Improving Document Clustering by Eliminating Unnatural Language

Myungha Jang[1], **Jinho D. Choi**[2], **James Allan**[1]
[1]College of Information and Computer Sciences, University of Massachusetts
[2]Department of Computer Science, Emory University
mhjang@cs.umass.edu, jinho.choi@emory.edu, allan@cs.umass.edu

Abstract

Technical documents contain a fair amount of *unnatural language*, such as tables, formulas, and pseudo-code. Unnatural language can be an important factor of confusing existing NLP tools. This paper presents an effective method of distinguishing unnatural language from natural language, and evaluates the impact of unnatural language detection on NLP tasks such as document clustering. We view this problem as an information extraction task and build a multiclass classification model identifying unnatural language components into four categories. First, we create a new annotated corpus by collecting slides and papers in various formats, PPT, PDF, and HTML, where unnatural language components are annotated into four categories. We then explore features available from plain text to build a statistical model that can handle any format as long as it is converted into plain text. Our experiments show that removing unnatural language components gives an absolute improvement in document clustering by up to 15%. Our corpus and tool are publicly available.

1 Introduction

Technical documents typically include meta components such as figures, tables, mathematical formulas, and pseudo-code to effectively communicate complex ideas and results. Let us define the term *unnatural language* as text blocks that consist of only meta components as opposed to natural language that consists of body text.

There are many effective NLP tools available as the field has been advanced. However, these tools are mostly built for input text that are natural language. As many of our tools for NLP can be badly confused by unnatural language, it is necessary to distinguish unnatural language blocks from natural language blocks, or else unnatural language blocks will cause confusion for natural language processing. Once we salvage natural language blocks from the documents, we can exploit NLP tools much better as they are intended for. This phenomenon is emphasized in technical documents that have a higher ratio of unnatural language compared to non-technical documents such as essays and novels.

Document layout analysis aiming to identify document format by classifying blocks into text, figures, and tables has been a long-studied problem (O'Gorman, 1993; Simon et al., 1997). Most previous work have focused on image-based documents, PDF and OCR formats, and used geometric analysis on the pages using the visual cues from its layout. This was a clearly important problem in many applications in NLP and IR.

This work was particularly motivated while we attempted to cluster teaching documents (e.g., lecture slides and reading materials from courses) in technical topics. We discovered that unnatural language blocks introduced significant noise for clustering, causing spurious matches between documents. For example, code consists of reserved programming keywords and variable names. Two documents can contain two very different code blocks from one another but their cosine similarity is high because they share many terms by programming convention (Figure 1). (Kohlhase and Sucan, 2006) also recognized this problem by explaining main challenges of semantic search for mathematical formula: (1) Mathematical notation is context-dependent; without human's capability to understand the formula from the context, formulas are just *noise*. (2) Identical presentations can stand for multiple distinct mathematical objects.

Proceedings of the 3rd Workshop on Noisy User-generated Text, pages 122–130
Copenhagen, Denmark, September 7, 2017. ©2017 Association for Computational Linguistics

```
int binarySearch(int[]
array, int value, int left, int right)
{
      if (left > right)
            return -1;
      int middle = (left + right) / 2;
      if (array[middle] == value)
            return middle;
      else if (array[middle] > value)
            return binarySearch(array,
value, left, middle - 1);
      else
            return binarySearch(array,
value, middle + 1, right);
}
```

```
void merge_sort(int[] array)
    if length(array) <= 1
        return;

    int[] array left, right
    int middle = length(array)
    for each x in array before middle
        add x to left
    for each x in array after or equal middle
        add x to right

    left = merge_sort(left)
    right = merge_sort(right)

    return merge(left, right)
```

Figure 1: An example of how unnatural language confuses NLP tools. The left and right pseudo-code are very different, but standard NLP similarity functions such as cosine similarity can easily be confused by the terms highlighted in yellow.

