Construction of the Literature Graph in Semantic Scholar

Waleed Ammar, Dirk Groeneveld, Chandra Bhagavatula, Iz Beltagy, Miles Crawford, Doug Downey,[†] Jason Dunkelberger, Ahmed Elgohary, Sergey Feldman, Vu Ha, Rodney Kinney, Sebastian Kohlmeier, Kyle Lo, Tyler Murray, Hsu-Han Ooi, Matthew Peters, Joanna Power, Sam Skjonsberg, Lucy Lu Wang, Chris Wilhelm, Zheng Yuan,[†] Madeleine van Zuylen, and Oren Etzioni
waleeda@allenai.org

Allen Institute for Artificial Intelligence, Seattle WA 98103, USA
[†]Northwestern University, Evanston IL 60208, USA

Abstract

We describe a deployed scalable system for organizing published scientific literature into a heterogeneous graph to facilitate algorithmic manipulation and discovery. The resulting literature graph consists of more than 280M nodes, representing papers, authors, entities and various interactions between them (e.g., authorships, citations, entity mentions). We reduce literature graph construction into familiar NLP tasks (e.g., entity extraction and linking), point out research challenges due to differences from standard formulations of these tasks, and report empirical results for each task. The methods described in this paper are used to enable semantic features in www.semanticscholar.org.

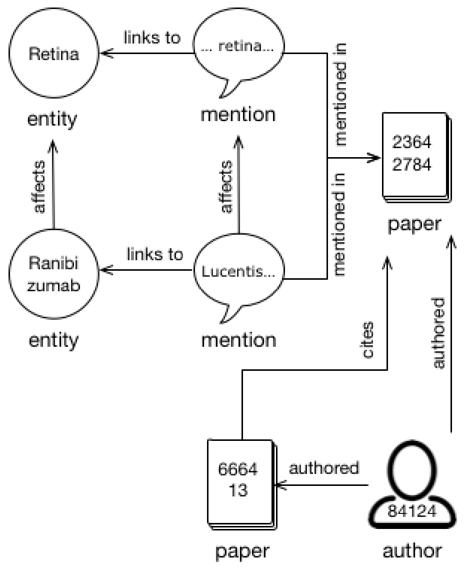

Figure 1: Part of the literature graph.

1 Introduction

The goal of this work is to facilitate algorithmic discovery in the scientific literature. Despite notable advances in scientific search engines, data mining and digital libraries (e.g., Wu et al., 2014), researchers remain unable to answer simple questions such as:

- What is the percentage of female subjects in depression clinical trials?
- Which of my co-authors published one or more papers on coreference resolution?
- Which papers discuss the effects of Ranibizumab on the Retina?

In this paper, we focus on the problem of extracting structured data from scientific documents, which can later be used in natural language interfaces (e.g., Iyer et al., 2017) or to improve ranking of results in academic search (e.g., Xiong et al., 2017). We describe methods used in a scalable deployed production system for extracting structured information from scientific documents into *the literature graph* (see Fig. 1). The literature graph is a directed property graph which summarizes key information in the literature and can be used to answer the queries mentioned earlier as well as more complex queries. For example, in order to compute the Erdős number of an author X, the graph can be queried to find the number of nodes on the shortest undirected path between author X and Paul Erdős such that all edges on the path are labeled "authored".

We reduce literature graph construction into familiar NLP tasks such as sequence labeling, entity linking and relation extraction, and address some of the impractical assumptions commonly made in the standard formulations of these tasks. For example, most research on named entity recognition tasks report results on large labeled datasets such as CoNLL-2003 and ACE-2005 (e.g., Lample et al., 2016), and assume that entity types in the test set match those labeled in the training set (including work on domain adaptation, e.g., Daumé, 2007). These assumptions, while useful for developing and benchmarking new methods, are unrealistic for many domains and applications. The paper also serves as an overview of the approach we adopt at `www.semanticscholar.org` in a step towards more intelligent academic search engines (Etzioni, 2011).

