2018 12th IEEE International Conference on Semantic Computing

Extracting Semantic Relations for Scholarly Knowledge Base Construction

Rabah A. Al-Zaidy
College of Information Science and Technology
Pennsylvania State University
University Park, Pennsylvania, 16802
Email: alzaidy@psu.edu

C. Lee Giles
College of Information Science and Technology
Pennsylvania State University
University Park, Pennsylvania, 16802
Email: giles@ist.psu.edu

***Abstract*—The problem of information extraction from scientific articles, found as PDF documents in large digital repositories, is gaining more attention as the amount of research findings continues to grow. We propose a system to extract semantic relations among entities in scholarly articles by making use of external syntactic patterns and an iterative learner. While information extraction from scholarly documents have been studied before, it has been focused mainly on the abstract and keywords. Our method extracts semantic entities as concepts and instances along with their attributes from the fully body text of documents. We extract two types of relationships between concepts in the text using an iterative learning algorithm. External data sources from the web such as the Microsoft concept graph, as well as query logs, are utilized to evaluate the quality of the extracted concepts and relations. The concepts are used to construct a scientific taxonomy covering the research content of the documents. To evaluate the system we apply our approach on a set of 10k scholarly documents and conduct several evaluations to show the effectiveness of the proposed methods. We show that our system obtains a 23% improvement in precision over existing web IE tools when they are applied to scholarly documents.**

I. Introduction

Scientific articles in digital libraries are constantly increasing in volume, making it more challenging for researcher to easily access their content by traditional search methods. This, along with other factors, motivated many studies that aim at mining scholarly documents and providing them with semantic structure. To enable automated understanding of scientific articles, studies showed that relying solely on the abstract of the articles, which is typically available as meta data, is not enough to infer sufficient findings and relevant information to the article's content [1]. By including the full text of the article, information gain increases and higher quality semantics can be obtained.

One approach for semantically structuring scholarly documents is to represent them as knowledge graphs [2]. By extracting semantics showing key components of the studies such as topic, problem or task, approach and methods, findings, building inferences on this graph allow for more sophisticated analyses of the content [3]. Academic search engines, for instance, can directly benefit from this graph representation by using it to leverage the ranking algorithm or for query parsing and question answering. By representing concepts as entities in the graph and their relationships as edges, graph algorithms used for web search and text mining can be applicable to research content.

Extracting a knowledge graph or constructing taxonomies requires the extraction of entities and their relations, i.e. a form of triples. Various academic attempts at constructing knowledge bases from text corpora have used two basic approaches. Extracting entities and then inferring the relationships, or inferring related entities. The former assumes a specific type of entity and typically applies NER methods to extract them, and follows that by a relation extractor designed for the specific entities. The latter is used when the type of entity is difficult to determine and is common in most Open IE methods. For open IE extractors that rely on iterative learners, the learner builds on an initial seed of annotated samples of several types of relationships. Other approaches rely on bootstrapping using syntactic patterns for extracting initial seed samples from the text and use those for the learner.

Knowledge base and taxonomy construction requires specifying a type of relationship that is extracted from the data. For openIE methods, the relationship types is typically not specified by approaches such as [4]. However, for knowledge base construction the hyponym relation, that represents a concept and an instance or a concept and sub concept, is used to build the taxonomy. A variety of syntactic patterns and dependency path learners extract specific types of relationships based on the application at hand.

A main challenge with scholarly documents is the sparsity of terms in the document. This is a result of the wide variety in topics where each article focuses on a narrow topic that in many cases contains newly introduced terms. Additionally, among different disciplines the same concept may have different names. Web query logs are considered as an external source of data that is used for knowledge base construction from web documents. To our knowledge, there is not yet a study that makes use of academic search engine query logs to learn new topics. In this

0-7695-6360-0/18/$31.00 ©2018 IEEE
DOI 10.1109/ICSC.2018.00017

paper we design a system to construct a knowledge base from scholarly documents. The knowledge base is a set of concepts found in the documents and the edges define two types of relationships among the concepts: hyponymy and attributes. Entities and relations are extracted using an iterative learner that bootstraps on an automatically generated set of seeds. We propose the use of external sources to evaluate our resulting relationships. The sources are query logs and the MS concept graph. Extracted triples are used to construct a taxonomy by performing sense disambiguation for each extracted concept. We apply the methods to a set of 10k scholarly articles and evaluate our system against current available baselines. We also evaluate the newly defined relation types using expert evaluations of the data. The paper is organized as follows. In the next section we discuss related work. In section 3 we discuss the syntactic extractor that is used for the bootstrapping of the learner. In section 4 we describe the iterative learner method and the use of multiple external sources. Section 5 contains the algorithms for constructing the taxonomy. The evaluation and experimentation results are presented in section 6. In section 7 we conclude and discuss future directions and applications.

