A Hybrid Approach to Discover Semantic Hierarchical Sections in Scholarly Documents

Suppawong Tuarob
Information and Communication Technology
Mahidol University, Thailand
Email: suppawong.tuarob@gmail.com

Prasenjit Mitra
Qatar Computing Research Institute
Doha, Qatar
Email: pmitra@qf.org.qa

C. Lee Giles
Computer Science and Engineering
Information Sciences and Technology
Pennsylvania State University, USA
Email: giles@ist.psu.edu

***Abstract*—Scholarly documents are usually composed of sections, each of which serves a different purpose by conveying specific context. The ability to automatically identify sections would allow us to understand the semantics of what is different in different sections of documents, such as what was in the introduction, methodologies used, experimental types, trends, etc. We propose a set of hybrid algorithms to 1) automatically identify section boundaries, 2) recognize standard sections, and 3) build a hierarchy of sections. Our algorithms achieve an F-measure of 92.38% in section boundary detection, 96% accuracy (average) on standard section recognition, and 95.51% in accuracy in the section positioning task.**

I. Introduction

A scholarly document is typically organized into various sections. If we can automatically identify and separate these sections and their content, we can then apply them in knowledge extraction and document analysis such as document summarization, scientific trend discovery, document element extractions, etc.

In this paper, we describe a strategy to automatically build a semantic hierarchical structure of sections of a scholarly paper. A section is defined as a pair of section header and its textual content. Oftentimes, a section in a scientific paper is a *standard section* (i.e. *Abstract*, *Introduction*, *Background and Related Work*, etc.). Hence, we also propose a rule-based approach to recognize these standard sections if they exist.

Specifically, this paper has 5 key contributions: 1.) We propose a machine learning based strategy to identify boundaries in a scholarly document by detecting section headers and use them as section boundaries. 2.) We construct a set of regular expressions to recognize standard sections. 3.) We propose a rule-based strategy to build a hierarchical structure of sections. 4.) We validate our methods using empirical evaluation, with standard precision, recall, F-measure, and accuracy metrics. 5.) We show an application of our section recognition algorithms in a pseudo-code detection task.

II. Background and Related Works

Document segmentation has long been an area of active research. Multiple document segmentation techniques have been proposed [6], [10], [12], [21]; however, they focus on document zoning analysis where content in a document page is segmented into visually similar regions (i.e. images, text chunks, background images, tables, etc.). Such methods would not directly apply to our problem since 1) a section in a scholarly document may contain heterogeneous document elements such as tables, algorithms, text of different types and styles, etc. These heterogeneous pieces of information would be disintegrated if the document is zoned. 2) A section in a scholarly document may spread across multiple pages. Zoning algorithms would not be able to stitch up textual information of the same section that spread across pages.

Some works on identifying sections in scholarly documents have been investigated. We discussed them and their limitations here. Treeratpituk et al. proposed a section detection algorithm as part of their keyphrase extraction work [15]. They used a set of regular expressions to detect 6 common sections including *Title*, *Abstract*, *Introduction*, *Related Work*, *Methodology + Experiments*, and *Conclusion + Future Work*. However, their algorithm only identifies common sections, while our algorithm aims to identify all the sections. Nguyen et al. have successfully used a Maximum Entropy (ME) classifier to classify a section header into one of 14 common section header types [13]. Their method, however, requires section headers to be pre-identified using a preprocessing technique, which they did not specify. Denny et al. proposed a method to automatically identify section headers in *history and physical examination* documents [7]. However, their method is designed specifically for medical notes and has a high reliance on medical keywords, and hence would only detect sections whose headers contain medical terms.

III. Datasets

Two datasets are used in this paper. The first dataset includes 100 scholarly documents manually selected from the Citeseer^X repository to cover diverse types of publications. We designed feature sets for our machine-learning based methods by examining this dataset, and construct rules and regular expressions for our rule-based methods.

Another dataset comprises 117 PDF scholarly documents randomly selected from Citeseer^X[1] repository, containing various types of scholarly documents namely conference papers, journals, theses, and academic articles. This dataset is used for evaluation. Inspired by Hassan [8], we use PDFBox[2] to extract text and modify the package to extract object information such as font information and co-ordinates of characters from PDF documents. On average, a document has 993 lines, 20 pages, and 17 sections. Note that images are disregarded as we are only interested in textual data.

