Knowledge Graph Construction for Academic Paper and Knowledge Graph Based Paper Review System

Zhongye Wang
Shanghai Jiao Tong University
517030910353

Yichen Xie
Shanghai Jiao Tong University
517030910355

Xinyu Zhan
Shanghai Jiao Tong University
517030910358

***Abstract*—In this paper, we focus on the extraction of knowledge graph from academic papers in PDF forms, which allow us to process PDF papers in the form of structured information with efficiency. We propose a framework combining existing XML extraction technique and our two-stage knowledge graph refinement to extract knowledge graph with logical meanings from PDF papers. It first build a primitive knowledge graph from XML file extracted from the paper, and then embed the knowledge graph with structured relation embeddings to obtain a refined knowledge graph with more logic relations. We then build a GNN-based review model above the refined knowledge graph to evaluate the performance of papers, which also fine-tune the embedding model by introducing human reviews of papers. The framework is implemented and tested on 3000+ ICCV papers from the AceMap database and we obtain promising some experiment results.**

***Index Terms*—Information Extraction, Knowledge Graph Embedding, Paper Review System, Graph Neural Network**

I. Introduction

Recent years have witnessed an explosion of the number of academic/scientific papers and submissions in many conferences and journals. This phenomenon yields both opportunities and challenges. On one hand, the exchange of ideas and knowledge among researchers is facilitated by the boost of academic papers, and the great number of papers makes them a perfect dataset for some automatic review models. On the other hand, it results in difficulty in undertanding and evaluating these papers. We lack approaches to convert pure PDF papers and text files into structured data that can be efficiently understood by machines while preserving most of the prior logical information from the paper.

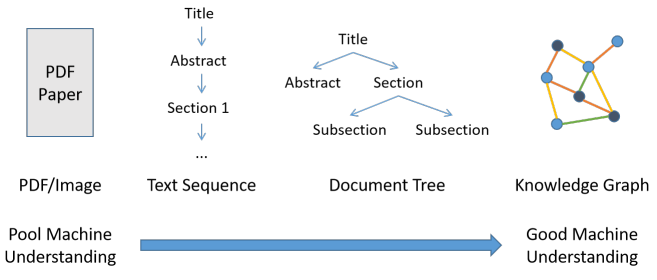


Fig. 1: Structure Information of Academic Paper

As shown in Figure 1, academic papers can be represented in multiple data forms.

- We can treat the entire PDF paper as an **image** and use image processing techniques to analyze it. However, we lose almost all logic information conveyed by texts, and we can hardly count this form as structured data.
- By converting PDF paper into pure **text sequences**, we can process it with common natural language processing techniques to analyze it, like using recurrent neural networks for text understanding tasks.
- We can also represent the paper as a **document tree** with hierarchical structure. This data form further illustrates the logic structure among different components of the paper.
- The ultimate data structure to model the paper is the **knowledge graph**, where each node represents an entity in the paper (section titles, section texts, figures, tables, *etc.*) and each edge represents the logic relation between two entities. The knowledge graph reserves logic information in the paper to great extent, while machines can still understand it efficiently.

In this project, we aims at constructing a framework to extracte a knowledge graph with meaningful entities and relations from an academic paper. Figure 2 demonstrates the pipeline of our project, which can be divided into two phases.

- In the first phase, we extract structure information in XML format from PDF papers and build primitive knowledge graph with only structure relations and reference relations inside a single paper. (Section III)
- The second phase is a two-stage refinement of the primitive knowledge graph.
  - In stage one, we train models to embed entities into vectors and train embedding vectors to represent relations. We use a scoring model to determine the existence of certain relation between two entities. Different from conventional knowledge graph embeddings, we assume a primitive relation is composed of multiple sub-relations with logic meaning and unsupervisedly train multiple embeddings for these sub-relations instead of a single embedding for the relation. With these sub-relations, we derive a refined knowledge graph with more meaningful relations than the primitive one. (Section IV-A)
  - In stage two, we build upon the refined knowledge graph a review model, which uses graph neural

network to evaluate the knowledge graph representation of the paper. The review model itself gives an efficient way to evaluate the performance of papers and therefore alleviating the workload of conferences reviewers. The training process of such model can further fine-tune the embedding model in the previous stage to derive sub-relations with realistic meaning by introducing human understandings of the paper in reviews. (Section IV-B)

- We implement our framework and conduct experiments to verify it in Section V.

II. Related Work

A. Structure Information Extraction

To handle the increasing volume of global research output, much attention was paid to the task of automatic document processing. Some early work [1] focused on the plain-text analysis, which had difficulty in dealing with camera-ready publications. It limited their application to more expansive data sources. As a popular file format, there existed rising requirement to extract information from PDF files. Notabale progress was achieved in this task. Djean *et.al.* [2] developed a method to convert PDF document to XML format. Since only words were extracted in their work, lots of other information such as figures was ignored. Further, Talbert and Wang [3], [4] took figures into consideration, but their approach could only recognize the article body as a whole instead of dividing it further. Some recent work [5], [6] successfully reconstructed the logical relationship inside the article by tagging different entities such as title, abstract and paragraph. These privious art mainly transformed PDF files into XML or HTML format. Despite the satisfying accuracy, relationship among different parts couldn't be modelled clearly. And we resort to knowledge graph to complement this drawback.

B. Knowledge Graph

Knowledge graph [7], [8] visualizes the semantic network among various stuff. A knowledge graph is composed of a set of interconnected typed entities and their attributes. Its construction can be divided into three core techniques: Data Acquisition, Information Acquisition and Knowledge Fusion. Previous work has widely applied knowledge graph in many different fields such as social relationship [9] and commodity recommendation [10]. However, we find that few previous reserach tried to utilize it to parse a single article structure despite its great potential to model the relationship.

In knowledge graph, each link is embedded as a triplet (h, l, t), where h, t represent the soruce and target entity and l denotes the relation. Bordes *et.al.* proposed an approach called TransE to construct the low dimensional dense vector h, l, t by making $h + l \approx t$. Based on it, [11]–[13] further improved this approxiation metric. However, supervision was indispensible for these work so the relationship must be defined manually by human., which made it hard to find some inapperent relations. Conversely, we design a semi-supervised algorithm to generate the embedding vectors of entities as well as both visible and invisible relations with little prior definition.

