# Lessons from Deep Learning applied to Scholarly Information Extraction: What Works, What Doesn't, and Future Directions

Raquib Bin Yousuf *
Subhodip Biswas*
Computer Science
Virginia Tech
Arlington, VA, USA
raquib@vt.edu
subhodip@vt.edu

Kulendra Kumar Kaushal†
Bloomberg
New York City, New York, USA
kulendra@vt.edu

James Dunham
Rebecca Gelles
CSET
Georgetown University
Washington, D.C. USA
james.dunham@georgetown.edu
rebecca.gelles@georgetown.edu

Sathappan Muthiah†
eBay
San Jose, California USA
sathap1@vt.edu

Nathan Self
Patrick Butler
Naren Ramakrishnan
Computer Science
Virginia Tech
Arlington, VA, USA
nwself@vt.edu
pabutler@vt.edu
naren@cs.vt.edu

## ABSTRACT

Understanding key insights from full-text scholarly articles is essential as it enables us to determine interesting trends, give insight into the research and development, and build knowledge graphs. However, some of the interesting key insights are only available when considering full-text. Although researchers have made significant progress in information extraction from short documents, extraction of scientific entities from full-text scholarly literature remains a challenging problem. This work presents an automated **En**d-to-end **R**esearch Entity **Ex**tractor called **EneRex** to extract technical facets such as dataset usage, objective task, method from full-text scholarly research articles. Additionally, we extracted three novel facets, e.g., links to source code, computing resources, programming language/libraries from full-text articles. We demonstrate how **EneRex** is able to extract key insights and trends from a large-scale dataset in the domain of computer science. We further test our pipeline on multiple datasets and found that the **EneRex** improves upon a state of the art model. We highlight how the existing datasets are limited in their capacity and how **EneRex** may fit into an existing knowledge graph. We also present a detailed discussion with pointers for future research. Our code and data are publicly available at https://github.com/DiscoveryAnalyticsCenter/EneRex .

## CCS CONCEPTS

• **Information systems** → *Information retrieval*; **Retrieval tasks and goals**.

## KEYWORDS

full-text information extraction, scholarly literature, deep learning, knowledge graph

## 1 INTRODUCTION

The number of scientific scholarly articles published each year is staggeringly high and continues to rise. According to DBLP, 400k articles were published in Computer science(CS)-based research areas in 2020 alone. Extracting key scientific insights from these papers is imperative for understanding emerging technologies, their prevalence, and relationships, and for enabling analysts and policymakers to identify key trends. Information extraction of these entities from large-scale datasets would facilitate the creation of structured knowledge graphs. Existing work builds these knowledge graphs from citation graphs and clusters, coupled with the classification of the papers by various conferences or libraries, such as the CSET Map of Science[13, 22] and the Microsoft Academic Graph [25]. These knowledge graphs can be used to discover clusters of papers belonging to topics of research. Recently, researchers have been attempting to automatically classify documents [11] and discover clusters of paper related to a specific task [4]. Moreover, such knowledge graphs are already in use helping policy makers look into research funding practices in artificial intelligence [22]. We believe that actual scientific entities from the papers would complement the existing citation-based knowledge graphs with more in-depth knowledge and further enrich the information. Motivated by this, we propose an information extraction pipeline for extracting technical entities from the full text of research articles, allowing us to establish trends present in large scholarly databases, particularly in the domain of CS.

Identifying salient scientific facets, or entity types, and extracting them from scholarly articles is an important research endeavor in the information retrieval community [14, 17]. Extraction involves encoding texts, identifying sections where relevant information is present, and then extracting the required information in a structured format. Most research work in this area has traditionally involved working with machine-readable metadata such as title, abstract, etc. [16, 26]. However, many important facets can only be extracted when the full text is taken into consideration since such information is not available in the metadata alone. A recent survey paper concluded that the majority of work on key insight extraction uses abstracts only [20]. They also concluded that the primary challenge of full-text analytics is that the complexity of the manual annotation processes grows as the dataset grows. Identifying which section of an article contains information relevant to a given entity type is challenging since much of the text data is irrelevant to that particular entity type. Previous work has focused primarily on information extraction from shorter documents like news articles, blogs, user posts, and comments on different social media platforms [5, 9, 24]. Instead, we focused on developing a

---

*Equal contribution. Author names arranged in reverse alphabetical order.
†The work was done when the author was at Virginia Tech.

full-text research entity extraction system which will enable us to discover trends in computer science research.

The scarcity of large-scale data to train and evaluate our models remains a challenge. Many publicly available, labeled datasets contain annotations for small documents. However, there is a lack of ground truth data for full-text scientific articles. We tackled this issue with an automated data generation process based on syntactic patterns to generate training data. Recently, there have been attempts to extract information from full-text scholarly articles and create full-text datasets that can help train and build full-text extraction modules, for instance, see [15]. However, existing datasets are limited in their capacity. In some cases, ground truth data does not have all the entities used for every paper. Moreover, to the best of our knowledge, three of our facets (links to source code, computing resources, and language/library used) have not been extracted in prior work, so no ground truth datasets are available for those facets.

In this work, we introduced a DL-based automated tool named End-to-end Research Entity Extractor (**EneRex**) to extract from a paper's full text six entity types: links to source code, names of any datasets used, the paper's objective task, the method by which the paper attempts the object task, how many computing resources were required, which programming languages were used, and which programming libraries were used. We ran **EneRex** on a CS-based large-scale scholarly dataset and used the extracted information to discover trends and insights about computer science research areas. We discussed how **EneRex** can fit into an existing knowledge graph to complement that information. We also came up with a novel dataset for computing resources and language/library (CORLL). **EneRex** uses transfer learning to generate scientific contextual embeddings from SciBERT [7], which enables it to learn a better representation of scholarly articles. SciBERT extends the BERT model [12] and has been trained on scientific texts. **EneRex** was able to improve upon SciREX [15] which is the only model aiming to extract similar facets from full-text scholarly articles, to the best of our knowledge. In addition to the facets provided by SciREX, we also extracted computing resources and any languages or libraries used in a paper. **EneRex** is computationally simple with different components for handling salience and entities. For generating training data, **EneRex** does not need manual annotations for all of the facets, and SciREX requires more involved annotation efforts for all the facets. Table 1 shows a comparison between existing models and **EneRex**. An example of **EneRex** in practice is shown in Figure 1.

