Mining Scientific Terms and their Definitions: A Study of the ACL Anthology

Yiping Jin[1] **Min-Yen Kan**[1,2] **Jun-Ping Ng**[1] **Xiangnan He**[1] *

[1]Department of Computer Science
[2]Interactive and Digital Media Institute
National University of Singapore
13 Computing Drive, Singapore 117417
{yiping, kanmy, junping, xiangnan}@comp.nus.edu.sg

Abstract

This paper presents DefMiner, a supervised sequence labeling system that identifies scientific terms and their accompanying definitions. DefMiner achieves 85% F_1 on a Wikipedia benchmark corpus, significantly improving the previous state-of-the-art by 8%.

We exploit DefMiner to process the ACL Anthology Reference Corpus (ARC) – a large, real-world digital library of scientific articles in computational linguistics. The resulting automatically-acquired glossary represents the terminology defined over several thousand individual research articles.

We highlight several interesting observations: more definitions are introduced for conference and workshop papers over the years and that multiword terms account for slightly less than half of all terms. Obtaining a list of popular defined terms in a corpus of computational linguistics papers, we find that concepts can often be categorized into one of three categories: resources, methodologies and evaluation metrics.

1 Introduction

Technical terminology forms a key backbone in scientific communication. By coining formalized terminology, scholars convey technical information precisely and compactly, augmenting the dissemination of scientific material. Collectively, scholarly compilation efforts result in reference sources such as printed dictionaries, ontologies and thesauri.

While online versions are now common in many fields, these are still largely compiled manually, relying on costly human editorial effort. This leads to resources that are often outdated or stale with respect to the current state-of-the-art. Another indirect result of this leads to a second problem: lexical resources tend to be general, and may contain multiple definitions for a single term. For example, the term "CRF" connotes "Conditional Random Fields" in most modern computational linguistics literature; however, there are many definitions for this acronym in Wikipedia. Because only one correct sense applies, readers may need to expend effort to identify the appropriate meaning of a term in context.

We address both issues in this work by automatically extracting terms and definitions directly from primary sources: scientific publications. Since most new technical terms are introduced in scientific publications, our extraction process addresses the bottleneck of staleness. Second, since science is organized into disciplines and sub-disciplines, we can exploit this inherent structure to gather contextual information about a term and its definition.

Aside from performance improvements, the key contributions of our work are in 1) recasting the problem as a sequence labeling task and exploring suitable learning architectures, 2) our proposal and validation of the the use of shallow parsing and dependency features to target definition extraction, and 3) analyzing the ACL Anthology Reference Corpus from statistical, chronological and lexical viewpoints.

This research is supported by the Singapore National Research Foundation under its International Research Centre @ Singapore Funding Initiative and administered by the IDM Programme Office.

2 Related Work

The task of definition mining has attracted a fair amount of research interest. The output of such systems can be used to produce glossaries or answer definition questions. The primary model for this task in past work has been one of binary classification: does a sentence contain a definition or not? Existing methods can be cast into three main categories, namely rule-based (Muresan and Klavans, 2002; Westerhout and Monachesi, 2007), supervised machine learning (Fahmi and Bouma, 2006; Westerhout, 2009), and semi-supervised approaches (Navigli and Velardi, 2010; Reiplinger et al., 2012).

Rule-based approaches are intuitive and efficient, and were adopted in early research. Here, system performance is largely governed by the quality of the rules. Muresan and Klavans (2002) developed a rule-based system to extract definitions from online medical articles. The system first selects candidates using hand-crafted cue-phrases (e.g. *is defined as*, *is called*; analogous to "IS-A" phrases), further filtering the candidates with grammar analysis. Westerhout and Monachesi (2007) augmented the set of rules with part-of-speech (POS) tag patterns, achieving an F_2 of 0.43.

While such manually-crafted expert rules have high precision, they typically suffer from low recall. Definitions can be expressed in a variety of ways, making it difficult to develop an exhaustive set of rules to locate all definition sentences. To address low recall, later research adopted data-driven machine learning approaches. Fahmi and Bouma (2006) made used of supervised machine learning to extract definitions from a corpus of Dutch Wikipedia pages in the medical domain. They showed that a baseline approach which simply classifies every first sentence as a definition works surprisingly well, achieving an accuracy of 75.9% (undoubtedly due to the regular structure and style for Wikipedia). Their final system, based on the important feature of sentence position, was augmented with surface-level features (bag-of-words, bigrams, etc.) and syntactic features (type of determiner, position of the subject in the sentence). Their study with three different learners – naïve Bayes, maximum entropy (MaxEnt) and the support vector machine (SVM) – showed that MaxEnt gave the best results (92.2% accurate).

