Knowledge and Information Systems (2023) 65:1989–2016
https://doi.org/10.1007/s10115-022-01826-x

REGULAR PAPER

Information extraction pipelines for knowledge graphs

Mohamad Yaser Jaradeh[1] · Kuldeep Singh[2] · Markus Stocker[3] · Andreas Both[4] · Sören Auer[3]

Received: 29 September 2021 / Revised: 16 December 2022 / Accepted: 25 December 2022 /
Published online: 7 January 2023

Abstract
In the last decade, a large number of knowledge graph (KG) completion approaches were proposed. Albeit effective, these efforts are disjoint, and their collective strengths and weaknesses in effective KG completion have not been studied in the literature. We extend PLUMBER, a framework that brings together the research community's disjoint efforts on KG completion. We include more components into the architecture of PLUMBER to comprise 40 reusable components for various KG completion subtasks, such as coreference resolution, entity linking, and relation extraction. Using these components, PLUMBER dynamically generates suitable knowledge extraction pipelines and offers overall 432 distinct pipelines. We study the optimization problem of choosing optimal pipelines based on input sentences. To do so, we train a transformer-based classification model that extracts contextual embeddings from the input and finds an appropriate pipeline. We study the efficacy of PLUMBER for extracting the KG triples using standard datasets over three KGs: DBpedia, Wikidata, and Open Research Knowledge Graph. Our results demonstrate the effectiveness of PLUMBER in dynamically generating KG completion pipelines, outperforming all baselines agnostic of the underlying KG. Furthermore, we provide an analysis of collective failure cases, study the similarities and synergies among integrated components and discuss their limitations.

Keywords Information extraction · NLP pipelines · Software reusability · Semantic search · Semantic web

1 Introduction

Since the early twenty-first century [8], there have been continuous efforts to extend the Web with a global data graph using the Resource Data Framework (RDF) to publish structured

✉ Mohamad Yaser Jaradeh
jaradeh@l3s.de

1 L3S Research Center, Leibniz University Hannover, Hanover, Germany

2 Zerotha-Research and Cerence GmbH, Aachen, Germany

3 TIB Leibniz Information Centre for Science and Technology, Hanover, Germany

4 Anhalt University of Applied Sciences, Bernburg, Germany

data on the Web. One of the pivotal steps in this effort was the emergence of publicly available Knowledge Graphs (KG) such as DBpedia [4] and Yago [28] as large sources of structured data. Since then, these KGs have become a rich source of structured content used in various applications, including Question Answering (QA), fact checking, and dialog systems [6]. The research community addresses the problem of populating a KG from multiple angles, one of them is the semantic labeling of (semi-)structured data [1, 60], others use unstructured text to populate a KG. We focus on the latter efforts in our work. Numerous approaches to extract triple statements [66], keywords/topics [16, 17], tables [36, 37, 67], or entities [54, 55] from unstructured text to complement KGs have been developed by the community. Despite extensive research, public KGs are not exhaustive and require continuous effort to align newly emerging unstructured information to the concepts of the KGs.

In this article, we extend our previous work in [40] in the following regards:

- We formalize the workflow of the framework and include details of how components are selected and how pipelines are generated.
- We further include eight new components to the list of community-created components that PLUMBER can use to generate dynamic information extraction pipelines.
- We evaluate further on another widely used KG (Wikidata) with a big dataset that is used by the community.
- We further analyze error rates, time efficiency, and conduct a more detailed ablation study on the newly supported dataset and the old ones. Furthermore, we show how the framework can be used in a human-in-the-loop workflow to improve extractions and results.

1.1 Research problem

This work was motivated by an observation with recent approaches [20, 26, 54, 55, 67] that automatically align unstructured text to structured data on the Web. Such approaches are not viable in practice for extracting and structuring information because they only address very specific subtasks of the overall KG completion problem. If we consider the exemplary sentence *Rembrandt painted The Storm on the Sea of Galilee. It was painted in 1633.* To extract statements aligned with the DBpedia KG from the given sentences, a system must first recognize the entities and relation surface forms in the first sentence. The second sentence requires an additional step of coreference resolution, where *It* must be mapped to the correct entity surface form (namely, *The Storm on the Sea of Galilee*). The last step requires mapping of entity and relation surface forms to the respective DBpedia entities and predicates.

There has been extensive research in aligning concepts in unstructured text to KG, including entity linking [26, 30, 35], relation linking [6, 55, 57], and triple classification [25]. However, these efforts are disjoint, and little has been done to align unstructured text to the complete KG triples [41, 64]. Furthermore, many entity and relation linking tools have been reused in pipelines of QA systems [42, 58]. The literature suggests that once different approaches put forward by the research community are combined, the resulting pipeline-oriented integrated systems can outperform monolithic end-to-end systems. For example, Liang et al. [43] propose a modular QA system built reusing a variety of existing NLP components that outperform all existing end-to-end methods on the DBpedia-based QA task. For the KG completion task, however, to the best of our knowledge, approaches aiming at dynamically integrating and orchestrating various existing components do not exist.

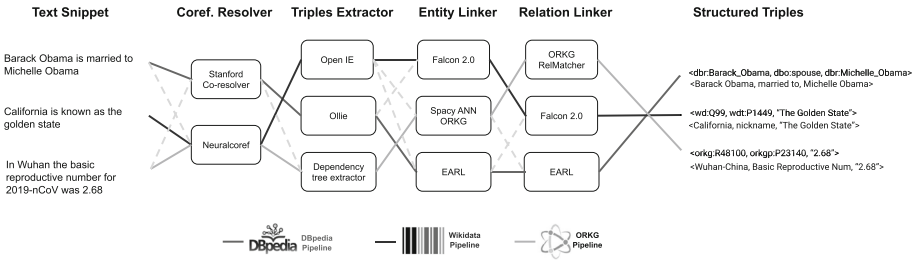


Fig. 1 PLUMBER in action: three information extraction pipelines that convert natural language text into structured triples aligned with respective knowledge graphs. The optimal pipeline for each text snippet and corresponding KG is highlighted. Throughout the article, we use CR, coreference resolution; TE, triple extraction; EL, for entity linking; RL, relation linking

1.2 Objective and contributions

Based on these observations, we dynamically assemble and evaluate information extraction pipelines from previously disjoint efforts on the KG completion task under a single umbrella. We present the PLUMBER framework (originally described here [40], cf. Fig. 1) for creating Information Extraction (IE) pipelines for KG completion. PLUMBER integrates 40 reusable components released by the research community for the subtasks entity linking, relation linking, text triple extraction (subject, predicate, object), and coreference resolution. Overall, there are 432 different composable KG completion pipelines[1] (generated by the possible combination of the available 40 components). PLUMBER implements a transformer-based classification algorithm that intelligently chooses the best pipeline based on the unstructured input text.

