A Toolbox for the Nearly-Unsupervised Construction of Digital Library Knowledge Graphs

Hermann Kroll*, Jan Pirklbauer*, Wolf-Tilo Balke*
*Institute for Information Systems, TU Braunschweig, Braunschweig, Germany
{kroll, balke}@ifis.cs.tu-bs.de and j.pirklbauer@tu-bs.de

***Abstract*—Knowledge graphs are essential for digital libraries to store entity-centric knowledge. The applications of knowledge graphs range from summarizing entity information over answering complex queries to inferring new knowledge. Yet, building knowledge graphs means either relying on manual curation or designing supervised extraction processes to harvest knowledge from unstructured text. Obviously, both approaches are cost-intensive. Yet, the question is whether we can minimize the efforts to build a knowledge graph. And indeed, we propose a toolbox that provides methods to extract knowledge from arbitrary text. Our toolkit bypasses the need for supervision nearly completely and includes a novel algorithm to close the missing gaps. As a practical demonstration, we analyze our toolbox on established biomedical benchmarks. As far as we know, we are the first who propose, analyze and share a nearly unsupervised and complete toolbox for building knowledge graphs from text.**

***Index Terms*—Knowledge Graph, Information Extraction, Digital Library**

I. Introduction

Knowledge graphs are essential for digital libraries to structure textual collections in an entity-centric way. They open up a variety of applications for all kinds of information needs, such as finding detailed descriptions of cultural heritage objects in the Europeana [1], exploiting drug-disease treatments harvested from PubMed in the SemMedDB [2], semantically querying relationships and properties of Linked Data in Wikidata [3], and many more. But crafting such knowledge graphs for all kinds of domains is time-consuming and expensive. This is because many of today's practical knowledge graphs are built completely manually, such as Wikidata [3] or the Europeana [1], or at best semi-automatically (given that the textual information is sufficiently structured, e.g., harvesting Wikipedia infoboxes in DBpedia [4]).

Yet, why is automatically building knowledge graphs so difficult? On the one hand, the content curated by digital libraries may be too heterogeneous to create good quality knowledge graphs by rule-based approaches. For example, the creators of SemMedDB were quite experienced with medical language. They used a variety of grammatical patterns to extract medical relations from PubMed [2]. The challenges are clear: Neither do the rules adapt to paraphrased pieces of information, nor are they easily transferable to other domains or disciplines. On the other hand, artificial intelligence and machine learning techniques that would cater for this heterogeneity rely on supervision; See [5] for a good overview. For the training of reliable extraction algorithms, tens of thousand training examples are necessary, which in turn are usually again handcrafted. Moreover, this kind of training is needed for each specific entity type, relation, etc.

Although the process of harvesting knowledge from unstructured texts is challenging, novel developments in the area of Open Information Extraction (OpenIE) promise to change the game: OpenIE tools are designed to extract as much information as possible without the need for supervision [5]–[7]. While this would account for the applicability across domains and the excessive need for training data, OpenIE tools still have practical limitations. Since these methods are designed to work on all kinds of information, their extractions within topically focused digital libraries tend to be far too general to result in a concrete graph structure describing the respective domain sufficiently well. Moreover, more complex natural language processing tasks like resolving synonyms or disambiguating homonyms still need domain experts' explicit input and data modeling.

The question is whether these limitations can be bypassed in practical digital library projects? Probably, we need a minimum of supervision. This paper focuses precisely on this gap: We develop a toolbox that converts a collection of unstructured text from arbitrary domains into a structured knowledge graph using as little supervision as possible.

Subsequently, our requirements for our nearly unsupervised toolbox are obvious: It must be capable of processing millions of documents for real-world scenarios, and the resulting knowledge graph should retain good quality. We analyze the necessary steps to build a knowledge graph, including entity linking and information extraction. Entity linking detects concepts of pre-known vocabularies in texts, and information extraction extracts relations between them [5]. Our findings will show that we need practical algorithms to transform general OpenIE outputs into a domain-specific knowledge graph using as little supervision as possible. Here, we develop a novel iterative semi-supervised cleaning algorithm with expert feedback. In addition, we develop a novel extraction technique called PathIE that reuses entity information in the extraction phase. PathIE is more flexible, faster, and has a better recall than established OpenIE tools, but suffers in precision.

In this paper, we will analyze the missing gap for constructing knowledge graphs in digital libraries: *Can we bypass the need for supervision completely? And how reliable and well will tools perform for practical applications in digital libraries?* As far as we know, we are the first who develop

a practical and nearly unsupervised extraction toolbox for digital libraries, see Sect. III. Our toolbox is not domain-specific and bypasses the extensive need for supervision when possible; See our discussion in Sect. V. We have applied and analyzed our toolbox in the biomedical domain; See Sect. IV. However, our evaluation will show that the toolbox will suffer in performance compared to established supervised methods. Although the quality might be lacking, the toolbox offers a nearly unsupervised way to build knowledge graphs in digital libraries. Further, we share our toolbox on GitHub[1] to make it reusable for other researchers. The code is written in Python and is published under the MIT license.

The contributions of our work are:

1) We design an unsupervised, fast, and easy-to-use information extraction method PathIE. PathIE is capable of finding subject-predicate-object facts as well as support the extraction of important keywords.
2) We develop a novel semi-supervised iterative predicate cleaning algorithm utilizing word embeddings and expert feedback.
3) We design a nearly unsupervised toolbox covering entity linking, information extraction, cleaning, and storage. We analyze the quality of our toolbox on established biomedical benchmarks.

II. Related Work

This section gives an overview of related work for the essential components to build knowledge graphs: Entity linking and relation extraction. Besides, we report on work about the canonicalization of open information extraction outputs.

a) Entity Linking: is the task to link text spans to pre-known entities [5]. Many algorithms and frameworks exist to perform entity linking in practice, such as the ConceptMapper. Funk et al. performed a large-scale evaluation of available annotation tools in the biomedical domain [8]. Their findings show that parameters should carefully be chosen for different ontologies to achieve good quality. Dictionary-based algorithms take a vocabulary and a text as an input and perform a direct string-matching, i.e., if an entity term is mentioned in the text, a mapping between the vocabulary entry and the text is produced. An advantage of dictionary-based approaches is their performance, i.e., a single iteration over the text with dictionary-based lookups is enough to produce the annotations. Suffering performance to be more error-tolerant may be done by searching via string similarities, i.e., slight derivations of vocabulary entries are allowed. If entity terms have ambiguous meanings (homonyms), then the context of the entity terms in the text must be considered. Here, more complex approaches are needed to resolve homonymous terms correctly. For example, short abbreviations in the biomedical domain refer to several diseases, genes, and drugs. Tools such as TaggerOne and GNormPlus are designed to consider the context of the words [9], [10]. Typically, these tools are supervised [5], i.e., they are trained with training data to learn the appropriate contexts [9], [10]. There was a long discussion about the complexity of tagging models in [11]. The authors argue that it might help train a language model like BERT to maximize the annotation quality. However, simpler models like classical decision trees perform slightly worse but are trained much faster. In summary, the decision is up to a specific domain and use case. Supervised models offer the best performance, but in practice, dictionary-based approaches might already be sufficient.

