

Logical Structure Analysis of Scientific Publications in Mathematics *

Valery Solovyev

Nikita Zhiltsov

Kazan (Volga Region) Federal University
Kazan, Russia

maki.solovyev@mail.ru, nikita.zhiltsov@gmail.com

ABSTRACT

Even though the Linking Open Data cloud is constantly growing, there is a serious lack of published data sets related to the domain of academic mathematics. At the same time, since most scholarly publications in mathematics are well-structured and conventional, it's promising to get their helpful detailed representation. The paper describes an approach to extracting and analyzing the structure of mathematical papers. We present the Mocassin ontology that is used by analysis algorithms and can be considered as an ontology of the structure of scholarly publications in mathematics. The proposed semantic model has been evaluated on a set of real mathematical papers and preliminary evaluation results are encouraging. Also we discuss potential applications of the model to specific information retrieval tasks including semantic search.

Categories and Subject Descriptors

H.3.1 [Content Analysis and Indexing]: Indexing methods

General Terms

Algorithms

Keywords

document structure analysis, ontology, Linked Data

1. INTRODUCTION

The last several years have seen the rapid development of the Linked Data paradigm [3]. It is based on the idea of describing the contents of documents, e.g. web pages, in terms of “resources” or objects. As a rule, an ontology provides a controlled vocabulary that can be used to model a domain,

*This work was supported by the Russian Foundation for Basic Research (grant 09-07-12059).

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

WIMS'11, May 25-27, 2011 Sogndal, Norway

Copyright 2011 ACM 978-1-4503-0148-0/11/05 ...\$10.00.

that is, defining types of objects and valid relations between them. The object descriptions are typically stored as RDF data in a knowledge base. This data model has prominent advantages with regard to such challenging applications as data integration and semantic web search [1, 4, 22, 23].

The work presented here is a part of the long-term project Mocassin¹. Its primary goal is ad-hoc object retrieval in a collection of mathematical scholarly publications. In particular, one of the most urgent problems is summarizing of the main theoretical results for a given publication. In our point of view, this requires a specific representation of original documents, in the context of which a mathematical document consists of interconnected objects or *segments*. In our terms, a segment is a document part characterized by its starting and ending positions and having a certain functional role. Therefore, document structure analysis is an important task that should be resolved during the indexing phase of the search engine. More precisely, the proposed approach considers this task as a semantic analysis of not only document segments, but also relations between them. The semantics is described by means of our ontology.

1.1 Modeling the Structure of Scientific Publications

Every traditional research paper is well-organized. Some ideas have been elaborated to represent the structure of a typical paper. For example, in [8] the ABCDE format for proceedings and workshop contributions was proposed. As stated by the authors it can be easily mined by semantic browsers and characterizes by the following elements that have clear meanings: Annotation, Background, Contribution, Discussion and Entities. The next relevant semantic model is SALT (Semantically Annotated L^AT_EX) [10], which can be considered as a rich extension of the ABCDE format. This is an ontology-based approach to authoring research papers. SALT gives options to annotate a document part according to its rhetorical role. Also the SALT Document Ontology introduces the semantics of document segments that every research paper tends to include: Section, Paragraph, Figure and Table. Additionally, this ontology covers basic relations between document parts, such as *hasParagraph*, *hasFigure*, *hasTable* and so on.

1.2 Mathematical Knowledge Representation

Mathematical scientific articles are probably the most

¹MocaSSIN – Mathematical Semantic Search EngINe.
<http://code.google.com/p/mocassin/>

highly structured papers in the science domain. The other essential features of mathematical texts are the presence of mathematical notation elements and so-called “mathematical vernacular”, a peculiar subset of natural language that abounds with fixed phrases (e.g. see section “Phrases Used in Mathematical Texts” in [21]). By the way, over the past decade there have been significant advances in representing semiformal mathematical knowledge. The languages for formalized mathematics give opportunities to build semantically enriched models of a mathematical document and natively support logical structures on the statement level of a document. For example, the HELM ontology [2, 19] defines the concept *Theory* as a structured document, which contains instances of the concept *TheoryItem*. It’s possible to specify the kind of theory items using the property *itemType*. The examples of an object category are Axiom, Fact, Definition, Theorem, Lemma, Corollary and Variable. In addition, there are relations to state dependency between theory items: *uses*, *hasConsequence* and *hasPremise*. Another example is the Document Rhetorical aspect (DRa) ontology of the MathLang project [12], which characterizes document structure units according to their “mathematical rhetorical roles”, such as *assertion*, *conjecture*, *claim*, *example*, *proposition*, *theorem* etc. Also DRa ontology introduces relations between structural units, namely *exemplifies*, *inconsistentWith*, *justifies*, *subpartOf*, *relatesTo* and *uses*. Last but not least, both the statement level of the OMDoc format [14] and its ontology [15] express a class hierarchy of the statements and their properties. The *Statement* class is the root of OMDoc statement hierarchy. An OMDoc statement must be either constitutive or non-constitutive. The statements of the former category can be represented as instances of the classes *Definition* and *Axiom*; the statements of the latter type are instances of the *Example*, *Assertion* or *Proof* classes. At the same time, some types of assertions have a proof (*Theorem*, *Proposition*, *Corollary*, *Lemma*), the others don not need a proof (*Conjecture*, *Postulate*). This is expressed by the *ProvenAssertion*, *UnProvenAssertion*, *RefutedAssertion* subclasses of *Assertion*. The fundamental OMDoc relations are *formalizes* (related to different degrees of formalization), *hasPart* and *dependsOn*. Besides them, there are *proves*, *exemplifies*, *defines* and other relations. Finally, the forementioned representation formats are supported with helpful tools for authoring documents and integrating with theorem provers like Mizar, Coq and Isabelle.

