Mining Algorithm Roadmap in Scientific Publications

Hanwen Zha, Wenhu Chen, Keqian Li, Xifeng Yan
University of California, Santa Barbara
{hwzha,wenhuchen,klee,xyan}@cs.ucsb.edu

ABSTRACT

The number of scientific publications is ever increasing. The long time to digest a scientific paper posts great challenges on the number of papers people can read, which impedes a quick grasp of major activities in new research areas especially for intelligence analysts and novice researchers. To accelerate such a process, we first define a new problem called *mining algorithm roadmap* in scientific publications, and then propose a new weakly supervised method to build the roadmap. The *algorithm roadmap* describes evolutionary relation between different algorithms, and sketches the undergoing research and the dynamics of the area. It is a tool for analysts and researchers to locate the successors and families of algorithms when analyzing and surveying a research field. We first propose abbreviated words as candidates for algorithms and then use tables as weak supervision to extract these candidates and labels. Next we propose a new method called Cross-sentence Attention NeTwork for cOmparative Relation (CANTOR) to extract comparative algorithms from text. Finally, we derive order for individual algorithm pairs with time and frequency to construct the *algorithm roadmap*. Through comprehensive experiments, our proposed algorithm shows its superiority over the baseline methods on the proposed task.

KEYWORDS

Relation Extraction, Taxonomy Construction, Knowledge Base Construction

ACM Reference Format:
Hanwen Zha, Wenhu Chen, Keqian Li, Xifeng Yan. 2019. Mining Algorithm Roadmap in Scientific Publications. In *The 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '19), August 4–8, 2019, Anchorage, AK, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3292500.3330913

1 INTRODUCTION

The number of scientific publications is ever increasing. According to the prominent STM report [38], the number of journal articles published in 2014 alone approached 2.5 million and this number is still increasing on a yearly basis. The long time to digest a scientific paper posts great challenges on the number of papers a researcher can digest. Experienced researchers may be familiar to identify the demanded papers. However, the problem becomes much severe for intelligence analysts who need to browse papers and quickly grasp the major activities in new research areas. The novice researchers may have a similar obstacle in finding out papers related to their research. They usually take plenty of time to come up with keywords, retrieve and read relevant papers and iterate this process.

One step assisting with this process is taxonomy construction [11, 14, 32, 42], which extracts concepts from a collection of documents and builds a tree structure to describe the hierarchical relation between different concepts. Analysts and researchers can follow this concept hierarchy to quickly identify more desired keywords or documents. However, previous taxonomy construction methods mostly focus on isA relation. They either rely on pattern-based methods [14, 32] which extract hierarchical relation leveraging linguistic features, or clustering-based methods [11, 42], which cluster concepts to induce an implicit hierarchy.

In this paper, we generate a graph called *Algorithm Roadmap*, focusing on a special type of concept – "algorithms", and its specific form – "abbreviations". Given a scientific corpus, we mine comparative algorithms (described in section 3), and construct a graph connecting mined algorithms. In Figure 1, for example, a roadmap for algorithm Generative Adversarial Network (GAN) [9], describes its successors and competitors in the scientific literature. The generated *algorithm roadmap* captures the development of algorithms, sketches the undergoing research, and models the dynamics of an area. It serves as a tool for analysts and researchers to locate the successors and families of algorithms when doing analysis and survey.

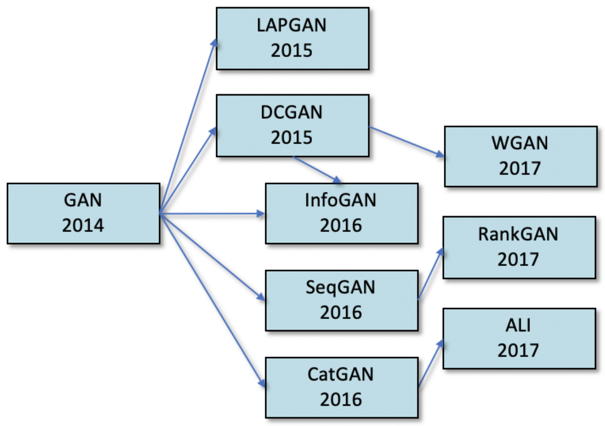


Figure 1: A pedagogical example of the *algorithm roadmap* for "GAN" algorithm.

Conclusively, there exist three major challenges for mining the *algorithm roadmap* in the scientific literature, corresponding to the label, entity, and relation respectively.

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. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
KDD '19, August 4–8, 2019, Anchorage, AK, USA
© 2019 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-6201-6/19/08...$15.00
https://doi.org/10.1145/3292500.3330913

Relation Type	Instance
Single Sentence	We train models using different GAN methods : WGAN-GP , WGAN with weight clipping and DCGAN.
Single Sentence	In almost all experiments BayesGAN outperforms DCGAN and W-DCGAN.
Cross Sentence	LapGAN proposed a Laplacian pyramid implementation of GANs. ... DCGAN used a deeper convolutional network.
Cross Sentence	GDL focuses on unsupervised learning. ... GAN and DCGAN show results for unsupervised learning and semi-supervised classification .

Table 1: Examples of comparative algorithms.

Label Scarcity: Collecting in-domain algorithm entities and relation labels in scientific publications are prohibitively expensive. Existing datasets or curated in-domain knowledge bases [1, 4] are rather small and frequently outdated with the development of science. Moreover, a newly invented algorithm probably only appears in a single paper. This scarcity raises a challenge for supervised and distantly supervised entity extraction methods like [18, 19] or weakly supervised phrase extraction approaches relying on frequency [30]. The low coverage of knowledge base can also influence the availability of relation labels when using distant supervision [22].

Entity: General entity recognition does not directly separate the algorithms with the others. Though using abbreviations as the representation of algorithms alleviates the problem of considering all types of entities, few types other than algorithm exists. In addition, the abbreviation, as a short form of text, is prone to ambiguity. Word sense disambiguation methods [23] have been studied to disambiguate word senses, however, deciding the sense for the abbreviation in the scientific domain is still challenging when lack of labeled data.