b) Relation Extraction: Supervised relation extraction supports the construction of knowledge graphs from text [5], [12]. Collecting training data for supervised methods means compiling tens of thousand example extractions. These examples are then used to train a relation extraction for a single relation. Modern relation extraction even builds upon pre-trained language models like BioBERT [13]. Further, relation extraction tools may build upon distant supervision, i.e., a training procedure does not require explicit sentences and their contained facts [14]. A ground truth of valid facts is sufficient, but no text evidence for them must be provided. A learning procedure then extracts facts from texts to learn which grammatical structures lead to correct extractions. Tools such as Snorkel [15] support the automatic generation of training data by formulating hints on which sentences would be good candidates for a relation. Although the quality might be promising, training relation extraction models means giving examples for every relation, i.e., these models cannot be transferred to another domain. And moreover, having such a ground truth for distant supervision is not always the case. So indeed, although methods exist that try to boil down the need for supervision, here, as far as we know, supervision cannot be bypassed completely. Hence, we design our toolbox to bypass the need for training data in the extraction phase completely.

c) Canonicalizing OpenIE Extractions: Research has already been done on canonicalizing OpenIE extractions [16], [17]. For example, CESI uses word embeddings and side-information to canonicalize open knowledge bases [16]. An open knowledge base may be understood as the output of an open information extraction process. The authors suggest clustering subjects, predicates, and objects in a high-dimensional vector space. They use side-information like additional databases and embeddings to embed a subject, predicate, or object into a high-dimensional vector space. A small part of all subjects and objects must be linked to some existing entity vocabulary. Then, a clustering step is applied to resolve synonymous subjects like *N.Y.C.* and *New York* and predicates like *born in* and *has birthplace*. However, CESI has two major limitations: First, some entity linking is required, and side information is domain-specific, i.e., it is not transferable. Second, using clustering does not yield explainable results. As an example, CESI outputs a list of different predicates belonging to the same cluster. On the one hand, the number of obtained clusters is quite unclear. Finding a good number of clusters is a general problem when clustering. On the other hand, adding a precise predicate label to represent all synonymous predicates is difficult, especially if the predicates'

[1] https://github.com/HermannKroll/KGExtractionToolbox

context is unavailable. Overall, CESI is an exciting approach, but it requires domain-specific side information and has hard-to-interpret outputs.

III. KG Extraction Toolbox

This chapter describes the essential components of our toolbox and our novel methods that close the missing gap between open information extraction and practical knowledge graphs. Returning to our scenario, we aim to build a biomedical knowledge graph that captures knowledge about drugs, diseases, and more. Subsequently, all examples stem from the biomedical domain. However, the toolbox can be transferred to other domains because we bypass the need for supervision.

A. Knowledge Graph

First, we will define knowledge graphs for our purposes. The Semantic Web community and the W3C recommend the Resource Description Framework (RDF) to store knowledge [18]. A triple, called **fact**, consists of a subject, a predicate, and an object. A fact represents a piece of knowledge, e.g., (*simvastatin*, *treats*, *hypercholesterolemia*). Collections of these facts are usually called **knowledge graphs**. Knowledge graphs are entity-centric, i.e., only one node represents the entity *simvastatin*. In a broad sense, an **entity** is an important concept someone is looking for, e.g., drugs and diseases. We denote the set of all entities as $\mathcal{E}$. *Values* such as dates, locations, numeric values, or strings might be of interest as well, e.g., the melting point of some substance. These values are called **literals**, and we denote the set of all literals as $\mathcal{L}$. Formally,

Definition 1: A knowledge graph $KG = (V, E, \Sigma)$ is a collection of knowledge. $V \subseteq (\mathcal{E} \cup \mathcal{L})$ is a set of nodes and $E \subseteq \mathcal{E} \times \Sigma \times (\mathcal{E} \cup \mathcal{L})$ is a set of directed and labeled edges. $f = (s, p, o) \in E$ is a fact with $s \in \mathcal{E}$ being a subject, $p \in \Sigma$ being a predicate and $o \in (\mathcal{E} \cup \mathcal{L})$ being an object.

Yet, a fact is a labeled relation, denoted by a predicate, between a subject and an object. These predicates stem from a set of predicate labels Σ. The RDF standard covers many more things that are beyond the scope of this paper [18]. We focus on relations between entities and literals as the core of each knowledge graph. We discuss the necessary steps to build knowledge graphs from texts in digital libraries in the following. A schematic overview of our pipeline is depicted in Fig. 1.

B. Entity Linking

Entity linking is the task to link text spans to pre-known entities [5]. These entities usually stem from vocabularies or ontologies. Vocabularies collect important entities plus adequate synonymous terms, descriptions, and more. Ontologies may provide additional information about entities like subclass relationships, e.g., *simvastatin is a drug* and *drugs are chemical compounds*. Biomedical researchers already spend much work designing suitable vocabularies and ontologies; see BioPortal[2] for an overview. Designing ontologies is a well-known task for digital libraries, e.g., PubMed uses so-called Medical Subject Headings[3] (MeSH) to accelerate the retrieval quality by resolving synonyms or finding relevant sub-concepts. In a broad sense, entities might be seen as arbitrary resources, e.g., drugs, processes, treatment options, study types, and many more. The Dublin Core Metadata Initiative[4] already proposes a plethora of different vocabularies and gives hints on how to design them in a standardized way. In the following, we will consider the terms vocabulary and ontology synonymous in being collections of entity entries. The process of entity linking is well-known in many digital libraries, e.g., PubMed uses human curators and automatic processes to annotate publications with additional MeSH terms. Returning to our toolbox, we must identify these entities in written texts to extract knowledge about them.

We implement a dictionary-based entity linker to support unsupervised entity linking in our toolbox. The entity linker is designed to handle large amounts of text, i.e., it is designed to have a fast performance. Our entity linker requires an entity vocabulary and text documents as its input. Then, our linker produces entity annotations between the text and the vocabulary as its output. Usually, supporting synonyms and resolving conflicts is straightforward, i.e., entities plus their adequate synonyms are identified by unique identifiers. However, dictionary-based linking typically struggles with minor typing errors, unknown synonyms, homonyms, or custom abbreviations by design. Therefore, our linker supports custom abbreviations in a document. Suppose an author introduces the abbreviation *ASR* for *Aspirin* via *Aspirin (ASR)*. In that case, our linker will resolve the abbreviation in the rest of the corresponding document correctly. Short entity names like well-known abbreviations of some entities may lead to wrongly tagged homonyms. Our linker only links short abbreviations if the corresponding entity is at least detected a single time with its complete mention in the document's scope. In this way, we minimize wrongly linked homonyms. The user can adjust the length of a required complete mention. We support a configuration file to adjust these settings for a user's purpose.