However, the workflow of creating such highly formalized mathematical documents is still a laborious process. Moreover, it’s generally known that most scientific publications in mathematics are currently stored as PDF files, which have been compiled from the L^AT_EX sources. As an example, the Cornell University Library arXiv.org² is a freely available collection of electronic preprints of scientific articles in various fields including mathematics. As we shall see later, L^AT_EX is not only a convenient and powerful language for authoring mathematical papers, but even more appropriate than PDF in the sense of document structure analysis. For now, we give some observations. First, L^AT_EX enables us to avoid the problems (faced by the authors of [17]) related to Optical Character Recognition (OCR) and document segmentation by font style differences. Second, L^AT_EX’s cross-referencing mechanism, which is widely used by mathemati-

cians, provides a universal device for extracting relations between document parts. On the other hand, the great extensibility of L^AT_EX serves as a double-edge sword. We should tackle the problem of recognizing semantically identical segments that could be annotated differently. For example, the well-known AMS-L^AT_EX package includes fairly flexible commands to represent theorem-like environments, but there are no limitations to declare, say, a theorem using the following names: *theorem*, *thm*, *thms* or *maintheorem*. Furthermore, we have to deal with the synonyms as in case of *remarks*, *comments* and *notes*.

At last, the arXMLiv dataset [20] is a great resource for research too. It contains over 60% of arXiv.org papers that were automatically converted to XHTML + MathML from L^AT_EX. Available arXMLiv conversion tools make use of manually created LaTeXML bindings [16] that support various macro packages. The arXMLiv representation format is built upon XML and, therefore, makes various structural elements (e.g. sections, theorem-like environments, equations) tractable.

1.3 Semantic Analysis of Mathematical Documents

Our approach is closely related to the one of [7], which automatically builds a semantic model for a mathematical document edited with the standardized sT_EX macro package [13]. This model makes such documents directly available from the Web converting them to XHTML+RDFa format and offers different types of services like notation explanation, versioning and semantic search. But this approach is apparently impractical for handling large collections of the papers. Also we would like to note linguistic analysis modules of the arXMLiv project [9]. For example, a module for context based formula understanding aims to resolve ambiguities of mathematical notation. For that purpose, document text fragments, which surround a formula and contain certain patterns, are used to seek appropriate meanings of formula symbols. Another useful application is a module for applicable theorem search. This module analyzes idioms and fixed word sequences to extract hypothesis and conclusion parts of statements in the document. The mArachna project [11] focuses on extracting ontologies combining the mathematical knowledge and information about the document structure. However, the authors do not provide many details about the specification of the document structure, and implementation of the entity extraction techniques. Concerning the automatic extraction of semantic relations between document parts, to the best of our knowledge, our study is the first to examine it in the context of the mathematics domain.

The paper is organized as follows. In section 2 we present our ontology capturing the semantics of mathematical publication segments. In section 3 we describe the proposed ontology-based methods of logical structure analysis. Section 4 reports on the experimental results and an evaluation of the algorithms. We conclude in section 5 and discuss future work.

2. MOCASSIN ONTOLOGY

The motivation behind the development of our own lightweight ontology³ is to find a trade-off between expres-

²<http://arxiv.org>

³<http://c11.niimm.ksu.ru/ontologies/mocassin>

siveness of a semantic model and feasibility of the extraction algorithms. Specifically, we follow these considerations:

- **synthesis** – a careful analysis of state-of-the-art approaches to formalization of the structure of mathematical documents;
- **pragmatism** – exploiting the statistics of the most frequent annotations occurred in genuine scientific papers;
- **functionality** – defining unique and non-redundant role of an object and its properties;
- **flexibility** – support for easy integration with the existing relevant models.

The main goal of the Mocassin ontology is to capture the structural layout of a mathematical scholarly paper. The segment represents the finest level of granularity and has basic characteristics such as starting and ending positions, the text or math contents, and a functional role. Leveraging the L^AT_EX markup facilitates the identification of the former properties. But the latter one needs to be explored further. In order to discover a candidate set of ontology concepts, we studied the distribution of L^AT_EX commands in two collections of scholarly papers in mathematics. The first one is a sample of 1031 documents from [arXiv.org](http://arxiv.org). This is an example of a “wild” collection because of the independent submission process encouraged by this online library. The second collection contains 1355 papers of the “Izvestiya Vysshikh Uchebnykh Zavedenii. Matematika” journal⁴ from 1997 to 2009. The documents of this collection have been created under more strict and well-organized formatting and style requirements. So, we extracted the instances of L^AT_EX commands by considering them to be structural elements. The environments unrelated to the logical structure (e.g. *align*, *itemize*, *center*) were excluded. As a postprocessing step the semantically similar structural elements were merged into one concept. Finally, we obtained the following results. There are 103680 and 40805 instances in the arXiv.org sample and the “Izvestiya ...” corpus, respectively. The greatest part of those instances (69% and 52%, respectively) are equations represented, as usual, by *equation* or *equationgroup* environments. Sections and subsections are also ubiquitous elements in the papers of two collections (8% and 7%). We consider them as instances of the SALT’s *Section* concept. Ten remaining high-frequent elements form a relatively stable set of concepts (see Figure 1).

