRF	Conditional Random Fields
CSV	Comma Separated Values
DCRF	Dynamic Conditional Random Fields
DT	Decision Tree
GRMM	Graphical Models for Mallet
LCRF	Linear-Chain Conditional Random Fields
LSTM	Long-short Term Memory
ML	Machine Learning
NB	Naive Bayes
PDF	Portable Document Format
PII	Publisher Item Identifier
POS	Part Of Speech
RF	Random Forest
RNN	Recurrent Neural Network
SSVM	Structural Support Vector Machine
SVM	Support Vector Machine
XML	Extensible Markup Language

1 Introduction

Electronic documents are increasingly becoming an important medium of communication between humans since the introduction of computers. Electronic documents are easy to distribute, search and store. The structure of electronic document can be explicitly represented into physical layout and logical structure. *Physical layout* refers to what physically exists on a document page when it is viewed. The perceived document may be composed of a paper and ink or bits and pixels, and these small units form larger structures that convey information to human readers (e.g. a page consists of some blocks, a block consists of some text line, a text line is composed of some words etc.). In contrast, *logical structure* refers to what the physical layout components actually are in terms of understanding how to interpret the contents in a page of a document. For example, a set of pixels may form a paragraph, title and section which are logical components. Extracting such kind of logical structural information can be very useful in many applications. For example, such structures which enable documents to be easily reedited and restyled help build citation-indexing systems that greatly improve indexing, retrieving and reuse of document content. While it is easy for humans to understand the structure of such documents, computers are only good at processing content without attaching any interpretation to it (Mao, Rosenfeld, and Kanungo 2003).

Logical document structure extraction refers to the process of extracting logical components such as title, abstract, affiliation etc. from documents. Logical structure extraction is not a trivial process. For instance, authors use different styles and typesetting while writing papers, which results in completely different structures. Moreover, research articles from different fields of study tend to have different structures. For example, research articles from computer science and social sciences have completely different structure.

In this master's thesis the focus is on studying the problem domain, defining what is being solved, exploring the existing research and methods, and implementing machine learning models to achieve a high performance in extracting logical document structure. The logical document structure extraction is formulated as a sequential learning problem. Sequential learning refers to learning a sequence of

labels for a sequence of input data (Jurafsky 2000). For example, a document can be viewed as a sequence of paragraphs, sentences or words. By assigning labels (classes) to these paragraphs, sentences or words the structure of a document can be recovered.

1.1 Introduction to Elsevier and Apollo project

Elsevier[1], established in 1880, is one of the biggest scientific publishers in the world. Elsevier publishes approximately 400,000 papers annually in more than 2,500 journals. The company works on different innovative projects in a quest to become a leading information provider for the scientific and corporate research community.

One project called the Apollo project at Elsevier is working on recovering logical structure from article submissions. This project focuses on automating the process between first article submission of an author's manuscripts and final structured publishable form with high quality. Starting from the author's manuscript, it undergoes a series of editing steps that guarantees a high-quality publication. A typical work flow for revising the article includes preflight, auto-structuring and copy-editing stages. In preflight, initial submissions are checked for missing files. In the auto-structuring stage, rearranging of the content with specific structure is done. Finally in copy-editing, final revision of the content of the article is done before the publishable form is produced.

In the auto-structuring stage, human annotators are involved to make sure articles are labeled with correct tags, such as abstract, heading, title, reference-item and so on. This task can be more efficient in producing correct tags and thereby providing high quality and consistent data if automated. However, the automatic extraction of structure is not a trivial process. Some articles are submitted with semi-structured information and unstructured in the worst case.

The aim of this master's thesis is to automate the auto-structuring stage by utilizing existing machine learning models. With present developments of machine learning approaches, it is possible to automate the structuring process by replacing the role of humans achieving quite a good performance.

[1] https://www.elsevier.com/en-gb

1.2 Research Questions

The thesis work is an extension to the work done by He 2017. The author used textual and visual information to extract logical structure of documents. My thesis focuses on the use of models that take into account the contextual information to improve the task of document structure extraction. In our case, documents exhibit sequential structure at different levels such as paragraphs, sentences, words etc. Hence, contextual information refers to the information that can be exploited from surrounding paragraphs, words, sections to assist us in extracting structure from documents. The thesis addresses the following research question:

Given a word document (e.g. doc, docx format), how do machine learning approaches perform for the task of logical document structure extraction by leveraging the textual, linguistic, markup and contextual information?

To answer this research question it is important to consider the following subquestions:

1. *What kind of features should be extracted to apply machine learning algorithms to the task of logical structure extraction?* Feature selection is an important step towards developing high performing machine learning models. This research subquestion explores the set of features that should be extracted to achieve high performance in the task of logical document structure extraction.

2. *How to take into account the contextual information?* Different machine learning models take contextual information into account differently. This research subquestion will answer how contextual information is utilized for extracting the structure of documents.

3. *Given a context considering machine learning approach, what is the utility of linguistic, textual and markup features?* This research subquestion will answer the extent markup (visual), linguistic and textual features affect performance of machine learning models.

4. *Can the feature extraction step be left out by using alternative deep learning methods (Long short-term memory) for extracting document structure ele-*

ments? This subquestion will answer if the extraction and selection of manual handcrafted features, which is a requirement for traditional machine learning approaches, can be totally cut out by introducing deep learning methods.

1.3 Contribution

The main contributions of this thesis are grouped into categories. The first group lists the contribution from research point of view and the second group discusses from the company's (Elsevier) perspective.

From a research point of view, the contribution of the thesis can be summarized as follows:

1. We developed a novel approach that combines LSTM with CRFs, which outperforms existing approaches on Elsevier's Apollo dataset of 146,333 paragraphs for the purpose of logical structure extraction.
2. Two ways of using contextual information in building machine learning models for logical structure extraction are proposed, which can be of a good insight for other researchers.
3. The underlying learning code of GRMM (Graphical Models for Mallet) supports continuous features but the interfaces provided by the software only support binary features. Hence, a modification of a Java interface in the GRMM package of Mallet to include continuous features was done.
4. Several feature categories for the task of logical document structure extraction are proposed, which can act as insights for other researchers.

From Elsevier's perspective, automating the process of structuring articles with minimal human involvement will solve the following problems:

1. The manual work of annotating sections of an article is a routine task and susceptible to mistakes, which can lead to a drop of the quality of article publications.
2. There is inconsistency in the quality of data because article structuring is done by different people.

3. It costs time and money to manually annotate sections of articles and it is a bottleneck to the overall work flow of publishing articles.

1.4 Structure of the Thesis

The rest of the thesis is structured as follows: Chapter 2 presents the background and prior research works for sequential learning and logical document structure extraction. The data that is available in the Apollo project at Elsevier is described in Chapter 3. Chapter 4 discusses how the problem is formulated, the features used and theoretical working principles of the models used in the thesis. In Chapter 5 the experimental setting and results of the developed models are presented. Discussion of the results is presented in Chapter 6. Finally, the thesis is concluded by giving an outlook on future activities (Chapter 7) and conclusions.

2 Background & Related Work

Several researches have been carried out in document structure recovering and sequential learning. Sequential learning refers to a type of structured prediction of sequence of labels to a sequence of input data (Jurafsky 2000) . Document structure extraction refers to extracting structural elements from documents. This section starts with defining document structure extraction and logical document structure. Next, techniques for labeling sequential data are explored by discussing related works from other researchers and how they applied them.

2.1 Document Structure Recognition

Document structure recognition also known as document understanding is defined in different ways by different authors. Paaß and Konya (2011) define a document understanding system as a "system that produces a complete representation of a document's logical structure, ranging from semantically high-level components to lowest level components for a given text representation input." The output of a document understanding system may range from a representation useful for document sorting (e.g. terms and their frequency) to a structured message with attribute-value pairs describing all aspects of communication necessary to drive particular work-flows.

Document understanding can also be thought of as the exact reverse process of document authoring/production (Aiello et al. 2002; Bloechle 2010). Document authoring is the process of creating documents whereby authors start with a rough idea of the content they will write and then structure it by considering the logical organization of the material. For example, the author may decide to organize the document in chapters, decide its reading order and use some layout typesetting convention such as font size, indentation etc. Therefore, it is crucial to consider the choices the author makes while writing a document. How to include the choices made by authors such as font size, font style etc in developing document structure recognizing systems is discussed in section 4.2.

Document understanding may depend on readers' interpretation but documents generally have a physical layout and logical structure. Document structure

recognition exploits these two basic sources of information from documents. Physical layout is the layout of text on the printed page and logical layout contains the rich set of features of the wording and contents. The physical/geometric layout gives many clues about the relationship between different units such as headings, body text and figures. The logical layout can be exploited to recognize the existing interrelation and semantics of the text representation of the document (Rahman and Finin 2017; Mao, Rosenfeld, and Kanungo 2003; Dengel and Shafait 2014; Mehler et al. 2011; Klink, Dengel, and Kieninger 2000; Paaß and Konya 2011).

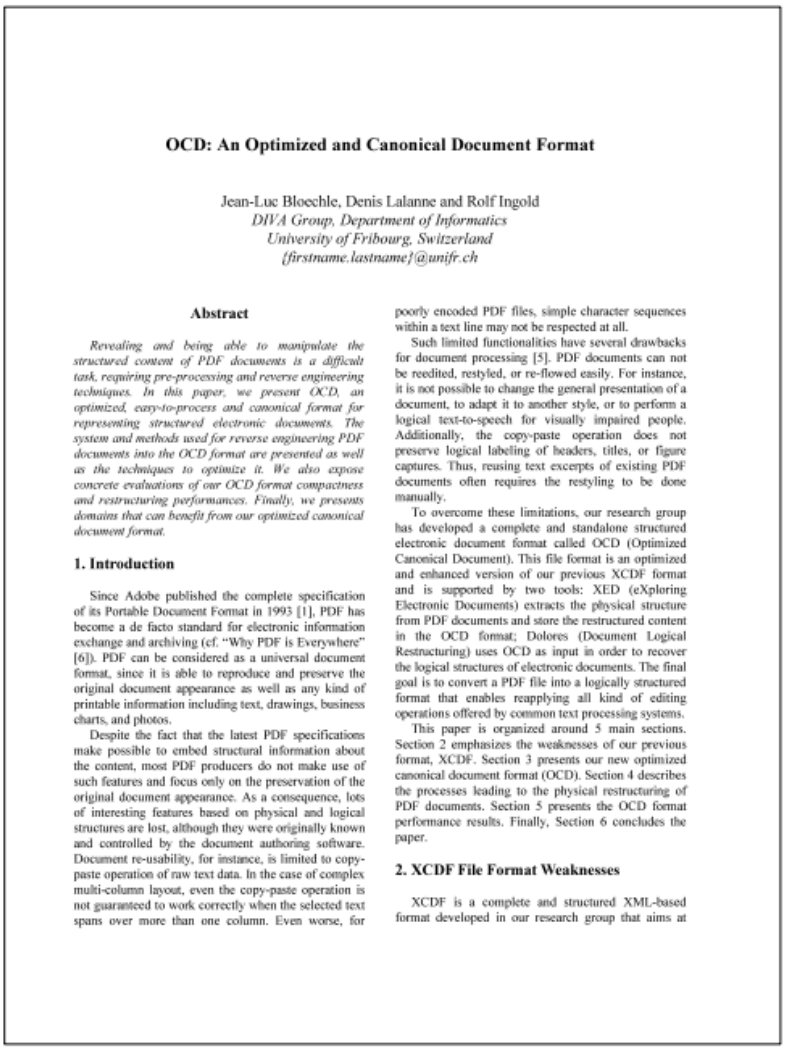
Extraction of the layout structure from scanned documents, which is called physical layout analysis (Geometric analysis or page segmentation) is further discussed in Klink, Dengel, and Kieninger 2000; Namboodiri and Jain 2007; Mao, Rosenfeld, and Kanungo 2003; Nagy 2000; Shafait, Keysers, and Breuel 2008. Classifying the layout structure into logical structure, which is called logical labeling, will be the point of interest in this thesis and hence will be discussed in the subsequent sections.

2.2 Logical Document Structure

The logical document structure couples the semantic meaning (logical label) to a physical section/zone of a document. It conveys the way information is structured in terms of logical units (e.g. title, figure, paragraph etc.) and how these units are interrelated with each other for example to determine the reading order. The relationships among the logical units are presented usually in the form of hierarchical include and sequence relations. For example, an article contains title, abstract, affiliation and a sequence of chapters, each of them including chapter title and sequence of section etc. The possible set of logical labels differs from document to document. For example: title, abstract, paragraph, section, table, figure and footnote are typical logical objects for technical papers whereas a business letter will have a sender, receiver, date, body, logical entities (Paaß and Konya 2011).

Logical structure is commonly represented as a hierarchical tree structure to reveal the relationships among the logical units of the document. Figure 2.2 shows an example tree structure of a paper given in Figure 2.1, borrowed from Bloechle

2010.



OCD: An Optimized and Canonical Document Format

Jean-Luc Bloechle, Denis Lalanne and Rolf Ingold
DIVA Group, Department of Informatics
University of Fribourg, Switzerland
{firstname.lastname}@unifr.ch

Abstract

Revealing and being able to manipulate the structured content of PDF documents is a difficult task, requiring pre-processing and reverse engineering techniques. In this paper, we present OCD, an optimized, easy-to-process and canonical format for representing structured electronic documents. The system and methods used for reverse engineering PDF documents into the OCD format are presented as well as the techniques to optimize it. We also expose concrete evaluations of our OCD format compactness and restructuring performances. Finally, we presents domains that can benefit from our optimized canonical document format.

1. Introduction

Since Adobe published the complete specification of its Portable Document Format in 1993 [1], PDF has become a de facto standard for electronic information exchange and archiving (cf. "Why PDF is Everywhere" [6]). PDF can be considered as a universal document format, since it is able to reproduce and preserve the original document appearance as well as any kind of printable information including text, drawings, business charts, and photos.

Despite the fact that the latest PDF specifications make possible to embed structural information about the content, most PDF producers do not make use of such features and focus only on the preservation of the original document appearance. As a consequence, lots of interesting features based on physical and logical structures are lost, although they were originally known and controlled by the document authoring software. Document re-usability, for instance, is limited to copy-paste operation of raw text data. In the case of complex multi-column layout, even the copy-paste operation is not guaranteed to work correctly when the selected text spans over more than one column. Even worse, for poorly encoded PDF files, simple character sequences within a text line may not be respected at all.