Named Entity Recognition (NER) is a broader method to detect entities and important concepts in texts. NER may recognize entity mentions in the text but does not link these mentions against pre-known entity vocabularies [5], [19]. For example, the Stanford Stanza NER detects person names, organizations, locations, dates, and more in texts [19]. Stanza supports the annotation of 18 different named entity types. Especially, the detection of dates and locations might be beneficial across domains. However, NER comes with the limitation of not providing unique entity ids. A text span is identified as an entity type, but a precise entity id is not provided. NER may lead to synonymous entities in a practical knowledge graph. To demonstrate the usefulness of NER in practice, we integrate an interface for Stanza into our toolbox supporting the annotation of more general named entity types.

[2]https://bioportal.bioontology.org last access: 06.2021

[3]https://meshb.nlm.nih.gov last access: 06.2021

[4]https://www.dublincore.org/specifications/dublin-core/dces/ l.a.: 06.2021

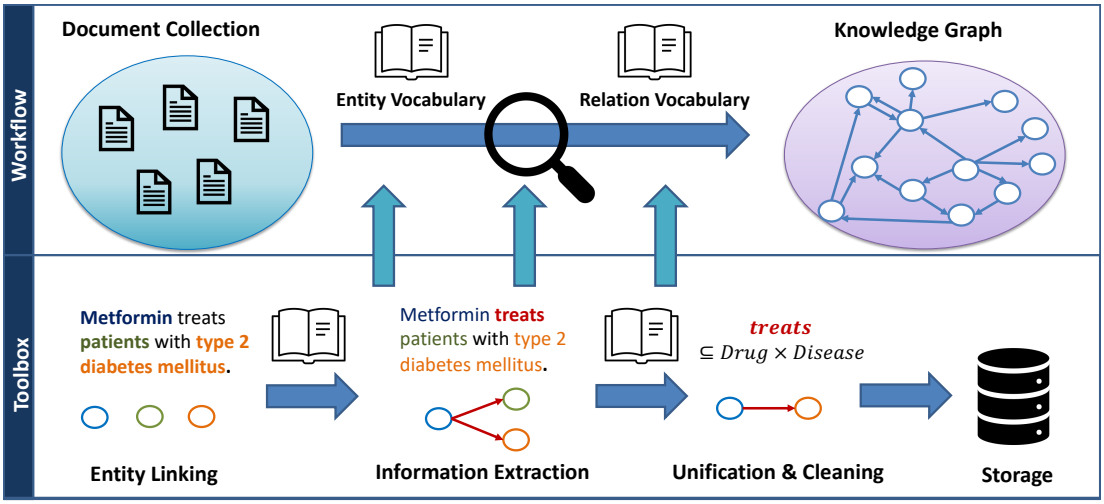

Fig. 1. The Toolbox's Systematic Overview: Entity linking detects important concepts and information extraction extracts important relations between them. Then, the output will be cleaned and loaded into a structured repository.

Details about the quality of Stanza can be found on its project website[5] or in [19]. However, our toolbox might be easily extended by integrating domain-specific and supervised entity linkers. For example, the National Library of Medicine (NLM) provides two powerful and freely usable tools: TaggerOne [10] detects chemicals and diseases, and GNormPlus [9] detects genes and species in texts. We have implemented interfaces to both tools into our toolbox to demonstrate how domain-specific entity linkers may be integrated. Lastly, the detection of arbitrary literals like melting points in texts might be solved via regular expressions. However, the literal detection strongly depends on a given domain's requirements. Thus, we do not integrate literal detection in our toolbox.

C. Open Information Extraction

Information extraction is the task to transform unstructured information into structured information [5], [6]. In this paper, we understand information extraction as the extraction of facts from texts. As a reminder, a fact is a simple triple containing a subject, a predicate and an object, e.g., the fact (*simvastatin*, *treats*, *hypercholestorelemia*) may be extracted from *simvastatin is used to treat hypercholestorelemia in patients*. Information extraction is usually limited to pre-defined relations and entities, and that is why we build upon *open* information extraction methods. Open information extraction is not limited to pre-known relations and hence, can be used across domains. They take arbitrary text as an input and produce facts as an output. Many OpenIE tools are available as free-to-use software and work out-of-the-box. In addition, these natural language processing toolkits work with a plethora of different languages, e.g., the Stanford OpenIE tool supports seven different languages [7], and Stanza supports even 66 different languages [19]. Recently, Kolluru et al. developed a novel OpenIE6 extraction method and analyzed the quality and performance compared to established OpenIE tools for the natural language processing community. Their findings show that OpenIE tools come at best with a F1-measure between 40.0% and 65.6% (tested on several benchmarks). The best performing system is OpenIE6 (2020), which can process up to 31.7 sentences per second on a Nvidia Tesla V100. OpenIE6 builds upon the latest neural extraction methods and is pre-trained on a large variety of text. It does not require domain-specific training and could thus be understood as an unsupervised extraction method. At the same time, the Stanford CoreNLP tool is an older rule-based and model-based approach that is well-supported and has a fast runtime performance [7]. Our toolbox implements interfaces for both OpenIE methods, namely Stanford CoreNLP and OpenIE6.

Although the quality and runtime performance sound sufficient, we can boost their performance further by using entity linking information. Our focus is on constructing knowledge graphs, and hence, we are only interested in facts between entities and literals. First, this restriction boosts the runtime performance, i.e., we only need to process sentences containing at least two different entities/literals mentions. This filtering may significantly reduce the number of sentences to process, depending on the number of annotated entities and literals. The toolbox applies this filtering step before extracting information automatically if desired. OpenIE output usually tends to be more general because OpenIE has no starting point, e.g., subject or object could be anything within a sentence's scope. Therefore, the toolbox uses entity linking information to filter OpenIE fact extractions by subjects and objects. Consider the sample sentence: *Metformin treats patients with diabetes*. OpenIE applied to that sentence result in extractions such as (*metformin*, *treats*, *patients with diabetes*). Subjects and objects should be entities (objects might be literals as well), which is not the default case for OpenIE; see our example above: *Patients with diabetes* is not an entity. The object includes *diabetes* only partially. We assume a partially included entity to be sufficient and rewrite the fact to: (*metformin*, *treats*,

[5]https://stanfordnlp.github.io/stanza/ last access: 06.2021

diabetes). Our toolbox supports a parameter to select the entity filtering mode: None (keep all OpenIE extractions), partial (subject and objects must partially include an entity mention), complete (subject and object must be a fully annotated entity). Then, it cleans the results of the supported OpenIE tools automatically. The toolbox automatically converts passive voice to active voice by the following rule: If the lemmatized predicate includes *be* and the predicate contains a verb in past particle. We utilize the Spacy NLP toolkit to quickly lemmatize the predicate and compute the Part-of-Speech tags for the predicate[6].

