Bull. Nov. Comp. Center, Comp. Science, 47 (2023), 43-51
@ 2023 NCC Publisher

Extraction of the author’ s terminology and definitions from mathematical texts

Alexander Turnaev, Zinaida Apanovich

Abstract. This paper describes a pipeline for extracting the author’ s terms and definitions from mathematical texts. We used two models: one, for detecting mathematical formulas to clear text from noise and the other, for converting images into LaTeX formulas to restore the deleted formulas. Experimental data show that noise clearing is an essential step, because it improves all quality metrics. To recognize the author’ s terminology, we applied the rule-based syntactic approach. The idea of “negative” rules shown here increases final precision significantly, though does not essentially reduce recall. In general, the results obtained are quite good, because so far there are no other solutions for recognizing the authors’ terms in mathematical texts, and the quality metrics are comparable to more general methods that would work worse in the mathematical domain.

Keywords: natural language processing, information extraction, terminology extraction, definition extraction, entity recognition.

Introduction

Modern science and technology have generated a multitude of new theories, materials, technologies, and concepts. At the same time, a wide use of computers and the Internet has provided faster and more efficient access to the latest results kept in the vast and rapidly growing collections of scientific literature. This expansion of knowledge and demand for access to this knowledge has created new challenges in the fields involved in the development of intelligent information processing systems using the computerized extraction of domain specific knowledge. The functionalities of such intelligent systems are largely determined by the comprehensiveness and accuracy of their knowledge bases. The value of the many kinds of intelligent domain specific information processing, such as automatic indexing, information extraction, automatic question answering systems, and automatic summarization, is ultimately dependent on the quality of the underlying knowledge graphs. In the context of natural language processing, knowledge graphs consist of concepts and their relations. To construct a knowledge graph for a particular field of science, one should identify the central concepts in this field. The task of identifying these central concepts in a particular field is known as *term extraction* or *concept extraction*. We are especially interested in the special kind of terms which are either defined or explained in the text of a paper. We will refer to these concepts as the *author’ s concepts*.

One of important scientific domains is mathematics. Extracting terminology from mathematical texts is a particularly difficult task because of the numerous formulas, which can make mathematical texts noisy. Moreover, a correct analysis of the text requires translating all the formulas into a text representation, for example, into the LaTeX code.

At the moment, we are not aware of any works dealing with the problem of extracting the authors' terminology from mathematical texts. Therefore, this paper proposes a complete pipeline for extracting the author's terms and their definitions from mathematical texts. This pipeline consists of the following steps.

1. Preprocessing text in order to remove noise and to normalize data. In particular, removing noise from a document implies solving the problem of mathematical formulas detection followed by breaking the text into fragments starting with the words "Theorem", "Definition", "Lemma", and "Remark".

2. Using the Named Entity Recognition (NER) model to recognize the author's terms defined in this chunk of text.

3. Post-processing of data. In particular, converting the images of mathematical formulas into the LaTeX code.

The paper is structured as follows. Section 1 provides an overview of the methods for solving the related problems. Section 2 contains a complete description of the pipeline for extracting the author's terms and definitions. Section 3 presents the results of numerical experiments. The developed pipeline can be used in further research.

1. Related work

At the moment, we are not aware of any papers considering the extraction of newly defined terms and their definitions from mathematical texts. However, some of the closest works are [1] and [2].

Paper [1] presents a comparison of several tools for extracting the mathematical concepts (terms) that are not necessarily the author's. These tools are DyGIE++ [3], OpenTapioca [4], Parmenides [5], and TextRank [6]. DyGIE++ [3] is a span-based neural scientific entity extractor. OpenTapioca [4] is a simple named entity linking system that links the phrases of a natural language text to the entities in WikiData [7]. It cannot identify new terms – only those already represented in WikiData. Parmenides [5] takes a linguistic approach to terminology extraction. It uses spaCy to identify a syntactic structure, then normalizes it and identifies the phrases for extraction. TextRank [6] is a graph-based ranking algorithm based on PageRank, which has been applied to keyword extraction and text summarization, as well as to automatic terminology extraction. The quality metrics of these tools are shown in Table 1. It can be seen that the results obtained are quite poor.