We perform an exhaustive evaluation of PLUMBER on the three large-scale KGs DBpedia, Wikidata [63] and Open Research Knowledge Graph (ORKG) [38] to investigate the efficacy of PLUMBER in creating KG triples from unstructured text. We demonstrate that independently of the underlying KG, PLUMBER can find and assemble different extraction components to produce optimized KG triple extraction pipelines, significantly outperforming existing baselines. In summary, we provide the following novel contributions:

- The PLUMBER framework is the first of its kind for dynamically assembling and evaluating information extraction pipelines based on sequence classification techniques and for a given input text. PLUMBER is easily extensible and configurable, thus enabling the rapid creation and adjustment of new information extraction components and pipelines. Researchers can also use the framework for running IE components independently for specific subtasks such as triple extraction and entity linking.
- A collection of 40 reusable IE components that can be combined to create 432 distinct IE pipelines.
- The exhaustive performance evaluation and our detailed ablation study of the integrated components and composed pipelines on various input text will guide future research for collaborative KG completion.

The article is organized as follows: Section 2 motivates our work. Related work is reviewed in Sect. 3, and we formalize our approach along with the problem definition in Sect. 4.

[1] For DBpedia: 3 CRs, 8 TEs, and 10 EL/RLs, hence, $3 * 8 * 10 = 240$. And the same for Wikidata: 3CRs*8TEs*7EL/RL = 168. For the ORKG: 4CRs*3TEs*2EL/RL = 24 pipeline. In total: $240 + 168 + 24 = 432$.

Section 5 presents PLUMBER, which is evaluated in Sect. 6. Section 7 discusses the results. Section 8 concludes and outlines directions for future work.

2 Motivating example

Let us consider as a running example the sentence *Rembrandt painted The Storm on the Sea of Galilee. It was painted in 1633*. The sentence can be represented using the DBpedia vocabulary as follows:

```
@prefix dbr: <http://dbpedia.org/resource/>.
@prefix dbp: <http://dbpedia.org/property/>.

dbr:Rembrandt dbp:artist dbr:The_Storm_on_the_Sea_of_Galilee.
    dbr:The_Storm_on_the_Sea_of_Galilee dbp:year "1633".
```

Multiple steps are required to extract these formally represented statements from the given text. First, the pronoun *it* in the second sentence should be replaced by *The Storm on the Sea of Galilee* using a coreference resolver. Next, a triple extractor should extract the correct text triples from the natural language text, i.e., `<Rembrandt, painted, The Storm on the Sea of Galilee>`, and `<The Storm on the Sea of Galilee, painted in, 1633>`. In the next step, the entity and relation linking component aligns the entity and relation surface forms extracted in the previous step to the DBpedia entities: `dbr:Rembrandt` for *Rembrandt van Rijn*, and `dbr:The_Storm_on_the_Sea_of_Galilee` for *The Storm on the Sea of Galilee*, and for relations: `dbp:artist` for *painted*, and `dbp:year` for *painted in*.

There exists a plethora of techniques and components for extracting such statements from a given text. However, the performance of the tools varies widely and depends strongly on the input text (cf. [58]). Figure 2 illustrates our running example and shows three PLUMBER IE pipelines with different results. In Pipeline 1, the coreference resolver is unable to map the pronoun *it* to the respective entity in the previous sentence. Moreover, the triple extractor generates incomplete triples, which also hinders the task of the entity and relation linker in the last step. Pipeline 2 uses a different set of components, and its output differs from the first pipeline. Here, the coreference resolution component is able to correctly co-relate the pronoun *it* to *The Storm on the Sea of Galilee* and extract the text triple correctly. However, the overall result is only partially correct because the second triple is not extracted. Also, the entity linking component is not able to spot the second entity. It is important to note that the entity linking component in the second pipeline (i.e., DBpedia Spotlight [18]) does not perform relation linking. Hence, even if the information extraction step produces the correct results, triples could not be mapped correctly.

Pipeline 3 correctly extracts both triples. This pipeline employs the same component as the second pipeline for coreference resolution but also includes an additional information extraction component (i.e., ReVerb [29]) and a joint entity and relation linking component, namely Falcon [54]. With this combination of components, the text triple extractors were able to compensate for the loss of information in the second pipeline by adding one more component. Using the extracted text triples, the last component of the pipeline, a joint entity and relation linking tool, can map both triple components correctly to the corresponding KG entities.

With the availability of a large pool of components, such as those employed in PLUMBER, a suitable pipeline for a given text can be identified experimentally by executing all possible pipelines. However, this brute force approach is impractical. Therefore, we suggest a machine-

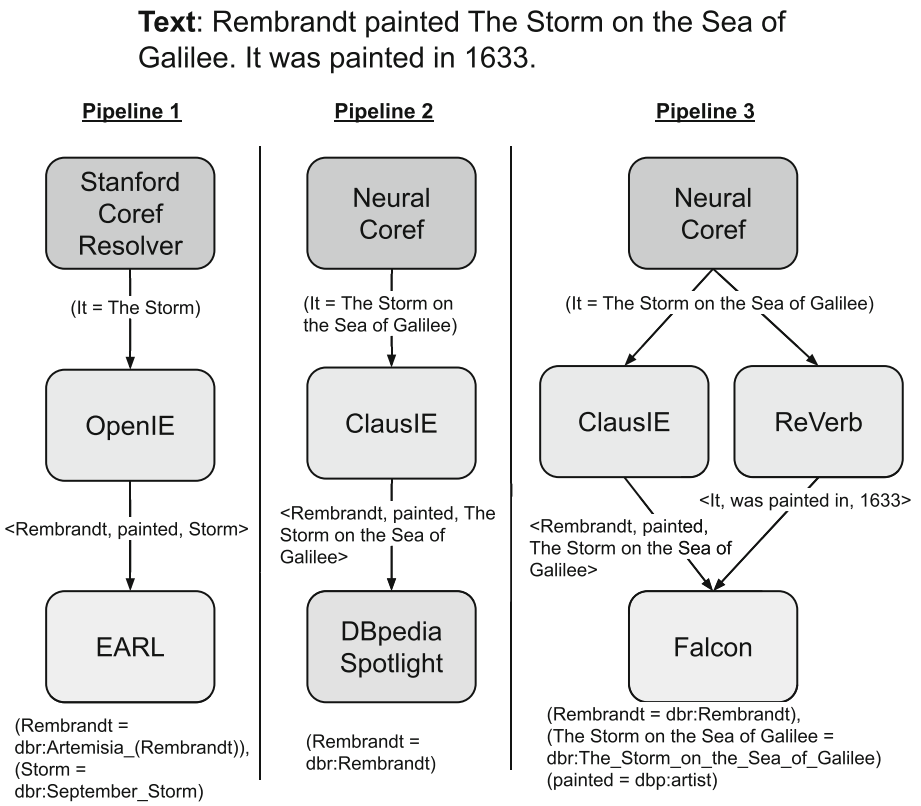


Fig. 2 Three example information extraction pipelines showing different results for the same text snippet. Each pipeline consists of coreference resolution, triple extractors, and entity/relation linking components. For the sake of readability, we hide some intermediate triples and mappings. DBpedia was chosen over other KGs because it has human-readable URIs. The first layer from the top denotes the coreference resolution components in the pipelines. The middle one refers to textual triple extractors in the pipelines. And the bottom one indicates joint entity and relation linkers

learning model (cf. Section 4.3) for identifying a suitable candidate pipeline for a given input text.