The remainder of the paper is organized as follows. Section 2 discusses the scientific entities in research articles and previous works to identify those. We introduce the **EneRex** system in Section 3 followed by evaluation in Section 4. Our system is used to
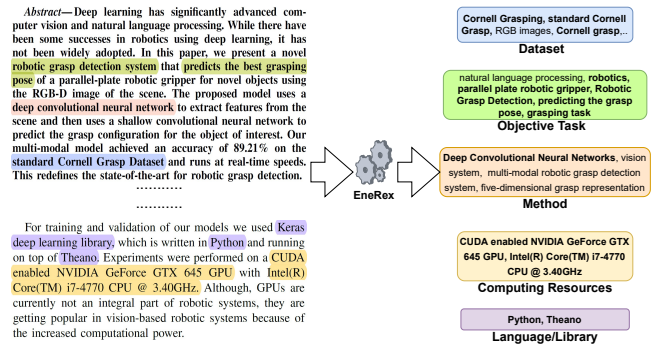


**Figure 1: EneRex in practice.**

discover trends in computer science scholarly articles in Section 5. Finally, in Section 6 we present some observations and conclude the paper in Section 7.

## 2 SCIENTIFIC FACETS IN RESEARCH ARTICLES AND RELATED WORKS

The traditional facets extracted by the information retrieval community from scientific papers have been objective task, method, and material used. These three facets are known to be helpful to researchers for quickly comprehending the main thrust of a scientific article. Gupta et al. [14] attempted to discover the focus, technique and domain of research articles via pattern matching in a bootstrapping manner. Tsai et al. [29] also used bootstrapping to identify two categories of concepts: techniques and application.

More recent approaches consist of applying neural models to identify entities and their relations. Luan [17] used neural models with semi-supervised learning to extract entities and their relationships. Mesbah et al. [19] developed NER models to detect scientific datasets and methods in an iterative manner. These automatic scientific entity extraction tasks revolved around using only the abstracts of scholarly articles [6]. SciERC [18] and Dygie++ [31] are two such models which work on a dataset of 500 annotated abstracts. Nasar et al. [20] details past research which extracted facets such as problems addressed, domain, tools, and evaluation measures. The SciREX [15] model works with full text documents in an end-to-end fashion, in comparison to previous research approaches. Our model adds three more features and improves upon the results from SciREX.

In an effort to improve reproducibility, researchers have increasingly been publishing their code along with their papers. Papers with Code[3] collects state-of-the-art papers for some research

**Table 1: Comparison of EneRex with similar work**

| Method | Extract Dataset | Extract Objective Task | Extract Method | Full Text Extraction | Extract Computing Resources | Extract Language Library | Extract Links to Source Code | No Need for Annotation | Simplified Pipeline |
|---|---|---|---|---|---|---|---|---|---|
| SciREX[15] | ✓ | ✓ | ✓ | ✓ | | | | | |
| **EneRex** | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | partially | ✓ |

categories that have been published with source code. The structured facets available from this website are dataset, metric, task and method.

Researchers have shown that complex natural language processing models like GPT [21], BERT [12], Turing-NLG [23] with billions of parameters are generally more efficient at solving challenging problems like information extraction, machine translation, and natural language generation. Of course, the power consumed and the required computing resources in training such complex models is an ever-present issue [8]. In this work, we aim to extract different computing resource entities which could help us track the efficiency of different computing resources across different neural networks. Furthermore, several libraries, such as PyTorch, TensorFlow, and Caffee are available for different programming languages like Python, Java, and MATLAB. Tracking the usage of different languages and library dependencies across research articles could help researchers determine the best possible combination and help analyze usage trends. All of the six facets extracted in our system are listed in Table 2.

**Table 2: List of facets for extraction**

| Facets |
| --- |
| Source Code Links |
| Dataset used |
| Objective Task |
| Method used |
| Computing Resources |
| Language/library |

## 3 THE ENEREX SYSTEM

**EneRex** is capable of identifying and extracting six facets from a full-text scholarly article. Extraction for these six facets is grouped into two different pipelines in terms of the adopted methodology, depending on context and other syntactic properties. The first group (source code link, dataset, computing resource and language/library) was extracted by a weakly-supervised learning task. Here, automated labeling of sentences and entities created a noisy ground truth set. For this group, **EneRex** identifies entities in two steps: i) identifying the relevant sentences; ii) identifying the entities from the selected sentences. The second group (objective task and method) was extracted with the help of transfer learning. Each of the facets were designed with syntactic properties in mind. The training and prediction modules were kept separate for simplicity, thereby adopting batch training as the default knowledge ingestion process. However, all of the facet extraction tasks share a full-text extraction and ingestion module. In order to build a structured representation from an article's full text, **EneRex** can extract full-text from PDF, plain text, and JSONL. Subsequent entity extraction submodules use this structured representation of the article to drive training and inference. The full-text extraction module is described first, followed by the details of each facet and module. A conceptional overview of **EneRex** architecture in inference is shown on Figure 2.

### 3.1 Data Ingestion

**EneRex** ingests PDFs of scholarly articles. To build a large dataset of PDF files, we collected PDF files and metadata from arXiv[1], an open-access research article distribution service. The arXiv corpus contains about 2 million scholarly articles in many fields. Since research articles published in fields such as physics or astronomy are not relevant to our entity extraction tasks, we collect only articles published in fields related to Computer Science. Using publication metadata and "cs." tags from the arXiv dataset to filter out documents from other disciplines, we collected PDFs and metadata for 241,646 articles. The full-text and metadata from each document were ingested using Grobid [2] reference parsing tools which can extract, among other things, headers, references, citation contexts, and authors from article text. Tkaczyk et al. [27] evaluated various reference parsing tools. Among those that handle full article text, Grobid was the best performing, followed by Cermine [28]. Grobid successfully processed 240,051 documents, about 99% and we generated structured representations with metadata, sections, bibliography entries, and footnotes for all papers.