In the next section, we start by describing our symbolic representation of the literature. Then, we discuss how we extract metadata associated with a paper such as authors and references, then how we extract the entities mentioned in paper text. Before we conclude, we briefly describe other research challenges we are actively working on in order to improve the quality of the literature graph.

2 Structure of The Literature Graph

The literature graph is a *property graph* with directed edges. Unlike Resource Description Framework (RDF) graphs, nodes and edges in property graphs have an internal structure which is more suitable for representing complex data types such as papers and entities. In this section, we describe the attributes associated with nodes and edges of different types in the literature graph.

2.1 Node Types

Papers. We obtain metadata and PDF files of papers via partnerships with publishers (e.g., Springer, Nature), catalogs (e.g., DBLP, MEDLINE), pre-publishing services (e.g., arXiv, bioRxive), as well as web-crawling. Paper nodes are associated with a set of attributes such as 'title', 'abstract', 'full text', 'venues' and 'publication year'. While some of the paper sources provide these attributes as metadata, it is often necessary to extract them from the paper PDF (details in §3). We deterministically remove duplicate papers based on string similarity of their metadata, resulting in 37M unique paper nodes. Papers in the literature graph cover a variety of scientific disciplines, including computer science, molecular biology, microbiology and neuroscience.

Authors. Each node of this type represents a unique author, with attributes such as 'first name' and 'last name'. The literature graph has 12M nodes of this type.

Entities. Each node of this type represents a unique scientific concept discussed in the literature, with attributes such as 'canonical name', 'aliases' and 'description'. Our literature graph has 0.4M nodes of this type. We describe how we populate entity nodes in §4.3.

Entity mentions. Each node of this type represents a textual reference of an entity in one of the papers, with attributes such as 'mention text', 'context', and 'confidence'. We describe how we populate the 237M mentions in the literature graph in §4.1.

2.2 Edge Types

Citations. We instantiate a directed citation edge from paper nodes $p_1 \longrightarrow p_2$ for each p_2 referenced in p_1. Citation edges have attributes such as 'from paper id', 'to paper id' and 'contexts' (the textual contexts where p_2 is referenced in p_1). While some of the paper sources provide these attributes as metadata, it is often necessary to extract them from the paper PDF as detailed in §3.

Authorship. We instantiate a directed authorship edge between an author node and a paper node $a \longrightarrow p$ for each author of that paper.

Entity linking edges. We instantiate a directed edge from an extracted entity mention node to the entity it refers to.

Mention–mention relations. We instantiate a directed edge between a pair of mentions in the same sentential context if the textual relation extraction model predicts one of a predefined list of relation types between them in a sentential context.[1] We encode a symmetric relation between m_1 and m_2 as two directed edges $m_1 \longrightarrow m_2$ and $m_2 \longrightarrow m_1$.

Entity–entity relations. While mention–mention edges represent relations between mentions in a particular context, entity–entity edges represent relations between abstract entities. These relations may be imported from an existing knowledge base (KB) or inferred from other edges in the graph.

3 Extracting Metadata

In the previous section, we described the overall structure of the literature graph. Next, we discuss how we populate paper nodes, author nodes, authorship edges, and citation edges.

[1]Due to space constraints, we opted not to discuss our relation extraction models in this draft.

Although some publishers provide sufficient metadata about their papers, many papers are provided with incomplete metadata. Also, papers obtained via web-crawling are not associated with any metadata. To fill in this gap, we built the ScienceParse system to predict structured data from the raw PDFs using recurrent neural networks (RNNs).[2] For each paper, the system extracts the paper title, list of authors, and list of references; each reference consists of a title, a list of authors, a venue, and a year.

Preparing the input layer. We split each PDF into individual pages, and feed each page to Apache's PDFBox library[3] to convert it into a sequence of tokens, where each token has features, e.g., 'text', 'font size', 'space width', 'position on the page'.

We normalize the token-level features before feeding them as inputs to the model. For each of the 'font size' and 'space width' features, we compute three normalized values (with respect to current page, current document, and the whole training corpus), each value ranging between -0.5 to +0.5. The token's 'position on the page' is given in XY coordinate points. We scale the values linearly to range from $(-0.5, -0.5)$ at the top-left corner of the page to $(0.5, 0.5)$ at the bottom-right corner.

