WIP: Creating a Database of Definitions From Large Mathematical Corpora

Luis Berlioz

University of Pittsburgh
lab232@pitt.edu

Abstract

We propose a method to gather large amounts of definitions from mathematical documents available online. Recent work indicates that text classification algorithms can have excellent accuracy at determining when a certain paragraph is in fact a definition or not. These algorithms are trained on large math corpora available online like the arXiv website. The LaTeX source code of these documents is first converted into a more structured format like XML or HTML with the software package LaTeXML. The training data for the classifier is then obtained by searching for the definitions that the author labeled with a LaTeX macro. The second phase of the system consists of extracting the term being defined from each definition. This task is performed by a Named Entity Recognition (NER) model trained using data from websites with mathematical content. The data is finally organized according to several different properties like semantic similarity and content dependency.

1 Introduction

In this paper we describe a system for the extraction of definitions and definienda from large collections of digital mathematical documents like the arXiv website. The main objective of this system is to organize all the mathematical lexicon both by dependency and semantically. This can be done using the content of each definition, since the definition of a new term depends on previously ones. And, by clustering terms that occur in similar contexts. We also go over the implementation of a prototype of such system and the processing of the different sources of digital documents used to create this first implementation. The resulting system, although unfinished, provides a convincing proof of concept as well as a baseline for the development of more effective systems of this type in the future.

2 Obtaining the Data

The two main sources of data used in this project are the arXiv and Wikipedia websites. To download the data from the arXiv without affecting the website's traffic, we used the bulk download service[1]. The LaTeX source of each article is compressed together in large tar files. Similarly, the Wikipedia data can be downloaded as a compressed multistream file[2].

In: C. Kaliszyk, E. Brady, J. Davenport, W.M. Farmer, A. Kohlhase, M. Kohlhase, D. Müller, K. Pąk, and C. Sacerdoti Coen (eds.): Joint Proceedings of the FMM and LML Workshops, Doctoral Program and Work in Progress at the Conference on Intelligent Computer Mathematics 2019 co-located with the 12th Conference on Intelligent Computer Mathematics (CICM 2019), Prague, Czech Republic, July 8–12, 2019, published at http://ceur-ws.org

[1] https://arxiv.org/help/bulk_data_s3

[2] https://dumps.wikimedia.org/enwiki/

'Bott-Danilov-Steenbrink vanishing theorem'	'Banach manifold'
'p-good cover'	'Loewy filtration'
'non-zerodivisor'	'graded-commutative product'
'Frobenius submanifold'	'local Harbourne constant'
'holomorphic one-dimensional foliations'	'donnés par'
'universal expansion'	'Berenstein-Zelevinsky triangles'
'non-toric purely log-terminal blow-up'	'DL-gallery'
'cubical algebra'	'smooth lifting'
'virtual bundle'	'stalkwise fibration'
'symplectic structure'	'4-dimensional quadric'

Table 1: Examples of the terms found on the math.AG articles from 2015

2.1 Processing the arXiv articles

The LaTeX source from the arXiv has to be further processed before it becomes useful. This is done with the LaTeXML software package [8]. LaTeXML converts the TeX source first to XML and optionally to HTML by using an additionally script. For the purpose of identifying the definitions labeled by the author, the XML output is enough.

```
<theorem class="ltx_theorem_definition" inlist="thm theorem:definition" xml:id="Thmdefinition1">
  <tags>
    <tag>Definition 1</tag>
    <tag role="refnum">1</tag>
    <tag role="typerefnum">Definition 1</tag>
  </tags>
  <title class="ltx_runin"><tag><text font="bold">Definition 1</text></tag>.</title>
  <para xml:id="Thmdefinition1.p1">
    <p class="ltx_emph"><text font="italic">Let <Math mode="inline"
                      tex="k" text="k" xml:id="Thmdefinition1.p1.m1">
        <XMath>
          <XMTok role="UNKNOWN">k</XMTok>
        </XMath>
```