Such limited functionalities have several drawbacks for document processing [5]. PDF documents can not be reedited, restyled, or re-flowed easily. For instance, it is not possible to change the general presentation of a document, to adapt it to another style, or to perform a logical text-to-speech for visually impaired people. Additionally, the copy-paste operation does not preserve logical labeling of headers, titles, or figure captures. Thus, reusing text excerpts of existing PDF documents often requires the restyling to be done manually.

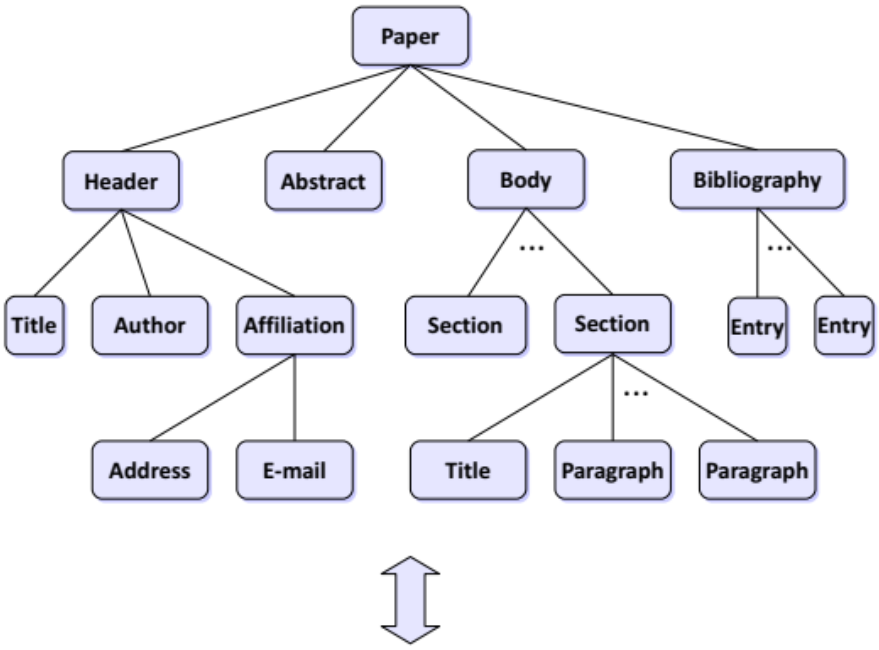
To overcome these limitations, our research group has developed a complete and standalone structured electronic document format called OCD (Optimized Canonical Document). This file format is an optimized and enhanced version of our previous XCDF format and is supported by two tools: XED (eXploring Electronic Documents) extracts the physical structure from PDF documents and store the restructured content in the OCD format; Dolores (Document Logical Restructuring) uses OCD as input in order to recover the logical structures of electronic documents. The final goal is to convert a PDF file into a logically structured format that enables reapplying all kind of editing operations offered by common text processing systems.

This paper is organized around 5 main sections. Section 2 emphasizes the weaknesses of our previous format, XCDF. Section 3 presents our new optimized canonical document format (OCD). Section 4 describes the processes leading to the physical restructuring of PDF documents. Section 5 presents the OCD format performance results. Finally, Section 6 concludes the paper.

2. XCDF File Format Weaknesses

XCDF is a complete and structured XML-based format developed in our research group that aims at

Figure 2.1: Sample first page of a paper (Source: Bloechle 2010)




```
<paper>
 <header>
  <title content="OCD: An Optimized and Canonical ..."/>
  <author> ... </author>
  <affiliation> ... </affiliation>
 </header>
 <abstract> ... </abstract>
 <body>
  <section>
   <title content="Introduction"/>
   <paragraph content="Since Adobe published the ..."/>
   ...
  </section>
  ...
 </body>
 <bibliography> ... </bibliography>
</paper>
```

Figure 2.2: Tree structure and formalized XML description (Bloechle 2010)

2.3 Techniques for Logical Structure Extraction

There are different approaches to extracting logical structure. Rule based approaches and machine learning approaches are discussed in the following sections.

2.3.1 Rule Based Approaches

Niyogi and Srihari 1995 developed a system called DeLoS which is knowledge-based to extract logical structure. This computational model is composed of a general rule-based control structure and a hierarchical multi-level knowledge representa-

tion. Rules governing document layouts are encoded to the domain knowledge base and the system has a global data structure to store document image data and incremental inferences. It has three types of rules: knowledge rules, control rules and strategy rules. The control rules manage the application of the knowledge rules and the strategy rules determine the use of control rules. The DeLoS system accepts images of documents, classifies them into blocks (segments) and finally produces the logical tree structure as an output. They tested their system on 44 binary images of newspaper pages and reported their results in terms of block classification, block grouping and read-ordering accuracy. The system achieved block classification accuracy of 88.65%.

Summers 1995 proposed an approach to automatically classify logical document structure based on the distance between the documents and predefined prototypes. The approach is composed of dividing a document into text zones and assigning logical labels to these zones. These logical structures are obtained by computing distance between a text zone and predefined prototypes. Distance between a prototype and an actual segment (text zone) is found by combining their attribute differences. Attributes include contours, successor, context, height, symbols and children. Shallow linguistic information is included to achieve higher accuracy. Nine randomly selected computer science technical reports which had 196 pages were used for testing. Summers reported performance that is greater than 86% accuracy.

All the above discussed approaches are rigid rule based methods. These methods cannot be extended to include more scenarios since all possible scenarios should be known before rules are formulated. We need to adopt solutions that will help us discover the rules from a variety of data. The rule based approaches are impractical nowadays due to the existence of big data which is semi-structured or unstructured and coming from different sources. As a result, in recent years research has shifted away towards machine learning approaches. In the next section, related work using machine learning approaches to logical document structure derivation is presented.

2.3.2 Machine Learning Approaches

In this subsection, previous related works which used machine learning models to solve their respective problems are discussed.

In the thesis by He 2017, different sections of scientific articles were extracted using machine learning models. He 2017 represented the problem of document structure extraction as a multi-class classification problem. He used six classes namely: "Title", "Author Group", "Affiliation", "Section Heading", "Table and figure caption" and "Reference". Basically, each paragraph in a document was classified as one of the classes. He used three categories of features to train his machine learning algorithms. The first group of features were visual features which included features like the percentage of bold, italic and underlined tokens in a paragraph. The second group called shallow textual features contained features such as the total number of commas, digits, words etc. found in a paragraph. The third group called syntactic textual features had features such as POS ratio (percentage of tokens marked with a certain POS tag) in a paragraph. He compared different machine learning models to extract the sections discussed above on Elsevier's Apollo dataset of 19,000 paragraphs. Random Forest classifier was reported to have achieved over all F1-score of 0.961. He further compared the contribution of each feature category to the performance of the Random Forest model and concluded shallow textual features have higher contributions.

Lopez 2009 developed a tool called GROBID (Generation of Bibliographic data) which is based on CRFs. GROBID works only on PDF documents. It extracts bibliographical data from scholar articles corresponding to the header information (title, author, abstract etc.) and to each reference-item (title, authors, journal title, etc.). It was trained on 1000 training examples for the header information and 1200 training examples for cited references. Both of these training examples are samples from a structured and hand curated published scientific articles. An evaluation was done on the publicly available CORA dataset. The results indicate an accuracy of 98.6% per header field and 74.9% per complete header instance, 95.7% per citation field and 78.9% per citation instance.

Beusekom et al. 2007 proposed an example based approach for labeling title

pages of scientific papers. The authors use "Title", "Abstract", "Affiliation" and "Author" as their logical labels. The method takes a new document page segmented into blocks as input and finds the best matching document from the set of labeled document layouts. The method is composed of two steps. First step is to find the best matching document from the training set to the unlabeled document. Second step involves assigning labels to the new unlabeled blocks of the document based on the best matching document layout. The similarity measures combined both geometrical layout and textual features at a block-level information. They applied their lightweight approach on a publicly available dataset called MARG database (1553 page images) and achieved accuracy rates ranging from 94.8% to 99.6%.

Zhang et al. 2010 proposed an approach to automatically extract bibliographic data from parsing references. They used the MEDLINE source of medical journal papers as their source of references. Their goal involved extracting 7 entities from a given reference, namely "Citation Number", "Author Names", "Article Title", "Journal title", "Volume", "Pagination", "Publication Year". The rest of the words in a reference were labeled as "Other". They extracted 14 binary features and one normalized position feature from each of the tokens they produced after preprocessing references. They applied CRF, Structural SVMs and SVMs to 2400 training references. Their performance shows above 98% accuracy for CRFs and Structural SVMs, and 95% for SVMs. They claim both CRFs and Structural SVMs high accuracy as the result of their strong sequence learning ability.

Rahman and Finin 2017 developed a framework to identify logical and semantic structure from large documents. Their system architecture basically consists of four units: pre-processing unit, annotation unit, classification unit and semantic annotation unit. The pre-processing unit takes PDF documents as input and produces CSV files where each row contains information about meta-data and text content for each line. The meta-data information includes font information, indentation, line spacing of each line etc. The annotation unit (Human Annotators) takes the layout and text information stored in the CSV file and annotates each line as a section-header or regular-text. The classification unit takes the annotated data and trains different classifiers to identify sections. It has sub units for line and section classification. The line sub unit outputs section-header or regular-text classes using Support Vector Machines (SVM), Decision Tree (DT), Naive Bayes (NB) and Recur-

rent Neural Networks (RNN). The section classifiers module takes the section-header as an input and classifies them as top-level, subsection or sub-subsection header using RNN. Finally the Semantic Annotation Unit labels each physically segmented section with a descriptive semantic name. It implements a topic modeling algorithm called Latent Dirichlet Allocation to get semantic concepts from each section and labels it with human understandable semantic concepts. It also produces a summary of a document and table of contents. The authors tested their methods on articles uploaded to the arXiv repository[2] in the period between 2000 to 2006. They reported their evaluation of their framework at line classification level and section classification level. They claim RNN performs better than SVM, DT and NB in line classification achieving an F1 score of 0.96. They also reported an F1 score of 0.81 for Section classification using RNN.

In the study by Tang et al. 2015, chemical entities embedded in the abstract of scientific biomedical articles were extracted. They used the BIO tags, a typical representation for named entities, where "B", "I", "O" denote beginning, inside and outside of an entity respectively. They used different features to train their machine learning models. Bag-of-words, Orthographical Information such as capital letters, Morphological Information (prefixes/suffixes), POS, Domain knowledge were among the features they used. They compared CRF (Conditional Random Field) and SSVM (structural SVM) on their sequence learning problem. They used 10,000 manually annotated abstracts from the CHEMDNER[3] corpus. The data was divided into a training set of 3,500 abstracts, a development set of 3,500 abstracts and a test set of 3000 abstracts. They report SSVM F-measure of 85.2% as compared to CRF's 85.05%. The authors also argue it is due to the higher recall of the SSVM-based systems that they outperformed the CRFs.

2.4 Conclusion

Summarizing, prior works show that a lot of research has been done on document structure extraction. We have seen that there are mainly two methods for extracting

[2]https://arxiv.org/

[3] CHEMDNER is a corpus of abstracts that contains chemical entity mentions labeled manually by expert chemistry literature curators. (https://mayoclinic.pure.elsevier.com/en/publications/the-chemdner-corpus-of-chemicals-and-drugs-and-its-annotation-pri)

logical document structure. The first one is the rule based approach where rigid predefined rules are used for extraction of entities from documents. It is impractical to formulate all the possible rules due to the existing of a lot of data with complex patterns. As a solution to this pitfall, a need to adopt solutions that will help us discover the rules from a variety of data is proposed, which leads to machine learning approaches.

Machine learning models unlike the rigid rule based methods, try to find patterns and rules automatically from available big training data, which can be semi-structured or unstructured and coming from different sources. The work by He 2017 and Lopez 2009 are two important prior works that applied machine learning methods to document structure extraction. He 2017 developed a system that classifies paragraphs of a document into different classes, but fails to consider the interdependence between consecutive paragraphs. On the other hand, GROBID, the tool developed by Lopez 2009 is trained using already published PDF documents and accepts PDFs as an input.

This thesis is different from these two prior works mainly for two reasons. First, it considers the fact that paragraphs occur in a sequential fashion and have dependence between each other. This sequential nature should be exploited in document structure extraction. Second, the thesis deals with article manuscripts that are not published yet. These documents are semi-structured or unstructured unlike the hand curated and well structured scientific articles used to build GROBID, which makes the problem challenging.

3 Elsevier's Apollo Dataset

In chapter two we have seen several publicly available datasets, which were used for the task of logical structure extraction. All the datasets are comprised of published papers, which are structured. The available dataset in Elsevier's Apollo project is, however, composed of raw semi-structured article manuscripts that are not published yet. This poses a greater challenge to the task of logical structure extraction. In this chapter, the available dataset in Apollo project at Elsevier is discussed.

It is known that available data plays a vital role in building machine learning models. As discussed in the first chapter, an author's article manuscript submitted to Elsevier goes through different phases before the final publishable document is produced. The Apollo project aims at innovating the process of how structured publishable documents are generated from raw article manuscripts. A typical work flow in the Apollo project at Elsevier is broken down into three stages namely preflight, auto-structuring and copy-editing. These phases each produce several versions of the article manuscript, which are called generations. Elsevier stores these different generations of a vast number of article manuscripts. Each article, which has many generations, is uniquely identified by an identifier called Publisher Item Identifier (PII). The use of the PIIs helps manage article manuscripts and provides a transparent way of looking into the changes introduced to the accepted article manuscripts during the several stages of the work flow. In the scope of the Apollo project, articles are collected from many fields and disciplines including the following journals:

- Carbon
- Journal of Molecular Structure
- Materials Chemistry and Physics
- Journal of Food Engineering
- Computers in Human Behavior
- Journal of Cleaner Production
- Renewable Energy

- Journal of African Earth Sciences
- Chemosphere
- Food Hydrocolloids
- Journal of Cereal Science
- Food Microbiology
- Superlattices and Microstructures

The list of journals covers quite a broad area and fields of studies. This variety of journals introduces different structures to documents and makes it challenging to recover logical structure elements. On the other hand, the existence of such a variety can be exploited to build robust machine learning models. In the scope of this thesis, total of 1000 PIIs are used as a dataset for building machine learning models.