There have been previous work to extract knowledge graph of author collaboration information based bibliography information in papers [14]. However, no one has ever tried to extract a knowledge graph that represents the organization and logic inside a paper like us.

C. Paper Review

Surging contribution amount leads to huge burden of reviewers, which calls for the assistance of automatic review models. Previous art mainly concentrated on two different insights: text based methods and vision based methods. Text-based models [15] trained a classifier to grade essays based on keywords or other features inside text, with human scored papers as training data. These techniques didn't take into consideration the rich visual information in the paper. Afterwards, computer vision techniques are introduced to this task. These work [16], [17] judged papers with their layout and visual effect. Unluckly, as a significant factor, these previous methods didn't attach importance to the logical structure inside an article. Thus, these approaches were easy to result in some flashy and superficial work. As a supplement, we also develop an approach to score papers based on the refined knowledge graph of papers, which consists of logical connections among different parts inside a paper.

III. Structure Information Extraction

Our project focuses on the refinement and utilization of structural information inside papers. However, the common PDF files can only be recognized as images by computers. As a pre-requisite of our project, we need to reconstruct the logic structure of articles in PDF form as well as ignore some unnecessary formatting style, which is shown in Sec. III-A. Further, we construct a primitive knowledge graph based on the extracted entities and relations. As described in Sec. III-B, it reveals the basic structure of articles.

A. PDF to XML Extraction

In this part, we recover the logic structure inside a paper based on the PDFX model [5]. PDF files are converted to XML format, where each elements of a paper, like abstract, title, text, *etc.*, are extracted as separated units.

It carries out a two-stage process in order to address structure recovery. Firstly, a geometric model is constructed to determine the spatial organization of texts and figures in the article. It identifies pages, words and bitmap images inside the PDF file. Each element is modelled as a separate object with specific features, including bounding box, textual content and font information. These features serve as the basis of content block partition and further classification.

Then, a second stage identifies the semantic type of these partitioned blocks, and groups them into regions with same types. Each time it identifies one semantic type across the whole article according to some specific features. Those regions composed of words with most frequent font are

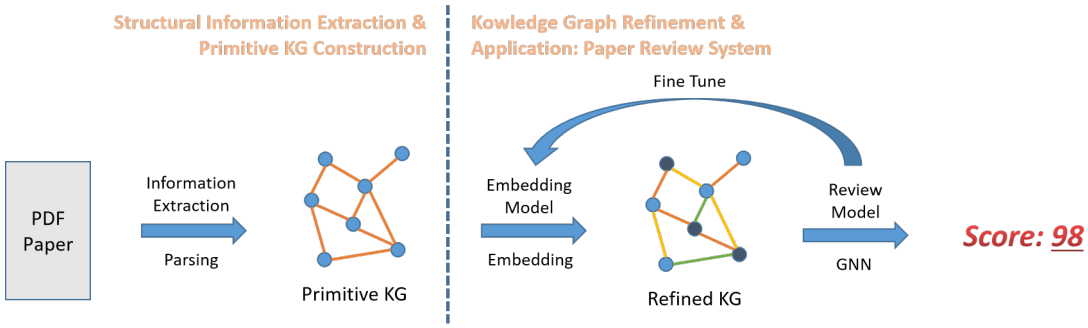


Fig. 2: Project Pipeline

identified as *body text*. Then, the column layout and intended reading order can be determined with it. Afterwards, other regions are tagged in a priority manner where the easiest elements will be identified first.

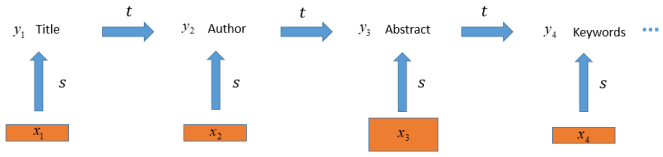


Fig. 3: Conditional Random Field (CRF) model

An approach [18] based on Conditional Random Field (CRF) is introduced to label the XML elements. The process is explained in Fig. 3. Unlike a discrete classifier focuses on a single object's own features, a CRF model takes the context into consideration. In other words, it refers to the neighbors of this region when labelling.

Given a sequence $\mathbf{x}$ of input regions, we define the probability to label sequence $\mathbf{x}$ with sequence y as

$$\begin{aligned} P(\mathbf{y}|\mathbf{x}) &= \frac{1}{Z(\mathbf{x})} e^{\sum_{i,k} \lambda_k t_k(\mathbf{y}_{i-1},\mathbf{y}_i,\mathbf{x},i)+\sum_{i,l} \mu_l s_l(\mathbf{y}_i,\mathbf{x},i)} \\ Z(\mathbf{x}) &= \sum_{\mathbf{y}} e^{\sum_{i,k} \lambda_k t_k(\mathbf{y}_{i-1},\mathbf{y}_i,\mathbf{x},i)+\sum_{i,l} \mu_l s_l(\mathbf{y}_i,\mathbf{x},i)} \end{aligned} \tag{1}$$

where $t(\cdot)$ and $s(\cdot)$ are functions relying on the previous region and current region. λ and μ are corresponding weight. $Z(\mathbf{x})$ is a normalization coefficient.

The model labels the sequence of regions x with y^* of maximal likelihood using Equ. (2).

$$\mathbf{y}^* = \arg\min_{\mathbf{y}} P(\mathbf{y}|\mathbf{x}) \tag{2}$$

As a result, each extracted element in the original article can be assigned a label with maximal overall likelihood.

B. Primitive KG Construction

With the converted XML file, we want to show the relations inside the paper through a knowledge graph.

The elements in the XML file will serve as entities in the knowledge graph, with corresponding tags, types, and texts as fields of the nodes. The logical structure in the XML file is considered as relations between elements *i.e.* edges between nodes.

We define two kinds of relations inside the paper: **Structural Relation** and **Reference Relation** (Tab. I).

