Extracting Document Structure of a Text with Visual and Textual Cues

UNIVERSITY OF TWENTE.

ELSEVIER

Yi He

Supervisor: Dr. M. Theune
Dr. R. op den Akker

Advisor: Dr. S. Petridis (Elsevier)
Dr. M. A. Doornenbal (Elsevier)

Human Media Interaction Group
University of Twente

This dissertation is submitted for the degree of
Master of Science

July 2017

Acknowledgements

During the whole process of my internship and master thesis, I received lots of help from many people, and I have to deliver my thanks to all of them.

At first, I have to thank my supervisors Mariët Theune and Rieks op den Akker for all the help and supports from the University of Twente. I really appreciate the knowledge and experience they have delivered to me when I studied in the university. When I was doing my internship and thesis, I got a lot of inspiration through our regular meetings and email communications, which helped me a lot in my experiments. I am grateful to Mariët for helping me contact Elsevier for this interesting topic and revise my thesis.

Second, I have to thank Sergios Petridis and Marius Doornenbal, my advisors at Elsevier. Thank you for coordinating the project and introducing the background of it to me, so that I was able to get familiar with the content quickly and move further in my experiments. Also, I have to thank you for kindly sharing your knowledge with me. It was a pleasure that I could work with you and learn so many things from you, from both academic and industry sides.

At the end, I also need thank all other professors and colleagues at Twente and Elsevier. It wouldn't be possible for me to finish this work without your help. Thank you for all your support.

Abstract

Scientific papers, as important channels in the academic world, act as bridges to connect different researchers and help them exchange their ideas with each other. Given a paper or an article, it can be analyzed as a collection of words and figures in different hierarchies, which is also known as document structure. According to the logical document structure theory proposed by Scott et al. [42], a document instance is made of elements in different levels, like document, chapter, paragraph, etc.

Since it carries meta information of an article, which reveals the relationship between different elements, document structure is of significance and acts as the foundation for many applications. For instance, people can search articles in a more efficient way if keywords of all papers are extracted and grouped in a specific category. In another scenario, parsing bibliography items and extracting information of different entities is helpful to build a citation net in a certain domain.

As one of the major providers for scientific information in the world, Elsevier deals with more than 1 million papers each year, in various contexts and processes. In the case of Apollo project at Elsevier, given any manuscript submitted by authors, elements with several document structure types need to be extracted, such as title, author group, etc., so that papers can be subsequently revised and finally published. Current process of identifying different document structure entities involves a lot of human work and resources, which can be saved if it can be automated, and this is also the starting point of this thesis work.

Many researchers have put their efforts for applying machine learning models to extract document structure information from articles. Existing approaches are mostly based on some visual observables, such as font-size, bold style or position of the element in a single page, but few of them focus on making use of textual information with syntactic analysis involved. In other words, these approaches are very limited when there is little or even no visual markup information available. In addition, current approaches are mostly designed to extract document structure information from one single document. However, in the scenario of Apollo project at Elsevier, information in manuscripts are normally distributed in several files, and current approaches will also fail combining information from different files. Besides, those documents can be provided in diverse formats, and such a method is still missing which is able to collect information from files in more than one format.

The main goal of this research work is to investigate how textual features can help machine learning models identify the document structure information from manuscripts, including title, author group, affiliation, section heading, caption and reference list item. Another aim of this research is to find a method that combines distributed information from several files in the manuscripts.

In this research, we proposed a Structured Document Format (SDF), which is able to merge contents of both texts and images from different files in the same manuscript package, and our subsequent machine learning models also took input in this SDF format. Besides, since at the beginning we had no suitable data set for training and evaluating our models, we provided a solution to build our data set by aligning content between raw manuscripts and published articles. We hope this solution can also give inspirations to other researchers who are faced with similar problem. In our experiments, we compared the performance of our machine learning models trained with different features. With all kinds of features, we found our models perform generally good on the document structure extraction task, where the caption and reference extraction subtasks outperformed others. By comparing different sets of features in the document structure extraction task, we also found that textual-based features actually provided more information than visual features. They complement each other and taking them together improves the overall performance.

Furthermore, through our experiments, we identified some factors that are crucial to our results. At first, since different authors may apply diverse styles to organize their work, like the selection of font size for the title, it is important to align different styles so that they can be later fed to our models. Secondly, such a machine learning model can only be trained and evaluated properly when a proper way has been found to deal with the very unbalanced data set. Last but not least, for the document structure extraction task, we still have useful information unexplored from the manuscripts, for instance, the relative position in the document for an element, and it means our approaches still have a lot of room to be improved.

Table of contents

List of figures

List of tables

Chapter 1

Introduction

In this research, we explore the possibilities of applying machine learning approaches to automate the process of extracting document structure information from scientific article manuscripts. During the experiments, we investigate the effectiveness of diverse information and cues from manuscripts and compare their influence to the performance of our machine learning models. In the following sections, we first introduce the concept of document structure, after which we will give an introduction of Elsevier and its Apollo project that this external project is based on. Then we will give a short description of machine learning techniques and We conclude this chapter with elaborative research questions and challenges we have to overcome during the experiments.

1.1 Document structure and Elsevier

1.1.1 What is document structure

As a channel of communication, papers are crucial tools for people to exchange their ideas in the scientific world. When we find some texts or articles, we can treat them as a collection of words on one or several pages. Sometimes, words are used together with other conventional diagrams or pictures, in other cases, texts themselves are parts of graphical constituents such as titles, section headings, chapters, captions, etc. In this aspect, "text" itself has a strong graphical component [42, 49].

In the document structure theory from Scott et al. [42], basic hierarchy of document units are proposed. For instance, a hierarchy of six levels is mentioned, including text-phrase, text-clause, text-sentence, paragraph, section and chapter, which can also be visualized as Figure 1.1. Through the example, we can tell that those units of document structure are

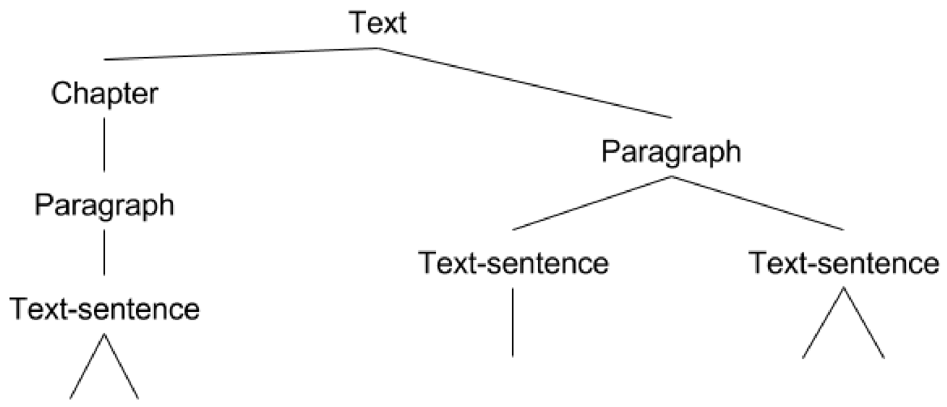


Fig. 1.1 An example of document structure hierarchy

ranked and aggregated with some rules like

$$Paragraph \rightarrow TextSentence^{+}$$

Though different documents may apply various hierarchy of categories to organize its content, for instance, some documents may introduce subsections or subsubsections but others may not, but there is still a certain hierarchy behind the document.

1.1.2 Introduction of Elsevier and Apollo project

Founded in 1880, Elsevier[1] functions as one of the major providers of scientific, technical and medical information in the world, which publishes approximately 400,000 papers annually in more than 2,500 journals. With this great amount of papers, Elsevier works on innovative methods to enhance authors' performance or empower them to make better decisions. As an external final project, this thesis is based on ongoing Apollo project at Elsevier, which aims at innovating the process how authors submit their articles and get feedback from editors.

Normally, after an author submits his manuscripts and the work is successfully accepted by Elsevier, there are still several steps to be finished to revise the paper and ensure that it meets the requirements to be published. In Elsevier's current work flow, those processes can be simply identified by three steps, pre-flight, auto-structuring and copy-editing, where the latter step always follows the previous one. To be specific, during the pre-flight process, the submission package is checked to ensure that authors have already submitted all the necessary documents, otherwise they will be contacted for further completion. In the subsequent auto-structuring step, content instances are extracted and marked as several entities based on their purposes, for example, "sections", "headings", "captions", etc. Normally, for a certain

[1]http://www.elsevier.nl

journal or conference, some specific layout format or presentation styles may be expected. Hence, after the previous two steps are finished, another process is conducted to make sure everything is converted in a correct way and the paper is publishable, and this step is called copy-editing.

1.1.3 Role of document structure

Document structure plays quite a significant role in the whole editing process. If the system misses some structure entities or fails identifying them, there is no way to complete the editing step of the paper and then the paper cannot be published. In addition, document structure also renders considerable instructions how an article should be printed out and published.

In Elsevier's current work flow, the auto-structuring process is still semi-automated. Some third-party suppliers and human labors are involved in the process to make sure those document structure types are identified correctly and then it is able to continue the next step. However, with present developments of machine learning methodology, we think it is possible to boost the automation of the structuring process or even replace humans' roles with quite good performance. From the perspective of a company, if the work flow is fully automated, a lot of extra cost can also be saved, and this is how the Apollo project started in Elsevier, which has the goal of automating the whole work flow mentioned above.

1.2 Machine learning

Defined as "giving computers the ability to learn without being explicitly programmed", machine learning is firstly proposed by Arthur Samuel [47]. Following the study of pattern recognition [2] and Artificial Intelligence [46], machine learning attracts lots of interests from researchers from various subfields of computer science. Consequently, an amount of works are done to enable computers identify further information behind data, or make independent responses to input data automatically.

Based on different types of data, machine learning approaches can be split into two groups, supervised learning where labels are given together with data, and unsupervised learning problems where no label is given. When it comes to scenarios where supervised machine learning can be applied, there are two types of problems: classification and regression analysis [8]. As indicated by the name, classification focuses on categorizing input instances to several groups. For instance, a typical classification problem can be determining a tumor either benignant or malignant based on scans of that tissue. On the contrary, a typical instance of regression problem can be predicting the price of a house based on the previous trading information in the same region.

With current research of machine learning theory, a number of industries are benefiting from all kinds of applications or subfields of machine learning. For example, development of computer vision [10] changes the way how we treat and perceive images. In other cases like texts, the efforts researchers put in natural language processing field also bring a number of innovations in this industry, like machine translation.

Since current machine learning models all learn solving regression or classification problems by making use of the patterns or information behind input data, training data then serves as the foundation of any model. In other words, the performance of a particular machine learning model is also constrained by the quality and availability of training data. As one of the major publishers in the scientific paper area, Elsevier has a huge database of earlier published papers that can act as training data and make the use of machine learning algorithms possible. As part of the Apollo project, the final thesis will focus on the part of automatically extracting logical document structures from manuscripts.

If we have a closer look at specific techniques of machine learning, a vast range of algorithms or models can be identified with their own characteristics. Machine learning models, including Support Vector Machines (SVM), Decision Trees (DT), Artificial Neural Networks based models, have proven a lot of successes in various domains, such as data analysis, computer vision [10], natural language processing [22], etc. Each category of machine learning models address problems in different approaches with their own advantages. In our experiments, we try several categories models to investigate their potentials for our task and they are introduced later in Chapter 4.

1.3 Research question and challenge

As we have introduced in previous sections, document structure plays quite a significant role in the journal publishing process and a lot of benefits and opportunities can be identified if this process is automated. From machine learning's aspect, document structure extraction is a typical information extraction task. Therefore, we aim to investigate the possibilities to address this information extraction task by machine learning models, and compare the effectiveness of different cues on this task, in particular for the title, author group, affiliation, section heading, caption and reference list item extraction.

1.3.1 Research questions

In our Apollo scenario, lots of information from manuscripts can be collected to help machine learning models make correct predictions, such as styles author apply to organize their work, the content itself or the element position in the document, etc. Those information may have diverse contribution to different subtasks. For instance, as the title is normally the first element in a document, the context location information may play a leading role, while the prefix "Fig" or "Tab" may often distinguish captions from other elements. In the scope of this work, we will focus on exploring the influence of visual markup information and textual content to build our machine learning models.

As article manuscripts are submitted by authors with different preference of organizing their work, systems relying heavily on the use of markup information may not be very robust. For example, systems designed to identify section heading entities with bold style may fail when authors apply little visual style information in their work or even provide plain text as manuscripts. Therefore, the main research question of this work can be summarized below, together with several subsequent subquestions:

> Given a plain text of content, where limited markup or style information can be expected and used, how can machine learning approaches perform on the task of extracting document structure from an article, specifically for the following aspects: title, author group, affiliation information, section heading, caption and reference item?

In order to carry out experiments and answer the above question, following subquestions are also crucial to consider:

1. For a single article manuscript, content can be provided with several documents in diverse formats like DOCX, PDF, JPG or even plain text. How can we collect information from all of them?
2. How can we create or find a data set from which we can train our machine learning models?
3. From all information fetched from manuscripts, what kinds of features or information can we use to solve our classification problems?

1.3.2 Challenges

To answer research questions listed above, there are several challenges we have to overcome.

At first, document instances can be provided in several formats or forms. Owing to the variety of documents, it is not easy to design a machine learning model that can take different formats of files as input. In other words, we need find a way to collect information spreading in different document instances and get the general document structure.

Secondly, such a data set is still missing which can perfectly represent the target we are dealing with. For tasks like document structure information extraction, well-labeled data set is indispensable if we want to benefit from supervised-learning algorithms and automate this structuring process with acceptable performance. Obviously it is not feasible to manually label all the entities and create the data set in the scope of this final project, but fortunately, Elsevier has a vast number of articles with all versions from first manuscript submission to final printed version. Yet we still need find a way to align those information and create a valid data set for our machine learning task.

Last but not least, in real cases, the data we can get is extremely unbalanced. If we take "article title extraction" as an example, for any one manuscript submission, only one article title is supposed to be found, which can be a positive instance for a machine learning model, and then any other paragraph or information is treated as negative sample in this case. This situation generally exists in all fields or document structure extraction task. Hence, the machine learning model should be robust enough to handle this issue.

1.4 Contribution

With this final project, we bring the following contributions:

1. An intermediate representation of manuscript packages which can cover all necessary information

2. An approach how we align the content from original manuscripts with well-formated papers and construct the data set for training our machine learning models

3. Several approaches to deal with unbalanced data in machine learning models and conduct error analysis to improve the performance

4. Several categories of features are proposed for our document structure extraction tasks, which can hopefully act as some insights for other researchers

1.5 Overview

In the following Chapter 2, we briefly introduce how document structure theory is developed and used in different scenarios by other researches. Subsequently, we will discuss how different methods are applied to handle document structure related tasks, including both template-based approaches and machine learning-based approaches. And then, in the subsequent chapter 3, we give a description of available data we have at Elsevier.

