# Identification of Non-textual Components in Technical Documents

**Myungha Jang**
Center for Intelligent Information Retrieval
School of Computer Science
UMass Amherst
mhjang@cs.umass.edu

**Jinho D. Choi**
Natural Language Processing Research Group
Emory University
jinho.choi@emory.edu

**James Allan**
Center for Intelligent Information Retrieval
School of Computer Science
UMass Amherst
allan@cs.umass.edu

## Abstract

We tackle the automatic identification of non-textual components such as tables, formulas, pseudo-code, and miscellaneous text in technical documents. We focus on documents in slide formats, which have been relatively under-explored in previous studies. Identification of non-textual components can provide better document representation for tasks such as clustering or information retrieval. We view this problem as an information extraction task and build a multi-classification model trained on texts extracted from PDF, PPT, and HTML. Our model can handle any format as long as it can be converted into line-preserving plain text. Our approach is evaluated intrinsically on the dataset we annotated and also extrinsically through clustering and document retrieval tasks, showing noticeable improvement on these tasks.

## 1 Introduction

Non-textual document components such as tables, formulas, or pseudo-code are useful elements in technical documents in that they deliver complicated concepts in a visual, organized, and logical manner. Identification of such components is an important task for document understanding because their semantics often cannot be interpreted by analyzing the individual elements themselves. For example, each cell in a table can be understood only if its association with the row and the column is known. Formulas and code convey semantics through the logic represented by certain conventions; without understanding the logic, they are simply sets of symbols, operators, reserved words, or variable names.

The motivation of this work arose from topic clustering on technical documents. During our experiments, we discovered that non-textual components introduced too much noise for clustering, causing spurious matches between documents containing many of these components. Therefore, we hypothesized that the removal of the non-textual components would lead to improvement in clustering.

Many previous studies have suggested automatic ways of extracting tables and formulas from documents in the PDF and OCR formats. However, most of these approaches are limited to specific file formats, which becomes an issue for systems processing documents in heterogeneous formats. In this paper, we propose a new approach for the identification of non-textual components using plain text extracted from files in the PDF, PPT, and HTML formats with little to no explicit visual layout information preserved. Our approach is less format-dependent as it only requires lines to be preserved, and does not incur any unnecessary format conversion except for the initial text extraction from the original file. The effectiveness of our approach is evaluated extrinsically through clustering and document retrieval tasks, showing noticeable improvement on these tasks.

## 2 Related Work

Douglas and Hurst (1996) proposed a deterministic algorithm using white spaces and punctuation for detecting table layout and lead-in-text as table description candidates. Various efforts have been made for table extraction using semi-supervised learning on the patterns of table layouts within ASCII text documents (Ng et al., 1999), web documents (Pinto et al., 2003; Lerman et al., 2001; Zanibbi et al., 2004), PDF and OCR image documents (Liu et al., 2007). Existing techniques exploit the graphical features such as primitive geometry shapes, symbols, and lines to detect table borders; however, no work has attempted to process plain text extracted from richer formats, where table layouts are unpreserved.

Lin et al. (2011) categorized existing approaches for mathematical formulas detection by 'character-based' and 'layout-based' with respect to key features. Character-based approaches use features of mathematical symbols, operators, and positions and their character sizes (Kacem et al., 2001; Suzuki et al., 2003). Chan and Yeung (2000) provide a comprehensive survey of mathematical formula extraction using various layout features available from image-based documents.

Tuarob et al. (2013) proposed 3 pseudo-code extraction methods: a rule based, a machine learning, and a combined method. Their rule based approach finds the presence of pseudo-code captions using keyword matching. The machine learning approach detects a box surrounding a sparse region and classifies whether the box is pseudo-code or not. They extracted four groups of features: font-style based, context based, content based, and structure based.

## 3 Corpus

### 3.1 Data collection

We collected two types of document sets: lecture slides used in Data Structure and Algorithms courses ($D_{dsa}$), and ACL'12-13 proceeding papers ($D_{acl}$). We chose these two sets becasuse they consisted of different ratios of non-textual components (Table 1) such that they were complementary to each other for better coverage of these non-textual components.

### 3.2 Text extraction

We extracted plain text from our datasets using several open-source software packages such as Apache Tika and Apache PdfBox[1]. These packages are available for text extraction from various formats including PDF, PowerPoints, and HTML. Figure 1 shows the snapshot of a table and text extracted from it using Apache Tika. Note that the visual layout that used to identify the non-textual components is lost.