This paper proposes a new approach for identifying unnatural language blocks in plain text into four types of categories: (1) TABLE (2) CODE (3) MATHEMATICAL FORMULA, and (4) MISCELLANEOUS (MISC). Text is extracted from technical documents in PDF, PPT, and HTML formats with little to no explicit visual layout information preserved. We focus on technical documents because they have a significant amount of unnatural language blocks (26.3% and 16% in our two corpora). Specifically, we focus on documents in slide formats, which have been underexplored.

We further study how removal of unnatural language improves two NLP tasks: document similarity and document clustering. Our experiments show that clustering on documents with unnatural language removed consistently showed higher accuracy on many of the settings than on original documents, with the maximum improvements up to 15% and 11% in two datasets, while it never significantly hurts the original clustering.

2 Related Work

2.1 Table Extraction

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 (Clark and Divvala, 2015; Liu et al., 2007). Existing techniques exploit the graphical features such as primitive geometry shapes, symbols, and lines to detect table borders. (Khusro et al., 2015) introduces and compares the state-of-the-art table extraction techniques from PDF articles. However, there does not appear to be any work that has attempted to process plain text extracted from richer formats, where table layouts are unpreserved.

2.2 Formula Extraction

Lin et al. (2011) categorized existing approaches for mathematical formulas detection by 'character-based' and 'layout-based' with respect to key features. (Chan and Yeung, 2000) provides a comprehensive survey of mathematical formula extraction using various layout features available from image-based documents. Since we have no access to layout information, character-based approaches are more relevant to our work. They use features of mathematical symbols, operators, and positions and their character sizes (Suzuki et al., 2003; Kacem et al., 2001).

2.3 Code Extraction

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 Problem Definition

Input to our task is the plain text extracted from PDF or PPT documents. The goal is to assign a class label to each line in that plain text, identifying it as natural language (regular text) or one of the

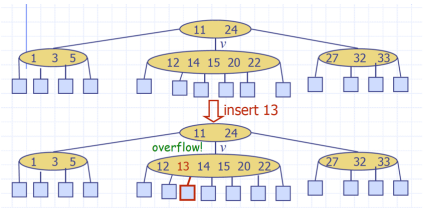
Chinese-to-English			
	NIST05	NIST06	NIST08
L-Hiero	25.57^{+}	25.27^{+}	18.33^{+}
AdNN-Hiero-E	26.37	25.93	19.42
AdNN-Hiero-D	26.21	26.07	19.54

```
Chinese-to-English
NIST05 NIST06 NIST08

L-Hiero 25.57+ 25.27+ 18.33+

AdNN-Hiero-E 26.37 25.93 19.42
AdNN-Hiero-D 26.21 26.07 19.54
```

Figure 2: A table in a PDF document (left) and its text-extracted version (right). Note that it is hard to distinguish the column headings from the extracted text without its layout.




```
11 2411 24
1 3 5 27 32 33
v
12 14 15 20 22
insert 13
overflow !
11 24
1 3 5 27 32 33
v
12 13 14 15 20 2212 13 14 15 20 22
```

Figure 3: An example of poor text extraction. The output from Apache Tika (right) has lost its original structure. Experiments will show that document clustering is improved by removing this kind of noise labeled as MISC

four types of unnatural language block components: table, code, formula, or miscellaneous text. In this work, we focus on these four specific types because our observations lead us to believe they are the most frequently occurring components in PPT lecture slides and PDF articles. Figures are also a frequent component but we do not consider them because they are commonly pictures or drawings and cannot be easily extracted to text. In this section, we briefly discuss the characteristics of each component and challenges in their identification from the raw text.

3.1 Table

Tables are prevalent in almost every domain of technical documents. Tables are usually conveyed by its two-dimensional layout and its column and/or row headings (Khusro et al., 2015). Tables typically have multiple cells merged for layout, which makes them particularly difficult to distinguish as a table once they are converted to flat text.

3.2 Mathematical Formula

Mathematical formulas exist in two ways: isolated formulas on their own lines or as formulas embedded within a line of text. In this work, we treat both types as a formula component. Because not all math symbols can be matched to Unicode characters and because the extraction software may not convert them correctly, the extracted text tends to contain more oddly formatted or even completely wrong characters. Superscripts and subscripts are no longer distinguishable and the original visual layout (e.g., math symbols over multiple lines such as Π and $\sum$) is lost.