Relation: The narrations of two comparative algorithms either lie in a single sentence or are distributed across sentences. For example, in Figure 1, the comparative relation may be described in one sentence, e.g., "Algorithm A outperforms Algorithm B ...," or in multiple-sentences, e.g., "Algorithm A ... ; Algorithm B ...." Moreover, it is likely more than two algorithms are compared or more than two abbreviations appear in a paragraph. Additional abbreviations may convey a meaning related to comparative relation. Unsupervised pattern-based methods such as [14] focus on isA relation, which are not suitable for finding compared algorithms. Most existing researches for the supervised relation extraction focus on single sentence relation extraction with an exception of [25, 37], which focus on general documents while not targeting on a specific narration of algorithm abbreviations and comparative relation. On the other hand, these supervised approaches require annotated corpora.

We propose a framework to mine the *algorithm roadmap* in scientific publications to tackle the previous raised challenges. It first extracts abbreviations with specific pattern as algorithm candidates. Then it leverages weak supervision from tables and text to create training data for comparative relation identification and entity typing. Next, it applies our proposed relation extraction method *Cross-sentence Attention NeTwork for cOmparative Relation* (CANTOR) to extract comparative algorithms in the text. It leverages words and abbreviations in the context, and jointly predicts the candidate types for addressing ambiguity during the roadmap construction. Finally, it connects the compared algorithms into a graph with time and frequency information.

Extensive experiments on three real-world datasets demonstrate our superior performance in finding the comparative relation. Our CANTOR model outperforms supervised and unsupervised baseline methods by a large margin. We perform case studies on the constructed *algorithm roadmaps* to further visualize the effectiveness of the construction.

2 RELATED WORKS

Knowledge base construction is a known technique for harvesting knowledge and storing facts in a structured format. The constructed knowledge base plays an important role in downstream applications such as information retrieval, question answering, and document analysis, etc. Most existing automatically constructed knowledge bases focus on general domain, which either extract facts from Wikipedia info-boxes [3, 34] or harvest knowledge with specific linguistic patterns [5, 39]. Taxonomy can be viewed as a tree-structure knowledge graph, where linked nodes have hierarchical relation. Plenty of methods have been proposed to extract these hierarchical relation, either leveraging linguistic patterns [14] or hierarchical clustering of concepts which implicitly captures the hierarchical relation [42]. These methods mainly focus on the general domain, harvesting common knowledge with pattern or statistics.

Many works focus on for mining scientific publications, for example, [1] proposed a keyphrase and relation extraction competition for scientific publications, [4] collected a dataset for scientific taxonomy construction, [13] studied the evolution of scientific topics through dynamic topic models [21] modeling implicit topics and obscure relations, and some technical reports [1] [2] manually analyzed the development of areas such as artificial intelligence. Some of these works collected datasets for scientific publications, but the process is known to be expensive and the collected datasets are normally small in size.

Word sense disambiguation [23] is a type of technique used to distinguish ambiguous word senses. They either disambiguate word senses with a sense inventory or distinguish super senses by clustering words. Inspired by methods using super senses, we use types as evidence to distinguish abbreviations. To leverage the constraint of abbreviations, we use predefined types as super senses for abbreviations.

Another line of work related to ours is relation extraction, which has attracted much attention from the community, while most of the works focus on news and web data [8, 29]. Recent neural

[1]https://www.technologyreview.com/s/612768/we-analyzed-16625-papers-to-figure-out-where-ai-is-headed-next/
[2]https://aiindex.org/

network based methods have achieved great success in relation extraction, including CNN-based approaches [40, 41] and LSTM-based approaches [31]. These approaches all consider relations lying in a single sentence. On the other hand, most relation extraction works assume entities and relation sets are given in the datasets, while others apply distant supervision to link entity mentions [22, 28] in the text to the knowledge base entities [19] and acquire relation labels. Their weaknesses lie in the fact that they require either annotated corpora or well-covered knowledge bases.

Beside single-sentence relation approaches, some previous works exist on cross-sentence relation extraction. [26] proposes to construct cross-sentence relation data for entities with minimal-span assumption. [25] proposes to use a Graph-LSTM to encode the shortest path in the extracted dependency parse tree, where the tree roots of different sentences are linked together. [37] proposes a method using self-attention [36] and bi-affine scoring algorithm to predict biological relations between all mention pairs in the abstract simultaneously. Our work differs from them in three key ways. First, we leverage weak supervision from the paper rather than using annotated corpus or distant supervision from an external knowledge base. Second, we consider typing of entities for abbreviation ambiguity and roadmap construction. Third, we model both single-sentence and cross-sentence comparative relations with words and abbreviations in the context.

3 PRELIMINARIES

In this paper, we mainly target at mining *algorithm roadmap* in scientific publications. In order to provide a better understanding of our paper, we first give definitions related to *algorithm roadmap* and then briefly overview our proposed method.

Algorithm Roadmap. It is a directed acyclic graph $\mathcal{G}$, where each node of the graph is an algorithm term in abbreviation form. Each directed edge $e_1 \rightarrow e_2$ in the graph $\mathcal{G}$ represents a directed evolutionary relation between two algorithm nodes e_1 and e_2. For example, in the computer science domain, there are algorithms such as GAN (Generative Adversarial Networks) [9] and DCGAN (Deep Convolutional Generative Adversarial Networks) [27]. A directed edge $GAN \rightarrow DCGAN$ represents "DCGAN" is a successor and is evolved from "GAN".

Comparative Relation. It is a relation between two algorithms, which means two terms were compared with each other in some papers. For example, pair $(GAN, DCGAN)$ having comparative relation means "DCGAN" was compared to "GAN" in some papers, but there is no direction information implies which technique is a successor.

Roadmap Construction. We are the first to mine algorithm pairs with comparative relation using weak supervision harvested from tables and texts. Moreover, we connect the compared algorithms into a directed graph $\mathcal{G}$ by deriving order with time and frequency information.

4 EXTRACTING COMPARATIVE RELATION

In this section, we present a framework to extract comparative algorithm pairs from papers. The framework consists of three steps: i) Extracting abbreviations as algorithm candidate mentions; ii) Applying weak supervision from tables and texts to create training data for comparative relation and typing; iii) Learning to predict the relation for candidate mention pairs.