D. PathIE

In contrast to conventional information extraction, where arbitrary information is extracted, we only consider interactions between entities and literals. Usually, supervised relation extraction methods do precisely this: They already know the subject and object candidates. OpenIE does not have this information available in the extraction phase. And obviously, we could hardly integrate it into existing tools. However, we already have entity linking information available but want to bypass supervised relation extraction. The central question for a fact extraction is how entities are related within the sentence. We design a high-performance extraction method called **PathIE** utilizing the available entity information. In typical natural language processing, each sentence is represented as a sequence of tokens, i.e., single words. Furthermore, each word is assigned a part-of-speech tag (POS tag), i.e., a word category as nouns, verbs, etc. Tokens are syntactically arranged in a so-called dependency parse tree, i.e., each token has specific relations to other tokens within the sentence (subject, etc.). PathIE utilizes the syntactical structure of a sentence to answer the question of how entities are related. Tools, such as the Stanford CoreNLP suite, offer high performance when tokenizing, POS-tagging, and dependency parsing a sentence [7]. Consider the following example sentence: *Metformin is widely considered to be the optimal initial therapy for patients with type 2 diabetes mellitus*. Our entity linking for this sentence results in *metformin* (drug) and *type 2 diabetes mellitus* (disease).

PathIE utilizes these entity linking information and searches upon the sentence's grammatical structure to derive the relation between both entities. We transform the syntactical sentence structure into a graph, i.e., nodes represent tokens, and edges represent grammatical dependencies between the tokens. We take advantage of the graph representation to perform a path search between the tokens of both entities. Here, we compute the shortest paths because we are interested in the shortest and most substantial syntactical relation. The shortest path for our example sentence is the following sequence of tokens: (*metformin*, *considered*, *therapy*, *patients*, *type 2 diabetes mellitus*). The corresponding relation between both entities can be identified by 1. searching for all verbs on the path (via POS-tags), and 2. by searching for special keywords like *treatment*, *therapy* or *inhibition* on the path. These special keywords can optionally be pre-defined in a vocabulary before applying PathIE. Hence, relations between entities are identified by 1) detecting all verbs on the path via the token's POS tags (VBN, VB, etc.), and 2) optionally detecting hand-crafted vocabulary terms on the path. These terms can be seen as special words like *treatment*, *metabolite* and more. Subject, object, and each identified predicate are composed to a fact extraction for the sentence. The path search is not directed, and thus, we extract both directions for the interaction-keyword *therapy*: (*metformin*, *therapy*, *diabetes*) and (*diabetes*, *therapy*, *metformin*). These facts may be cleaned in the cleaning step discussed subsequently. In some cases, such a path might contain words like *not* or *may* which could lead to a wrong extraction. We support two parameters for PathIE to ignore extractions which contain a *not* or *may*. We assume our path-based extraction technique allows a more flexible extraction yielding a higher recall, but on the other side, decreasing the precision. PathIE relies on a NLP tool to compute dependency parses. We support the computation of dependency parses via Stanford CoreNLP (rule-based and faster) and Stanford Stanza (neural and more precise).

E. Unifying Synonymous Predicates

OpenIE and PathIE yield a variety of different predicates in their extractions by design. As an example, the predicates *treats* and *aids* have the same meaning when talking about the cure of some disease by a drug. We have to unify these synonymous predicates to build a knowledge graph with a manageable set of relations. Hence, our goal is to design a *relation vocabulary*, i.e., a set of relations with a list of synonymous predicates. The relation *treats* might have the synonyms *aids*, *improves* and *prevents*. Using a relation vocabulary allows us to unify the extracted synonymous predicates. Obviously, going through thousands of synonyms manually and building a relation vocabulary is too time-consuming. Hence, the process must be automated, at best, without supervision.

Word embeddings embed words into a high-dimensional vector space by considering their context [20]. Word vectors whose words share a similar context should be located close to each other. Moreover, word embeddings can be trained on arbitrary text without supervision and are already known for their ability to find synonyms for a given word. But, how can we create a vocabulary of relations by unifying the extracted predicates? We cannot entirely bypass the need for supervision here because we need information on how relevant some predicates are in a domain. We design an iterative semi-supervised algorithm allowing domain experts to make these decisions. The algorithm is depicted in Fig. 2 and works as follows:

1) All fact extractions are grouped by their predicate. Then, the group's size is counted.
2) The distances between each predicate and all entries of the relation vocabulary are computed. The nearest neighbor is kept for each predicate. Hence, we obtain

[6]https://spacy.io last access: 06.2021

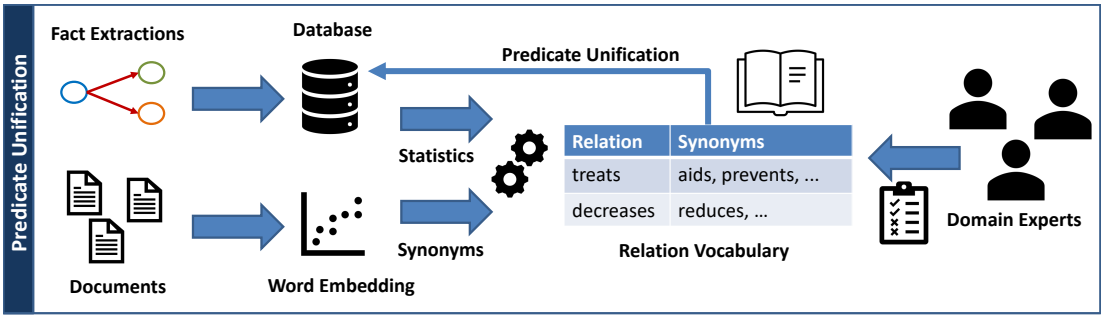

Fig. 2. Systematic overview of our novel semi-supervised iterative predicate unification algorithm. The algorithm reuses extraction information, a word embedding and expert feedback to build a relation vocabulary iteratively. The relation vocabulary is used to clean the open information extractions later.

a mapping between a predicate and a term of the relation vocabulary.

3) All mappings with a distance below a threshold t are removed.
4) A list of mappings between the predicates and relations of the vocabulary is computed. This list is sorted by the number of extractions per predicate. The sorted list is shown to the domain experts.
5) The experts can go through the top entries of the list (maybe top 10). Suppose a predicate is mapped to the wrong relation. In that case, they can improve their relation vocabulary by introducing a new relation or adding the predicate as a synonym to an already included relation.
6) If all the important predicates are mapped correctly, the experts can abort here. If not, the algorithm will repeat at step 2 with the new relation vocabulary. The algorithm outputs the predicate mappings against the relation vocabulary.