3.3 Code

Articles in Computer Science or related fields often contain pseudo-code or actual program code to illustrate their algorithm. We assume that even indents, one of the strong code visual cues, are not preserved in the extracted text although some extraction tool saves them, not to limit ourselves to the detailed performances of text extraction tools.

3.4 Miscellaneous Non-text (Misc.)

In addition to the components mentioned above, there are other types of unnatural language blocks that are left during conversion to text and that may provide spurious sub-topic matches between documents. To allow for those, we denote those components as miscellaneous text. One example of miscellaneous text is the text and caption that are part of the diagrams in slides. Figure 3 shows an example of miscellaneous text that lost its structure and meaning while being converted to text without the original diagram.

4 Corpus

4.1 Data Collection

We collected 1,561 lecture slides from various Computer Science and Electrical Engineering courses that are available online, and 5,898 academic papers from several years of ACL/EMNLP

Purpose	Name	Content
Classification Training	T_{SLIDES}	35 lecture slides (8,514 lines) whose components are annotated
	T_{ACL}	35 ACL papers (25,686 lines) whose components are annotated
	$T_{COMBINED}$	Combination of T_{SLIDES} and T_{ACL}
Word Embedding Training	$T_{WORD2VEC}$	1,190 lecture slides and 5,863 ACL/EMNLP papers archived over a few years that are used for training word embedding.
Clustering	C_{DSA}	128 lecture slides from 'data structure' and 'algorithm' classes
	C_{OS}	300 lecture slides from, 'operating system' classes

Table 1: Datasets used in our paper. All data are available for download at [http://cs.umass.edu/~mhjang/publications.html]

archive[1]. We divided the dataset for several purposes: training the classification model, training word embedding model for feature extraction, and clustering for extrinsic evaluation. The details of the dataset we used are summarized in Table 1. We make the data publicly available for download at http://cs.umass.edu/~mhjang/publications.html.

For classification, we constructed three datasets using two different data sources: (1) lecture slides, (2) ACL papers, and (3) a combination of both. We chose these two types of data sources because they have different ratios of unnatural language components and complement each other for coverage. Table 2 shows the ratio of the four components from each annotated dataset. For example, 1.4% of lines in T_{SLIDES} are annotated as part of table.

4.2 Text Extraction

We extracted plain text from our datasets using an open-source software package, Apache Tika. The package is available for text extraction from various formats including PDF, PPT, and HTML.

4.3 Annotation

To train a statistical model, we need ground-truth data. We created annotation guidelines for the 4 types of unnatural language components and annotated 35 lectures slides (7,943 lines) and 35 ACL papers (25,686 lines). We developed an annotation tool to support the task and also to enforce annotators to follow certain rules[2]. We hired four undergraduate annotators who have knowledge of the Computer Science domain for this task.

[1]https://aclweb.org/anthology

[2]The guidelines and the tool are available at http://cs.umass.edu/~mhjang/publications.html

	TABLE	CODE	FORMULA	MISC	ALL
T_{SLIDES}	1.4%	14.6%	0.5%	9.8%	26.3%
T_{ACL}	4.0%	0.6%	5.0%	6.4%	16%

Table 2: % of lines by unnatural category. Both datasets have quite a bit of unnatural language (26.3% for T_{SLIDES} and 16% for T_{ACL}), though T_{ACL} has more TABLES and FORMULAS and less CODE.

5 Features

We find line-based prediction has an 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. We introduce five sets of features used to train our classifier and discuss each feature's impact on the accuracy.

5.1 N-gram (N)

Unigrams and bigrams of each line are included as features.

5.2 Parsing Features (P)

Unnatural languages are not likely to form any grammar structure. When we attempt to parse the unnatural language line, the resultant parsing tree would form unusual syntactic structure. To capture this insight, we parse each line using the dependency parser in ClearNLP (Choi and McCallum, 2013) and extract features such as the set of dependency labels, the ratio of each POS tag, and POS tags of each dependent-head pair from each parse tree.