Figure 1: Excerpt of the output produced by LaTeXML on a definition in an article

3 Classifying Definitions

Recent work indicates that well known text classification algorithms [1, 2] can have excellent accuracy at determining whether a given paragraph is in fact a definition. In [4] for example, a supervised learning method is first trained using word embeddings. These word embedding are created using the contents of the arXiv articles fed into a embedding algorithm like GloVe [11]. This has been implemented already and is available in [3]. Our system still does not use word embeddings for its classification, it is one of the main features we plan to add to the system in order to improve the classifier.

As training data for the classifier, we use the passages of certain articles that are labeled as definitions by the author by placing them in certain LaTeX macro environments. These macros are normally defined in the preamble of the document using the `\newtheorem` macro. LaTeXML resolves the user defined macros and labels the corresponding XML tag in the output file like in figure 1.

In order to produce the negative examples, we randomly sample paragraphs out of the article and assume they are not definitions. This introduces some imperfections in the training set, because some of the selected paragraphs necessarily contain some definitions.

We have performed successful experiments using common general purpose algorithms implemented in the scikit–learn Python library [10]. And these were confirmed with the results shown on the website https://corpora.mathweb.org/classify_paragraph. In table 2 we can observe the result of the classifier on some simple examples.

Text classifiers normally take each paragraph of an article and output an estimate of the probability of it being a definition or not. Figure 2 presents the basic performance metrics of the some of the classifiers implemented in the scikit-learn library. The Support Vector Classifier was observed to have the best performance and a more

Input to the Classifier	Result
a banach space is defined as a complete vector space.	**True**
This is not a definition honestly. even if it includes technical words like scheme and cohomology	**False**
There is no real reason as to why this classifier is so good.	**False**
a triangle is equilateral if and only if all its sides are the same length.	**True**

Table 2: Simple examples of the behaviour of the classifier

```
==============================
MultinomialNB ,  ngrams=(1,4)
****Results****
Accuracy: 86.6386%
Log Loss: 2.723683288081348
==============================
MultinomialNB
****Results****
Accuracy: 86.1733%
Log Loss: 1.8957941996159562
==============================
SVC ,  C= 2000
****Results****
Accuracy: 89.3283%
Log Loss: 0.29110830190582887
==============================
NuSVC
****Results****
Accuracy: 84.0215%
Log Loss: 0.34342025343628446
```

```
==============================
DecisionTreeClassifier
****Results****
Accuracy: 81.3609%
Log Loss: 6.437730639470123
==============================
RandomForestClassifier
****Results****
Accuracy: 83.6580%
Log Loss: 0.6044826387423514
==============================
AdaBoostClassifier
****Results****
Accuracy: 84.4868%
Log Loss: 0.6717126298133219
==============================
GradientBoostingClassifier
****Results****
Accuracy: 85.8098%
Log Loss: 0.3531810109520398
```

Figure 2: comparison of the most common classification algoriths on classifying definitions

detailed view of the result is pictured in table 3. In the future we plan to use the *fasttext* method [6] which has the best tradeoff between classification speed and accuracy.

	precision	recall	F_1-score	support
nondefs	0.73	0.91	0.81	2,217
definitions	0.95	0.84	0.89	4,661
micro avg	0.86	0.86	0.86	6,878
macro avg	0.84	0.87	0.85	6,878
weighted avg	0.88	0.86	0.87	6,878

Table 3: Overall performance of the SVC classifier on the test set

4 Extracting Definienda

After determining the definitions in the text, the system is required to find what is the term that is being defined in each definition. It is assumed that the *definiendum* is one or more adjacent words in the definition. This task can be interpreted as a Named Entity Recognition (NER) problem. Several different techniques have been developed to deal with it; as it is considered one of the most important subtasks of Information Extraction [9].