Our methodology and details of our experiments can be found in the Chapter 4, which covers features involved in our experiments and what kinds of models we use. In the Chapter 5, results of our experiments are recorded for both our binary classifiers and multi-class classifiers. In the last chapter, this thesis ends with a discussion of our results and future work of this research.

Chapter 2

Related Works

As one of the significant constituents of a document, document structure works as the foundation for many subsequent tasks or applications, like document digitalization where information should be extracted and organized from document images. In this chapter, we will at first introduce how document structure is defined by other researchers and how current tools provide supports for it. Then we continue introducing some literature reviews that have emphasis on extracting document structure information with different approaches. This chapter will end with a brief discussion of current methods.

2.1 Logical document structure theory

In the book *The Linguistics of Punctuation* [37] from Nunberg, two crucial clarifications are made to explain text structure and text-grammar behind the structure. First, text structure is distinguished from syntactic structure. Second, it also makes a distinction between the abstract concept of text structure elements and concrete form how they are represented [42]. To be specific, in linguistics, a sentence is often analyzed by parse trees such as $S \rightarrow NP + VP$, from which we can capture the structure between different constituents in terms of their syntactic relationships, and it is then called syntactic structure. However, when we treat a sentence as a collection of words starting from a letter and ending with a specific punctuation, we can end up with another structure, which Nunberg calls as text-structure, and the specific representation of different entity is interpreted as its concrete or graphical feature. Instead of analyzing the syntax from lexical view, with text-grammar, texts can be seen as a formation of several text elements with different levels, such as text-clause, text-sentence, section, etc. For instance, a structure rule as

$$S_t \rightarrow C_t^+$$

means that a text-sentence can be made of one or more text-clauses. In general, this relation can be illustrated in a mathematical form, where L_N represents the unit of level N (for example L_0 represents text-phrase, when L_1 means text-clause, etc.):

$$L_N \to L_{N-1}^{+} (N > 0)$$

Nunberg's text grammar is also a trigger of extensive discussion of document structure led by Scott and Power [42].

In the work by Summers et al. [53], logical structure trees are introduced to visualize the structure for scientific articles.

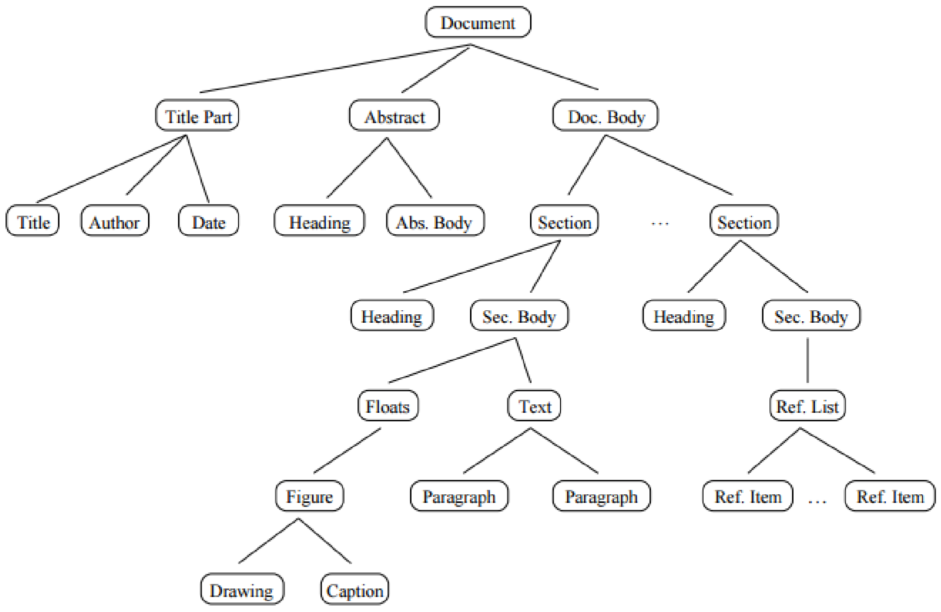


Fig. 2.1 A typical logical document structure tree for papers [53]

Ignoring the concrete content conveyed by a paper, through the example illustrated by Figure 2.1, we can identify that a paper can be treated as a combination of several abstract elements in various hierarchies based on a set of meronymy relationships between those constituents, which defines if one constituent is part of another one or not.

Having logical document structure as a fundamental information, a lot of tasks in the field of Natural Language Understanding(NLU) and Natural Language Generation(NLG) become feasible. For instance, a research has been done to extract the cross-references from scientific articles, such as footnotes, captions, references, etc. and link them together with their mentions in the same article [28]. In those works, document structure behind the texts acts as the fundamental information to make those tasks feasible. In other cases, since a lot of previous papers or literatures are kept in the form of images, it is quite important for people to digitize them and integrate them to our modern system or search database so that more people can make use of them. In this case, with logical document structure, those literature images can also be parsed with similar structure.

As a classical problem, a lot of researches have been done in this area in order to extract logical document structures. In the work from Summers et al. [53], given an article, four types of observables and cues have been proposed which can be applied to extract structure information. First, geometric observables, like the height of the line or the location of a certain element, are proposed. Apart from it, it is common that indented lists apply bullets as a mark of each item, and this type of information is summarized as marking observables. The other two are linguistic cues and contextual observables. In the former case, linguistic features like part of speech tags, punctuations or orthographic cases may be researched to find logical structures. Contextual cues can be split into local and global context-based observables. Local contextual cues only depend on a limited number of surrounding nodes, siblings or parents, while global observables make use of the context of the document as a whole. For instance, for a business letter, the return address and closing are both left-justified blocks across the page; however, it is easy to distinguish them by having a look at their previous neighborhood, because the return address should not be proceeded by any text but closing should. Similarly, by comparing presentation forms all around the paper, special paragraphs or elements can also be identified. These four different types of observables cues also give us inspirations about how we can form our own features to extract document structure information in our case, which are explained in detail in the following sections.

Generally, current systems or methods to extract document structure information from contexts can be mainly separated into two genres, template-based and machine learning model-based.

2.2 Document structure and rhetorical structure

There are also some comparisons between document structure and rhetorical structure, which is also commonly referred when a text is analyzed.

Developed as an approach of analyzing rhetorical organization of texts, rhetorical structure theory was firstly put forward by Mann and Thompson [30]. Based on this theory, a text can be analyzed by rhetorical function into some nested units, which are also nodes of an ordered tree. In a text, such an relation is defined as multinuclear if different constituents have equal importance. In some other cases, if an element is rhetorically subordinated to another one, then a set of roles including satellite and nucleus are defined.

Not like rhetorical structure, however, document structure does not analyze or explain a text based on their rhetorical organizations. Instead, it provides people with a view how a text is built, regardless of detailed content behind. For instance, sentences are grouped into paragraphs, paragraphs into sections, sections to chapters, and so forth. In other words, document structure, which is defined as a separate descriptive level in the texts analysis and generation system, mediates between the message underneath texts and their physical presentation from such graphical constituents such as headings, figures, references, etc.

These two different structure theories work together to provide people with a more complete analysis of text from both their syntactic aspect and graphical representation logic aspect.

2.3 Language support for document structure

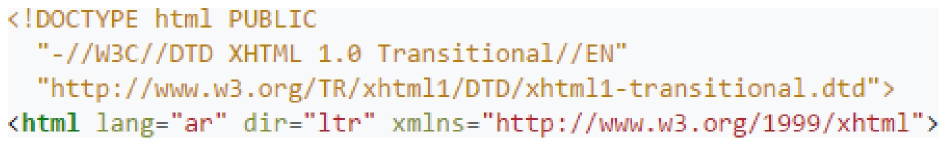
On the one hand, document structure is well defined by researchers including Nunberg, Power, etc., where the hierarchy relationships between elements are illustrated by several rules. On the other hand, from technical aspects, so as to keep those document structure information at binary files in computers, some tools are required to achieve this. Some programming languages do provide their supports in this aspect, such as Standard Generalized Markup Language (SGML) [14], Extensible Markup Language (XML) [3] or Hypertext Markup Language (HTML) [44].

As a standard for defining generalized markup languages for documents, SGML proposes two postulates [14]:

1. Markup should be declarative - it should focus on describing the structure of document rather than specifying the processes to be conducted on it.

2. Markup should be rigorous - available techniques for processing rigorously-defined targets can also make use of it

Two kinds of validity are defined for SGML, type-valid SGML and tag-valid SGML. For type-valid SGML, normally there is an associated document type declaration (DOCTYPE) associated with each document instance, while such a DOCTYPE is not required by tag-valid SGML, where a document instance can still be parsed without it. To illustrate, a DOCTYPE declaration is an instruction linking a SGML document with a particular document type definition (DTD), which is then used to parse the content of the document. Figure 2.2 shows an example of a DOCTYPE. Though a tag-valid SGML document can be parsed without DOCTYPE declared, however, it is useful if a user want to enforce some additional constrains on his documents.

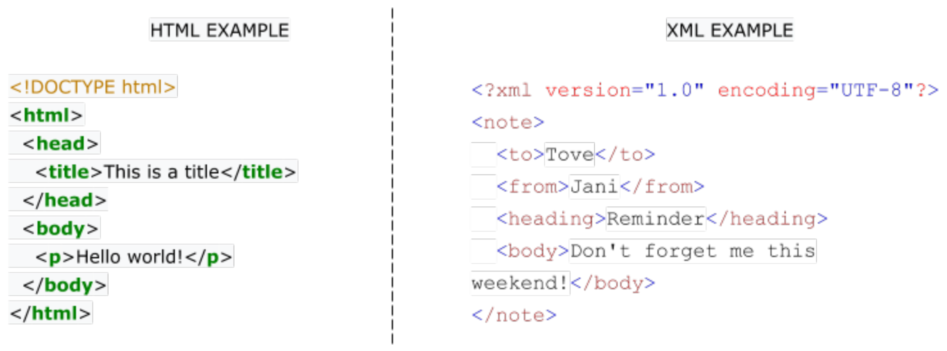


```
<!DOCTYPE html PUBLIC
  "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html lang="ar" dir="ltr" xmlns="http://www.w3.org/1999/xhtml">
```

Fig. 2.2 An example of document type declaration

From syntax point of view, a SGML document can include the following three components:

1. SGML declaration
2. A prologue which contains a DOCTYPE and other declaration entities making a Document Type Definition (DTD)
3. Document instance itself that contains several elements with different levels of hierarchy



HTML EXAMPLE

```
<!DOCTYPE html>
<html>
 <head>
  <title>This is a title</title>
 </head>
 <body>
  <p>Hello world!</p>
 </body>
</html>
```

XML EXAMPLE

```
<?xml version="1.0" encoding="UTF-8"?>
<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this
weekend!</body>
</note>
```

Fig. 2.3 An example of HTML file and XML file

As a standard for defining generalized markup languages, SGML provides guidelines how a language can be generated. HTML and XML, as two most common SGML-based languages, both have similar supports for recording the structure of document instance. Born as the standard markup language for creating web applications, HTML defines several types of HTML elements with their own purposes. For instance, content of a document instance is most likely to be surrounded by tag <p>...</p> as paragraph element. In other cases, when image or text field need to be embedded, tags like <img /> or <input /> will also be introduced. Working together with Cascading Style Sheets (CSS) and JavaScript, HTML provides a great way to organize the document structure and visualize it in browsers.

Similarly, for the purpose of being both human-readable and machine-readable, XML also defines a set rules for organizing documents. Not like HTML, which is mostly used in web applications, XML is used as more arbitrary data structures. Since XML is used in wider cases, there is no any universal schema for it, and people can create tag with any content as long as it makes sense in the specific use case. Hence, it normally contains a reference of a specific Document Type Definition (DTD) and its elements and attributes are normally declared in that DTD with some certain grammatical rules.

An example of HTML and XML file can both also be found from Figure 2.3.

Not only limited in HTML or XML, similar components to record document structure information can be found in other programming languages like LaTeX, which is also commonly used in documenting scientific articles.

2.4 Approaches to extract document structure

2.4.1 Template-based approaches

A template-based approach is sometimes used to solve tasks like citation structure parsing, whose goal is to identify author, institution or journal information. A canonical example is a set of Perl modules to parse reference strings, as proposed by Jewell [20], which is called ParaTools. More than 400 templates are included to match references strings. Besides, users are able to append new templates to this tool to adjust it.

Even though users can make variations of the tool and expand it, it is still unscalable. Generally, it is almost impossible to require all authors strictly follow some certain input styles, which means it will never be possible to cover all the situations and some parts of those references information will be outside the scope of this tool.

Given a more general context, template-based systems are also applied to evaluate predefined rules and assign labels to text regions, such as title, author, headline, etc. [25] [26] [35]. Still, any of those template-based approaches only suit for some certain tasks and are not general enough to be applied into different context.

2.4.2 Machine learning model-based approaches

In the scenario of meta-data components extraction from research articles, with the development of machine learning theory, if fed enough training data, a classifier can produce very high accuracy performance, regardless of certain references styles. Typical useful models are Hidden Markov Model (HMM), Support Vector Machine (SVM) and Conditional Random Field (CRF). In the work from Seymore et al. [50], a method is proposed that applies HMM to extract certain information, such as abstract, address affiliation, etc. SVM models are originally designed to assign a classification label based on the input. However, it is not easy for them to produce a sequence of labels, where the order between those labels matters.

Thanks to works from Han et al. [16], by separating the entire sequence labeling issues into two subtasks, assigning labeling for each line, and then adjusting these labels based on another classifier, this problem is solved. Taking the advantages from both finite state HMM and discriminative SVM, a CRF model is mentioned and discussed in the paper from Peng et al. [Peng and McCallum].

Features are crucial factors for machine learning approaches. For the same model, various selections of features may result in totally different performance. Focusing on identifying meta-data information from reference strings such as author name, affiliation and journal information, Isaac et al. tried out 9 differenr types of linguistic features to feed their CRF model in ParsCit [7] to compare their performances. In detail, these features include Token identify, N-gram prefix/suffix, Orthographic case, Punctuation, Number, Dictionary, Location and Possible editor. For some of them, features are expanded a bit deeper. For instance, the token identify feature consists of three different forms such as ’as-is’, ’lowercased’ and ’lowercased stripped of punctuation’. Some modifications may also be applied to other kinds of features. Though these features are originally focused on extracting meta-data information from reference strings, some of them can still be applied in a more general use case and give us some insights of feature candidates.

In addition to linguistic features, layout or markup styles information are also commonly used by researchers to identify logical structure from articles, especially for primary structures such as headings, captions, etc., which usually have discriminative font sizes, styles or positions with other elements. With these types of features, some heuristic decisions may be made to tag elements with their document structure information. For instance, headings typically are typeset in boldface, while rest parts of a section may not. Sometimes, regular expressions are applied together with Levenshtein distance in order to identify figure captions and references. In previous work from Nguyen et al. [36] about key phrase extraction, logical structures are also extracted when they preprocess their data. For instance, headings typically are typeset in boldface, while rest parts of a section may not. Sometimes, regular expressions are applied together with Levenshtein distance in order to identify figure captions and references [48].