5.3 Table String Layout (T)

Text extracted from tables loses its visual layout as a table but still preserves implicit layout through its string patterns. Tables tend to convey the same

type of data along the same column or row. For example, if a column in a table reports numbers, it is more likely to contain numeral tokens in the same location of the lines of the table in parallel. Hence, a block of lines will more likely be a table if they share the same pattern. We encode each line by replacing each token as either S (String) or N (Numeral). We then compute the edit distance among neighboring lines weighted by language modeling probability computed from the table corpus (Equation 1, 2).

$$P_{table}(l_i) \propto P_{table}(l_i|l_{i-1}) = TableLanguageModel(l_i) \cdot editDistance(encode(l_i), encode(l_{i-1})) \quad (1)$$

$$TableLanguageModel(l_i) = \Pi_j^n(P(encode(t_{i,j+1})|encode(t_{i,j})) \quad (2)$$

where l_i refers to a i-th line in a document, $t_{i,j}$ refers to a j-th token in l_i.

5.4 Word Embedding Feature (E)

We train word embeddings using $T_{WORD2VEC}$ using WORD2VEC (Mikolov et al., 2013). The training corpus contained 278,719 words. Since we do a line-based prediction, we need a vector that represents the line, not each word. We consider three ways of computing a line embedding vector: (1) by averaging the vector of the words, (2) by computing a paragraph vector introduced in (Le and Mikolov, 2014), and (3) by using both.

5.5 Sequential Feature (S)

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.

6 Classification Experiments

We use the Liblinear Support Vector Machine (SVM) (Chang and Lin, 2011) classifier for training and run 5-fold cross-validation for evaluation. To improve the robustness of structured prediction, we adopt a learning to search algorithm known as DAGGER to SVM (Ross et al., 2010). We first introduce two baselines to compare the accuracy against our statistical model.

6.1 Baselines

Since no existing work is directly applicable to our scenario, we consider two straightforward baselines.

- **Weighted Random (W-Random)**
  This assigns the random component class to each line. Instead of uniform random prediction, we made more educated guesses using the ratio of components known from the annotated dataset (Table 2).

- **Component Language Modeling (CLM)**
  Among the five language models of the five component classes (the four non-textual components and text component) generated from the annotations, we predict the component for each line by assigning the component whose language model gives the highest probability to the line.

6.2 Classification Result

We first conduct single-domain classification. Annotations within each dataset, T_{SLIDES} and T_{ACL} are split for training and testing using 5-fold cross validation scheme. Table 3 reports F1-score for prediction of the four components in the two dataset using our method as well as baselines.

	Precision	Recall	F1-score
TABLE	94.60	76.39	**84.53**
CODE	89.56	84.01	**86.69**
FORMULA	85.07	79.32	**82.10**
MISC	85.59	90.24	**87.86**
TEXT	97.76	98.79	98.27

Table 4: Multi-domain classification improves the single-domain classification in Table 3. Identification of categories with particularly low accuracy in each datasets (TABLE and FORMULA in T_{SLIDES} and CODE in T_{ACL}) are improved to be as good as the other categories.

The proposed method dramatically increased the prediction accuracies for all of the components against the baselines. CLM baseline showed the highest accuracy on CODE among the four categories in both datasets. Because pseudo-code use more controlled vocabulary (e.g., reserved words and common variable names), the language itself becomes distinctive characteristics. We also include the numbers reported by Tuarob et al. (2013)

	T_{SLIDES}				T_{ACL}			
	TABLE	CODE	FORMULA	MISC	TABLE	CODE	FORMULA	MISC
W-Random	1.69	14.62	2.82	10.57	4.15	0.62	4.44	6.08
CLM	5.41	28.62	0.00	10.47	13.10	16.45	10.32	5.18
Proposed Method	**67.89**	**90.22**	**29.09**	**89.63**	**86.58**	**63.70**	**80.98**	**87.63**
PC-CB (Tuarob et al., 2013)	N/A	75.95	N/A	N/A	N/A	75.95	N/A	N/A

Table 3: Single-domain Classification Result in F1-score: Proposed method is much better than baselines for classifying unnatural language. Note that we borrowed the F1-score reported on their dataset for reference. The number is not directly comparable to other numbers since the datasets are different.

for comparison. Since their dataset was 258 PDF scholarly articles, T_{ACL} is more a comparable dataset than T_{SLIDES}, but our training set is much smaller than their dataset. However, their number reported on Table 3 is not directly comparable to other numbers because the numbers are on different datasets.