TABLE I: Primitive Relations among Entities

Subject	Predicate	Object
Structural Relation		
title	parent	top-level heading/abstract
top-level heading	parent	second-level heading/text body
second-level heading	parent	third-level heading/text body
third-level heading	parent	text body
i-th heading/paragraph	lead	$(i+1)$-th heading/paragraph
figure/table/formula caption	caption	figure/table/formula
Reference Relation		
text body	refer	heading/figure/table/formula

Three structural relations naturally come from the paper structure.

- The *parent* relation defines the hierarchical structure of the paper as a document tree (the *title* is the *parent* of a *section*).
- The *lead* relation defines the sequential structure of the paper (the first *section leads* the second *section*).
- The *caption* defines the relation between text captions with figures and tables, which is crucial if we want to consider interactions among texts and figures/tables when formulating the knowledge graph.

We also take reference relations into consideration. By considering references among sections, figures, tables, *etc.*, we can enrich the primitive knowledge graph with interactions among elements of the paper meant by the author. However, in this part, we can only distinguish these apperent references while other semantic references, like *"in the following sections, ..."*, remain invisible. The later knowledge graph embedding model in Sec. IV may achieve this. An example of primitive knowledge graph is demonstrated in Fig. 5, where each node represents an enity and each edge reveals a relation.

The primitive knowledge graph presented in this section have already conveyed more information than any other forms of structure information of academic papers presented in Sec I (text sequences and document trees). In later sections, we further improve the quality of the knowledge graph.

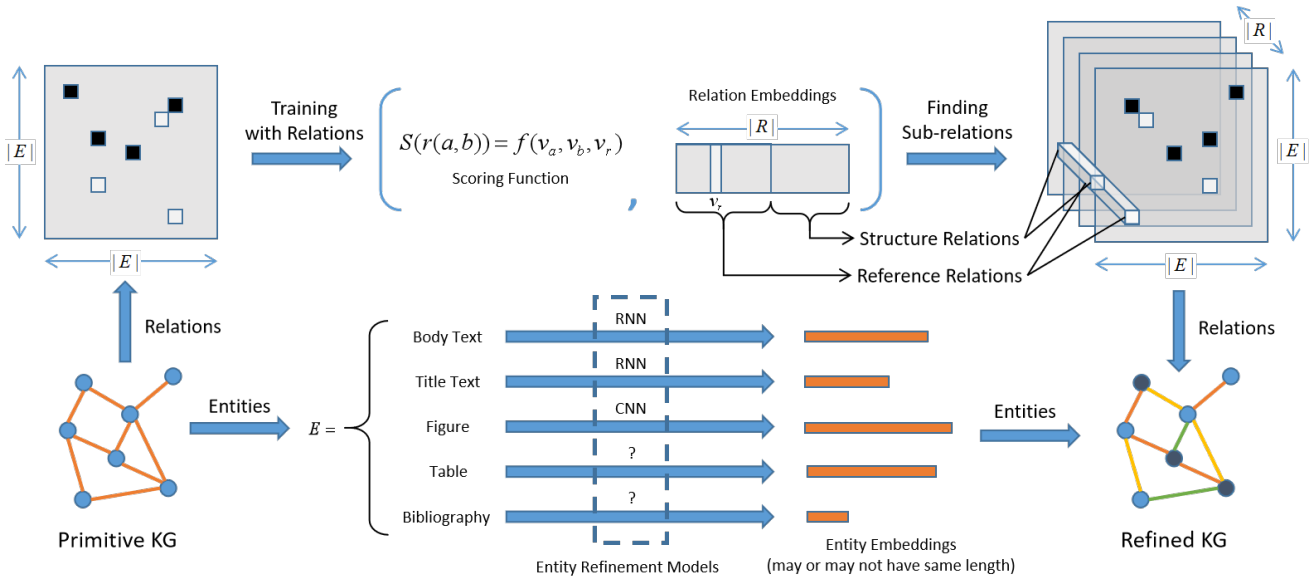


Fig. 4: The Embedding Model

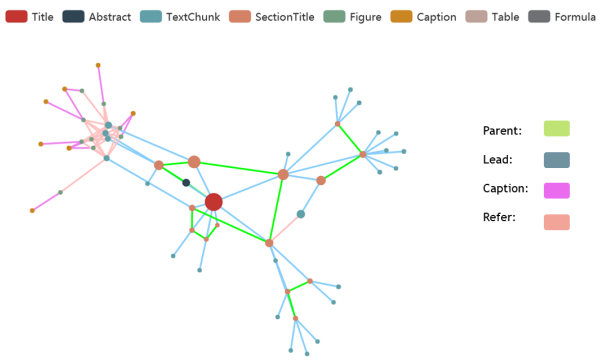


Fig. 5: Primitive Knowledge Graph

IV. Two-Stage Refinement of Knowledge Graph

In the previous section, we manage to parse PDF papers into XML format files and extract primitive knowledge graph with simple structure relations and reference relations. However, there are several problems with the primitive knowledge graph.

- The knowledge graph may be incomplete where we have missing relations between two entities. The aforementioned implicit reference in Section III-B is an example of such cases.
- The relation types we have are too naive. They only allow us to recover the paper structure but ignore the logic relation that links different components of a paper. For example, we know section 1 leads section 2, but we do not know whether section 2 is in parallel with section 1 or further explains concepts in section 1.

To overcome these problems, we use embedding method to refine the relations in primitive knowledge graph. We assume a primitive relation in the primitive knowledge graph is composed of several sub-relations, and we use embeddings for these sub-relations to define the one for the original relation in Section IV-A.

Primitive KG vs. Refined KG. The primitive knowledge graph contains only four types of primitive relations (structure relations and reference relations), which do not identify potential logic relations among entities; the refined knowledge graph has various types of relations, which are possible relations that underlies primitive relations. For example, we know entities a and b have relation r in the primitive knowledge graph, and r is composed of sub-relations r_1 and r_2, then the refined knowledge graph tells us whether a and b have relation r_1 and r_2. In short, the adjacency matrix for the refined knowledge has more channel numbers (more types of edges) than the primitive knowledge graph.