Among the several generations, the initial article submission generation and the final published generation are of interest in this thesis. The Initial article submission is generated as an output of the preflight stage. The final published article is generated after the whole work flow is carried out. These two generations are specifically important because what the thesis intends to do is, given the initial submission, to extract the document structure and produce the final publishable article, bypassing all the generations produced by the human involvement in between. In the next sections these two datasets (generations) are discussed.

3.1 Article Manuscripts

In the preflight step, the author's initial article submission to Elsevier is checked for missing files through an iterative process of communication between the author and a clerk called Apollo Data Administrator (ADA). During these iterative communications several versions of the article manuscript are produced and recorded in Elsevier.

The preflight process culminates in a stable generation of the submitted article manuscript which contains all the required files and information. We will

refer to this stable generation as initial manuscript submission generation. There is no standard to how authors provide their articles and the type of the files they submit. For example, the main body of the article can be provided in a doc, docx or PDF format. Figures can be provided as separate image files or embedded within the Word file. In other cases, the title, affiliation and abstract can be provided in a different Word file. A typical folder which contains one initial manuscript submission is shown below in figure 3.1.

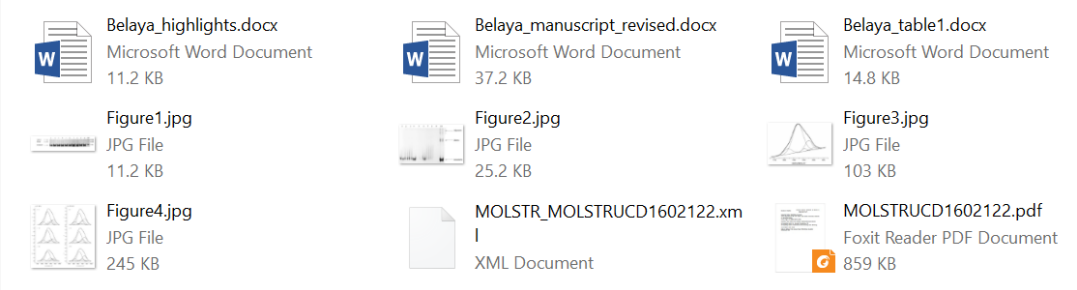


Figure 3.1: Initial manuscript submission with all contents and files

In the context of the thesis, document structure extraction from articles submitted to Elsevier in MS-WORD format are of highest importance. Word documents have both layout information and rich textual and visual mark up features that can facilitate extraction of structure. Moreover, the other files such as image files and PDFs are disregarded because roughly more than 95% of the total submissions are submitted in Word files.

3.2 Published Articles

In the auto-structuring stage, human annotators are involved in rearranging the content of the initial submission to a specific acceptable structure. They manually label each section of the article, which produces many versions of the submissions. The final stage of the process, which is the copy-editing stage, the final revision of the content of the article such as spelling checking is done. The published article generation is produced as the result of this final stage. Figure 3.2 shows one PII which contains all the image files, final publishable PDF and the corresponding XML file with all the meta-data of the article.

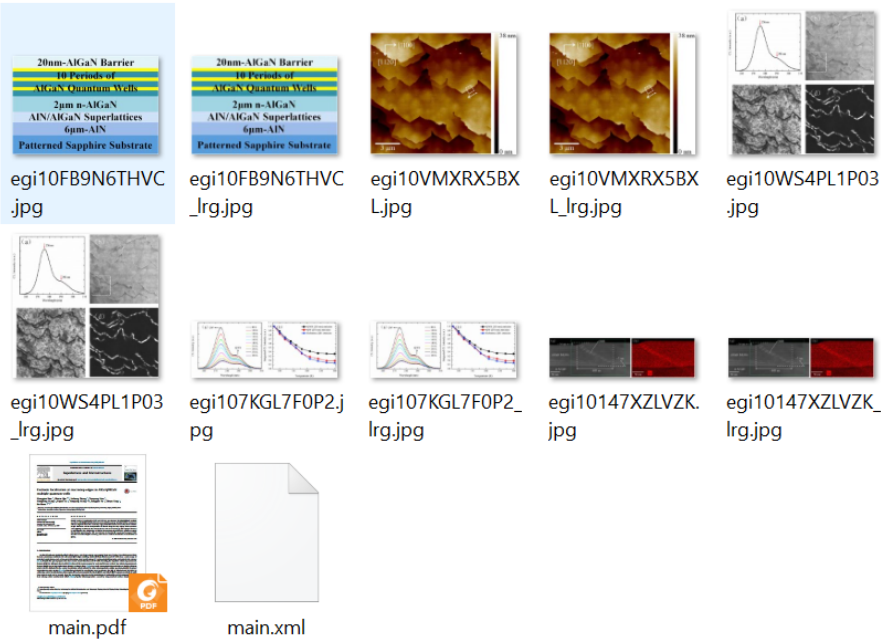


Figure 3.2: Final published article

The XML file stores all the article information in a tree like structure. It represents the hierarchical structure that exists in an article in terms of different entities and their relationship. For example, the "Head" element has child elements such as the "title", "affiliation","authors-group" and "keyword". The "Body" element contains elements like "section-title", and many "simple-paragraphs" since the body section is expected to have most of the content. The "Tail" element contains many "Reference-Item" elements within it. Figure 2.2 depicts a typical XML meta-data representation of an article.

To build supervised machine learning models, having a labeled dataset is obligatory. The labels in the published articles should be mapped into the initial article manuscript submissions. The Apollo team at Elsevier has done this transferring of labels which is called alignment task and produced an annotated dataset. The team produced a total of annotated 1000 PIIs that are used for our experiments.

3.3 Summary

Several versions of the same article submitted to Elsevier are produced, which are called generations. There are mainly two generations that are important to build

machine learning models in this thesis. The first one is the complete package of initial submission which contains all the required contents and files. There is no standard to the number of files submitted per author's article and the formats of these files. The second one is a final publishable article which exists as an XML format. This XML contains correct labels for each section of a given article in a tree like format.

4 Methods

In this chapter, we start by explaining how the problem of document structure extraction is modeled in the context of the thesis. The features used to develop our machine learning models are discussed next. Finally, a brief explanation of the theoretical working principles of Random Forest (section 4.3), Conditional Random Fields (section 4.4) and Long short-term memory (section 4.5) are presented. Random forest is chosen because it was the best performing model reported by He 2017, who used the same dataset (Elsevier's Apollo dataset) for the purpose of logical structure extraction. It serves as a baseline model. The choice behind CRFs lies on their suitability for this type of problem. They have been successfully applied by other researchers (Lopez 2009; Councill, Giles, and Kan 2008). LSTMs are selected because they don't require manual feature extraction. Moreover, they achieve state of the art performance in sequential learning problems (Huang, Xu, and Yu 2015). The details about these models are discussed at the end of this chapter.

4.1 Modeling the Problem

There are many different ways of describing a document structure as discussed in Chapter 2. He 2017 in his thesis defined a document as a list of paragraphs, where the order of the paragraphs is disregarded. In our context, a document is seen as a sequence of paragraphs. Other than in He 2017's research, the interdependence between paragraphs (context) in a given document is taken into account. He 2017 clearly leaves out the dependence between consecutive paragraphs while representing the problem of document structure extraction. In this thesis, the problem of logical document structure extraction is formulated as a sequential learning problem. For example, if the previous paragraph is classified as a "reference-item", it is unlikely that the current paragraph will be classified as "abstract" because "reference-items" are found at the end of a document while "abstract" is found at the beginning of a document. He 2017 does not take into account this unlikely transition from "reference-item" to "abstract" while classifying paragraphs into classes. In contrast, in this thesis we exploit the sequential structure of paragraphs by considering the possible transitions from one class to other.

Figure 4.1 shows the representation of a document as a sequence of paragraphs. Each paragraph can have multiple labels forming a hierarchical structure. The labels are categorized into levels. For example, in the figure there are two levels, lower and top level labels. The top level labels can be used to segment a document into "Head", "Body" and "Tail". The lower level labels can be used to classify paragraphs into structural elements such as "Title", "Abstract" ,"Reference-item" and so on. The labels at a given level are dependent with each other. For example, "Abstract" comes after "Title" in a document. The labels at different levels are also dependent on each other. For example, "Reference-item" is usually found at the last part of a document, in this case the "Tail" label as depicted in figure 4.1. Jancsary 2008 refers to these dependencies as the in-chain and between-chain dependencies respectively.

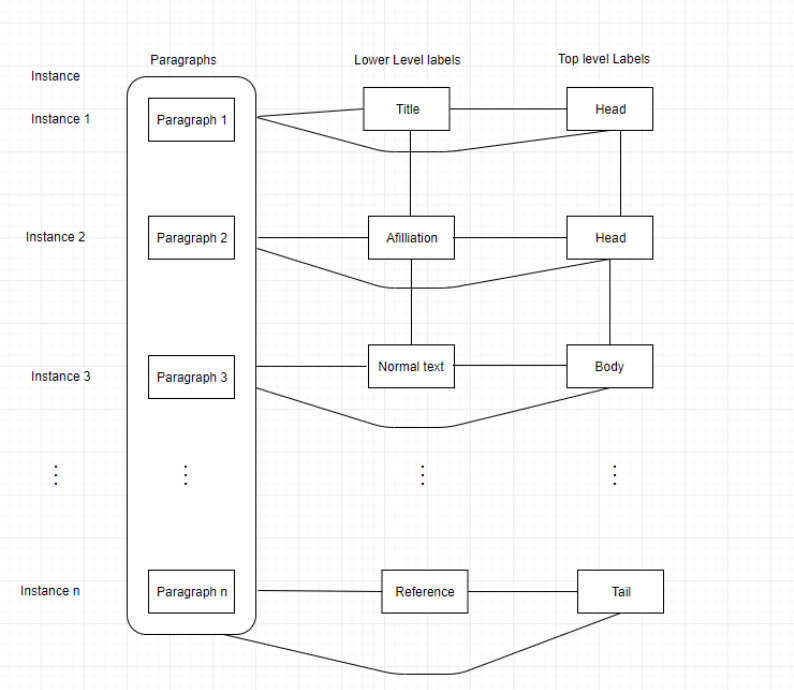


Figure 4.1: Sequence of paragraphs with possible multi-level labels

4.2 Feature Extraction

Feature extraction or feature generation is an important step in building high performance machine learning models such as RFs and CRFs. Features are extracted and fed to models in a way computers can understand. Features need to be provided for each instance (i.e paragraph instance in our case) of an article. These

features are expected to indicate as strongly as possible which class labels need to be assigned to which paragraphs. Random Forests and CRFs try to find out which features typically occur in conjunction with which class labels; this is the foundation for predicting class labels of unseen paragraphs in article documents.

A lot of feature extraction strategies and grouping feature sets into categories has been proposed and used by other researchers. For example, Jancsary 2008 used n-gram features, textual features, syntactic features, relative position features of entities to segment medical reports into different sections. Bloechle 2010 proposed two categories of features, self-related features and cross-related features in his work to recover structure from PDF documents. The self-related features are features such as the height of a paragraph, the number of tokens in a text-block, the percentage of spaces in a text-block. The cross-related features were features like the font-size of adjacent blocks, the position of the text-block in relation with the whole page etc. He 2017 proposed three categories of features, shallow textual, visual and syntactic features in his document structure extraction task. He used features such as total number of dots in a paragraph and if a paragraph starts with a capital letter from the shallow textual category. In the visual feature category, features like average font size of a paragraph and italic percentage of tokens in a paragraph were used. In the syntactic features category, noun percentage, verb percentage and adverb percentage of tokens in a paragraph were among the features he used. Inspired by these prior researchers we propose our own set of features. The extracted features are categorized into three groups; textual, markup and linguistic features as shown in table 4.1 along with a short description for each extracted feature.

- **Textual features:** these are features related to the statistical properties of the tokens inside a paragraph (e.g., count of tokens, percentage of commas from all tokens in a paragraph) and information that is obtained by a shallow checking of the appearance of tokens in a paragraph (e.g., if the first token of a paragraph is capitalized). For example, reference-items usually are part of an enumerated list of paragraphs. So, features such as *Begins_digit* and *Begins_bracket* can be important to encode such information.

  A new feature we came up with in this thesis is the *author_in_list* feature. For a non-numbered citation style, for every in-text citation in a paper, there must

Feature Name	Description
Textual Features	
Paragraph_length	The total number of tokens in a paragraph
Is_first_token_title	A binary feature indicating if the first letter of a paragraph is a capital letter
Is_first_token_capital	A binary feature indicating if the first token of a paragraph is all capitalized
Paragraph_len_norm	Normalized number of tokens in a paragraph in a given document
Begins_with_bracket	Binary feature indicating if a paragraph starts with a round bracket or a square bracket
Comma_per_paragraph	Percentage of commas from total number of tokens in a paragraph
Comma_doc_norm	The number of commas in a paragraph normalized by total commas in a document
Comma_count	Total count of commas in a paragraph
Dot_per_paragraph	Percentage of dots from total number of tokens in a paragraph
Dot_doc_norm	The number of dots in a paragraph normalized by total dots in a document
Dot_count	Total count of dots in a paragraph
Begins_with_digit	Binary feature indicating if a paragraph starts with a number
author_name_in_list	Binary feature that indicates if the any of the first ten tokens in a paragraph are found in a list of author names for specific document.
Mark up Features	
background-color	A normalized feature that indicates how different is the background-color of the paragraph with respect to the document
color	A normalized feature that indicates how different the color of the text in the paragraph with respect to the document
font-family	A normalized feature that indicates how different the font family of the text in the paragraph with respect to the document
font-size	A normalized feature that indicates how different the font size of the text in the paragraph with respect to the document
font-style	A normalized feature that indicates how different the font style (i.e italics) of the text in the paragraph with respect to the document
font-weight	A normalized feature that indicates how different the font weight of the text in the paragraph with respect to the document
text-align	A normalized feature that indicates how different the text alignment of the text in the paragraph with respect to the document
text-decoration	A normalized feature that indicates how different the text decoration (i.e underline, strike-through) of the text in the paragraph with respect to the document
Linguistic Features	
Noun_percentage	Percentage of noun tokens in a paragraph
Verb_percentage	Percentage of verb tokens in a paragraph
Adjective_percentage	Percentage of adjective tokens in a paragraph
Adverb_percentage	Percentage of adverb tokens in a paragraph
Conjunction_percentage	Percentage of conjunction tokens in a paragraph
Determiner_percentage	Percentage of determiner tokens in a paragraph

Table 4.1: List of Extracted Features

be a corresponding entry in the reference-items list. This feature is introduced by looking closely into how these reference-items are structured. Reference-items usually start with the names of the authors, then title of the article etc. The author names occurring at the beginning part of these reference-items usually are cited elsewhere in the body of the article. Hence, the *author_in_list* feature is a regular expression based feature that indicates if a paragraph contains an author name which is cited in the document. This regular expression first looks for the in-text citations of author names in the article. It produces a dictionary that contains all the author names cited in an article. This binary feature is activated if the first 10 tokens of a given paragraph contains an author name which already exists in the dictionary produced by the regular expression for an article.