### 3.2 Training Data Generation

For the task of extracting these facets from scholarly articles, the main challenge is the lack of ground truth available for training inference models. We tackled this issue by using weakly supervised learning tasks. For the first group of facets (source code link, dataset, computing resource and language/library), we extracted salient ground truth by following the lexical and syntactic properties of the sentences. These facets required annotations on different levels to build classifiers and named entity recognizers (NERs). The patterns were designed by taking advantage of sentence-level dependency parsing and the part-of-speech (POS) tags on each word. The goal of this automated extraction pipeline is to require the least human effort to extract each facet. However, the quality of this automated pattern based extraction varied by facets. For the source code and dataset facets, this pattern-based extraction produced less noise than computing resources and language/library, for which the pipeline produced significant noise in the entity level annotation. After identifying candidate sentences, a follow-up manual annotation process was carried out to identify the entities within the sentences for these two facets.

*Facet 1: Source Code Links.* To extract references to source code in a paper, our data generation script first selects any sentences with references or footnotes that contain URLs. Using spaCy [30] for dependency parsing obtains universal dependency relation tags for the selected sentences and selects a contextual location of a term (i.e., subject, object, root, etc.). Any sentence with at least two patterns and three occurrences is considered a candidate sentence. The sample patterns are shown in Appendix C . Special importance was given to adjective and object patterns. Sentences containing certain common words such as figure and table were excluded. These constraints were decided by empirical analysis of the results. This algorithm created a set of sentences that were used to train a sentence classifier for this facet.

*Facet 2: Dataset Used.* This process is similar to the source code process except the number of patterns and their templates are
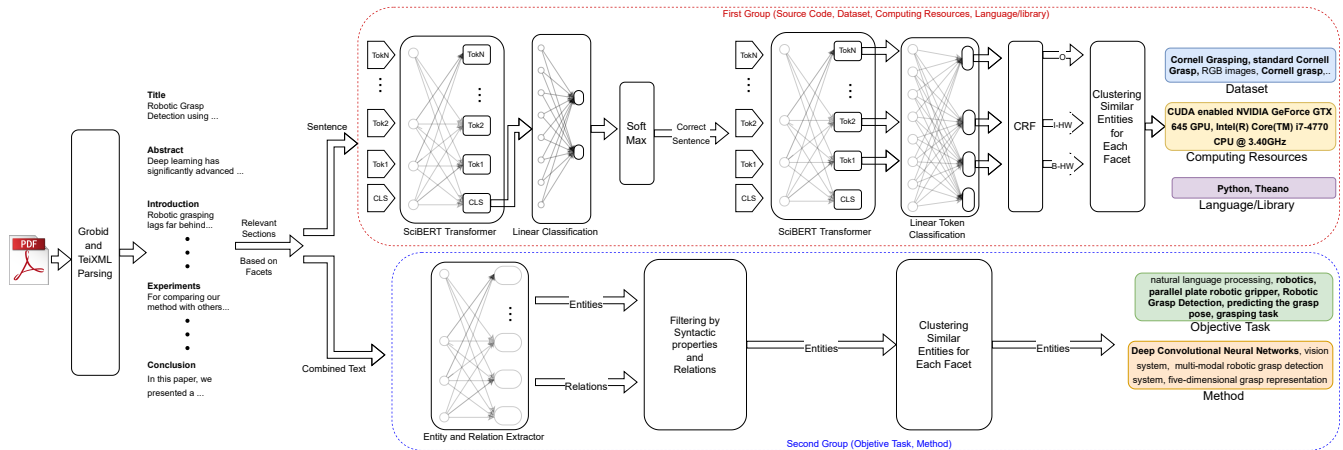
**Figure 2: The architecture of the EneRex system.**

different. There are ten patterns for extracting mentions of datasets used in a paper; seven of which depend on dependency relation tags, as shown in Appendix C. The rest are based on whether a sentence contains a reference, footnote, URL, or well-known dataset name. As a first step, the pipeline extracts sentences containing words about using dataset materials (e.g., dataset, corpus, database etc.) and checks the next five sentences against our patterns. In practice, sentences that contain the word "dataset" often do not contain details of the dataset name or its usage, reserving the actual mention for subsequent sentences. We identified dataset entities by following several heuristic rules: a dataset name must be a noun or noun phrase, must start with capital, but may end with digits. We used the shortest dependency path to assign scores to each candidate entity and empirically decided on the threshold for selecting a candidate sentence and candidate entity. These were then used to train a sentence classifier and NER for this facet.

*Facet 5 and 6: Computing Resources, Programming Language/Library.* For these facets, our data generation script selects sentences which contain certain seed words from a set we curated. Then, Natural Language Toolkit (NLTK) is used to tokenize, lemmatize, and remove stop words from each sentence. The algorithm extracts patterns using the part-of-speech (POS) tags and handcrafted rules to identify new candidate seed words based on these patterns. Each candidate seed word is assigned a score. If the score is over a certain threshold, the candidate seed word is appended to the original set of seed words, otherwise it is discarded. After identifying the candidate sentences, we tried an automated extraction of the entities for these two facets. However, the results were not satisfactory so we carried out a manual annotation of the entities within the sentences for training the named entity recognition (NER) model for these two facets. In this process, we tagged 600 sentences with both computing resource and language/library entities.

***CORLL***: *A Novel Dataset with Computing Resources and Language/Library Entities.* As there is no previous work on extracting hardware-related entities, we propose a novel NER dataset for computing resources and language/library. **CORLL** is comprised of sentences containing entities that have been used and not just

mentioned in the research article, and the text span in which the entity is present in the BILUO format. The dataset contains over 600 such salient sentences and around 1400 annotated entities overall. Furthermore, the computing resources facet is divided into computing platform, compute time, and hardware resources entities. Table 3 shows the distribution of different facets in the **CORLL** dataset, from which we can discern that some facets, like compute time, are sparse and therefore may be challenging to extract.

**Table 3: Facets and their Distribution in CORLL**

| Facets | Count |
|---|---|
| Compute Platform | 181 |
| Compute Time | 51 |
| Hardware Resources | 576 |
| Programming Language | 367 |
| Programming Library | 168 |

## 3.3 Model Training

For each of these four facets (source code links, dataset, computing resources, language/library), we trained separate models to identify related sentences and entities. In general, an input sentence is first passed through a SciBERT transformer model to create contextual embeddings. Then a linear sequence classification layer is trained on top of that to identify it as a "correct" or "incorrect" sentence for the respective facet. The second step is to train a NER model. For source code, the correct sentences are sufficient. For the dataset, computing resources, language/library facets, we trained three named entity recognizers. The ground truth data for each entity are first tokenized using spaCy. The tokens are tagged in BILUO format (beginning, inside, last, unit, outside) and trained using a BERT + CRF based model. The model layers are initialized with the SciBERT model, followed by token classification layer and a CRF tagger.