II. Related Work

Many information extraction approaches for harvesting entities and their relationships from textual data exist in the literature. These methods aim to provide semantic structure to free text in the web. Several of these resulted in the commonly known knowledge bases such as Google Knowledge Graph [5] , NELL [6] , YAGO [7] and DBpedia [8]. These KBs were constructed from large corpora such as Wikipedia pages or at web-scale. Some approaches make use of IE techniques, such as the afore mentioned ones, while others are constructed using manual curation, such as the Cyc project [9] that has a limited concept space coverage of only 120,000 concepts. Methods that build on a curated method, can use collaborative crowd-sourcing to increase coverage, an example of such a system is the Freebase KB [10] and Wikidata [11]. These have increased coverage results however, they do not contain fine-grained scientific concepts. Automated approaches include the NELL [6] , KnowItAll [12], ReVerb [13]. They contain many concepts that may exist in scientific texts but are far from covering many concepts in research articles. Common-sense KB approaches aim to find relationships that allow for common-sense inference over text data. A main approach for this type is Probase [14]. Probase is constructed from a corpus of 1.68 billion web pages and harvested over 2.7M concepts.

A hyponymy relationship between two entities occurs when one of the entities is an instance of the other, also referred to as the *is-a* relationship. The super-concept is called *hypernym* whereas the instance, or sub-concept is called the *hyponym*. One of the initial works is [15] that introduced the Hearst patterns used by most other approaches for bootstrapping. Latent semantic analysis is used to improve the results obtained by using the Hearst patterns by using latent semantic indexing for the hypernym-hyponym terms pair [16]. It then compares the similarity score between the two to infer if one of the terms is actually an instance of the other. In [17] they use a training set of known hypernym-hyponym pairs to train a model using the dependency path features. The model then learns new dependency paths and uses them to extract new *is-a* relation pairs. Syntactic patterns have been the basis for several approaches for extracting concept attributes [18], [19]

Semantic learners based on iterative algorithms such as TextRunner [20], NELL [6], and Probase [14] use a bootstrapping approach. The first two iteratively learn new syntactic patterns to improve the extraction for the next iterations. Our approach is similar to the one in [14] in that our iterations learn more entities and relations by using knowledge extracted from previous iterations.

Approaches for taxonomy construction from knowledge base data have been proposed in many studies [21]. Some well known sytsems are YAGO [7] and WikiTaxonomy [22]. Our approach differs from these in that we do not use an external knowledge base to build our taxonomy, and we build the taxonomy based on the scholarly articles, rather than Wikipedia or web pages.

III. System Overview

Our system builds on the approach described in [2]. Our system is given a set of documents in PDF format, and generates a knowledge base represented as a graph. The system is comprised of three main modules: a parser, an entity-relation extractor, and a taxonomy graph constructor. Below is a description of each module.

A. Parser

The first module in the pipeline is the parser. This is responsible for two types of parsing. The first is a rendering parser, which extracts the raw text from the PDF file. For this task we use the GROBID [23] parser. The second parser is a syntactic parser that extracts noun phrases. This module passes a set of parsed and part-of-speech-tagged sentences to the next phase.

B. Entity-relation Extractor

The module extracts related entities from a set of parsed sentences. The entities are nouns and noun phrases and the type of relationships are (1) the hyponymy relation among entities, (2) attributes of entities. In this section we discuss the types of relationships extracted by our framework.

Concept-Instance Relationships A concept-instance, or hypernym-hyponym, relationship is one where one entity is an instance of the concept represented by the other entity. This is also known as the *is-a* relationship. We define hyponymy relations to occur between 1) a *concept* and *sub-concept* or 2) between a *concept* and an *instance*. We

follow the same formal notation used in most hyponymy extraction studies [16], [14]. For a document set D, we extract a set of sentences S that contain candidate tuples (X_i, Y_j) such that,

$$s = \{(X, Y_1), (X, Y_2), ..., (X, Y_k)\}$$

where X is a hypernym for candidate hyponyms Y_j. Once the extraction is complete we have a set of extracted pairs,

$$P = \{(X_1, Y_1), (X_2, Y_2), ..., (X_N, Y_N)\}$$

Property Relationships The other relationship we extract is the *isPropertyOf* relationship. This extracts attributes of entities of the knowledge base. For example, in the phrase "the speedup of the algorithm", we can infer that *speedup* is an attribute of the concept *algorithm*.