Textual features proposed by He 2017 such as *comma_per_paragraph* and *dot_per_paragraph*, which are normalized by the total number of tokens found in a paragraph are also used. In addition to these features, we introduced new features such *comma_doc_norm* and *dot_doc_norm*, which are commas and dots found in a paragraph but are normalized at a document level. This way of normalizing gives an indication of how different a paragraph is in relation with the other paragraphs in that specific document.

- **Markup features:** Authors apply specific formatting styles to different entities of their document while editing their articles. To emphasize a part of text authors use bold, different font family, different font size etc. A typical example is the title of an article. The title is often bold, underlined and has bigger font size when compared to the rest of the paragraphs in an article. Another example is, journal/conference names found in reference-items can be italic. To accommodate such information markup features that are listed in table 4.1 are used in this thesis. All these visual markup features indicate how much emphasis is given to a paragraph by an author with respect to the whole document. The normalization of these features is done per document to reflect that the choices made by authors are specific to documents. The font size used for title and section headings can be consistent in a document but is usually inconsistent among documents (for example one author can use 12 points of font size for all section headings and another author can use same font size for

all the paragraphs) . To have a consistent feature representation throughout all the documents, we devised a method to compute the feature values as explained below.

Let us take the *Font-family* feature as an example. Let us assume a document is written 10% with **Times New Roman** and 90% with `Courier New`.

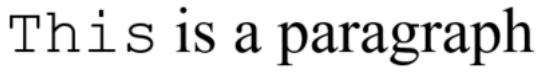


Figure 4.2: Font family feature example

The paragraph shown in figure 4.2 contains 16 characters where 4 characters are `Courier New` and 12 characters are **Times New Roman**. The *Font-family* feature value for the paragraph is calculated by taking the weighted average of the count of letters in each font-family subtracted from one.

$$\textbf{feature\_value} = 1 - \left[\frac{4 \times 0.9}{16} + \frac{12 \times 0.1}{16}\right] = 0.7 \tag{1}$$

The weighted average as shown in the parenthesis in equation 1 calculates how similar the paragraph is with the whole document. The feature value is then calculated by subtracting the weighted average from one to show how different the paragraph is (i.e emphasis given to the paragraph) from the whole document. Therefore, a feature value closer to one indicates high emphasis and a feature value closer to zero encodes lower emphasis is given to the paragraph.

- **Linguistic features:** these are features that provide possible syntactic categories of tokens in a paragraph. These features are introduced to encode information for families of tokens. For instance, one can argue that reference-items contain more noun phrases than the other document structure elements because there will be names of authors and journals, which convinces us that the POS tags contain useful information to help our machine learning models learn from the data.

The above discussed three categories are adopted from He 2017. The features in each category are a combination of features from prior works and new features we introduced. The normalization of features done per document, the regular expression

based *author_in_list* feature and the computation of the visual markup feature values are among the new features.

All the features are generated for each paragraph instance. However, the features generated for surrounding paragraphs can also be used to predict the class of the current paragraph instance. To include such an information from the neighboring paragraphs in classifying the current paragraph instance, the features generated for the previous and next paragraphs are used in this thesis. This implies that the feature vector used for each instance will be 3 times bigger since the features generated for the previous, current and next paragraph are concatenated. This combination of the features will be referred to as contextual features in the rest of the thesis.

4.3 Random Forest

Random forests are an ensemble learning method for classification. For classification tasks, random forests construct several decision trees at training time and produce the class label based on the majority decisions of these individual trees. A random forest classifier creates a set of decision trees from a randomly selected subset of the training set. It then aggregates the votes from different decision trees to decide the final class. Random forests are called random because they randomness in selecting the subset of training instances and also in selecting subset of features the algorithm will train on. Let's consider the task of binary classification of paragraphs into a reference-item or other paragraph. Let the matrix M in Figure 4.3 be a training data set which has a total number of N paragraph instances as rows and K number of features extracted from each paragraph as columns. The last column in the matrix shows the possible class labels for each paragraph (i.e for example reference-item or normal paragraph). To build a random forest from this matrix the following steps are done.

First, three parameters; the percentage of N paragraph instances, the percentage of K features and the total number of decision trees to be built are specified. Next, based on the first two parameters the random forest algorithm randomly selects a subset of the paragraph instances and a subset of the features to grow a

$$\mathbf{M} = \begin{bmatrix} f_{11} & f_{12} & f_{13} & \dots & f_{1K} & L_1 \\ f_{21} & f_{22} & f_{23} & \dots & f_{2K} & L_2 \\ \dots & \dots & \dots & \dots & \dots & \\ f_{N1} & f_{N2} & f_{N3} & \dots & f_{NK} & L_N \end{bmatrix}$$

Figure 4.3: Sample Dataset For Random Forest

decision tree. This process is done several times until the required number of decision trees have been built based on the third parameter.

Each of these built decision trees predict if a paragraph is a reference-item or normal text. Finally, a prediction is given based on the aggregation of the predictions from the individual grown decision trees.

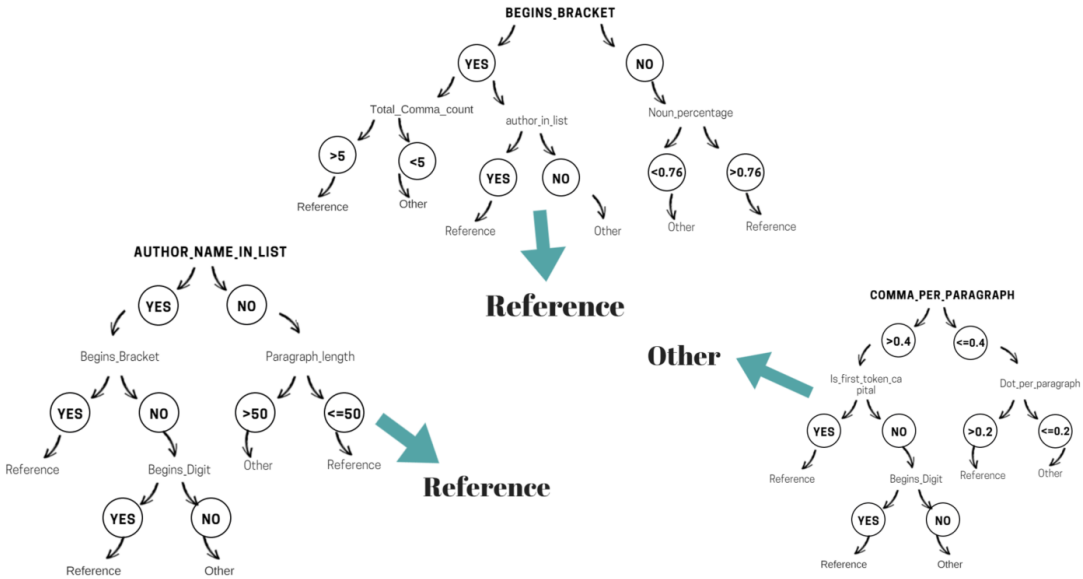


Figure 4.4: Example of Random Forest

Figure 4.4 shows three decision trees built on a randomly selected set of four features. Based on this random forest the class label prediction will be reference-item, since two of the grown decision trees predicted reference-item as the class label for the given paragraph instance.

Random Forest has been proven to be a robust and easy to use algorithm in classification tasks. It is easy to understand because the number of hyper-parameters are not that high and default parameters often produce a good prediction result, as shown by He 2017. Breiman 2001 in the original paper of Random Forests states that

the problem of overfitting, which is one big problem in machine learning, doesn't apply to Random Forest if there is enough data and enough trees are built. Since we have a considerable amount of data, overfitting will not be an issue in this work. It is important to note that random forests does not take into account the dependency that exists among sequential instances for example the dependency that exists between consecutive reference-items that are found at the end of a document.

4.4 Conditional Random Fields (CRF)

Conditional Random Fields (CRFs) were proposed as a framework for building probabilistic models to segment and label sequence data by Lafferty, Andrew McCallum, and Pereira 2001. To understand how CRFs work let us consider the problem of identifying the author and title from the text snippet in figure 4.5.

The new bestseller:
Tears of Love
by Paula Lowe

Figure 4.5: Text snippet from Paaß and Konya 2011

We can rewrite this text snippet in terms of words with their respective tags. For simplicity let us mark the words including newlines with T if they belong to a the title, with A if they belong to the authors, and with O if they don't belong to author name or title of the text snippet. This results with two vectors: x words and y labels as shown below in figure 4.6.

y	O	O	O	O	T	T	T	O	O	A	A
x	The	new	bestseller	\n	**Tears**	**of**	**Love**	\n	by	Paula	Lowe

Figure 4.6: Word and State Vectors from Paaß and Konya 2011

To infer the unknown labels one approach can be to use generative models. Generative models such as Hidden Markov models explicitly attempt to model the joint probability $p(x, y)$ over the input (words) and output (states) sequence. This approach explicitly models the interaction between the input features. Features of

a word we want to use can be like its identity, capitalization, prefixes, neighboring words, and category in semantic databases like WordNet[4]. Modeling these kind of input features leads to two important limitations. First, the dimensionality of x (input features) can be very large. Second, the features can have complex inter-dependencies. Constructing a probability distribution over such kind of features becomes very difficult. Modeling the dependencies can lead to intractable models and leaving them out can lead to performance reduction (Sutton, Andrew McCallum, and Rohanimanesh 2007).

Another approach is discriminative approach which directly models the conditional distributions $p(y|x)$ over the input and output sequences. This is the approach taken by CRFs. CRFs can model multivariate outputs with the ability to leverage a large number of input features used for prediction. The advantage of conditional models such as CRFs is that dependencies that involve only the input features play no role for prediction and hence are not modeled.

CRFs are represented by undirected graphs. In these undirected graphical models, nodes represent random variables and edges indicate the dependency between them. The dependencies between variables are expressed in the form of potential functions (factors). These factors are non-negative functions of the variables they are defined over. These factors are defined over cliques[5] in the graph.

Having introduced the general principle and advantage of CRFs in sequential learning, let us formally describe how $p(y|x)$ is defined by Lafferty, Andrew McCallum, and Pereira 2001.

Let **G** be undirected graph model over a set of random variables y and a fixed observed variable x. A CRF is then a conditional distribution p(y|x), where :

$$p(y|x) = \frac{1}{Z(x)} \prod_{\psi_A \epsilon G} \psi_A(y_A, x_A; \theta) \tag{2}$$

and the factors ψ_A of the undirected graph model **G** are parameterized as :

$$\psi_A(x_A, y_A; \theta) = exp(\sum_{k=1}^{K(A)} \lambda_{Ak} f_{Ak}(x_A, y_A)) \tag{3}$$

K(A) is the number of feature functions f_{Ak} defining factor ψ_A, and $\theta \epsilon \mathbb{R}^N = \{\lambda_{Ak}\}$ are real valued parameters of the CRF. The normalization function $Z(x)$ is defined

[4] https://wordnet.princeton.edu/

[5] A clique is a set of pairwise adjacent nodes in an undirected graph

as:

$$Z(x) = \sum_{y} \prod_{\psi_A \epsilon G} \psi_A(y_A, x_A; \theta) \tag{4}$$

and it sums over all possible assignments of y.

The factors above can be partitioned into set of clique templates $C = \{C_1, C_2 ... C_p\}$, where each of the clique templates C_p is a set of factors whose parameters $\theta_p \ \epsilon \ \mathbb{R}^N$ are tied (i.e. factors from the same template share the same parameters). Hence the equation to CRF can be rewritten as:

$$p(y|x) = \frac{1}{Z(x)} \prod_{C_p \epsilon C} \prod_{\psi_c \epsilon C_p} \psi_c(y_c, x_c; \theta_p) \tag{5}$$

and the normalization function Z(x) is defined accordingly as:

$$Z(x) = \sum_{y} \prod_{C_p \epsilon C} \prod_{\psi_c \epsilon C_p} \psi_c(y_c, x_c; \theta_p) \tag{6}$$

Let us further discuss from the formal definitions of CRF the important points that are related with the sequential labeling task this thesis intends to solve. The following points are inherited from comments stated in the thesis by Jancsary 2008 and apply to our case.

- In sequence labeling x is typically a sequence of input data. Here, the input data is sequence of paragraphs that forms a document.
- A separate random variable $y_i \ \epsilon \ y$ is linked with each node in the undirected graphical model G. The random variables are discrete for the purpose of sequential labeling since the outcomes of these variables correspond to the labels that can be assigned to the input sequences. If variables are adjacent in the graph G then there is a dependency between them.
- Factors ψ_c are defined over cliques c of G. These factors assign potential to variables of a clique. If G is a pairwise graph, there are only univariate or bivariate factors. Bivariate factors define potentials for the joint outcomes of the two vertex variables in a two node clique.
- The feature functions f_{ck} for a clique template C_p determine the value of variables of factor ψ_c over the clique c. Typically G has a repetitive structure and the parameters $\theta_p = \{\lambda_{pk}\}$ of each clique template are tied across time.

Feature functions are often binary which depend on the local context x_c and the variable assignment y_c of clique c. A typical example of a feature function can be a binary test that has value of 1 if and only if the previous label is "adjective", current label is "proper noun", and the current word begins with a capital letter. It can be defined as follows:

$$f_{ck}(y_c, x_c) = \begin{cases} 1 & if\ y_c = (\text{ADJ,NNP})\ and\ x_c = (\text{ begins with a capital letter , ...}) \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

Having introduced the general working principles of CRFs let us briefly describe the two types of CRFs that are used in this thesis.

Many families of CRFs are described in Andrew McCallum, Rohanimanesh, and Sutton 2003 in detail. The different families are identified based on the structure of the graphical model and the form in which the parameters are tied. Among these families, Linear CRFs and Dynamic CRFs are applied in this thesis and hence are discussed next.