2.5 Summary

In summary, logical structure extraction is quite an important work because it acts as the foundation of many subsequent tasks. A lot of efforts have been put into this area by researchers, and various systems are proposed to approach different areas or subfields in this field. With current template-based methods or machine learning models, problems like keywords extraction or citation parsing have been solved with quite good quality. However, a system is still missing which covers the entire work flow, from identifying primary structures like headings, figures to secondary structures such as section, cross-references and their mentions in the body, etc..

As explained by the name, template-based methods require some pre-defined templates that can be utilized later to identify meta-data information from a document. Hence, for inputs with certain formats covered by the templates, they can get quite good performance. However, for inputs that do not share the same template with those pre-defined templates, it is almost impossible to parse them with template-based methods unless the scope of the template is expanded. In other words, template-based methods are not so scalable. On the other hand, machine learning approaches, which do not explicitly define templates, rely on labeled data to guess the classification output. Sufficient data for training the model is essential to make it possible. In addition, features extracted from training data are also crucial to machine learning approaches, because they should cover enough information to enable the model make the correct decision and keep as little noise as possible at the same time.

Previously mentioned systems relied heavily on features like font size and bold, which means they have difficulties dealing with documents where markup information is not explicitly mentioned. It is interesting and potentially useful to expand the scope of those machine learning models and see the performance they can reach with only making use of linguistic features.

In the following chapters, we will discuss details of the available data we have in Elsevier and methods we apply to measure the performance.

Chapter 3

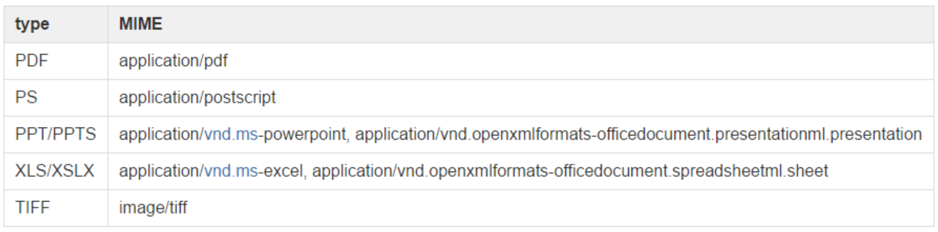
Data

In the scenario of Apollo project at Elsevier, article manuscripts will be taken as input and the final version of article with document structure information explicitly mentioned is expected as the output. In this chapter, we will describe available data in different stages and their relationship between each other. This chapter ends with a description of intermediate data representation we used in our experiments and the process how we create our data set.

3.1 Manuscripts

In order to innovate and automate document-structuring process at Elsevier, we have to at first figure out what source data we are faced with.

As we mentioned above, before a structuring process is actually started, another preflight process will be done beforehand to ensure authors provide enough document instances that cover all the content they want to deliver. During a preflight process, there may be several iterations of communications between the author and an Apollo Data Administer (ADA), who is responsible for those processes in the Apollo project. During these correspondences, several versions of manuscript document package will be recorded in the system, which is also called "Generations" inside Elsevier.



type	MIME
PDF	application/pdf
PS	application/postscript
PPT/PPTS	application/vnd.ms-powerpoint, application/vnd.openxmlformats-officedocument.presentationml.presentation
XLS/XSLX	application/vnd.ms-excel, application/vnd.openxmlformats-officedocument.spreadsheetml.sheet
TIFF	image/tiff

Fig. 3.1 A list of possible types with image information

Once the preflight process is completed, a stable generation of manuscript package should exist where all required document instances should be found. Actually, there is no any obligatory regulation which specifies how and what type of document authors should use and provide their article content. In other words, as long as authors think it makes sense to put the content in some certain types of files, they are allowed. For instance, main body of an article can be provided as either a Microsoft Word document or a PDF file. In other cases, an image of a figure is most commonly provided as a separate image file, but sometimes it can also be embedded in a Word file together with other information. Normally, image information can be expected in several types of files, as shown with in the example of Figure 3.1. In general, an initial manuscript submission looks like the example shown in Figure 3.2

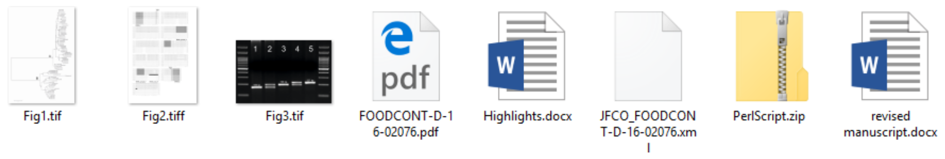


Fig. 3.2 A typical example of manuscript submission with all information

As one of the most important scientific articles providers in the world, Elsevier keeps different generations of vast papers, varying from their first initial submission of manuscript to the final published version. For each one of those articles, there is a Publisher Item Identifier (PII) bound with it, and those PIIs are then applied to manage those huge amount of articles. In the scope of Apollo Project, articles are collected from several fields and areas, including the following journals:

- Carbon
- Journal of Molecular Structure
- Chemosphere
- Materials Chemistry and Physics
- Journal of Food Engineering
- Food Hydrocolloids
- Journal of Cereal Science
- Food Microbiology
- Computers in Human Behavior

- Superlattices and Microstructures
- Food Control
- Journal of Cleaner Production
- Renewable Energy
- Journal of African Earth Sciences

Through the list, though many of them are related with food or chemical fields, quite a broad areas is still covered by the scope of our journals, which still provide us with a good starting point to experiment and see if machine learning approaches can actually contribute to the structuring task.

In the scope of this final project, 293 PIIs are randomly picked out in total from our database and serve as the data used in our machine learning task.

3.2 Published articles

When it comes to supervised machine learning tasks, labeled data is indispensable. In the specific case of our document structure extraction scenario, for each published item with a certain PII, a final published generation of the article is expected to be found. As shown in Figure 3.3, the published version of an article consists of several files such as some image files, a standard PDF of the paper and a XML file with all the metadata of this article.

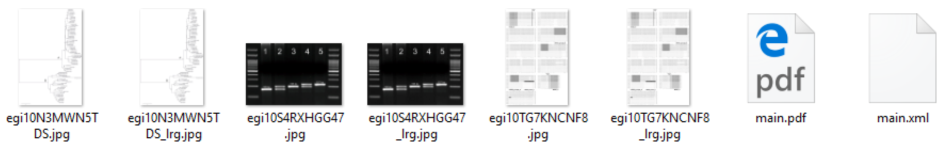


Fig. 3.3 A typical example of published article

Together with a separate Document Type Definition (DTD) schema, all document structure informations are recorded in the XML file by all kinds of tags with different meanings. For instance, if we look for "Title" information, it is always encompassed by tag <ce:title>...</ce:title>. Since entire XML file is built with a typical tree-like structure, according to the meaning of different structure entities and their relationships between each other, those elements are also categorized and placed with different hierarchies and levels. For instance, metadata informations including "Title", "Author groups" are stored as children elements of a <head>...</head> element. Similarly, most content of an article is supposed to be found inside <body>...</body> element.

Even though, it should be noted that not every structure element follows strict hierarchy rules. For entities such as figures and tables, instead of being embedded somewhere in the document structure, they should be seen as floating beside rest part of the article. This point is also reflected in the main XML document. As a "floating" section, a <ce:floats>...</ce:floats> element contains all content about images and tables, which are referred in other part of the document.

3.3 Relations between Manuscripts and Published Articles

For each PII, a complete manuscript package and published article can be seen as two different versions of one similar content. After a set of document instances are submitted by the author as manuscripts, several steps and processes are involved before the editing is completed and it is finally published. For example, some texts in a figure caption can be removed or inserted in order to ensure the entire sentence make more sense. Or in the case where correspondence author part does not contain enough information, compared with original manuscripts, much more data may be found in the final published version of the same article.

In summary, the content covered in the manuscripts and published articles are not exactly the same. However, all editing or revision process are not supposed to modify or change a lot of the contents, which means the content between manuscripts and published version paper is still similar. This characteristic provides us with possibilities to align the contents between two versions of articles and then produce a data set with label for subsequent supervised machine learning.

This thesis work includes several document structure extraction experiments for the following tasks:

- title
- author group
- affiliation information
- section heading
- figure and table caption
- reference list

3.4 Intermediate data representation

3.4.1 Motivation

As shown in the previous sections, in our particular Apollo scenario, content of an article is split into a set of document instances with different types. In other words, we need combine all files and extract the document structure from them. For a machine learning model, it takes data with a certain format as input during the training process and then it is able to make the correct prediction to new-coming unseen data with the same format. Hence, designing a machine learning model that takes multiple type of inputs is not easy. Considering that authors all have their preferences grouping and providing their manuscripts, it is even harder to cover all cases. Therefore, some kind of intermediate representation of manuscripts is needed, which should be able to provide us with a standard way presenting all the contents.

On the other hand, representing or encoding text in a both machine-readable and human-readable way has been discussed for a long time by a lot of researchers. Since 1994, a set of guidelines of texts encoding methods have been delivered by the Text Encoding Initiative (TEI), which are widely used by researchers, libraries and individual scholars to present or preserve their works [51].

Even though XML or HTML is commonly applied to record the content, the usage of these formats are still limited in some cases. As originally designed for web applications, HTML defines a quite concise but clear way to display content in browsers. By splitting style formats from content, HTML is easy for humans to understand. However, at the same time, the usage of HTML is also limited by its predefined tags. In other words, it is not easy to extend the current tags to satisfy new needs or situations. When compared with HTML, XML goes without the limitation of predefined tags, which means people are more free to extend the tags to meet any new use case. Yet, owing to the arbitrariness of XML, there is always a DTD document associated together with the XML in order to specify a certain use case. Hence, it is common that various scholars or companies may apply distinct DTD to match their own businesses, and let alone those DTDs are normally confidential and cannot be accessed by people outside, which makes it even more difficult to build a general standard. For instance, Elsevier owns its own DTD specification which should not be used by any external organization. Besides, considering all authors may have their own preferences and selections of ways to organize their works, it is also not easy to reorganize all their work separately and meet such a complex convention. Furthermore, since contents in a XML file is properly organized with a particular hierarchy by all kinds of tags, before the document structure information is obtained, there is no way to make best use of tags and build a meaningful structure for the content. Therefore, another kind of representation of those articles and works is needed, which encodes content in an easy way but loses as little information as possible from original document instances.

In the intermediate content presentation applied in the text generation system called ICONOCLAST [42], input text is specified in an XML file together with their rhetorical relationships and propositions [4]. Inspired by this work, we also propose some representation of a manuscript document by specifying its elements and their relationships behind, which is called Structure Document Format (SDF).

3.4.2 Structured document format

Proposed as a structured representation of a document such as a scientific report, scientific article or a scientific book, SDF consists of elements, tags and relations, which can be grouped into two levels. In the first level, which is also the superficial level, we keep the simple elements like text or image, and tags that remains the markup information from manuscripts. On the top of first level, the second level of our SDF document is made of complex elements and different relations, including meronomy, ordering and reference relations. Complex elements can be regarded as elements serving some structure role. By defining the relations between different elements, our SDF documents are able to keep the structure information underneath the article.

From technical aspects, a persistent form of SDF document is also required in order to store it as a file in computer. Since based on different needs or requirements, a SDF can be exported to several formats which specify ways of laying down the structural elements in a printed form, hence, a pure SDF should be separated from any detailed guideline how it should be presented. Hence, only content and relations regarding their abstract document structure are persisted.

Technically, elements and different relations are stored separately in several files so that it is easier to understand them with respect to their purposes behind. For each SDF document, in order to manage several files at the same time, we store them as a single ZIP file, which provides us a easy way to organize them in directories and files. Files regarding text, functions or annotations are in JSON format, where all elements are represented with their own unique identifier. On the other hand, all image files in a SDF instance are stored in PNG or JPG format.

3.5 Creating data set for machine learning

Since current machine learning models approach classification problem by applying mathematical analysis and finding the patterns beneath the data provided, hence, models can fail finding patterns and making correct predictions for new unseen data if a lot of noise or even wrong data is provided during the training process. On the other hand, provided that data is not clean, there is no way to evaluate the performance of the model correctly. In a word, quality of source data is crucial to the performance of any machine learning model, and it is even more significant for supervised learning tasks, where a model can result in something totally wrong when incorrect labels are provided.

In that our goal about document structure extraction is a typical classification problem, we should tackle it by utilizing supervised learning and getting a classifier. For this classifier, it will not be useful until it takes raw data as input and make the correct prediction. In our specific scenario regarding processing manuscripts, the raw input data is the whole manuscript including several files with distributed information. As an output, a specific document structure for an article is expected at the end. In our case, we tackle this problem in the level of Structured Document Format (SDF) that is previously proposed.

Because a SDF instance contains both content and structure information for an article, we transform our original document structure extraction task to guessing potential relations between different elements in the SDF based on their content and tags. For instance, given an initial SDF document, our classifier should be able to create a set of complex elements and define proper relations for them and other elementary element. The entire work flow is visualized below with Figure 3.4.

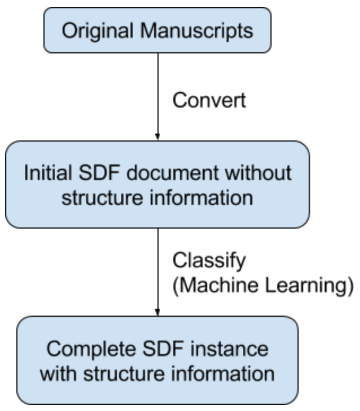

Original Manuscripts
Convert
Initial SDF document without structure information
Classify (Machine Learning)
Complete SDF instance with structure information

Fig. 3.4 The process of machine learning task

In order to train and evaluate our classifier, we propose an approach to make use of available manuscripts documents as well as published version of articles to produce a data set that is valid to boost our machine learning task. Though with only using published versions of articles, we can still get data set for training our classifiers, however, contents will go through an revision process and mistakes will be corrected before a paper is actually published. In other words, the content itself is not exactly what we will deal with from the beginning of our work flow. Hence, in order to build classifiers robust enough to take the original manuscripts as input, we propose the method to create the data set as shown by Figure 3.5. To be specific, for each article package corresponding to one single PII number, we pick out the final Elsevier XML file where all contents are kept together with their document structure tags. For one specific structure content, such as "section heading", we pick out all the contents and align them with elements from initial SDF document, which is converted from all original manuscript documents. During the alignment process, for each positive element found from the reference SDF document with a certain structure type such as "caption", we calculate the text similarity between it and every element entity from initial SDF document, which in our case is each paragraph, and then we come up with following two different strategies to link them and mark the element from the initial SDF document with the most suitable label:

1. Greedy Searching - Iteratively, for each positive sample from reference SDF document, we find the element from the initial SDF document with the highest similarity, and once the similarity surpasses a certain threshold, we mark this element as a positive label and will not consider this element anymore for other samples from the reference SDF document. It should be noted that before the manuscript is finally published, several revisions can happen, and revisions are not exactly the same for information with different structure types. Therefore, for different structure types, the particular threshold may vary. In order to find out the best threshold for each document structure type, we print out best matchings with their similarity and manually decide it.