Among others, there are such elements as *question*, *problem*, *condition*, *property* and so on. They are much less frequent and were excluded from the concept set. By way of exception, the *Axiom* concept was added to the ontology as a constitutive element, even though it occurs fairly seldom. The *DocumentSegment* class is stated as a root concept of the ontology hierarchy.

The next step is defining a set of valid relations between segments. We analyzed three prior-art semantic models. Table 1 contains the definition of the relations (SALT’s concepts are emphasized) along with information about the presence of them in HELM, MathLang and OMDoc. Although it should be noted that the semantics of interpretations for a given relation is not always immutable across

all the models. Nevertheless, we have included this set of relations in our ontology after redefining them through the ontology concepts. In addition, the relations can be divided into two groups. The first one consists of *navigational* relations that aim to point to either an object of interest or an object required for reasoning. They are *refersTo* and *dependsOn*. The *refersTo* relation means that one segment contains a reference to another one. Most authors exploit this device to point at significant parts of a publication in the form of referential sentences. The *dependsOn* relation means that one segment uses another one as an essential preliminary fact, e.g. for defining a term or justifying a particular statement. If a statement contained by the second segment is untrue, the uncertainty of the statements contained by the first segment arises. For example, a false statement of a theorem leads to the falsity of a proof grounded in the theorem. It should be stressed that *dependsOn* is transitive. It means that one can derive semantic dependencies between segments from the transitive closure of this relation. The second group contains relations with quite restricted domains. Among them, there are *hasConsequence* (the relation between a segment and its corollary), *exemplifies* (the relation between an example and a segment) and *proves* (the relation between a proof and a statement that the proof justifies). The *hasPart* relation is a part-whole relation between segments.

Eventually, in order to integrate the Mocassin Ontology with the SALT Document Ontology, we could declare *hasSegment* relation using SALT’s *Publication* class and Mocassin’s *DocumentSegment* class as its domain and range, respectively. Another dependency relationship between these ontologies is a declaration of Mocassin’s *refersTo* property, that is, SALT’s *Section* class is permitted to be both a domain and a range of the property.

Expressiveness of the proposed ontology can be enhanced with additional rules that might be useful for applications. For example, the rule

$$dependsOn(p, t) \leftarrow hasPart(t, e), dependsOn(p, e)$$

states that a segment p (say, a proof) depends on t (e.g. theorem) if t contains a segment e (e.g. equation) and p depends on e . Inference engines might use such rules during the reasoning process to discover some helpful properties of involved objects.

Thus, one can build semantic representation for a given mathematical paper using concepts and relations of the ontology. Since each ontology may be seen as a digraph, where the concepts are nodes and relations are edges, the constructed model can be serialized into various RDF formats in the form of subject-predicate-object triples. For the purpose of illustration, we provide an example of annotating the fragment of a document with the ontology elements using the RDF/N3 syntax:

```
<example.tex#Theorem1> a <m:Theorem> .
<example.tex#Proof1> a <m:Proof> .
<example.tex#Section1> <m:hasPart>
<example.tex#Theorem1> .
<example.tex#Section1> <m:hasPart>
<example.tex#Proof1> .
<example.tex#Proof1> <m:proves>
<example.tex#Theorem1> .
```