Table 1. The quality metrics of the models presented in [1]

	DyGIE++	OpenTapioca	Parmenides	TextRank
Precision	0.26	0.31	0.07	0.16
Recall	0.36	0.22	0.91	0.56
F_1	0.30	0.26	0.12	0.24

A similar problem, namely, the problem of binary classification "whether a block of text contains a definition," was dealt with in [2]. However, this problem was not considered for mathematics. In general, the models were tested on two data sets, WCL [8] and W00 [9]. WCL and W00 are two sets of manually annotated definitions and distractors. The former, taken from Wikipedia, contains mostly general terminology. The latter is from the ACL-ARC anthology and contains novel terminology in the NLP research papers. Table 2 shows the quality metrics of the best models obtained on these two datasets. The CNN means Convolutional Neural Networks, the BLSTM is Bidirectional Long Short Term Memory Neural Network, and the C-BLSTM100d is a combination of the CNN and BLSTM.

Table 2. Quality metrics for the binary classification problem "whether a block of text contains a definition" [2]

	WCL			W00		
Metrics	P	R	F_1	P	R	F_1
CNN_d	90.6	90.9	90.7	34.2	69.4	45.8
CNN_1	94.2	94.2	94.2	42.8	65.5	51.3
C-BLSTM100_d	93.2	92.2	92.6	52.0	67.6	57.4

Here P, R and F1 mean precision, recall and F1, respectively.

It can be seen that the results for the second data set are much worse, and they differed, in fact, only by the syntactic constructions for definition descriptions. That is, each model is sensitive to the field of application. Naturally, in contrast to other fields, the formalism developed by mathematicians for text writing has a unique syntactic structure. In addition, taking into account the strong impact of noise generated by formulas in mathematical texts on the final metrics, the results of these models would be even worse. We have managed, among other things, to extract the sequences of tokens of the terms defined and go beyond solving the binary classification problem.

2. Pipeline description

2.1. Preprocessing

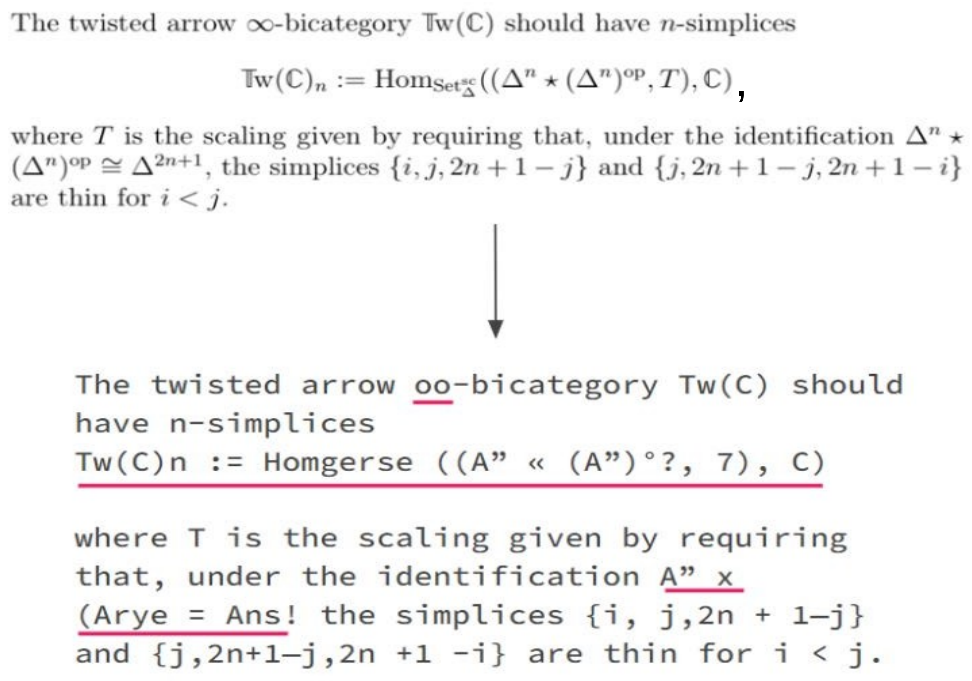
The preprocessing step depends significantly on the document reading tool. There are two approaches to document reading: an optical character recognition (OCR) and conventional PDF readers. We have chosen the tesseract[1] program to read our set of documents.