2. Alignment with Hungarian Algorithm [27] - Instead of linking the sample from reference SDF document to initial SDF document one by one, we get their similarities with all initial element entities all together in a matrix. And then we apply Hungarian Algorithm to reach the best assignment between those elements, which decomposes the whole task to bipartite graphs and find the perfect matching.

   in order to reach a best overall assignment between those elements with a lowest cost.

For these two different strategies, they all have the their own advantages and disadvantages. The greedy solution is simpler and costs fewer time and space resources to get the result. However, a threshold value is involved in this process and it matters a lot to the alignment result, but choosing a suitable threshold is a heuristic process. In addition, a greedy

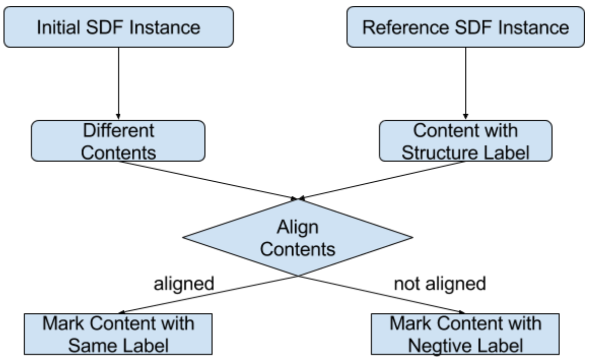


Fig. 3.5 The process of aligning elements between available resources and producing data set

solution will take a risk when a "positive candidate" from initial SDF document is aligned to another positive sample from reference SDF document with great similarity and will not be "visible" to any other sample even for the correct one. Then when it comes to the sample whose "candidate" is already aligned to something else, it fails finding the correct one and will match something else with best similarity. At the end, noise is introduced in our data set.

When compared with greedy solution, the strategy based on Hungarian Algorithm is more robust because it will always return results with the lowest overall error rate. In other words, if some one sample fails finding the best alignment, it is not likely to affect others and cause some chain reaction. However, solution based on Hungarian Algorithm requires more space and time resources to accomplish the task. The original time complexity of Hungarian Algorithm is $O(n^4)$, but improved variations are also proposed by some researchers with time complexity as $O(n^3)$, for example Jack Edmonds [9] and Richard M. Karp [23].

Considering the computing capability of our computers and efficiency, we started our experiments from the greedy solution on the section heading extraction task. After contents were aligned and the data set was created, we manually checked the results and it turned out to be a acceptable one with quite a few mistakes. Then we expand the greedy solution to other structure types and postprocess results with a manual check so as to remove as many noise as possible.

Chapter 4

Methods

For any machine learning system, as information is extracted from data and packaged in different features, features selection is of significance to the whole performance. In this chapter, we discuss how we are inspired by other researches and propose our own sets of features, after which we will introduce machine learning models that were used in our experiments and measures to evaluate our performance. At the end of this chapter, we will briefly describe the tools and libraries that have been used in this work.

4.1 Selection of features

For our classifiers, the goal is predicting the potential document structure category for each element from the initial SDF instance. We decompose this problem to several subtasks, each of which focuses on a certain structure type, such as "section heading", "author group", etc.

In order to convert original text elements to a format which can be interpreted by computers and fed into our machine learning models, we need propose some feature sets and extract as much useful information as possible from original texts. Many strategies of choosing and grouping features are proposed by other researchers. In the research we previously referred in the Chapter 2, four observables cues are proposed to group different features, including geometric, marking, linguistic and contextual cues [53]. Besides, in the work of sentiment classification from Awais et al. [1], features are separated to word-level features, contextual features and sentence structure based features. Through those categories of features, we can see information is extracted in different levels to boost corresponding machine learning models. Inspired by those works, we also extract features by applying different level of analysis to the textual content. As the contextual location information is not in the scope of this research work, we mainly focus on the textual information and visual markup information that can be obtained from manuscripts. Similarly, we propose our features in three groups, which is shown by Table 4.1 together with short descriptions.

Visual features. When authors edit their articles, it is likely that they will apply some specific formating styles to different document structure elements in their works. Moreover, with the help of all kinds of editors like Microsoft Office Word, authors can easily mark texts with different styles, bold or italic style, or change the size of fonts. Different authors may have their own preference towards styles to be applied to their articles, for example, people may choose different font sizes to write their papers, and it means that exact values of font size are not things we can widely use in our classifiers as features. Nevertheless, though styles may vary among different authors or papers, they are more likely to be aligned and consistent within a single article. For example, with regard to a certain article, if the author apply a bold style to one section heading, then this style should also be expected from other elements serving as section heading. From this aspect, those features do provide some information that can help our classifiers to make decisions. In our rich-text set of features, we propose several styles or markup information as features, trying to make best use of information that we can find from manuscripts.

Shallow-textual features. Given a piece of text, without doing any syntactic or semantic analysis, they still provide a lot of information which can boost our document structure extraction task. Hence, with this group of shallow-textual features, we focus on information that can be easily obtained by checking the appearance or counting. As a crucial formation part of sentences, punctuations are always used by authors in their articles for all kinds of purposes. For instance, comma is usually used to separate a long sentence to multiple shorter parts, while parenthesis may be used to contain some comments or supplements for the content. When it comes to document structure types, normally people will not put a lot of punctuations in the "title", but in some other cases, we may be able to find a lot of comma or dot in an reference item, which are commonly used to separate fields in a bibliography item. Several punctuation-based features are included in our feature sets to make use of those information.

In some particular document structure category like "section heading", we notice that it is quite common authors prefer to arrange them as numbering list items and mark them with prefix indexing number, which means some paragraphs will start with digits. In our features,

[1]This gazetteer list is case insensible, and it is the same for the table-figure gazetteer list

[2]Authors can use either "acknowledgement" or "acknowledgment", and we pick out the common part to include both situations

[3]It consists of seven types of pos tag: noun (NN), verb (VB), adjective (JJ), adverb (RB), conjunction (CC), determiner (DT) and adposition (IN).

[4]Pre-defined list of verb stem: "show", "illustr", "depict", "compar", "present", "provid", "report", "indic", "suggest", "tell", "demostr" and "reveal".

Table 4.1 Features for machine learning models

Feature Category	Feature Name	Explanation
visual features	average font size	the average token font size in a paragraph divided by global average token font size
	bold percentage	the percentage of tokens that are marked as bold in a paragraph
	italic percentage	the percentage of tokens that are marked as italic in a paragraph
	sub exist	a boolean value indicating the existence of any subscript in a paragraph
	super exist	a boolean value indicating the existence of any superscript in a paragraph
	underline exist	a boolean value indicating the existence of any underline in a paragraph
shallow-textual feature	paragraph length	the count of tokens in a paragraph
	sentence count	the count of sentences found in a paragraph
	capital percentage	the percentage of tokens in a paragraph that are started with capital letter
	number count	the count of number found in a paragraph
	comma count	the count of comma found in a paragraph
	other punctuation count	the count of any other punctuations except comma
	begin with digit	a boolean value indicating either a paragraph is started with a digit or not
	end with dot	a boolean value indicating either a paragraph ends with a dot or not
	end with colon	a boolean value indicating either a paragraph ends with a colon or not
	begin with tf	a boolean value indicating either a paragraph starts with word "Tab" or "Fig"
	begin with capital	a boolean value indicating either the content starts with a capital letter or not
	in sh gazetteer	a boolean value indicating either any word in the predefined section-heading gazetteer list is found in a paragraph or not. (section-heading gazetteer list[1]: abstract, reference, keyword, highlight, acknowledg[2])
	in tf gazetteer	a boolean value indicating either any word in the predefined table-figure gazetteer list is found in a paragraph or not. (table-figure gazetteer list: table, figure)
syntactic-textual feature	POS[3] ratio	the percentage of tokens marked with a certain part-of-speech tag in a paragraph.
	first vb in list	a boolean value indicating either any verb in a specific list[4] can be found in the first sentence of a paragraph.

we include "begin with digit" specifically for this reason. Similar support can also be found with our "begin with tf" feature, given that figure or table captions are very likely to start with words "Figure" or "Table".

Though originally designed as a geographical dictionary or directory, gazetteer list is a great concept to link a specific use case together with some keywords or common words. In the natural language processing area, gazetteer lists are also commonly used as a specific type of features, such as the research from Jochim et al. [21], where an annotated list of resources in the NLP domain is used for parsing citation content. In our experiments, we also propose two gazetteer lists to include some common words used in section heading and caption, such as "Abstract", "Acknowledge" or "Table".

Syntactic-textual features. This set involves linguistic features requiring syntactic analysis. In some previous sentiment analysis works from researches like Pang et al. [39] and Athar et al. [1], n-grams of letters and Part-of-speech (POS) tags are used to train models that can tell the sentiment information as positive or negative. In our manuscripts scenario, because authors are free to write any content in their articles, which means the variance of words used may be quite big and statistics analysis based on n-grams features may not be very helpful. For this reason, we do not include n-grams as our features. However, from another aspect, normally authors will still follow some grammar convention to organize their article structure. For instance, for a common "section heading" in an article such as "Introduction" or "Results and Discussions", people may not expect too much verb phrase in it, but they should be widely used in the body part of the article. Therefore, we have the reason to believe that POS tags do contain information to help our classifiers tell the document structure type of an element. In order to get a more general information, we choose to use seven categories of POS tag information including noun, verb, adjective, adverb, conjunction, determiner and adposition.

With respect to some particular POS tag like "Noun phrase(NN)", because the length of each paragraph varies and this information is not aligned among all paragraphs we have obtained, it is not fair to only count the frequency of their appearance in a text. Hence, we select to normalize this information by getting the percentage of POS tag in a single paragraph.

From articles in our data set, we also find a distinction between the caption for a figure (or table) and its mentions in the rest part of the article. For instance, a caption often looks like "Figure 1. Result of experiments.", while its mention is likely to be "Figure 1 shows the result of experiments.". By analyzing the syntax of these two sentences, we find that a verb is more likely to be found in the first sentence in the mention of a figure (or table) than its caption. Furthermore, in our data set, we identify several verbs that are most commonly found in mentions of figure or tables, and we also narrow them down to a list of verb. On the other hand, as authors may apply different tenses to illustrate their works, we keep the stem of those verbs to make it more robust.

In general, we propose our features in three categories, each of which respectively has emphasis on markup information, text with superficial information, and content with syntactic analysis. As mentioned in the previous chapters, a lot of current applications dealing with document structure relies on markup or style information, we want to compare those markup features together with other types of features and see how their performance vary. At the same time, we also want to investigate how we can extract as much information as possible from texts with features other than markup styles and improve the performance of our classifiers.

4.2 Preprocessing

For each paragraph, before a certain feature value is extracted, it should be noted that we take some preprocess steps to normalize the paragraph content and remove some information that may introduce noises to the feature itself. Those steps are designed differently in order to suit specific document structure type. In the rest of this section, we explain how the data is preprocessed and the relevant purpose for it.

1. Remove the numbers at the beginning of each paragraph (Section heading)
   For any paragraph that is to be classified as either a section heading or not, such as "1. Introduction", the prefixed number like "1. " is a good indication that this paragraph is one of the section headings, which is also reflected by the feature "begin with digit". Hence, when we get other features like "number count", we expect that it only reflects the main body of the paragraph, which is "Introduction" in our above example, and then the prefixed numbers together with the dots or spaces are removed to make data clean

2. Remove prefixed "Fig*" or "Tab*" for each paragraph (Caption)[5]
   The prefixed word "Fig*" or "Tab*" is a strong indication that one paragraph can be a caption of some tables or figures, and this attribute is already reflected by "in tf gazetteer" feature. Hence those prefixed words are removed so that these information will not influence other features.

3. Remove End of Line (EOL) symbols at the beginning of each paragraph (Title)
   Depending on the style how authors provide their original manuscripts, in some cases,

[5]"Fig*" - "Figure" or "Fig" regardless of capital letter; "Tab*" - "Table" and "Tab" regardless of capital letter

some EOL symbols can be found at the beginning of the paragraph where the title exists without any explicit meaning. Therefore, those prefixed EOL symbols are removed to avoid mistakes to other features.

4. Remove the EOL symbol, commas, footnote digit and $*$ symbols (Author)
Because of the preferred style from authors' perspectives, some meaningless EOL symbols can also be included as part of a author group content, which is removed in our case. Furthermore, some $*$ marks or digits can be used as superscripts or subscripts to indicate an author as corresponding author or supply some further information. However, those information will have negative effects on other features such as "other punctuation count" or "number count" because they are not strictly parts of the content itself in terms of semantics. Therefore they are also removed.

4.3 Description of models and methods to evaluate performances

For all subtasks in our document structure extraction task, we apply each group of features to train and evaluate several models. In the rest part of this section, we at first describe models used in our experiments, and then we will describe how we evaluate the performance of our models.

4.3.1 Naive Bayes models

As a family of simple probabilistic models, naive Bayes classifiers has been studied extensively since 1950s [46] and brought a lot of application in text classification, like spam classification.

Based on statistics and probability, naive Bayes classifiers tackle classification problems with a naive assumption that features are strongly independent with each other. In other words, it assumes there is no any correlations among features, calculate their conditional probability and put all together and do the final decision. As indicated in the name, the essence of naive Bayes model is Bayes' theorem [54], which can be explained by the following formula:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

where A and B are events and probability of event B is not 0. The member of the equation is also know as joint probability, $P(A,B)$ Through this theorem, we can tell that conditional probability of observing one event A with another event B being true can be calculated by probabilities of observing these two events regardless of each other and another conditional probability of them.

In machine learning task, with feature vectors in a certain dimension, the classification problem can be solved by getting a set of conditional probability

$$p(C_k|x_1,x_2,...,x_n) = \frac{p(x_1,x_2,...,x_n|C_k)p(C_k)}{p(x_1,x_2,...,x_n)}$$

$$p(C_k|x_1,x_2,...,x_n) = \frac{p(C_k,x_1,x_2,...,x_n)}{p(x_1,x_2,...,x_n)}$$

$$p(C_k,x_1,x_2,...,x_n) = p(x_1|x_2,...,x_n,C_k)p(x_2|x_3,...,x_n,C_k)...p(x_{n-1}|x_n,C_k)p(x_n|C_k)p(C_k)$$

where C_k is a certain target class and x_i is the value of each feature.

With the assumption that features are independent with each other, the joint probability can be easily calculated as $p(x_i|x_{i+1},...,x_n,C_k) = p(x_i|C_k)$, and then the classification problem can be solved.

4.3.2 Tree models

Born as a decision support tool, decision trees make use of tree-like graph or model to visualize decisions and possible consequences together with their chances. Since this model helps identify best strategies based on input data, it is commonly used in machine learning. This is also called "decision tree learning".Figure 4.1 is an example of decision tree used in machine learning area.