The reason we have extracted primitive knowledge graph first is that we do not have labeled data for relations in refined knowledge graph. What we can get from the PDF paper without additional annotations for logic organization of the paper are only four primitive relations. Therefore, the learning of the refined knowledge graph using embedding model in the first stage is an unsupervised process and the embeddings for sub-relations may lack realistic meaning. To tackle this problem, we use a second stage to fine-tuning the embedding model by introducing human understandings of the paper, the review, and train a review model to analyze the performance of the paper based on the refined knowledge graph in Section IV-B.

A. Stage 1: KG Embedding with Structured Relation Embeddings

Figure 4 demonstrates the entire process of our knowledge graph embedding method, and we will explain it part by part.

Entity Embedding. The lower part of Figure 4 shows the embedding process of entities in primitive knowledge graph.

We embed different types of entities into numerical vectors using different models. For texts, we can use pre-trained recurrent neural network as the embedding model; for figures, we can use encoders implemented with convolutional neural network. However, it is not yet clear how to handle entities such as tables and bibliographies. Tables are texts structured in a more visual fashion, which cannot be handled properly as either images or text sequences. Bibliographies contain many information about referred papers and their authors, which cannot be efficiently encoded either.

In our implementation of the framework, we only consider text elements (titles, headings, body texts) for simplicity. And to save development time, we use weighted-average word vectors to encode paragraphs, rather than training an RNN model from scratch[1].

For a paragraph x with a sequence of words $[x_1, x_2, x_3, \cdots]$, we calculate the its encoding by averaging over word vectors (existing model) weighted by the TF-IDF values of each word.

$$\vec{v}(x) = \frac{\sum_{x_i \in x} \text{TF-IDF}(x_i)\vec{v}(x_i)}{\sum_{x_i \in x} \text{TF-IDF}(x_i)}$$

In this way, we have an efficient encoding of the paragraph while capturing most of its core ideas.

We embed entities into vectors for two reasons.

- In this way, we can have a more consistent representation of various types of entities in a paper, which is beneficial if we want to implement applications above the refined knowledge graph.
- The embedding of entities is necessary if we want to perform relation embedding and therefore achieve relation prediction.

Relation Embedding. Relation embedding aims at represent different relations in numerical form, and using the embeddings for one relation r and two entities a and b, we should be able to determine whether relation r exists between a and b using a scoring function. This is the most important step in knowledge graph embedding, which defines the concept of "graph" given a set of entities.

The upper part of Figure 4 includes the scoring function and the embedding vectors of relations, which composes of the embedding model for the relation. We can train the scoring function and embeddings with the conventional triplet loss and by randomly sampling a batch of positive edges (relations that do exist) and negative edges (relations that do not exist) in each optimization step.

The model needs to have ability to distinguish whether a paricular relation exists between two entities, i.e. whether a directional edge exists between two nodes. To accomplish this, within each iteration a batch of positive edges are sampled with label 1, as well as a batch of negative edges with label 0. The loss for optimizing this multi-label problem can be formulized as follows:

$$\mathcal{L}_{cls} = \text{CrossEntropy}\Big(\mathbb{S}(e_i, e_j, r_k), a_{gt}\Big)$$

[1]We haven't found any pre-trained model that can fit into the paragraph encoding task in our framework.

where $\mathbb{S}$ denotes our model for relation inferencing; e_i, e_j are entity embeddings for entity i and j respectively; r_k is the relation embedding for (sub)relation k, which is a optimizable parameter; a_{gt} denotes the ground-truth label, i.e. whether the triplet (i, j, k) exists in original knowledge graph $\mathbf{G}$.

In the subrelation learning settings (described in following paragraphs) where each opserved relation is decoupled into a number of subrelations and embeddings of them are learned separately, only optimizing the $\mathcal{L}_{cls}$ without any constraint on relation embedding $\mathbf{R} = [r_k], k \in K$ will lead into degenerating cases. The phenomenon is all subrelations inhering from common parent relation will have the same embedding. This diverts from our original intention of discovering latent relations in knowledge graph, for in semantic sense if two relations share the same embedding, they could be regarded as one relation. Inspired by triplet loss, we introduce a new loss to "push" this embeddings away from each other. Within each optimization step, a pair of relations k_1, k_2 sharing the same parent relation will be sampled, and a loss, embed loss as we called, is introduced to let the distance between them as large as possible.

$$\mathcal{L}(r_{k_1}, r_{k_2}) = -\|r_{k_1} - r_{k_2}\|_2^2$$

where

$$\text{Parent}(k_1) = \text{Parent}(k_2)$$

Then the loss on all relation embedding can be formulized as follows:

$$\text{L}_{embed} = \mathop{\mathbb{E}}_{k_i, k_j}\Big[-\mathbf{1}_{(\text{Parent}(k_1)=\text{Parent}(k_2))}\|r_{k_1} - r_{k_2}\|_2^2\Big]$$

The total loss used in relation embedding thus will be:

$$\text{L} = \text{L}_{cls} + \alpha \cdot \text{L}_{embed}$$

where α is the balancing factor between two part of loss.

Different from traditional knowledge graph embedding, we assume a relation may be composed of different sub-relations, and instead of directly computing the embedding vector for each relation, we find the embedding vectors for their sub-relations. In this way, the embedding of a relation is a structure consisting of multiple sub-relation embeddings. Under such assumption, the score of the original relation is determined by scores of the sub-relations.

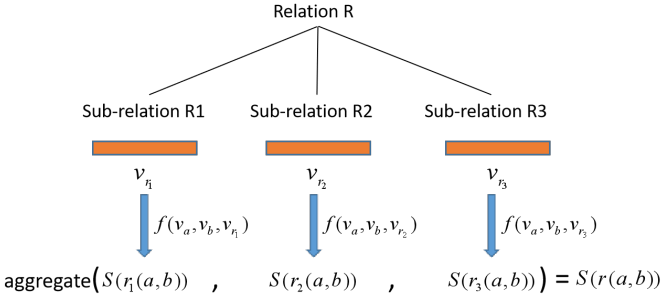


Fig. 6: (Sub-)Relation Embeddings

We introduce a set of functions with permutation symmetry to aggregate the scores of sub-relations and assign the result to the parent relation, as shown in Figure 6. These sub-relation score aggregation functions, denote in symbol $A(R)$, are defined as:

$$A(R) = \text{Aggregate}\Big(\mathbb{S}(i,j,r_{k_1}), \mathbb{S}(i,j,r_{k_2}), \ldots, \mathbb{S}(i,j,r_{k_{|K|}})\Big)$$

where $\text{Aggregate}(\cdot)$ is the main part that actually functions.