3 Related work

In the last decade, many open-source tools have been released by the research community to tackle IE tasks in the context of KG completion (see Table 2). These IE components are not only used for end-to-end KG triple extraction but also for various other tasks, such as:

3.1 Text triple extraction

The task of open information extraction is a well-studied researched task in the NLP community [3]. It relies on NER (Named Entity Recognition) and RE (Relation Extraction). MinIE [34] extracts relation surface forms. SalIE [52] uses MinIE in combination with PageRank

and clustering to find facts in the input text. Furthermore, OpenIE [3] leverages linguistic structures to extract self-contained clauses from the text. For a detailed survey on open information extraction, we point readers to a comprehensive survey by Niklaus et al. [51]. Another system Graphene [11] employs two layered transformations of clausal and phrasal embedding to simplify text and extract linguistic triples.

3.2 Entity and relation linking

Entity and relation linking is a widely studied researched topic in the NLP, Web, and Information Retrieval research communities [5, 6, 20]. Often, entity and relation linking is performed independently. DBpedia Spotlight [18] is one of the first approaches for entity recognition and disambiguation over DBpedia. TagMe [30] links entities to DBpedia using in-link matching to disambiguate candidates entities. Open Tapioca [20] uses semantic matching of entity candidates for Wikidata. Other tools such as RelMatch [57] do not perform entity linking and only focus on linking the relation in the text to the corresponding KG relation. Recon [6] assumes entities are already linked in the text and aims to map relations between the entities to the KG using a graph neural network. EARL [26] is a joint linking tool over DBpedia and models the task as a generalized traveling salesperson problem. Sakor et al. [54] proposed Falcon, a linguistic rule-based tool for joint entity and relation linking over DBpedia. Falcon 2.0 [55] performs joint entity and relation linking on Wikidata.

3.3 Coreference resolution

This task is used in conjunction with other tasks in NLP pipelines to disambiguate text and resolve syntactic complexities. The Stanford Coreference Resolver [53] uses a multipass sieve of deterministic coreference models. Clark and Manning [15] use reinforcement learning to fine-tune a neural mention-ranking model for coreference resolution. More recently, Sanh et al. [56] introduced a hierarchical model that is capable of multitask learning including coreference resolution.

3.4 Frameworks and dynamic pipelines

There have been few attempts in various domains aiming to consolidate the disjoint efforts of the research community under a single umbrella for solving a particular task. The Gerbil platform [62] provides an easy-to-use web-based platform for the agile comparison of entity linking tools using multiple datasets and uniform measuring approaches. OKBQA [42] is a community effort for the development of multilingual open knowledge base and QA systems. Frankenstein integrates 24 QA components to build QA systems collaboratively on-top of the Qanary integration framework [10]. PLUMBER is closely inspired by the concept of Frankenstein but also differs in several ways. Firstly, PLUMBER surpasses Frankenstein in the number of integrated components. Furthermore, Frankenstein implements one classifier per component to predict the performance of each component for a given input question. Frankenstein treats each task independently. Hence, even if a classifier predicts that a specific component should be a part of the pipeline, it may be possible that the error propagated from previous steps reduces the overall performance. We argue that for selecting the best pipeline it is crucial to consider the end-to-end performance instead of treating each component independently. Hence, we trained a single transformer-based classifier that not only

learns from the linguistic features but also from the results of other pipelines while suggesting the optimal pipeline. The results in Table 5 support our hypothesis, which is fundamentally different from the Frankenstein approach.

3.5 End-to-end extraction systems

More recently, end-to-end systems are gaining more attention due to the boom of deep learning techniques. Such systems draw on the strengths of deep models and transformers [23, 44]. Kertkeidkachorn and Ichise [41] present an end-to-end system to extract triples and link them to DBpedia. Other attempts such as KG-Bert [66] leverage deep transformers (i.e., BERT [23]) for the triple classification task, given the entity and relation descriptions of a triple. KG-Bert does not attempt end-to-end alignment of KG triples from a given input text. Liu et al. [45] design an encoder–decoder framework with an attention mechanism to extract and align triples to a KG.

4 Approach formalization

An end-to-end information extraction pipeline is composed of all IE tasks (i.e., KG completion subtasks) needed to transform a sequence of natural language text into a set of structured triples in the form of (*subject, predicate, object*). However, since each component of the IE pipeline performs different tasks, we first formalize the interfaces of the IE tasks. We then state the problem, a formal approach implemented in PLUMBER and how pipelines are generated.

4.1 Defining various IE task interfaces

We formally define a pipeline P as a triple extraction and alignment function, from text T to a set of aligned KG triples Υ.

$$P : T \rightarrow \Upsilon \tag{1}$$

A text element T is a white-spaced separated sequence of words and sentences. Let $\oplus$ be the composition operator (i.e., the input of a function is the output of the previous one). We define P as a composition of four subfunctions, each corresponding to an IE task in the pipeline:

$$P := \Theta \oplus Z \oplus \Psi \oplus \Omega \tag{2}$$

An overview of the notation used in the problem formalization can be found in Table 1.

(i) Coreference resolution (CR) The first step is to disambiguate the input text and replace pronouns and acronyms with its associated entity mention. This step is formally defined as:

$$\theta := T \rightarrow T', T' := \Lambda(T, c), c := \{(m, a) \subseteq T | m, a \neq \emptyset\} \tag{3}$$

where T' is a text resulting from the transformation function Λ and the coreference chain c. m is the mention in the text T and a is the pronoun or other alias that refers to the mention m. The resulting text T' is a text without ambiguities in mentions and pronouns.

(ii) Text triple extraction (TE) For the second step, we define a text triple as combination of three keyphrases or text snippets usually in the form of (subject, predicate, object). TE

Table 1 Symbol notation used to formalize PLUMBER pipelines interfaces

P	IE extraction pipeline P composed of several IE components
Θ, θ	Set of coreference resolution (CR) components Θ and individual CR components $\theta \in \Theta$
Z, ζ	Set of triple extractor (TE) components Z and individual TE components $\zeta \in Z$
Ψ, ψ	Set of entity linking/relation linking components Ψ and individual EL/RL components ψ
Ω, ω	Set of end-to-end (E2E) components Ω and individual E2E components ω
Υ, υ	Set of KG triples Υ and individual KG triple υ
$\Lambda(.)$	Transformation function that enriches a text T
c	Coreference chain $c = \{m, a\}, m := Mention, a := Alias$
γ	Text triple $\gamma =< s, p, o >, s := subject, p := predicate, o := object$
λ	Mapping pair $\lambda = (m, u), m := Mention, a :=$ KG uri
K	Knowledge Graph K also referred to as Knowledge Base

components extract such textual triples from the disambiguated text T'. Formally:

$$\zeta := T' \rightarrow \bar{T}, \bar{T} := \Lambda(T', \gamma), \gamma := \{(s, p, o) \in T' | s, p, o \neq \emptyset\} \tag{4}$$

Each triple set γ is formed of triplets (i.e., $\{s, p, o\}$). The transformation function Λ in this step enriches the disambiguated text T' with set of triples producing $\bar{T}$.