Westerhout (2009) worked on a hybrid approach, augmenting a machine learner to a set of hand-written rules. A random forest classifier is used to exploit linguistic and structural features. Informed by Fahmi and Bouma (2006)'s study, she included article and noun types in her feature set. Lexico-structural cues like the layout of the text are also exploited. She evaluated the performance of different cue phrases including the presence of "IS-A" phrases, other verbs, punctuations and pronouns. The highest F_2 score of 0.63 is reported for the "IS-A" pattern.

Since 2009 the focus of the research shifted to methods not limited to feature engineering. Borg et al. (2009) implemented a fully automated system to extract and rank definitions based on genetic algorithms and genetic programming. They defined two sub-problems including 1) acquiring the relative importance of linguistic forms, and 2) learning of new linguistic forms. Starting with a small set of ten simple hand-coded features, such as having sequence "FW IS" (FW is a tag for foreign word) or containing keyword identified by the system, the system is able to learn simple rules such as "NN is a NN". However, their system is optimized for similar "IS-A" patterns, as was used in (Westerhout, 2009). Their system, achieving an average f-measure of 0.25, also performs poorer than machine learning systems which exploit more specific features.

To cope with the generality of patterns, Navigli and Velardi (2010) proposed the three-step use of directed acyclic graphs, called Word-Class Lattices (WCLs), to classify a Wikipedia dataset of definitions. They first replace the uncommon words in the sentences with a wildcard (*), generating a set of "star patterns". Star patterns are then clustered according to their similarity. For each cluster, a WCL is generated by aligning the sentences in the cluster to form a generalized sentence. Although they reported a higher precision and recall compared with previous work, the result for WCL (F_1 of 75.23%) is not significantly better than the baseline system which exploits only star patterns (F_1 of 75.05%) without generating the directed graphs.

Reiplinger et al. (2012) took a semi-supervised approach. They employed bootstrapping to extract glossary sentences from scientific articles in

the ACL Anthology Reference Corpus (ACL ARC) (Bird et al., 2008). Their results show that bootstrapping is useful for definition extraction. Starting from a few initial term-definition pairs and hand-written patterns as seeds, their system iteratively acquires new term-definition pairs and new patterns.

We note that these previous systems rely heavily on lexico-syntactic patterns. They neither sufficiently exploit the intrinsic characteristics of the term and definition, nor invest effort to localize them within the sentence[1]. Given the significant structure in definitions, we take a more fine-grained approach, isolating the term and its definition from sentences. According to Pearson (1996), a definition can be formally expressed as:

$$X = Y + \text{distinguishing characteristics},$$

where "X" is the *definiendum* (defined term; hereafter, *term*), and "Y + distinguishing characteristics" can be understood as the *definiens* (the term's definition; hereafter, *definition*). The connector "=" can be replaced by a verb such as "define", "call", a punctuation mark or other phrase. Our task is thus to find pairs of terms and their associated definitions in input scholarly articles. The sentence-level task of deciding whether a sentence s is a definition sentence or not, is thus a simplification of our task.

3 Methodology

Our DefMiner system is based on the sequence labeling paradigm – it directly assigns an annotation a_i from $A \in \{$(T)*erm*,(D)*efinition*,(O)*ther*$\}$ for each input word w_i. We post-process our labeler's results to achieve parity with the simplified definition sentence task: When we detect both a term's and definition's presence in a sentence, we deem the sentence a definition sentence. To be clear, this is a requirement; when we detect only either a term or a definition, we filter these out as false positives and do not include them as system output – by definition in DefMiner, terms must appear within the same sentence as their definitions.

To train our classifier, we need a corpus of definition sentences where all terms and definitions are annotated. While Navigli and Velardi (2010) compiled the WCL definition corpus from the English Wikipedia pages, we note that Wikipedia has stylistic conventions that make detection of definitions much easier than in the general case (i.e., "The first paragraph defines the topic with a neutral point of view"[2]). This makes it unsuitable for training a general extraction system from scholarly text.

As such, we choose to construct our own dataset from articles collected from the ACL ARC, following (Reiplinger et al., 2012). We compiled a corpus – the W00 corpus, named for its prefix in the ACL Anthology – derived from a total of 234 workshop papers published in 2000. Due to limited resources and time, only one individual (the first author) performed the corpus annotation. We built three disjoint prototype classifiers to further filter the sentences. The prototype classifiers are based on surface patterns, keyphrases and linguistic phenomena (# of NP, # of VP, etc.). We took all the 2,512 sentences marked as definition sentences by at least one of our individual prototypes and proceeded to annotate all of them. In total, 865 of the total 2,512 sentences were real definition sentences.

The annotation instance is a single token (including single word or punctuation mark). Each token w_i was marked with a_i from $A \in \{$T,D,O$\}$ depending on whether it is part of a term, a definition or neither (other). Therefore, a sentence that is not a definition sentence would have all its tokens marked as O. The corpus and its annotations are available for comparative study[3].