4.1 Candidate Mention Extraction

We use abbreviations as our algorithm candidates. The intuition of using abbreviations as algorithm candidates lies in two folds: entity and relation label availability.

Lack of annotated corpus and well-covered in-domain knowledge bases, general entity recognition methods [18, 19] do not fit our candidate mention extraction. With low occurrence frequency, phrase extraction approaches do not satisfy the job as well.

We observed that abbreviation is a commonly used representation of algorithm terms. With a unified form, it is easy to harvest from the corpus. More importantly, using abbreviations as candidates provides a possibility to gather supervision from tables for comparative relation, which we will show in section 4.3.

Abbreviations follow specific patterns and may refer to several types of meanings. For example, Table 2 shows algorithms such as CNN (Convolutional Neural Network), datasets such as MNIST (Modified National Institute of Standards and Technology dataset), and metrics such as AUC (Area under curve). Types of a few abbreviations can be distinguished by checking the signal words following the abbreviation. For example, an algorithm abbreviation may be followed by algorithm, method, model etc. in the text.

We use regular expression with pattern consists of *capital letters*, *lowercase letters*, *numbers*, and *hypen*, to unsupervisedly harvest abbreviations as algorithm mention candidates from the text. We extract type of a few abbreviations identified by signal words to provide weak supervision for entity typing in section 4.2.5, unidentified abbreviations are randomly sampled as type *Others*.

Type	Abbreviations	Signal Word
Algorithm	CNN, LSTM, GAN	algorithm
Metric	AUC, MAP, MAE	metric
Dataset	MNIST, CIFAR10, SQuAD	dataset
Others	NP, VP, POS	/

Table 2: An example of different types of abbreviations.

4.2 Cross-Sentence Relation Extraction

We designed our model to incorporate both single-sentence and cross-sentence information, and consider all abbreviations in a paragraph. To this end, our model consists of a single-sentence module with Piecewise CNN [40], and a cross-sentence module which leverages self-attention to attend to all words capturing the paragraph-level relation information, and abbreviation-attention to attend to all abbreviations helping describe the relation of the candidate pair. Moreover, typing is jointly done on the attended candidates to assist downstream roadmap construction. Mention pair predictions are pooled on single-sentence module and cross-sentence module for the entity pair prediction. Finally, the predictions of the two modules are interpolated with weights learned simultaneously with other parameters.

4.2.1 Inputs. Both the single-sentence module and the cross-sentence module take a sequence of N token embeddings in $\mathbb{R}^d$. The input

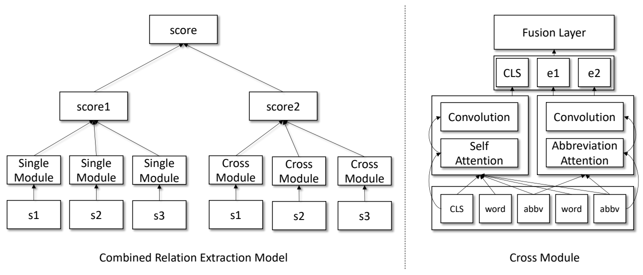


Figure 2: Architecture of our CANTOR model.

embedding of each token is x_i, which is a concatenation of word embedding and positional embedding [40].

4.2.2 Single-Sentence Module. We use PCNN (piecewise convolutional neural networks) [40] as our single-sentence relation extractor, which is a well-performed model for short-context relation extraction.

PCNN is a variation of CNN that adopts piecewise max pooling in relation extraction. It divides the sentence into three segments: part before first entity, part between two entities and part after second entity. Thus each convolutional filter q_i is divided into three segments (q_{i1}, q_{i2}, q_{i3}). The max-pooling is performed on three segments separately, which is defined as

$$p_{ij} = max(q_{ij}) \quad 1 \le i \le n, \quad 1 \le j \le 3 \tag{1}$$

where n is the number of convolutional filters, and p_i is equal to the concatenation of p_{ij} over all segments j, which aggregates information from different parts. A non-linear layer is added on top of the sentence relational encoding which is represented by all filters $p_{1:r}$, to get the relation prediction:

$$o_1 = W_1 tanh(p_{1:r}) + b_1. \tag{2}$$

4.2.3 Cross-Sentence Module. Our cross-sentence module focuses on finding paragraph-level comparative relation, where two algorithm mention candidates lie across sentences. We base on recent Transformer architecture [7, 36] to build this module, due to its better performance in encoding long-distance context compared to Long Short Term Memory Networks (LSTMs) [15] and Convolutional neural networks (CNNs).

Self-Attention. We adapt Transformer [36] to encode word sequences in a paragraph, where we calculate the self-attention of the words, and use a convolutional layer in self-attention blocks similar to [37] to alleviate the burden on the model to attend to local features. We add residual connections [12] to both multi-head attention and convolutional layers. The Transformer contains stacked layers of Transformer block, which contains its own set of parameters. The token embedding $X = \{x_1, ..., x_N\}$ is fed to the first-layer transformer block and the output of the kth-layer block $\widehat{A}_k$ is calculated by

$$\widehat{A}_k = A_k + A_k \times softmax(\frac{A_k^T A_k}{\sqrt{d_{Ak}}}) \tag{3}$$

where $softmax(\cdot)$ is a column-wise normalizing function, and d_{Ak} is the dimension of the input token embedding of kth transformer block used for self-attention. A convolutional layer $Conv$ with residual connection follows the self-attention layer:

$$H_{Ak} = \widehat{A}_k + Conv(\widehat{A}_k). \tag{4}$$

We follow BERT [7] which recently achieves great success in multiple NLP tasks, to add a special <CLS> token at the start of the paragraph and a special <SEP> token at the end of each sentence in the paragraph. The representation of <CLS> is used for gathering relational information in a paragraph. With self-attention layer, all other tokens in a paragraph attend to this <CLS> token. <SEP> is a special token stands for the end of a sentence, which is used to incorporate the sentence boundary information in the model.

Abbreviation-Attention. The abbreviation-attention layer calculates attention over all abbreviations in the sentences. When additional algorithms are also compared or share a similar relation to two candidates, two candidate mentions may have a high probability to be comparative.