The relation extraction module has two basic components. The first is a syntactic based extractor that takes the set of pre-defined extraction patterns as a basis for matching. The syntactic-based extractor, generates a set S of sentences containing candidate *is-a* triples. The next component is the iterative semantic learner, which iteratively learns new triples from the set S and populates the set of extracted pairs P. This step passes a set of hypernym-hyponym triples to the taxonomy construction phase.

C. Taxonomy Graph Constructor

Taxonomy construction is accomplished through three main graph operations. The set of extracted triples are initially mapped to local taxonomy graphs. Next, based on the existence of similarities among nodes of distinct local taxonomies, we apply modifier, simple, and vertical node merging operations to generate the final knowledge graph G. The knowledge base is constructed by building a taxonomy T by combining the local taxonomies identified in by the triples in the set P. We define the sets X={x_1, ... ,x_m} and Y={y_1, ... ,y_n} , with $|X|$=m and $|Y|$=n, to be the sets of individual words comprising the noun phrase of a candidate hypernym and hyponym, respectively.

IV. Extraction Approach

In this section we describe the two modules responsible for generating the knowledge base tuples of entities related with a hypernym-hyponym relation and the attributes. The input to this step is a set of PDF documents D and the output is the set P of pairs (X,Y).

A. Syntactic-based Extraction

Since our approach does not rely on an existing knowledge base, we bootstrap our algorithm by extracting pairs using a set of hand-crafted patterns. The use of lexico-syntactic patterns for bootstrapping hyponymy extraction is a common practice since they have relatively high precision while compromising recall, which is tolerable for a bootstrapping step. We use the Hearst patterns from [15] that define syntactic patterns that are commonly used to denote a hyponymy relationship. Table I shows the patterns used in our system. NP stands for *Noun Phrase*, that are first identified and then matched against the pattern. The Stanford lexical parser [24] parses the sentence to extract noun phrases. These are in turn passed to a *lexical tree* matcher that identifies the patterns according to the noun phrase identified at the highest level in the node. For instance the sentence by applying pattern 1 in Table I: NP_1 such as NP_2 to the following sentence:

> *"high and low level programming languages* ***such as*** *Assembly, Java, and C++"*

will extract NP_1 as the entire phrase of *high and low level programming languages* instead of *low level programming languages*. The Stanford token matcher [25] is used to apply the patterns to the tokenized sentence to obtain this result. The extracted candidate sentences are then parsed by identifying the sets of X and Y words comprising each of the hypernym and hyponym phrases, respectively. Stemming and removing initial stop words that are common adjectives is performed to cleanse the phrase prior to the hypernym detection step.

B. Semantic Unit Extraction

The bootstrapping approach relies on syntactic patterns that exploit part-of-speech tags to identify noun phrases. However, the resulting phrases can be very noisy if they contain identifiers and certain types of adjectives. In order to improve the quality of extracted concept phrases, we use the notion of *semantic units* defined as "the maximal-length noun phrase representing a single entity uniquely and fully, regardless of the textual contextâĂİ. We adopt an empirical approach based on large-scale corpus experimentation to derive heuristics. We define two sets of rules to help us linguistically categorize a semantic unit, frontier rules and parsing rules shown in table II. The frontier rules are used to strip the *front* of the phrase by recursively matching the first token in the phrase against the rule until none of the rules can be applied. The parsing rules take advantage of the pos tags to further reduce noun phrases to as basic a unit as possible.

C. Iterative Semantic Learning

Syntactic extraction generates a set C of candidate pairs that need to be classified as acceptable pairs or ones that should be discarded. The iterative learning steps populates a set P of valid pairs that is initially empty. A first pass over the sentences will add any pair for which the hypernym phrase contains only a single word. If the hypernym phrase contains more than one word, the phrase must be processed by the hypernym extraction algorithm for phrase-detection. If a valid hypernym phase is detected we add the triple to the set P. We note here that since we used a lexical token matcher, the hyponym phrases required very little processing for validation. Thus, we simply used stemming and lemmatization to reduce them to their base form.