In order to capture case information, we add seven numeric features to the input representation of each token: whether the first/second letter is uppercase/lowercase, the fraction of uppercase/lowercase letters and the fraction of digits.

To help the model make correct predictions for metadata which tend to appear at the beginning (e.g., titles and authors) or at the end of papers (e.g., references), we provide the current page number as two discrete variables (relative to the beginning and end of the PDF file) with values 0, 1 and 2+. These features are repeated for each token on the same page.

For the k-th token in the sequence, we compute the input representation $\mathbf{i}_k$ by concatenating the numeric features, an embedding of the 'font size', and the word embedding of the lowercased token. Word embeddings are initialized with GloVe (Pennington et al., 2014).

Model. The input token representations are passed through one fully-connected layer and then fed into a two-layer bidirectional LSTM (Long Short-Term Memory, Hochreiter and Schmidhuber, 1997), i.e.,

$$\mathbf{g}_k^{\rightarrow} = \text{LSTM}(\mathbf{W}\mathbf{i}_k, \mathbf{g}_{k-1}^{\rightarrow}), \mathbf{g}_k = [\mathbf{g}_k^{\rightarrow}; \mathbf{g}_k^{\leftarrow}],$$
$$\mathbf{h}_k^{\rightarrow} = \text{LSTM}(\mathbf{g}_k, \mathbf{h}_{k-1}^{\rightarrow}), \mathbf{h}_k = [\mathbf{h}_k^{\rightarrow}; \mathbf{g}_k^{\leftarrow}]$$

where W is a weight matrix, $\mathbf{g}_k^{\leftarrow}$ and $\mathbf{h}_k^{\leftarrow}$ are defined similarly to $\mathbf{g}_k^{\rightarrow}$ and $\mathbf{h}_k^{\rightarrow}$ but process token sequences in the opposite direction.

Following Collobert et al. (2011), we feed the output of the second layer $\mathbf{h}_k$ into a dense layer to predict unnormalized label weights for each token and learn label bigram feature weights (often described as a conditional random field layer when used in neural architectures) to account for dependencies between labels.

Training. The ScienceParse system is trained on a snapshot of the data at PubMed Central. It consists of 1.4M PDFs and their associated metadata, which specify the correct titles, authors, and bibliographies. We use a heuristic labeling process that finds the strings from the metadata in the tokenized PDFs to produce labeled tokens. This labeling process succeeds for 76% of the documents. The remaining documents are not used in the training process. During training, we only use pages which have at least one token with a label that is not "none".

Decoding. At test time, we use Viterbi decoding to find the most likely global sequence, with no further constraints. To get the title, we use the longest continuous sequence of tokens with the "title" label. Since there can be multiple authors, we use all continuous sequences of tokens with the "author" label as authors, but require that all authors of a paper are mentioned on the same page. If the author labels are predicted in multiple pages, we use the one with the largest number of authors.

Field	Precision	Recall	F1
title	85.5	85.5	85.5
authors	92.1	92.1	92.1
bibliography titles	89.3	89.4	89.3
bibliography authors	97.1	97.0	97.0
bibliography venues	91.7	89.7	90.7
bibliography years	98.0	98.0	98.0

Table 1: Results of the ScienceParse system.

[2]The ScienceParse libraries can be found at `http://allenai.org/software/`.

[3]`https://pdfbox.apache.org`

Results. We run our final tests on a held-out set from PubMed Central, consisting of about 54K documents. The results are detailed in Table 1. We use a conservative evaluation where an instance is correct if it exactly matches the gold annotation, with no credit for partial matching.

To give an example for the type of errors our model makes, consider the paper (Wang et al., 2013) titled "Clinical review: Efficacy of antimicrobial-impregnated catheters in external ventricular drainage - a systematic review and meta-analysis." The title we extract for this paper omits the first part "Clinical review:". This is likely to be a result of the pattern "Foo: Bar Baz" appearing in many training examples with only "Bar Baz" labeled as the title.