[1] http://citeseer.ist.psu.edu
[2] http://pdfbox.apache.org

978-1-4799-1805-8/15/$31.00 ©2015 IEEE

We tag each line in the document with the following labels: **0** (not a section header), **ABS** (Abstract), **INT** (Introduction), **REL** (Background and Related Work), **RAD** (Experiment, Results, and Discussion), **CON** (Conclusions), **ACK** (Acknowledgment), **REF** (References), **OTHER** (Other Section).

Note that though published document segmentation datasets exist, such as GROTOAP [14] and PSET [11], these datasets were developed for document zoning analysis problems, and hence would not directly accommodate our experiment.

IV. Identifying Section Boundaries

We observe that scholarly documents have the following properties: 1) Each section has a section header, usually with a section number. 2) Section headers usually have different font styles from the surrounding content. 3) The majority of the sections are common sections such as *Abstract*, *Introduction*, *Background*, *Conclusions*, and *References*.

TABLE I. Features used to characterize section headers.

Grp	Feature	Description
PAT	IS SEC HEADER W/ NUM	Whether the line matches the number-leading section header pattern.
	IS UPPER SEC HEADER W/ NUM	Whether the above line matches the #-leading section header pattern.
	IS LOWER SEC HEADER W/ NUM	Whether the lower line matches the #-leading section header pattern.
	IS SEC HEADER W/O NUM	Whether the line matches the section header without number pattern.
	IS UPPER SEC HEADER W/O NUM	Whether the above line matches the section header without pattern.
	IS LOWER SEC HEADR W/O NUM	Whether the lower line matches the sec. header without number.
	IS STANDARD SEC	Whether the line is one of the standard section headers.
	IS CAPTION	Whether the line is a caption or a figure, table, or algorithm.
STY	MODE FONTSIZE	Mode fontsize (in Pt.) of all the characters in the line
	FRAC FONT MODE TO DOC AVG	Fraction of the mode fontsize in the line to the avg. fontsize in the doc.
	FRAC FONT MODE TO DOC MODE	Fraction of mode fontsize in the line to the mode fontsize in the doc
	FRAC UPPER GAP TO MODE GAP	Fraction of the gap space between the line and the upper line (in cm) to the mode gap space between two lines in the document.
	FRAC LOWER GAP TO MODE GAP	Fraction of the gap space between the line and the lower line (in cm) to the mode gap space between two lines in the document.
	FRAC UPPER GAP TO AVG GAP	Fraction of the gap space between the line and the upper line (in cm) to the average gap space between two lines in the document.
	FRAC LOWER GAP TO AVG GAP	Fraction of the gap space between the line and the lower line (in cm) to the average gap space between two lines in the document.
	ARE ALL CHARS BOLD	Whether all the characters in the line have boldface font style.
	ARE ALL WORDS CAPITALIZED	Whether all the words in the line are capitalized.
	FRACTION NUMWORDS TO AVG NUMWORDS	Fraction of the number of words in the line to the average number of words per line of the document.
STR	IS AFTER ABS INT	Whether the abs or int sections has already been detected.
	IS BEFORE REF	Whether the reference section header has NOT yet been detected.
	IS FIRST LINE OF PAGE	Whether the line is the first line of page.
	IS LAST LINE OF PAGE	Whether the line is the last line of page.

Such observations lead us to identify 22 features that characterize section headers (listed in Table I). The features can be divided into 3 groups: *pattern based* (PAT), *style based* (STY), and *structure based* (STR). The *PAT* features are used for capturing standard section headers or section headers with section numbers. The *STY* features filter out lines that look like section headers but are actually fragments of sentences, lines in tables of content, or textual fragments from tables/diagrams. The *STR* features mostly concern the locations of the section headers. For example, lines that occur between the *Abstract/Introduction* section and the *References* section are better candidates than those outside this region. Moreover, the `IS FIRST LINE OF PAGE` and `IS LAST LINE OF PAGE` features are used to filter out page headers and footers.

A. Balancing Training Data

Our dataset is highly skewed with only 1.75% being positive samples; hence multiple data balancing techniques are explored to alleviate the *class imbalance problem* [1]. These techniques include:

None. No data balancing is applied.