Different from the self-attention mechanism, the abbreviation attention is calculated based on all abbreviations in a paragraph. Denoting all token embeddings of abbreviations as B, transformer blocks with a new set of parameters are applied. Similar to self-attention, with kth-layer input embedding B_k, the kth-layer output of abbreviation attention $\widehat{B}_k$ is calculated as

$$\widehat{B}_k = B_k + B_k \times softmax(\frac{B_k^T B_k}{\sqrt{d_{Bk}}}). \tag{5}$$

Similarly a convolutional layer with residual connection is applied to the output of abbreviation-attention layer:

$$H_{Bk} = \widehat{B}_k + Conv(\widehat{B}_k). \tag{6}$$

With abbreviation-attention layer, all the abbreviations in the sentences are attending to the algorithm candidates. The final output $H_{Bk,e1}$ and $H_{Bk,e2}$ of the algorithm candidates in H_{Bk} are selected as the entity representation, which fuses all abbreviation information in the paragraph.

Character Embedding. Some of the abbreviations are rarely mentioned in the text, which may result in an insufficiently trained word embedding. Since the abbreviation is often created by summarized text, similar abbreviations probably imply overlapped word sequences. To leverage this intuition, we use character embedding that describes character-level information of abbreviations and we apply a character-level convolutional layer followed by a max-pooling layer to get a character-level abbreviation representation.

For an abbreviation with corresponding character embedding sequences $C =< c_1, c_2, ..., c_n >$, we apply a convolution kernel followed by a max-pooling layer.

$$H_c = max(Conv(c_i)) \quad 1 \le i \le n \tag{7}$$

Fusion Layer. Finally, We use a single layer on top of the encoded paragraph representation $H_{Ak,<CLS>}$ and the algorithm candidate representation E to model their interaction. The candidate representation E is constructed by concatenating original word embedding X_{e1}, X_{e2}, character embedding $H_{c,e1}, H_{c,e2}$, attended abbreviation embedding $H_{Bk,e1}, H_{Bk,e2}$. The final fusion layer predicts a final relational score for one instance.

$$E = [X_{e1}, X_{e2}, H_{c,e1}, H_{c,e2}, H_{Bk,e1}, H_{Bk,e2}] \tag{8}$$

$$o_2 = W_2([H_{Ak,<CLS>}, E]) + b_2 \tag{9}$$

4.2.4 Combined Relation Extraction. Predicting whether an algorithm candidate pair is compared forms a multi-instance learning problem [29, 35]. For each pair, a bag of instances may contain two candidates. The entity-level prediction is an aggregation over multiple mention pair instances. Based on different assumptions, different weighting strategies have been proposed such as max-pooling [35] and selective attention [17].

We follow at-least-one assumption, where a positive example has at least one instance implies the comparative relation, and use max-pooling to select the instance with the maximum score for an entity pair in both single-sentence and cross-sentence module.

The final score of an algorithm candidate pair (e_1, e_2) is a interpolation of the aggregated prediction score $O_1(z|S)$ of the single-sentence module and $O_2(z|S)$ of the cross-sentence module. The trainable weights λ_1 and λ_2 are jointly learned from the data to reflect the importance of single-sentence and cross-sentence part. The weights are limited to be positive and have a total sum of 1.

$$\begin{aligned} O(z|S) &= \lambda_1 O_1(z|S) + \lambda_2 O_2(z|S) \\ \lambda_1, \lambda_2 &> 0, \lambda_1 + \lambda_2 = 1 \end{aligned} \tag{10}$$

Finally, we use softmax to normalize the scores to get a probability distribution $p_z = softmax(O(z|S))$, and relation prediction loss is defined as a cross-entropy loss: $L_{RE} = -\sum_{i=1}^{2} y_i \cdot log(p_{z,i})$, where each $y_i \in \{0, 1\}$ indicates algorithm candidate pair relation is true for which class (without/with relation).

4.2.5 Entity Typing. Previous relation extraction modules do not distinguish the types of the abbreviations. Few types other than algorithm exists, even though using abbreviations as algorithm candidates addresses the problem of candidate recognition. In addition, introducing abbreviations may increase chance of ambiguity. For example, "GAN" could be an algorithm (Generative Adversarial Network) but also a gene in biology. "CNN" could be an algorithm Convolutional Neural Network but also a television channel (Cable News Network).

Inspired by word sense disambiguation methods that label super sense types for word clusters [23], we jointly predict the types for abbreviation candidates with relation extraction task to distinguish abbreviations for downstream roadmap construction. Considering the limited types of abbreviation, we pre-define a fixed type inventory instead of using clustering and labeling word clusters.

We use a projection matrix W_3 on top of the attended algorithm candidate representation after abbreviation attention to predict the type of the candidate abbreviation, and the scores are normalized with softmax function: $p_t = softmax(W_t H_{Bke})$.

The type prediction loss also applies the cross-entropy loss: $L_{TP} = -\sum_{i=1}^{T} y_{t,i} \cdot log(p_{t,i})$, where there are total T types and each $y_{ti} \in \{0, 1\}$ indicates the correctness for ith type.

We add a type constraint to the loss function, considering that a comparative relation only holds for candidates with the same type. For a compared algorithm candidate pair $e1, e2$ in the ground truth, a type constraint loss is defined as a kl-divergence of two predicted types, where $L_{TC} = D_{KL}(p_{t,e1}, p_{t,e2})$.

The final score is a weighted sum of all the loss functions, with weights as hyper parameters.

$$L = \gamma_1 \cdot L_{RE} + \gamma_2 \cdot L_{TP} + \gamma_3 \cdot L_{TC} \tag{11}$$

4.3 Weakly Supervised Training Data

The labels for comparative relation is hardly available from existing datasets and curated knowledge bases. We propose a weakly supervised approach based on our observation that in the same row or same column of the table, mentioned abbreviations are often comparative, including compared algorithms, datasets, or metrics etc. This gives us an opportunity to create positive training examples from the table without human effort.

We first used a table parsing tool [6] [3] to extract tables from raw pdf files of papers. Then we processed the parsed results to identify abbreviations in the same row or column. We enumerated and labeled aligned abbreviation pairs as positive examples with comparative relation. The supervision from the table gives compared abbreviations of various types.