[1] tika.apache.org, pdfbox.apache.org

| Chinese-to-English | | | |
|---|---|---|---|
| | NIST05 | NIST06 | NIST08 |
| L-Hiero | $25.27^+$ | $25.27^+$ | $18.33^+$ |
| AdNN-Hiero-E | 26.37 | 25.93 | 19.42 |
| AdNN-Hiero-D | 26.21 | 26.07 | 19.54 |

```
Chinese-to-English
NIST05 NIST06 NIST08
L-Hiero 25.27+ 25.27+ 18.33+
AdNN-Hiero-E 26.37 25.93 19.42
AdNN-Hiero-D 26.21 26.07 19.54
```

Figure 1: The table (top) and its extracted text (bottom)

### 3.3 Annotation

In this study, we tackle 4 types of non-textual components: table, code, math formula, and misc. text. Misc. text refers to a chunk of garbled text mostly caused by processing figures or diagrams. Any line that is neither prose nor the other type of non-textual components is considered miscellaneous. Furthermore, we assume that there is no overlap between these components. We carefully created annotation guidelines for the 4 types of non-textual components and annotated 35 lectures slides (7,943 lines) and 35 proceeding papers (25,686 lines).

| Dataset | Ratio of components (%) | | | |
|---|---|---|---|---|
| | Table | Code | Formula | Misc |
| $D_{dsa}$ | 1.4 | 14.8 | 0.5 | 9.8 |
| $D_{acl}$ | 4.0 | 0.6 | 5.0 | 6.4 |

Table 1: The ratio of non-textual components in each set

## 4 Features

This section introduces types of features used for our experiments. We find line-based prediction has more advantage over token-based prediction because it allows us to observe the syntactic structure of the line, how statistically common the grammar structure is, and how layout patterns compare to neighboring lines. The sequential nature of the lines is also an important feature because the components usually occur over a block of contiguous lines. The limitation of line-based prediction is that components that are embedded in the midst of the line either cannot be extracted or are extracted with false positives within the line. We leave this part as future work.

**Syntactic features** Lines containing non-textual components are likely to form unusual syntactic structures. We parsed each line using the dependency parser in ClearNLP (Choi and McCallum, 2013) and extracted features such as unigrams and bigrams, the set of dependency labels, the ratio of each POS tag, and POS tags of each dependent-head pair from each parse tree.

**Implicit table layout** Text extracted from tables still preserves implicit layout through its string patterns. Tables tend to convey the same string pattern along the same column or row, which is frequently parallel across multiple columns or rows. To capture the layout represented by these string patterns, we encode each line; if the token, potentially representing one cell, is a string, we replace the token with `S`, and `N` if it is numeric.

| | |
|---|---|
| SSS | NIST05 NIST06 NIST08 |
| SNNN | L-Hiero 25.27+ 25.27+ 18.33+ |
| SNNN | AdNN-Hiero-E 26.37 25.93 19.42 |
| SNNN | AdNN-Hiero-D 26.21 26.07 19.54 |

Table 2: The encoded line of the table in Figure 1

We then compute the edit distance between the encoded line and its neighboring lines. The idea is that the lines in a table are likely to have the same type of patterns nearby. However, edit distance alone often leads to false positives because it is easy to have the same simple string patterns between two lines that are not from tables. Having a smaller edit distance between two lines whose patterns are statistically more table-like should be a stronger evidence. Hence we compute Pattern Language Bigram Probability (PLB), the probability of seeing an S/N sequence, with bigrams learned from training data.

Additionally, we use features such as the edit distance and length difference of the current and the previous two lines, the edit distance multiply by the line's PLB, and the ratio of the consecutive exact match between the current and previous encoded lines leftwards and rightwards.

**Code features** We use six code patterns for the detection of pseudo-code. Features include the number of tokens that use brackets (e.g., `sum[i]`), that follows variable naming convention such as `camelCase` or having underscore in the middle, that are operators, and that are the number of reserved programming keywords. Line-level patterns inspect whether the line has comment symbols (`//`, `/*`, `*/`) or whether the line ends with semi-colon.

**Sequential features** The sequential nature of the lines is also an important feature because the component most likely occurs over a block of contiguous lines. We train two models. The first model uses the annotation for the previous line's class. We then train another model using the previous line's predicted label, which is the output of the first model.