We use Conditional Random Fields (CRFs) (Lafferty et al., 2001) to extract the term and definition from input. We incorporate evidence (features) drawn from several levels of granularity: from the word-, sentence- and document-levels, not limiting ourselves to the window of previous n words. CRFs allow us to encode such features which may not be conditionally independent. We use the open source implementation, CRF++ (Kudo, 2005) in our work[4].

One straightforward approach is to train independent classifiers for terms and definitions, which we

[1] While Navigli and Velardi (2010) tagged *terms* and *definitions* explicitly in their corpus, their evaluation restricts itself to the task of definition sentence identification.

[2] http://en.wikipedia.org/wiki/Wikipedia:Manual_of_Style/Lead_section

[3] http://www.comp.nus.edu.sg/downloads/term_definition_mining/

[4] http://code.google.com/p/crfpp/

test in Section 4.1. While simple, this is suboptimal as it ignores the correlation between the presence of the two components. Term identification (in the guise of key-word/-phrase extraction) is well studied and common features such as $tf.idf$ and English orthography achieve satisfactory results (Nguyen and Kan, 2007). In contrast, definitions exhibit more flexible structure and hence are more difficult to distinguish from normal English text.

As such, we further investigate a serial architecture where we perform the classifications in sequence. *I.e.*, first utilizing the results from *term* classification, and then incorporating them into *definition* classification. This two-stage architecture is explored later in Section 4.2

Our expanded feature set is an amalgamation of related works and utilizes a mix of simple lexical, orthography, dictionary lookup and corpus features (here, idf). Note that each feature class may derive more than one feature (e.g., for the POS tag feature, we extract features from not only the current word but the surrounding contextual words as well). We now enumerate the feature classes (FCs) that we exploit, marking whether they apply to the (W)ord, (S)entence or (D)ocument levels:

FC1) Lexical (W): The word, POS tag, stemmed word, and if the word contains a signal suffix[5].
FC2) Orthography (W): Whether the word is 1) capitalized or 2) mixed case; whether the word contains 3) hyphens or 4) digits.
FC3) Keyphrase List (W): Whether the word is in the keyphrase list of the origin document. We use the open source KEA keyphrase extraction system (Witten et al., 1999) to extract 20 keyphrases for each document.
FC4) Corpus (W): Discretized Inverse Document Frequency (idf), calculated as $log(\frac{N}{c})$, where N is the total number of documents and c is the number of occurrences of the word in all the documents. IDF values are discretize into eight uniform partitions.
FC5) Position (D,S): The 1) Section ID, 2) name and 3) the sentence's relative position in the document.
FC6) Has acronym (S): Whether the sentence contains an acronym. We use Stanford dependency parser (Cer et al., 2010) to parse the sentences. We deem the sentence to contain an acronym if the dependency type "abbrev" is present in the output of the parser.
FC7) Surface pattern (S): Whether the sentence contain one of the hand-crafted pattern, as listed in Table 1. The list is compiled from previous works and augmented based on our observations on the corpus.

[5]The suffixes we extract are "-ion","-ity","-tor","-ics","-ment","-ive" and "-ic".

<term > defined (as|by) <definition>
define(s)? <term> as <definition>
definition of <term> <definition>
<term> a measure of <definition>
<term> is DT <definition> (that|which|where)
<term> comprise(s)? <definition>
<term> consist(s)? of <definition>
<term> denote(s)? <definition>
<term> designate(s)? <definition>
<definition> (is|are|also) called <term>
<definition> (is|are|also) known as <term>

Table 1: Hand-crafted surface patterns used in DefMiner.

Long Distance Features. During development, we noticed that the syntactic variation of the definition might benefit from features that identify long-distance dependencies. As such, we further studied the impact of including additional features developed from the shallow (chunk) and dependency parses of the input. Compared to the above features, these features are much more computationally-intensive.

FC8) Shallow tag (W): Shallow parsing tag for each word (e.g., np, vp). We used OpenNLP toolkit to shallow parse the sentences. (Baldridge, 2005)
FC9) Shallow pattern (S): If the shallow parsing sequence contains one of the seven parse patterns listed in Table 2. We also give some example sentences which can be detected by the patterns.
FC10) Governor (W): The word that the current word depends on in a binary dependency relation. (e.g., for the phrase $computational\ linguistics$, the governor of the word $computational$ is $linguistics$).
FC11) Dependency path distance (W): Distance from the current word to the root of the sentence in the dependency tree.
FC12) Typed dependency path (W): The dependency path from the current word to the root of the sentence (recording the dependency types instead of the words in the path).

Pattern	Example
NP : NP	JavaRAP : An open-source implementation of the RAP
NP is * NP	IR is the activity of obtaining information resources
NP is * NP that/of/which	NLP is a subject which is well studied
NP or NP	Conditional Random field or CRF tackles ...
known as NP	The corpus of English Wikipedia pages, known as EnWiki
NP (* NP)	Hidden Markov Model (HMM) is used to solve ...
NP defined by/as * NP	The accuracy is defined by the production of ...