In T_{SLIDES}, the classification F1-score for FORMULA is relatively low as 29.09% compared to the other components in the same dataset, and also compared to the FORMULA prediction in T_{ACL} (80.98%). This is due to too small amount of training data (only 0.5% of FORMULA in T_{ACL}), which is overcome in T_{SLIDES} that contain 5% of FORMULA training data (refer to Table 2).

In the proposed method, classification of CODE and MISC was significantly improved in T_{SLIDES} (around 90%), while that of TABLE and FORMULA was improved in T_{ACL} (over 80%). This shows the complementary nature between the two datasets, which suggests that a combination of both, $T_{combined}$, would further improve classification performance. Table 4 shows the multi-domain classification result using $T_{combined}$, in which all four categories are identified with an F1-score higher than 80%.

6.3 Feature Analysis

We conducted feature analysis to understand the impact of single feature and their combination. We started from single features and incrementally combined them to observe the performance (Figure 5). Features are added in a greedy fashion such that a feature that gives the higher accuracy when used alone is added first.

We first compare the three ways of computing sentence vector features mentioned in Section 5 (Figure 4). When we experiment with only embedding features, averaging word vectors performed 9-12 times better than paragraph vectors. When

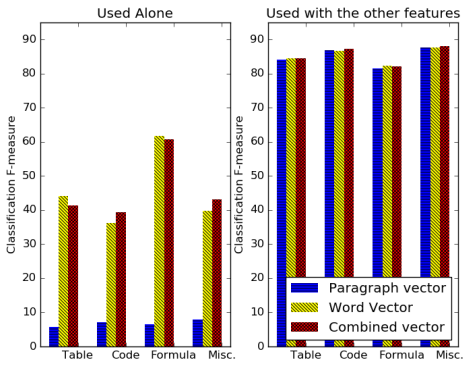


Figure 4: Three ways of computing sentence embedding vector

both features were used, there are some gains in CODE and MISC but losses in TABLE and FORMULA. However, when we experiment with all the other features in addition to embedding features, losses were covered by the other features such that combined vectors give overall the highest performances.

N-gram (N) features was the most powerful feature with 68% of F1-score when used alone. The next useful features are parsing feature (P), table layout (T), and embedding features (E) in order for TABLE, while embedding vectors were more effective than parsing feature for CODE (Figure 5).

7 Removal Effects of Unnatural Language on NLP tools

We observe how removal of unnatural language from documents affects the performance of two NLP tools: document similarity and document clustering. For the set of experiments, we prepared a gold standard clustering for each dataset, C_{DSA} and C_{OS}.

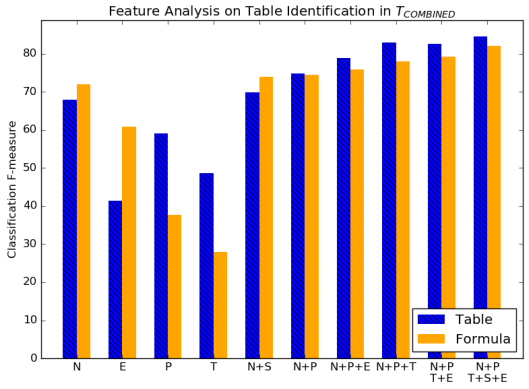


Figure 5: Feature analysis for TABLE and FORMULA identification in $T_{combined}$. N: N-gram, E: Embedding, P: Parsing, T: Table String Layout, S: Sequential.