The aggregate functions Aggregate as component of subrelation score aggregation functions, should have the following property:

$$\text{Aggregate}(S) = \text{Aggregate}(PS)$$

where S is a vector/matrix and P is any permutation matrix having appropriate dimension.

Here we consider 3 simple forms of aggregation functions $\text{Aggregate}(\cdot)$:

- Maximum: simply taking the element with largest value

$$\text{Aggregate}(s_1, \ldots, s_n) = \max(s_1, \ldots, s_n)$$

- Sum: taking the sum of the elements:

$$\text{Aggregate}(s_1, \ldots, s_n) = \sum_{i=1}^{n} s_i$$

- Adjusted Sum: still taking the sum, but each value is adjusted by a weight p_i. Here we tested using softmax to produce the weights of average function

$$\begin{aligned}
&\text{Aggregate}(s_1, \ldots, s_n) \\
=&\sum_{i=1}^{n} p_i s_i \\
=&\sum_{i=1}^{n} \text{Softmax}(s_i; s_1, \ldots, s_n) s_i \\
=&\sum_{i=1}^{n} \frac{e^{s_i}}{\sum_{j=1}^{n} e^{s_j}} s_i
\end{aligned}$$

With the scoring method, we can do inference for how likely a (sub-)relation exists between two entities by applying a sigmoid function to the score as (3) shows. This yields the low-channel adjacency matrix on the left and the high-channel adjacency matrix on the right in Figure 4.

$$\begin{aligned}
\Pr(r(a,b)) &= \sigma(S(r(a,b))) \\
\Pr(r_i(a,b)) &= \sigma(S(r_i(a,b)))
\end{aligned} \tag{3}$$

With the methods described above, we could obtain (sub)relation embeddings $\mathbf{R}$ and refined knowledge graph constructed based on $\mathbf{R}$. However, the learning process of relation embeddings is unsupervised, i.e. no more information is introduced and utilized during the relation embedding process. This may lead to uniterpretable (sub)relation embeddings. It will be drawback in subrelation discovery, as we want the subrelations should have some semantic meanings.

Therefore we need a framework to introduce more information to finetune the relation embedding model, so that it can not only distinguish the presence triplet (i,j,k), but also generates relation embedding r_k for (sub)relation k with more semantic information. As a result, we could get a new refined knowledge graph $\hat{\mathbf{G}}$ which better corresponds to human knowledge. This is purpose of the second stage of refinement.

B. Stage 2: Fine Tune Embeddings with Review Model

As reasoned in the previous section, we need to introduce extra information that can reflect human understanding of the paper to equip refined knowledge graphs with more semantic meanings. An ideal type of such information is human reviews of the paper, which not only carry human understandings but also evaluate the performance of papers. Examples of human reviews are acceptance scores of judges, judges' comments, or simply citation count of the paper.

We propose a review regression framework as shown in Figure 7 to train an additional review model while finetuning the embedding model. This regression task not only assign semantic meaning in reviews to the refined knowledge graph, but also builds an application above it, which automatically evaluates the paper.

In this framework, we need a dataset of PDF papers and their reviews. We first extract primitive knowledge graphs from PDF papers. We sample a batch of them and embed them into refined knowledge graphs. We then use a GNN-based review model to process the refined knowledge graph and obtain a review of it, which is compared to the ground truth review to get regression error. Therefore, we can back-propagate this error throughout the GNN review model and eventually to the adjacency matrix and node embeddings of the refined knowledge, which can be further passed to the embedding model to finetune it.

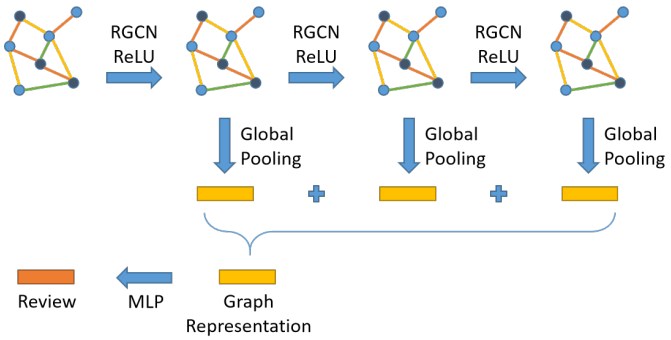


Fig. 8: Structure of Review Model

As for this project, we choose to use the annually average reference counts for the paper. These data are easily accessible from AceMap database, and we believe they have latent relationship with the structure of the refined knowledge graph $\hat{\mathbf{G}}$. To predict the value and minimizes the error might have positive effect on the process of refining knowledge graph $\mathbf{G}$.

Figure 8 shows the structure of the GNN we use as the review model. The relational graph conventional network