(iii) Entity and relation linking (EL/RL) The third and concluding step in the pipeline is the alignment of triples in $\bar{T}$ to a knowledge graph K. We define a KG triple $\upsilon \in \Upsilon$ similarly to a text triple γ whereby they have the same structure alike $\{s, p, o\}$. However, a KG triple assumes that the subject s and predicate p components are URIs referring to a KG K, and the object o is either a URI as well or a literal value (i.e., string snippet). Formally, it is defined as:

$$\begin{aligned} \psi &:= \bar{T} \rightarrow \Upsilon, \Upsilon := M(\bar{T}, \lambda, K), \\ \lambda &:= \{(m, u), m \in \bar{T}, u \subset K | m, u \neq \emptyset\} \end{aligned} \tag{5}$$

Here $M(\bar{T}, \lambda, K)$ denotes a mapping function that takes the enriched text $\bar{T}$, a knowledge graph K (which is constant), and a set of mappings λ to construct a set of aligned KG triples Υ.

(iv) End-to-end extraction (E2E) resembles the composed pipelines P and is defined as $\omega := T \rightarrow \Upsilon$. End-to-End pipelines produce results that are concatenated to results of the Ψ components.

4.2 Generating candidate IE pipelines

Generating candidate pipelines relies on the interfaces of the IE tasks and the set of requirements r. The set of pipelines $\xi(r)$ is populated with candidate pipelines ϱ_i following a composition:

$$\varrho_i := \oplus_{\tau \in \Delta} \left\{ \chi_r^{\tau} \right\} \tag{6}$$

where Δ the list of IE tasks following the specifications of the interfaces formalized previously. χ_r^{τ} a set of possible IE components that perform the IE task τ and comply with the requirements r (i.e., the knowledge graph K). It is created by concatenating IE components

carrying out a task τ (i.e., concatenating components is running them in parallel and concatenating the results). If $\|$ denotes the concatenation of IE components, then the set of IE components χ is defined as follows:

$$\chi_r^{\tau} := \|_i \{C(\tau)\}, \text{for } i = 1, \ldots, n \tag{7}$$

where $C(\tau)$ retrieves a component from the set of IE components addressing task τ and n is the number of needed components per task. The n parameter is introduced to limit the space of candidates generation. Hence, pipelines can be generated and added to the pool of IE pipeline selection mechanism.

4.3 Determining suitable IE pipeline

4.3.1 Problem

Here we tackle the pipeline selection problem. The pipeline selection problem deals with finding a suitable pipeline of IE components ρ_s^r, for a sequence of text s and a set of requirements r. Formally, we define the optimization problem as follows:

$$\rho_s^r := \arg \max_{\varrho \in \xi(r)} \{\mathcal{Q}(\varrho, s)\} \tag{8}$$

where $\xi(r)$ constitute the set of IE extraction pipelines that conform to the requirements r and $\mathcal{Q}(\varrho, s)$ corresponds to the estimated performance of a pipeline ϱ on a text sequence s.

4.3.2 Solution

We understand the problem at hand as a k-class classification problem [46]. In order to be able to solve this problem, we decompose it into a series of smaller and simpler two-class problems. Suppose we have k pipelines (i.e., classes). Let $\mathcal{W}$ be the training set for a k-class problem:

$$\mathcal{W} := \{(X_l, Y_l)\}_{l=1}^{L} \tag{9}$$

where X_l is an input text sequence, $Y_l \in \mathbb{R}^k$ is the desired output, and L is the number of training data. Following the class decomposition methods [2, 14, 31], a k-class problem can be divided into k two-class problems. The training set for each of the two-class problems is as follows:

$$\mathcal{W}_i := \left\{(X_l, y_l^i)\right\}_{l=1}^{L}, \text{for } i = 1, ..., k \tag{10}$$

where $y_l^i \in \mathbb{R}^1$ is the desired output defined as:

$$y_l^i = \begin{cases} 1 - \epsilon & , X_l \in \mathcal{C}_i \\ \epsilon & , X_l \in \bar{\mathcal{C}}_i \end{cases} \tag{11}$$

where ϵ is a small positive real number and $\bar{\mathcal{C}}_i$ denotes all classes except $\mathcal{C}_i$. A sequence classifier Γ can now be trained on the decomposed training dataset $\mathcal{W}$ and is able to classify the performance of a pipeline $\mathcal{Q}(\varrho, s)$ into a class $\kappa \in k$ that maps to a pipeline configuration (i.e., a set of IE components). This is the best pipeline to run on the input sequence s. Hence, we rewrite Eq. 8 as follows:

$$\rho_s^r := \arg \max_{\kappa \in k} \{\Gamma(s, \xi(r))\} \tag{12}$$

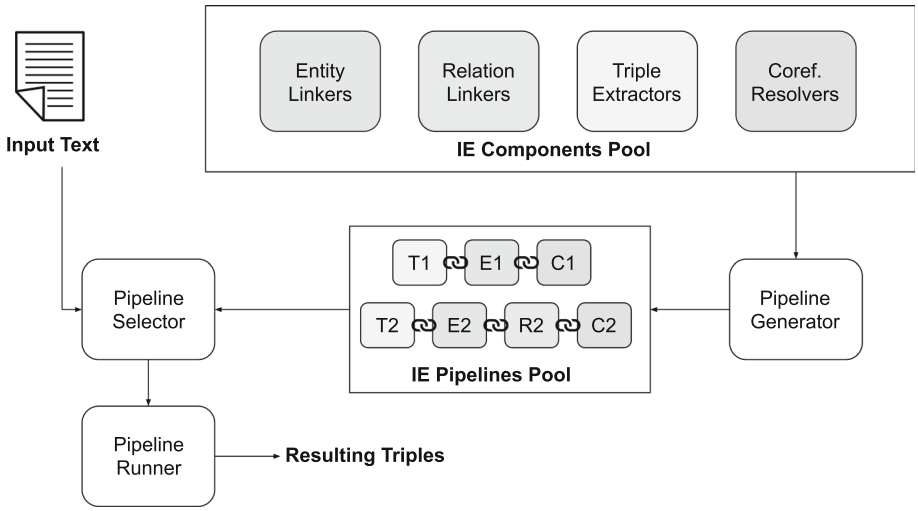


Fig. 3 Overview of PLUMBER's architecture highlighting the components for pipeline generation, selection, and execution. PLUMBER receives an input sentence and requirement (underlying KG) from the user. The framework intelligently selects the optimum pipeline based on the contextual features captured from the input sentence

which stands for a problem of classifying a sequence of text s based on a set of candidate pipelines $\xi(r)$.

5 Dynamic pipelining framework

PLUMBER orchestrates and evaluates IE components to select the most suitable pipeline configuration based on the input text for the KG completion task. We now detail its architecture.

5.1 Architecture overview

PLUMBER has a modular design (see Fig. 3) where each component is integrated as a microservice. To ensure a consistent data exchange between components, the framework maps the output of each component to a homogeneous data representation using the Qanary [10] methodology. Qanary follows the linked data principles [7] employing an ontology to systematize the process of connecting components. PLUMBER follows three design principles: (i) *Isolation*, the IE components are independent from each other and they can be accessed through exchangeable interfaces; (ii) *Reusability*, the framework is open source and can be reused in different contexts and variations; (iii) *Extensibility*, PLUMBER provides common interfaces to expand and add further IE components in such a way that new components and tools are directly integrated in the framework and operate within the pipelines. The design of the framework is inspired by the work of [10, 58, 62]. The Qanary framework was used as basis for our architecture because it demonstrated why each of the previously mentioned principles is required, and it has been used extensively by other frameworks and systems such as [24, 58].