7.1 Document Similarity

If two documents are similar, they must be topically relevant to each other. A good similarity measure should reflect that; two topically relevant documents should have a high similarity score. To test whether the computed similarity reflects the actual topic relevance better once the unnatural language is removed, we conduct regression analysis.

We convert the gold standard clustering to pairwise binary relevance. If two documents are in the same ground-truth cluster, they are relevant, and otherwise irrelevant. We then fit a log-linear model in R for predicting binary relevance from the cosine similarity of document pairs.

Regression models fitted in R are evaluated using AIC (Akaike, 1974). The AIC is a measure used as a means for model selection, which measures the relative quality of statistical models learned from the given data. When AIC is smaller, the fit is better and the complexity of the model is smaller since it requires fewer parameters. Table 5 shows that AIC was reduced by 53 and 118 respectively on the models trained with documents whose unnatural language blocks are removed, compared to the original documents. Since AIC does not provide a test for a model, AIC does not suggest anything about the quality of the model in an absolute sense, but relative quality. From this result, we can conclude that cosine similarity can fit a better model that predicts documents' topic relevance with significance after unnatural language blocks have been removed.

	AIC($D_{original}$)	AIC($D_{removed}$)	Improvement
C_{DSA}	-40975	-41028	-53
C_{OS}	-61404	-61522	-118

Table 5: The statistical model is trained better with documents whose unnatural language categories are removed ($D_{removed}$) than the model with the original documents ($D_{original}$) in both datasets. *Smaller* AIC scores imply *better* models.

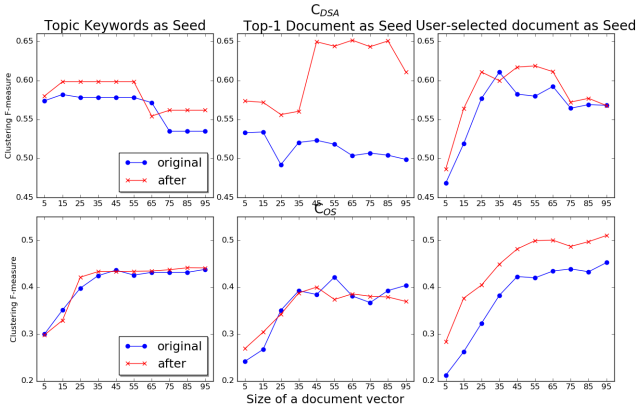


Figure 6: Clustering result on two datasets, C_{DSA} (top) and C_{OS} (bottom). X axis referes to the the size of document vector K, which controls the top-K TF-IDF terms included from documents. Y axis: Clustering F1-score.

7.2 Document Clustering

Comparing general clustering performance on two document sets is tricky because clustering performance varies by many factors, e.g., clustering algorithm, similarity function, document representation, and parameters. To make a safe claim that clustering quality of one set of documents is better than the other, clustering on one set should consistently outperform the other under many different settings. To validate this, we perform clustering experiments with multiple settings such as different document vector size and and initialization schemes.

In this experiment, we consider seeded K-means clustering algorithm (Basu et al., 2002) for teaching documents. In our application scenario, users initially submit a topic list (e.g., syllabus) of the course. Then lecture slides are grouped into the given topic cluster. Depending on users' interaction level, we consider a semi-interactive scenario where users only provide a topic list, and a fully-interactive setting where users not only provide a topic list but also provide an answer document for each topic cluster, further specifying the intended topic.

Input: Set of document vectors D = $\{d_1, ...d_n\}$, $d_i \in R^T$, set of seed vectors S = $\{s_1, ...s_k\}$, user-provided topic keywords vector T = $\{t_1, ...t_k\}$
Result: Disjoint K partitioning of D into $C_{l=1}^{k}$
Seed Initialization:
if *Topic-keywords seeding* **then**
  $s_i = t_i$
if *Top-1 document seeding* **then**
  $s_i = d_j$,
  $argmax_j(\text{COSINESIMILARITY}(t_i, d_j))$
if *User-selected document seeding* **then**
  $s_i =$DOCSELECTEDBYUSER(t_i)
while *convergence* **do**
  K-means clustering document selection process

Algorithm 1: Seeded K-means with User Interaction

In a semi-interactive setting, topic keywords are sparse seeds as they usually consist of two or three words. Therefore, we expand the topic keywords by finding the top-1 document retrieved from the keywords and use it as a seed. For experiments, we simulate the fully-interactive setting; instead of having an actual user to pick an answer document, we use an answer document randomly chosen from a gold cluster. The seeded K-means clustering algorithm with three interactive seeding schemes is described in Algorithm 1.