In practice, the RDF metadata along with source files

⁴http://www.ksu.ru/eng/el_journ/izvuz/

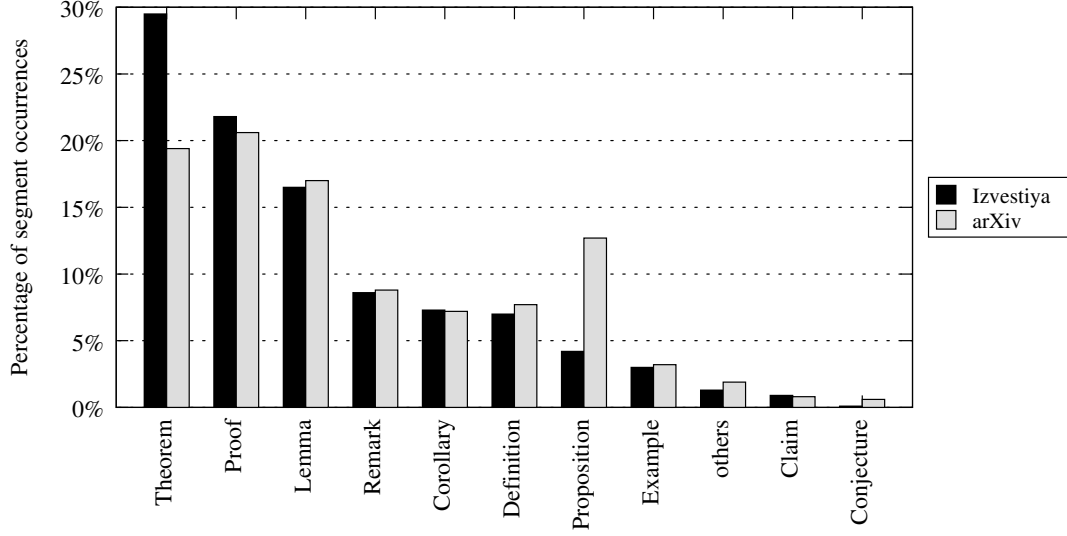


Figure 1: Occurencies of the structural elements in the test collections

Table 1: The definitions of the ontology relations and their presence in the different semantic models

Relation	Domain	Range	HELM	ML	OMDoc
hasPart	DocumentSegment, <i>Section</i>	DocumentSegment, <i>Section</i> , <i>Figure</i> , <i>Table</i>	+	+	+
refersTo	<i>Section</i> , Axiom, Claim, Conjecture, Corollary, Definition, Example, Lemma, Proof, Proposition, Remark, Theorem	<i>Section</i> , <i>Figure</i> , <i>Table</i> , Axiom, Claim, Conjecture, Corollary, Definition, Equation, Example, Lemma, Proof, Proposition, Remark, Theorem	+	+	+
dependsOn	Claim, Conjecture, Corollary, Definition, Example, Lemma, Proof, Proposition, Theorem	Axiom, Claim, Conjecture, Corollary, Definition, Equation, Lemma, Proposition, Theorem	+	+	+
exemplifies	Example	Axiom, Claim, Conjecture, Corollary, Definition, Lemma, Proposition, Theorem	-	+	+
proves	Proof	Claim, Corollary, Lemma, Proposition, Theorem	-	+	+
hasConsequence	Axiom, Claim, Conjecture, Corollary, Lemma, Proposition, Theorem	Corollary	+	-	-

Input: $S_d, \mathbb{C}$, threshold
Output: $O = \{\langle s, c_s \rangle\}$

```

1: for  $s \in S_d$  do
2:   string  $\leftarrow$  ExtractSegmentTitle( $s$ )
3:   if string = NIL then
4:     string  $\leftarrow$  GetEnvironmentName( $s$ )
5:   end if
6:   for  $c \in \mathbb{C}$  do
7:     name  $\leftarrow$  GetClassName( $c$ )
8:      $\langle c, v \rangle_i \leftarrow$  ComputeStringSimilarity(string, name)
9:   end for
10:   $\langle c, v \rangle_{max} \leftarrow$  GetMaxValue( $\langle c, v \rangle$ )
11:  if  $v_{max} \geq$  threshold then
12:     $c_s \leftarrow c_{max}$ 
13:  else
14:     $c_s \leftarrow \emptyset$ 
15:  end if
16:   $O \leftarrow O \cup \langle s, c_s \rangle$ 
17: end for

```

Figure 2: Algorithm for classifying document segments

($\text{\LaTeX}$, PDF) can be stored in a knowledge base or a file system for each mathematical paper in a collection.

3. LOGICAL STRUCTURE ANALYSIS

The ontology presented in the previous section specifies a controlled vocabulary to semantic analysis of document parts in a mathematical scholarly paper. Therefore, logical structure analysis inherently falls into two tasks: (i) recognizing the types of structural elements and (ii) recognizing the semantic relations between them.

3.1 Recognizing the types of segments

TASK 1 (DOCUMENT SEGMENT CLASSIFICATION).

Given a fixed set of Mocassin ontology classes $\mathbb{C}$. For each segment $s \in S_d$ of a document d , find a classification function γ that maps segments to classes:

$$\gamma : S \rightarrow \mathbb{C} \cup \emptyset,$$

where $\emptyset$ denotes a segment type undefined by the ontology.

The set $\mathbb{C}$ can be extended with classes from the aforementioned SALT Document Ontology. Introducing the null type $\emptyset$ is important because our ontology does not cover all possible types of segments. The result of Task 1 is a mapping between segments and ontology concepts. To solve it, we propose the following algorithm (see Figure 2).

The input of the algorithm is a set S_d of segments of a given document d , the set of Mocassin ontology classes $\mathbb{C}$, and the predefined real-valued threshold, which ranges from 0 to 1. In lines 2 and 4 of the algorithm we exploit the $\text{\LaTeX}$ markup. The segment titles are extracted from the declarations, which look like “ $\backslash newtheorem \{thm\} \{Theorem\}$ ”, in the header of a document. If these elements are unavailable, the environment names are used. For example, one could encounter the following declared environment names of proofs: “ $\backslash begin\{proof\}$ ”, “ $\backslash begin\{pf\}$ ”, “ $\backslash begin\{prf\}$ ”. Then, we seek the most similar ontology class according to a string similarity measure between its name and the string corresponding to a given segment. There are numerous metrics computing