Table 2: Hand-crafted shallow parsing patterns used in DefMiner.

4 Evaluation

We now assess the overall effectiveness of DefMiner, at both the word and sentence level. Additionally, we want to ascertain the performance changes as we add features to an informed lexical baseline. We not only benchmark DefMiner's performance over our own W00 collection, but also compare DefMiner against previous published work on the definition sentence identification task on the WCL (English Wikipedia) corpus.

4.1 Single-Pass Extraction from W00

We run ten-fold cross validation on our annotated W00 corpus. We first evaluate our results at word level, calculating the precision, recall, and F_1 scores for each incrementally enhanced feature set. We present results on the corpus in the top portion of Table 3 (Rows 1–9).

We calculate both micro and macro- (category) averaged F_1 scores for term and definition extraction. F_{micro} assigns equal weight to every token, while F_{macro} gives equal weight to every category. As definition tokens greatly outnumber term tokens in our corpus (roughly 6:1), we feel that the macro-average is a better indicator of the balance between term and definition identification.

Our baseline system makes use of basic word and POS tag sequences as features (FC1), which are common to baselines in other sequence labeling works. We can see that most features result in performance improvements to the baseline, especially for recall. Interestingly, although the shallow parsing and dependency features we use are rather simple, they effectively improve the performance of the system. In System 7, we only use the seven shallow parsing patterns shown in Table 2, but the F_{macro} measure improves 3%. Our best single-stage system (System 9 in Table 3) boosts recall for term and definition classification by 7% and 5%, respectively, without sacrificing precision. The F_{macro} measure is improved from 0.44 to 0.48.

Unexpectedly, the inclusion of the position features cause performance to drop. One possible reason is that the authors of scientific papers have more flexibility to choose the positions to present definitions. This makes the position feature much less indicative (compared to running on a corpus of Wikipedia articles). Due to this observation, we exclude the position features when carrying out following experiments.

4.2 Serial Term and Definition Classification

We now investigate the two-stage, serial architecture where the system first performs term classification before definition classification (i.e., term$\succ$definition). We provision the second-stage definition classifier with three additional features from the first-stage term classification output: whether the current word (1) is a term, and (2) appears before or (3) after a term.

Row 10 shows this resulting system, which we coin as DefMiner. Interestingly, there is a 10% increase in the precision of definition classification. With the two-stage classifier, F_{macro} score further increases from 0.48 to 0.51. The results verify our intuition that term classification does help in definition classification. Pipelining in the opposite direction (definition$\succ$term; Row 11) does not show any improvement. We posit that since the advantage is only in a single direction, joint inference may be less likely to yield benefits.

To determine the upper bound performance that could result from proper term identification, we provided correct, oracular term labels from our ground truth annotations in our corpus to the second-stage definition classifier. This scenario effectively upper-bounds the performance that perfect term knowledge has on definition classification. The results of this system in Row 12 indicates a strong positive influence on definition extraction, improving definition extraction from 49% to 80%, a leap of 31%. This

System / Feature Class (*cf* Section 3)	Term			Definition			Overall	
	P	R	F_1	P	R	F_1	F_{micro}	F_{macro}
1: Baseline (FC1)	0.49	0.34	0.40	0.41	0.49	0.45	0.45	0.44
2: (1) + Orthography (FC2)	0.46	0.35	0.40	0.42	0.51	0.46	0.46	0.44
3: (2) + Dictionary (FC3)	0.48	0.36	0.41	0.41	0.49	0.44	0.44	0.43
4: (3) + Corpus (FC4)	0.50	0.35	0.41	0.40	0.52	0.45	0.45	0.44
5: (4) + Position (FC5)	0.47	0.37	0.42	0.36	0.48	0.41	0.41	0.41
6: (4) + Shallow parsing tag (FC8)	**0.51**	0.38	0.43	0.41	0.50	0.45	0.45	0.44
7: (6) + Shallow parse pattern (FC9)	0.50	0.40	**0.45**	0.42	0.52	0.47	0.47	0.47
8: (7) + Surface pattern (FC7)	0.49	0.39	0.44	0.43	0.53	0.48	0.48	0.47
9: (8) + Dependency + acronym (FC6,10,11,12)	0.50	**0.41**	**0.45**	0.45	0.54	0.49	0.49	0.48
10 [DefMiner]: (9) + 2-stage	0.50	**0.41**	**0.45**	**0.55**	**0.58**	**0.56**	**0.55**	**0.51**
11: (9) + Reverse 2-stage	0.50	0.40	0.44	0.45	0.54	0.49	0.49	0.48
12: (9) + Term Oracle	*N/A*	*N/A*	*N/A*	0.79	0.82	0.80	*N/A*	*N/A*

Table 3: 10-fold cross validation word-level performance over different system configurations on our W00 corpus.

motivates future work as how to improve the performance of the term classifier so as to reap the benefits possible with our two-stage classifier.