Weighting (WEIGHT). Let $npRatio = \frac{\#\_-ve\ instances}{\#\_+ve\ instances}$. A positive instance is given a weight of $npRatio$, while a negative instance is given a weight of 1.

Random Over-sampling (ROver). Minority class instances are randomly selected and duplicated until the populations of the two classes are equal.

Random Under-sampling (RUnder). Majority class instances are randomly selected and removed until the populations of the two classes are equal.

Resampling (ReS). A random subsample of the training data is reproduced using sampling with replacement. The new training data has the same number of total samples as the old one, but equal populations of both classes.

SMOTE. The minority class samples are over-sampled using the Synthetic Minority Oversampling TEchnique (SMOTE) [4]. The SMOTE algorithm avoids the overfitting problem by forming new minority class examples by interpolating between several minority class examples that lie together.

B. Classification Algorithms

In our experiment, we employ four classification algorithms: **Random Forest (RF)** [3], **Support Vector Machine (SVM)** [2], **Repeated Incremental Pruning to Produce Error Reduction (RIPPER)** [5], and **NaiveBayes (NB)** [9].

We use the LibSVM[3] implementation for SVM, and the Weka[4] implementation for the other classifiers.

C. Experiment, Results and Discussions

For each text line in a test document, the classifier determines whether it is a section header or not. Ten fold document-wise cross validation is used to validate our methods.

1) Results and Discussion: Standard *precision*, *recall*, and *F-measure* are used as our evaluation metrics. For a test document, a text line is said to be *correctly classified as a section header* if the classifier identifies it as a section header and it is within K (flexible window size) lines above or below the actual section header line. Note that $K = 0$ allows strict evaluation, while $K > 0$ allows the classifier to predict fuzzy boundaries. In our experiment, we set K = 2. Let T_p be the set of all the lines *correctly* classified as section headers, T_r the set of all lines classified as section headers, and T_g the set of all actual section header lines. We define precision, recall, and F-measure as:

$$precision = \frac{|T_p|}{|T_r|}, recall = \frac{|T_p|}{|T_g|}, F = \frac{2 \cdot precision \cdot recall}{precision + recall}$$

TABLE II. Performance of each classifier with best data balancing techniques (in terms of F1) *ATT* denotes Average Training Time.

Classifier	Balancing	Pre	Rec	F1	ATT (s)
RF	WEIGHT	0.9374	0.9106	**0.9238**	52.96
RIPPER	none	**0.9118**	0.8623	0.8863	62.39
SVM	WEIGHT	**0.8570**	0.8828	0.8697	6.74
NB	none	0.4420	0.8744	0.5872	1.19

[3]http://www.csie.ntu.edu.tw/ cjlin/libsvm/

[4]http://www.cs.waikato.ac.nz/ml/weka/

Table II lists the best result of each classification algorithm (in terms of F1), when combined with the best data balancing method for that algorithm. Tree-based classifiers such as Random Forest perform relatively better than other classifiers. This improvement is because most of the features are binary features. In fact, the best performance in terms of F-measure is yielded by the Random Forest classifier trained with weighted balancing data.

Furthermore, the best classification model is tested on 25 documents randomly selected from the GROTOAP dataset [14], yielding precision of 0.71, recall of 0.50, and F1 of 0.59. Note that the performance on this dataset is worse than those in Table II because the GROTOAP dataset comprises papers of much various fields, including Statistics, Biology, and Medicine, while the classifier is trained with a dataset comprising only computer science papers.

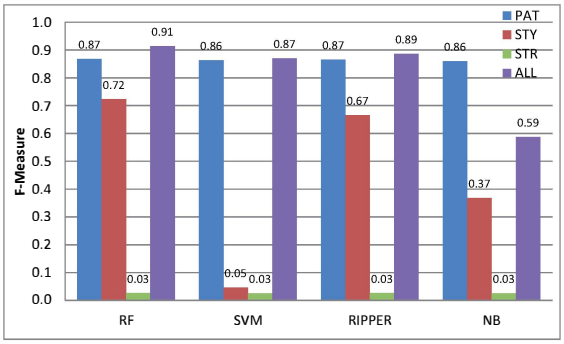

Fig. 1. Comparison of different classifiers trained with different feature types.