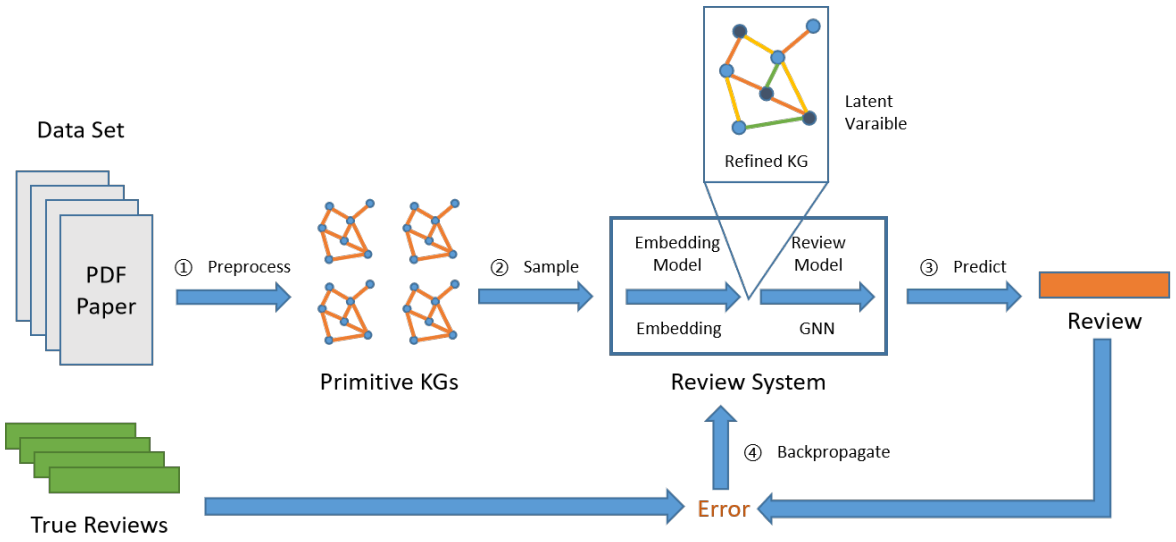


Fig. 7: The Review System and Training Scheme

(RGCN) [19] is a variation of GCN layer, which can handle multi-graphs like knowledge graphs. However, to allow gradient being propagated to the adjacency matrix, we need to use the dense graph version of GCNs, which takes entire adjacency matrix as inputs instead of edge lists. We use three layer skip-connections to compute the global graph representation, which is processed by another multi-level perceptron to get the predicted review.

This concludes our framework of knowledge graph refinement, which distinguishes sub-relations underlying primitive relations and use review information to finetune those sub-relations and builds a paper review application over the refined knowledge graph. In the next part, we will demonstrate our implementation through sets of experiments.

V. Experiments

A. Stage 1: Embedding Model

We execute a number of experiments to evaluate the proposed embedding models and study the effects of its building components. We survey different factorization methods and a number of score aggregation functions and evaluate them on a custom-made dataset to prove their effectiveness. Here we will present the settings and results of experiments we have done on embedding models.

a) Experiment Setup:

Dataset. The training dataset contains 2857 knowledge graphs extracted from ICCV papers using the techniques introduced in Section III-B. It contains 247428 entities and 385776 edges in total. The test dataset contrains 500 knowledge graphs, 42511 entities and 68441 edges.

Training. Within each optimization step, we sample a batch of positive triplets as well as a batch of negative triplets, along with their ground truth labels. Besides, in subrelation embedding, two subrelations sharing the same parent relation are sampled and the negative value of their distance is penalized.

Metric. Since it is impractical to directly evaluate the quality of relation embeddings, we evaluate relation embedding together with relation inference model. We evaluate the accuracy and recall for each triplet (i, j, k) consists of two entities i, j, and one relation k from former to latter. We also draw PR curve and ROC, and AUC is computed for each experiment. Here we weight more on the recall score because we assume many relationships doesn't show up in training data, and as a result instead of using F-scores, we evaluate accuracy and recall respectively, not taking precision into account.

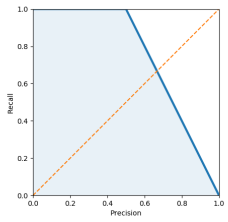
b) Experiment Evaluation: Here we present the experiment results for different model architectures.

TABLE II: Evaluation results for different model architectures.

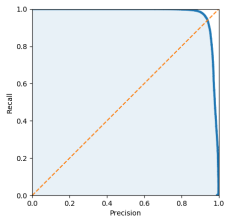
Model	Accuracy	Recall	AUC
trans-E + max	0.3752	0.0005	0.3752
trans-E + sum	0.5006	0.0011	0.5006
trans-E + adjusted sum	0.5000	0.0000	0.5000
mlp + max	0.9429	0.9704	0.9803
mlp + sum	0.9441	**0.9748**	0.9814
mlp + adjusted sum	**0.9459**	0.9630	**0.9817**

We test 2 different factorization models: explicit trans-E model and implicit MLP model. We also test 3 different score aggregation methods: max, sum, adjusted sum. The evaluation results shows our method can achieve 0.9748 recall with MLP and sum structure. The PR Curve is shown in figure 9. The ROC for each experiment is shown in fig 10.

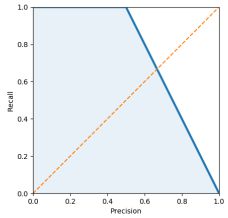
The experiment results shows the implicit MLP model performs better than the explicit Trans-E model. While Trans-E model tries to disentangle entities and relationships and embed them to single space independently, it turns out to be far less competent than MLP which implicitly captures the joint information about the entity-relation triplet.