Input: $S_d, \{\langle s, c_s \rangle\}$
Output: $O = \{\langle s, s_o, d_1, d_2, \text{verbs} \rangle\}$

```

1: for  $s \in S_d$  do
2:    $b_s \leftarrow$  GetSegmentBeginning( $s$ )
3:    $e_s \leftarrow$  GetSegmentEnd( $s$ )
4:    $F_s = (f_1, \dots, f_k) \leftarrow$  ExtractReferentialSentences( $s$ )
5:   for  $f \in F_s$  do
6:      $m \leftarrow$  FindMentionedSegment( $f, S_d$ )
7:      $b_m \leftarrow$  GetSegmentBeginning( $s_m$ )
8:      $e_m \leftarrow$  GetSegmentEnd( $s_m$ )
9:      $d_1 \leftarrow b_s - b_m$ 
10:     $d_2 \leftarrow e_s - e_m$ 
11:    verbs  $\leftarrow$  ExtractVerbTokens( $f$ )
12:     $O \leftarrow O \cup \langle s, m, d_1, d_2, \text{verbs} \rangle$ 
13:  end for
14: end for

```

Figure 3: Algorithm for extracting features of navigational relations between segments

similarity or dissimilarity score between two text strings (for example, see [5]), the best known of which are edit distance based (e.g. Levenshtein distance) algorithms and q-gram index based algorithms. *GetClassName* function takes into account synonyms of the concept names, e.g. both “*conjecture*” and “*hypothesis*” merge into one concept. After all, the predefined threshold is used for checking whether a found solution is appropriate. We assume that a preprocessing step that makes the extraction of $\text{\LaTeX}$ markup elements trivial has been done. Then, the time complexity of the algorithm is $\Theta(|S_d| \times |\mathbb{C}| \times L)$, where L is the time complexity of the selected string similarity algorithm, which generally depends on both string lengths.

3.2 Recognizing the semantic relations between segments

TASK 2 (RELATION CLASSIFICATION). Given a fixed set of Mocassin ontology relations $\mathbb{R}$; a mapping $\{\langle s, c_s \rangle\}$ between segments of a document d and Mocassin ontology classes $\mathbb{C}$, find a function δ that maps pairs of segments to relations:

$$\delta : S \times S \rightarrow \mathbb{R}.$$

Before we introduce the relation extraction algorithm, it’s necessary to divide the considered relations into two groups: navigational and restricted relations. Then, such a separation leads to the following insights about relation recognition. We assume that navigational relations are expressed by means of explicit references between segments. In other words, our algorithm will tend to detect relations of this kind in the context of given referential sentences. For example, there is a sentence “*By applying Lemma 1, we obtain ...*” placed in a proof of a statement. This fact yields a relation *dependsOn* between the proof and Lemma 1. Yet another example is the sentence “*Theorem 2 provides an explicit algorithm ...*” contained in a segment of a document. This statement produces the relation *refersTo* between the document part and the theorem. That’s why, we have the following algorithm for extracting features of navigational relations (see Figure 3).

The input of the algorithm is a set S_d of segments of a given document d and a mapping between segments and on-

tology classes that was acquired earlier. For each document segment in S_d we obtain its starting and ending positions normalized by the size of the document (*GetSegmentBeginning* and *GetSegmentEnd* functions). In line 4 we extract referential sentences contained by a given segment. *ExtractReferentialSentences* function splits the text contents of the segment into sentences and seeks occurrences of the L^AT_EX command *ref*, e.g. “\ref {thm3.1}” or “\ref {lem2.2}”. To simplify explanation, we assume that each referential sentence contains the only occurrence of *ref*. Then, for each referential sentence we find the segment, which is mentioned in it. *FindMentionedSegment* function looks for the segment having corresponding L^AT_EX label. It finds either the label of the segment itself or the one of the environment (e.g. *itemize*) contained by the desired segment. In the latter case, this function associates the label with the least enclosing segment. In lines 7 and 8 we obtain the starting and ending positions of the found segment. In lines 9 and 10 we compute the relative position of the initial segment, which contains a referential sentence, with respect to the found one (“distance” feature). This is calculated as corresponding subtractions of their starting and ending positions. Both the values d_1 and d_2 range from 0 to 1. This feature takes into account not only the order of two segments, but also the distance between them. In particular, d_1 and d_2 are expected to be negative and near zero values in the context of *dependsOn* relation. In line 11 we extract verbs from a given referential sentence. Verbs appear to be the most discriminative parts of referential sentences (“verb” feature). For example, *see*, *describe*, *show*, *notice*, *discuss* are characteristic of *refersTo* relation. On the contrary, *get*, *obtain*, *use*, *recall*, *apply*, *derive* are more likely to occur in the context of *dependsOn* relation. *ExtractVerbTokens* function initially performs tokenizing, stemming and part-of-speech (POS) tagging a sentence. It means that each word in a sentence maps to its lexeme marked up with appropriate POS tag. After that, the function filters out all non-verb tokens using “VB*” pattern for POS tags to determine verb tokens. The “verb” feature can be encoded as a boolean vector or TF-IDF weighted vector after processing all the referential sentences in a document corpus.