PLUMBER uses a RoBERTa [44]-based classifier that given a text and a set of requirements, it predicts the most suitable pipeline to extract KG triples (further details in Sect. 6.2).

PLUMBER includes the following modules:

- *IE components pool* All information extraction components that are integrated within the framework are parts of the pool. The components are divided based on their respective tasks, i.e., coreference resolution, text triple extraction, as well as entity and relation linking (cf. Table 2).
- *Pipeline generator* This module creates possible pipelines depending on the requirements of the components (i.e., the underlying KG). Users can manually select the underlying KG and, using the metadata associated with each component, PLUMBER aggregates the components for the concerned KG (cf. Sects. 4.1, 4.2).
- *IE pipelines pool* PLUMBER stores the configurations of the possible pipelines in the pool of pipelines for faster retrieval and easier interaction with other modules.
- *Pipeline selector* Based on the requirements (i.e., underlying KG) and the input text, a RoBERTa-based model extracts contextual embeddings from the text and classifies the input into one of the possible classes. Each class corresponds to one pipeline configuration that is held in the IE pipelines pool (cf. Sect. 4.3 for details of proposed optimization solution).
- *Pipeline runner* Given the input text, and the generated pipeline configuration, the module executes the pipeline to add the extracted triples to the KG.

Since PLUMBER seeks to converge the disjoint efforts of the community under one umbrella, it does not reinvent the wheel or re-implement any IE component it encompasses. PLUMBER relies on whatever interface the different IE components provide (see Table 2). In other words, if public APIs are available, then such interfaces are reused by the framework. However, if no API is available, then the IE component is integrated locally into the codebase (e.g., ClausIE & MinIE).[2]

6 Evaluation

In this section, we detail the empirical evaluation of the framework in comparison with baselines on different datasets and knowledge graphs. As such, we study the following research question **RQ**: *How does the dynamic selection of pipelines based on the input text affect the end-to-end KG completion task?*. Throughout the paper, we use the term dynamic pipeline to refer to dynamic pipeline generation.

6.1 Assumptions

Our empirical evaluation of the PLUMBER framework is conducted with one assumption in mind. All externally accessible components (i.e., via APIs) are considered always operational, and any type of failure from their side is considered empty result set and treated accordingly in the evaluation procedure.

[2] See below for more details about such tools.

6.2 Experimental setup

Knowledge graphs To study the effectiveness of PLUMBER in building dynamic KG completion pipelines, we use the following KGs during our evaluation:
DBpedia [4] (DBP) is containing information extracted automatically from Wikipedia info boxes. DBP consists of approximately 11.5B triples [54].
Wikidata [63] (WD) is a crowd-sourced knowledge base providing structured data for integration in Wikipedia. In contrast to DBP, WD also allows user-created entities and predicates. WD consists of over 4.9B triples [47].
Open research knowledge graph [38] (ORKG) collects structured scholarly knowledge published in research articles, using crowd sourcing and automated techniques. In total, ORKG consists of approximately 1.7 M triples.
Datasets Throughout our evaluation, we employed a set of both existing and newly created datasets for structured triple extraction and alignment to knowledge graphs: the WebNLG [33] dataset for DBP, the T-Rex [27] dataset for WD, and COV-triples for ORKG. The choice of datasets relied on three aspects: (i) the availability of the data (i.e., how much it is used in the community and the size of the data), (ii) the domain of the dataset (i.e., what is the dataset about), (iii) whether the selected IE components can handle the dataset (i.e., can extract entities and triples from it).
WebNLG[3] is the Web Natural Language Generation Challenge. The challenge introduced the task of aligning unstructured text to DBpedia. In total, the dataset contains 46K triples with 9K triples in the testing and 37K in the training set.
T-Rex[4] is a dataset of a large-scale alignment of Wikipedia text with Wikidata. It comprises approximately 11 M triples aligned to the WD knowledge graph. The data were split using an 80/20 ratio for training and testing.

> **Note**: Though COV-triples is a small dataset in comparison with the others, it is used here to show the generalizability of the system specially on domains where not much training data is available.

COV-triples is a handcrafted dataset that focuses on COVID-19-related scholarly articles. The COV-triples dataset consists of 21 abstracts from peer-reviewed articles and aligns the natural language text to the corresponding KG triples in the ORKG. Three semantic Web researchers verified annotation quality, and triples approved by all three researchers are part of the dataset. The dataset contains only 75 triples. Hence, we use the WebNLG dataset for training, and 75 triples are used as a test set.
Components and implementation The PLUMBER framework integrates 40 components, shown in Table 2 along with the associated subtasks and underlying KG.

> **Note**: the architecture of PLUMBER allows for easy integration of new components.

The IE components were chosen purely from the background knowledge of the authors and related work research. Our framework is implemented in Python, and we adapted a pretrained version of RoBERTa from its public GitHub[5] and fine-tuned it on the employed

[3] https://webnlg-challenge.loria.fr/.
[4] https://hadyelsahar.github.io/t-rex/.
[5] https://github.com/pytorch/fairseq/.

datasets. All experiments were performed on a system with 768GB RAM, 96 CPUs, and one GPU (NVIDIA GeForce 1080 Ti). The implementation code of PLUMBER and all related resources are publicly available online.[6]

Baselines We include the following baselines:

- *T2KG* [41] is an end-to-end static system that aligns a given natural language text to DBpedia KG triples.
- *Frankenstein* [58] dynamically composes Question Answering pipelines over the DBpedia KG. It employs logistic regression-based classifiers for each component for predicting the accuracy and greedily composes a dynamic pipeline of the best components per task. We adapted Frankenstein for the KG completion over DBpedia and Wikidata since some of its components also perform entity and relation linking.
- *KnowledgeNet* [49] represents a benchmarking dataset for KG completion alongside a baseline model. The KnowledgeNet baseline model performs knowledge graph completion and population on the WD knowledge graph.

Training the model PLUMBER relies on a classification model to find a suitable pipeline. Each pipeline is represented as a class, making this a multiclass classification problem. To train the model, for every entry in the datasets, every possible pipeline is composed run on the input snippet.

> **Note**: Running all possible pipelines is just needed for the training phase of the system, afterwards the framework generalizes on unseen data.

The results in terms of F1 scores are used to decide which pipeline performed better than the others. Next step for our underlying model (RoBERTa) is to create contextualized embeddings from the input text and learn to classify it into its corresponding class (i.e., the better performing pipeline from the IE pipelines pool). We choose a transformer-based architecture due to its ability to encode the contextual knowledge from the input text, providing a more accurate classification.

With the underlying model trained on the data, it can now dynamically compose information extraction pipelines on unseen data. However, if new IE components are added to the pool of available components, the framework would require the retraining of the model to include these new components as candidates.

6.3 Experiments

This section summarizes a variety of experiments to compare the PLUMBER framework against other baselines. Note that evaluating the performance of individual components or their combination is out of this evaluation's scope, since they were already used, benchmarked, and evaluated in the respective publications. We report values of the standard metrics precision (P), recall (R), and F1 score (F1) adapted from [13]. In all experiments, end-to-end components (e.g., T2KG) are not part of PLUMBER.