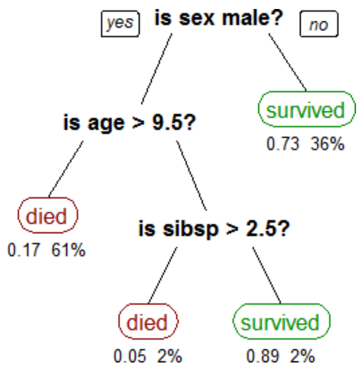


Fig. 4.1 An example of Decision Tree Classifier

For decision tree learning, the input data always looks like the sample below, where Y represents the label of this data and X is a vector including all the features or variables of the data.

$$(X,Y) = (x_1, x_2, ..., x_n, Y)$$

In a typical decision tree model, each node corresponds to one feature from the input data vector, and every leaf node contains the target value of data points which follow all values marked on the paths from root node to the leaf. It should be noted that each interior node of a decision tree can be split to any number of branches, not only by two edges as shown in the example. During the process when the decision tree model is trained, based on the minimal number of samples for each branch, or the maximal number of branches coming out from one single node, branches can be split in different ways.

In general, decision tree models have several advantages. For instance, the usage of graphs brings the feasibilities to visualize models and make it easier to understand. Furthermore, as decision tree models use strategy like information gain to split their branches and no complex calculation is involved during the training process, they can also handle large datasets in reasonable time.

In the rest of this subsection, three different variation of tree-like models are described, and those are also models that we try out in our experiments.

Decision stump

Decision stump is a one-level decision tree, which means only one internal node exists and it is connected to leaves directly [19]. Since decision stump are simple and easy to build, they are often used as components to ensemble other machine learning models, such as meta models that will be discussed subsequently in this section.

REP Tree

REP Tree is a normal decision tree model that is built with strategies like information gain or variance, and if necessary, reduced-error pruning [12] with be done to prune the decision tree and prevent it from being to deep or complex. With reduced-error pruning, the pruning process starts from leaves nodes and each node is replaced with its most popular class, unless the prediction accuracy is affected. Though the concept behind reduce-error pruning is somehow naive, but it is a simplest way to prune decision trees because of its speed and simplicity.

Random forest

Random forest, which is also known as random decision forest, is another variation from original decision tree model. Random forest models solve classification problem by constructing multiple decision trees at training time and output the class that appears most times in all candidate results [17].

Normally, decision tree has the habit of overfitting, which means the model is too deep or complex and then perform very poor versus new unseen data, though it can reach high performance with training data. However,when an random forest model is trained, a small number of features are randomly picked and used to build each decision tree instance. In other words, not all of features will be used at once. In this way, random forest model improve the overfitting habit of decision tree.

On the other hand, because random forest will only take parts of features to construct inner decision trees, the performance may not be very satisfying if the dimension of features is too high, which means a lot of information is lost during the training step.

4.3.3 Support vector machine

Given a classification problem where data points scatter in a 2-dimension space, normally a line is expected to separate different classes. Figure 4.2 shows an example of a typical linear separable classification problem.

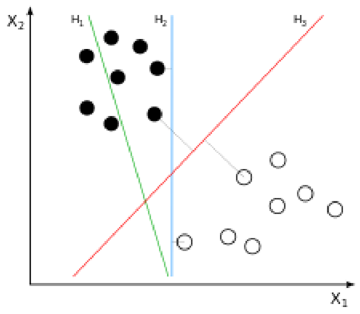


Fig. 4.2 An example of linear separable classification problem[6]

Similar, for a typical classification problem in machine learning, a data point is normally treated as a p-dimension vector, and a solution is usually a $(p-1)$-dimension hyperplane to split those data points with some margin, which is also called "linear classification". However in real situations, sometimes data is impossible to be linearly classified, and in this case, Support Vector Machine (SVM) or Support Vector Network [6] provide another type of way to tackle such a classification problem.

Instead of projecting data points to a $(p-1)$-dimension hyperplane, SVM models utilize a kernel function $k(x,y)$ to project seen data points to a higher-dimensional space, where the

[6]https://en.wikipedia.org/wiki/Support_vector_machine

data is able to be linearly separated. Based on the characteristics of specific data the model is tackling, kernel functions can be either linear or non-linear, which also expands its usage.

4.3.4 Multilayer perceptron

As a type of feedforward artificial neural network model, multilayer perceptron models (MLP) are able to map a set of input nodes to their appropriate output nodes [45]. A MLP model can be nicely represented as a directed graph, where several hidden layers of nodes can be found between input nodes and output nodes. Figure 4.3 shows an example of neural network model, and multiple layers of hidden nodes are expected in a multilayer perceptron model.

In a MLP model, each layer of nodes is fully connected to the next one, and each node except input nodes is also called neuron with a certain nonlinear activation function. Not like a linear perceptron, a MLP is able to separate data that is not linearly separable.

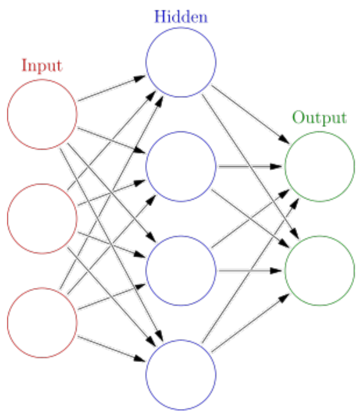


Fig. 4.3 An example of neural network model[7]

4.3.5 Ensemble models

Generally speaking, for a certain machine learning problem or scenario, selecting the "best" model or method is not easy at most of time. From this aspect, a lot of researchers have studied how to make best use of current available machine learning algorithms and improve general performance.

Ensemble learning, as one of the successful samples, is then proposed as combining multiple models to hopefully achieve a better performance [38]. As indicated by its name, ensemble learning is usually interpreted as generating multiple models with the same base

[7]https://en.wikipedia.org/wiki/Artificial_neural_network

learner, but different types of learners can also be used in the wider scope of ensemble learning. From technical aspects, there are several types of ensembles, including Bayes optimal classifier, bagging, boosting, etc. And in the subsequent content, we will introduce two types of them, boosting and bagging, which we will also use in our experiments.

LogitBoost

By training each new model to emphasize instances that are misclassified by previous models, boosting can be interpreted as incrementally building a ensemble model. Though boosting proves higher accuracy towards target data, however, it also somehow risks overfitting input data.

Originally proposed by Jerome et al. [11], LogitBoost model applies logistic regression to update and minimize the overall cost of the whole ensemble model.

Random SubSpace

Bootstrap aggregating, which is also abbreviated as bagging, means each single model in a ensemble model will vote with the same weight and the most voting result will be picked as the final decision. During the training process of a bagging model, a smaller-scale subset of data will be randomly drawn and used to train each model. In this way, the ensemble promotes the variance for entire model and prevent overfitting.

As an example of bagging model, random subspace model tries reducing the correlation between inner models of a ensemble model by randomly picking part of features instead of the entire feature [18].

4.3.6 Performance evaluation

In order to correctly reveal the performance of our machine learning models, we record the precision, recall and F1-score as measures for our models [43]. Given the results returned by a classifier, with precision score, we are able to tell the percentage of correct prediction among all prediction results for a certain class. On the other hand, for all instances with a certain class in the data set, recall score of a classifier reveals how much of them are correctly predicted.

When we evaluate the performance of our models, we adopt 10-fold cross validation to prevent overfitting, which means a machine learning model is so complicated that it can perform well on the same training data but perform poorly if new unseen data is coming. By splitting all data to 10 folds, training the model with 9 of them and evaluating the model with the rest one, we are able to simulate some artificial unseen data for the model and avoid overfitting. Besides, by switching the evaluating fold and repeat this process 10 times, we are able to reduce the influence brought by the potential unbalanced distribution of data.

4.4 Method to deal with unbalanced data

In the scope of our data set, we collected 43,888 paragraphs in total. Among all available paragraphs, we have 293 titles, 282 author groups, 656 affiliations, 5,637 section headings, 3,244 captions and 13,867 reference items, along with 19,908 normal texts in the main body. As shown by the numbers, we are faced with a extreme unbalanced data set, especially in terms of title, author group and affiliation entities. If we take titles as an example, since the data distribution of positive and negative instance is extremely unbalanced, a binary classifier can easily get very nice overall performance by predicting each instance as "normal text", but actually it contributes nothing because it fails all instances with positive class, which we should care most. In our experiments, we address this issue by assigning different cost weights to train our models. In other words, for different type of mistakes during the training process, the algorithm will update parameters of the model with different scales, so that our classifiers can care more about the minority class and do not ignore it. There are also several alternative ways to address unbalanced data issue, such as sampling, which means randomly sampling the same number of positive and negative instances to build a balanced data set for training and evaluating the classifiers.

4.5 Libraries and toolkits

For programming language, we select PYTHON as the main language for this final project because of the active community, where a lot of open-source resources can be found and make work much easier.

For manipulate our data including label and specific features, we used a PYTHON library called Pandas.

When it comes to training and evaluating our models, we use WEKA [15] as the tool to conduct our initial experiments as it provides plenty of predefined models, and it is convenient to quickly try several different models and compare their performances for different tasks. After intuitive experiments are completed, in order to make the code base consistent with rest parts of Apollo project and embed our model, we implement the best model in another PYTHON library called Scikit-learn [40].

Chapter 5

Results

In this chapter, we present the results of our experiments. At first, we display the performance of our binary classifiers, each of them focusing on one specific document structure type, such as section heading, caption, etc. After this, we continue with the results of our multi-class classifier, which contains all functionalities from previous binary classifiers and is able to predict one paragraph to all document structure types.

In our Apollo project, there is a human operator involved in the document structure extraction process so as to make sure they are correctly extracted, so that the rest editing processes can be proceeded. Hence, the performance of our machine learning models may have different impacts on the tasks that human operators need do. For instance, if our machine learning focuses on improving precision rather than recall score, the human operator may not need worry too much about the confidence of returned positive instances however, he may put more effort on diving into the manuscripts and finding out those missing components. On the contrary, if our machine learning system ends up with nice recall performance but poorer precision score, for example, almost 100% recall, which means all the entities with a certain document structure are returned as positive prediction, then the human operator do not need spend too much time looking for the rest elements from the manuscripts, yet more efforts are expected to filter out elements that are misclassified as positive class.

In general, the trade off between precision and recall score depends on the subsequent tasks and may be different, hence, in the scope of this thesis, we do not have any preference between them and a Receiver Operating Characteristic(ROC) curve is shown together with other performance measures for each task, which visualizes how we can sacrifice one aspect and boost another one.

Table 5.1 Performance of binary title classifier - Random Forest

Feature	precision	recall	F-score
shallow-textual	0.539	0.642	0.586
syntactic-textual	0.133	0.160	0.145
shallow, syntactic-textual	0.703	0.614	0.656
visual	0.165	0.143	0.153
all	**0.829**	**0.710**	**0.765**

5.1 Binary classifier for each document structure type

As the beginning of our experiments, we decompose the whole document structure extraction task into six small-grained tasks, including title, affiliation, author, section heading, caption and reference. We repeat similar experiments for each one of our sub tasks and get the following results.

A binary classifier can be built based on several specific machine learning models. Each family of models has their own specifications and advantages, hence we experiment with at least one specific model for each commonly used family of machine learning models and record their results. As a result, we collect the precision, recall and F-score for all positive classes, which are title, author group, etc. In this section, we keep those scores for the model that returns the best performance.

5.1.1 Title classification

Our binary classifier that does title classification takes a paragraph of texts from manuscripts as input and predict it as a "title" or "normal text". In our experiment, we get the best performance with our Random Forest classifier. Table 5.1 shows precision, recall and F-score we have got from different groups of features.

Figure 5.1 demonstrates the ROC curve of Random Forest model with all features.

5.1.2 Author group classification

Our binary classifier that does author group classification takes a paragraph of texts from manuscripts as input and predict it as a "author" or "normal text". In our experiment, we get the best performance with our Random Forest classifier. Table 5.2 shows precision, recall and F-score we have got from different groups of features.

Figure 5.2 demonstrates the ROC curve of Random Forest model with all features.

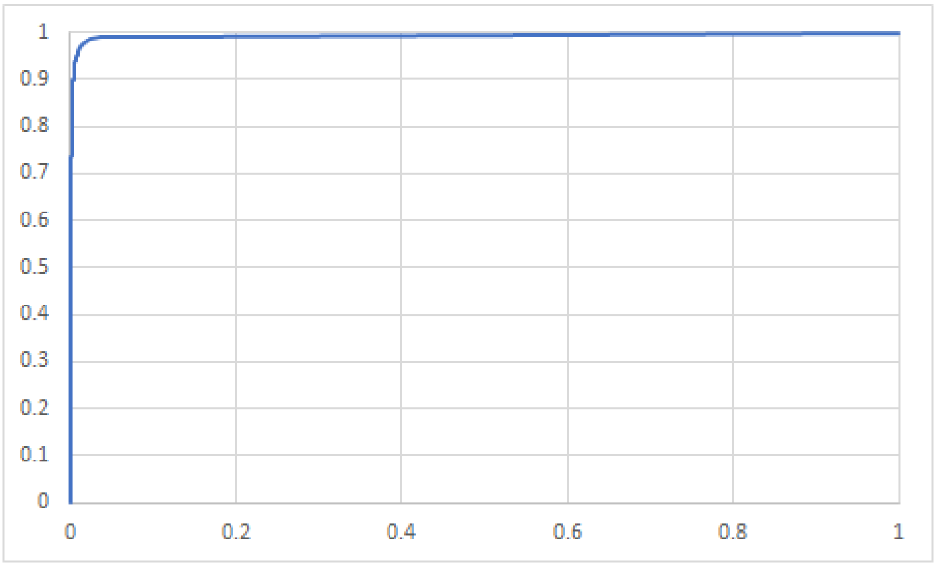


Fig. 5.1 The ROC curve of Random Forest title binary classifier with all features

Table 5.2 Performance of binary author classifier - Random Forest

Feature	precision	recall	F-score
shallow-textual	0.723	0.777	0.749
syntactic-textual	0.197	0.273	0.229
shallow, syntactic-textual	0.747	0.755	0.751
visual	0.205	0.191	0.198
all	**0.833**	**0.851**	**0.842**

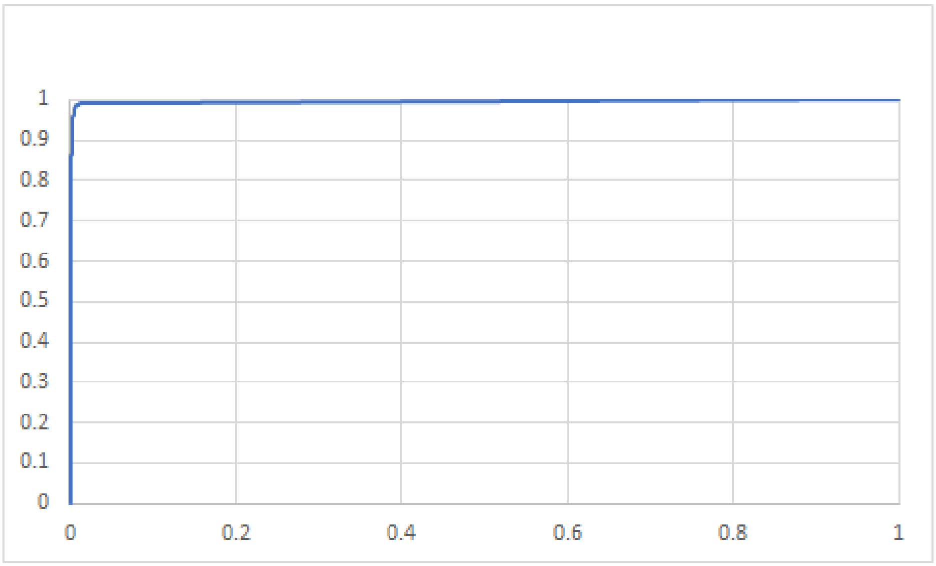


Fig. 5.2 The ROC curve of Random Forest author binary classifier with all features

Table 5.3 Performance of binary affiliation classifier - Random Forest

Feature	precision	recall	F-score
shallow-textual	0.789	0.848	0.817
syntactic-textual	0.239	0.578	0.339
shallow, syntactic-textual	0.826	0.841	0.834
visual	0.310	0.341	0.325
all	**0.866**	**0.869**	**0.868**

5.1.3 Affiliation classification

Our binary classifier that does affiliation classification takes a paragraph of texts from manuscripts as input and predict it as a "affiliation" or "normal text". In our experiment, we get the best performance with our Random Forest classifier. Table 5.3 shows precision, recall and F-score we have got from different groups of features.