4.3 Comparative results over the WCL Dataset

For most of the related research reviewed, we could neither obtain their source code nor the corpora used in their work, making comparative evaluation difficult. To the best of our knowledge, Reiplinger et al. (2012) is the only attempt to extract definitions from the ACL ARC corpus, which is a superset of our W00 corpus. It would be desirable to have a direct comparison with their work, but their evaluation method is mainly based on human judges and their reported coverage of 90% is only for a sample, short list of domain terms they defined in advance.

To directly compare with the previous, more complex state-of-the-art system from (Navigli and Velardi, 2010), we evaluate DefMiner on the definition sentence detection task. For the sentence-level evaluation, we calculate the $P/R/F_1$ score based on whether the sentence is a definition sentence. We applied DefMiner on their whole WCL annotated corpus, reporting results in Table 4. We randomized the definition and none-definition sentences in their corpus and applied 10-folds cross validation. In each each iteration we used 90% of the sentences for training and 10% for testing.

Compared to their results reported in (Navigli and Velardi, 2010), DefMiner improves overall F_1 by 8%. While certainly less precise (precision of 92% versus 99%), recall is improved over their considerably more complex WCL-3 algorithm by almost 20%. Even using just the simple heuristic of only classifying sentences that have identified terms as well as definitions as definition sentences, DefMiner serves to competitively identify definition sentences.

System	Token Level		Sentence Level
	Term	Definition	
	P / R / F_1	P / R / F_1	P / R / F_1
DefMiner	.82/.78/.80	.82/.79/.81	.92/.79/.85
N&V '10	– / – / –	– / – / –	.99/.61/.77

Table 4: Comparative performance over the WCL.

4.4 Manual Inspection of DefMiner Output

To gauge the noise from our system outside of our cross-validation experiments, we conducted a manual inspection of results over other workshop papers from other years (2001 and 2002), as a sanity check. DefMiner identifies 703 and 1,217 sentences in W01 and W02 as definition sentences separately.

Overall, 77.8% of the extracted sentences are real definition sentences, while the remaining are false positives.

4.4.1 Analysis of Common Errors

The $P/R/F_1$ score by itself only gives a hint of the system's overall performance. We are also interested to study the common errors made by our system, which could help us engineer better features for improving DefMiner. As our two-stage classifier still lags behind the system with oracular term labels by 24% in F_1 for definition detection (Section 4.2), we believe there is still much room for improvement. We show three example misclassified sentences that represent the major types of errors we observed, where DefMiner's output annotations follow tokens marked as part of terms or definitions.

1) *A PSS/TERM thus contains abstract linguistic values for closed features (tense/DEF ,/DEF mood/DEF ,/DEF voice/DEF ,/DEF number/DEF ,/DEF gender/DEF ,/DEF etc/DEF ./DEF) .*

This first instance shows that DefMiner tends to mark the first several tokens as "TERM" while the real term appears somewhere else in the sentence. The actual term being defined is "closed features" instead of "PSS". Many terms in the training set appear at the beginning of the sentence and are preceded by a determinant. "PSS" is also likely to receive a high IDF and orthographic shape (capitalized) score and therefore are misclassified as terms. It may be useful to thus model the (usual) distance between the term and its definition in a feature in future work.

2) *Similarly , 'I'/TERM refers to an/DEF interior/DEF character/DEF and/DEF 'L'/DEF indicates/DEF the/DEF last/DEF character/DEF of/DEF a/DEF word/DEF .*

DefMiner is occasionally confused when encountering recursive definition or multiple definitions in a single sentence. Sentence 2 contains two parallel definitions. The classifier fails to classify "L" as a separate term, incorporating it as part of the definition. One possible improvement is to break the original sentence into clauses that are independent from each other, perhaps by using even simple surface cues such as coordinating conjunctions marked by commas or "and".

3) *Again one could argue that the ability to convey such uncertainty and reliability information to a non-specialist/TERM is a/DEF key/DEF advantage/DEF of/DEF textual/DEF summaries/DEF over/DEF graphs/DEF .*

Another difficult problem faced by the classifier is the lack of contextual information. In sentence 3), if we just look at part of the sentence "a non-specialist is a key advantage of textual summaries over graphs", without trying to understand the meaning of the sentence, we may well conclude that it is a definition sentence because of the cue phrase "is a". But clearly, the whole sentence is not a definition sentence. More sophisticated features based on the sentence parse tree have to be exploited to detect such false positive examples.

5 Insights from the Definitions Extracted from the ACL ARC

In this second half of the paper, we apply DefMiner to gain insights on the distributional and lexical properties of terms and definitions that appear in the large corpus of computational linguistics publications represented by the ACL ARC (Bird et al., 2008). The ARC consists of 10,921 scholarly publications from ACL venues, of which our earlier W00 corpus is a subset (*n.b.*, as such, there is a small amount of overlap). We trained a model using the whole of the W00 corpus and used the obtained classifier to identify a list of terms and definitions for each publication in the ACL ARC.