[1] https://github.com/tesseract-ocr/tesseract

Tesseract is an optical character recognition (OCR) tool currently developed by Google. The choice of the OCR approach, rather than a conventional PDF-reader, is due to the following reasons:

1. Using a program to read a specific document format obviously imposes a restriction on the document collection format.
2. The common problem of corrupted metadata (for example, the location of characters) may complicate pre-processing a document performed in order to clean it of noise.
3. This approach does not allow you to read mathematical formulas correctly.

However, both approaches generate a lot of noise in text with mathematical formulas. An example of such noise generated by mathematical formulas is shown in Figure 1.



The twisted arrow ∞-bicategory $\mathbb{T}\mathrm{w}(\mathbb{C})$ should have n-simplices

$$\mathbb{T}\mathrm{w}(\mathbb{C})_n := \mathrm{Hom}_{\mathrm{Set}_\Delta^{sc}}((\Delta^n \star (\Delta^n)^{\mathrm{op}}, T), \mathbb{C}),$$

where T is the scaling given by requiring that, under the identification $\Delta^n \star (\Delta^n)^{\mathrm{op}} \cong \Delta^{2n+1}$, the simplices $\{i, j, 2n+1-j\}$ and $\{j, 2n+1-j, 2n+1-i\}$ are thin for $i < j$.

$$\downarrow$$

```
The twisted arrow oo-bicategory Tw(C) should
have n-simplices
Tw(C)n := Homgerse ((A” « (A”)°?, 7), C)

where T is the scaling given by requiring
that, under the identification A” x
(Arye = Ans! the simplices {i, j,2n + 1—j}
and {j,2n+1—j,2n +1 -i} are thin for i < j.
```

Figure 1. An example of noise generated by mathematical formulas

As shown in Section 3, the produced noise has a negative effect on the subsequent quality metrics for extracting the authors' terms. Therefore, we used, as the preprocessing step, an additional tool for mathematical formulas detection. As a formula detection tool, we have chosen the model that took first place at the ICDAR 2021 (International Conference on Document Analysis and Recognition)[2] [10] competition for mathematical

[2] https://icdar2021.org/

formulas detection. The tool [10] leverages the deformable convolutional networks (DCN) [11] combined with Split-Attention Networks (ResNeSt) [12]. Table 3 demonstrates good F1-measure indicators of the model.

Table 3. F1-measure of the formula detection models [10]

Method	Embedded	Isolated	Total
ResNeSt50-DCN	95.67	97.67	96.03
ResNeSt101-DCN	96.11	97.75	96.41

To extract mathematical formulas, the 50-layer model ResNeSt50-DCN [10] was used, because it has a positive effect on the processing speed, and the difference of F1 measures between the models is not significant.

Small formulas fit in one line of a text and *large formulas* take up to several lines. To avoid losing syntactic dependencies, large and small formulas are replaced with ISOLATED and EMBEDDED keywords, respectively, and removed from the document.

After that, the text obtained is normalized. First, each text is divided into the blocks starting with the words "Theorem", "Definition", "Lemma", and "Remark". Then we apply stemming, tokenization, lemmatization, POS-tagging, syntax dependency tagging, etc., using the python spaCy [3] library and its pre-trained model "en_core_web_lg".

2.2 Extracting term titles

Each block extracted from the preprocessed text corresponds to at least one term of the author. Some part of the block is a description (semantics) of a certain author's term and, obviously, its title is included in the text of its description. Therefore, it is necessary to extract only the titles of the author's terms. To solve this problem, our rule-based approach uses the Matcher and DependencyMatcher classes from the spaCy library.

The Matcher rules are based on POS (Part-Of-Speech) tags. In fact, each rule of the Matcher is a modified regular expression that takes into account POS-tags. Similar approach to the extraction of terms (not necessarily the author's) was used in Parmenides [5]. Nevertheless, Table 1 shows that, although Parmenides has an excellent completeness metric, its accuracy is not satisfactory. Therefore, when creating our model, we used only three rules of the Matcher class, which are reliable and do not recognize unnecessary tokens (see Table 4).