Figure 5.3 demonstrates the ROC curve of Random Forest model with all features.

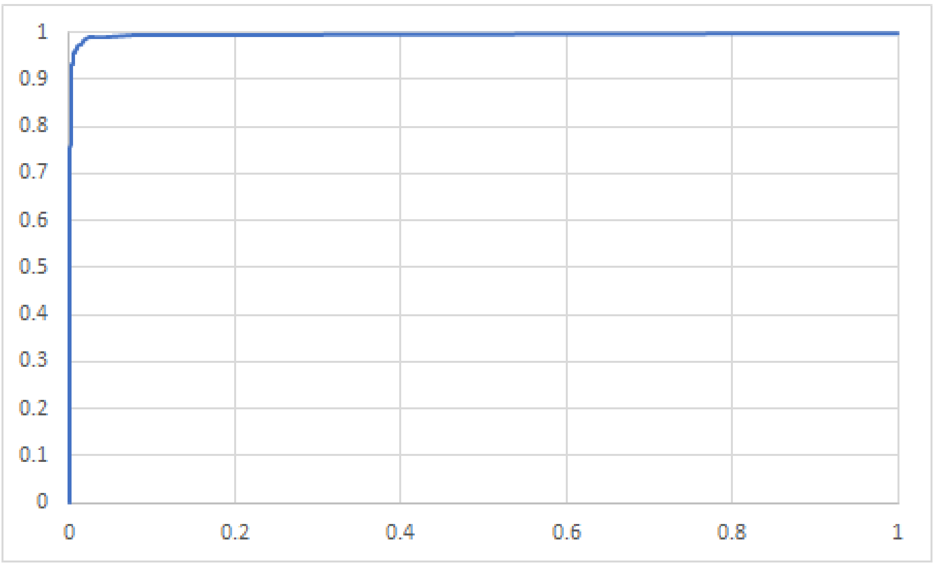


Fig. 5.3 The ROC curve of Random Forest affiliation binary classifier with all features

5.1.4 Section heading classification

Our binary classifier that does section heading classification takes a paragraph of texts from manuscripts as input and predict it as a "section heading" or "normal text". In our experiment, we get the best performance with our Random Forest classifier. Table 5.4 shows precision, recall and F-score we have got from different groups of features.

Table 5.4 Performance of binary section heading classifier - Random Forest

Feature	precision	recall	F-score
shallow-textual	0.963	0.938	0.950
syntactic-textual	0.558	0.911	0.692
shallow, syntactic-textual	0.938	0.966	0.952
visual	0.606	0.645	0.625
all	**0.967**	**0.968**	**0.967**

Figure 5.4 demonstrates the ROC curve of Random Forest model with all features.

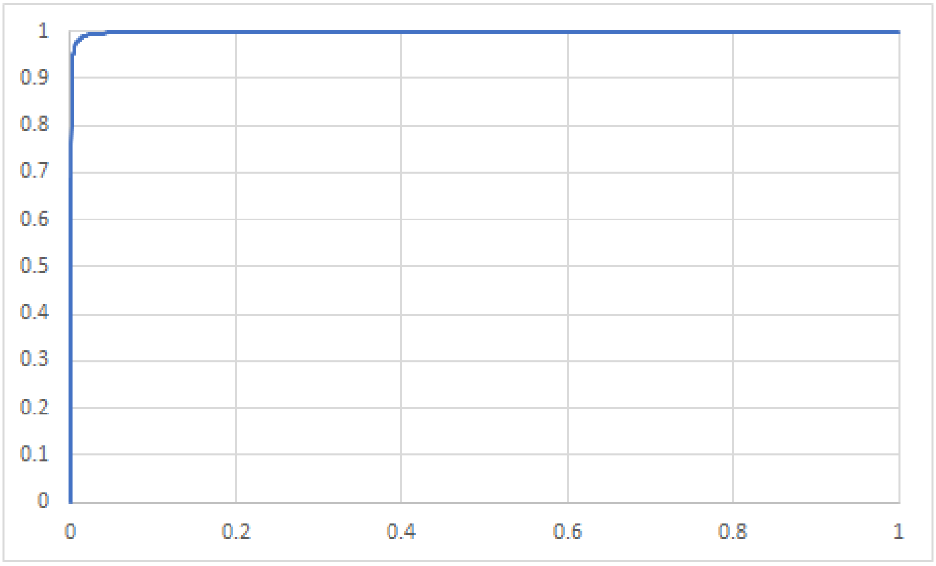


Fig. 5.4 The ROC curve of Random Forest section heading binary classifier with all features

5.1.5 Table and figure caption classification

Our binary classifier that does caption classification takes a paragraph of texts from manuscripts as input and predict it as a "caption" or "normal text". In our experiment, we get the best performance with our Random Forest classifier. Table 5.5 shows precision, recall and F-score we have got from different groups of features.

Figure 5.5 demonstrates the ROC curve of Random Forest model with all features.

5.1.6 Reference list item classification

Our binary classifier that does reference list item classification takes a paragraph of texts from manuscripts as input and predict it as a "reference" or "normal text". In our experiment,

Table 5.5 Performance of binary caption classifier - Random Forest

Feature	precision	recall	F-score
shallow-textual	0.950	0.942	0.946
syntactic-textual	0.373	0.545	0.443
shallow, syntactic-textual	0.968	0.946	0.957
visual	0.361	0.452	0.401
all	**0.978**	**0.951**	**0.964**

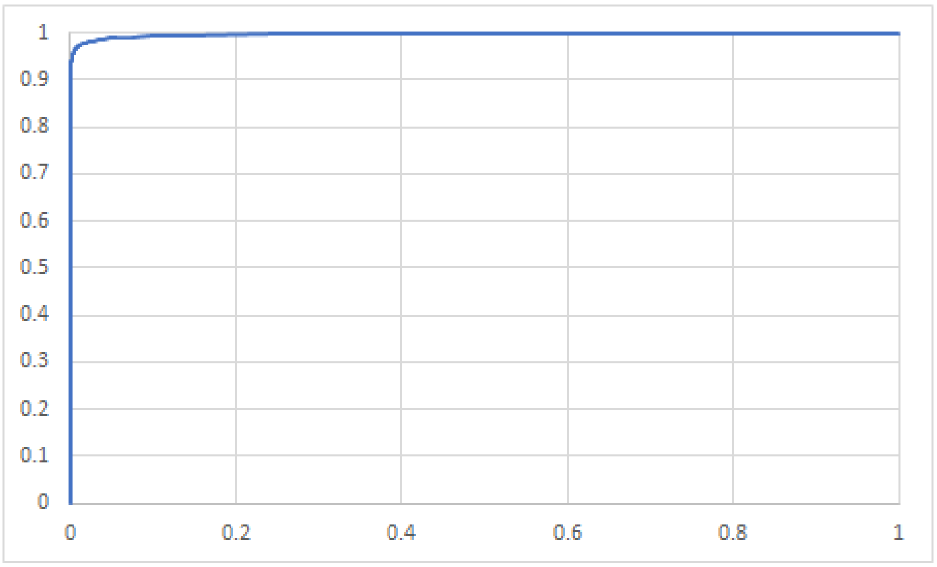


Fig. 5.5 The ROC curve of Random Forest caption binary classifier with all features

Table 5.6 Performance of binary reference classifier - Random Forest

Feature	precision	recall	F-score
shallow-textual	0.963	0.975	0.969
syntactic-textual	0.898	0.904	0.901
shallow, syntactic-textual	0.967	0.978	0.972
visual	0.584	0.559	0.571
all	**0.972**	**0.976**	**0.974**

we get the best performance with our Random Forest classifier. Table 5.6 shows precision, recall and F-score we have got from different groups of features.

Figure 5.6 demonstrates the ROC curve of Random Forest model with all features.

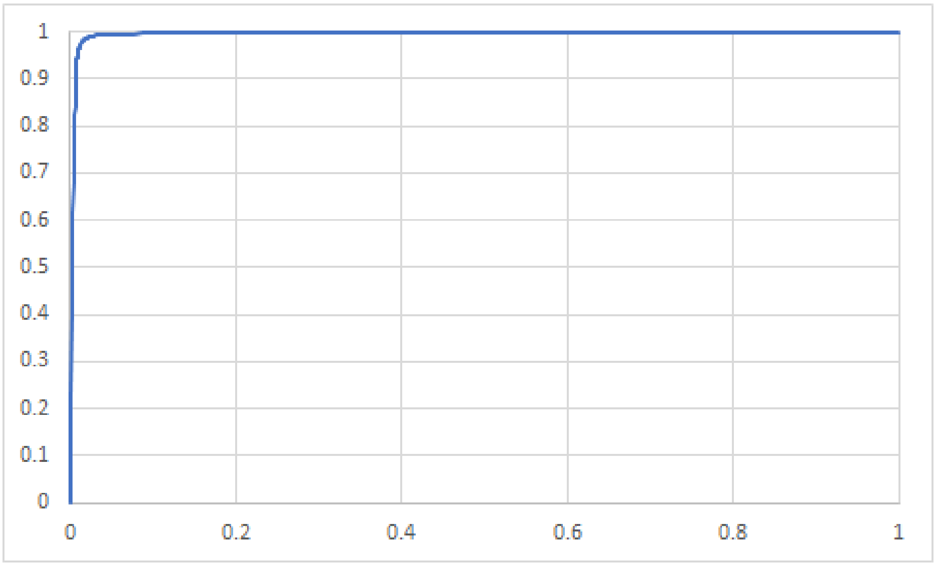


Fig. 5.6 The ROC curve of Random Forest reference binary classifier with all features

5.2 Multi-class classifier for all document structure types

In the real scenario of the Apollo project, it is not very meaningful if we predict a content instance to different document structure entity separately. For instance, with all of our binary classifiers, a paragraph is possible to end up with possibility being both section heading and caption item, and then we still fail extracting the real structure type of this element because it should only belong to one category. Hence, we continue our experiments by implementing a multiple-class classifier, which is able to take in one paragraph, consider its probabilities versus all document structure types and export the most likely category.

To tackle a multi-class classification problem, there are two strategies with which we can narrow down the problem, "1-against-all" and "1-against-1". To illustrate, when it comes to "1-against-all" strategy, which is also known as "1-against-rest", based on the observations from original data set, one classifier will be built for each class, where such a certain class is fitted against all the other classes. After all classifiers are created, the target class is obtained by the classifier with the highest positive probability. Instead, for the "1-against-1" strategy, classifiers are trained for each pair of classes. And when a new sample is predicted, the class receiving the most votes will be chosen and applied as the prediction result.

In our multi-class classification experiment, several categories of models are tested, which are the same as what we test in previous binary classifier experiments. Similarly, we also conduct experiments and investigate how the performance with different groups of features differs. As a result, we record precision, recall and f scores for each possible feature-model combination. Furthermore, for all models except multilayer perceptron, we apply both "1-against-1" and "1-against-all" strategy and figure out which one performs better in a certain scenario, and we record the better result in our report together with the strategy behind it. It should be noted that multilayer perceptron is natively designed to support multi-class classification, and it is the reason why we do not decompose the problem with different strategies for this model.

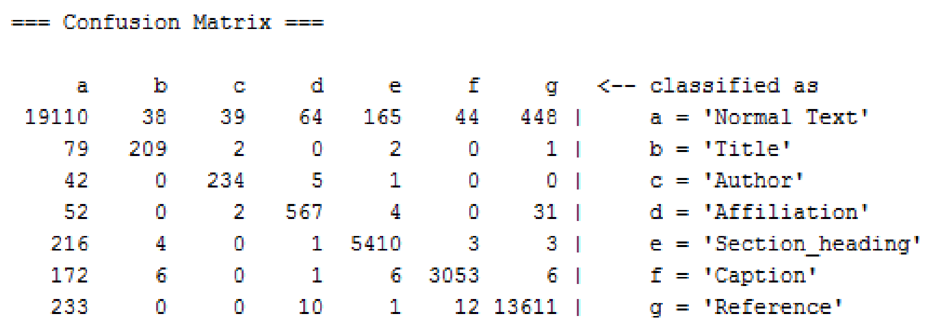
Considering the feasibility of displaying results we have got, in this section, we choose to show the performance with all implemented features, because generally it returns the best performance when compared with others. Table 5.7, Table 5.8 as well as Table 5.9 specifically records the precision, recall and f score of our experiments.

Since F1 score takes both precision and recall score into consideration and get a overall performance score for our model, we can notice that our machine learning model based on Random Forest returns us the best general performance for all classes, and then we put more details of our experiments with this specific model. As shown in Table 5.10, Table 5.11 and Table 5.12, we records the results of our experiment with different feature groups.

As the original information from which the precision, recall and F1 score of our models can be computed, confusion matrix [52] visualizes the number of instances that is either correctly classified or predicted to other classes. Figure 5.7 shows the confusion matrix we obtained from our random forest multi-class classifier.