Inspecting such output gives us an understanding of the properties of definition components, eventually helping the community to define better features to capture them, as well as intrinsically deepening our knowledge of the natural language of definitions and the structure of scientific discourse.

5.1 Demographics

From a term's perspective we can introspect properties of the enclosing paper, the host sentence, the term itself and its definition.

At the *document* level, we can analyze document metadata: its venue (journal, conference or workshop published) and year of publication.

At the *sentence* level, we analyze the position of the sentences that are definition sentences.

Focusing on *terms*, we want to find out in more detail the technical terminology that is defined. Are they different from general keyphrases? What type of entities are defined? What words do these terms consist of? What structures are common?

We are interested in analogous questions when focusing on the accompanying *definitions*. How many words or clauses do definition sentences consist of? Do we lose a lot of recall by restricting definitions to a single sentence? Are embedded definitions (definitions embedded with other definitions) common?

We highlight some specific findings from our basic analyses here:

Where do definitions occur? As terms are usually defined on first use, we expect the distribution of definition sentences to skew towards the beginning portions of a scientific document as input. We count the occurrences of definition sentences in each of ten equally-sized (by number of sentences) non-overlapping partitions. The results are shown in Figure 1, aligning with our intuition: The first three quantiles contribute almost 40% of all detected definition sentences, while the last three quantiles contain only 17.8%.

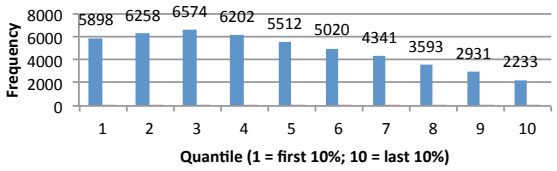
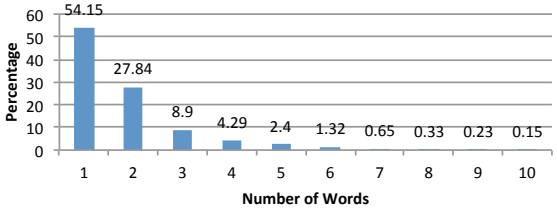
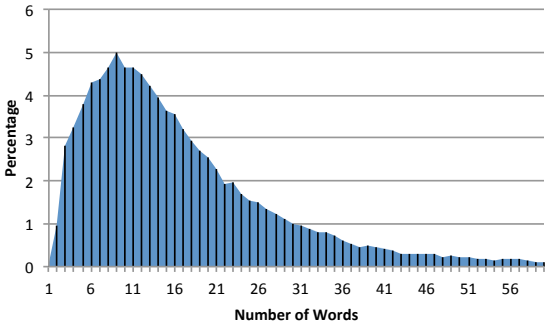


Figure 1: Occurrences of definitions within different segments of an article.

How long are the detected terms and definitions? Figure 2 shows the detected aggregate distributions. Over 54% of the detected terms are single tokens, where majority of the remaining 45% of terms being multi-word terms of six words or less. Among the single token terms, a further analysis reveals that 17.4% are detected as single-character variables, 34.9% are acronyms (consisting of more than one capitalized letters), while the remaining 47.7% are normal words. Definitions, in contrast, are longer and more varied, with a peak length of nine words. Slightly over half of the definitions have a length of 5–16 words; 75% have lengths between 3 and 23 words.



(t) Term length distribution.



(b) Definition length distribution.

Figure 2: Length distributions of (top) terms, (bottom) definitions.

In Table 5, we present the 10 most frequent POS tag bigrams for terms and definitions. We can see that among terms, a sequence of consecutive two nouns is most common, making up four out of the top five bigrams. We notice that determiners and prepositions are absent from the term list but are common in definitions.

5.2 Inspection of Definition over Time

The ACL ARC covers journal articles, conference and workshop proceedings over a few decades. As with other fields in recent years, the amount of computational linguistics literature has steadily increased over the number of years.

We study if definitions appear as frequently in different types of scientific articles (e.g. journals, conference papers or workshop papers). We also want to investigate if there is a significant shift in the distribution of definitions across years. In Figure 3, we present the density of definitions (defined as the

Term	Definition
NNP NNP	DT NN
NN NN	NNP NNP
NNP NN	NN IN
NN NNP	IN DT
JJ NN	NN NN
NN JJ	JJ NN
NNP :	NN :
: NNP	DT JJ
NN NNS	NNS IN
NN ”	NN NNS

Table 5: Most frequent POS bigrams for terms and definitions.

percentage of sentences that are identified as definition sentences), for these three different categories of publications[6].