Wrapping up, the algorithm takes a word embedding, a threshold t, a relation vocabulary, and a set of fact extractions as input. It results in mappings between predicates and relations of the vocabulary. The algorithm shows the most important predicate (sorted by the number of extractions) to the experts by design. In this way, they can iteratively build the relation vocabulary. They may start with an empty relation vocabulary or may have some first ideas. The parameter t allows experts to choose a gap between precision and recall. If a high threshold is chosen, the algorithm only maps those predicates that are more likely to be synonymous (closer location in the vector space). Hence the precision of correct mappings will be higher. If a lower threshold is chosen, the algorithm may yield wrong mappings but include more correct ones (higher recall). The threshold should be adjusted for domain-specific word embeddings. Finally, the algorithm produces a reliable relation vocabulary plus predicate mappings against it. In this way, many synonymous predicates can be unified with an acceptable amount of labor. Indeed, the algorithm cannot bypass supervision completely but boils the supervision down to building a relation vocabulary with a manageable set of entries iteratively. Our biomedical relation vocabulary needs three iterations and around 63 different vocabulary entries.

F. Knowledge Graph Constraints

Entity linking, information extraction, and predicate cleaning return a set of fact extractions. In practice, knowledge graphs should contain relations with precise semantics, e.g., *treats* is a relation between drugs and diseases. Good examples are **type constraints**, e.g., *treats* should be a relation where all subjects are *drugs* and all objects are *diseases* (*treats* $\subseteq$ *Drugs* $\times$ *Diseases*). In a large-scale extraction scenario, extraction errors are likely to occur, e.g., an erroneously extracted treat relation between two diseases. Obviously, cleaning such a relation by type constraints will increase the overall quality, e.g., *treats* is a relation that has drugs as its subjects and diseases as its objects. Hence, a relation type constraint defines the allowed entity types for subjects and objects. Our toolbox supports type constraints to clean the fact extractions to increase the overall quality. Domain experts can formulate these integrity constraints, and our tool will automatically check them in the cleaning phase. In addition to type constraints, broader integrity constraints might be helpful. For example, if we know that X is the treatment for some disease Y, then Y cannot be an adverse effect of X. Such integrity constraints require domain-specific logic and hence, must manually be formulated by domain experts.

G. Storage and Provenance

The toolbox generates various outputs like entity linking information and information extraction. To minimize the need to handle several files, the toolbox utilizes an object-relational mapper as an interface to an underlying relational database. We use a Postgres system by default, but the relational mapper also supports other systems like SQLite or MySQL. The toolbox stores all information within a single place to reuse specific information if necessary, e.g., the toolbox automatically queries entity linking information in the extraction phase if required. However, the toolbox supports the export of outputs in different formats; See the GitHub page for more details. Entity linking information can be exported as PubTator documents or in a JSON format. Consider the following scenario: *Metformin is used to treat patients with diabetes* is a sentence in some document. The entity linking steps may yield that *Metformin* represents the ChEMBL[7] identifier *CHEMBL1431*

[7]https://www.ebi.ac.uk/chembl/ last access: 06.2021

and *patients with diabetes* are associated with the disease *Diabetes Mellitus* also known as the MeSH identifier *D003920*. The information extraction steps yields (*CHEMBL1431*, *is used to treat*, *D003920*). The cleaning step will map the predicate *is used to treat* to *treats*, so that, (*CHEMBL1431*, *treats*, *D003920*) is obtained. Next, the toolbox must export the extractions in some format. The easiest way would be to export facts like (*CHEMBL1431*, *treats*, *D003920*). However, if the knowledge graph is used in downstream applications, it might be helpful to provide additional provenance information. Provenance ranges from just storing a reference to the document, in which a fact was extracted, to storing all information starting by the entity linking step. That means we must store a tuple with the signature (*document*, *subject_str*, *subject_id*, *subject_type*, *predicate_str*, *predicate*, *object_str*, *object_id*, *object_type*, *sentence*). For our example, a tuple would look like (*doc_123*, *Metformin*, *CHEMBL1431*, *Drug*, *is used to treat*, *treats*, *patients with diabetes*, *D003920*, *Disease*, *Metformin is used to treat patients with diabetes*). The toolbox supports the export of facts plus provenance information to support both scenarios. The fact extractions and useful provenance information can be exported as a TSV file or RDF-serialization format. More details can be found on our GitHub page.

IV. Evaluation

In the following, we evaluate our toolbox by applying it to established biomedical benchmarks. All experiments are performed on our server, which has two Intel Xeon E5-2687W (3.1 GHz, eight cores, 16 threads), 377 GB of DDR3 main memory, NVME SDDs as its primary storage, and a Nvidia 1080 GTX TI as a GPU. We enable the GPU support for the Stanford Stanza toolkit and OpenIE6.

A. Entity Linking

We evaluate and report the quality of entity linking with our toolbox subsequently. Therefore, we have selected four established biomedical benchmarks: 1. disease normalization in NCBI Disease [21], 2. disease normalization in Biocreative V CD-R [22], 3. chemical normalization in Biocreative V CD-R [22] and 4. human gene normalization in Biocreative II Gene Normalization [23]. All of these benchmarks require entity detection in text. Then, the entity mentions must be linked to normalized (disambiguated) concepts (entity identifiers). All benchmarks provide entity vocabularies that we use as inputs for our linker. In comparison to our entity linker, we report the results of the latest biomedical entity linkers TaggerOne [10] and GNormPlus [9]. TaggerOne and GNormPlus are both supervised. We report all results in Table I.

TaggerOne recognizes diseases on the NCBI Disease Benchmark with a precision of 82.2% and a recall of 79.2%. Our entity linker achieves a precision of 74.5% and recall of 55.1%. On the BioCreative V benchmark, TaggerOne detects diseases with a precision of 84.6% and a recall of 82.7%. In comparison, our entity linker achieves a precision of 82.8% and a recall of 62.0%. Chemicals are found with a precision of 88.8% and a recall of 90.3% by evaluating TaggerOne on the BioCreative V benchmark. Our entity linker achieves a precision of 76.6% and a recall of 78.7% when linking chemicals. GNormPlus achieves a precision of 87.1% and a recall of 86.4% on BioCreative II. In comparison, our linker achieves a precision of 60.1% and a recall of 52.4%.

Next, we evaluate the linking quality when designing our own entity vocabularies. In joint work with two pharmaceutical domain experts, we design entity vocabularies for drugs, plant families, and dosage forms. We apply our entity linker against a random sample of PubMed abstracts and randomly pick 50 produced entity annotations for each entity type for evaluation purposes. We gave these entity annotations and the corresponding sentence to both domain experts. They carefully read the sentence (context) and decide together if the annotated entity is mentioned. Hence, we could only estimate the precision for these entity types. Drugs are tagged with 90% precision, plant families with 82% precision, and dosage forms with 82% precision. Concerning NER, Stanza has already been evaluated on two different benchmarks, namely CoNLL03 and OntoNotes. Stanza achieves a F1-score of 92.1% on CoNLL03 and 88.8% on OntoNotes [19].