During prediction, we take each sentence of a paper and feed it through the sentence classifier first. The selected sentences are then fed through the NER to get the final entity list. For datasets, there can be multiple mentions of the entities in a paper so a single

dataset name can come up through multiple sentences in the system. So we further clustered similar entities for each scholarly article. For source code links, instead of an NER module, correct sentences go through a URL extraction algorithm involving the sentence and any footnotes the sentence references.

*3.3.1 Objective Task and Method.* For the other main group of facets (objective task and method), candidate entities were extracted using a state-of-the-art NER model and salience was achieved by pruning the entities using syntactic properties. In particular, **EneRex** took advantage of the structured full-text representation to focus this process on only the introduction, conclusion, and similar sections. This allows more salient entities and relations to be extracted, thereby increasing the chances of finding a mentioned task or method in a document. We observed that a task and method pair often appear in a sentence connected by a 'USED-FOR' relation. A task (and/or method) may be connected to the method (and/or task) through other general or specific terms (entity or not) by 'PART-OF', 'FEATURE-OF', or 'HYPONYM-OF' relations. The entities and the relations are extracted by adopting a scientific entity and relation extractor described by Wadden et al. [31]. We followed these heuristics and an exhaustive search algorithm with our above hypothesis to find the most salient objective task and method for an article. **EneRex** clustered similar entities using fuzzy matching to get final task and method extractions for an article.

## 4 EMPIRICAL EVALUATION

We evaluate each of our facet extraction tasks separately against multiple gold datasets. Not all available gold datasets are suitable for our purposes since they are often designed to the needs of the researcher and community that generated them. Moreover, we could not find one encompassing all the facets we are looking for. E.g., SciREX dataset contains only three of our targeted facets. So, we extracted the appropriate facets from each dataset and matched our results against them. For computing resources and language/library dependencies, there is no established ground truth dataset available. Moreover, available datasets are limited in their capacity. We built in-house annotated ground truth sets to address such issues. The following subsections detail the results of component-based evaluation for some of the facets. We also compared **EneRex** with end-to-end predictions from available state-of-the-art extraction models.

### 4.1 Evaluation Metrics

For each facet, we calculated precision, recall, and macro F1 for every facet available in a particular dataset.For links to source code, if the extracted URL matched with ground truth URL the extraction is correct. Example of cases for links to source code is given in Appendix D For the dataset, objective task, and method facets, we compare the gold truth with our extracted entity clusters. We used fuzzy string matching with a threshold value of 0.85, empirically determined, to decide if the gold truth entity matched with any element of the clusters. We calculated recall as how many of the gold truth were extracted by the model and precision as correct extractions divided by total number of clusters.

### 4.2 Human Annotated Dataset

We annotated 145 artificial intelligence(AI) papers from 2014-2018, roughly 30 from each year with BRAT. We tested the extractions of source code link, dataset usage, computing resources and language/library against this ground truth dataset. The distribution of different facets in the dataset are given in Appendix B. For most papers, the first, or most prominent, related sentence was noted because sometimes there are too many sentences that mention an entity. On the contrary, the pipeline may extract any sentence that mentions an entity, which is often more than one. The metric values for all facets are presented in Table 4. A possible reason for low precision and recall on the language/library facet is that many entities in this set have too few characters, for instance the programming languages C and R which can be hard to differentiate from abbreviations and sentences with equations.

### 4.3 The "Papers with Code" dataset

The "Papers with Code" dataset contains state-of-the-art papers for some research categories and highlights trends in research along with source code. We tested the extraction of source code links and dataset usage against a subset of the Papers with Code dataset. The results are presented in Table 4.

For source code links, we randomly selected 980 papers from the nearly 15,000 papers with source code in the Papers with Code dataset. For dataset usage, we were able to download 701 papers from 764 papers with such annotations. However, the Papers with Code dataset has limitations that hamper the completeness of information available for each paper. Papers with Code maintains a list of datasets and lists papers underneath each one if the paper uses the dataset. This format often failed to capture all the dataset usage of a paper because either Papers with Code had no listing for a dataset used in a paper or papers simply are not listed under datasets which they did in fact use. For example, at least one paper in the corpus that uses four datasets (COCO, ImageNet, Kinetics, and Cityscapes) is listed under only one of those datasets in the Papers with Code database so taking Papers with Code as the ground truth shows only one dataset used in this paper: COCO. However, **EneRex** was able to pick up all four datasets from the paper. This gives our result a false impression of low precision.

### 4.4 Comparison with SciREX dataset and model

The SciREX system introduced a dataset with 438 papers and a model for information extraction from scientific articles. The dataset was built with entities from the "Papers with Code" dataset. The same limitations of the Papers with Code dataset apply here. It has four types of entities for each paper: method, metric, task and material. We evaluated our dataset, objective task, and method facets on a test set from the SciREX dataset. To make an end-to-end comparison between SciREX model and **EneRex**, we extracted "salient entity clusters" from SciREX. The "salient entity clusters" were then used in our evaluation process to generate evaluation metrics (section 4.1). The performance of SciREX model on SciREX dataset by our evaluation metrics are shown on the rightmost column of Table 4. We found that SciREX usually churns out too many entity clusters which increases the recall but reduces precision and

**Table 4: Evaluation of EneRex on different datasets**

| Facet | EneRex on Annotated Dataset | | EneRex on Papers with Code Dataset | | EneRex on SciREX Dataset | | SciREX on SciREX Dataset Salient Entity Clusters (By our evaluation method) | |
|---|---|---|---|---|---|---|---|---|
| | Precision | Recall | Precision | Recall | Precision | Recall | Precision | Recall |
| Source Code | 0.29 | 0.40 | 0.43 | 0.50 | N/A | | N/A | |
| Dataset | 0.58 | 0.77 | 0.37 | 0.71 | 0.53 | 0.79 | 0.39 | 0.88 |
| Objective Task | N/A | | N/A | | 0.21 | 0.59 | 0.17 | 0.85 |
| Method Used | N/A | | N/A | | 0.17 | 0.48 | 0.13 | 0.71 |
| Computing Resources | 0.34 | 0.53 | N/A | | N/A | | N/A | |
| Language/Library | 0.10 | 0.42 | N/A | | N/A | | N/A | |
| Macro P & R | 0.33 | 0.53 | 0.4 | 0.61 | 0.30 | 0.62 | 0.23 | 0.81 |
| Macro F1 | **0.41** | | **0.48** | | **0.40** | | 0.36 | |

hampers the overall macro F1 for all facets in a dataset. In contrast, **EneRex** performed well over all the facets with an 11% increase in the F1 value over SciREX.