In Figure 3, the three data series overlap each other, so we cannot conclude definitions appear more often in one type of papers than another. However, as a side effect, we see that while the definition density for journal papers remain relatively constant, for conference and workshop papers the number of definitions extracted per sentence has increased noticeably over time. The average number of definitions presented in conference papers, for instance, increased more than 100% in the 40 years represented in the ACL ARC.

The increasing number of definitions alone does not show that new knowledge is introduced at a faster rate, as definitions may be repeated. To control for this effect, we also need to know which definitions are new or defined in previous year(s). We studied this effect in more detail for the relatively smaller set of journal papers (Figure 4). For journal papers, the number of definitions of previously introduced terms in each year against the number of new definitions. We say a definition is new when the detected term was not identified in any article (not limited to journals) in previous years.

We see that the number of new terms being defined also increases with the years. But the increase is much slower than that for the total definitions. The area between the two lines denotes the definitions where the same term has been multiply defined from the same or previous years as the current year under investigation.

[6]The ACL ARC is organized by the venue of the publication, which is associated to a category. For the assigned category for each venue please refer to Appendix A.

5.3 Trends

We can use terms and definitions to also introspect how the computational linguistics literature has changed over time. Table 6 shows a subset of the most frequently defined terms in the ACL ARC, where we exclude single-character terms (“variables”).

WordNet (292)	Part Of Speech (45)
Precision (172)	Probablistic CFG (43)
Recall (167)	FrameNet (38)
Noun phrase (97)	Conditional Random Field (29)
Word sense disambiguation (60)	Inverse document frequency (28)
Support Vector Machine (60)	PropBank (27)
Hidden Markov Model (54)	Context Free Grammar (25)
Latent Semantic Analysis (57)	Accuracy (20)

Table 6: Subset of most frequently defined terms. Raw counts in parentheses. Variations of the same term (*e.g.* plurals, acronyms) are collapsed into one instance.

To be expected, these popular terms are mostly specific to computational linguistics. From our observation, we can fit these terms into one of three categories, including 1) resources (WordNet, FrameNet, PropBank), 2) methodologies (SVM, HMM, LSA), and 3) evaluation metrics (Precision, Recall, Accuracy). We feel that the final category of evaluation metrics is more general and would be shared among other scientific disciplines.

An interesting analysis that follows from this categorization is that we can study major trends and changes in the research directions of the community. This can help to draw the attention of researchers to emerging trends. We illustrate this approach in Figure 5 that focuses on three sequence labeling methodologies that have been used to address similar problems – namely, hidden Markov model (HMM), maximum entropy Markov model (MEMM), and conditional random fields (CRF) – during the period from 1989 to 2006 (where we have sufficient data points). From the early 90s, we see that HMM was a clear favorite. However since 2000, MEMM gained in popularity and use. Lafferty et al. (2001) introduced CRFs in 2001 and the new methodology was widely adopted soon after that.

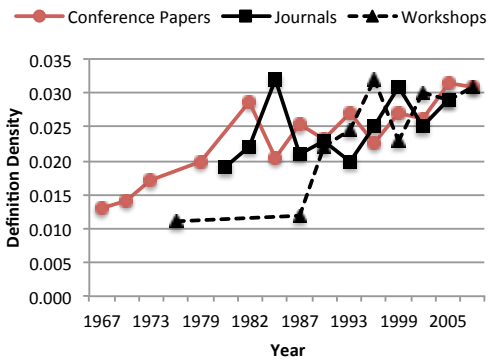


Figure 3: Occurrences of definitions across publication categories.

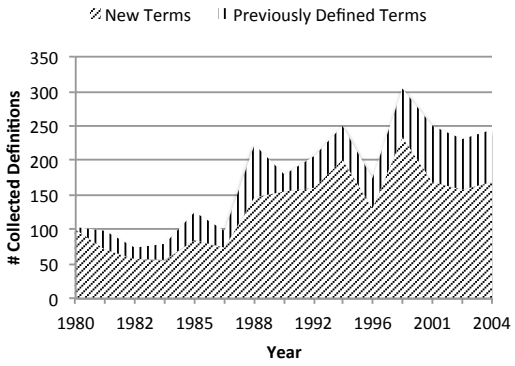


Figure 4: Relative proportions of new and recurring definitions in journal papers.

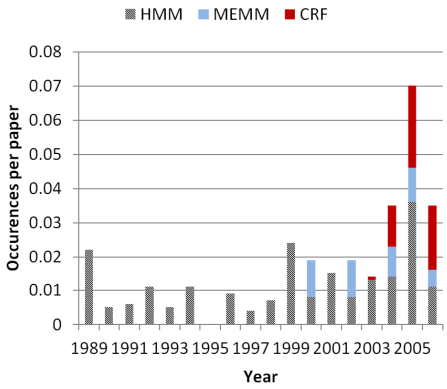


Figure 5: The occurrence of definitions for various sequence labeling methodologies over the years.