TABLE I: Lexicosyntatic Patterns

#	Hearst Hyponymy Patterns
1	*NP* such as *NP* (, *NP* and/or *NP*)
2	such *NP* as *NP* (,*NP* and/or *NP*)
3	*NP*, including *NP* (,*NP* and/or *NP*)
4	*NP* (, *NP*) and other *NP*
5	*NP* (, *NP*) or other *NP*
6	*NP*, especially | called | named | including *NP* (, *NP* and/or *NP*)
#	Attribute Patterns
6	the *NP* of (the | a | an) *NP* [is]

TABLE II: Semantic Unit Extraction Rules

Frontier Rules	Parsing Rules
verb+!noun	*NP/NN* conj. *NP/NN*
non-attr. adj.	*NP/NN* prep. *NP/NN*
determiner	
special char.	
special adj.	
numeric	

The challenge in extracting a concept and instance in an *is-a* pair is the identification of the concept to a precise level. Instances are typically listed by authors with less descriptors when they are enumerated in a sentence, thus the pattern recovers them in a more clean form than the concept. The concept is usually a noun phrase that can sometimes be a full sentence and more common than not, can be reduced to a more basic noun phrase or single noun. Thus, for our hypernym (concept) extraction algorithm we apply a modified version of the probabilistic approach used in [14]. Given a candidate {X,Y} pair, we detect the values of $x_i \in X$ for the hypernym phrase. This may be a single value of x, or a subset, which we use as a sorted set by computing the n-grams from the set X. For that, we compute the likelihood ratio between the two words with highest likelihood of X. For each $x_i \in X$ and $\chi_j \subset X$, where χ_j is a subset of ordered n-gram words from X, we compute the ratio between the two terms x_1 and x_2 with highest posterior $p(x_i \mid Y)$.

V. Scientific Taxonomy Construction

The knowledge base extracted from the previous step in the set P is a set of N hypernym-hyponym pairs. In order to construct a taxonomy graph from these pairs we first represent each pair as a local taxonomy tree rooted at the hypernym. The hyponym will be a child node. The construction of the taxonomy is thus reduced to a graph merging problem where initially we have N local taxonomy trees. We define three rules used as a basis for the taxonomy construction.

Axiom 1 (Modifiers). *A noun phrase comprised of a modifier and a noun, is almost always a subconcept of the noun. Thus, if* $(X, Y) \in P$ *and* X=$\{x_1$,$x_2\}$*, then* $(x_2, X) \in T$*.*

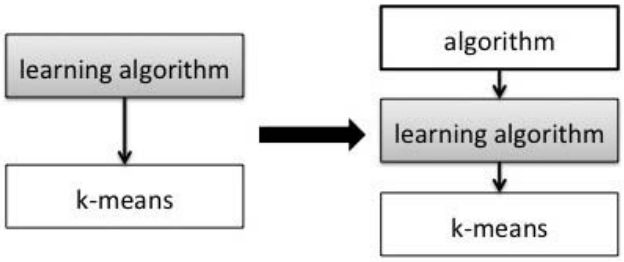
Input : Local taxonomy graphs set T
Output : Taxonomy Graph G
$G \leftarrow \emptyset$ **foreach** $t \in T$ **do**
  if $|t| = 2$ **then**
    $G \leftarrow \text{node}(t_2, t)$
  end
end
foreach $t_i, t_j \in T$ **do**
  if $t_i.children \cap t_j.children \neq \emptyset$ **then**
    $G \leftarrow \text{merge}(t_i.root, t_j.root)$
  end
end
foreach $t_i \in T$ **do**
  foreach $v \in t_i.children$ **do**
    foreach $t_j \in T$ **do**
      if $t_j.children \cap t_i.children \neq \emptyset$ **then**
        $G \leftarrow \text{merge}(v, t_j.root)$
      end
    end
  end
end
return : G

Algorithm 1: Taxonomy Graph Construction Algorithm

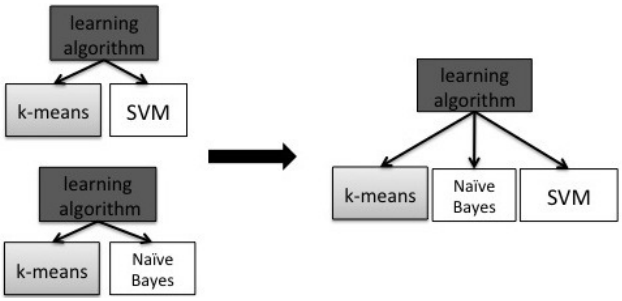
For example, we can easily infer from the phrase *learning algorithm* that it is a subconcept of the more abstract concept *algorithm*. This allows to identify the specific type of algorithm for other instances in other pairs by adding them under their specific concept, this process is covered by the other merge operations.

In order to construct the taxonomy from the individual pairs, pairs containing the same concepts must be merged into a single node in the graph. Simple surface form comparing will not do in this case because of possible polysemous terms and phrases. Thus we derive the following two propositions to address the case of a single surface form that has more than one meaning.