A simulated setting is more realistic when the selected document is suggested to the user as the top or near-top choice. In our dataset, 60% of the selected documents were ranked in top 10 in C_{DSA}, and 13% of the selected documents were ranked in top 10 in C_{OS}, which implies that the simulated setting in C_{DSA} was more realistic than in C_{DSA}. For top-1 document seeding, 64% and 78% of document seeds matched with the gold standard in C_{DSA} and C_{OS}, respectively.

Figure 6 shows the clustering result of original documents ($D_{original}$) and documents whose unnatural language blocks are removed ($D_{removed}$), with three different seeding schemes over two lecture slide datasets. In C_{DSA}, $D_{removed}$ consistently outperformed with all three seeding schemes. The clustering performed the best with $D_{removed}$ when top-1 document was used as a seed. Overall, in C_{DSA}, clustering was improved 94% of the time with the maximum absolute gain of 14.7% and the average absolute gain of 4.6%. The average absolute loss was 0.8% when 6% of the time the removal of unnatural language made the clustering worse. In C_{OS}, clustering was improved 73% of the times with the maximum absolute gain of 11.4% and the average absolute gain of 3.9%. The average absolute loss was 1.7%. Our results suggest that removal of unnatural language blocks can significantly improve clustering most of the times with a bigger gain than occasional losses.

8 Conclusion

In this paper, we argued that unnatural language should be distinguished from natural language in technical documents for NLP tools to work effectively. We presented an approach to the identification of four types of unnatural language blocks from plain text, which is not dependent on document format. The proposed method extracts five sets of line-based textual features, and had an F1-score that was above 82% for the four categories of unnatural language. We showed how existing NLP tools can work better on documents if we remove unnatural language from documents. Specifically, we demonstrated removing unnatural language improved document clustering in many settings by up to 15% and 11% at best, while not significantly hurting the original clustering in any setting.

Acknowledgments

This work was supported in part by the Center for Intelligent Information Retrieval and in part by NSF grant #IIS-1217281. 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

Hirotugu Akaike. 1974. A new look at the statistical model identification. *Automatic Control, IEEE Transactions on*, 19(6):716–723.

Sugato Basu, Arindam Banerjee, and Raymond J. Mooney. 2002. Semi-supervised clustering by seeding. In *Proceedings of the Nineteenth International Conference on Machine Learning*, ICML '02, pages 27–34, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

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

Chih-Chung Chang and Chih-Jen Lin. 2011. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27.

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.

Christopher Clark and Santosh Divvala. 2015. Looking beyond text: Extracting figures, tables and captions from computer science papers. In *AAAI Workshops*.

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.

Shah Khusro, Asima Latif, and Irfan Ullah. 2015. On methods and tools of table detection, extraction and annotation in pdf documents. *J. Inf. Sci.*, 41(1):41–57.

Michael Kohlhase and Ioan Sucan. 2006. A search engine for mathematical formulae. In *AISC*, volume 4120 of *Lecture Notes in Computer Science*, pages 241–253. Springer.

Quoc V. Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. *CoRR*, abs/1405.4053.

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.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

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.

L. O'Gorman. 1993. The document spectrum for page layout analysis. *IEEE Trans. Pattern Anal. Mach. Intell.*, 15(11):1162–1173.

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.

Stéphane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. 2010. No-regret reductions for imitation learning and structured prediction. *CoRR*, abs/1011.0686.

Anikó Simon, Jean-Christophe Pret, and A. Peter Johnson. 1997. A fast algorithm for bottom-up document layout analysis. *IEEE Trans. Pattern Anal. Mach. Intell.*, 19(3):273–277.

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.