Performance of static pipelines In this experiment, we report results of the static pipelines, i.e., no dynamic selection of a pipeline-based on the input text is considered.

[6] https://github.com/YaserJaradeh/ThePlumber.

Table 2 IE components implemented and integrated within the PLUMBER framework along with their respective publications, API links, underlying KGs, and respective task

Pipeline component	IE Task	Knowledge graph	Open source	Custom built	Rest API
Ollie [48]	TE	–	✓	✗	✗
OpenIE [3]	TE	–	✓	✗	✗
ClausIE [19]	TE	–	✓	✗	✗
MinIE [34]	TE	–	✓	✗	✗
POS Extractor[a]	TE	–	✓	✓	✗
Dependency Extractor[b]	TE	–	✓	✓	✗
Graphene [11]	TE	–	✓	✗	✗
ReVerb [29]	TE	–	✓	✗	✗
R0 Extractor	TE	ORKG	✓	✓	✗
Stanford KBP Extractor [12]	TE	–	✓	✗	✗
Falcon [54]	EL+RL	DBP	✓	✗	✓
Falcon 2.0 [55]	EL+RL	WD	✓	✗	✓
EARL [26]	EL+RL	DBP	✓	✗	✓
Spacy ANN[c]	EL+RL	DBP+WD	✓	✓	✗
Spacy ANN[c]	EL+RL	ORKG	✓	✓	✗
Falcon NER [54] + *ES*[d]	EL+RL	DBP+WD	✓	✓	✗
Falcon 2.0 NER + *ES*[d]	EL+RL	WD	✓	✓	✗
EARL NER [26] + *ES*[d]	EL+RL	DBP+WD	✓	✓	✗
Meaning Cloud[e]	EL	DBP	✗	✗	✓
Text Razor[f]	EL	DBP+WD	✗	✗	✓
DBpedia Spotlight [18]	EL	DBP	✓	✗	✓
TagMe [30]	EL	DBP	✓	✗	✓
OpenTapioca [20]	EL	WD	✓	✗	✓
TagMe NER [30] + *ES*[d]	EL	DBP+WD	✓	✓	✗
Ambiverse-nlu [35]	EL	WD	✓	✗	✗
RelMatch [57]	RL	DBP	✓	✗	✗
Stanford Coref Resolver [53]	CR	–	✓	✗	✗
NeuralCoref [15]	CR	–	✓	✗	✗
PyCobalt[g]	CR	–	✓	✗	✗
HMTL [56]	CR	–	✓	✗	✗

[a] Based on https://github.com/tdpetrou/RDF-Triple-API/
[b] Adapted from https://github.com/anutammewar/extract_triplets/
[c] https://github.com/microsoft/spacy-ann-linker/
[d] https://www.elastic.co/elasticsearch/
[e] https://www.meaningcloud.com/
[f] https://www.textrazor.com/technology/
[g] https://github.com/Lambda-3/PyCobalt

Table 3 Performance comparison of the PLUMBER static pipeline against the baselines on different KGs

System	Dataset	Knowledge graph	P	R	F1	# Mapped triples
T2KG [41]	WebNLG	DBP	0.133	0.140	0.135	1.26K/9.0K
KnowledgeNet [49]	T-Rex	WD	0.243	0.254	0.247	0.56 M/2.2 M
Frankenstein [58]	WebNLG	DBP	0.177	0.189	0.181	1.70K/9.0K
	T-Rex	WD	0.228	0.249	0.238	0.55 M/2.2 M
PLUMBER	WebNLG	DBP	**0.210**	**0.225**	**0.215**	2.02K/9.0K
	T-Rex	WD	**0.282**	**0.296**	**0.289**	0.65 M/2.2 M

The total number of mapped triples in test set (Extracted/Expected) is given in the last column to indicate how many triples the systems produce regardless of correctness

> **Note**: T2KG can only be evaluated against WebNLG because it is a DBpedia-oriented tool and is built with only that direction in mind. Same goes for KnowledgeNet on Wikidata KG and it does not work on other KGs. Hence, their evaluation was restricted to their corresponding compatible dataset.

We ran all 432 pipelines, and Table 3 reports the performance of the best PLUMBER pipeline against the baselines. PLUMBER static pipeline for DBpedia comprises NeuralCoref [15] for coreference resolution, OpenIE [3] for text triple extraction, TagMe [30] for EL, and Falcon [54] for RL tasks. For Wikidata, the static pipeline contains NeuralCoref [15] for coreference resolution, Graphene [11] for text triple extraction, Falcon 2.0 [55] jointly for EL and RL tasks. Also, in case of Frankenstein, we choose its best-performing static pipeline. Results illustrated in Table 3 confirm that the static pipeline composed by the components integrated in PLUMBER outperforms all baselines on DBpedia and Wikidata. We observe that the performance of pipeline approaches is better than an end-to-end monolithic KG completion approaches. Although the PLUMBER pipeline outperforms the baselines, the overall performance is relatively low. All our components have been trained on distinct corpora in their respective publications, and our aim was to put them together to understand their collective strengths and weaknesses. Note Frankenstein addresses the QA pipeline problem and not all components are comparable and can be applied in the context of information extraction. Thus, we integrated the NeuralCoref coreference resolution component and the OpenIE triple extraction component used in the PLUMBER static pipeline into Frankenstein in order to provide the same experimental settings.

Static pipeline for scholarly KG In order to assess how PLUMBER performs on domain-specific use cases, we evaluate the static pipelines' performance on a scholarly knowledge graph. We use the COV-triples dataset for ORKG. To the best of our knowledge, no baseline exists on completing KGs of research contribution descriptions over ORKG. Hence, we execute all static pipelines in PLUMBER tailored to ORKG to select the best one as shown in Table 4. The best-performing pipeline over the COV-triples was composed of the HMTL [56] coreference resolution component in combination with our own custom created R0 Extractor and Spacy ANN joint entity and relation linkers.[7] PLUMBER pipelines over ORKG extract statements determining the reproductive number estimates for the COVID-19 infectious disease from scientific articles as shown below.

[7] Adaption of https://github.com/microsoft/spacy-ann-linker/, trained on the ORKG entites and relations.

Table 4 Performance of the best-performing pipeline for scholarly knowledge extraction from COVID-19 research papers

System	Dataset	Knowledge graph	P	R	F1	# Mapped triples
PLUMBER	COV-triples	ORKG	0.403	0.423	0.413	32/75

Table 5 Tenfold cross-validation of pipeline selection classifiers wrt Precision, recall, and F1 score

Pipeline selection approach	Dataset	Knowledge graph	Classification		
			P	R	F1
Random	WebNLG	DBP	0.081	0.092	0.086
	T-Rex	WD	0.090	0.103	0.096
	COV-triples	ORKG	0.092	0.114	0.102
Frankenstein	WebNLG	DBP	0.732	0.751	0.741
	T-Rex	WD	0.770	0.791	0.780
	COV-triples	ORKG	0.832	0.858	0.845
PLUMBER	WebNLG	DBP	**0.877**	**0.900**	**0.888**
	T-Rex	WD	**0.891**	**0.912**	**0.901**
	COV-triples	ORKG	**0.901**	**0.917**	**0.909**

```
@prefix orkg: <http://orkg.org/orkg/resource/>.
@prefix orkgp: <http://orkg.org/orkg/property/>.

orkg:R48100   orkgp:P16022   "2.68" .
```

In this example, *orkg:R48100* refers to the city of Wuhan in China in the ORKG and *orkgp:P16022* is the property *has R0 estimate (average)*. The number "2.68" is the reproductive number estimate.