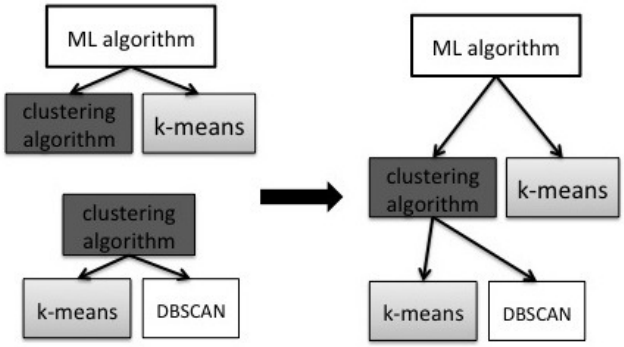
Axiom 2 (Horizontal Merging). *For two pairs with same text values for their hypernym noun phrases, the noun*



(a) Modified noun node insertion



(b) Simple similar root node merge.



(c) Vertical similar node insertion.

Fig. 1: Graph insertions and merges for taxonomy construction.

phrases have the same meaning if they were extracted from the same sentence. If they are not in the same sentence, they have the same meaning if they have similar instances. For pairs $(X_1, Y_1), (X_2, Y_2) \in P$*, if* $X_1 \cap X_2 = X_1 = X_2$ *and* $Y_1 \cap Y_2 \neq \emptyset$*, then* $(X_1, Y_1 \cup Y_2) \in T$*.*

Axiom 3 (Vertical Merging). *For two pairs, if the instance noun phrase of the first pair has the same text value as that of the concept of the other pair, they have the same meaning if the instance's coordinate terms are similar to the instances of the concept in the other pair. For pairs* $(X, Y_1), (V, Y_2) \in P$*, if* $Y_1 \cap V = Y_1 = V$ *and there exist other* Y_i *and* Y_j *such that* $(X_1, Y_i) \in P$*,* $(V, Y_j) \in P$ *and* $Y_i \cap Y_j \neq$ *then* $(X, V) \in T$*.*

Our taxonomy construction algorithm is described in Algorithm 1. Initially the taxonomy graph is empty. Based on the our first proposition, the method performs a `ModifierMerge` operation. This generates a new local taxonomy t of a concept-subconcept relationship. This steps adds these nodes to the set of local taxonomies. This is illustrated in Figure 1a.

Next, using the second proposition, we perform a `SimpleMerge` operation, where nodes whose roots are similar and belong to the same sentence are horizontally merged into one parent node with the union of children of both nodes. In the case where the two nodes did not occur in the same sentence, the similarity between their children is used to determine whether to merge them or not. If the children have some overlap then we consider the nodes as the same node and merge them into one node. An example is shown in Figure 1b

The third proposition is used for the `VerticalMerge`, shown in Figure 1c, which aims to increase the hierarchy level in the graph. We apply this merge by observing the overlap between the each child of a node and the root nodes of other local taxonomies.

VI. Evaluation

In this chapter we discuss a detailed evaluation on the knowledge base construction method. We evaluate the proposed algorithms as well as discuss the properties of taxonomy that is built using the proposed approach. First we discuss the details of our data set, and then describe the experimental setup used to evaluate the knowledge base construction portion of this thesis.

The first step is to evaluate the quality of the proposed hyponymy extraction method. The second is to evaluate the quality of the taxonomy construction method. Since no ground truth labeled data is available at the time of the study, we adopt the most common evaluation measures used in prominent studies of similar problems [14], [26], [6]. For the hyponymy extraction evaluation we define two evaluation metrics: correctness using precision and recall, and relevance using scoring scale well known in the literature [26]. The taxonomy construction experiments are designed to capture the quality of the resulting knowledge base in terms of: quality of entity categories and entity coverage of the taxonomy.

The hyponymy extraction method was applied to a data set of 10k research articles from CiteSeerX[1] published between 2004 and 2014 in top 50 computer science conferences. The topics covered include data mining, artificial intelligence, computer security and most computer science topics.

A. Hyponymy Extraction

In this section we describe two separate experiments conducted to evaluate correctness of the extracted triples and relevance of the triple statement, respectively.

[1]http://citeseerx.ist.psu.edu

TABLE III: Extractions per iteration in Iterative Learning algorithm (ILA) starting with 16,465 candidate sentences& Taxonomy node counts after each merge operation

ILA iteration #	# of triples found	#of new triples found
1	15,006	15,006
2	15,064	58
3	15,065	1
4	15,066	1
5	15,066	0
Number of local taxonomies after merging		
`Triples to Local Tax.`	15,066	
`Simple Node Merge`	3,564	
`Vertical Expansion`	1,405	

TABLE IV: Taxonomy nodes with most number of children and a subset of instances.