## 5 TRENDS AND KEY INSIGHTS FROM SCHOLARLY LITERATURE

**EneRex** can be used to find trends and insights from a large number of papers. We attempted to answer several questions with the output of **EneRex** to showcase its utility and scalability. For source code links, dataset, method and task, the first four questions try to find existing trends by running our pipeline on the preprocessed arXiv CS papers. We also posed questions for usages of computing resource and language/library, specifically in the domain of AI. These questions are based on a subset of cs.AI papers from arXiv.

**Q1. What is the usage of GitHub as choice of platform to share code?** We looked into the output of the source code link to find out how many of the papers are using GitHub as platform to share source code. We found that the percentage of papers sharing source code links with GitHub is increasing each year. The trend line is presented in Figure 3a.

**Q2. Does one area of research share source code links more often than others?** To identify and designate a paper to a research area, we used the metadata available from arXiv, which tags each paper with several tags indicating the research areas. We calculated the total number of papers for each of the 40 computer science sub-categories and found the percentage of papers publishing a source code link. "Computation and language" has the highest percentage with 31.64%, followed by "computer vision" with 21.26%, "artificial intelligence" with 19.99%, and "machine learning" with 19.79%. Figure 3b summarizes the findings.

**Q3. What is the most used dataset? What are the top objective tasks, methods and other datasets used along with it?** We found out that MNIST is the most used dataset in our full arXiv corpus. In fact, papers that use MNIST have increased over time as shown in Figure 3c.

The top used datasets after MNIST are CIFAR and ImageNet. We present other top datasets used in these papers in Figure 3d. The top tasks for MNIST papers are computer vision, image classification,

and object detection. Similarly we found that the top methods are CNN, deep neural network, and GAN.

**Q4. What are the research trends in sentiment analysis? What are the top datasets used for this topic?** We created a trend line with the number of papers working on sentiment analysis, a classic research problem in the NLP community, over the years in Figure 3e. Among the papers, Twitter is the most used dataset, followed by Amazon, SemEval, IMDB, and Stanford. We presented these findings in Figure 3f.

**Q5. How much memory do researchers use in AI research?** **EneRex**'s computing resource output contains information on the amount of physical memory (RAM) and GPU memory used for simulations. We did not differentiate between the GPU memory and physical memory for this use case. So, this may be the maximum memory of the hardware platform used by each paper rather than the actual used memory for the systems. We found out that the median physical memory has been increasing almost each year until 2017. 2019 also saw same median and same third quartile values for memory used throughout the community which is 16GB and 32GB respectively. A boxplot is shown in Figure 3g.

**Q6. What is the market share of different hardware manufacturers in the AI research community?** We looked in our computing resource output specifically for Intel, Nvidia, and AMD. We excluded papers which mentioned two of the brands and considered papers using only one of these brands. We found that historically Intel has been the strongest in market share in CS research community. However, in 2018, Nvidia surpassed Intel as the most used hardware platform and the upward trend is continuing, possibly due to an increase in deep learning frameworks which rely on high GPU usage. On the other hand, AMD does not hold any significant share of the research platform. The trend is shown in Figure 3h.

**Q7. What is the relative usage of popular deep learning frameworks in AI research?** We compared the usage of two most popular deep learning frameworks: TensorFlow and PyTorch. The trend as paper count over time is presented in Figure 3i. Based on our findings, TensorFlow was more popular than PyTorch before 2018. Since then PyTorch has surpassed TensorFlow with a higher

growth rate and has continued its climb over the following year.

**Q8. Which programming language is the most popular in AI research?** We compare the number of papers for which we extracted Java or Python for the language/library facet in Figure 3j. Java was the more popular language until 2013. Python narrowly beat Java in 2014 and 2015 from which point Python started growing with a sharp upward trend. In 2019, four times as many papers mentioned Python as Java.

## 6 LESSONS LEARNT & FUTURE RESEARCH

Here we present lessons learnt during our attempt at entity extraction from full-text documents. Furthermore, the comparison with multiple dataset and state-of-the-art information extraction systems also pointed out some important aspects of existing systems in contrast to our approach. We discuss these issues below.

### 6.1 Lack of ground truth

The foremost difficulty involved in this type of full-text extraction is the lack of an established ground truth dataset which hampers both development and the evaluation of such a project. To the best of our knowledge, the SciREX and Papers with Code datasets are the only datasets in this area. As described, the Papers with Code dataset is not complete so could give a false impression of low precision. SciREX suffers from the same issues as it was built with Papers with Code entities. Moreover, three of our facets have not been well studied in the literature. Because of these limitations, we developed our own training data generator to train classifiers with a weakly supervised learning paradigm.

### 6.2 Issue with automated evaluation

We discovered that the ground truths in the Paper with Code dataset are generalized versions of the entities and these are traditionally decided by community members. The entities may not appear exactly as they do in the paper. SciREX was built with these same entities. For SciREX, we observed that 66.72% task entities, 61.16% method entities, and 84.12% material entities do not appear as the ground truth in the paper. However, **EneRex** can only extract what is in the text of the paper. Consequently, for some papers, the automated evaluation system failed to correctly capture the comparison between ground truth and **EneRex**-extracted entities. We partially solved this issue by adopting a fuzzy string matching based evaluation process (see Section 4.1) and cleaning the ground truth values.