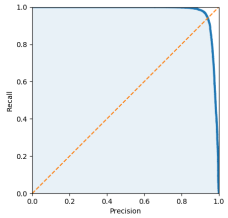
(a) trans-E + max



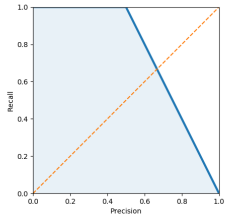
(b) mlp + max



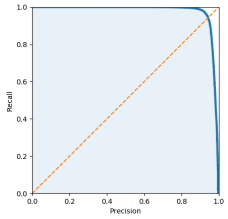
(c) trans-E + sum



(d) mlp + sum

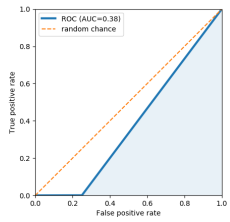


(e) trans-E + adjusted sum

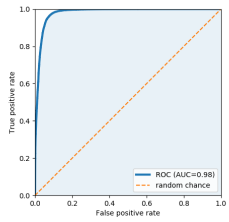


(f) mlp + adjusted sum

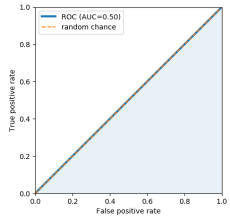
Fig. 9: Evaluation result: PR curves



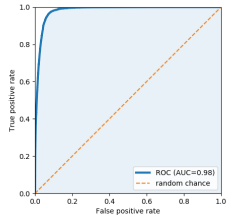
(a) trans-E + max



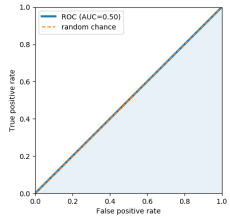
(b) mlp + max



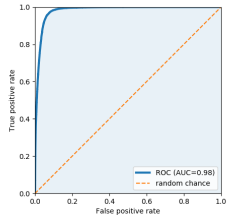
(c) trans-E + sum



(d) mlp + sum



(e) trans-E + adjusted sum



(f) mlp + adjusted sum

Fig. 10: Evaluation result: ROC

It also turns out that among three different scroe aggregation functions we have tested, the adjusted sum aggregation has the best accuracy and AUC value, while the simple sum aggregation has the best recall. It can be concluded that if we want the subrelations stick closer to original relations in primitive knowledge graph, adjusted sum aggregation should be adopted; on the other hand, if we want more latent subrelation been captured, the simple sum aggregation is prefered.

Figure 11 shows the distributions of subrelations. It can be seen that different surelations are learned for each relation. This indicates the model can capture subrelations and the degeneration case doesn't happen.

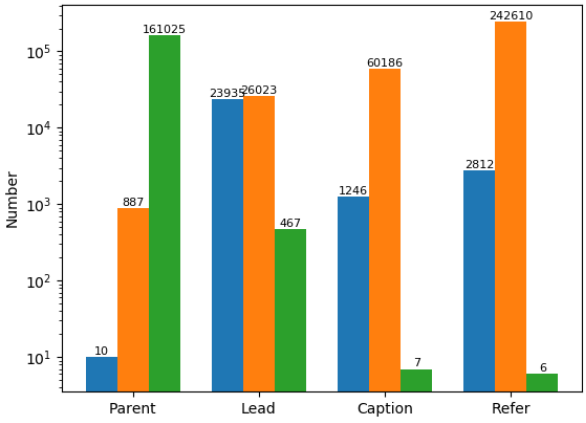


Fig. 11: Subrelations count for each relation

B. Stage 2: Review Model

a) Experiment Setup:

Dataset. In this part of the experiment we use the same data as the experiment in stage 1, though the format is altered. Instead of feeding triplets into the model, we feed all entities within one graph along with the supervision information. Here the supervision information in this experiment is the annually average reference count of each paper.

Training. During the training process, within each step, we first predict the refined knowledge graph with the learned relation embedding and relation inference model we obtained in stage 1 experiment. Afterwards we adopt a series of dense graph convolutional layers to generate prediction. The criterion corresponding to supervision information is applied to jointly optimize relation embedding and relation inference model in stage 1, and the newly introduced prediction model. Here we use MSE-Loss to optimize the whole network.

Note that the framework we proposed here is extensible and

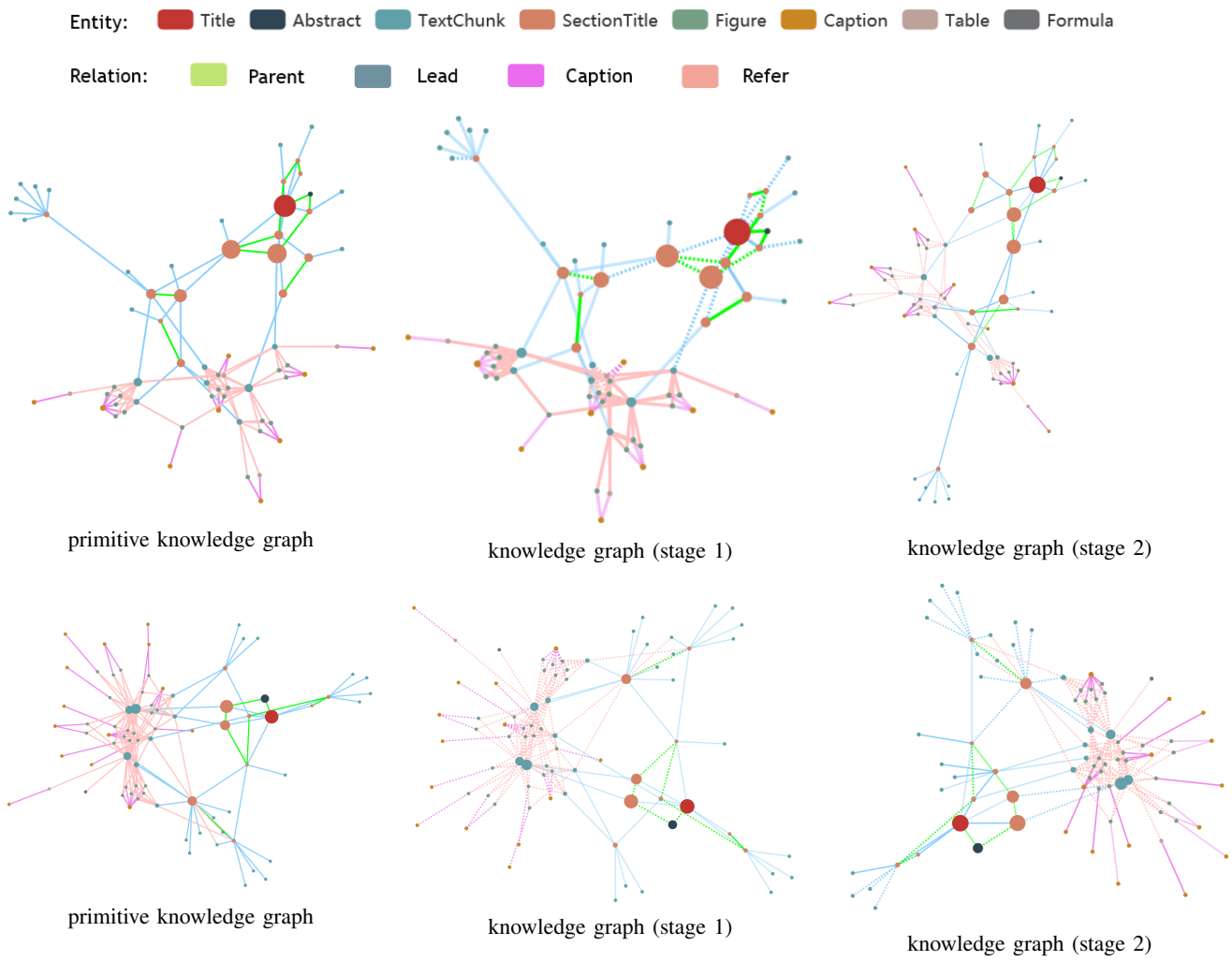


Fig. 12: Visualization Samples: Different linestyles show different sub-relations

compatible with more fine-grained supervision information, not only for regressing one number. To introduce new supervision information, just to modify the prediction model and select the appropriate criterion, and follow the train process mentioned above.