Comparison of the classification approaches for dynamic pipeline selection In this experiment, we study the effect of the transformer-based pipeline selection approach implemented in PLUMBER against the pipeline selection approach of Frankenstein. For a comparable experimental setting, we re-use Frankenstein's classification approach in PLUMBER, keeping the underlying components precisely the same. We perform a tenfold cross-validation for the classification performance of the employed approach. Table 5 demonstrates that the PLUMBER pipeline selection outperforms Frankenstein across all KGs.

Performance comparison for KG completion task Our third experiment focuses on comparing the performance of PLUMBER against previous baselines for an end-to-end KG completion task. We also report the values of best-performing static pipelines from Table 3. The results in Table 6 illustrate that the dynamic pipelines built using PLUMBER for KG completion outperform the best static pipelines of PLUMBER as well as the dynamically selected pipelines by Frankenstein. The end-to-end baselines, such as [41, 49], significantly underperform compared to dynamic pipelines. We also observe that in cross-domain experiments for COV-triples datasets, dynamically selected pipelines perform better than the static pipeline. In the cross-domain experiment, the static and dynamic PLUMBER pipelines are relatively better-performing than the other two KGs. Unlike components for DBpedia and Wikidata, components integrated into PLUMBER for ORKG are customized for KG triple extraction. We conclude that when components are integrated into a framework such as PLUMBER aiming

Table 6 Overall performance comparison of static and dynamic pipelines for the KG completion task

System	Dataset	Knowledge graph	Performance		
			P	R	F1
T2KG [41]	WebNLG	DBP	0.133	0.140	0.135
KnowledgeNet [49]	T-Rex	WD	0.243	0.254	0.247
Frankenstein (static) [58]	WebNLG	DBP	0.177	0.189	0.181
	T-Rex	WD	0.228	0.249	0.238
PLUMBER (static)	WebNLG	DBP	0.210	0.225	0.215
	T-Rex	WD	0.282	0.296	0.289
	COV-triples	ORKG	0.403	0.423	0.413
Frankenstein (dynamic) [58]	WebNLG	DBP	0.199	0.208	0.203
	T-Rex	WD	0.244	0.263	0.253
	COV-triples	ORKG	0.403	0.424	0.413
PLUMBER (dynamic)	WebNLG	DBP	**0.287**	**0.307**	**0.297**
	T-Rex	WD	**0.361**	**0.397**	**0.378**
	COV-triples	ORKG	**0.411**	**0.437**	**0.424**

for the KG completion task, it is crucial to select the pipeline based on the input text dynamically. The superior performance of PLUMBER shows that the dynamic pipeline selection for KG completion has a positive impact agnostic of the underlying KG and dataset. This also answers our overall research question *How does the dynamic selection of pipelines based on the input text affect the end-to-end KG completion task?*.

6.4 Ablation studies

The performance of PLUMBER and baselines on all the employed datasets is relatively low. Hence, in the ablation studies our aim is to provide a holistic picture of underlying errors, collective success, and failures of the integrated components.

In the first study, we calculate the proportion of errors in PLUMBER. The modular architecture of the proposed framework allows us to benchmark each component independently. We consider the erroneous cases of PLUMBER on the test set of the WebNLG dataset. We calculate the performance (F1 score) of the PLUMBER dynamic pipeline (cf. Table 6) at each step in the pipeline. Figure 4 presents the results of the error evaluation. Each box in the figure corresponds to an IE task. The results show that the coreference resolution components caused 21.54% of the errors, 33.71% are caused by text triple extractors, 18.17% by the entity linking components, and 26.58% are caused by the relation linking components.

We conclude that the text triple extractor components contribute to the largest chunk of the errors over DBpedia. One possible reason for their limited performance is that open-domain information extracting components were not initially released for the KG completion task. Also, these components do not incorporate any schema or prior knowledge to guide the extraction. We observe that the errors mainly occur when the sentence is complex (with more than one entity and relation), or relations are not explicitly mentioned in the sentence. We further analyze the text triple extractor errors. The error analysis at the level of the triple subject, predicate, and object showed that most errors are in predicates (40.17%) followed by objects (35.98%) and subjects (23.85%).

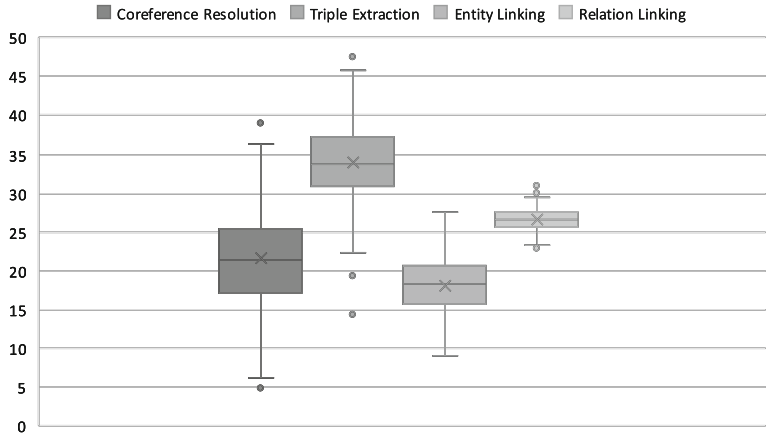


Fig. 4 Box plot of error percentage per IE task. The Y axis shows the error percentage. Each box shows the error percentage by all components, the average error, and some of the outliers. Higher values mean a greater error rate. The figure shows that text triple extraction is the highest impacting component followed by relation linking, coreference resolution, and the least impacting is the entity linking

Further analysis Aiming to understand why IE pipelines perform with low accuracy, we conduct a more in-depth analysis per IE task. In the first analysis, we evaluated each component independently on the WebNLG dataset. Researchers [21, 59] proposed several criteria for micro-benchmarking tools/components for KG tasks (entity linking, relation linking, etc.) based on the linguistic features of a sentence. We motivate our analysis based on the following criteria per task:

Text triple extraction We consider the number of words (*wc*) in the input sentence (a sentence is termed "simple" if it has an average word length of 7.41 [58]. Sentences with higher numbers of words than seven are complex sentences). Furthermore, having a comma in a sentence (subclause) to separate clauses is another factor. Atomic sentences (e.g., *cats have tails*) are a type of sentence that also affects triples extractors' behavior. Moreover, nominal relation as in *Durin, son of Thorin* is another impacting factor on the performance. Uppercase and lowercase mentions of the words (i.e., correct capitalization of the first character and not the entire word) in a sentence are standard errors for entity linking components. We consider this a micro-benchmarking criterion.