### 6.3 Full Text vs Abstract vs Specific Sections

The choice of the input for each facet should be selected following the usual structure of scholarly articles. We ingested full text to create a representation which is structured by sections. Firstly, using full text is necessary for the extraction of some facets as they are generally only mentioned in the full text, especially source code, dataset usage, hardware related entities. Secondly, working with full text in machine learning models is sometimes too computation-heavy. Moreover, some sections related to methodological or mathematical components generally introduce noise for some facets. For objective tasks and methods, researchers tend to summarize and

mention the main objective tasks and goal in the abstract, introduction and conclusion sections. With a sample set of papers, we checked the extraction rate of **EneRex** when processing abstracts alone vs processing abstracts, introductions, and conclusions and found that the inclusion of introductions and conclusions improves the extraction rate by 26%. We therefore struck a balance between these practical findings and only used the sections that were most likely to contain the specific information we seek for each facet.

### 6.4 Full Text Ingestion

The most common digital object format is PDF but it does not save any structural information along with its graphical representation. Even using a state-of-the-art PDF ingestion system (Grobid), we found instances where footnotes were not extracted properly. References and URLs were also often incorrectly extracted. Improvements to this part of the ingestion system, may also improve **EneRex**'s performance.

### 6.5 Entity Variations and Sub-Entities

**EneRex** often picks up different variations of the same entity. For instance, datasets can be mentioned in abbreviated form or full form or may have designated subsets. Objective tasks can be phrased in different ways. Moreover, each entity can have multiple sub-entities. We solved this issue by clustering such lists of entities and considering them a single entity. However, we did not particularly identify the sub-entities and main entity those belong to. Ideally, we would like to map each variation of an entity to a standard entity list but making this entity list remains as a future task.
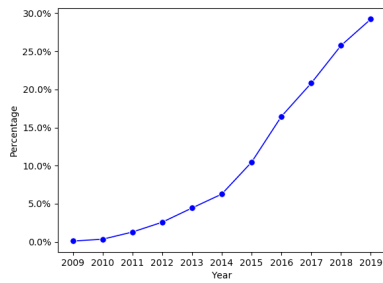
### 6.6 Division of Tasks and Computational Load

Instead of running a single pipeline to churn out all the facets at once, we divided our pipeline into manageable modules. The sentence classifier part detects salient sentences first and those sentences are fed into an NER module to extract word level entities. Moreover, we further divided our pipeline into separate parallel executable modules for each facet. This keeps our pipeline simpler and annotation effort can be precise for each facet as required. However, by doing this and not merging the facets, we lose relationships between them. Only a few of our facets have a obvious relationship to each other such as computing resources and language/library can be a group which sometimes have a "feature-of" or "used" relationship within the entities. Similarly dataset, objective task and method could be combined in a relationship. We leave this task for future expansion.
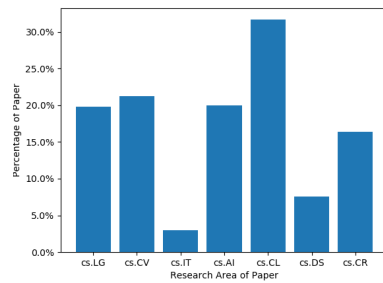
### 6.7 Knowledge Graph and Application

We posit that our extracted facets can be used to enrich existing knowledge graphs and create new layers of information. Here we will show one use case of how researchers are using such knowledge graphs and where our developed **EneRex** fits into equation.
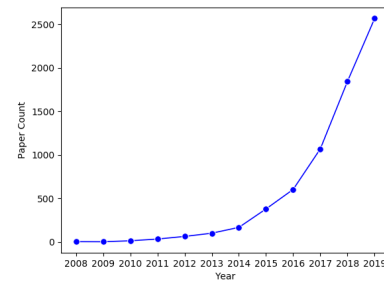
A recent article from a policy research organization focused on measuring the development of several AI-related topics: "re-identification", "speaker recognition", and "image synthesis" [10]. The authors singled out the CSET Map of Science[22] clusters for each topic using seed papers and carried out manual analysis to map out progress in each of these topics. However, the authors noted
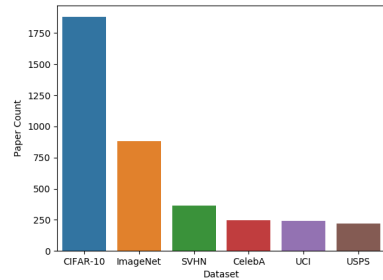
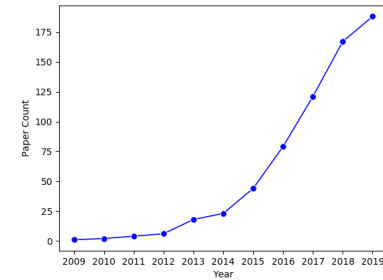(a) Papers with source code hosted on GitHub.



(b) Percentage of papers containing source code links for various research areas.
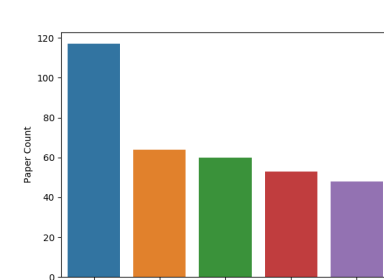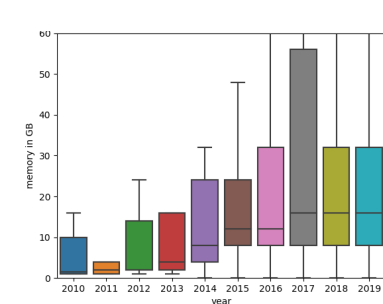


(c) Papers per year using MNIST.



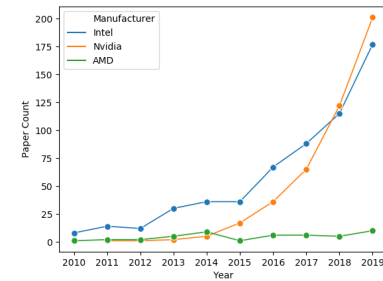(d) Top datasets associated with the MNIST dataset.



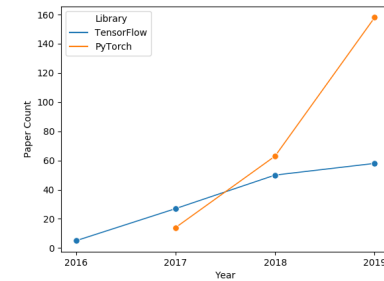(e) Count of papers on sentiment analysis over time.