Hypernym	#of children	Sample categories	Sample Instances
`application`	962	web application, real-world application, data-driven application	iChat, telnet, online game, internet telephony, Adobe Acrobat, Gmail, Facebook, web browser, automated tuturing, GIMP, Bugzilla, Emule
`system`	375	operating system, voip system, manipulation system, NLP system, database system, complex system, storage system, content-based system, version control system, navigation system, legacy system,	email, YouTube, PEACH, instant messaging IM, content sharing site, voip system, JNI, SWIG, Poly-glot, ePic, Snort, KronoBase, XSB, PersonalRAID, Six Sigma, gesturePen, SRAM cache, Gnutella, NFS
`algorithm`	178	mining algorithm, learning algorithm, hasg-based algorithm, clustering algorithm, parallel algorithm	expectation maximization EM, SampleRank, SHA-2, mRMR, AdaBoost, Spherical Deconvolution, SATF, Support Vector Machine, Schapire

TABLE V: Comparison of taxonomy graph properties.

	# tuples	Avg. # children	Avg. depth	Max depth
UG-PL	10,589	0.0001	1	3
NG-PL	15,066	0.001	1	3

1) Correctness: To evaluate the quality of our extracted hypernym-hyponym pairs, we randomly select pairs extracted from 400 documents. We extract the *is-a* pairs from these documents using our approach that incorporates the ngrams of the hypernym phrase and compare the results with that of applying the SuperConceptDetection algorithm described in [14]. We refer to our approach as NG-PL, from n-gram probabilistic learner and the baseline as UG-PL, since it considers only unigrams. From the 10k documents the amount of harvested pairs is 17k, from which we select $1,000$ by each method. The triples were annotated by three computer science graduate students. The annotators were presented with the triple and the text containing from which it was extracted and is asked to assign a score to each extracted triple. The scoring system uses a scale ranging 0-4, similar to the one in [26], defined as follows:

0: The extracted relation is nonsense.
1: The extracted relation is not nonsense but unhelpful. This covers the case where the relationship is too abstract to be useful or the phrase is not descriptive enough to express a clear relationship.
2: Opinion/I don't know. An example of this case the pair may contain a very specific concept that cannot be known without knowing the context of the paper it was extracted from.
3: The extracted relation is somewhat true. This is used when the pair is generally correct but requires only small modification to become fully true.
4: The extracted relation is correct as is.

Table VII shows the scores given by the annotators.

2) Relevance: To evaluate the relevance of the extracted triples we ask the annotators to evaluate the extracted triples based on whether they are concrete or abstract. A concrete relationships is where the hypernym is considered to be one of the closest terms or categories the hyponym can be described with. If the hypernym is a general concept and not considered descriptive enough of the hyponym, it is considered to be abstract. Table VI shows the percentages of both as labeled by the annotators. The last row is a measure of whether the extracted phrase is a complete phrase in itself. This evaluates the semantic unit detection, if the phrase can hold it's meaning and carry the same meaning even when taken out of the context it is mentioned in, it is considered a well-phrased semantic unit.

TABLE VI: Relevance of Extracted Hyponymy Relationships

Measure	% of Triples
Concrete	82.8
Abstract	17.2
Phraseness	93.39

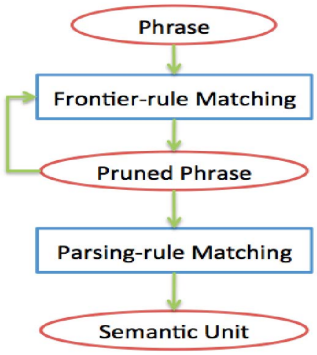


Fig. 2: Semantic Unit Extraction Steps.

B. Taxonomy Evaluation

In this section we report on the evaluation of the constructed taxonomy graph and the resulting knowledge base as a source for scientific knowledge.

1) Correctness: To evaluate the accuracy of the categories assigned to concepts extracted in our knowledge base, we conduct two evaluations. The first is similar to the one in the previous section where we ask users to annotate the categories of a subset of $1,000$ entities. For this task we ask 6 computer science graduate and senior undergraduate students to score the extractions on the same scale of 0-4. The results are shown in Table VII.

The second evaluation of the terms is conducted by a comparison with an existing knowledge base. We compare the assigned hypernym for each entity to those extracted by the Microsoft Concept knowledge base[2]. This is built using the methods proposed in [14] and provides an API to look up entities and retrieve their hypernyms, which is referred to as "class" in the API. By applying our taxonomy construction system to the 17k candidate sentences, we harvest over 15k entities and their hypernym. For each entity if the MS concept graphs returns a hypernym for that entity that matches the one extracted by our system we consider it a match. Out of $15,066$ entities, $9,160$ were found in MS concept graph, and $9,026$ of those entities' hypernyms were found in the classes returned by the API.