[3] http//spacy.io

Table 4. Positive and negative rules for the Matcher

№	Pattern	Type
1	is $(S)^+$ if	positive
2	called $(S)^+$ if	positive
3	DET $(S)^+$is $(S)^+$ if	negative

The DependencyMatcher rules are based on the syntactic dependencies in the parse tree, though POS-tags can also be used. The rules describe the restriction on traversing the parse tree from a certain vertex. Figure 2 shows an example of the parse tree and arranged POS-tags for the fragment of the sentence “The operation is called associative if...” POS tags are shown below each word. The dependency relations between words are shown as labels of the edges.

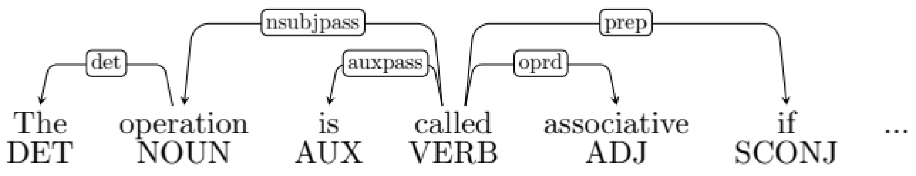
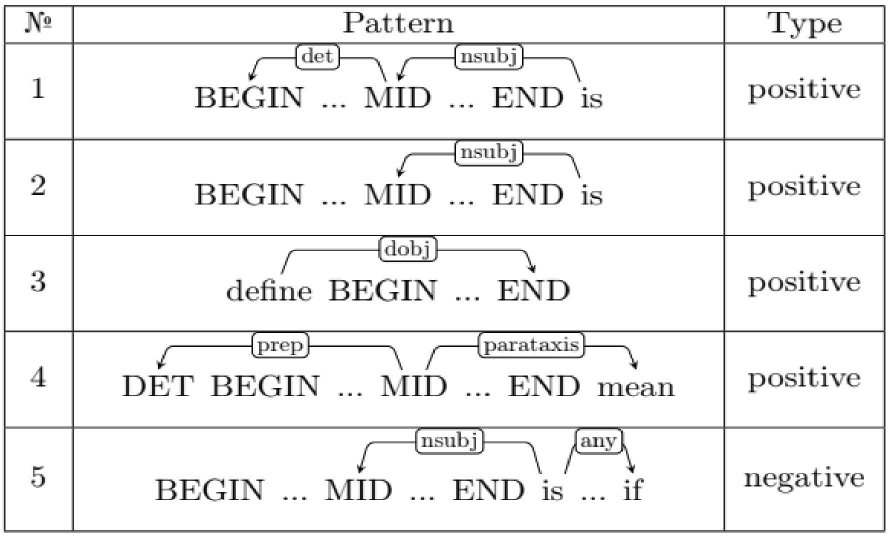


Figure 2. An example of a dependency tree and POS-tags

The approach to extracting the concepts using the DependencyMatcher rules is much more powerful than that implemented by the Matcher class.

It is worth saying that the conventional rules for the correct recognition of the author’s terms are not sufficient, because even the simplest rule described by the “SUBJECT is/are/... DEFINITION” pattern can recognize a term that is not authored.

For example, the “SUBJECT” is preceded by the word “let”. To solve this problem, we decided to create so-called “negative” rules, that is, the rules prescribing which sequences of tokens should be definitely not recognized, even if they are recognized by the conventional rules, which will now be called “positive”. Figure 3 shows some of the rules significantly affecting the final quality metrics.



№	Pattern	Type
1	BEGIN ... MID ... END is	positive
2	BEGIN ... MID ... END is	positive
3	define BEGIN ... END	positive
4	DET BEGIN ... MID ... END mean	positive
5	BEGIN ... MID ... END is ... if	negative

det
nsubj
nsubj
dobj
prep
parataxis
nsubj
any

Figure 3. Positive and negative rules for the Dependency Matcher

In general, our program has included many more rules, but the rules not mentioned here affect less the final quality metrics, totaling ≈ 10% for precision and recall.

2.3. Post-processing

Since formulas can be included in the descriptions and even titles of terms, it is necessary to restore the formulas previously replaced with the keywords “EMBEDDED” and “ISOLATED”. To do this, the image-to-latex[4] model has been used. The Character Error Rate (CER) metric of the model is equal to 0.17. The Character Error Rate is the percentage of the characters transcribed incorrectly by the Text Recognition model. When CER is equal to 0, the result is ideal.