4 Entity Extraction and Linking

In the previous section, we described how we populate the backbone of the literature graph, i.e., paper nodes, author nodes and citation edges. Next, we discuss how we populate mentions and entities in the literature graph using entity extraction and linking on the paper text. In order to focus on more salient entities in a given paper, we only use the title and abstract.

4.1 Approaches

We experiment with three approaches for entity extraction and linking:

I. Statistical: uses one or more statistical models for predicting mention spans, then uses another statistical model to link mentions to candidate entities in a KB.

II. Hybrid: defines a small number of hand-engineered, deterministic rules for string-based matching of the input text to candidate entities in the KB, then uses a statistical model to disambiguate the mentions.[4]

III. Off-the-shelf: uses existing libraries, namely (Ferragina and Scaiella, 2010, TagMe)[5] and (Demner-Fushman et al., 2017, MetaMap Lite)[6], with minimal post-processing to extract and link entities to the KB.

[4]We also experimented with a "pure" rules-based approach which disambiguates deterministically but the hybrid approach consistently gave better results.

[5]The TagMe APIs are described at `https://sobigdata.d4science.org/web/tagme/tagme-help`

[6]We use v3.4 (L0) of MetaMap Lite, available at `https://metamap.nlm.nih.gov/MetaMapLite.shtml`

Approach	CS		Bio	
	prec.	yield	prec.	yield
Statistical	98.4	712	94.4	928
Hybrid	91.5	1990	92.1	3126
Off-the-shelf	97.4	873	77.5	1206

Table 2: Document-level evaluation of three approaches in two scientific areas: computer science (CS) and biomedical (Bio).

We evaluate the performance of each approach in two broad scientific areas: computer science (CS) and biomedical research (Bio). For each unique (paper ID, entity ID) pair predicted by one of the approaches, we ask human annotators to label each mention extracted for this entity in the paper. We use CrowdFlower to manage human annotations and only include instances where three or more annotators agree on the label. If one or more of the entity mentions in that paper is judged to be correct, the pair (paper ID, entity ID) counts as one correct instance. Otherwise, it counts as an incorrect instance. We report 'yield' in lieu of 'recall' due to the difficulty of doing a scalable comprehensive annotation.

Table 2 shows the results based on 500 papers using v1.1.2 of our entity extraction and linking components. In both domains, the statistical approach gives the highest precision and the lowest yield. The hybrid approach consistently gives the highest yield, but sacrifices precision. The TagMe off-the-shelf library used for the CS domain gives surprisingly good results, with precision within 1 point from the statistical models. However, the MetaMap Lite off-the-shelf library we used for the biomedical domain suffered a huge loss in precision. Our error analysis showed that each of the approaches is able to predict entities not predicted by the other approaches so we decided to pool their outputs in our deployed system, which gives significantly higher yield than any individual approach while maintaining reasonably high precision.

4.2 Entity Extraction Models

Given the token sequence $t_1, \ldots, t_N$ in a sentence, we need to identify spans which correspond to entity mentions. We use the BILOU scheme to encode labels at the token level. Unlike most formulations of named entity recognition problems (NER), we do not identify the entity type (e.g., protein, drug, chemical, disease) for each mention since the output mentions are further grounded in a KB with further information about the entity (including its type), using an entity linking module.

Model. First, we construct the token embedding $\mathbf{x}_k = [\mathbf{c}_k; \mathbf{w}_k]$ for each token t_k in the input sequence, where $\mathbf{c}_k$ is a character-based representation computed using a convolutional neural network (CNN) with filter of size 3 characters, and $\mathbf{w}_k$ are learned word embeddings initialized with the GloVe embeddings (Pennington et al., 2014).

We also compute context-sensitive word embeddings, denoted as $\mathbf{lm}_k = [\mathbf{lm}_k^{\rightarrow}; \mathbf{lm}_k^{\leftarrow}]$, by concatenating the projected outputs of forward and backward recurrent neural network language models (RNN-LM) at position k. The language model (LM) for each direction is trained independently and consists of a single layer long short-term memory (LSTM) network followed by a linear project layer. While training the LM parameters, $\mathbf{lm}_k^{\rightarrow}$ is used to predict t_{k+1} and $\mathbf{lm}_k^{\leftarrow}$ is used to predict t_{k-1}. We fix the LM parameters during training of the entity extraction model. See Peters et al. (2017) and Ammar et al. (2017) for more details.