6 Conclusions and Future Work

We study the task of identifying definition sentences, as a two-part entity containing a term and its accompanying definition. Unlike previous work, we propose the harder task of delimiting the component term and definitions, which admits sequence labeling methodologies as a compatible solution.

Leveraging the current best practice of using conditional random fields, we contribute two additional ideas that lead to DefMiner, a state-of-the-art scholarly definition mining system. First, we show that shallow parsing and dependency parse features that may provide additional non-local information, are useful in improving task performance. Second, viewing the problem as two correlated subproblems of term and definition extraction, we measure the tightness and dependency of the correlation. We find that a two-stage sequential learning architecture (term first, definition second) leads to best performance. DefMiner outperforms the state of the art, and we feel is fit for macroscopic analysis of scientific corpora, despite significant noise.

We thus deployed DefMiner on the *ACL Anthology Reference Corpus*. We demonstrate how DefMiner can yield insights into both the structure and semantics of terms and definitions, in both static and diachronic modes. We think future work could pursue more in-depth analysis of the distributional and demographic properties of automatically extracted lexica. We can use the lexicon obtained from different years to carry out trend prediction, which we have illustrated here. Downstream systems may predict which term will become popular, or could alert an author if their definition of a term significantly differs from the original source.

We hope to tackle the annotation bottleneck in future work on definition extraction, common in many data-driven learning fields. We plan to explore iterative, semi-supervised methods to best manage human effort to maximize the effectiveness of future annotation.

In addition, with respect to modeling, although we showed that doing definition classification before term classification does not improve over our single-stage classifier, we hope to study whether suitable joint inference models can benefit from the interaction between the two classification processes.

References

Jason Baldridge. 2005. The opennlp project.

Steven Bird, Robert Dale, Bonnie J. Dorr, Bryan Gibson, Mark T. Joseph, Min-Yen Kan, Dongwon Lee, Brett Powley, Dragomir R. Radev, and Yee Fan Tan. 2008. The ACL anthology reference corpus: A reference dataset for bibliographic research in computational linguistics. In *Proceedings of the 6th International Conference on Language Resources and Evaluation Conference (LREC08)*, pages 1755–1759, Marrakech, Morocco.

Claudia Borg, Mike Rosner, and Gordon Pace. 2009. Evolutionary algorithms for definition extraction. In *Proceedings of the 1st Workshop on Definition Extraction*, WDE '09, pages 26–32, Stroudsburg, PA, USA. Association for Computational Linguistics.

Daniel Cer, Marie-Catherine de Marneffe, Daniel Jurafsky, and Christopher D. Manning. 2010. Parsing to stanford dependencies: Trade-offs between speed and accuracy. In *7th International Conference on Language Resources and Evaluation (LREC 2010)*, Valletta, Malta.

Ismail Fahmi and Gosse Bouma. 2006. Learning to identify definitions using syntactic features. In *Proceedings of the EACL workshop on Learning Structured Information in Natural Language Applications*, Trento, Italy.

Taku Kudo. 2005. Crf++: Yet another crf toolkit. *Software available at http://crfpp. sourceforge. net*.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, CA, USA.

Smaranda Muresan and Judith Klavans. 2002. A method for automatically building and evaluating dictionary resources. In *Proceedings of the Language Resources and Evaluation Conference (LREC)*, Las Palmas, Canary Islands, Spain.

Roberto Navigli and Paola Velardi. 2010. Learning word-class lattices for definition and hypernym extraction. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1318–1327, Uppsala, Sweden.

Thuy Dung Nguyen and Min-Yen Kan. 2007. Keyphrase extraction in scientific publications. In *Proceedings of International Conference on Asian Digital Libraries*, pages 317–326, Hanoi, Vietnam.

Jennifer Pearson. 1996. The expression of definitions in specialised texts: a corpus-based analysis. In *Proceedings of Euralex 96*.

Melanie Reiplinger, Ulrich Schafer, and Magdalena Wolska. 2012. Extracting glossary sentences from scholarly articles: a comparative evaluation of pattern bootstrapping and deep analysis. In *Proceedings of the ACL-2012 Special Workshop on Rediscovering 50 Years of Discoveries*, ACL '12, pages 55–65, Stroudsburg, PA, USA. Association for Computational Linguistics.

Eline Westerhout and Paola Monachesi. 2007. Extraction of dutch definitory contexts for elearning purposes. In *Proceedings of Computational Linguistics (CLIN 2007)*.

Eline Westerhout. 2009. Definition extraction using linguistic and structural features. In *Proceedings of the 1st Workshop on Definition Extraction*, WDE '09, pages 61–67, Stroudsburg, PA, USA. Association for Computational Linguistics.

Ian H Witten, Gordon W Paynter, Eibe Frank, Carl Gutwin, and Craig G Nevill-Manning. 1999. Kea : Practical automatic keyphrase extraction. *Computer*, pp:254–255.