Thus, the output of the algorithm gives three types of features for each relation instance. We have studied the significance and discriminative power of these features for classifying extracted relations. For that purpose we train a learning model using a given data set of labeled instances. Among various supervised training models, we have chosen the decision tree model due to its easily interpretable output. We report on the evaluation results in the next section.

In practice, the algorithm should be better executed after preprocessing steps building the modified full text index, which only contains verb tokens, for each referential sentence of a segment. It would enable to reduce the time complexity of the algorithm at least to $\Theta(|S_d|^2 \times |F_{ave}|)$, where $|F_{ave}|$ is the average number of referential sentences per segment.

On the other hand, when we discover instances of the restricted relations, it’s sufficient to attach a complementary segment for a given segment, the type of which is one of *Corollary*, *Example* or *Proof* classes. We therefore present a baseline algorithm for classifying restricted relations (Figure 4).

The algorithm makes use of the ontology, the mapping between segments and ontology concepts, and information

Input: $S_d, \{\langle s, c_s \rangle\}, \mathbb{C}, r \in \mathbb{R}$

Output: $O = \{\langle s, p \rangle\}$

```

1: domain  $\leftarrow$  GetValidDomain( $r$ )
2: ranges  $\leftarrow$  GetValidRanges( $r$ )
3: for  $s \in S_d$  do
4:    $c_s \leftarrow$  GetSegmentClass( $s, \mathbb{C}$ )
5:   if  $c_s = \text{domain}$  then
6:      $p \leftarrow$  GetClosestPredecessor( $s, \text{ranges}$ )
7:      $O \leftarrow O \cup \langle s, p \rangle$ 
8:   end if
9: end for
```

Figure 4: Algorithm for classifying restricted relations between segments

about relative positions. We assume that each restricted relation r has the only domain and several ranges. According to Table 1, this is clearly true for *proves* and *exemplifies* relations, and one can swap the domain and range of *has-Consequence* relation to obtain the similar declaration. In line 1 *GetValidDomain* function queries the Mocassin ontology to return the ontology concept being the domain of a given relation r . For example, it returns *Proof* concept for given *proves* relation. In line 2 *GetValidRanges* function does almost the same to retrieve all the valid ranges for a given relation r . Then, we bind each appropriate segment s in a document d to its corresponding segment. Specifically, in line 4 *GetSegmentClass* function determines the type of a segment s using the given mapping. If the type of a segment s does not fit to a given domain of a relation r , the segment will be filtered out. Otherwise *GetClosestPredecessor* function seeks the closest precedent segment, the type of which is among the valid ranges.

If the position index of segments is built and number of ontology classes $|\mathbb{C}|$ is the known small integer, the time complexity of the algorithm is $\Theta(|S_d|^2)$.

Thus, the algorithm seeks a valid segment related to a given instance of a corollary, a proof or an example. It should be noted that leveraging referential sentences might improve the accuracy of the algorithm. Certain ideas for improving the baseline algorithm will be discussed in the next section.

4. EXPERIMENTS AND EVALUATION

We implemented the proposed algorithms as a Java library⁵, based on L^AT_EX-to-XML converters of the arXMLiv project [20], TeXlipse plugin API⁶, and GATE framework [6] for annotation and document management.

At first, we provide the statistics from both the collections. These results highlight the rationale behind using referential sentences as an important factor of the approach. Here is the first question: what is the share of documents marked up appropriately using L^AT_EX’s cross-referencing mechanism? It turns out that only 24.0% of documents in the “Izvestiya” corpus contain occurrences of *ref* command, while the analogous share was 89.1% of documents in the arXiv.org sample. The second question is how many referential sentences does a document contain on the average? We consider documents with at least one referential sentence. In this case, the results are quite similar for both the collections: there

⁵<http://code.google.com/p/mocassin/downloads>

⁶<http://texlipse.sourceforge.net/>

Table 2: Evaluation of document segment classification

Class	# of true instances	F_1 -score
Axiom	5	100.0
Claim	114	98.7
Conjecture	152	98.7
Corollary	1715	99.5
Definition	1838	100.0
Equation	71314	100.0
Example	771	99.9
Lemma	4061	99.8
Proof	4943	99.7
Proposition	3052	99.9
Remark	2114	100.0
Theorem	4670	99.1
<i>other</i>	671	89.2

is the mean value $\mu = 53.9$ sentences (standard variance $\sigma = 48.7$) for the “Izvestiya” corpus, and there is the mean value $\mu = 51.4$ sentences (standard variance $\sigma = 52.6$) for the arXiv.org sample. Thus, taking into account the average numbers of segments per document for both the collections, one could reasonably expect that the generated digraphs of the logical structure are at least weakly connected.

Hence, we proceed with evaluation results. In order to evaluate our approach, we needed a tool to reduce the efforts in assessment of our tasks. The GATE Developer application has been used for that purpose, due to its opportunity to visualize the XML output of the arXMLiv converters. That’s why the evaluation of the proposed algorithms has been carried out using only the arXiv.org sample.