Given the $\mathbf{x}_k$ and $\mathbf{lm}_k$ embeddings for each token $k \in \{1, \dots, N\}$, we use a two-layer bidirectional LSTM to encode the sequence with $\mathbf{x}_k$ and $\mathbf{lm}_k$ feeding into the first and second layer, respectively. That is,

$$\mathbf{g}_k^{\rightarrow} = \text{LSTM}(\mathbf{x}_k, \mathbf{g}_{k-1}^{\rightarrow}), \mathbf{g}_k = [\mathbf{g}_k^{\rightarrow}; \mathbf{g}_k^{\leftarrow}],$$
$$\mathbf{h}_k^{\rightarrow} = \text{LSTM}([\mathbf{g}_k; \mathbf{lm}_k], \mathbf{h}_{k-1}^{\rightarrow}), \mathbf{h}_k = [\mathbf{h}_k^{\rightarrow}; \mathbf{h}_k^{\leftarrow}],$$

where $\mathbf{g}_k^{\leftarrow}$ and $\mathbf{h}_k^{\leftarrow}$ are defined similarly to $\mathbf{g}_k^{\rightarrow}$ and $\mathbf{h}_k^{\rightarrow}$ but process token sequences in the opposite direction.

Similar to the model described in §3, we feed the output of the second LSTM into a dense layer to predict unnormalized label weights for each token and learn label bigram feature weights to account for dependencies between labels.

Results. We use the standard data splits of the SemEval-2017 Task 10 on entity (and relation) extraction from scientific papers (Augenstein et al., 2017). Table 3 compares three variants of our entity extraction model. The first line omits the LM embeddings $\mathbf{lm}_k$, while the second line is the full model (including LM embeddings) showing a large improvement of 4.2 F1 points. The third line shows that creating an ensemble of 15 models further improves the results by 1.1 F1 points.

Model instances. In the deployed system, we use three instances of the entity extraction model with a similar architecture, but trained on different datasets. Two instances are trained on the BC5CDR (Li et al., 2016) and the CHEMDNER datasets (Krallinger et al., 2015) to extract key entity mentions in the biomedical domain such as diseases, drugs and chemical compounds. The third instance is trained on mention labels induced from Wikipedia articles in the computer science domain. The output of all model instances are pooled together and combined with the rule-based entity extraction module, then fed into the entity linking model (described below).

Description	F1
Without LM	49.9
With LM	54.1
Avg. of 15 models with LM	55.2

Table 3: Results of the entity extraction model on the development set of SemEval-2017 task 10.

4.3 Knowledge Bases

In this section, we describe the construction of entity nodes and entity-entity edges. Unlike other knowledge extraction systems such as the Never-Ending Language Learner (NELL)[7] and OpenIE 4,[8] we use existing knowledge bases (KBs) of entities to reduce the burden of identifying coherent concepts. Grounding the entity mentions in a manually-curated KB also increases user confidence in automated predictions. We use two KBs:

UMLS: The UMLS metathesaurus integrates information about concepts in specialized ontologies in several biomedical domains, and is funded by the U.S. National Library of Medicine.

DBpedia: DBpedia provides access to structured information in Wikipedia. Rather than including all Wikipedia pages, we used a short list of Wikipedia categories about CS and included all pages up to depth four in their trees in order to exclude irrelevant entities, e.g., "Lord of the Rings" in DBpedia.

4.4 Entity Linking Models

Given a text span s identified by the entity extraction model in §4.2 (or with heuristics) and a reference KB, the goal of the entity linking model is to associate the span with the entity it refers to. A span and its surrounding words are collectively

[7]`http://rtw.ml.cmu.edu/rtw/`

[8]`https://github.com/allenai/openie-standalone`

referred to as a mention. We first identify a set of candidate entities that a given mention may refer to. Then, we rank the candidate entities based on a score computed using a neural model trained on labeled data.