2) Coverage: In order to evaluate the coverage of the We compare the properties of the constructed knowledge graph using our hypernym detection algorithm for ngrams against the same method used only for unigrams, as in [14].

[2]https://concept.research.microsoft.com/

TABLE VII: Annotator scores for triple extraction on 1000 triples

Score	%Entity-Concept pairs
4	61.27%
3	04.27%
2	11.77%
1	06.77%
0	15.93%

Table V shows the values for the knowledge base size for each method.

To evaluate the quality of the knowledge base we asses the coverage of the entities. To do this we perform a concept-space evaluation similar to that used in [14] to evaluate the coverage of the knowledge base. This measures whether the concepts extracted are ones that are frequently queried, the assumption is if a query term is found in our knowledge base, then it has good coverage. To do this, we compile a set of 3 million query logs submitted to the scholarly search engine CiteSeerX during the years 2015 and 2016. After removing stop-words are from the terms in the queries, we rank the top unique query terms in the entire set of just over 500,000 unique terms. We found that 3504 of the 10k were found in the knowledge base. For the 50k most frequent query terms, 5810 of them were found in the KB of 8k entities.

C. Comparisons with Open Information Extractors

In this section we conduct an experiment to compare the effectiveness of our approach as whole in terms of: the choice of extracting only hyponymy relationships and the quality of the results per the metrics in the previous sections.

Open information extraction (OpenIE) tools such as Reverb [4], [13] and StanfordOpenIE [27] are designed to extract triples from unstructured text. OpenIE approaches do not assume a predefined schema for the triples extracted, thus the predicate, which is the relationship, can be of any type. In order to evaluate the effectiveness of our approach we compare the results obtained from our system with those obtained by the Reverb and Stanford OpenIE tools.

We run both openIE tools on a set of 100 candidate sentences extracted from over 150 research articles. Both the StanfordOpenIe tool, denoted S.OpenIE, and the Reverb tool extract all relationship types possible, whereas our approach, denoted NG-PL, extracts only hyponymy relationships. Table VIII shows the resulting evaluation of the triples extracted by each tool. The first row shows that the StanfordOpenIe tool, denoted S.OpenIE, has the highest recall as it harvests 396 triples from only 100 sentences.The phraseness column reports the amount of triples of which the subject and object were well-formed English phrases. The correctness column measures the triples that scored a 3 or 4 based on our correctness scale described in the previous section. For those correct

TABLE VIII: Hyponymy Extraction using NG-PL vs. OpenIE Systems on 100 Sentences

System	#Triples	is-A	Hyponymy	Phraseness
`S.OpenIE`	396	28	1	17
`ReVerb`	88	8	1	0
`NG-PL`	23	23	13	16

triples, the quality of the information carried in the triple is measured by whether the fact is concrete or abstract. As noted from the table, our system outperforms openIE tools in quality of phrase extraction, second best is Stanford OpenIE, which is expected since it relies on a dependency parse similar to our approach. Although, the lowest in the amount of extracted triples, our method has the highest ration in quality of triples, i.e. concrete facts.

Table VIII shows the amount of triples that are extracted as is-a relationships. Since both openIE tools extract all types of relationships, the table illustrates the importance of proposing our method for hyponymy extractions as the values show that none of these tools have high precision or recall when it comes to hyponymy relations Extraction. The column displaying the is-a values, shows the amount of triples that have the is-a as a relationship. However, since "is-a" can be used in English for purposes other than expressing hyponymy, we report the number of these triples that actually are instances of hyponymy use of the is-a predicate. The table shows that our method significantly outperforms both methods in extracting triples if we focus on hyponymy relations, which is the goal of building a knowledge base.

References

[1] A. Elkiss, S. Shen, A. Fader, G. Erkan, D. States, and D. Radev, "Blind men and elephants: What do citation summaries tell us about a research article?" *Journal of the Association for Information Science and Technology*, vol. 59, no. 1, pp. 51–62, 2008.

[2] R. A. Al-Zaidy and C. L. Giles, "Automatic knowledge base construction from scholarly documents," in *Proceedings of the 2017 ACM Symposium on Document Engineering*. ACM, 2017, pp. 149–152.

[3] I. Augenstein, M. Das, S. Riedel, L. Vikraman, and A. McCallum, "Semeval 2017 task 10: Scienceie-extracting keyphrases and relations from scientific publications," *arXiv preprint arXiv:1704.02853*, 2017.