We randomly sample other non-positive candidate pairs as negative examples in training. To reduce the huge number of unrelated and non-informative negative examples, we follow the minimum-span strategy in [26], and limit sampled negative candidate pairs to the co-occurred pairs shown in a limited length of continuous sentences. Intuitively, most compared algorithms are kept since authors tend to describe compared algorithms coherently in a short paragraph.

5 GENERATING ALGORITHM ROADMAP

Previous comparative relation extraction step produces a large set of compared abbreviation pairs, and each pair corresponds to an undirected edge in *algorithm roadmap.* Our goal is to derive direction for the edge and connecting individual pairs.

The evolutionary relation has a strong association with the comparative relation. The publication time is a strong indicator of evolution direction for compared algorithms. We use first occurrence time in the corpus of an abbreviation as an approximation. For those pairs identified with the same occurrence time, we expect usually low frequent algorithm is evolved from high frequent one.

Algorithm 1 Algorithm Roadmap Construction

```
Input: Comparative algorithm list $P$, time dictionary $T$, and frequency dictionary $F$
Output: Algorithm roadmap $\mathcal{G}$
1: for all $(e1, e2) \in P$ do
2:     if $T[e1] < T[e2]$ or ($T[e1] = T[e2]$ and $F[e1] > F[e2]$) then
3:         $\mathcal{G}.add(e1 \rightarrow e2)$
4:     else
5:         $\mathcal{G}.add(e2 \rightarrow e1)$
6: return $\mathcal{G}$
```

Example. Pairs "GAN" and "DCGAN" are mined compared algorithms. We locate their first appearance time, and find that "GAN" was published first in 2014, and "DCGAN" was first published in

[3]https://github.com/allenai/pdffigures

2015. *GAN*(2014) $\rightarrow$ *DCGAN*(2015) is predicted as the direction where "DCGAN" is a successor.

When connecting individual pairs, only candidates above certain probability threshold are kept. In addition, candidates with different types in different pairs except for type *Other* are considered as separated nodes for roadmap construction.

6 EXPERIMENT

The following section is organized in this way, first we describe datasets and implementation details, second, we show held-out and manual evaluation results of different methods in comparative relation extraction task, third, we perform case studies to visualize the constructed *algorithm roadmaps*.

6.1 Dataset

We crawled papers from scientific conferences in domains including machine learning, natural language processing, and database. The corpora include papers in NeurIPS/NIPS (Annual Conference on Neural Information Processing Systems) from 1987 to 2017 [4], ACL (Annual Meeting of the Association for Computational Linguistics) from 1974 to 2017 [5], and VLDB (The Proceedings of the VLDB Endowment) from 2008 to 2017 [6]. The statistics of each of datasets is shown in Table 3.

Dataset	documents	sentences	positive pairs	abbreviations
NeurIPS	7k	3144k	9k	66k
ACL	5k	2277k	22k	71k
VLDB	2k	1289k	5k	40k

Table 3: Dataset statistics of NeurIPS, ACL, VLDB dataset.

From these datasets, we extract algorithm candidate mentions, apply weak supervision to extract types from texts and comparative relation labels from tables as described in section 4. We split train and test data with a ratio of 80% and 20%. Among training data, 10% is separated as validation data. Additional implementation details are included in Appendix A.

6.2 Results

We conduct both held-out evaluation and manual evaluation on our method and several baseline methods in the task of comparative relation extraction, where models predict whether given candidate pairs are comparative or not. Evaluated methods can be divided to unsupervised methods including co-occurrence based methods [10], word-similarity based methods [20], and supervised relation extraction methods [40]. The pattern-based method [14] is not compared due to its low recall in our task.

Test set from weakly supervised table data is used for held-out evaluation. The evaluation is harsh due to the limited number of positive examples, and noisy because of few table parsing errors. In the manual evaluation, for each supervised method, we randomly sample 100 examples from their positive predictions and ground truth positive set in test data, and combine those into a unified manual test set. We let human annotators to label the pairs, where we do not distinguish compared algorithms, datasets or metrics following the criteria of our weak supervision. In the following we introduce evaluated methods in detail.

PCNN_single: Piecewise CNN model [40], which is one of the state of art single-sentence relation extraction methods. PCNN_single only uses single-sentence instances for candidate pairs.

PCNN_cross: The same PCNN model as PCNN_single where cross-sentence instances are also used.

Sent_cooccur: A method similar to co-occurrence method used in hypernym detection [10]. Sent_cooccur calculates the co-occurrence frequency of candidate pairs in one sentence. A threshold that decides a positive-negative ratio closest to the ground-truth test data is used.

Doc_cooccur: Similar to Sent_cooccur, where the co-occurrence frequency in one document is used instead.

Word_similarity: A method predicts comparative relation score based on word embedding similarity, where the embedding is pre-trained with the Skip-Gram model [20] implemented in Gensim [7] for each corpus. The threshold is decided similarly to Sent_cooccur.

CANTOR: Our proposed cross-sentence relation extraction method, which considers both single-sentence and cross-sentence instances, all abbreviations in the context, and jointly typing the candidates.

Method	NeurIPS		ACL		VLDB	
	AUC	F1	AUC	F1	AUC	F1
sent_cooccur	0.68	0.71	0.66	0.71	0.70	0.67
doc_cooccur	0.57	0.69	0.46	0.68	0.62	0.68
word_similarity	0.62	0.70	0.67	0.72	0.66	0.72
PCNN_single	0.73	0.71	0.72	0.68	0.78	0.67
PCNN_cross	0.75	0.71	0.76	0.74	0.82	0.76
CANTOR (ours)	**0.82**	**0.74**	**0.79**	**0.78**	**0.85**	**0.78**

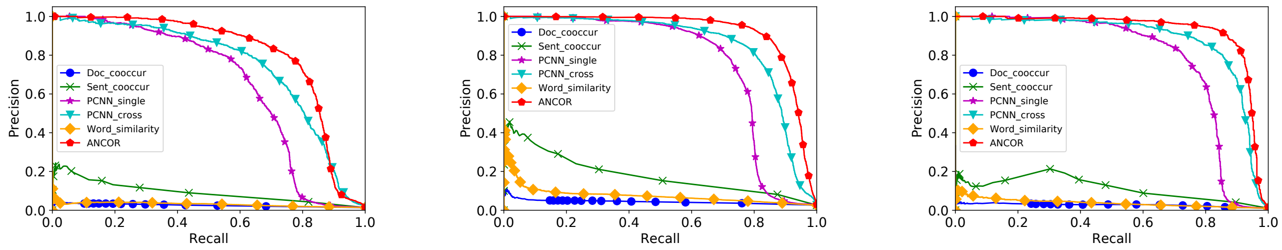
Table 4: Manual evaluation of different relation extraction methods for finding comparative relation on NeurIPS, ACL and VLDB datasets.