2) Impact of Different Types of Features: Figure 1 compares the performance in terms of the F-measure of each classifier trained with different types of features. We find that the pattern-based features are most useful. The structure-based features alone do not provide much information about the data; however, they can be additionally useful when combined with other types of features. The SVM classifiers turn out to benefit only little from the style-based features compared to other classifiers, possibly due to the data normalization problem since most of such features are numeric values. Hence combining the style-based features with other features impedes the overall performance of SVM classifiers. The same analysis applies to the NaiveBayes classifiers–the overall performance of the NaiveBayes classifiers drops significantly when trained with the style-based features. We could explore topical based techniques [17], [18] to extract features for future work.

3) Impact of Data Balancing Methods: Figure 2 displays the performance (in terms of F-measure) of each classifier trained with data balanced by different balancing strategies, with fixed $K = 2$. We make the following observations:
1.) All classifiers perform poorly when trained with RUnder'ed data, compared to other balancing techniques. This is because, RUnder method removes roughly 96.50% of negative samples, causing the classifier to obtain insufficient knowledge about the negative class samples which naturally have a much wider variety in style than that of the positive samples.
2.) The performance of Random Forest does not change much when trained with balanced data (except with RUnder'ed data). This robustness is because the algorithm already incorporates

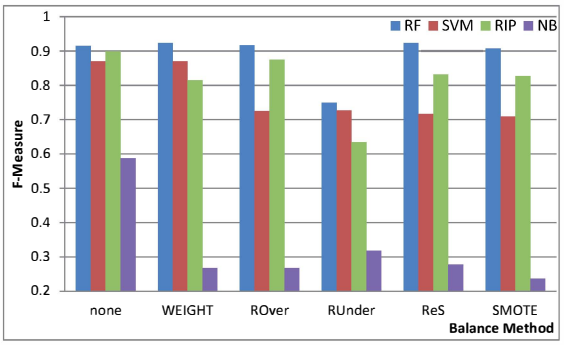

Fig. 2. Comparison of different data balancing methods on each classifier.

sampling techniques and cost matrices to handle class imbalance [19].
3.) All balancing techniques that involve duplicating/removing instances such as ROver, RUnder, ReS, and SMOTE seem to reduce the learning ability of SVM and NaiveBayes. These techniques distort the distribution of the population of the data in the training set. Consequently, the learned model loses accuracy.
4.) The performance of NaiveBayes classifiers decreases with all balancing techniques.

D. Impact of Flexible Window Sizes

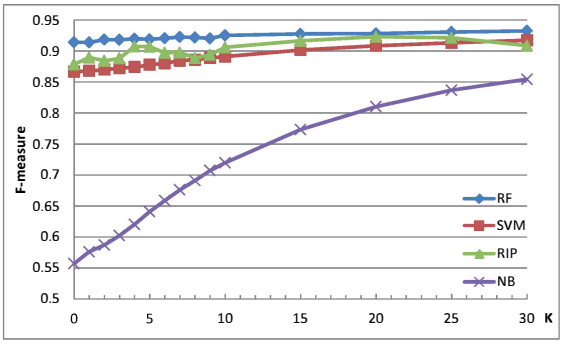

Fig. 3. Comparison of flexible window sizes (K) on each classifier.

We investigate how varying the values of K impacts the classification performance. $K = 0$ means the detected section headers are correctly classified if they exactly match the line numbers of the actual section headers. In general, the classification performance should increase as K increases, as it allows higher matching flexibility. Figure 3 compares the performance (in terms of F-measure) of each classifier when evaluated with different values of K.

The performance of RF, SVM, and RIPPER classifiers do not change much as the window size increases. These results imply that our proposed features are effective in discriminating section-header lines from their surrounding context. Note that the NaiveBayes performance increases significantly as the window grows large. This observation is not surprising since NaiveBayes performs very poorly at lower K; hence the effect of window size increase is prominent in NaiveBayes' performance. We also try other Bayes-based classifiers such as NaiveBayes Updatable, NaiveBayesMultinomial, ComplementNaiveBayes and BayesNet, all of which we find to per-

form similar to NaiveBayes. As a result, we conclude that NaiveBayes and other Bayes based classification algorithms are not suitable for our task.