For example, given the string "... *database of facts, an ILP system will* ...", the entity extraction model identifies the span "ILP" as a possible entity and the entity linking model associates it with "Inductive_Logic_Programming" as the referent entity (from among other candidates like "Integer_Linear_Programming" or "Instruction-level_Parallelism").

Datasets. We used two datasets: i) a biomedical dataset formed by combining MSH (Jimeno-Yepes et al., 2011) and BC5CDR (Li et al., 2016) with UMLS as the reference KB, and ii) a CS dataset we curated using Wikipedia articles about CS concepts with DBpedia as the reference KB.

Candidate selection. In a preprocessing step, we build an index which maps any token used in a labeled mention or an entity name in the KB to associated entity IDs, along with the frequency this token is associated with that entity. This is similar to the index used in previous entity linking systems (e.g., Bhagavatula et al., 2015) to estimate the probability that a given mention refers to an entity. At train and test time, we use this index to find candidate entities for a given mention by looking up the tokens in the mention. This method also serves as our baseline in Table 4 by selecting the entity with the highest frequency for a given mention.

Scoring candidates. Given a mention (m) and a candidate entity (e), the neural model constructs a vector encoding of the mention and the entity. We encode the mention and entity using the functions **f** and **g**, respectively, as follows:

$$\mathbf{f}(\mathrm{m}) = [\mathbf{v}_{\mathrm{m.name}}; \mathrm{avg}(\mathbf{v}_{\mathrm{m.lc}}, \mathbf{v}_{\mathrm{m.rc}})],$$
$$\mathbf{g}(\mathrm{e}) = [\mathbf{v}_{\mathrm{e.name}}; \mathbf{v}_{\mathrm{e.def}}],$$

where m.surface, m.lc and m.rc are the mention's surface form, left and right contexts, and e.name and e.def are the candidate entity's name and definition, respectively. $\mathbf{v}_{\mathrm{text}}$ is a bag-of-words sum encoder for text. We use the same encoder for the mention surface form and the candidate name, and another encoder for the mention contexts and entity definition.

Additionally, we include numerical features to estimate the confidence of a candidate entity based on the statistics collected in the index described earlier. We compute two scores based on the word overlap of (i) mention's context and candidate's definition and (ii) mention's surface span and the candidate entity's name. Finally, we feed the concatenation of the cosine similarity between **f**(m) and **g**(e) and the intersection-based scores into an affine transformation followed by a sigmoid nonlinearity to compute the final score for the pair (m, e).

	CS	Bio
Baseline	84.2	54.2
Neural	84.6	85.8

Table 4: The Bag of Concepts F1 score of the baseline and neural model on the two curated datasets.

Results. We use the Bag of Concepts F1 metric (Ling et al., 2015) for comparison. Table 4 compares the performance of the most-frequent-entity baseline and our neural model described above.

5 Other Research Problems

In the previous sections, we discussed how we construct the main components of the literature graph. In this section, we briefly describe several other related challenges we are actively working on.

Author disambiguation. Despite initiatives to have global author IDs ORCID and ResearcherID, most publishers provide author information as names (e.g., arXiv). However, author names cannot be used as a unique identifier since several people often share the same name. Moreover, different venues and sources use different conventions in reporting the author names, e.g., "first initial, last name" vs. "last name, first name". Inspired by Culotta et al. (2007), we train a supervised binary classifier for merging pairs of author instances and use it to incrementally create author clusters. We only consider merging two author instances if they have the same last name and share the first initial. If the first name is spelled out (rather than abbreviated) in both author instances, we also require that the first name matches.