Figure 3 shows the held-out evaluation and Table 4 shows the manual evaluation for all different methods. For held-out evaluation, we draw the precision-recall curve of all methods, and for manual evaluation we calculate the weighted macro F1 and AUC(Area under the ROC Curve). AUC depicts the ranking correctness, where F1 does not take the rank into account.

Overall, due to a limited number of positive examples from weak supervision, the unsupervised methods show a low precision on the held-out evaluation. The manual test set looses the strict condition of positive examples, moreover, its construction filters most negative examples from the unsupervised methods. These two result in increased performance of unsupervised methods. However, neither co-occurrence based model or similarity is good at modeling the ranking of comparative pairs, thus result in a low AUC.

Co-occurrence is one indicator for comparative relation of abbreviations with good recall while suffering from low precision. This is because counting co-occurrence introduces non-comparative abbreviations into the results. Sentence-level co-occurrence model

[4]https://nips.cc/
[5]http://aclweb.org/anthology/
[6]https://www.vldb.org/pvldb/
[7]https://radimrehurek.com/gensim/



(a) NeurIPS (b) ACL (c) VLDB

Figure 3: Precision-Recall curve of different relation extraction methods for finding comparative relation on NeurIPS, ACL and VLDB dataset with held-out evaluation.

has a better performance than the document-level model since compared candidates are more likely to appear in a short context. Word similarity model performs between two co-occurrence methods. The word embedding captures the context of type rather than comparative relation. On the other hand, a large number of candidates are rarely mentioned, which leads to insufficient training of word embedding.

The supervised relation extraction methods generally outperform the unsupervised methods. The relation extraction model PCNN_single that uses single-sentence works well, but its precision drops rapidly when recall increases. PCNN_cross considering cross-sentence instances further improves the performance of the model, which shows the importance of cross-sentence instances in finding comparative relation. Our CANTOR method outperforms all these methods, which implies better modeling of cross-sentence comparative relation.

Ablation Study We do an ablation study to show the performance of different components. We use the manual test data in NeurIPS dataset collected by random samples from positive predictions and held-out positive examples from supervised methods to evaluate the components. As shown in Table 5, stacking self-attention, abbreviation-attention, typing and combined modeling improves the model performance.

Method	AUC	F1
Self-Attention	0.77	0.72
+Abbreviation-Attention	0.78	0.72
+Tpying	0.78	0.74
CANTOR (ours)	**0.82**	**0.74**

Table 5: Abalation study on different components in NeurIPS dataset.

6.3 Case Study

For each dataset, we mine compared algorithms from the entire corpus with our trained CANTOR model and connect the individual pairs with the approach described in section 5. In Figure 4, we show parts of the *algorithm roadmaps* constructed from different datasets. In each figure, each node contains its abbreviation name and its first occurrence time described in section 5 in its dataset. To be noticed, this time is not necessarily equal to the first publication time, as the algorithm is non-necessarily published in this conference.

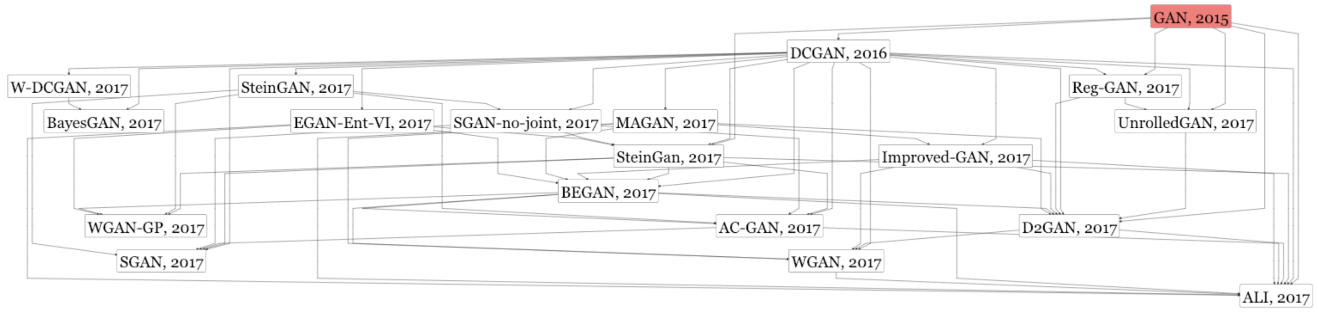
"GAN" (Generative Adversarial Networks) is a deep generative model [9], which has been extensively cited since proposed. Researchers even maintain a "GAN zoo" [8] to keep track of various kinds of "GAN" successors.

In NeurIPS dataset, our method mines its direct successors such as "DCGAN," "SteinGAN," "UnrolledGAN," "Reg-GAN" and "ALI." Then we keep identifying the successors of each successor. For example, "DCGAN" has successors including "W-DCGAN," "SteinGAN," and "Improved-GAN" etc. The comparison of our mined algorithms with algorithms in "GAN zoo" reveals a good precision in found successors. Our current method does not distinguish different forms of an abbreviation, thus "SteinGAN" and "SteinGan" are viewed as separated candidates. A minimum confidence score threshold can be used to control each level of the roadmap to trade off the precision and recall.