For the first approach to this problem, we used the ChunkParserI package from the NLTK library [7]. This module uses a supervised learning algorithm that is trained on examples of definitions tagged with part of speech (POS) and IOB. Each word in the definition is tagged with the token O for Outside, B–DFNDUM for the beginning of a definition and I–DFNDUM for the inside of definitions. Figure 3 specifies the order in which

these tags are allowed to appear. The POS is obtained using the pretrained model included in the NLTK library.

After the training is done, the model tries to predict the IOB tags. In table 4 an example of a successful identification of the definiendum is shown.

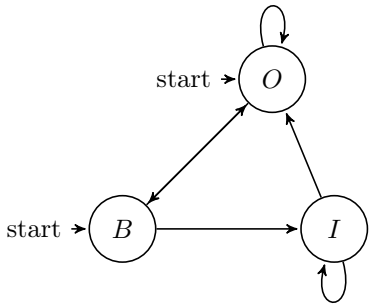

Figure 3: Possible states and switches for the IOB tags

text	We	define	a	Banach	space	as	a	complete	vector	space
POS	PRP	VBP	DT	NNP	NN	IN	DT	JJ	NN	NN
IOB	O	O	O	B–DFNDUM	I–DFNDUM	O	O	O	O	O

Table 4: Input example used in training the IOB parser.

To obtain the tagged text, the whole body of text from Wikipedia was used. The examples of definitions were obtained by filtering the articles with the two following properties:

- Articles that have a section with the word *definition*.
- The title of the article must appear at least once in this section.

These sections were assumed to be definitions and the title of the article which they belong to was assumed to be the definiendum. Only 5,229 articles were found matching this criteria (February 2019) out of the more than 6 million articles in the English Wikipedia. The dataset was split into training and test data, the results are shown in figure 5. When run on the definitions found on the algebraic geometry (math.AG) articles uploaded to arXiv on 2015, the results are pictured on figure 4.

Several difficulties were observed with this approach, for instance, many of the articles from Wikipedia are about topics completely unrelated to mathematics. Also, after stripping all the wiki markup from the text some

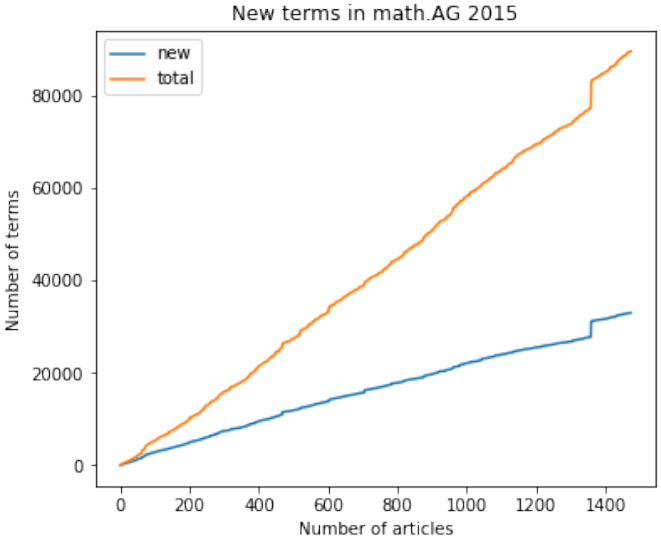

Figure 4: cumulative count of new and total term in the math.AG article of 2015.

IOB Accuracy:	91.1%
Precision:	31.5%
Recall:	67.6%
F-Measure:	43.0%

Figure 5: Metrics for the IOB task using the ChunkScore function in NLTK

of the remaining text made no sense, this means that there was a high chance of the POS tags to be defective and hence the IOB would also be tainted.