Table 5.7 Precision scores of multi-class classifier with all features

Model	Title	Author	Affiliation	Section Heading	Caption	Reference	General
Naive Bayes (one-vs-all)	0.183	0.169	0.528	0.679	0.852	0.951	0.871
Multilayer Perceptron (MLP)	0.552	0.705	0.763	0.944	0.972	0.947	0.946
Decision Stump (one-vs-one)	0	0	0.794	0.911	0.724	0.880	0.843
REPTree (one-vs-one)	0.756	0.820	0.841	0.958	0.975	0.960	0.947
Random Forest (one-vs-one)	**0.813**	**0.845**	0.875	**0.968**	0.981	**0.965**	**0.961**
Random SubSpace (one-vs-one)	0.792	**0.845**	**0.895**	0.949	**0.983**	0.959	0.952
LogitBoost (one-vs-one)	0.808	0.798	0.861	0.951	0.956	0.947	0.936
Sequential Minimal Optimization (SMO) (one-vs-one)	0.649	0.726	0.873	0.956	0.959	0.919	0.910



```
=== Confusion Matrix ===

     a     b     c     d     e     f     g   <-- classified as
 19110    38    39    64   165    44   448 |     a = 'Normal Text'
    79   209     2     0     2     0     1 |     b = 'Title'
    42     0   234     5     1     0     0 |     c = 'Author'
    52     0     2   567     4     0    31 |     d = 'Affiliation'
   216     4     0     1  5410     3     3 |     e = 'Section_heading'
   172     6     0     1     6  3053     6 |     f = 'Caption'
   233     0     0    10     1    12 13611 |     g = 'Reference'
```

Fig. 5.7 The confusion matrix for our random forest multi-class classifier

Table 5.8 Recall scores of multi-class classifier with all features

Model	Title	Author	Affiliation	Section Heading	Caption	Reference	General
Naive Bayes (one-vs-all)	**0.782**	0.794	0.837	0.926	0.760	0.907	0.824
Multilayer Perceptron (MLP)	0.778	0.770	0.803	0.952	0.928	0.976	0.944
Decision Stump (one-vs-one)	0	0	0.229	0.746	0.940	0.961	0.851
Random Forest (one-vs-one)	0.713	**0.830**	**0.864**	**0.960**	**0.941**	**0.982**	**0.961**
REPTree (one-vs-one)	0.710	0.727	0.771	0.946	0.933	0.963	0.947
Random SubSpace (one-vs-one)	0.638	0.755	0.764	0.953	0.926	0.979	0.953
LogitBoost (one-vs-one)	0.676	0.730	0.745	0.925	0.920	0.960	0.936
Sequential Minimal Optimization (SMO) (one-vs-one)	0.693	0.695	0.692	0.930	0.916	0.902	0.910

Table 5.9 F1 scores of multi-class classifier with all features

Model	Title	Author	Affiliation	Section Heading	Caption	Reference	General
Naive Bayes (one-vs-all)	0.296	0.278	0.647	0.784	0.803	0.929	0.838
Multilayer Perceptron (MLP)	0.646	0.736	0.782	0.948	0.949	0.961	0.945
Decision Stump (one-vs-one)	0	0	0.355	0.821	0.818	0.919	0.842
Random Forest (one-vs-one)	**0.760**	**0.837**	**0.870**	**0.964**	**0.961**	**0.973**	**0.961**
REPTree (one-vs-one)	0.732	0.771	0.804	0.952	0.954	0.961	0.947
Random SubSpace (one-vs-one)	0.707	0.798	0.824	0.951	0.953	0.969	0.952
LogitBoost (one-vs-one)	0.736	0.763	0.799	0.938	0.938	0.953	0.936
Sequential Minimal Optimization (SMO) (one-vs-one)	0.670	0.710	0.772	0.943	0.937	0.911	0.910

Table 5.10 Precision scores of Random Forest multi-class classifier with different sets of features

Feature	Title	Author	Affiliation	Section Heading	Caption	Reference	General
shallow-textual	0.620	0.812	0.850	0.955	0.964	0.959	0.948
syntactic-textual	0.139	0.344	0.347	0.584	0.494	0.883	0.789
shallow& syntactic-textual	0.689	0.820	0.856	0.958	0.979	0.962	0.955
visual	0.180	0.216	0.325	0.609	0.402	0.580	0.568
all	**0.813**	**0.845**	**0.875**	**0.968**	**0.981**	**0.965**	**0.961**

Table 5.11 Recall scores of Random Forest multi-class classifier with different features

Feature	Title	Author	Affiliation	Section Heading	Caption	Reference	General
shallow-textual	0.546	0.734	0.802	0.937	0.934	0.976	0.948
syntactic-textual	0.109	0.199	0.323	0.831	0.355	0.906	0.785
shallow& syntactic-textual	0.659	0.741	0.806	0.950	0.938	0.979	0.955
visual	0.164	0.177	0.305	0.623	0.379	0.556	0.568
all	**0.713**	**0.830**	**0.864**	**0.960**	**0.941**	**0.982**	**0.961**

Table 5.12 F1 scores of Random Forest multi-class classifier with different features

Feature	Title	Author	Affiliation	Section Heading	Caption	Reference	General
shallow-textual	0.581	0.771	0.825	0.946	0.949	0.967	0.948
syntactic-textual	0.122	0.252	0.335	0.686	0.413	0.894	0.783
shallow& syntactic-textual	0.674	0.778	0.830	0.954	0.958	0.970	0.955
visual	0.172	0.195	0.314	0.616	0.390	0.567	0.567
all	**0.760**	**0.837**	**0.870**	**0.964**	**0.961**	**0.973**	**0.961**

Chapter 6

Discussion

In this chapter we will have a discussion about the results from our experiments, as shown in the previous chapter. Firstly, we will have a high-level overview towards our results for both binary classifiers and multi-class classifiers, followed by a brief discussion how different category of features perform differently towards different tasks. After this, we will have a discussion what challenges we have met during the experiment process and how they are overcome.

6.1 Binary classification

From the F1-score of our previous experiments of binary classification, we can have an overview how our binary classifiers perform on each sub task. At a first glance, we obtained better performance with our ensemble classifiers, logitboost, random subspace and random forest, among which random forest model outperformed the other two. In general, it is impossible to regard a certain model always performs better than the other, and any machine learning classifier holds their advantages towards data set with some certain characteristics. As we previously mentioned in the "Method" chapter, ensemble models are proposed to make use of the advantages of available machine learning algorithm and achieve overall better performance. Many researches have also been done to compare the performance of different models in diverse contexts or situations, and some of them have emphases on random forest [29] [24].

If we have a closer look at the advantages of random forest and the data we have, we consider the following attributes distinguish our random forest models from the others and achieve the best performance:

1. **Random forest does not require specific preprocessing of data.** In our features extracted from manuscripts, they are not strictly normalized. In other words, though values of the same feature is comparable between different instances, however, for each instance, different features values are not aligned to the same benchmark. For instance, the average font-size is a real number between 0 and 1, while the value of feature paragraph_length can expand from 1 to hundred. Without proper preprocessing of data, the performance of models like SVM or MLP may be affected. Nevertheless, random forest models have no similar requirements, because they just split a node based on the distribution of feature values.

2. **Random forest is robust against overfitting.** By randomly picking a number of features to construct base estimator, random forest naturally prevent creating too complicated models, especially when we did not explicitly employ other methods to prevent overfitting in our experiments.

3. **Our features have correspondence with others.** In our experiments, our features are proposed as we mentioned in the previous "Method" chapter, and some of them are not strictly independent with others. For instance, the value of feature "end with dot" cannot be True for those data point whose "end with colon" is True. Similarly, if we sum up the ratio of all POS tag in our features for each paragraph, the result will always be 1. As some correspondence can be found in our features, it is then reasonable why statistic models like Naive Bayes perform poorly, where the independence of different features is expected when the model is trained.

With the best performance we obtained with our random forest models, by comparing the F1-score of models trained with different sets of features, we can identify a general order for those performance as shallow&syntactic-textual, shallow-textual, syntactic-textual and visual, where the former set of features perform better than the latter one. From the results we can identify that most of information for our document structure extraction task can be obtained through different levels of textual features. In other words, in the situation where limited or even no visual markup information is provided, we can still extract needed document structure information with good performance. Nevertheless, by including other visual features, the overall performance is improved because additional information is introduced. And there is still room to improve the performance.

From the results of our binary classification experiments, we can tell that the performance of our models vary quite a lot over different tasks, with lowest F1-score 76.5% for title extraction and highest F1-score 98.8% for reference extraction task. Considering all the features we have proposed, we can find this tendency is reasonable. For instance, several features aim at distinguishing captions from the rest, such as "in_tf_gazetteer" and "begin_with_tf". Though a few captions from manuscripts only contain contents illustrating images or tables, however, most of them are provided by authors with words "Fig" or "Tab" prefixed at the beginning, but those words are not expected in any element with other document structure type. As for reference items, tokens inside are mostly capitalized, which is reflected in our "capital_percentage" feature. Those features all provide crucial information to help us extract captions or references from articles. On the contrary, as the title of an article depends heavily on its content and no specific format or structure can be expected, it is then not that easy to find such a feature that is able to separate most titles apart from the other entities. From this point of view, it makes sense that our models outperform on the caption or reference extraction task and perform poorer for the title extraction.

By comparing the performance of models trained with different sets of features, we can also see how each set of features perform on each subtask. For instance, when only visual features are introduced, we can see the performance of our models vary greatly on different subtasks, where we get the best performance on the reference extraction task but worst result on the title task. Since our visual features cover information including font-size, bold, italic, subscript, superscript and underline, it makes sense that we get more information for references, where texts are likely to be marked with bold or italic styles. However, the result for title task is not consistent with our expectation, where the font-size should provide much information because titles are usually set with the biggest font-size in an article. In our implementation of the font-size feature, considering authors may have their own preference in the choices of actual font-size, instead of real values of font-size, we adopt relative font-size, which is obtained by dividing the real font-size with average font-size in the same document. Though the value is normalized by average font-size in the same document, however, authors still have much freedom to select the margin between the font-size of different elements. For example, some authors may set up font-size for titles as only 150% of normal text, while others are possible to double the size of normal text and apply it to title. From this aspect, our "relative" font-size is still not aligned between different documents and authors. It explains the reason why our models perform worse than our expectation on title extraction task when only visual features are applied.

Through the ROC curves we plotted for each task, we can also have a visualized performance of our models. For a machine learning model, there are two important measures, True Positive (TP) and False Positive (FP) rate. Given all the instances that are classified as positive class for the classifier, TP rate is able to reveal the percentage of instances that are correctly classified, while the FP rate represents the ratio of misclassified instances. Since precision and recall score of a model can be calculated as

$$Precision = \frac{TP}{TP+FP} = \frac{1}{1+\frac{FP}{TP}}$$

$$Recall = \frac{TP}{TP+FN} = \frac{TP}{P}$$

where FN represents the number of instances that are misclassified as negative class. From the above formulas, if a model pursues higher precision score, then it should have higher TP rate but lower FP rate. Similarly, since the number of positive instances is fixed when we evaluate our models, high recall score can only be achieved when a model can get good TP rate. As a result, in a ROC curve, we should expect a high TP rate without sacrificing increasing FP too much. In our ROC curve, we can see our models are able to get good TP rate and still keep the FP rate as a small value. In general, the area under a ROC curve is able to reveal the performance of a model, and the area under our ROC curves for different models also cover most part of the graph, which is also consistent with the results we got from the experiments.

6.2 Multi-class classification

Through the F1-score table of our multi-class classifiers shown in the tables in previous chapter, we can get an overview how our features contribute to the multi-class classification problem. In the results shown by table 5.9, for each document structure type, we highlight the best performance with bold. We can see our Random Forest models outperform all the rest. This is consistent with the results we have gained in previous binary classification experiments.

In our results of multi-class classifiers, we can also compare the precision, recall and F1-score of models versus different features. In the table 5.12, we can see how each set of features perform on different tasks. If we horizontally compare results in the same row, we can get an initial results how elements with different document structure types benefits from different sets of features. For instance, our visual features perform best on the section heading task, which proves all the visual information extracted from manuscripts are useful to distinguish section heading with the rest. As section headings are mostly marked as bold style and without any subscript or superscript, it is then reasonable that our visual features are able to separate them from other elements. Similarly, in our case of syntactic-textual, we can see it perform worst in tasks like title, section heading and caption. Our current syntactic-textual features contain limited information like percentage of different POS tag in a single paragraph, and from this aspect, there is no significant difference between titles, section heading, caption or normal content in the body of the article. As a result, it makes sense that they perform poorly to distinguish titles, section headings and captions from the other types of document structure, where all the POS tags can be expected to be found. On the contrary, given a typical reference like the following paragraph:

"Matthewsa, K. K., O Briena, D. J., Whitley, N. C., Burkec, J. M., Millerd, J. E. & Barczewski, R. A. (2016). Investigation of possible pumpkin seeds and ginger effects on-gastrointestinal nematode infection indicators in meat goat kids and lambs. Small Ruminant Research, 136, 1-6.",

we can expect that most tokens in this paragraph are noun phrase like author names. Besides, the existence of some conjunctions and adjectives can also be expected because title of an article or journal is sometimes also mentioned in an reference item. It is also reflected in the results of our experiments, where syntactic-textual features can perform the best versus the rest types of document structure.

Through the results we collected from our multi-class classification experiment, we can also tell the difference between two different strategies, one-vs-one (OVO) and one-vs-all(OVA). In most of our models, the OVO strategy outperformed the OVA strategy. The comparisons between these two strategies are also discussed by many researchers, in terms of the number of class in the classification problem [34], or the characteristic of data set [13]. For OVA strategy, by decomposing the classification problem as one category versus all the others, a larger number of instances from different classes are considered, and in most of case they produce more complexity to the basic classification problem. Besides, as our data set is unbalanced, by considering larger size of instances, the gap is even bigger than the OVO strategy where only instances from two classes will be taken, and it also makes the basic classification problem much harder. Moreover, we also noticed a significant difference between the time consumed to build classifiers with these two strategies, where models based on OVO took much less time. Given n classes classification problem, though OVO strategy requires $n(n-1)/2$ discriminants but OVA only demands n of them, the time taken to build each single binary classifier has great difference. In the case of OVO, much fewer of instances will be used to train the model, yet OVA-based model will take all the instance every time. From this point of view, this explains why models with OVA strategy took more time for constructing, and it is especially obvious when the base classifier is not well scalable with the number of instances, such as SVM.

Moreover, as the confusion matrix shown in the Figure 5.7, we can tell that most misclassification occurred between a certain category of document structure element and normal text. If we have a look at the confusion matrix between different classes except normal text, we can see our models performed well distinguishing each category from the others.

6.3 What challenges have been solved

In this section, we want to have a brief discussion of the challenges we mentioned in the first chapter and how they are solved.

At first, when we align the content from manuscripts to published articles, we utilized the text similarity to draw the conclusion either two elements are corresponding or not. In this way, the original elements from manuscripts should always find the most similar content in the published articles and be marked with the correct labels. In fact, after our data set was created with the method we proposed, we randomly sampled some instances and manually check the label attached with them. After our investigation, we are confident that our method works well in the most of cases, though we also noticed a few typical mistakes. For instance, as article title can be provided more than once in the whole manuscript, with our greedy solution, only one of them can be correctly identified and marked as positive title class, while the rest will be marked as normal texts. After we identify this issue, we took another step to further clean our data set, where we eliminated the conflicts of labels for duplicate elements. After all the processes are finished, we are confident that no information from the manuscripts is mission or wrong and take this data set for our experiments.