## 5 Evaluation

### 5.1 Intrinsic evaluation

This section reports classification results of the non-textual components in multi-domains where two datasets are combined as one for better generalization. We used the Liblinear SVM classifier for training (Fan et al., 2008) and ran 5-fold cross-validation for evaluation. As shown in Table 3, our classifier shows above 80% F1-scores for all components.

| | Precision | Recall | F1 score |
|---|---|---|---|
| Table | 92.76 | 75.42 | 83.20 |
| Code | 91.05 | 85.20 | 88.03 |
| Formula | 84.90 | 78.24 | 81.43 |
| Miscellaneous | 85.69 | 90.50 | 88.03 |

Table 3: Multi-domain classification accuracy trained and tested on $D_{dsa}$ and $D_{acl}$ combined

### 5.2 Extrinsic evaluation: clustering

We generated the gold-standard clusters based on the topics provided in the course syllabus. We used the topic names as the initial centroids and set the similarity distance threshold to 0.1 because not all documents could fit in the given cluster category. We constructed TF-IDF vectors from each document using the top 30 terms among unigrams, bigrams, and trigrams. Since the collection is small, we used the Google *N*-gram data as more reliable collection statistics for computing IDF.

We hypothesized that these non-textual components hampered clustering quality so that removal of them would improve clustering results. To verify this hypothesis, we manually removed the components from a subset of $D_{dsa}$ (147 documents). As

|  | Before noise-removal |  |  | After noise-removal |  |  |  |
|---|---|---|---|---|---|---|---|
| $D_{dsa}$ | P | R | F1 | P | R | F1 | Gain |
| (1): Unigrams | 60.06 | 57.24 | 58.62 | 73.80 | 66.61 | 70.02 | +16.3 |
| (2): (1) + Bigrams | 76.75 | 68.88 | 72.60 | 77.48 | 69.94 | 73.52 | +1.2 |
| (3): (2) + Trigrams | 76.83 | **70.83** | **73.71** | **78.63** | 68.60 | 73.28 | -0.6 |
| $D_{os}$ | P | R | F1 | P | R | F1 | Gain |
| (1): Unigrams | 60.07 | 55.15 | 57.50 | 62.99 | 58.26 | 60.54 | +5.0 |
| (2): (1) + Bigrams | 67.47 | 56.17 | 61.30 | 70.16 | 59.79 | 64.56 | +5.1 |
| (3): (2) + Trigrams | 69.89 | 60.03 | 64.59 | **70.69** | **61.98** | **66.05** | +2.2 |

Table 4: Clustering accuracy on the two datasets before and after noise-removal

shown in Table 5, with manual removal, the clustering accuracy improves over 7% compared to no removal. Even with automatic removal, the accuracy improves over 2%.

|  | Prec. | Rec. | F1 |
|---|---|---|---|
| Baseline | 59.71 | 55.40 | 57.47 |
| Automatic Removal | **63.61** | **56.01** | **59.57** |
| Manual Removal | 67.23 | 63.28 | 65.20 |

Table 5: Clustering accuracy using manual and automatic noise removal on a subset of $D_{dsa}$

In addition to $D_{dsa}$ of 289 documents, we collected another set of 326 lecture slides on the topic of Operating Systems ($D_{os}$). We compared clustering accuracy of before and after automatic removal in $D_{dsa}$ and $D_{os}$ (Table 4). Removal of non-textual components generally improved clustering results; surprisingly, it made greater impact on $D_{os}$, which was not included in our training data.

### 5.3 Extrinsic evaluation: document retrieval

Identifying non-textual components can improve a document retrieval task for queries searching for ones consisting of these components. We focused on pseudo-code and conducted retrieval experiments to observe how the identification of this component would improve document retrieval.