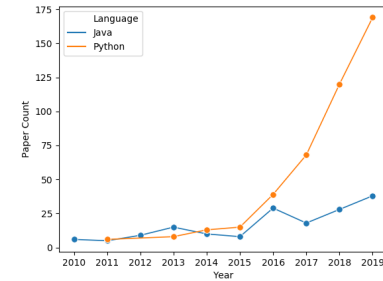(f) Top datasets for sentiment analysis.



(g) Hardware memory by year.



(h) Popularity of hardware manufacturers by paper count.



(i) Deep learning library usage (Tensor-Flow vs PyTorch)



(j) Programming language usage (Java vs Python)

Figure 3: Evaluation of EneRex

that such a clustering tool cannot fully encompass the subject area and struggled to find a single representative cluster for "image synthensis". Moreover, to establish the quality of a research topic, performance metrics (i.e., dataset used) need to be tracked which was done manually in the paper. By contrast, **EneRex** can extract objective task, methods, and datasets used. The first two facets can

be used to filter out a list of papers with that particular objective task (and/or method). **EneRex** can also produce lists of papers that use a specific dataset. The authors also recommended the establishment of a continuous analysis system to keep track of research development. Using an automated system like **EneRex** to find out the interesting facets would enable us toward that goal.

After incorporating the extracted entities of the papers into such a knowledge graph, the next step will be to translate the extracted raw entities back to a standardized list of generic entities so that a hierarchical taxonomy of the facets and sub-facets can be created within the knowledge graph. But there is no established taxonomy available for standardizing different scientific facets, sub-facets and their variations, i.e., objective tasks, methods and application areas. Most researchers adopt ad-hoc taxonomies suited to their specific research goal. Thus this issue remains an open research problem for future expansion.

## 7 CONCLUSION

This work presents an information extraction pipeline, **EneRex**, developed for extracting six scientific facets from full-text scholarly research articles: link to source code, dataset used, objective task, method, computing resources and language/library dependency. We demonstrated how the results from **EneRex** can be helpful in discovering research trends and emerging technologies from a large scale dataset. We evaluated **EneRex** on multiple datasets and highlighted the shortcomings in existing datasets. In addition to establishing trends, extracted entities from large scale scholarly datasets can be used to build knowledge graphs which have tremendous promise for the research community. We discussed how **EneRex** can complement such knowledge graphs with a use case. This research for full-text entity extraction serves as a stepping stone towards automated systems for extracting entities and updating such knowledge graphs without human effort.

## ACKNOWLEDGMENTS

## REFERENCES

[1] 1991. arXiv.org e-Print archive. https://arxiv.org/.
[2] 2008–2021. GROBID. https://github.com/kermitt2/grobid.
[3] 2022. The latest in Machine Learning | Papers With Code. https://paperswithcode.com/. Accessed: 2022-02-01.
[4] Ashwin Acharya, Max Langenkamp, and James Dunham. 2022. Trends in AI Research for the Visual Surveillance of Populations. https://doi.org/10.51593/20200097
[5] Rexy Arulanandam, Bastin Tony Roy Savarimuthu, and Maryam A Purvis. 2014. Extracting crime information from online newspaper articles. In *Proceedings of the second australasian web conference-volume 155*. 31–38.
[6] 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 11th International Workshop on Semantic Evaluation (SemEval-2017)*. ACL, Vancouver, Canada, 546–555.
[7] Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. SciBERT: A pretrained language model for scientific text. *arXiv preprint arXiv:1903.10676* (2019).
[8] Emily M. Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. 2021. On the Dangers of Stochastic Parrots: Can Language Models Be Too Big?. In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency (FAccT '21)*. ACM, New York, NY, USA, 610–623.
[9] Sergey Brin. 1998. Extracting patterns and relations from the world wide web. In *International workshop on the world wide web and databases*. Springer, 172–183.
[10] Jack Clark, Kyle Augustus Miller, and Rebecca Gelles. 2021. Measuring AI Development: A Prototype Methodology to Inform Policy. https://doi.org/10.51593/20210008