V. Identifying Standard Sections

In this section, we propose to recognize these standard section using a set of regular expressions displayed in Figure 4. These regular expressions are derived by observing on the writing variation of section headers.

A. Experiments and Results

We frame the standard section recognition as a binary classification task. For example, in order to see if a section header is an *Abstract* section or not, we check if it matches the *Abstract* regular expression rule in Figure 4 or not. We use standard *accuracy* as the evaluation metric for this task.

$$Accuracy = \frac{|Correctly\ Classified\ Section\ Headers|}{|All\ Section\ Headers|} \quad (1)$$

TABLE III. Accuracy on each standard section classification.

Section	Acc.%
ABS	100
INT	100
REL	90.48
RAD	88.46
CON	94.44
ACK	100
REF	100

We report the accuracy of the recognition task in Table III. Our method works well on capturing almost all standard sections, except for the *RAD* section whose accuracy is the lowest. This discrepancy is because authors use a much wider variety of words such as *Experiment, Results and Discussion* to name such sections.

VI. Building a Hierarchy of Sections

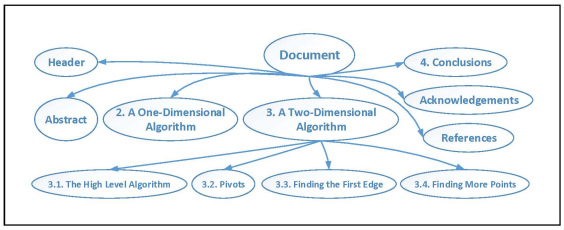

Fig. 5. A scholarly document is organized into a hierarchy of sections.

The section layouts in scholarly articles are usually composed of sections and sub-sections (Example, Figure 5). We propose a simple set of heuristics that build a hierarchy of sections from the extracted section headers.

We notice that there are two types of section headers: *numbering* (e.g., `3.1 The High Level Algorithm`) and non-numbering (e.g., `Spammer Detection Model`). For the former, the numberings are directly used to identify sub-section relationship. For the latter, we use the indentation spacing from the column margin to infer sub-subsection relationship.

We evaluate the section-hierarchy-building task by framing the problem into the section-positioning task. Specifically, a section is *correctly positioned* if it is placed under the right super-section in the hierarchy. Standard accuracy is used as the evaluation metric:

$$Accuracy = \frac{|Correctly\ Positioned\ Sections|}{|All\ Sections|} \quad (2)$$

Our section positioning algorithm is 95.51% accurate on the evaluation dataset. An error analysis reveals that most of the incorrectly positioned sections are deep-level subsections that are not numbered or whose numberings change to alphabets. We suggest that using fontsizes as a feature could remedy this issue, and would explore this option as future work.

VII. Application in Pseudo-code Detection

We recently presented a machine-learning based approach for detecting pseudo-codes in scientific documents [16], [20]. Two major variants of pseudo-code detection approaches were proposed: machine-learning based (PC-ML) and combined method (PC-CB). The PC-ML method consists of two main steps. First, it detects the presence of sparse regions in a document page. A sparse region is a set of consecutive lines with noticeably fewer textual content than other lines in the documents. In the second step, a machine learning classification technique is employed to classify each sparse region whether it is a pseudo-code or not. 47 features are extracted from each sparse box including font-style based (FS), context based (CX), content based (CN), and structure based (ST) features. The *FS* features capture the various font styles used in pseudocodes. The *CX* features detect the presence of pseudocode captions. The *CN* features capture the pseudocode specific keywords and coding styles. The *ST* features characterize the sparsity of pseudocodes and the symbols used.

The PC-CB method merges the results from the PC-ML method with a set of results using regular expressions to detect the presence of pseudo-code captions. Details of the PC-ML and PC-CB methods are in our previous paper [16].

Pseudo-codes are typically in the methodology sections of the papers and not in the Abstract, Introduction, Related Work, and Conclusion sections. Sparse boxes found in these sections are most likely mathematical expressions or textual remnant from tables and figures. With this intuition, a modification is made to the previously proposed pseudo-code detection techniques by adding 12 *section-based* features into the feature set. Such features characterize the semantic location of the sparse region in the scholarly paper.

In order to see if these additional section-based features would improve the classification performance, we perform the experiments on the same dataset and experiment settings as in our previous work [16], using 10-fold document-wise cross validation.