We replaced all identified code lines in $D_{dsa}$ with `<CODE>` tags and created a new dataset, $D^c_{dsa}$. We generated 4 queries for pseudo-code: sorting algorithm, shortest path, priority queue, graph traversal. We collected the top 20 retrieved documents for the four queries over the two dataset, and annotated the relevance using 3 graded scale: bad, fair, and excellent, based on whether documents are topically relevant and they contain the pseudo-code of interest. Using the relevance dataset, we evaluated the accuracy of the top 10-ranked list. Query liklihood model was used for retrieval in Galago.[2]

As shown in Table 6, four evaluation metrics, MAP, NDCG, and Precision at K = 5, 10, $D^c_{dsa}$, returned better ranked lists for the queries looking for containment of a pseudo-code component. Although this experiment is carried on a small scale and the way of using component information may be naive, this result suggests that understanding the anatomy of documents information can be crucial for improving information retrieval.

|  | $D_{dsa}$ | $D^c_{dsa}$ |
|---|---|---|
| MAP | 0.3113 | 0.3741 |
| NDCG | 0.4317 | 0.5443 |
| Precision@5 | 0.2500 | 0.3500 |
| Precision@10 | 0.3000 | 0.3500 |

Table 6: Accuracy of the top 10 ranked list on pseudo-code queries in $D_{dsa}$ and $D^c_{dsa}$

## 6 Conclusion

In this paper, we presented a less format-dependent approach to the identification of non-textual components in technical documents. Our approach uses line-based feature extraction to exploit grammar soundness and implicit textual layout of a line, and layout pattern matching with its neighboring lines. We evaluated our approach on multi-domain datasets, which showed promising performance for the four types of components we targeted. For the extrinsic evaluations, we demonstrated that our approach can improve both clustering and document retrieval task.

[2]http://www.lemurproject.org/galago.php

## Acknowledgments

This work was supported in part by the Center for Intelligent Information Retrieval and in part by NSF grant IIS-0910884. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsor.

## References

Kam-Fai Chan and Dit-Yan Yeung. 2000. Mathematical expression recognition: A survey.

Jinho D. Choi and Andrew McCallum. 2013. Transition-based dependency parsing with selectional branching. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, ACL'13, pages 1052–1062.

Shona Douglas and Matthew Hurst. 1996. Layout and language: lists and tables in technical documents. In *In Proceedings of ACL SIGPARSE Workshop on Punctuation in Computational Linguistics*, page pages.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. Liblinear: A library for large linear classification. *J. Mach. Learn. Res.*, 9:1871–1874, June.

Afef Kacem, Abdel Belaïd, and Mohamed Ben Ahmed. 2001. Automatic extraction of printed mathematical formulas using fuzzy logic and propagation of context. *IJDAR*, 4(2):97–108.

Kristina Lerman, Craig Knoblock, and Steven Minton. 2001. Automatic data extraction from lists and tables in web sources. In *In Proceedings of the workshop on Advances in Text Extraction and Mining (IJCAI-2001), Menlo Park*. AAAI Press.

Xiaoyan Lin, Liangcai Gao, Zhi Tang, Xiaofan Lin, and Xuan Hu. 2011. Mathematical Formula Identification in PDF Documents. In *International Conference on Document Analysis and Recognition*, ICDAR, pages 1419–1423.

Ying Liu, Kun Bai, Prasenjit Mitra, and C. Lee Giles. 2007. TableSeer: automatic table metadata extraction and searching in digital libraries. In *Joint Conference on Digital Library*, JCDL, pages 91–100.

Hwee Tou Ng, Chung Yong Lim, and Jessica Li Teng Koo. 1999. Learning to recognize tables in free text. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics on Computational Linguistics*, ACL '99, pages 443–450, Stroudsburg, PA, USA. Association for Computational Linguistics.

David Pinto, Andrew McCallum, Xing Wei, and W. Bruce Croft. 2003. Table extraction using conditional random fields. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval*, SIGIR '03, pages 235–242, New York, NY, USA. ACM.

Masakazu Suzuki, Fumikazu Tamari, Ryoji Fukuda, Seiichi Uchida, and Toshihiro Kanahori. 2003. Infty- an integrated ocr system for mathematical documents. In *Proceedings of ACM Symposium on Document Engineering 2003*, pages 95–104. ACM Press.

Suppawong Tuarob, Sumit Bhatia, Prasenjit Mitra, and C. Lee Giles. 2013. Automatic detection of pseudocodes in scholarly documents using machine learning. In *Proceedings of the 2013 12th International Conference on Document Analysis and Recognition*, ICDAR '13, pages 738–742, Washington, DC, USA. IEEE Computer Society.

Richard Zanibbi, Dorothea Blostein, and R. Cordy. 2004. A survey of table recognition: Models, observations, transformations, and inferences. *Int. J. Doc. Anal. Recognit.*, 7(1):1–16, March.