Secondly, for the unbalanced data set issue, though the results of our experiments, we can tell that assigning different cost weight to the classifiers does help our models make correct predictions. Even for the document structure entities like title and author group, which are extremely unbalanced in our data set, our classifiers do not ignore the minority and still return reasonable performance. Furthermore, as machine learning models always make trade-off between precision and recall, we will have a more balanced view when these two numbers are close. Hence, through the result of our experiments, we can see our models somehow reach the limit on their performance, unless new information or feature is introduced in the future.

In addition to the unbalanced data set, we also met another issue when we created our data set. As we obtain the label of each paragraph by aligning content in the original manuscripts with published article, we cannot expect a certain format existing in manuscripts. For example, it is possible that such a paragraph is manually split into multiple lines by authors, even though they work as one single entity and should stay together. Based on the specific purpose of classifiers, this issue can be tackled in different ways, like taking subsequent content elements and merging them as the input. In our Apollo scenario, by aligning the content for entire paragraph and part of the paragraph, we design our classifiers to be able to guess the document structure type of a complete paragraph or part of it. In this way, we have some subsequent concerns:

1. When creating the data set with correct labels for both a complete paragraph and part of a paragraph, have we obtained correct label for all of them?
2. When extracting features from a sentence, how to deal with incomplete sentence when part of a paragraph is introduced?

The above two aspects may have crucial effects on the performance of our classifiers. At first, if we omit some parts of a paragraph, it will introduce a lot of noise to our data set. In this case, our classifiers may get confused at both training and evaluating processes and probably end up with poor performance. Secondly, for some of our features, such as those involving POS tag analysis, we may have problem with them if we analyze an incomplete sentence. In this case, how this issues can be worked around and produce some meaningful feature values will be crucial to the performance of our models, especially when we have a lot of incomplete paragraph instances. At this moment, we tackle the first challenge by marking the entire paragraph or the first part of the paragraph as positive class, while the other elements are all negative. In this way, we keep the consistency in our training and testing set and make sure the experiment results actually reveal the true performance of our models. However, for the second issue, we have not come up with any specific solution for it yet, but it remains some hints how the performance of our machine learning classifiers can be improved.

Chapter 7

Conclusion and Future Works

In this thesis, we mainly illustrate the process how we decompose our document structure extraction task to several smaller-grained modules, including processing data, feature extraction and training machine learning models. With those works, we want to show an end-to-end work flow how we start from the original manuscript files and end up with structured content with document structure information. Here in the "Conclusion" section, we explain how our research questions are answered. And at the end, we describe some future works that can be included in our work and probably improve our performance.

7.1 Conclusion

For our main research question, "When limited markup or visual information is provided in an article, how can we apply machine learning approaches to extract document structure from it, such as title, author group?", the quick answer is that we can still make use of other text-based information and get good performance for this task. In our experiment, by comparing the performance of machine learning models trained with different groups of features, we showed that classifiers trained with different level of textual features returned reasonable performance in our document structure extraction tasks, and actually they performed much better than models trained with only visual markup features. From another aspect, by including both textual and visual information collected from manuscript files, our machine learning models return us the best performance. In other words, different categories of information are able to complement each other and achieve a better overall result.

As for our subquestions, we also provide some insights or answers to them in our experiments. At first, for the question "How can we collect information from all kinds of files, including DOCX, JPG, etc.", we answer this question by proposing a Structured Document Format (SDF). By converting our original documents to our SDF format, we are able to gather all article content distributed into several document instances and then extract the document structure information for the whole article. With our SDF format, we also want to give some insights how information can be aggregated from different files in other similar scenarios.

Secondly, for another question regarding the specific scenario in our Apollo project, "How can we create a data set to get our machine learning models?", we show an approach how we obtain our data set by aligning original manuscripts and final well-formated version of previous published articles. By discussing this process, we also want to show some insights how available resources can be reused and new data set can be created in other similar scenarios.

Besides, by proposing three different categories of features, we answered the question "What information can we extract and apply in our machine learning models?". In the results of our experiments, we can see that all categories of features provide information from which our models can benefit to detect the document structure behind. Furthermore, as shown in our results, without conducting complicated syntactic or semantic analysis on the text, we already get most of the information.

All in all, we have presented the whole work flow how we begin with manuscript documents, collect information from them and at the end extract six subjects of document structure entities from them, including title, author group, affiliation information, section heading, caption and reference list item. Though our machine learning models perform worse for tasks like title, author and affiliation extraction, however, we consider this results still acceptable in current state of our Apollo project. As title is always expected as the first element in a document, and authors together with affiliations will mostly follow the title in the first page, it is potential to get better results with additional positions or location information included.

7.2 Future work

We have seen that our machine learning classifiers can give us reasonable performance for document structure extraction task, but there is still room to pursue a better performance. In this section, we put some ideas how it can be improved in the future.

During our experiments, we extract both visual and textual information from original manuscripts and apply them to our machine learning models. In total, we have used 6 visual features, 8 syntactic-textual features and 13 shallow-textual features to feed our machine learning models. Yet for each group of features, there is much more that can be explored. For instance, information like "Term Frequency (TF)", "Inverse Document Frequency (IDF)" is sometimes used to extract keyphrases from scientific papers [5]. Currently, we only make use of local observables information that is only extracted from a piece of texts itself, but a lot of information still exists globally and they can also contribute to this information extraction task. For example, the position information in a document can be an interesting one, especially for title extraction task, where titles are normally expected at the beginning of each document. Expanding the size of available features can be one direction how we can move further in our document structure extraction task.

In our experiments, we tried out several combinations of features and machine learning models. However, with current development of word-embedding technology, researchers have proposed ways to get vector form of words, where similarity between different words can be obtained by calculating the distance between different vectors [32]. Nevertheless, as such a word-embedding model is trained from texts in a certain domain, it may perform poor when it meets new unseen words which are from outside of the corpus. From another aspect, training such a word-embedding model normally requires a lot of data. As Elsevier has a vast number of previous paper data, it is also possible to train such a word2vec model for some specific area or domain. We think it will be interesting to conduct some experiments to investigate how word2vec model can perform on this document structure extraction task. And some pretrained model like Google word2vec trained from News can be a good start [31] [33].

References

[1] Athar, A. (2011). Sentiment analysis of citations using sentence structure-based features. In *Proceedings of the ACL 2011 student session*, pages 81–87. Association for Computational Linguistics.

[2] Bishop, C. M. (2006). Pattern recognition. *Machine Learning*, 128:1–58.

[3] Bray, T., Paoli, J., Sperberg-McQueen, C. M., Maler, E., and Yergeau, F. (1997). Extensible markup language (xml). *World Wide Web Journal*, 2(4):27–66.

[4] Cahill, L., Doran, C., and Evans, R. (1999). *Towards a reference architecture for natural language generation systems.*

[5] Constantin, A. (2014). Automatic structure and keyphrase analysis of scientific publications.

[6] Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine learning*, 20(3):273–297.

[7] Councill, I. G., Giles, C. L., and Kan, M.-Y. (2008). Parscit: an open-source crf reference string parsing package. In *LREC*, volume 2008.

[8] Draper, N. R., Smith, H., and Pownell, E. (1966). *Applied regression analysis*, volume 3. Wiley New York.

[9] Edmonds, J. (1965). Maximum matching and a polyhedron with 0, 1-vertices. *Journal of Research of the National Bureau of Standards B*, 69(125-130):55–56.

[10] Forsyth, D. and Ponce, J. (2011). *Computer vision: a modern approach.* Upper Saddle River, NJ; London: Prentice Hall.

[11] Friedman, J., Hastie, T., Tibshirani, R., et al. (2000). Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). *The annals of statistics*, 28(2):337–407.

[12] Fürnkranz, J. and Widmer, G. (1994). Incremental reduced error pruning. In *Proceedings of the 11th International Conference on Machine Learning (ML-94)*, pages 70–77.

[13] Galar, M., Fernández, A., Barrenechea, E., Bustince, H., and Herrera, F. (2011). An overview of ensemble methods for binary classifiers in multi-class problems: Experimental study on one-vs-one and one-vs-all schemes. *Pattern Recognition*, 44(8):1761–1776.

[14] Goldfarb, C. F. and Rubinsky, Y. (1990). *The SGML handbook*. Oxford University Press.

[15] Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. H. (2009). The weka data mining software: an update. *ACM SIGKDD explorations newsletter*, 11(1):10–18.

[16] Han, H., Giles, C. L., Manavoglu, E., Zha, H., Zhang, Z., and Fox, E. A. (2003). Automatic document metadata extraction using support vector machines. In *Digital Libraries, 2003. Proceedings. 2003 Joint Conference on*, pages 37–48. IEEE.

[17] Ho, T. K. (1995). Random decision forests. In *Document Analysis and Recognition, 1995., Proceedings of the Third International Conference on*, volume 1, pages 278–282. IEEE.

[18] Ho, T. K. (1998). The random subspace method for constructing decision forests. *IEEE transactions on pattern analysis and machine intelligence*, 20(8):832–844.

[19] Iba, W. and Langley, P. (1992). Induction of one-level decision trees. In *Proceedings of the ninth international conference on machine learning*, pages 233–240.

[20] Jewell, M. (2000). Paracite: An overview.

[21] Jochim, C. and Schütze, H. (2012). Towards a generic and flexible citation classifier based on a faceted classification scheme.

[22] Jurafsky, D. and James, H. (2000). Speech and language processing an introduction to natural language processing, computational linguistics, and speech.

[23] Karp, R. M. (1972). Reducibility among combinatorial problems. In *Complexity of computer computations*, pages 85–103. Springer.

[24] Khoshgoftaar, T. M., Golawala, M., and Van Hulse, J. (2007). An empirical study of learning from imbalanced data using random forest. In *Tools with Artificial Intelligence, 2007. ICTAI 2007. 19th IEEE International Conference on*, volume 2, pages 310–317. IEEE.

[25] Kim, J., Le, D. X., and Thoma, G. R. (2000). Automated labeling in document images. In *Photonics West 2001-Electronic Imaging*, pages 111–122. International Society for Optics and Photonics.

[26] Klink, S., Dengel, A., and Kieninger, T. (2000). Document structure analysis based on layout and textual features. In *Proc. of International Workshop on Document Analysis Systems, DAS2000*, pages 99–111. Citeseer.

[27] Kuhn, H. W. (1955). The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97.

[28] Li, S., Gao, L., Tang, Z., and Yu, Y. (2015). Cross-reference identification within a pdf document. In *SPIE/IS&T Electronic Imaging*, pages 940209–940209. International Society for Optics and Photonics.

[29] Liu, M., Wang, M., Wang, J., and Li, D. (2013). Comparison of random forest, support vector machine and back propagation neural network for electronic tongue data classification: Application to the recognition of orange beverage and chinese vinegar. *Sensors and Actuators B: Chemical*, 177:970–980.

[30] Mann, W. C. and Thompson, S. A. (1988). Rhetorical structure theory: Toward a functional theory of text organization. *Text-Interdisciplinary Journal for the Study of Discourse*, 8(3):243–281.

[31] Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient estimation of word representations in vector space. *arXiv:1301.3781.*

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

[33] Mikolov, T., Yih, W.-t., and Zweig, G. (2013c). Linguistic regularities in continuous space word representations. In *Hlt-naacl*, volume 13, pages 746–751.

[34] Milgram, J., Cheriet, M., and Sabourin, R. (2006). "one against one" or "one against all": Which one is better for handwriting recognition with svms? In *Tenth international workshop on frontiers in handwriting recognition*. Suvisoft.

[35] Nakagawa, K., Nomura, A., and Suzuki, M. (2004). Extraction of logical structure from articles in mathematics. In *International Conference on Mathematical Knowledge Management*, pages 276–289. Springer.

[36] Nguyen, T. D. and Luong, M.-T. (2010). Wingnus: Keyphrase extraction utilizing document logical structure. In *Proceedings of the 5th international workshop on semantic evaluation*, pages 166–169. Association for Computational Linguistics.

[37] Nunberg, G. (1990). *The linguistics of punctuation*. Number 18. Center for the Study of Language (CSLI).

[38] Opitz, D. and Maclin, R. (1999). Popular ensemble methods: An empirical study. *Journal of Artificial Intelligence Research*, 11:169–198.

[39] Pang, B., Lee, L., and Vaithyanathan, S. (2002). Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 79–86. Association for Computational Linguistics.

[40] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. (2011). Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12(Oct):2825–2830.

[Peng and McCallum] Peng, F. and McCallum, A. Accurate information extraction from research papers using conditional random fields. retrieved on april 13, 2013.

[42] Power, R., Scott, D., and Bouayad-Agha, N. (2003). Document structure. *Computational Linguistics*, 29(2):211–260.

[43] Powers, D. M. (2011). Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation.

[44] Raggett, D., Le Hors, A., Jacobs, I., et al. (1999). Html 4.01 specification. *W3C recommendation*, 24.

[45] Rosenblatt, F. (1961). Principles of neurodynamics. perceptrons and the theory of brain mechanisms. Technical report, DTIC Document.

[46] Russell, S., Norvig, P., and Intelligence, A. (1995). A modern approach. *Artificial Intelligence. Prentice-Hall, Egnlewood Cliffs*, 25:27.

[47] Samuel, A. L. (1959). Some studies in machine learning using the game of checkers. *IBM Journal of research and development*, 3(3):210–229.

[48] Schäfer, U. and Weitz, B. (2012). Combining ocr outputs for logical document structure markup: technical background to the acl 2012 contributed task. In *Proceedings of the ACL-2012 Special Workshop on Rediscovering 50 Years of Discoveries*, pages 104–109. Association for Computational Linguistics.

[49] Scott, D. and Power, R. (1998). Generating textual diagrams and diagrammatic texts. In *International Conference on Cooperative Multimodal Communication*, pages 13–29. Springer.

[50] Seymore, K., McCallum, A., and Rosenfeld, R. (1999). Learning hidden markov model structure for information extraction. In *AAAI-99 workshop on machine learning for information extraction*, pages 37–42.

[51] Sperberg-McQueen, C. M., Burnard, L., et al. (1994). *Guidelines for electronic text encoding and interchange*, volume 1. Text Encoding Initiative Chicago and Oxford.

[52] Stehman, S. V. (1997). Selecting and interpreting measures of thematic classification accuracy. *Remote sensing of Environment*, 62(1):77–89.

[53] Summers, K. (1995). Toward a taxonomy of logical document structures. In *Proceedings of DAGS*, volume 95, pages 124–133. Citeseer.

[54] Vapnik, V. N. and Vapnik, V. (1998). *Statistical learning theory*, volume 1. Wiley New York.