[11] Arman Cohan, Sergey Feldman, Iz Beltagy, Doug Downey, and Daniel S Weld. 2020. Specter: Document-level representation learning using citation-informed transformers. *arXiv preprint arXiv:2004.07180* (2020).
[12] 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).
[13] James Dunham, Jennifer Melot, and Dewey A. Murdick. 2020. Identifying the Development and Application of Artificial Intelligence in Scientific Text. *CoRR* abs/2002.07143 (2020). arXiv:2002.07143 https://arxiv.org/abs/2002.07143
[14] Sonal Gupta and Christopher D Manning. 2011. Analyzing the dynamics of research by extracting key aspects of scientific papers. In *Proceedings of 5th international joint conference on natural language processing*. 1–9.
[15] Sarthak Jain, Madeleine van Zuylen, Hannaneh Hajishirzi, and Iz Beltagy. 2020. SciREX: A Challenge Dataset for Document-Level Information Extraction. In *ACL*. 7506–7516.
[16] Shilpa Lakhanpal, Ajay Gupta, and Rajeev Agrawal. 2015. Towards Extracting Domains from Research Publications.. In *MAICS*. 117–120.
[17] Yi Luan. 2018. Information extraction from scientific literature for method recommendation. *arXiv preprint arXiv:1901.00401* (2018).
[18] Yi Luan, Luheng He, Mari Ostendorf, and Hannaneh Hajishirzi. 2018. Multi-Task Identification of Entities, Relations, and Coreference for Scientific Knowledge Graph Construction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Brussels, Belgium, 3219–3232. https://doi.org/10.18653/v1/D18-1360
[19] Sepideh Mesbah, Christoph Lofi, Manuel Valle Torre, Alessandro Bozzon, and Geert-Jan Houben. 2018. TSE-NER: An Iterative Approach for Long-Tail Entity Extraction in Scientific Publications. In *SEMWEB*.
[20] Zara Nasar, Syed Waqar Jaffry, and Muhammad Kamran Malik. 2018. Information extraction from scientific articles: a survey. *Scientometrics* 117, 3 (2018), 1931–1990.
[21] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. (2018).
[22] Ilya Rahkovsky, Autumn Toney, Kevin W. Boyack, Richard Klavans, and Dewey A. Murdick. 2021. AI Research Funding Portfolios and Extreme Growth. *Frontiers in Research Metrics and Analytics* 6 (2021). https://doi.org/10.3389/frma.2021.630124
[23] Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. 2020. Zero: Memory optimizations toward training trillion parameter models. In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, 1–16.
[24] Alan Ritter, Sam Clark, Oren Etzioni, et al. 2011. Named entity recognition in tweets: an experimental study. In *Proceedings of the 2011 conference on empirical methods in natural language processing*. 1524–1534.
[25] Arnab Sinha, Zhihong Shen, Yang Song, Hao Ma, Darrin Eide, and Kuansan Wang. 2015. An Overview of Microsoft Academic Service (MAS) and Applications. In *WWW - World Wide Web Consortium (W3C)*. https://www.microsoft.com/en-us/research/publication/an-overview-of-microsoft-academic-service-mas-and-applications-2/
[26] Yuka Tateisi, Tomoko Ohta, Sampo Pyysalo, Yusuke Miyao, and Akiko Aizawa. 2016. Typed Entity and Relation Annotation on Computer Science Papers. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*. European Language Resources Association (ELRA), Portorož, Slovenia, 3836–3843. https://aclanthology.org/L16-1607
[27] Dominika Tkaczyk, Andrew Collins, Paraic Sheridan, and Joeran Beel. 2018. Machine learning vs. rules and out-of-the-box vs. retrained: An evaluation of open-source bibliographic reference and citation parsers. In *Proceedings of the 18th ACM/IEEE on joint conference on digital libraries*. 99–108.
[28] Dominika Tkaczyk, Paweł Szostek, Mateusz Fedoryszak, Piotr Jan Dendek, and Lukasz Bolikowski. 2015. CERMINE: Automatic Extraction of Structured Metadata from Scientific Literature. *Int. J. Doc. Anal. Recognit.* 18, 4 (Dec. 2015), 317–335.
[29] Chen-Tse Tsai, Gourab Kundu, and Dan Roth. 2013. Concept-based analysis of scientific literature. In *Proceedings of the 22nd ACM international conference on information & knowledge management*. 1733–1738.
[30] Yuli Vasiliev. 2020. *Natural Language Processing with Python and SpaCy: A Practical Introduction*. No Starch Press.
[31] David Wadden, Ulme Wennberg, Yi Luan, and Hannaneh Hajishirzi. 2019. Entity, Relation, and Event Extraction with Contextualized Span Representations. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. ACL, Hong Kong, China, 5784–5789.

## A    MODEL DETAILS

For sentence classifiers, we initialized the model with SciBERT base [7]. We isolated the candidate sentences from our training data generation script and randomly sampled twice as many incorrect sentences from the papers. The train, dev and test split is (85%:10%:5%). For dataset and source code links, We trained the models for 3 epochs using AdamW optimizer with learning_rate 2e-5 and seed being 1. The batch size was 8. For computing resources and language/library, we trained the models for 10 epochs with batch size 32. For the dataset named entity recognizer, we tokenized the candidate sentences from our weakly supervised learning with Spacy [30], tagged the tokens in BILUO format and trained a transformer model, initialized with SciBERT for 3 epochs with batch size 16. The maximum sequence length is 256 for both cases. For computing resource and language/library, we trained the model with CORLL dataset with batch size 32.

To evaluate, we trained the SciREX model ourselves following the guideline in the paper. We trained the main model for 20 epochs and the coreference model for 10 epochs, using 10 and 4 as patience value respectively. To ensure an even comparison, we identified the salient entity clusters from the dataset and feed them into our own evaluation metric which used a fuzzy string matching. All of our training and prediction is using a NVIDIA Tesla P100 GPU with 16GB memory.

## B    ANNOTATED DATASET STATISTICS

We annotated a dataset, specifically for the evaluation of our computing resource and language/library facets. The distribution of different facets inside this dataset are presented below.

**Table 5: Facets and their distribution in the annotated ground truth for evaluation**

| Facet Type | Percentage |
|---|---|
| Datasets | 50.34% |
| Source Code | 6.89% |
| Computing Resources | 28.97% |
| Language/Library | 40.69% |

## C    SYNTACTIC PATTERN CREATION

For developing the syntactic patterns, we focused on four major CS areas in which discussion of these facets is more prevalent: Computer Vision, Machine Learning, Hardware Architecture, and Artificial Intelligence. Each of these areas can be identified via arXiv metadata tags. To design the patterns, we selected 80 documents for each of the facets (20 each from the four aforementioned areas) via some sample seed words related to that facet. As a next step, we isolated the sentences containing these seed words and identified their syntactic properties and patterns. The patterns and templates for each of the facets created by this process are the building blocks of our syntactic pattern based extraction algorithms. For source code links, only sentences sufficed as ground truth. For dataset, computing resources and language/library, we required both sentence level and entity level ground truth to train our models.

**Table 6: Syntactic Patterns for Source Code Extraction**

| Pattern | Example |
|---|---|
| **(subj)** (*) (root) | it we model implementation source code supplementary material |
| **(root)** | is are find release |
| (root) (*) **(obj)** | Github website open-source implementation project page supplementary material |
| **(adj/adv)** | publicly available online opensource opensource supplementary |

**Table 7: Syntactic Patterns for Dataset Extraction**

| Pattern | Example |
|---|---|
| **(subj)** (*) (root) | performance paper we dataset experiment |
| **(root)** | make utilize adopt create construct include consist perform introduce contain feed is use implement evaluate release focus conduct constitute |
| (root) (*) **(obj)** | database dataset github website repository online collection benchmark numerical study |
| **(adj/adv)** | publicly available online large-scale constructed synthetic dataset popular constructed |
| (root) (*) (which/that) **(verb)** | generate provide utilize adopt create construct include consist introduce contain feed use release |
| **(root)** (*) (number) | include consist contain constitute compose comprise |
| (root) (*) **(adj clause)** (number) | compose consist comprise |

## D    EVALUATION METRICS

For links to source code, we considered an extraction as correct if at least the first part of the path (excepting the network location) matches the ground truth. For example, we consider the following case as correct.

ground truth:"github.com/pwc/pwc-data"
extraction: "github.com/pwc"

But the following extraction is not correct:

ground truth:"github.com/pwc/pwc-data"
extraction: "github.com"