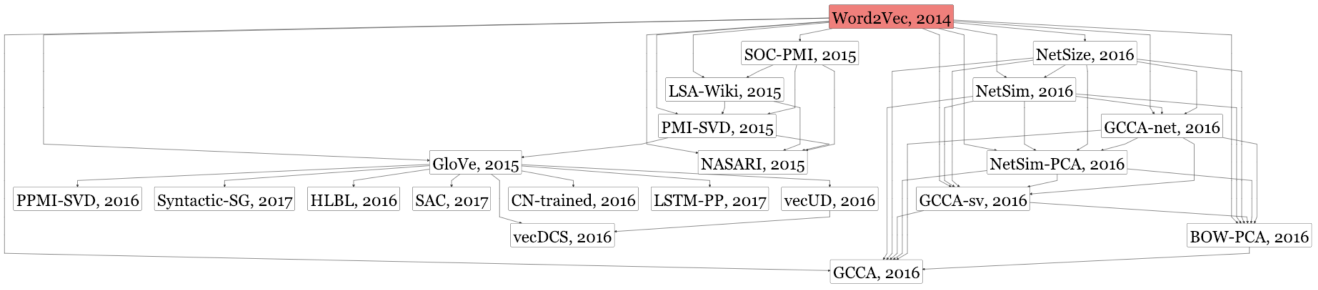
Similarly in ACL dataset, query "Word2Vec" usually stands for a word embedding method. Our method identifies its direct successor such as "Glove," "GCCA," "NetSize," and "NetSime." And Glove has successors including "HLBL," "SAC" and "vecDCS" etc. In VLDB dataset, query "MonteDB" is a database management system, our method finds its direct successor such as "VectorWise," "HyperR," "PostgreSQL." And "MXQuery" has successors such as "BDB" and "MapReduce-RDF-3X." Among the results, "LLVM" is a compiler backend used by some database management system. This error comes from incorrect table parsing, and the pair is treated as a positive example in training data.

Overall our method mines compared algorithms with good quality, though it has potential drawbacks. Some errors come from the direction derivation, mostly because of the incorrect time information and the lack of entity linking. For example, in ACL dataset, "LSA-Wiki" is actually a baseline method compared with "Word2Vec" that uses Latent Semantic Analysis used on Wikipedia. However, this abbreviation as a whole first appears in 2015, resulting in an error in direction. On the other hand, the first appearance time of an algorithm in the dataset is non-necessarily the first time this algorithm was proposed since algorithms could firstly show up in

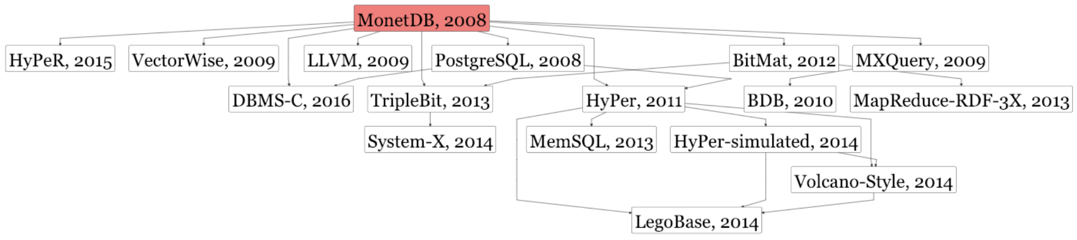
[8]https://github.com/hindupuravinash/the-gan-zoo



(a) Partial roadmap for "GAN" in NeurIPS dataset.



(b) Partial roadmap for "Word2Vec" in ACL dataset.



(c) Partial roadmap for "MonetDB" in VLDB dataset.

Figure 4: Examples of partial *algorithm roadmaps* for query "GAN" in NeurIPS dataset, query "Word2Vec" in ACL dataset, and query "MonetDB" in VLDB dataset.

other conferences/journals or even in other domains. Some of these conferences/journals are not non-open access, which means data sources for mining *algorithm roadmap* are naturally incomplete. Fortunately, the rise of open access repositories such as Arxiv [9], alleviates the data source incompletion problem.

7 CONCLUSION

We propose a new task of *mining algorithm roadmap* in scientific literature, and present a weakly-supervised method towards solving this problem. Our method automatically identifies candidate mentions and relation labels, then jointly predicts abbreviation types and extracts comparative relation across sentences leveraging attention context of words and abbreviations, finally it connects individual pairs into a roadmap. Our model outperforms baseline methods on three real-world datasets and shows good mining results.

Our current model mainly focuses on algorithms in the form of abbreviations. However, this could be extended to general forms of entities and relations by integrating our model with general phrase mining algorithms [30], entity linking [19], and general cross-sentence relations with corresponding supervision signal. We will leave these directions for the future work.

[9]https://arxiv.org/

ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers for their thoughtful comments. This research was sponsored in part by the Army Research Laboratory under cooperative agreements W911NF09-2-0053 and NSF IIS 1528175. The views and conclusions contained herein are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notice herein.

REFERENCES

[1] Isabelle Augenstein, Mrinal Das, Sebastian Riedel, Lakshmi Vikraman, and Andrew McCallum. 2017. Semeval 2017 task 10: Scienceie-extracting keyphrases and relations from scientific publications. In *Proceedings of The 12th International Workshop on Semantic Evaluation.*

[2] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450* (2016).

[3] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data.* AcM, 1247–1250.

[4] Georgeta Bordea, Els Lefever, and Paul Buitelaar. 2016. Semeval-2016 task 13: Taxonomy extraction evaluation (texeval-2). In *Proceedings of the 10th International Workshop on Semantic Evaluation.* 1081–1091.

[5] Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R Hruschka Jr, and Tom M Mitchell. 2010. Toward an architecture for never-ending language learning.. In *AAAI*, Vol. 5. Atlanta, 3.

[6] Christopher Andreas Clark and Santosh Divvala. 2015. Looking beyond text: Extracting figures, tables and captions from computer science papers. In *Workshops at the Twenty-Ninth AAAI Conference on Artificial Intelligence.*

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[8] George R Doddington, Alexis Mitchell, Mark A Przybocki, Lance A Ramshaw, Stephanie Strassel, and Ralph M Weischedel. 2004. The Automatic Content Extraction (ACE) Program-Tasks, Data, and Evaluation.. In *LREC*, Vol. 2. 1.

[9] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems.*

[10] Gregory Grefenstette. 2015. INRIASAC: Simple hypernym extraction methods. In *Proceedings of the 9th International Workshop on Semantic Evaluation.*

[11] Thomas L Griffiths, Michael I Jordan, Joshua B Tenenbaum, and David M Blei. 2004. Hierarchical topic models and the nested chinese restaurant process. In *Advances in neural information processing systems.* 17–24.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition.* 770–778.