For implementing the algorithm of document segment classification, the q-gram string matching algorithm from the SimMetrics library⁷ was used as a string similarity measure. Its value ranges from 0 to 1. The threshold value optimized with regard to F_1 -score was estimated to be equal to 0.501. During the evaluation it turns out that almost all the documents have the proper header declared. In general, this means that we have left without evaluation rather difficult cases, when the algorithm has only to deal with environment names. On the other hand, if the arXiv.org sample is indeed unbiased in that sense, we can imply that the method is acceptable for applying on the entire arXiv.org collection. The method gave very accurate results (Table 2). Although, it encountered difficulties when compounds are used in element titles, e.g. “Smirnov’s Theorem” or “Thurston’s Virtual Fibration Conjecture”. By the way, a warning should be provided here, because we evaluated only a segment recognition task: we did not actually annotate by hand all the potential instances, e.g. particular pieces of text, with analogous functional roles. In the conclusion, we will discuss probable misses of the algorithm that the evaluation results do not reflect.

To evaluate our supervised approach to recognizing navigational relations, we randomly selected 243 referential sentences from the document collection and manually annotated them with the following labels: *dependsOn*, *refersTo*

Table 3: Evaluation of classifying navigational relations

Features	Acc.	F_1 -score refersTo	F_1 -score dependsOn
type	0.663	0.566	0.752
type+distance	0.658	0.663	0.704
type+verb _{bool}	0.704	0.653	0.770
type+verb _{tfidf}	0.638	0.576	0.705
type+distance+	0.716	0.706	0.762
verb _{tfidf}			
type+distance+	0.741	0.744	0.772
verb _{bool}			

Table 4: Evaluation of classifying relations

Relation	# of instances	F_1 -score
refersTo	243	0.744
dependsOn	243	0.772
hasConsequence	178	0.687
exemplifies	62	0.613
proves	216	0.954

and *other*. After that, a decision tree learner implemented in the RapidMiner application⁸ (based on C4.5 [18]) was trained with features extracted by the algorithm. The dimension of vectors for verb tokens turned out to be equal to 121 in the experiment. Totally we had 2 nominal values of structural element types and 123 numeric values associated with each sentence. Table 3 shows the performance results of 10-fold cross validation for all types of features and both the boolean and TF-IDF weighted vectors for the verb characteristic.

All the values were observed after optimization of confidence and minimal leaf size parameters of the tree learner. Further, the experiment established that about 95% (127 *dependsOn* and 104 *refersTo* relation instances) of the labeled sentences did, in fact, produce true navigational relations between segments. The quality of the feature-based algorithm is demonstrated by its fairly promising results. It showcases that even if we have a very limited training data set, it’s reasonably expected almost 3/4 of unlabeled examples to be handled correctly. However, acquiring new labeled instances is usually quite laborious. For practical reasons some rules can be derived from the output of the decision tree learner. For example, let the first segment is denoted by “from”, and “to” is the second segment, respectively. Then, one can use the rule *to*=‘equation’ & *from*=‘proof’ $\rightarrow$ *dependsOn* to detect *dependsOn* relations with extremely high accuracy. For the rule *to*=‘theorem’ & *from*=‘section’ & ‘imply’ $\notin$ *verbs* $\rightarrow$ *refersTo*, the estimated probability of a given relation with such characteristics to be *refersTo* is nearly 0.74 on our test set.

To evaluate our baseline algorithm for classifying the restricted relations, we randomly selected at most 10% of the documents, which contain corresponding segments (a corollary, an example or a proof). Then, we annotate the re-

⁷<http://sourceforge.net/projects/simmetrics/>

⁸<http://rapidminer.com/>

lations for each such a segment in a selected document, and execute the algorithm. The results are shown in Table 4. The algorithm fails to process those corollaries, which were just the imports from other papers (26 out of 178 instances). By the way, it shows that handling such cases should be incorporated into an extension of the algorithm. This can be done by searching for citations to other papers located in the title of a corollary. Further, the algorithm fails to retrieve relations of particular examples that exemplify some text fragments in a section (11 out of 62 cases). Our ontology does not define the semantics of such relations. On the other hand, leveraging references can improve the performance of the algorithm. This gives opportunities to find elements besides precedent ones involved in relations. The results for retrieving instances of *proves* relation are very encouraging. Furthermore, they might be better if the algorithm extracts references from the titles of proofs, which look like “Proof of Theorem 1”. Thus, we imply that the baseline algorithm fairly succeeds in case of the *proves* relation, and requires improvement for both remaining relations.

5. DISCUSSION AND FUTURE WORK

The proposed ontology-based approach to structure analysis of mathematical scholarly publications aims to semantically annotate them in an automatic way. During the development, we took into account various state-of-the-art models to formalize a structure of a mathematical document. The analysis algorithms extensively use essential characteristics of original texts. The reported evaluation results enable us to consider potential applications of the model. For instance, a provided semantic graph can be used for discovering significant document parts such as a main theorem or a proof of an important statement. Another helpful application is similar document search. Additionally, we should note a few caveats regarding the proposed approach. It still does not cover rather difficult cases of ambiguous semantic roles of some segments, e.g. when a subsection actually contains a text of a proof. Furthermore, relying on the explicit markup of a segment, it ignores particular representation examples of statements tagged with only initial commands. Anyway, an enormous portion of the appropriately marked up document segments is successfully recognized. The next minor issue is a common redundancy of some instances of navigational relations. This may cause an overhead in processing some relation instances that could be merged into one without loss of information.