[4] O. Etzioni, A. Fader, J. Christensen, S. Soderland *et al.*, "Open information extraction: The second generation," in *Twenty-Second International Joint Conference on Artificial Intelligence*, 2011.

[5] A. Singhal, "Introducing the knowledge graph: things, not strings," https://googleblog.blogspot.com/2012/05/introducing-knowledge-graph-things-not.html, 2012.

[6] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. R. Hruschka Jr, and T. M. Mitchell, "Toward an architecture for never-ending language learning." in *AAAI*, vol. 5, 2010, p. 3.

[7] F. M. Suchanek, G. Kasneci, and G. Weikum, "Yago: a core of semantic knowledge," in *Proceedings of the 16th international conference on World Wide Web*. ACM, 2007, pp. 697–706.

[8] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives, "Dbpedia: A nucleus for a web of open data," *The semantic web*, pp. 722–735, 2007.

[9] D. B. Lenat and R. V. Guha, *Building large knowledge-based systems; representation and inference in the Cyc project*. Addison-Wesley Longman Publishing Co., Inc., 1989.

[10] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor, "Freebase: a collaboratively created graph database for structuring human knowledge," in *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. AcM, 2008, pp. 1247–1250.

[11] D. Vrandečić and M. Krötzsch, "Wikidata: a free collaborative knowledgebase," *Communications of the ACM*, vol. 57, no. 10, pp. 78–85, 2014.

[12] O. Etzioni, M. Cafarella, D. Downey, S. Kok, A.-M. Popescu, T. Shaked, S. Soderland, D. S. Weld, and A. Yates, "Web-scale information extraction in knowitall:(preliminary results)," in *Proceedings of the 13th international conference on World Wide Web*. ACM, 2004, pp. 100–110.

[13] A. Fader, S. Soderland, and O. Etzioni, "Identifying relations for open information extraction," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2011, pp. 1535–1545.

[14] W. Wu, H. Li, H. Wang, and K. Q. Zhu, "Probase: A probabilistic taxonomy for text understanding," in *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*. ACM, 2012, pp. 481–492.

[15] M. A. Hearst, "Automatic acquisition of hyponyms from large text corpora," in *Proceedings of the 14th conference on Computational linguistics-Volume 2*. Association for Computational Linguistics, 1992, pp. 539–545.

[16] S. Cederberg and D. Widdows, "Using lsa and noun coordination information to improve the precision and recall of automatic hyponymy extraction," in *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*. Association for Computational Linguistics, 2003, pp. 111–118.

[17] R. Snow, D. Jurafsky, A. Y. Ng *et al.*, "Learning syntactic patterns for automatic hypernym discovery." in *NIPS*, vol. 17, 2004, pp. 1297–1304.

[18] M. Pasca and B. Van Durme, "What you seek is what you get: Extraction of class attributes from query logs." in *IJCAI*, vol. 7, 2007, pp. 2832–2837.

[19] T. Lee, Z. Wang, H. Wang, and S.-w. Hwang, "Attribute extraction and scoring: A probabilistic approach," in *Data Engineering (ICDE), 2013 IEEE 29th International Conference on*. IEEE, 2013, pp. 194–205.

[20] A. Yates, M. Cafarella, M. Banko, O. Etzioni, M. Broadhead, and S. Soderland, "Textrunner: open information extraction on the web," in *Proceedings of Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*. Association for Computational Linguistics, 2007, pp. 25–26.

[21] T. Swoboda, M. Hemmje, M. Dascalu, and S. Trausan-Matu, "Combining taxonomies using word2vec," in *Proceedings of the 2016 ACM DocEng Symposium on Document Engineering*. ACM, 2016, pp. 131–134.

[22] S. P. Ponzetto and M. Strube, "Deriving a large scale taxonomy from wikipedia," in *AAAI*, vol. 7, 2007, pp. 1440–1445.

[23] P. Lopez, "Grobid," https://github.com/kermitt2/grobid, 2008-2016.

[24] C. D. Manning, M. Surdeanu, J. Bauer, J. R. Finkel, S. Bethard, and D. McClosky, "The stanford corenlp natural language processing toolkit." in *ACL (System Demonstrations)*, 2014, pp. 55–60.

[25] A. X. Chang and C. D. Manning, "Tokensregex: Defining cascaded regular expressions over tokens," *Tech. Rep. CSTR 2014-02*, 2014.

[26] X. Li, A. Taheri, L. Tu, and K. Gimpel, "Commonsense knowledge base completion," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL), Berlin, Germany, August. Association for Computational Linguistics*, 2016.

[27] G. Angeli, M. J. Premkumar, and C. D. Manning, "Leveraging linguistic structure for open domain information extraction," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL 2015)*, 2015.