Table IV shows the performance comparison between two pseudo-code detection techniques (i.e. PC-ML and PC-CB) before and after (denoted by the '+' extension) adding the section-based features into the feature space. The classification performance is improved by 7.17% in PC-ML approach, and 9.63% in the PC-CB approach, bringing the overall pseudo-code detection performance to 85.58% (F1). This improvement does not only infer that section-based features could help

Abstract	Abstract—ABSTRACT—(Abstract—ABSTRACT)\.\s+.*—(Abstract—ABSTRACT)(\-)[A-Z].*—(Abstract—ABSTRACT)[\s]?[\\-]?[\s]?[A-Z].*—(Abstract—ABSTRACT)\-—ABSTRACT\s+[A-Z].*—(Abstract—ABSTRACT):\s+.*
Introduction	(([1iI](\.)?)(\s+))?(Introduction—INTRODUCTION)([:\.]?(\s+)(and—[A-Z]).*)?
Background	(([1-9][0-9]*(\.[1-9][0-9]*)*)—[iIvVxX]+)(\.?)(\s+).*(Background—BACKGROUND—Previous Work—Previous work —PREVIOUS WORK—Similar work—Similar Work—SIMILAR WORK—Related Research—Related research —RELATED RESEARCH—PRELIMINARIES—Preliminaries—Preview—PREVIEW—Related Work—Related work —RELATED WORK—Motivation—MOTIVATION—motivation—Overview—overview—OVERVIEW—Review—review —REVIEW—PREAMBLE—Preamble—STATE OF THE ART—State of the Art—State of the art)[^\.:]*
Conclusion	((([1-9][0-9]*(\.[1-9][0-9]*)*)—[iIvVxX]+)(\.?)(\s+))?.*(Future Work—Future work—FUTURE WORK—Further Work—Further work —FURTHER WORK—CONCLUSION—Conclusion—CONCLUSIONS—Conclusions—CONCLUDING REMARKS—Concluding remarks —Concluding Remarks—SUMMARY—Summary—OPEN QUESTIONS—Open Questions—Open questions)[^\.:]*
Acknowledgment	((([1-9][0-9]*(\.[1-9][0-9]*)*)—[iIvVxX]+)(\.?)(\s+))?(Acknowledgments—ACKNOWLEDGMENTS—Acknowledgements —ACKNOWLEDGEMENTS—Acknowledgment—ACKNOWLEDGMENT—Acknowledgement—ACKNOWLEDGEMENT)
Reference	((([1-9][0-9]*(\.[1-9][0-9]*)*)—[iIvVxX]+)(\.?)(\s+))?(Reference—REFERENCE—References—REFERENCES—Bibliography —BIBLIOGRAPHY)
GEN-WN	(([1-9][0-9]*)|([1-9][0-9]*\.)|([1-9][0-9]*(\.[0-9]+)+(\.)?)|([IVX]+(\.)*))
GEN-WON	([A-Z][a-zA-Z]*)(\s+[A-Z][a-zA-Z]*)+

Fig. 4. Regular expressions used to capture standard and general section headers. GEN-WN: General Section with numbers. GEN-WON: General section without numbers.

TABLE IV. THE CLASSIFICATION PERFORMANCE (PRECISION, RECALL, F1, AND IMPROVEMENT IN F1) OF PSEUDO-CODE DETECTION BEFORE AND AFTER ADDING SECTION BASED FEATURES INTO THE FEATURE SPACE (DENOTED BY THE '+' EXTENSION).

Method	Pr (%)	Re (%)	F1 (%)	Δ F1 (%)
PC-ML	**88.84**	53.74	66.97	-
PC-ML+	79.03	**71.33**	**74.14**	7.17
PC-CB	87.37	67.17	75.95	-
PC-CB+	**94.6**	**79.5**	**85.58**	9.63

discovering pseudo-codes in scholarly documents, but also illustrates that our document segmentation algorithm could potentially be applied to a wide range of applications in scholarly digital libraries where section information is useful.