3. Evaluation

To test the model, the TAC (Theory and Applications of Categories) data set[5] was used. About 400 chunks of text were manually annotated and other 400 synthetic chunks were obtained by replacing token sequences with the same POS-tags. Table 5 demonstrates standard quality metrics precision, recall and F1-measure for our approach.

[4] http://www.tac.mta.ca/tac/

[5] https://github.com/kingyiusuen/image-to-latex

Table 5. Evaluation of the rule-based author’ s terms extraction method

Metric	Noise-free data	Noisy data
Precision	0.75	0.26
Recall	0.82	0.47
F_1	0.79	0.33

A comparison of our results with those obtained in [1] and [2] (see Tables 1 and 2, respectively) shows that our model is more effective. Clearing text of the noise generated by mathematical symbols significantly improves the final quality metrics.

4. Conclusion

Based on the results of the work, a pipeline has been created to solve the problem of the automatic extraction of the author’ s terms. A rule-based model has been developed for extracting the titles of the author’ s terms using the analysis of syntactic dependencies. The approach to clearing the text of noise makes a significant contribution to the final metrics.

In comparison with similar research, the results obtained here are quite good. The rule-based model can be used not only for the named entity recognition task, but also for the binary classification task “does a block of text contain a definition” . Narrowing down from ordinary terms to the author’ s terms is an important step, since the metrics of the existing models for the task of extracting mathematical terms are rather poor.

References

[1] Collard J., De Paiva V., Fong B. Subrahmanian E. Extracting Mathematical Concepts from Text. – https://doi.org/10.48550/arXiv.2208.13830.

[2] Espinosa-Anke L., Steven Schockaert S. Syntactically aware neural architectures for definition extraction // Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. Proc. – New Orleans, Louisiana, USA. – 2018. – Vol. 2 (Short Papers).

[3] Wadden D., Wennberg U., Luan Y., Hajishirzi H. Entity, relation, and event extraction
[4] with contextualized span
representations. – https://doi.org/10.48550/arXiv.1909.03546.

[5] Delpeuch A. Opentapioca: Lightweight entity linking for Wiki data. – https://doi.org/10.48550/arXiv.1904.09131.

[6] Bhat T., Elliott J., Kattner U., Campbell C., Subrahmanian E., Sriram R., Collard J., Ira M. Generating domain terminologies using root- and rule-based terms // J.

Washington Acad. Sci. – 2018. – Vol. 104, No. 4. – P. 31 – 78.

[7] Mihalcea R. Textrank: Bringing order into text // Empirical Methods in Natural Language Processing. Proc. – 2004. – P. 404 – 411.

[8] Vrandečić D., Krötzsch M. Wikidata: a free collaborative knowledgebase // Communications of the ACM. – 2014. – Vol. 57, Iss. 10. – P. 78 – 85. – https://doi.org/10.1145/2629489.

[9] Navigli R., Velardi P., RuizMart´ınez J. M. An annotated dataset for extracting definitions and hypernyms from the web. // Language Resources and Evaluation. Proc. LREC' 10. – Valletta, Malta. – 2010.

[10] Jin Y., Kan M.-Y., Ng J.-P., He X. Mining scientific terms and their definitions: A study of the ACL anthology // Empirical Methods in Natural Language Processing. Proc. – Seattle, Washington, USA. – 2013. – P. 780 – 790.

[11] Zhong Y., Qi X., Li S., Gu D., Chen Y., Ning P., Xiao R. 1st place solution for ICDAR 2021 competition on mathematical formula detection. – https://doi.org/10.48550/arXiv.2107.05534.

[12] Dai J., Qi H., Xiong Y., Li Y., Zhang G., Hu H., Wei Y. Deformable convolutional networks // IEEE international conference on computer vision. Proc. – 2017. – P. 764 – 773.

[13] Zhang H., Wu C., Zhang Z., Zhu Y., Lin H., Zhang Z., Sun Y., He T., Mueller J., Manmatha R. et al. Resnest: Split-attention networks. – https://doi.org/10.48550/arXiv.2004.08955.