Metric. Since we are trying to regress a number here, the choice of metric is very limited. Here we use Mean Absolute Error as metric, showing the amount the prediction differs from the ground truth values.

b) Experiment Evaluation:

Here we present the experiment results for finetuned models in stage 2 experiments. The results shows our framework achieve best MAE with structure DenseGCNConv and it can capture the distributions of supervision information.

TABLE III: Evaluation results for Stage 2 experiment.

Model	Mean Absolute Error
RelEmbed + RelModel + DenseGCNConv	**5.81**
RelEmbed + RelModel + DenseGraphConv	5.94
RelEmbed + RelModel + DenseSAGEConv	7.71

However, we have not discovered it until we start training the model that the the graph convention layers in the library we use does not support back-propagating gradient to the adjacency matrix, which in turn cause the embedding model in the previous stage cannot be fine-tuned. But this problem can be resolved by implementing our own GCN layers that support such operation. As a result, we can only build the review model above the fixed embedding model, and the fine-tuning of the embedding model will be our future work.

Visualization Samples. In this part we present a few results generated with the refined embeddings obtained in stage 2. Each sample contains 3 figures, the primitive knowledge graph, knowledge graph obtained in stage 1 and knowledge graph in stage 2. The figures are presented in figure 12.

VI. Conclusion

In this paper, we have proposed a framework for the extraction of knowledge graph from academic papers in PDF forms. It first convert PDF papers into XML formats and extract primitive knowledge graphs with only structure relations and reference relations from them. Then we embed entities and

relations of knowledge graphs into vectors. Different from conventional knowledge graph embedding work, we decompose a relation into multiple sub-relations and embed those sub-relations instead, which can be aggregated back to the original relation again. After the embedding, we get a refined knowledge graph, which not only identify missing relations with high accuracy, but also have more logical structure than the primitive one.

We further build a review model based on GNN above the refined knowledge graph which predicts the annually averaged reference count of the paper. Due to limited time, we cannot further improve the network structure to get better results. Also, although the training of the review model can fine-tune the embedding model to assign more logical meaning to it, we do not have time to modify the library to support gradient back-propagation through the adjacency graph. However, this are promising future work that we will look into.

REFERENCES

[1] I. G. Councill, C. L. Giles, and M. Y. Kan, "Parscit: an open-source crf reference string parsing package," in *International Conference on Language Resources and Evaluation*, 2008.

[2] H. Djean and J. L. Meunier, "A system for converting pdf documents into structured xml format," in *International Conference on Document Analysis Systems*, 2006.

[3] M. Talbert, "Mobipocket.com pdf2xml," [EB/OL], https://launchpad.net/pdf2xml.

[4] L. Wang, "The pdf2htmlex project," [EB/OL], http://coolwanglu.github.io/pdf2htmlEX/.

[5] A. Constantin, S. Pettifer, and A. Voronkov, "Pdfx: fully-automated pdf-to-xml conversion of scientific literature," in *Proceedings of the 2013 ACM symposium on Document engineering*, 2013, pp. 177–180.

[6] M. Ley, "Dblp - some lessons learned," *PVLDB*, vol. 2, pp. 1493–1500, 08 2009.

[7] S. A., "Official google blog: Introducing the knowledge graph: things, not strings." *Official Google Blog*, pp. 1–8, 2012.

[8] M. R. Quillian, "Semantic networks," *Approaches to Knowledge Representation Research Studies*, vol. 23, no. 92, pp. 1–50, 1968.

[9] Z. Wang, T. Chen, J. Ren, W. Yu, H. Cheng, and L. Lin, "Deep reasoning with knowledge graph for social relationship understanding," 2018.

[10] Z. Sun, J. Yang, J. Zhang, A. Bozzon, L.-K. Huang, and C. Xu, "Recurrent knowledge graph embedding for effective recommendation," 09 2018, pp. 297–305.

[11] Z. Wang, J. Zhang, J. Feng, and Z. Chen, "Knowledge graph embedding by translating on hyperplanes," in *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, ser. AAAI14. AAAI Press, 2014, p. 11121119.

[12] G. Ji, S. He, L. Xu, L. Kang, and J. Zhao, "Knowledge graph embedding via dynamic mapping matrix," in *Meeting of the Association for Computational Linguistics and International Joint Conference on Natural Language Processing*, 2015.

[13] H. Xiao, M. Huang, Y. Hao, and X. Zhu, "Transa: An adaptive approach for knowledge graph embedding," *Computer Science*, 2015.

[14] Z. Ye, J. Yuting, F. Luoyi, and W. Xinbing, "Acemap academic map and acekg academic knowledge graph for academic data visualization," *Journal of Shanghai Jiaotong University*, 2018.

[15] L. Larkey, "Automatic essay grading using text categorization techniques," *SIGIR Forum (ACM Special Interest Group on Information Retrieval)*, 06 1998.

[16] C. V. Bearnensquash, "Paper gestalt," in *Secret Proceedings of Computer Vision and Pattern Recognition*, 2010.

[17] J.-B. Huang, "Deep paper gestalt," 2018.

[18] A. Mccallum, "Efficiently inducing features of conditional random fields," *UAI-03*, 10 2012.

[19] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. vanden Berg, and M. Welling, "Modeling relational data with graph convolutional networks," in *European Semantic Web Conference*, 2018.