4.4.1 Linear chain CRFs

Linear chain CRFs are one well known type of CRFs and are similar to HMMs. Linear chain CRFs have only one connected chain of labels where their parameters are tied across time. Factors are typically defined over single-node and two-node cliques. These factors capture the local probabilities and transition probabilities respectively. In section 4.1 we discussed the possible multi-level labels a sequence of paragraphs can have. The linear-chain CRFs have only one level of labels. For example, from figure 4.1 only the lower level labels are used to build a linear-chain CRF.

4.4.2 Dynamic CRFs

Dynamic CRFs are sequence models which allow multiple labels at each time step, rather than single labels as in linear-chain CRFs. Dynamic CRFs are generalizations of linear-chain CRFs. In this thesis, a two-level dynamic CRF for document structure

extraction is used. Each instance (i.e. paragraph) has two class labels. For example, the first category of labels identifies if a paragraph is a reference-item or not. The second category can further group paragraphs and identify if a paragraph is found in the first part of the document, middle part or the end of the document.

Dynamic CRFs learn the dependencies between the categories of labels rather than learning two different classifiers for each category. Ghamrawi and Andrew McCallum 2005 shows that improved classification performance is achieved using dynamic CRFs than using different classifiers for each category.

4.5 Long Short-Term Memory (LSTM)

Long Short-Term Memory (LSTM) is one variant of recurrent neural networks. To get better understanding of LSTMs let us first discuss artificial neural networks and recurrent neural networks. Artificial neural networks (ANNs) are weighted directed graphs in which artificial neurons are nodes and weighted directed edges are connections between neuron outputs and neuron inputs. Based on the architecture, ANNs are grouped into two types. (Jain, J. Mao, and Mohiuddin 1996)

- Feed-forward neural networks - graphs with no loop and neurons have unidirectional connections between them.
- Recurrent (Feedback) neural networks (RNNs) - loops occur because of feedback connections. These loops allow information to persist unlike Feed-forward networks.

In theory RNNs can be applied to connect contextual information from any time in the past (also known as long-term dependency) to a present task since their feedback loops allow information to persist. However in practice they suffer a problem called error back-flow problem. Error signals flowing backwards in time tend to either blow up or vanish. Before discussing the two types of error back-flow problems let's discuss how RNNs are trained. Training of RNNs is an iterative process. Both the inputs and outputs are provided to the network. The network processes the inputs and compares the actual outputs with the predicted outputs. The difference (error) between the actual and predicted outputs is then back propagated

through the network to adjust the weights, which control the network. Error gradient is the direction and magnitude calculated in training RNNs, which is used to update network weights in the right direction and by the right amount. Therefore, the exploding gradient problem is created, when repeatedly multiplying gradients through the layers of the network that have values larger than 1.0. This yields an unstable network. In contrast, vanishing gradients occur when the weight doesn't change. This stops networks from learning long-term dependencies. Consider a language model that predicts a next word given the previous ones. In a text "I grew up in the Netherlands ... I speak fluent *Dutch*."; to predict the last word in this case *Dutch*, the context of Netherlands is needed. When the gap between the relevant information and the point where it is needed grows RNNs become unable to learn the long-term dependency (Bengio, Simard, and Frasconi 1994).

Hochreiter and Schmidhuber 1997 introduced a variant of RNN called Long Short-Term Memory (LSTM) designed to overcome the error back-flow problems that exist in the traditional RNNs. LTSM can learn to bridge time intervals in excess of 1000 steps in input sequences without the loss of short time lag capabilities. This is achieved by enforcing constant (neither exploding nor vanishing) flow of error via the hidden states of the network.

Figure 4.7 shows the building block of one LSTM unit in an LSTM network. The network takes three inputs. X_t is the input of the current time step, $h_t - 1$ is the output of the previous LSTM unit and $C_t - 1$ is the memory of the previous unit. It has two outputs, h_t and C_t are the output and the memory of the current unit respectively.

The top pipe in the diagram is the memory pipe (also called memory state). The input is $C_t - 1$ which is old memory. The first cross ($\times$) it passes through is called the forget gate. This gate controls how much of the previous memory is needed. It is an element-wise multiplication operation. For example, if you multiply old memory $C_t - 1$ with a vector close to 0, most of the old memory will be forgotten. The next operation the memory flow goes through is the + operation. This operator sums up the old memory and the new memory. How much of the new memory should be merged is controlled by another gate called input gate, the $\times$ below the + sign. After these operations, new memory C_t is produced.

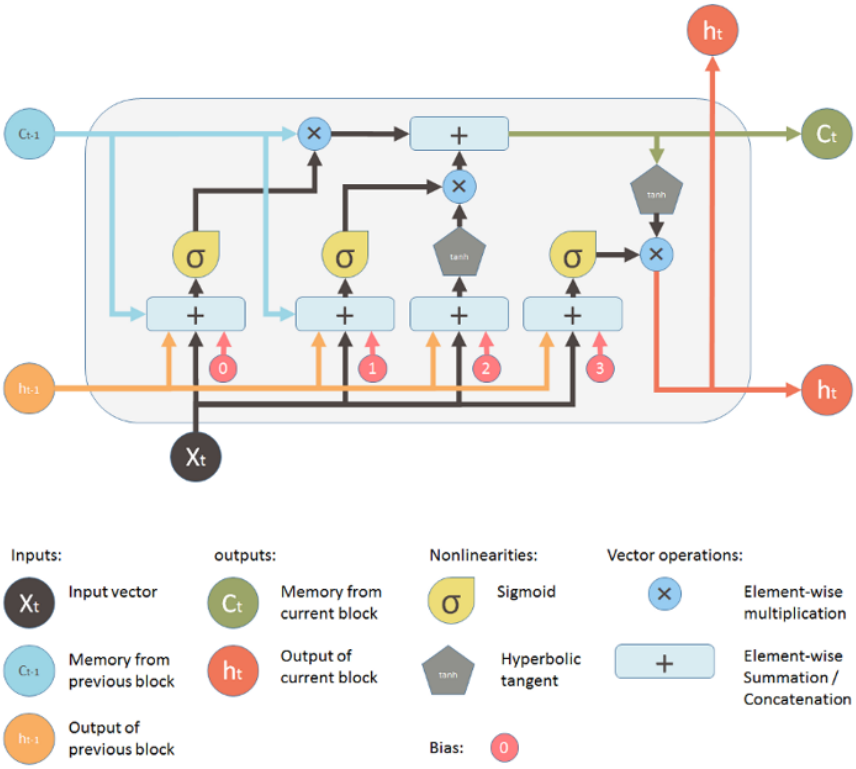


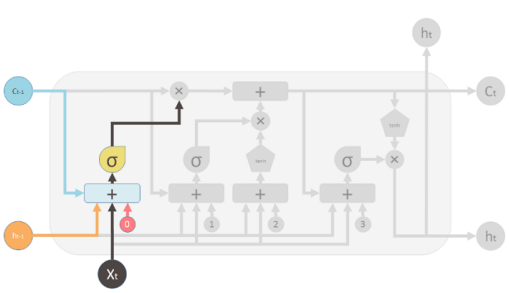
Figure 4.7: Architecture of one LSTM block [6]

Let's discuss the gates in detail. The forget gate shown in 4.8a is controlled by a simple one layer neural network and its inputs are the output of the previous LSTM block ($h_t - 1$), input for the current LSTM block (X_t), the memory from the previous block ($C_t - 1$), and a bias vector b_0. This neural network has a sigmoid function as activation and its output is the forget gate which is applied to the previous memory $C_t - 1$ by element-wise operation.

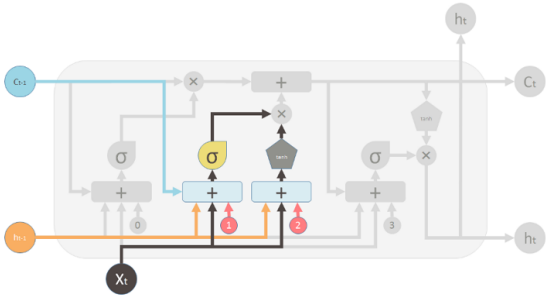
The next gate is called the input gate depicted in 4.8b. It is also controlled by a simple neural network that takes same inputs as the forget gate. This gate controls how much of the new memory should influence the old memory. The new memory however is generated by another neural network. It is a one layer network, but uses tanh as the activation function. The output of this network is element-wise multiplied with the output of the input gate and is added to the old memory to form the new memory.

Finally, output for this LSTM unit should be generated. This step has the output gate (shown in 4.8c) that is controlled by new memory, the previous output,

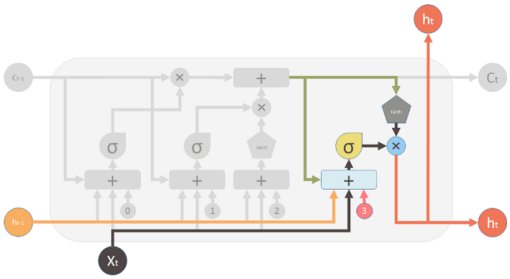
[6]Source: https://medium.com/mlreview/understanding-lstm-and-its-diagrams-37e2f46f1714



(a) Forget Gate



(b) Input gate



(c) Output gate

Figure 4.8: LSTM gates [7]

the input and bias vector. This gate controls how much new memory should be produced as an output to the next LSTM unit.

In this thesis, the LSTM based classifier is used to classify paragraphs into predefined categories such as "title", "abstract", "reference-item" etc . Paragraphs are seen as a sequence of words (tokens). The LSTM makes use of the interdependency that exists between the sequence of tokens in a paragraph. The use of LSTMs for sequential text classification into categories has been successfully applied by Hassan and Mahmood 2017; Huang, Xu, and Yu 2015. In addition, several researches have shown that feature extraction can be done jointly along with training RNNs such as LSTMs achieving a better performance when compared with the cumbersome manual feature extraction (Saeed 2017) that is required for traditional machine learning models such as CRFs and RFs.

4.6 Summary

Summarizing, the problem of logical document structure extraction is formulated as a sequential learning problem. In our context, a document is defined as a sequence

[7] Source: https://medium.com/mlreview/understanding-lstm-and-its-diagrams-37e2f46f1714

of paragraphs where there is an interdependence between consecutive paragraphs.

Three set of feature categories that are textual, linguistic and markup features are extracted. Textual features are related to the appearance and statistical properties of the tokens in a paragraph such as percentage of dots in a paragraph and the capitalization of a token. Markup features catch the information introduced by the formatting styles applied to paragraphs in a document by authors (for example font size and font-family). Linguistic features encode information for families of tokens in a paragraph (e.g. percentage of noun tokens in a paragraph).

The theoretical working principles of Random forests, CRFs and LSTMs are also discussed. Random forest are an ensemble learning method for classification. They build many decision trees by randomly using subsets of the training instances and the features to classify instances into classes. The final class of an instance is the aggregation of the decisions from the grown trees. This method does not take into account the sequential structure that exists in documents.

CRFs are probabilistic models to segment and label sequential data. Linear-chain CRFs (LCRFs) have one chain of connected labels. Dynamic CRFs (DCRFs) are generalizations of LCRFs. They can have multiple chains of connected labels. This feature of DCRFs allows for further introduction of context (structure) into sequential data, which in turn allows the use of multi-level labels unlike the LCRFs.

LSTMs are one variant of recurrent neural networks. They are designed to overcome the error back-flow problems that exist in traditional RNNs. LSTMs are successfully applied in sequential text classification achieving high classification accuracy, because they take into account long range dependencies among tokens. In addition, manual feature extraction is not required in training LSTMs unlike CRFs and RFs.

5 Experiments and Results

5.1 Experimental Setup

In this section, the design of the different experiments carried out is discussed. In addition, how the data was prepared, the performance measures used and the implementation setup is covered. The experiments ideally can be applied to all sections of an article document. One special case of document structure recognition is the reference-item structuring. In this section the experiments are done for extraction of the reference-items. The reference-items are selected mainly for two reasons. First, there are several reference-items in articles providing us with enough training samples. The second reason is that the consecutive nature that exists in reference-items list makes them ideal to experiment with sequence learning algorithms.

5.1.1 Dataset Preparation

The quality of the ground truth data plays an important role in building machine learning models. The dataset that is used for experiments is produced by the Apollo team and the quality depends on the alignment process carried out. The alignment task is the process of transferring labels of entities from structured final publishable XMLs into the initial manuscript submissions to produce an annotated dataset for training machine learning models. How the alignment process was done is out of the scope of this thesis. However, there are some errors that are introduced in the alignment process such as an article manuscript with all paragraphs tagged as "Normal_text". Mistakes such as this one were corrected by excluding submissions, which have all paragraphs tagged "Normal_text" from the corpus. This resulted a total of 1000 PII annotated submissions that are used in the experiments. As discussed in chapter 3, one PII usually contains many files such as image files, word files and PDFs. The word files are used for experiments. But again, usually there are multiple doc, or docx files for one PII. For a given PII, all the Word files in the directory are parsed and the one with most paragraphs is selected as the main file to train our models. A total of 146333 paragraphs were collected from these 1000 submission package PIIs.

Two class labels are used for extracting the reference-items of article documents, namely "Reference" and "Other" tags. All paragraphs other than the reference-items are tagged as "Other". The class label distribution is 32% reference-items to 68% other paragraphs as shown in 5.1.