Finally, we consider applying the demonstrated techniques to analysis of relations between document segments via citation links within the whole collection of mathematical publications. In our point of view, this can provide opportunities to build an intriguing logical dependency graph for particular mathematical knowledge objects that are represented as statements in scholarly publications. We are also going to extend our approach to other domains such as computer science and physics.

6. ACKNOWLEDGMENTS

The authors gratefully thank Alexandr Elizarov and members of the KWARC research group, especially Deyan Ginev and Christoph Lange, for providing access to the document collections. The authors also sincerely thank the reviewers for their valuable comments and suggestions.

7. REFERENCES

- [1] Yahoo! Search BOSS. At <http://developer.yahoo.com/search/boss/>. Accessed on January 18, 2011.
- [2] A. Asperti, L. Padovani, C. S. Coen, F. Guidi, and I. Schena. Mathematical knowledge management in HELM. *Ann. Math. Artif. Intell.*, 38(1-3):27–46, 2003.
- [3] T. Berners-Lee. Linked Data – Design Issues. At <http://www.w3.org/DesignIssues/LinkedData.html/>. Accessed on January 18, 2011.
- [4] G. Cheng, W. Ge, H. Wu, and Y. Qu. Searching semantic web objects based on class hierarchies. In *proceeding of LDOW Conference*, available at: <http://iws.seu.edu.cn/publications/cgwq08>. Citeseer, 2008.
- [5] W. Cohen, P. Ravikumar, and S. Fienberg. A comparison of string metrics for matching names and records. In *KDD Workshop on Data Cleaning and Object Consolidation*, volume 3, pages 73–78, 2003.
- [6] H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan. GATE: A framework and graphical development environment for robust NLP tools and applications. In *Proceedings of the 40th Annual Meeting of the ACL*, 2002.
- [7] C. David, M. Kohlhase, C. Lange, F. Rabe, N. Zhiltsov, and V. Zholudev. Publishing math lecture notes as linked data. In *ESWC (2)*, pages 370–375, 2010.
- [8] A. de Waard and G. Tel. The abcde format enabling semantic conference proceedings. In M. Völkel and S. Schaffert, editors, *Proceedings of the First Workshop on Semantic Wikis – From Wiki To Semantics*, Workshop on Semantic Wikis. ESWC2006, 2006.
- [9] D. Ginev, C. Jucovschi, S. Anca, M. Grigore, C. David, and M. Kohlhase. An architecture for linguistic and semantic analysis on the arXMLiv corpus. In *Applications of Semantic Technologies (AST) Workshop at Informatik*, 2009.
- [10] T. Groza, S. Handschuh, K. Möller, and S. Decker. Salt - semantically annotated latex for scientific publications. In *ESWC*, pages 518–532, 2007.
- [11] S. Jeschke, N. Natho, S. Rittau, and M. Wilke. mArachna Automatically Extracting Ontologies from Mathematical Natural Language Texts. In *IMECS*, pages 958–963, 2007.
- [12] F. Kamareddine and J. B. Wells. Computerizing mathematical text with MathLang. *Electr. Notes Theor. Comput. Sci.*, 205:5–30, 2008.
- [13] M. Kohlhase. sTeX: Semantic Markup in T_EX/L^AT_EX. *Mathematical User Interfaces*, 2004.
- [14] M. Kohlhase. *OMDoc - An Open Markup Format for Mathematical Documents [Version 1.2]*. Springer Verlag, 2006.
- [15] C. Lange. *Integrated Semantic Web Collaboration on Semiformal Mathematical Knowledge*. PhD thesis, 2010.
- [16] B. Miller. LaTeXML: A LaTeX to XML converter. At <http://dlmf.nist.gov/LaTeXML>. Accessed on January 18, 2011.
- [17] K. Nakagawa, A. Nomura, and M. Suzuki. Extraction of logical structure from articles in mathematics. In *MKM*, pages 276–289, 2004.

- [18] J. Quinlan. *C4.5: programs for machine learning*. Morgan Kaufmann, 1993.
- [19] I. Schena. *Towards a Semantic Web for Formal Mathematics*. PhD thesis, University of Bologna, Mar. 2002. Technical Report UBLCS-2002-6.
- [20] H. Stamerjohanns, M. Kohlhase, D. Ginev, C. David, and B. Miller. Transforming Large Collections of Scientific Publications to XML. *Mathematics in Computer Science*, 3(3):299–307, May 2010.
- [21] J. Trzeciak. *Writing Mathematical Papers in English – a practical guide*. European Mathematical Soc., 1995.
- [22] G. Tummarello, R. Cyganiak, M. Catasta, S. Danielczyk, R. Delbru, and S. Decker. Sig. ma: live views on the web of data. *Web Semantics: Science, Services and Agents on the World Wide Web*, 2010.
- [23] G. Tummarello, R. Delbru, and E. Oren. Sindice. com: Weaving the open linked data. In *Proceedings of the 6th international The semantic web and 2nd Asian conference on Asian semantic web conference*, pages 552–565. Springer-Verlag, 2007.