Input to the Classifier	Result
Let $n \geq 1$. Recall that the *lexicographic order* $\leq_l$ on $\mathbb{N}^n$ is defined by $v = (v_1, \ldots, v_n) \leq_l (w_1, \ldots, w_n) = w$ if and only if either $v = w$ or there is some i, $1 \leq i \leq n$, with $v_j = w_j$, for all j in the range $1 \leq j < i$, and $v_i < w_i$. Then $\leq_l$ is an *admissible* order on $\mathbb{N}^n$ in the sense of cite{BWK:98}. Indeed $\mathbb{N}^n$, together with componentwise addition and $\leq_l$, forms a totally ordered abelian monoid. The lexicographic order $\leq_l$ can be defined similarly on $\mathbb{Z}^n$, forming a totally ordered abelian group.	**True Positive**
(Upper semicontinuity of valuation) Let f be a nonzero element of $k[x_1, \ldots, x_n]$ and let $a \in k^n$. Then there exists a neighbourhood $V \subset k^n$ of a such that for all $b \in V$ $v_b(f) \leq_l v_a(f)$.	**False Positive**
This claim concerns valuation-invariant lifting in relation to $P_L(A)$: it asserts that the condition, 'each element of $P_L(A)$ is valuation-invariant in S', is sufficient for an A-valuation-invariant stack in $\mathbb{R}^n$ to exist over S.	**True Negative**
Let f/g be a nonzero element of K, let $U \subset k^n$ be an open set throughout which $g \neq 0$, and let $a \in U$. Then there exists a neighbourhood $V \subset U$ of a such that for all $b \in V$ $\mathrm{ord}_b(f/g) \leq \mathrm{ord}_a(f/g)$.	**False Positive**

Table 5: Examples of the performance of the classifier and NER system on article arXiv:1501.06563
The terms identified by the NER system have been underlined, the rest of the typesetting is taken verbatim from the original article.

5 Conclusions and Future Work

The main objective of this article is to showcase the feasability of a system that can search for definitions and important terms in large bodies of mathematical text. This system should have extremely good classification performance in order to avoid the errors propagating to the NER system. On the other hand, the system needs to be fast enough to tackle large corpora such as all the mathematical articles in the arXiv. Considering the performance of the state of the art methods for text classification and NER available today, and after observing the performance of the current prototype, we believe that this system is possible.

The next step in order to use more sophisticated methods is to use word embeddings or language models. Methods that utilize these achieve better performance in both of the classification and NER tasks.

In order to further better the performance of the NER subtask we also plan on increasing the amount of training data. The technique we used for the Wikipedia data can be adapted to other websites that host similar content like The Stacks Project (`https://stacks.math.columbia.edu/`) and the Groupprops subwiki (`https://groupprops.subwiki.org`). Additionally, applying *domain adaptation* methods might help to improve performance in case that the labeled data deviates significantly from nonlabeled data [5].

References

[1] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155, 2003.

[2] Tao Chen, Ruifeng Xu, Yulan He, and Xuan Wang. Improving sentiment analysis via sentence type classification using bilstm-crf and cnn. *Expert Systems with Applications*, 72:221–230, 2017.

[3] Deyan Ginev. arxmliv:08.2018 dataset, an html5 conversion of arxiv.org, 2018. SIGMathLing – Special Interest Group on Math Linguistics.

[4] Deyan Ginev. A web demo for scientific paragraph classification. `https://github.com/dginev/web-scipara-demo`, 2018.

[5] Jing Jiang and ChengXiang Zhai. Instance weighting for domain adaptation in nlp. In *Proceedings of the 45th annual meeting of the association of computational linguistics*, pages 264–271, 2007.

[6] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 427–431. Association for Computational Linguistics, 2017.

[7] Edward Loper and Steven Bird. NLTK: the natural language toolkit. *CoRR*, cs.CL/0205028, 2002.

[8] Bruce Miller. Latexml: A latex to xml converter. url: http://dlmf. nist. gov. *LaTeXML/(visited on 03/12/2013)*, 2013.

[9] David Nadeau and Satoshi Sekine. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26, 2007.

[10] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[11] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.