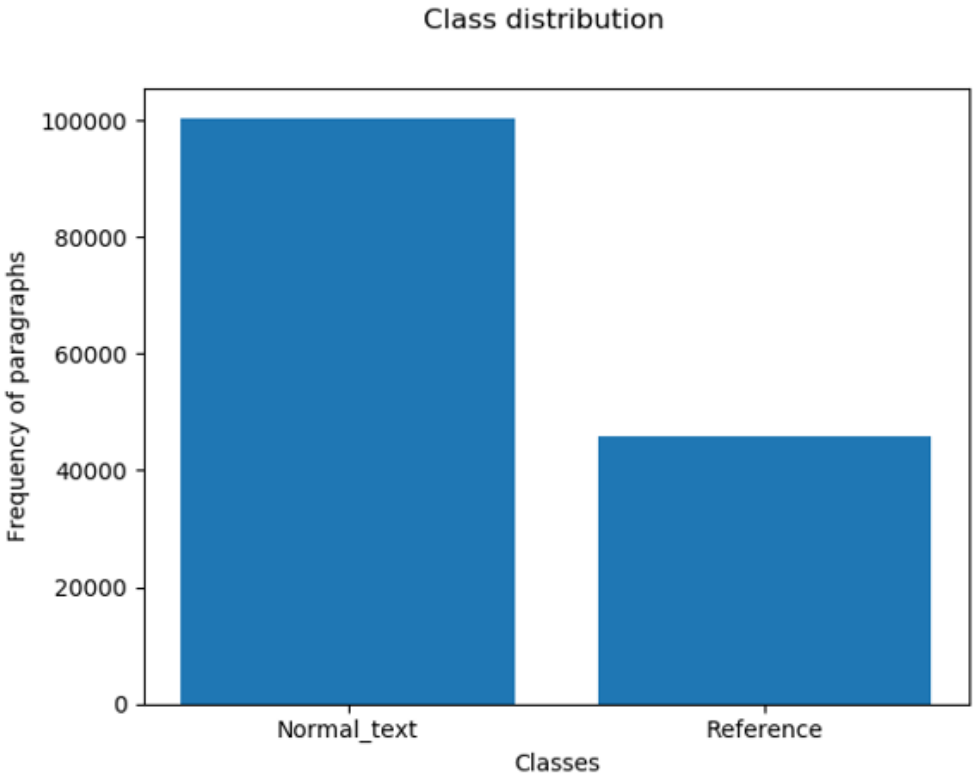


Figure 5.1: Class Label Distribution

The dataset was split randomly into 50% train set and 50% test set. Since the sequential nature of paragraphs in documents has to be preserved, the split was done on document level. It resulted in 500 PII submissions in each set. A two fold cross validation is used because the number of training instances is big enough. In this approach, the skill of the machine learning model on unseen data is estimated. Initially, the first fold is used as training set and the second fold is used as a test set. Then the folds are swapped, where the second field is used as a training set and the first fold is used as a testing set. Finally, the result of the two fold cross validation is summarized with the mean of the scores from the models.

5.1.2 Baselines

Our baselines are a random forest based binary classifier and an LSTM based binary classifier to identify if a paragraph is a reference-item or not. The random forest is selected because it was used by He 2017 in his thesis in extracting document structure and is a flexible, easy to use machine learning algorithm that produces, even without hyper-parameter tuning, a great result most of the time. It is also one of the most used algorithms, because of its simplicity. An LSTM based binary classifier is used because it does not require feature engineering and can learn informative features from data.

For the LSTM, word embedding was used as the input to the model. A word embedding is a class of approaches for representing words and documents using a dense vector representation. It is an improvement to the bag-of-words model encoding scheme where large sparse vectors were used to represent each word within a vector to represent an entire vocabulary. This representation is sparse because the vocabulary is vast and a given word would be represented by a large vector comprised of mostly zeros. In contrast, in word embeddings, words are represented by dense vectors where each vector represents the projection of the word into continuous vector space. The position of a word within the vector space is learned from text and is based on the words that surround the word when it is used. The position of a word in the learned vector space is referred to as its embedding. In building LSTM models (generally deep learning models) there are two ways to use word embeddings. One approach is to use pre-trained word embedding models such as GloVE[8] and Word2Vec[9] . The second approach is to learn word embedding while fitting the LSTMs. Learning word embeddings can be a slower approach, but tailors the model to a specific training dataset. In this experiment, the second approach was used since the dataset was specific to article manuscripts (Mikolov et al. 2013).

Both of these baseline models were considered to motivate the use of linear conditional random fields and dynamic conditional random fields for sequential learning. The examined baseline models do not take the sequential nature of documents into account. Every instance (which is a paragraph) is treated independently

[8] https://nlp.stanford.edu/projects/glove/
[9] https://en.wikipedia.org/wiki/Word2vec

with respect to the others, which is not true for documents since nearby paragraphs have a strong interdependence with each other. However, the LSTM classifier baseline model can be seen as an improvement over the random forest model since hand-crafted features are not provided to the LSTM and it is able to utilize the dependence between tokens local to each paragraph.

5.1.3 Experimental setup of Linear CRFs

Three experiments are carried out with respect to Linear CRFs.

1. **Linear CRF using direct features:** This experiment trains the Linear CRF directly on the extracted features listed in table 4.1. Fair comparison of a Random Forest and Linear CRF can be done since the same feature vector is used for both of the models.

2. **Random Forest followed by Linear CRF:** The first experiment does not consider the differences that can be possibly introduced due to how the feature vector is utilized by random forest and linear-chain CRF. To make it an even more fair comparison, and to see if considering context really adds to the performance of extracting document structure, we have designed this experiment. This experiment as in the first experiment is done to test the effect of taking into account the sequential nature of documents (i.e. sequence of paragraphs). First the Random Forest binary classifier is trained based on the handcrafted list of extracted features (Table 4.1) on the training dataset to predict if a paragraph is a reference-item or not. The Random Forest gives two probabilities as an output. The first probability is the probability of a paragraph being a reference-item and the second probability is the probability of a paragraph being other. These two probabilities sum up to one. These probability predictions of the Random Forest are used as the only two features to build the Linear CRF model.

3. **LSTM followed by Linear CRF:** This experiment also follows the same procedure as the previous experiment. The main difference is that manual feature extraction is not required to train the LSTM binary classifier. The LSTM during training learns important features by itself if provided with enough data.

The Linear CRF uses the two probability outputs of the LSTM as features to assign the sequence paragraphs a reference-item or other tag.

5.1.4 Experimental setup of Dynamic CRFs

The dynamic CRF unlike the linear-chain CRF allows us to use multi-level labels. The experimental design of Dynamic CRF slightly differs from the linear CRF because higher level labels were added to the dataset. In the experimental design of linear-chain CRFs we discussed that two labels (i.e. "Reference" and "Other") were considered that can be assigned to each paragraph. Here, we go one level higher. The new level labels were added to the dataset in the following manner. All the paragraphs tagged as "Other" before the first "Reference-item" tag were tagged with an extra higher level label called "Body". Second, all the paragraphs which have "Reference-item" tags were added with "Tail" label as a higher level tag. Any paragraphs that come after these "Reference-item" paragraphs were tagged as "AfterTail" label. Using this procedure a dataset the dynamic CRF can train on was prepared. Figure 5.2 depicts the multi-level labels used for building the dynamic CRF model. At the lowest level the sequence of paragraphs are shown. The next level in the hierarchy shows the possible labels that can be assigned to each paragraph. The highest level introduces another level of structure grouping the sequence of paragraphs into "Body", "Tail" and "AfterTail" sections. In section 4.1 we discussed the higher level labels to be "Head", "Body", "Reference" as a general labels that can be used for document structures. However, here the focus is on the "reference-items" structuring and we used the aforementioned labels to effectively represent the problem.

As with the linear CRFs, three experiments are carried out. The first experiment uses directly the computed features in table 4.1 to train the dynamic CRF. This experiment is designed to test if performance of extracting document elements is improved by introducing another level of structure.

The second experiment builds the dynamic CRF using probability predictions from random forest binary classifier as the only two features. Similarly as presented in the experimental design of linear-chain CRFs, a fair comparison between random forest and dynamic CRFs can be achieved by fixing how the feature vector is utilized.

The last experiment trains a dynamic CRF using output probability predictions from an LSTM based binary classifier.

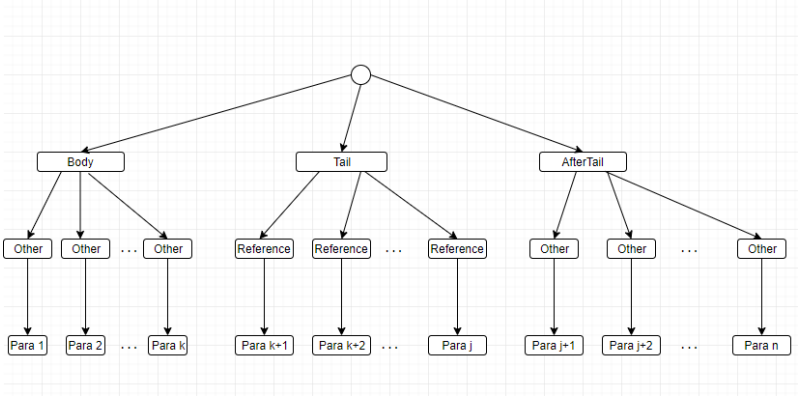


Figure 5.2: Two level labels for DCRF

5.1.5 Performance Measures

The classification correctness can be evaluated using the number of correctly predicted class instances (true positives), the number of correctly predicted instances that do not belong to the class (true negatives), the number of instances that were incorrectly assigned to the class (false positives) and the number of instances that were incorrectly not assigned to the class (false negatives). These counts form a confusion matrix (Sokolova and Lapalme 2009).

In this thesis, the performance of the models is evaluated using a widely used metric called F1-score which takes into account the four counts discussed above. F1-score is the harmonic mean of precision and recall. Precision is the percentage of true positives from all positive results returned by the classifier. Recall is the percentage of truly positive instances from all instances that should have been identified as positive by the model. The formulae for precision, recall and F1-score are given below:

$$\textbf{Precision} = \frac{tp}{tp + fp} \tag{8}$$

$$\textbf{Recall} = \frac{tp}{tp + fn} \tag{9}$$

$$\textbf{F1-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{10}$$

where tp, fp and fn denote true positive, false positive and false negative instances respectively for a given class.

5.1.6 Implementation Environment

The Python programming language is used to experiment with the different machine learning algorithms. Due to its active community and intuitive syntax it is widely used in scientific and research communities. It contains many built in libraries for AI and Machine learning such as scikit-learn. Since it is easy to experiment with new ideas and code prototypes and algorithms for ML it was used for implementing the data preparation, feature extraction and binary Random Forest classifier.

A Python deep learning library called Keras is used in experimenting with neural networks specifically LSTMs. Keras[10] is a high level neural network API capable of running on top of Tensorflow[11], CNTK[12] or Theano[13] back-ends. It was developed with a focus on going from idea to result with the least possible delay providing fast prototyping. As open source product of Google, Tensorflow is widely used by a lot of researchers and there is a huge support. Another aspect of using Tensorflow is that it can run on large scale servers. Hence it was used in the experiments.

A tool called GRMM[14] (Graphical models in Mallet) which is a Java based toolkit for performing inference and learning in graphical models of arbitrary structure is used to implement both linear and Dynamic CRFs (AK McCallum 2002). There are a of lot available packages for linear chain CRFs such as CRFSuite, CRF++ but not for dynamic-CRFs. We used GRMM because it has packages implemented for both the linear-chain CRFs and Dynamic CRFs. To make fair comparison of these two models it is required to use the same implementation. Even though the underlying Java code of GRMM supports continuous features, the

[10] https://keras.io/
[11] https://www.tensorflow.org/
[12] https://docs.microsoft.com/en-us/cognitive-toolkit/
[13] http://deeplearning.net/software/theano/
[14] http://mallet.cs.umass.edu/grmm/index.php

interfaces that are provided support only binary features. For this purpose, an interface in the GRMM package was modified to support continuous feature values since we have a lot of continuous features in our feature vector as it can be seen in table 4.1. The modified code can be found in the source code listing in Appendix B.1. The command-line interface GRMM provides accepts both the training data and testing data as an input, builds the model. In order to track the time taken for training and predicting, a separate interface Java code is written that predicts class labels from a trained model (See Appendix B.2).

5.2 Results

In this section, the results obtained from the carried out experiments are presented. The results are presented in terms of precision, recall and F1-score metrics taking the "Reference" class label as a positive class. Confusion matrices for best performing models in each category of experiments are also shown. First, the evaluation of the baselines is presented. Next, the results of several variants of Linear-chain CRFs and Dynamic CRFs are presented.

5.2.1 Baselines

The two baselines as discussed in section 5.1.2 are the random forest binary classifier and LSTM based binary classifier. Both baselines accept paragraphs and determine which class (i.e. "Normal Text" or "Reference") these paragraphs belong to.

Result 1: *LSTM based binary classifier performed better than the random forest binary classifier model.*

We first evaluate the baseline models as their classification performance will be compared with the proposed models in the subsequent sections. The 2-fold cross validation results for the random forest model is shown in table 5.1. This table shows the results obtained for each feature category and the combination of feature categories. The random forest model achieves the best F1-score of 0.931 with all the features. The confusion matrix for the random forest model with all the features is presented in table 5.2.

Features	**Precision**	**Recall**	**F1-Score**
Textual	0.96	0.897	0.928
Linguistic	0.96	0.877	0.862
Markup	0.778	0.108	0.190
Textual+Linguistic	0.967	0.885	0.924
All features	**0.967**	**0.897**	**0.931**

Table 5.1: Performance of Random Forest classifer

The second baseline (i.e. LSTM based binary classifier) shows an improvement over the random forest model with an F1-score of 0.964 as shown in table

Actual/Predicted	Normal Text	Reference
Normal Text	99055	1428
Reference	4698	41252

Table 5.2: Random Forest binary classifier confusion matrix with all features

5.3. The corresponding confusion matrix for this model is presented in table 5.4. This improvement suggests that LSTMs learn the interdependencies between the sequence of tokens within a paragraph.

	Precision	**Recall**	**F1-Score**
LSTM binary classifier	0.947	0.981	0.964

Table 5.3: Performance of LSTM binary classifier

Actual/Predicted	Normal Text	Reference
Normal Text	98009	2485
Reference	846	45104

Table 5.4: LSTM binary classifier confusion matrix

5.2.2 Linear Chain CRFs

Linear chain CRF with various experiments results are discussed in this section.

Result 2: *Linear Chain CRF with direct features performed better than the random forest binary classifier model.*

The linear chain CRF achieved the best F1-score of 0.955 with all the features as compared to random forest baseline model with 0.931 F1-score. This suggests that linear chain CRF, which takes into account the dependency between adjacent paragraphs by exploiting the tag information, outperforms a random forest model that neglects the interdependence between paragraphs while predicting class labels. Table 5.5 shows the scores achieved with the different feature categories.

The model achieved its best performance combining all the feature categories. This implies that the feature categories complement each other. The confusion matrix for the model with all the features is shown in table 5.6.