Next, we report the linkers' runtime to estimate if they are applicable in a large-scale scenario. First, we randomly sample 10k PubMed titles and abstracts containing at least a single drug (to ensure that they contain relevant entities). Then, we run each entity linker three times on this sample to measure its runtime. TaggerOne takes around $(149 \pm 1)\,\mathrm{min}$ and GNormPlus takes around $(118 \pm 1)\,\mathrm{min}$ to complete. Our dictionary-based linker takes around $(77 \pm 1)\,\mathrm{s}$ to complete. Stanza takes around $(41 \pm 1)\,\mathrm{min}$ utilizing our GPU. Deactivating GPU supports leads to a runtime of about 9 hours.

a) Discussion: The evaluation of linking entities reveals how well an unsupervised method might work. Our entity linker lacks around 7.7% points (NCBI disease) and 1.8% points (BioCreative V) precision behind TaggerOne when detecting diseases. Although the precision of TaggerOne is not far ahead, the recall of our linker clearly lacks behind: 79.2% and 82.7% (TaggerOne) vs. 55.1% and 62.0% (our linker). However, TaggerOne takes around 150 minutes, whereas our linker needs around 77 seconds. A similar observation could be made for linking chemicals on BioCreative V. Indeed, linking human genes is challenging because gene descriptions are often short and ambiguous to other terms. Here, our entity linker clearly falls behind GNormPlus. Especially if terms are unambiguous, our entity linker achieves a high precision, e.g., 90% when linking drugs. Hence, our entity linker is a worthy competitor: Our linker is fast, achieves good precision but lacks behind in recall. Nevertheless, our entity linker does not require supervision which is a significant advantage. In summary, the development of an entity linker for a specific domain depends on the complexity and disambiguation of entity terms. Indeed, dictionary-based methods already achieve a good performance and bypass the need for supervision here. However, supervised methods should be preferred in scenarios where context is essential, e.g., when linking genes.

TABLE I
ENTITY LINKING QUALITY ON BIOMEDICAL BENCHMARKS: STATE-OF-THE-ART (SOTA) TAGGERS ARE COMPARED TO OUR UNSUPERVISED ENTITY LINKER. THE SOTA-TAGGING QUALITY RESULTS ARE FROM TAGGERONE [10] AND GNORMPLUS [9].

Benchmark	Entity Type	Quality of SOTA Entity Linker				Quality of our Entity Linker		
		Name	Precision	Recall	F-measure	Precision	Recall	F-measure
NCBI Disease [21]	Disease	TaggerOne	82.2 %	79.2 %	80.7 %	74.5 %	55.1 %	63.3 %
BioCreative V CD-R [22]	Disease	TaggerOne	84.6 %	82.7 %	83.7 %	82.8 %	62.0 %	70.9 %
BioCreative V CD-R [22]	Chemical	TaggerOne	88.8 %	90.3 %	89.5 %	76.6 %	78.7 %	77.6 %
BioCreative II GN [23]	Human Gene	GNormPlus	87.1 %	86.4 %	86.7 %	60.1 %	52.4 %	56.0 %

TABLE II
QUALITY OF OUR SEMI-SUPERVISED ITERATIVE PREDICATE CLEANING ALGORITHM. WE APPLY THREE ITERATIONS ON A PUBMED SAMPLE.

Relation	W. Prec.	Top Predicate Mappings
decreases	80.4%	reduce(1.6M), decrease(1M), mediate(430K), attenuate(339K), lower(275K), ...
induces	88.8%	induce(3.5M), increase(1.9M), cause(1.3M), result(800K), lead(698K), ...
treats	77.8%	treatment(1.1M), treats(713K), use(654K), therapy(456K), improve(192K), ...
metabol.	99.8%	metabolism(31K), catalyze(19K), metabolite(10K), metabolize(8.7K), oxidize(3K), ...
inhibits	98.6%	inhibitor(182K), inhibit(149K), inhibition(89K), suppress(44K), downregulate(9.8K), ...
interacts	69.6%	bind(497K), regulate(345K), act(148K), modulate(131K), interact(118K), ...

TABLE III
CDR2015 BENCHMARK EVALUATION [22]. THE TABLE REPORTS THE EXTRACTION QUALITY FOR OPENIE TOOLS, PATHIE AND BASELINES.

Method	Quality		
	Prec.	Rec.	F1
CoreNLP OpenIE	64.9 %	5.8 %	10.6 %
OpenIE6	53.1 %	5.5 %	10.0 %
PathIE	50.8 %	31.7 %	39.1 %
PathIE Stanza	51.1 %	30.9 %	38.5 %
Workshop Best Precision [22]	90.5 %	80.8 %	85.4 %
Workshop Best Recall [22]	86.1 %	86.2 %	86.1 %

B. Predicate Unification

We perform an expert evaluation to estimate our novel semi-supervised predicate cleaning algorithm's quality in the following. Therefore, we apply PathIE on a PubMed sample of about 5.6 million PubMed documents. The sample contains documents in which at least a single drug was linked (because we are interested in pharmaceutical relations). We use the biomedical word embedding trained on PubMed from [24]. Together with two pharmaceutical domain experts, we have designed a relation vocabulary with ten relations and around 53 entries. We build the vocabulary incrementally by performing three iterations. We tested a few thresholds for this paper and found a threshold of 0.4 to deliver good results.

Next, we evaluate six relations by selecting the top-30 predicate mappings for each relation (ranked by the vector distance). We give these mappings to two domain experts for evaluation, i.e., they decide whether a mapping is correct. The results are listed in Table II. We compute the weighted precision to weight the mapped predicates based on their frequency, i.e., predicates that occur more frequently have a greater influence on the weighted precision. We report the top five predicates that are mapped to the corresponding relation with their extraction frequency. For example, the top 30 predicates mapped to the relation *decreases* have a weighted precision of 80.4%. The weighted precision of the results is between 69.6% (interacts) up to 99.8% (metabolizes). The quality depends on how precise a relation can be formulated with corresponding synonyms, e.g., metabolizes has precise and unambiguous terms. Hence, most of the mapped predicates are correct. In contrast, the predicate *uses* is wrongly mapped to treats. Further improvements to the vocabulary can quickly be made by applying the predicate unification algorithm again., e.g., *uses* could be mapped to another relation.

C. Information Extraction

In the following, we evaluate the information extraction quality and measure the runtime to estimate whether OpenIE tools are applicable in large-scale scenarios. We evaluate both OpenIE tools in our toolbox, namely Stanford OpenIE and OpenIE6. In comparison, we analyze our PathIE extraction method based on Stanford CoreNLP and PathIE Stanza based on Stanford Stanza. For the evaluation, we apply our toolbox to already established benchmarks: 1. BioCreative V CD-R (relations between chemicals and diseases), and 2. BioCreative VI ChemProt (relations between chemicals and proteins).

a) BioCreative V CD-R: The benchmark [22] requires the extraction of *induces* relations between chemicals and diseases. The benchmark provides PubMed abstracts that are annotated with chemicals and diseases. Here, we apply our unsupervised extraction methods plus cleaning to extract *induce* relations from texts. We reuse the previously defined relation vocabulary. It comprises around ten synonyms for the *induces* relation. We did not adjust the vocabulary for this benchmark. Hence, we do not require training data here at all. The results are reported in Table III. For comparison, we include the workshop's best-performing systems concerning precision and recall.