Ontology matching. Popular concepts are often represented in multiple KBs. For example, the concept of "artificial neural networks" is represented as entity ID D016571 in the MESH ontology, and represented as page ID '21523' in DBpedia. Ontology matching is the problem of identifying semantically-equivalent entities across KBs or ontologies.[9]

Limited KB coverage. The convenience of grounding entities in a hand-curated KB comes at the cost of limited coverage. Introduction of new concepts and relations in the scientific literature occurs at a faster pace than KB curation, resulting in a large gap in KB coverage of scientific concepts. In order to close this gap, we need to develop models which can predict textual relations as well as detailed concept descriptions in scientific papers. For the same reasons, we also need to augment the relations imported from the KB with relations extracted from text. Our approach to address both entity and relation coverage is based on distant supervision (Mintz et al., 2009). In short, we train two models for identifying entity definitions and relations expressed in natural language in scientific documents, and automatically generate labeled data for training these models using known definitions and relations in the KB.

We note that the literature graph currently lacks coverage for important entity types (e.g., affiliations) and domains (e.g., physics). Covering affiliations requires small modifications to the metadata extraction model followed by an algorithm for matching author names with their affiliations. In order to cover additional scientific domains, more agreements need to be signed with publishers.

Figure and table extraction. Non-textual components such as charts, diagrams and tables provide key information in many scientific documents, but the lack of large labeled datasets has impeded the development of data-driven methods for scientific figure extraction. In Siegel et al. (2018), we induced high-quality training labels for the task of figure extraction in a large number of scientific documents, with no human intervention. To accomplish this we leveraged the auxiliary data provided in two large web collections of scientific documents (arXiv and PubMed) to locate figures and their associated captions in the rasterized PDF. We use the resulting dataset to train a deep neural network for end-to-end figure detection, yielding a model that can be more easily extended to new domains compared to previous work.

Understanding and predicting citations. The citation edges in the literature graph provide a wealth of information (e.g., at what rate a paper is being cited and whether it is accelerating), and opens the door for further research to better understand and predict citations. For example, in order to allow users to better understand what impact a paper had and effectively navigate its citations, we experimented with methods for classifying a citation as important or incidental, as well as more fine-grained classes (Valenzuela et al., 2015). The citation information also enables us to develop models for estimating the potential of a paper or an author. In Weihs and Etzioni (2017), we predict citation-based metrics such as an author's h-index and the citation rate of a paper in the future. Also related is the problem of predicting which papers should be cited in a given draft (Bhagavatula et al., 2018), which can help improve the quality of a paper draft before it is submitted for peer review, or used to supplement the list of references after a paper is published.

[9]Variants of this problem are also known as deduplication or record linkage.

6 Conclusion and Future Work

In this paper, we discuss the construction of a graph, providing a symbolic representation of the scientific literature. We describe deployed models for identifying authors, references and entities in the paper text, and provide experimental results to evaluate the performance of each model.

Three research directions follow from this work and other similar projects, e.g., Hahn-Powell et al. (2017); Wu et al. (2014): i) improving quality and enriching content of the literature graph (e.g., ontology matching and knowledge base population). ii) aggregating domain-specific extractions across many papers to enable a better understanding of the literature as a whole (e.g., identifying demographic biases in clinical trial participants and summarizing empirical results on important tasks). iii) exploring the literature via natural language interfaces.

In order to help future research efforts, we make the following resources publicly available: metadata for over 20 million papers,[10] meaningful citations dataset,[11] models for figure and table extraction,[12] models for predicting citations in a paper draft [13] and models for extracting paper metadata,[14] among other resources.[15]

[10]`http://labs.semanticscholar.org/corpus/`

[11]`http://allenai.org/data.html`

[12]`https://github.com/allenai/deepfigures-open`

[13]`https://github.com/allenai/citeomatic`

[14]`https://github.com/allenai/science-parse`

[15]`http://allenai.org/software/`

References

Waleed Ammar, Matthew E. Peters, Chandra Bhagavatula, and Russell Power. 2017. The ai2 system at semeval-2017 task 10 (scienceie): semi-supervised end-to-end entity and relation extraction. In *ACL workshop (SemEval)*.

Isabelle Augenstein, Mrinal Das, Sebastian Riedel, Lakshmi Vikraman, and Andrew D. McCallum. 2017. Semeval 2017 task 10 (scienceie): Extracting keyphrases and relations from scientific publications. In *ACL workshop (SemEval)*.