Result 3: *The random forest followed by linear CRF (RF-LCRF) model performed*

Features	**Precision**	**Recall**	**F1-Score**
Textual	0.969	0.936	0.952
Linguistic	0.860	0.760	0.807
Markup	0.733	0.594	0.656
Textual+Linguistic	0.969	0.938	0.954
All features	**0.969**	**0.940**	**0.955**

Table 5.5: Performance of Linear Chain CRF with direct features

Actual/Predicted	Normal Text	Reference
Normal Text	99125	1358
Reference	2753	43197

Table 5.6: LCRF confusion matrix

better than the random forest baseline model.

This experiment was introduced to make a fair comparison between random forest and linear chain CRF. Here we want to test the effect of considering contextual information that exists in sequence of paragraphs in a document while also fixing, how the features are utilized by two models, to be the same. This experiment achieves an F1-score of 0.944 compared to the random forest's F1-score of 0.931 as shown in 5.7. The confusion matrix for the best performing model with all features is presented in table 5.8.

Features	**Precision**	**Recall**	**F1-Score**
Textual	0.970	0.898	0.933
Linguistic	0.958	0.937	0.947
Markup	0.934	0.074	0.136
Textual+Linguistic	0.970	0.910	0.939
All features	**0.973**	**0.918**	**0.944**

Table 5.7: Performance of RF-LCRF

Actual/Predicted	Normal Text	Reference
Normal Text	99301	1182
Reference	371	42169

Table 5.8: RF-LCRF confusion matrix

Result 4: *The LSTM followed by linear CRF (LSTM-LCRF) model outperformed the LSTM based binary classifier baseline model.*

The LSTM-LCRF model achieved F1-score of 0.974 improving the baseline LSTM binary classifier F1-score of 0.964. This suggests that considering the tag information of the adjacent paragraphs (i.e. context) improves the performance of models for extracting document structure. The table 5.10 shows the confusion matrix for the LSTM-LCRF model.

	Precision	**Recall**	**F1-Score**
LSTM-CRF model	0.968	0.979	0.974

Table 5.9: Performance of LSTM-LCRF binary classifier

Actual/Predicted	Normal Text	Reference
Normal Text	99036	1458
Reference	939	45011

Table 5.10: LSTM-LCRF binary classifier confusion matrix

Result 5: *The LSTM-LCRF model outperformed the RF-LCRF model.*

The LSTM-LCRF with 0.974 F1-score outperforms the best performing variant of RF-CRF with F1-score of 0.944. This implies that the manual feature extraction can be replaced with an LSTM, where no handcrafted features are used, achieving a better performance.

5.2.3 Dynamic CRFs

The results obtained from the various Dynamic CRF experiments are discussed in this section.

Result 6: *The DCRF model with direct features outperformed the CRF with direct features model.*

Table 5.11 compares the performance of DCRF using the different feature categories. As in the previous results, the combination of all the features produces the best F1-score with 0.959. There is a slight improvement over the linear chain CRF which has F1-score of 0.955. This suggests that the introduction of a higher

level of structure (multi-level labels) indeed increases the performance of classifying paragraphs.

Features	**Precision**	**Recall**	**F1-Score**
Textual	0.965	0.941	0.954
Linguistic	0.877	0.783	0.827
Markup	0.767	0.632	0.693
Textual+Linguistic	**0.971**	0.939	0.955
All features	0.970	**0.952**	**0.961**

Table 5.11: Performance of Dynamic CRF with direct features

Actual/Predicted	Normal Text	Reference
Normal Text	99161	1349
Reference	2203	43720

Table 5.12: DCRF confusion matrix with all features

The confusion matrix for DCRF with all the features is given in table 5.12. The number of true positives (i.e. correctly predicted "Reference-items") raised to 43720 when compared to LCRF's 43197.

Result 7: *The RF-DCRF model outperformed the RF-LCRF model*

In this experimental setting, DCRF and LCRF are trained using the probability predictions of a random forest binary classifier as their only two features. The RF-DCRF model achieves an F1-score of 0.953 beating the F1-score 0.944 recorded by RF-LCRF. This improvement is result of introducing a higher level structure as discussed above. The best score is achieved by combining all the features as illustrated in table 5.13.

Features	**Precision**	**Recall**	**F1-Score**
Textual	0.974	0.916	0.945
Linguistic	0.961	**0.938**	0.950
Markup	0.969	0.089	0.164
Textual+Linguistic	0.974	0.927	0.951
All features	**0.976**	0.930	**0.953**

Table 5.13: Performance of RF-DCRF

The confusion matrix for the RF-DCRF is shown in table 5.14. The number

Actual/Predicted	Normal Text	Reference
Normal Text	102890	1339
Reference	3259	42691

Table 5.14: RF-DCRF confusion matrix

of correctly classified "Reference-items" increased by 522 paragraphs when compared with the RF-LCRF.

Result 8: *The LSTM-DCRF model outperformed the LSTM-LCRF model.*

The comparison of LSTM-LCRF and LSTM-DCRF is again to show that if a structure is added then the performance of extracting the document elements increases. Both the LCRF and DCRF are trained by two features. These features are the predictions for each paragraph instance produced by an LSTM binary classifier. The F1-score for the LSTM-DCRF as can be seen in table 5.15 is 0.982 when compared to LSTM-LCRF's F1-score of 0.974.

	Precision	**Recall**	**F1-Score**
LSTM-DCRF model	0.979	0.985	0.982

Table 5.15: Performance of LSTM-DCRF

Actual/Predicted	Normal Text	Reference
Normal Text	99499	984
Reference	676	45274

Table 5.16: LSTM-DCRF confusion matrix

The confusion matrix for the LSTM-DCRF is shown in table 5.16.

Result 9: *The LSTM-DCRF model outperformed the RF-DCRF model.*

A DCRF trained by the probability outputs of an LSTM outperforms a DCRF trained by the probability outputs of a random forest. This implies again that LSTMs capability to automatically learn highly discriminative features data as compared with the manual feature extraction used by RF.

5.2.4 The Effect of Using Contextual Features

So far, the comparisons were focused on the ability of models like linear chain CRF and dynamic CRF to consider the context by taking into account the tag information of surrounding paragraphs. Taking into account the dependency that occurs between nearby paragraphs is the success factor in the better performance of the sequential models as compared to Random Forest. In this section, we test the effect of using the features generated for surrounding paragraphs as part of the input feature vector while predicting class labels.

Model	F1-score without contextual features	F1-score with contextual features
RF	0.931	**0.954**
CRF	0.955	**0.968**
RF-LCRF	0.944	**0.955**
LSTM-LCRF	0.974	**0.975**
DCRF	0.961	**0.971**
RF-DCRF	0.953	**0.971**
LSTM-DCRF	0.982	**0.986**

Table 5.17: The effect of using contextual features on the performance of models

Result 10: *The inclusion of contextual features improves performance of document structure extraction*

The features for the previous paragraph and the next paragraph in addition to the features for the current paragraph are used to evaluate the effect of including contextual features. The goal was to see if performance can be improved for "Reference-item" extraction by including the contextual features. Table 5.17 illustrates the comparison of the models with and without the use of contextual features. Including the contextual features improved the F1-scores of all the models.

5.2.5 Conclusion

From the carried out experiments, it can be observed that the feature categories are complementary. Combining the feature categories improves the performance of the models. Considering the structure of a document also proved to improve the performance of extraction of elements. The general order of performance of docu-

ment structure extraction was found out to be DCRF, LCRF and RF respectively. RF doesn't consider the dependency that exists between consecutive paragraphs in a document accounting to its lowest F1-score when compared to linear-chain CRF and dynamic CRF. Linear-CRF as a sequential learning model improved the performance of RF. Dynamic CRF which adds another layer of structure at the top of the linear-chain CRF is able to achieve the best performance.

Another important observation from the experiments is that the LSTM can be used as an alternative to handcrafted feature extraction that is inherent in models like RF. LSTM's better classification accuracy is the result of learning the interdependency among tokens that are local to each paragraph. The use of LCRFs and DCRFs in conjunction with LSTM achieved the best performance. The inclusion of contextual features (i.e. features generated for the previous and next paragraph) improved the F1-scores of all the models.

6 Discussion

After the background research and carrying out several experiments, the results from the previous chapter and the challenges faced with implementing the models will be discussed here.

The use of context has shown to improve the overall "Reference-item" extraction from scientific articles. The random forest model, which is a robust and widely used model doesn't consider the dependency among paragraphs accounting to its lowest F1-score. Linear-chain CRFs on the other hand consider the dependency between consecutive paragraphs and thereby improve the F1-score achieved by random forest. Further introducing another level of hierarchy into the paragraphs (i.e structure or context) on the top of the linear-chain CRFs produced an even better extraction performance. Dynamic CRFs, which are generalizations of linear-chain CRFs are able to use multi-level labels (i.e. hierarchical tag information) and improved the F1-score of linear-chain CRFs. The more context (structure) was introduced the better extraction performance of document entities was achieved.

The best F1-score reported for "Reference-items" extraction in the work by He 2017 was 0.974. The best F1-score achieved in this thesis is 0.986 using LSTM in conjunction with DCRF. This reasonable F1-score is achieved because LSTM learns efficiently discriminative features local to each paragraph. The DCRF is able to learn the contextual information through the use of tag information of surrounding paragraphs.

The generated feature categories were textual, markup and linguistic features. The textual features were found to be crucial features when compared with the other two features, linguistic features being the next best. However, the combination of all these three categories produced the best performance, which means they complement each other.

The markup feature category for extracting the "Reference-items" showed a very low F1-score throughout all the experiments. As discussed in section 4.2 the computation of these features is done for the whole paragraph. However, if we look into the "Reference-items" closely, the markup visual effects are not applied to the paragraph as a whole by authors. One solution for this problem could be to divide

paragraphs into subparts and compute the markup features for these subparts. For example, the journal and conference names that are found in a bibliographic-items usually are italic and are found at end part of bibliographic-items. So, generating features for these subparts can be more informative. Another interesting observation of the markup features is that when used directly by DCRF and LCRF they achieved an F1-score of 0.65 and 0.69. But the F1-score of RF using markup features is as low as 0.19. This significant improvement of F1-score is the result of using context considering machine learning models.

The GRMM package of Mallet, which was used to implement LCRF and DCRF, is an excellent and very flexible toolkit written in Java. However, the flexibility comes at a price. The code makes generous use of runtime polymorphism. Runtime polymorphism refers to resolving which implementation of the same method to invoke during runtime. This runtime polymorphism basically degrades the performance as decisions are taken at runtime. The training time of DCRF was taking too much time (around one and half hours for 70,000 paragraph instances) and was a major challenge. For the small improvement in F1-score the DCRF gives over LCRFs is not worth using because it takes less than ten minutes to train the LCRFs on the same number of training instances. The memory requirements for GRMM is also quite substantial.

Initial investigation for extracting the head section, which includes "title", "author", "affiliation", "abstract", "keywords" and the tail section, which has "reference-items" from manuscript submissions was carried out using the LSTM-LCRF model. The F1-scores and the confusion matrix for LSTM-LCRF is depicted in Appendix A. Generally, the accuracy of extracting labels in the head section proved to achieve a lower performance when compared with extracting the reference-items. The first reason for this reduced performance is the data imbalance that exists between the distribution of the labels. The class distribution for the labels is shown in the bar chart A.1. The second reason is two elements do occur in the same paragraph. For example, an abstract and an affiliation co-occur in the same paragraph in many cases.

7 Conclusion and Future work

In this thesis, we illustrated mainly how the contextual information that exists in sequence of paragraphs in a document can be exploited to improve the overall performance of document structure extraction, particularly with extracting the reference-items that appear at the end of scientific articles. In the conclusion, how the research questions are answered is explained. The remaining challenges to be solved and directions for future work is discussed in the future work section.

7.1 Conclusion

The main research question was decomposed into 4 subquestions. Let us discuss how these subquestions were answered.

"What kind of features should be extracted to apply machine learning algorithms to the task of logical structure extraction?"

Three categories of features were selected to build our machine learning models; textual, linguistic and visual markup features. Each of the feature categories provide information from which our models can learn to extract the document structure extraction. As can be shown from the results, the extraction of "Reference-items" was achieved at a high accuracy without conducting a complicated feature analysis and selection.

"How to take into account the contextual information?"

To answer this subquestion, experiments were done in two settings. First, context considering sequential learning models (i.e that take into account the tag information of surrounding paragraphs) such as linear-CRFs and Dynamic CRFs were compared with a random forest that does not take contextual information into account. It was shown that linear-Chain CRFs and dynamic CRFs achieve better performance in extracting "Reference-items" from documents. They exploit the tag information at a paragraph level.

The second way of incorporating contextual information is using contextual features. In this thesis, the features extracted from the previous and the next

paragraph were used as contextual features. The inclusion of contextual features improved the performance of all the models.

"Given a context considering machine learning approach, what is the utility of linguistic, textual and markup features?"

It was consistently shown from the experiments that the order of importance of the category of features to be textual features, linguistic features and markup features respectively. Another important observation is that all the feature categories complement each other. The combination of all the features proved to give best performance in all the experiments.

"Can the feature extraction step be left out by using alternative deep learning methods (Long-Short term memory) for extracting document structure elements?"

Manual feature engineering is a requirement of traditional machine learning algorithms such as random forest and CRF. This process is sometimes cumbersome, time consuming and limited by the ability of the researcher to create discriminative features, something which requires extensive domain knowledge. Alternatively, we showed that by using a recurrent neural network architecture (in particular, an LSTM architecture) we can obtain even a superior performance to handcrafted feature extraction for document structure recognition. This shows that the capability of LSTM to learn highly discriminative features directly from data with no feature engineering. The conjunction of an LSTM architecture with CRF improves the overall performance of extracting document structure elements.

7.2 Future works

Reference-item structuring is a special case of document structure extraction, which was the core part in this thesis. However, these experiments should be extended to extract other elements of article manuscripts such as "Title", "Section headings","Affiliation", "Caption" etc. This means that not only word files should be considered but also PDFs and images. From the preliminary results it was indicated that extracting the likes of "authors" and "affiliation", which usually occur in the same paragraph, can lead to a reduced performance. One possible solution can be to build models that work on a at token level instead of at paragraph level.

It was shown that LSTM can be used to replace the manual feature engineering required in traditional machine learning models. It can be further enhanced by using Bi-directional LSTMs. Bi-directional LSTMs are a variant of LSTMs that can both use the future and past feature inputs unlike the LSTM that uses only the past feature inputs.