CoreNLP OpenIE yields a precision of 59.3% and a low recall of 5.1%. OpenIE6 yields a precision of 53.1% and a recall of 5.5%. PathIE yields a precision of 50.8% and a recall of 31.7%. PathIE Stanza produces comparable results, i.e., 51.1% precision and 30.9% recall. The workshop's best performing and supervised systems achieve a precision of

TABLE IV
BIOCREATIVE VI CHEMPROT EVALUATION [25]. THE TABLE REPORTS THE EXTRACTION QUALITY FOR OPENIE, PATHIE AND BASELINES.

Method	Quality		
	Prec.	Rec.	F1
CoreNLP OpenIE	59.3 %	5.1 %	9.3 %
OpenIE6	55.9 %	6.2 %	11.1 %
PathIE	30.3 %	55.3 %	39.1 %
PathIE Stanza	29.4 %	56.6 %	38.7 %
Sentence Co-Mention [25]	4.4 %	98.0 %	0.08 %
Workshop Best Precision [25]	74.4 %	55.3 %	63.4 %
Workshop Best Recall [25]	56.1 %	67.8 %	61.4 %
BioBERT [13]	77.0 %	75.9 %	76.5 %

90.5% and 86.1%, with a corresponding recall of 80.8% and 86.2%.

b) BioCreative VI ChemProt: The benchmark [25] requires the extraction of relations between chemicals and proteins from the text. Therefore, PubMed abstracts with chemical and protein annotations are given. The task is to extract five relations, namely, *inhibits*, *upregulates*, *agonist*, *antagonist* and *substrate*. Together in cooperation with both domain experts, we carefully read the relation descriptions and build a relation vocabulary for this benchmark. The relation vocabulary comprises five relations and a few synonyms for each relation. To assist the process of finding suitable synonyms, we briefly had a look at the benchmarks training data. The creation of the vocabulary takes around one hour. Then, we evaluate our extraction methods on the benchmark's test data. The results are listed in Table IV. For comparison, we include the workshop's best performing concerning precision and recall. In addition, we include a simple sentence co-mention baseline [25] and the BioBERT relation extraction findings [13].

CoreNLP OpenIE yield 59.3 % precision and 5.1 % recall. OpenIE6 comes with a precision of 55.9 % and a recall of 6.2 %. PathIE achieves 30.3 % precision and 55.3 % recall. PathIE Stanza has a slightly lower precision 29.4 %, but higher recall of 56.6 %. Just for a comparison, the sentence co-mention baseline yields only a precision of 4.4 % and a recall of 98.0 %. Hence, a few relations must be mentioned across sentences. The best precision-oriented baselines achieves 74.4 % precision and 55.3 % recall. The best recall-oriented baseline system achieves 56.1 % precision and 67.8 % recall. BioBERT, a language model trained on the whole PubMed collection, was fine-tuned for the relation extraction task [13]. Then, BioBERT yields 77.0 % precision and 75.9 % recall. Both workshop baselines and the fine-tuning of BioBERT rely on supervision.

c) Performance Analysis: Next, we analyze the runtime of our extraction methods on a random sample of two million PubMed abstracts that include at least a single drug (biomedical focus). In summary, this sample has 178.5k entity annotations. We extract 52.6k sentences that include two different entity mentions. PathIE takes about two minutes, and PathIE Stanza takes 42 minutes on our GPU. CoreNLP takes 8.5 minutes, and OpenIE6 takes about one hour on our GPU.

d) Discussion: The runtime evaluation demonstrates that all four extraction methods are applicable in a large-scale scenario. However, the comparison to supervised methods shows disadvantages concerning precision and recall. Although supervised methods outperform our unsupervised methods, especially PathIE is a strong competitor. PathIE does not require training data at all and still may come with a precision of 50%. PathIE is designed to extract all relations between entities in sentences if connected via a predicate or a special keyword in the grammatical structure. Having a closer look at the BioCreative VI ChemProt benchmark, PathIE yields about 40% precision for the *inhibits* relation, but only 18% precision for *upregulates*. Thus, PathIE can extract some relations with good quality, but not in all cases. As already expected, OpenIE tools lose recall in comparison with PathIE. Here, OpenIE fails to extract facts from long, complex, or nested sentences. For example, OpenIE can find an inhibition in a precise clause like *Metformin inhibits mtor*. However, OpenIE could not extract the relation *inhibits* in a phrase like *Metformin is a known inhibitor for mtor*. The problem here is that the verb *is* does not give enough information to extract a meaningful *inhibits* relation. Further advancement in OpenIE would be necessary to extract such relations with a higher recall. As a last remark, biomedical relation extraction benchmarks tend to include complex, long, and nested sentences. The extraction quality of our toolbox might hence be better in another domain if sentences are more straightforward.

V. CONCLUSION

In this paper, we have developed a nearly-unsupervised toolbox to construct knowledge graphs from texts in digital libraries. An overview of our toolbox and its components is given in Table V. We have implemented a dictionary-based entity linker supporting custom abbreviations and short abbreviation resolution. In practice, our toolbox may be extended by domain-specific entity linkers like we already demonstrated with TaggerOne and GNormPlus. Our toolbox provides interfaces for the latest OpenIE tools, namely Stanford CoreNLP and OpenIE6. In addition, we design a recall-oriented and flexible extraction method PathIE. Reliable fact extractions are produced by combining these unsupervised extractions methods with entity-based filtering and a novel iterative semi-supervised predicate unification algorithm. Type constraints ensure precise semantics for relations, and integrity constraints might minimize errors in the extraction phase.

The evaluation demonstrates that we already achieve a good quality on established benchmarks. Supervised methods outperform our linker by a small margin for entity linking, but they rely on training data and are way slower. Next, supervised relation extraction outperforms our unsupervised extraction methods clearly. Moreover, the best quality can only be achieved by utilizing language models like BioBERT for relation extraction. However, training a language model for a given domain can be a very cost-intensive task [13]. The training of BioBERT took even 23 days on eight Nvidia V100 GPUs [13]. Collecting enough training data for a re-

TABLE V
AN OVERVIEW OF OUR TOOLBOX'S COMPONENTS. WE REPORT WHETHER THE COMPONENT RELIES ON SUPERVISION AND IS DOMAIN-SPECIFIC.