Chandra Bhagavatula, Sergey Feldman, Russell Power, and Waleed Ammar. 2018. Content-based citation recommendation. In *NAACL*.

Chandra Bhagavatula, Thanapon Noraset, and Doug Downey. 2015. TabEL: entity linking in web tables. In *ISWC*.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel P. Kuksa. 2011. Natural language processing (almost) from scratch. In *JMLR*.

Aron Culotta, Pallika Kanani, Robert Hall, Michael Wick, and Andrew D. McCallum. 2007. Author disambiguation using error-driven machine learning with a ranking loss function. In *IIWeb Workshop*.

Hal Daumé. 2007. Frustratingly easy domain adaptation. In *ACL*.

Dina Demner-Fushman, Willie J. Rogers, and Alan R. Aronson. 2017. MetaMap Lite: an evaluation of a new Java implementation of MetaMap. In *JAMIA*.

Oren Etzioni. 2011. Search needs a shake-up. *Nature* 476 7358:25–6.

Paolo Ferragina and Ugo Scaiella. 2010. TAGME: on-the-fly annotation of short text fragments (by wikipedia entities). In *CIKM*.

Gus Hahn-Powell, Marco Antonio Valenzuela-Escarcega, and Mihai Surdeanu. 2017. Swanson linking revisited: Accelerating literature-based discovery across domains using a conceptual influence graph. In *ACL*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* .

Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, Jayant Krishnamurthy, and Luke S. Zettlemoyer. 2017. Learning a neural semantic parser from user feedback. In *ACL*.

Antonio J. Jimeno-Yepes, Bridget T. McInnes, and Alan R. Aronson. 2011. Exploiting mesh indexing in medline to generate a data set for word sense disambiguation. *BMC bioinformatics* 12(1):223.

Martin Krallinger, Florian Leitner, Obdulia Rabal, Miguel Vazquez, Julen Oyarzabal, and Alfonso Valencia. 2015. CHEMDNER: The drugs and chemical names extraction challenge. In *J. Cheminformatics*.

Guillaume Lample, Miguel Ballesteros, Sandeep K Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *HLT-NAACL*.

Jiao Li, Yueping Sun, Robin J. Johnson, Daniela Sciaky, Chih-Hsuan Wei, Robert Leaman, Allan Peter Davis, Carolyn J. Mattingly, Thomas C. Wiegers, and Zhiyong Lu. 2016. Biocreative v cdr task corpus: a resource for chemical disease relation extraction. *Database : the journal of biological databases and curation* 2016.

Xiao Ling, Sameer Singh, and Daniel S. Weld. 2015. Design challenges for entity linking. *Transactions of the Association for Computational Linguistics* 3:315–328.

Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *ACL*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global vectors for word representation. In *EMNLP*.

Matthew E. Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. 2017. Semi-supervised sequence tagging with bidirectional language models. In *ACL*.

Noah Siegel, Nicholas Lourie, Russell Power, and Waleed Ammar. 2018. Extracting scientific figures with distantly supervised neural networks. In *JCDL*.

Marco Valenzuela, Vu Ha, and Oren Etzioni. 2015. Identifying meaningful citations. In *AAAI Workshop (Scholarly Big Data)*.

Xiang Wang, Yan Dong, Xiang qian Qi, Yi-Ming Li, Cheng-Guang Huang, and Lijun Hou. 2013. Clinical review: Efficacy of antimicrobial-impregnated catheters in external ventricular drainage - a systematic review and meta-analysis. In *Critical care*.

Luca Weihs and Oren Etzioni. 2017. Learning to predict citation-based impact measures. In *JCDL*.

Jian Wu, Kyle Williams, Hung-Hsuan Chen, Madian Khabsa, Cornelia Caragea, Alexander Ororbia, Douglas Jordan, and C. Lee Giles. 2014. CiteSeerX: AI in a digital library search engine. In *AAAI*.

Chenyan Xiong, Russell Power, and Jamie Callan. 2017. Explicit semantic ranking for academic search via knowledge graph embedding. In *WWW*.