In the future, it may be a good approach to experiment with CRFs that make use of both manually generated handcrafted features and features generated by LSTMs.

Training DCRF in GRMM is very slow because the Java code makes generous use of runtime polymorphism and leaves a lot of room for micro optimization that can be handy for mass production. If the code can be optimized, the well written and organized software can be used for the future.

Appendices

A Preliminary results of LSTM-LCRF for the whole document

Sections	**F1-Score**
Title	0.818
Authors	0.713
Affiliations	0.864
Other	0.978
Abstract	0.764
Keywords	0.800
Reference-items	0.981

Table A.1: F1-scores for LSTM-LCRF

Actual/Predicted	Title	Authors	Affiliations	Other	Abstract	Keywords	Reference-items
Title	824	8	11	199	5	1	15
Authors	9	578	23	235	0	3	4
Affiliations	3	13	2052	287	26	10	4
Other	91	124	258	91378	224	56	975
Abstract	1	0	4	546	1316	6	1
Keywords	9	9	5	306	5	851	8
Reference-items	0	0	3	760	0	1	45186

Table A.2: Confusion Matrix for LSTM-LCRF

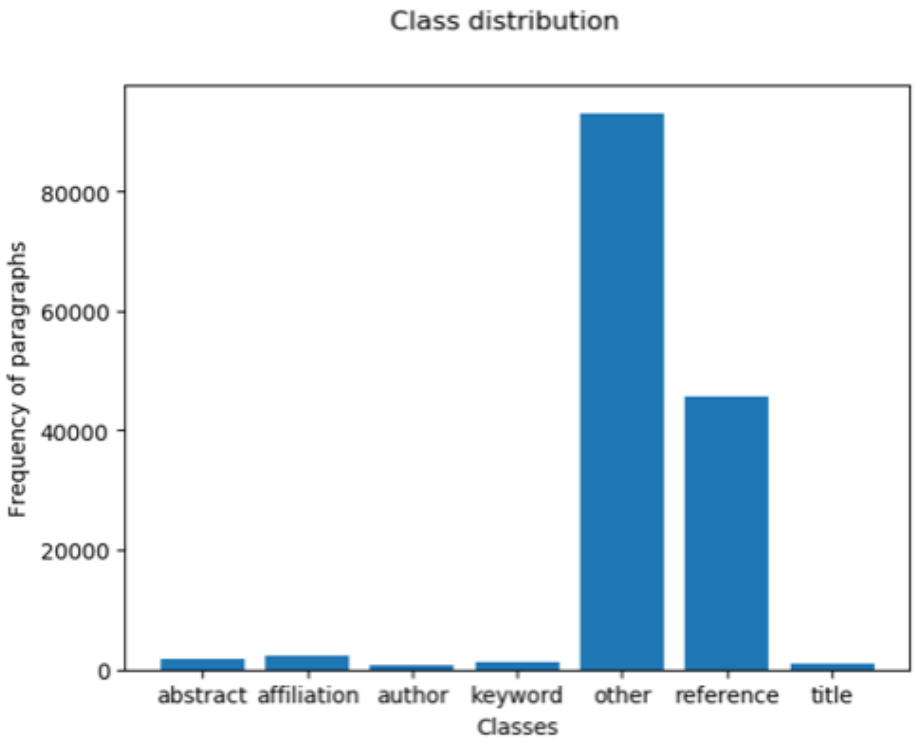


Figure A.1: Class Label Distribution For six classes

B Source Code Listings

B.1 Source code for including continuous features

```
/*
The GenericAcrfData2TokenSequence.java file is modified to to include the
    continuous features. The underlying code for GRMM supports continuous features
    but the interface only supports the binary features as stated in the GRMM
    homepage (http://mallet.cs.umass.edu/grmm/general_crfs.php).

Modified May 8,2018
by Semere Bitew
*/

/*Original code snippet from GenericAcrfData2TokenSequence.java that only supports
    binary features*/
while (j < maxFeatureIdx) {
        span.setFeatureValue (toks[j].intern (), 1.0);
        System.out.println(toks[j].intern());
        j++;
      }
```

```
/*New code that supports continuous features*/
//Added code to include continuous features separated by the token "="  in the
    GenericAcrfData2TokenSequence.java

      String featValueSeparator = "=";
        String[] featAndValue;

      while (j < maxFeatureIdx) {

          featAndValue = toks[j].split(featValueSeparator);
          span.setFeatureValue (featAndValue[0].intern(),Double.parseDouble(
              featAndValue[1]));
          System.out.println(featAndValue[1]);
          //System.out.println(maxFeatureIdx);
          j++;
      }
```

B.2 Source code for predicting class labels from a trained model

```
/*A new interface called PredictedFromSavedmodel.java. This interface is written
    for predicting labels based on a trained model.*/
package edu.umass.cs.mallet.grmm.learning;

import java.io.BufferedWriter;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.List;
import java.util.regex.Pattern;

import edu.umass.cs.mallet.base.pipe.iterator.LineGroupIterator;
import edu.umass.cs.mallet.base.pipe.iterator.PipeInputIterator;
import edu.umass.cs.mallet.base.types.InstanceList;
import edu.umass.cs.mallet.base.util.CommandOption;
import edu.umass.cs.mallet.base.util.FileUtils;
import edu.umass.cs.mallet.base.pipe.*;
import edu.umass.cs.mallet.grmm.learning.ACRF.Template;
import edu.umass.cs.mallet.grmm.learning.ACRFEvaluator;
import edu.umass.cs.mallet.grmm.learning.ACRFTrainer.LogEvaluator;

public class PredictFromSavedModel {
```

```
//private static CommandOption.File testFile = new CommandOption.File
  //        (GenericAcrfTui.class, "testing", "FILENAME", true, null, "File
      containing testing data.", null);

public static void main(String[] args) throws FileNotFoundException {
        // TODO Auto-generated method stub

         String testfile = args[1];
         String trainedmodel = args[0];
         PipeInputIterator testSource;
         testSource = new LineGroupIterator (new FileReader(testfile),
             Pattern.compile ("^\\s*$"), true);
        ACRF acrf = (ACRF) FileUtils.readObject(Paths.get("
            BOD_EOD_Lstm_CRF_head_tail_fold_one_500_train.ser.gz").toFile
            ());

         Pipe pipe = acrf.getInputPipe();
         InstanceList testing = new InstanceList (pipe);
         testing.add (testSource);
         List predictedLabels = acrf.getBestLabels(testing);
         ACRFTrainer acrfTr = new ACRFTrainer ();
         ACRFEvaluator myeval = new ACRFEvaluator() {

                @Override
                public void test(InstanceList gold, List returned, String
                    description) {
                        // TODO Auto-generated method stub


                }

                @Override
                public boolean evaluate(ACRF acrf, int iter, InstanceList
                    training, InstanceList validation, InstanceList testing
                    ) {
                        // TODO Auto-generated method stub
                        return false;
                }
        };
        acrfTr.test(acrf, testing, myeval);
        LogEvaluator lg_eval = new LogEvaluator ();
        acrf.print(System.out);
        lg_eval.test(acrf, testing, "TestingPrediction");


        try {
        PrintWriter out_new = new PrintWriter(new BufferedWriter(new
            FileWriter("C:\\Users\\bitews\\Documents\\grmm-0.1.3\\data\\
            grmm\\new_predictions_from_savedModel_200.txt", true)));
         out_new.println(predictedLabels);
```

```
                out_new.close();
              }
              catch (IOException e) {
                //do nothing
            }


        }

}
```

References

Aiello, Marco et al. (2002). "Document understanding for a broad class of documents". In: *International Journal on Document Analysis and Recognition* 5.1, pp. 1–16.

Bengio, Yoshua, Patrice Simard, and Paolo Frasconi (1994). "Learning long-term dependencies with gradient descent is difficult". In: *IEEE transactions on neural networks* 5.2, pp. 157–166.

Beusekom, J. v. et al. (2007). "Example-Based Logical Labeling of Document Title Page Images". In: *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*. Vol. 2, pp. 919–923. DOI: 10.1109/ICDAR.2007.4377049.

Bloechle, Jean-Luc (2010). "Physical and logical structure recognition of pdf documents". PhD thesis. University of Fribourg (Switzerland).

Breiman, Leo (2001). "Random forests". In: *Machine learning* 45.1, pp. 5–32.

Councill, Isaac G., C. Lee Giles, and Min-yen Kan (2008). "ParsCit: An open-source CRF reference string parsing package". In: *INTERNATIONAL LANGUAGE RESOURCES AND EVALUATION*. European Language Resources Association.

Dengel, Andreas and Faisal Shafait (2014). "Analysis of the logical layout of documents". In: *Handbook of Document Image Processing and Recognition*, pp. 177–222.

Ghamrawi, Nadia and Andrew McCallum (2005). "Collective multi-label classification". In: *Proceedings of the 14th ACM international conference on Information and knowledge management*. ACM, pp. 195–200.

Hassan, A. and A. Mahmood (2017). "Deep learning for sentence classification". In: *2017 IEEE Long Island Systems, Applications and Technology Conference (LISAT)*, pp. 1–5. DOI: 10.1109/LISAT.2017.8001979.

He, Yi (2017). *"Extracting document structure of a text with visual and textual cues"*. Masters thesis, University of Twente. URL: http://essay.utwente.nl/72979/.

Hochreiter, Sepp and Jürgen Schmidhuber (1997). "Long short-term memory". In: *Neural computation* 9.8, pp. 1735–1780.

Huang, Zhiheng, Wei Xu, and Kai Yu (2015). "Bidirectional LSTM-CRF Models for Sequence Tagging". In: *CoRR* abs/1508.01991. arXiv: 1508.01991. URL: http://arxiv.org/abs/1508.01991.

Jain, Anil K, Jianchang Mao, and K Moidin Mohiuddin (1996). "Artificial neural networks: A tutorial". In: *Computer* 29.3, pp. 31–44.

Jancsary, Jeremy (2008). "Recognizing structure in report transcripts". MA thesis. Faculty of Informatics, Vienna University of Technology.

Jurafsky, Dan (2000). *Speech & language processing*. Pearson Education India.

Klink, Stefan, Andreas Dengel, and Thomas Kieninger (2000). "Document Structure Analysis Based on Layout and Textual Features". In: *Proc. of International Workshop on Document Analysis Systems, DAS2000*. IAPR, pp. 99–111.

Lafferty, John D., Andrew McCallum, and Fernando C. N. Pereira (2001). "Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data". In: *Proceedings of the Eighteenth International Conference on Machine Learning*. ICML '01. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., pp. 282–289. ISBN: 1-55860-778-1. URL: http://dl.acm.org/citation.cfm?id=645530.655813.

Lopez, Patrice (2009). "GROBID: Combining automatic bibliographic data recognition and term extraction for scholarship publications". In: *International Conference on Theory and Practice of Digital Libraries*. Springer, pp. 473–474.

Mao, Azriel Rosenfeld, and Tapas Kanungo (2003). "Document structure analysis algorithms: a literature survey". In: *Proc. SPIE Electronic Imaging*. Vol. 5010. International Society for Optics and Photonics, pp. 197–208.

McCallum, AK (2002). "MALLET: A Machine Learning for Language Toolkit". In: *http://mallet.cs. umass. edu*.

McCallum, Andrew, Khashayar Rohanimanesh, and Charles Sutton (2003). "Dynamic conditional random fields for jointly labeling multiple sequences". In: *NIPS-2003 Workshop on Syntax, Semantics and Statistics*.

Mehler, Alexander et al. (2011). *Modeling, learning, and processing of text-technological data structures*. Vol. 370. Springer.

Mikolov, Tomas et al. (2013). "Distributed representations of words and phrases and their compositionality". In: *Advances in neural information processing systems*, pp. 3111–3119.

Nagy, George (2000). "Twenty years of document image analysis in PAMI". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22.1, pp. 38–62.

Namboodiri, Anoop M and Anil K Jain (2007). "Document Structure and Layout Analysis". In: *Digital Document Processing: Major Directions and Recent Advances*. Ed. by Bidyut B. Chaudhuri. London: Springer London, pp. 29–48. ISBN: 978-1-84628-726-8. DOI: 10.1007/978-1-84628-726-8_2. URL: https://doi.org/10.1007/978-1-84628-726-8_2.

Niyogi, Debashish and Sargur N Srihari (1995). "Knowledge-based derivation of document logical structure". In: *Document Analysis and Recognition, 1995., Proceedings of the Third International Conference on*. Vol. 1. IEEE, pp. 472–475.

Paaß, Gerhard and Iuliu Konya (2011). "Machine learning for document structure recognition". In: *Modeling, Learning, and Processing of Text Technological Data Structures*. Springer, pp. 221–247.

Rahman, Muhammad Mahbubur and Tim Finin (2017). "Understanding the Logical and Semantic Structure of Large Documents". In: *CoRR* abs/1709.00770. arXiv: 1709.00770. URL: http://arxiv.org/abs/1709.00770.

Saeed, Aaqib (2017). *Deep physiological arousal detection in a driving simulator*. URL: http://essay.utwente.nl/73268/.

Shafait, Faisal, Daniel Keysers, and Thomas M Breuel (2008). "Efficient implementation of local adaptive thresholding techniques using integral images". In: *Document Recognition and Retrieval XV*. Vol. 6815. International Society for Optics and Photonics, p. 681510.

Sokolova, Marina and Guy Lapalme (2009). "A systematic analysis of performance measures for classification tasks". In: *Information Processing & Management* 45.4, pp. 427–437.

Summers, K. (1995). "Near-wordless document structure classification". In: *Proceedings of 3rd International Conference on Document Analysis and Recognition*. Vol. 1, 462–465 vol.1. DOI: 10.1109/ICDAR.1995.599036.

Sutton, Charles, Andrew McCallum, and Khashayar Rohanimanesh (2007). "Dynamic conditional random fields: Factorized probabilistic models for labeling and segmenting sequence data". In: *Journal of Machine Learning Research* 8.Mar, pp. 693–723.

Tang, Buzhou et al. (2015). "A comparison of conditional random fields and structured support vector machines for chemical entity recognition in biomedical literature". In: *Journal of Cheminformatics* 7.S1, S8.

Zhang, X. et al. (2010). "A Structural SVM Approach for Reference Parsing". In: *2010 Ninth International Conference on Machine Learning and Applications*, pp. 479–484. DOI: 10.1109/ICMLA.2010.77.