Component	Supervision	Domain-Specific	Supported Tools
Entity Linking	no	no	A dictionary-based entity linker for arbitrary vocabularies. Named Entity Recognition (NER) via Stanford Stanza (Location, Time, and more) [19]. We integrate TaggerOne (Diseases, Chemicals) [10] and GNormPlus (Genes, Species) [9] as examples.
Information Extraction	no	no	PathIE, PathIE Stanza, CoreNLP OpenIE [7] and OpenIE6 [6]
Predicate Cleaning	yes	yes	Entity-based filtering and an iterative semi-supervised predicate unification
Constraint Cleaning	no	yes	Cleaning via type constraints
Storage	no	no	Object-relational-mapper for relational databases and JSON/RDF-serialization export

liable entity linking or relation extraction comes even with a price. In practice, this could hinder the construction of a knowledge graph. Precisely here, we propose our toolbox. The toolbox requires entity vocabularies and expert interaction when cleaning the extracted predicates. Many domains already have designed entity vocabularies that are ready to use. And if not, tools like Stanza or utilizing entity information of existing knowledge graphs like Wikidata may close the gap here. In practice, predicate cleaning boils down to selecting a few hand-crafted relations plus synonyms in an iterative fashion. Experts control which predicates are mapped to the corresponding relation, and similar predicates are found via unsupervised word embeddings. We believe that our toolbox offers the possibility of harvesting knowledge from text across domains. Although the quality might not be the best, there is often no alternative in practice. Collecting training data and training language models is often too cost-intensive to concern.

ACKNOWLEDGMENT

Supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation): PubPharm – the Specialized Information Service for Pharmacy (Gepris 267140244).

REFERENCES

[1] A. Isaac and B. Haslhofer, "Europeana linked open data–data. europeana. eu," *Semantic Web*, vol. 4, no. 3, pp. 291–297, 2013.

[2] H. Kilicoglu, D. Shin, M. Fiszman, G. Rosemblat, and T. C. Rindflesch, "Semmeddb: a pubmed-scale repository of biomedical semantic predications," *Bioinformatics*, vol. 28, no. 23, pp. 3158–3160, 2012.

[3] F. Erxleben, M. Günther, M. Krötzsch, J. Mendez, and D. Vrandečić, "Introducing wikidata to the linked data web," in *The Semantic Web – ISWC 2014*. Cham: Springer Int. Publishing, 2014, pp. 50–65.

[4] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives, "Dbpedia: A nucleus for a web of open data," in *The Semantic Web*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 722–735.

[5] G. Weikum, L. Dong, S. Razniewski, and F. M. Suchanek, "Machine knowledge: Creation and curation of comprehensive knowledge bases," *CoRR*, vol. abs/2009.11564, 2020.

[6] K. Kolluru, V. Adlakha, S. Aggarwal, Mausam, and S. Chakrabarti, "OpenIE6: Iterative Grid Labeling and Coordination Analysis for Open Information Extraction," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online: Association for Computational Linguistics, 2020, pp. 3748–3761.

[7] C. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. Bethard, and D. McClosky, "The Stanford CoreNLP natural language processing toolkit," in *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. Baltimore, Maryland: Association for Computational Linguistics, 2014, pp. 55–60.

[8] C. Funk, W. Baumgartner, B. Garcia, C. Roeder, M. Bada, K. B. Cohen, L. E. Hunter, and K. Verspoor, "Large-scale biomedical concept recognition: an evaluation of current automatic annotators and their parameters," *BMC Bioinformatics*, vol. 15, no. 1, p. 59, 2014.

[9] C.-H. Wei, H.-Y. Kao, and Z. lu, "Gnormplus: An integrative approach for tagging genes, gene families, and protein domains," *BioMed research international*, vol. 2015, p. 918710, 2015.

[10] R. Leaman and Z. Lu, "TaggerOne: joint named entity recognition and normalization with semi-Markov Models," *Bioinformatics*, vol. 32, no. 18, pp. 2839–2846, 2016.

[11] J. Li, Y. Li, X. Wang, and W.-C. Tan, "Deep or simple models for semantic tagging? it depends on your data," *Proceedings of the VLDB Endowment*, vol. 13, no. 12, pp. 2549–2562, 2020.

[12] B. D. Trisedya, G. Weikum, J. Qi, and R. Zhang, "Neural relation extraction for knowledge base enrichment," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, 2019, pp. 229–240.

[13] J. Lee, W. Yoon, S. Kim, D. Kim, S. Kim, C. H. So, and J. Kang, "Biobert: a pre-trained biomedical language representation model for biomedical text mining," *Bioinformatics*, vol. 36, no. 4, pp. 1234–1240, 2020.

[14] A. Smirnova and P. Cudré-Mauroux, "Relation extraction using distant supervision: A survey," *ACM Comput. Surv.*, vol. 51, no. 5, 2018.

[15] A. R. S. H. Bach, H. Ehrenberg, J. Fries, S. Wu, and C. Ré, "Snorkel: Rapid training data creation with weak supervision," *Proceedings of the VLDB Endowment*, vol. 11, no. 3, 2017.

[16] S. Vashishth, P. Jain, and P. Talukdar, "Cesi: Canonicalizing open knowledge bases using embeddings and side information," in *Proceedings of the 2018 World Wide Web Conference*, ser. WWW '18. Republic and Canton of Geneva, CHE: International World Wide Web Conferences Steering Committee, 2018, p. 1317–1327.

[17] S. Dash, G. Rossiello, N. Mihindukulasooriya, S. Bagchi, and A. Gliozzo, "Joint entity and relation canonicalization in open knowledge graphs using variational autoencoders," *CoRR*, vol. abs/2012.04780, 2020.

[18] F. Manola and E. Miller, "RDF primer," WWW Consortium, Recommendation REC-rdf-primer-20040210, 2004.

[19] P. Qi, Y. Zhang, Y. Zhang, J. Bolton, and C. D. Manning, "Stanza: A python natural language processing toolkit for many human languages," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. Online: Association for Computational Linguistics, 2020, pp. 101–108.

[20] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*, 2013.

[21] R. I. Doğan, R. Leaman, and Z. Lu, "Ncbi disease corpus: A resource for disease name recognition and concept normalization," *Journal of Biomedical Informatics*, vol. 47, pp. 1–10, 2014.

[22] C.-H. Wei, Y. Peng, R. Leaman, A. P. Davis, C. J. Mattingly, J. Li, T. C. Wiegers, and Z. Lu, "Overview of the biocreative v chemical disease relation (cdr) task," in *Proceedings of the fifth BioCreative challenge evaluation workshop*, vol. 14, 2015.

[23] A. A. Morgan, Z. Lu, X. Wang, A. M. Cohen, J. Fluck, P. Ruch, and et al., "Overview of biocreative ii gene normalization," *Genome Biology*, vol. 9, no. 2, p. S3, 2008.

[24] Y. Zhang, Q. Chen, Z. Yang, H. Lin, and Z. Lu, "Biowordvec, improving biomedical word embeddings with subword information and mesh," *Scientific data*, vol. 6, no. 1, pp. 1–9, 2019.

[25] M. Krallinger, O. Rabal, S. A. Akhondi, M. P. Pérez, J. Santamaría, G. P. Rodríguez *et al.*, "Overview of the biocreative vi chemical-protein interaction track," in *Proceedings of the sixth BioCreative challenge evaluation workshop*, vol. 1, 2017, pp. 141–146.