VIII. CONCLUSIONS

Most scholarly documents in modern digital libraries are in PDF format which does not maintain section structures. We propose a set of methods to build a semantic hierarchy of sections in a scholarly document. The problem is divided into three subtasks: 1) identifying section headers, 2) recognizing standard sections (i.e., *Abstract*, *Introduction*, etc), and 3) building a hierarchy of sections. In the first task, we employ machine-learning based classification techniques to detect section headers and use them as section boundaries. In the second task, we use a set of regular expressions that capture lexical patterns in standard sections. For the final task, we show that a set of simple heuristics can be effectively used to capture sub-section relationship. Future work could explore ensemble strategies and strengthen the evaluation by performing large scale evaluation.

IX. ACKNOWLEDGMENTS

We gratefully acknowledge partial support from the National Science Foundation.

REFERENCES

[1] G. E. A. P. A. Batista, R. C. Prati, and M. C. Monard. A study of the behavior of several methods for balancing machine learning training data. *SIGKDD Explor. Newsl.*, 6(1):20–29, 2004.

[2] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[3] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[4] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. Smote: synthetic minority over-sampling technique. *J. Artif. Int. Res.*, 16(1):321–357, 2002.

[5] W. W. Cohen. Fast effective rule induction. In *ICML '95*, pages 115–123. Morgan Kaufmann, 1995.

[6] A. Dengel and F. Shafait. Analysis of the logical layout of documents. *Handbook of Document Image Processing and Recognition*, pages 177–222, 2014.

[7] J. C. Denny, A. S. III, K. B. Johnson, N. B. Peterson, J. F. Peterson, and R. A. Miller. Evaluation of a method to identify and categorize section headers in clinical documents. *JAMIA*, 16(6):806 – 815, 2009.

[8] T. Hassan. Object-level document analysis of pdf files. In *DocEng '09*, pages 47–55, New York, NY, USA, 2009. ACM.

[9] G. H. John and P. Langley. Estimating continuous distributions in bayesian classifiers. UAI '95, pages 338–345, 1995.

[10] D. Malerba, F. Esposito, O. Altamura, M. Ceci, and M. Berardi. Correcting the document layout: a machine learning approach. ICDAR '03, pages 97 – 102, 2003.

[11] S. Mao and T. Kanungo. Software architecture of pset: A page segmentation evaluation toolkit. *International Journal on Document Analysis and Recognition*, 4(3):205–217, 2002.

[12] S. Marinai, E. Marino, and G. Soda. Conversion of pdf books in epub format. ICDAR '11, pages 478–482, 2011.

[13] T. D. Nguyen and M.-Y. Kan. Keyphrase extraction in scientific publications. In *ICADL'07*, pages 317–326, 2007.

[14] D. Tkaczyk, A. Czeczko, K. Rusek, L. Bolikowski, and R. Bogacewicz. Grotoap: ground truth for open access publications. In *JCDL '12*, 2012.

[15] P. Treeratpituk, P. Teregowda, J. Huang, and C. L. Giles. Seerlab: A system for extracting key phrases from scholarly documents. In *SemEval '10*, pages 182–185, 2010.

[16] S. Tuarob, S. Bhatia, P. Mitra, and C. Giles. Automatic detection of pseudocodes in scholarly documents using machine learning. In *ICDAR '13*, pages 738–742, Aug 2013.

[17] S. Tuarob, L. C. Pouchard, and C. L. Giles. Automatic tag recommendation for metadata annotation using probabilistic topic modeling. In *JCDL '13*, 2013.

[18] S. Tuarob, L. C. Pouchard, P. Mitra, and C. L. Giles. A generalized topic modeling approach for automatic document annotation. *International Journal on Digital Libraries*, pages 1–18, 2015.

[19] S. Tuarob, C. S. Tucker, M. Salathe, and N. Ram. An ensemble heterogeneous classification methodology for discovering health-related knowledge in social media messages. *Journal of biomedical informatics*, 49:255–268, 2014.

[20] Z. Wu, J. Wu, M. Khabsa, K. Williams, H.-H. Chen, W. Huang, S. Tuarob, S. R. Choudhury, A. Ororbia, P. Mitra, et al. Towards building a scholarly big data platform: Challenges, lessons and opportunities. In *JCDL '14*, pages 117–126, 2014.

[21] S. Yacoub and J. A. Peiro. Identification of document structure and table of content in magazine archives. ICDAR '05, pages 1253–1259, 2005.