[13] Qi He, Bi Chen, Jian Pei, Baojun Qiu, Prasenjit Mitra, and Lee Giles. 2009. Detecting topic evolution in scientific literature: how can citations help?. In *Proceedings of the 18th ACM conference on Information and knowledge management.* ACM, 957–966.

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

[15] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[16] Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015.*

[17] Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. 2016. Neural relation extraction with selective attention over instances. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, Vol. 1. 2124–2133.

[18] Xuezhe Ma and Eduard Hovy. 2016. End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, Vol. 1. 1064–1074.

[19] Pablo N Mendes, Max Jakob, Andrés García-Silva, and Christian Bizer. 2011. DBpedia spotlight: shedding light on the web of documents. In *Proceedings of the 7th international conference on semantic systems.* ACM, 1–8.

[20] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems.* 3111–3119.

[21] David Mimno, Hanna Wallach, and Andrew McCallum. 2008. Gibbs sampling for logistic normal topic models with graph-based priors. In *NIPS Workshop on Analyzing Graphs*, Vol. 61.

[22] Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics.* Association for Computational Linguistics, 1003–1011.

[23] Alexander Panchenko, Eugen Ruppert, Stefano Faralli, Simone Paolo Ponzetto, and Chris Biemann. 2017. Unsupervised does not mean uninterpretable: The case for word sense induction and disambiguation. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, Vol. 1. 86–98.

[24] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in PyTorch. (2017).

[25] Nanyun Peng, Hoifung Poon, Chris Quirk, Kristina Toutanova, and Wen-tau Yih. 2017. Cross-sentence n-ary relation extraction with graph lstms. *TACL* (2017).

[26] Chris Quirk and Hoifung Poon. 2017. Distant Supervision for Relation Extraction beyond the Sentence Boundary. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics.* 1171–1182.

[27] Alec Radford, Luke Metz, and Soumith Chintala. 2016. Unsupervised representation learning with deep convolutional generative adversarial networks. In *4th International Conference on Learning Representations, ICLR 2016.*

[28] Xiang Ren, Zeqiu Wu, Wenqi He, Meng Qu, Clare R Voss, Heng Ji, Tarek F Abdelzaher, and Jiawei Han. 2017. Cotype: Joint extraction of typed entities and relations with knowledge bases. In *Proceedings of the 26th International Conference on World Wide Web.* International World Wide Web Conferences Steering Committee, 1015–1024.

[29] Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases.* Springer, 148–163.

[30] Jingbo Shang, Jialu Liu, Meng Jiang, Xiang Ren, Clare R Voss, and Jiawei Han. 2018. Automated phrase mining from massive text corpora. *IEEE Transactions on Knowledge and Data Engineering* 30, 10 (2018), 1825–1837.

[31] Vered Shwartz, Yoav Goldberg, and Ido Dagan. 2016. Improving hypernymy detection with an integrated path-based and distributional method. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016.*

[32] Rion Snow, Daniel Jurafsky, and Andrew Y Ng. 2005. Learning syntactic patterns for automatic hypernym discovery. In *Advances in neural information processing systems.* 1297–1304.

[33] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research* 15, 1 (2014), 1929–1958.

[34] Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web.* ACM, 697–706.

[35] Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D Manning. 2012. Multi-instance multi-label learning for relation extraction. In *Proceedings of the 2012 joint conference on empirical methods in natural language processing and computational natural language learning.* 455–465.

[36] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems.* 5998–6008.

[37] Patrick Verga, Emma Strubell, and Andrew McCallum. 2018. Simultaneously Self-Attending to All Mentions for Full-Abstract Biological Relation Extraction. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018.*

[38] Mark Ware and Michael Mabe. 2015. The STM report: An overview of scientific and scholarly journal publishing. (2015).

[39] Wentao Wu, Hongsong Li, Haixun Wang, and Kenny Q Zhu. 2012. Probase: A probabilistic taxonomy for text understanding. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data.* ACM, 481–492.

[40] Daojian Zeng, Kang Liu, Yubo Chen, and Jun Zhao. 2015. Distant Supervision for Relation Extraction via Piecewise Convolutional Neural Networks.. In *Proceedings of Conference on Empirical Methods in Natural Language Processing.* Association for Computational Linguistics, 1753–1762.

[41] Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, Jun Zhao, et al. 2014. Relation Classification via Convolutional Deep Neural Network.. In *Proceedings of the International Conference on Computational Linguistics.* 2335–2344.

[42] Chao Zhang, Fangbo Tao, Xiusi Chen, Jiaming Shen, Meng Jiang, Brian Sadler, Michelle Vanni, and Jiawei Han. 2018. TaxoGen: Constructing Topical Concept Taxonomy by Adaptive Term Embedding and Clustering. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining.*

A IMPLEMENTATION DETAILS

A.1 Preprocessing

The paper pdf files are converted into plain text files by using Linux pdftotext tool, and non-ascii letters are removed. For each dataset, we keep a word vocabulary with all abbreviations and other words with minimum frequency threshold 5. The max paragraph length is set to 160 words, and the max number of continuous sentences considered is set to 20. Paragraphs longer than the threshold are cut off.

A.2 Training

The model is implemented in pytorch [24] and trained on a single GeForce GTX 1080 GPU. The dimensions of word embedding, character embedding, and positional embedding are set to 100, 50 and 10 respectively. The word embedding is pre-trained in each scientific publication corpus with Skip-Gram model implemented in Gensim. The kernel size of the convolutional layer is set to 7. vfill

We use 200 filters for the convolutional layer in the single-sentence module, and the same number of filters as input dimension for the convolutional layer in each Transformer block. We apply layer normalization [2] to each component of transformer block, and adopt dropout [33] to the input layer, piece-wise max-pooling and Transformer block with a dropout rate 0.3. The number of Transfomer block layer is set to 1, since we did not observe performance gain in increasing layers. We use Adam optimizer [16] with a learning rate 0.001. In training, the batch size is set to be 32, and for each positive example, we sample 5 different negative examples. In validation and test, we use all examples. The maximum number of epochs is set to be 16, where the result with best positive-class validation F1 is kept.