Coreference resolution We focus on the length of the coreference chain (i.e., the number of aliases for a single mention). Additionally, the number of clusters is another criterion in the analysis. A cluster refers to the groups of mentions that require disambiguation (e.g., *mother bought a new phone, she is so happy about it* where the first cluster is *mother* $\rightarrow$ *she* and the second is *phone* $\rightarrow$ *it*). The presence of proper nouns in the sentence is studied as well as acronyms. Furthermore, the demonstrative nature of the sentence is also observed as a factor. Demonstrative sentences are the ones that contain demonstrative pronouns (this, that, etc.).

Entity linking The number (e=1,2) of entities in a sentence is a crucial observation for the entity linking task. Capitalization of the surface form is another criterion for micro-benchmarking entity linking tools. An entity is termed as an explicit entity when the entity's surface form in a sentence matches the KG label. An entity is implicit when there is a vocabulary mismatch. For example, in the sentence *The wife of Obama is Michelle Obama.*, the surface form *Obama* is expected to be linked to `dbr:Barack_Obama` and considered

as an implicit entity [59]. The last linguistic feature is the number of words (w) in an entity label (e.g., *The Storm on the Sea of Galilee* has seven words).
Relation linking Similar to the entity linking criteria, we focus on the number of relations in a sentence (rel= 1,2).[8] The type of relation (i.e., explicit or implicit) is another parameter. Covered relation (sentences without a predicate surface form) is also used as a feature for micro-benchmarking: *Which companies have launched a rocket from Cape Canaveral Air Force station?* where the `dbo:manufacturing` relation is not mentioned in the sentence. Covered relations highly depend on common sense knowledge (i.e., reasoning) and the structure of the KG [59]. Lastly, the number of words ($w>=$ N) in a predicate surface form is also considered.

Figure 5 illustrates micro-benchmarking of various PLUMBER components per task. We observe that across IE tasks, the F1 score of the components varies significantly based on the sentence's linguistic features. In fact, there exists no single component which performs equally well on all the micro-benchmarking criteria. This observation further validates our hypothesis to design PLUMBER for building dynamic KG completion pipelines based on the strengths and weaknesses of the integrated components.

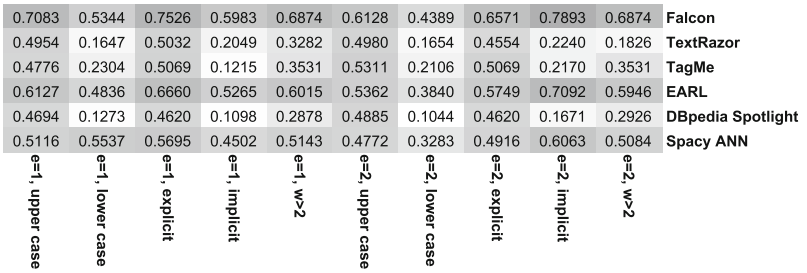
In Fig. 5, all the CR components report limited performance for the demonstrative sentences (*demonstratives*). When there is more than one coreference cluster in a sentence, all other CR components observe a discernible drop in F1 score. The NeuralCoref [15] component performs best for *proper nouns*, whereas PyCobalt [32] performs best for the *acronyms* feature (almost being tied with NeuralCoref). In the TE task, Graphene [11] shows the most stable performance across all categories. However, the performance of all components (except Dependency Parser) drops significantly when the number of words in a sentence exceeds seven ($wc>7$). Case-sensitivity also affects the performance and all components observe a noticeable drop in F1 score for lowercase entity mentions in the sentence. A similar behavior is observed for entity linking components where case-sensitivity is a significant cause of poor performance. When the sentence has one entity and it is implicit (e= 1, *implicit*); all entity linking components face challenges in correctly linking the entities to the underlying KG. Relation linking components also report lower performance for implicit relations.

We then extended micro-benchmarking of the components to Wikidata and reported their performance in isolation. We considered all the sentences present in the T-Rex test set (approx 1.2 M sentences). Figure 6 illustrates the findings per linguistic feature for all IE subtasks. Similarly as for DBpedia, we observe that no single component is superior to all micro-benchmarking criteria. Issues such as capitalization of entity surface forms continue to impact EL and TE components' overall performance negatively. Relation linking components on Wikidata inherit a similar trend as DBpedia components, where the implicit and hidden nature of relation surface forms has the highest impact on their performance.

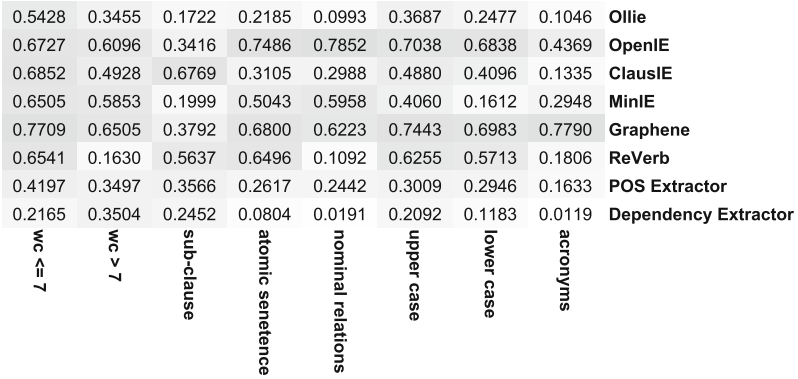
7 Discussion

Even though the dynamic pipelines of PLUMBER outperform static pipelines, the overall performance of PLUMBER and baselines for the KG completion task remains low. Our detailed and exhaustive ablation studies suggest that when individual components are plugged together, their individual performance is a major error source. However, this behavior is expected, considering that earlier research works in other domains observe a similar trend.

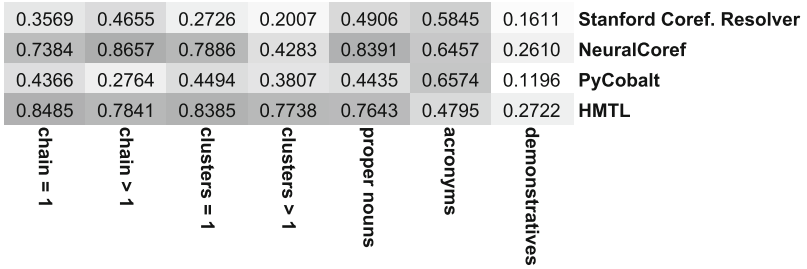
[8] This number of relations in inconsequential for a triple extractor, but it affects relation linkers and their complexity.



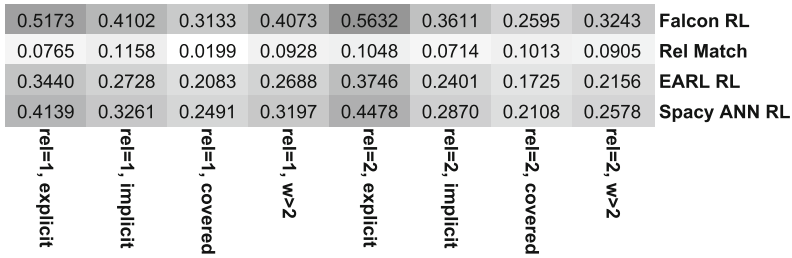
(a) F1 score heatmap of the EL task



(b) F1 score heatmap of the Text TE task

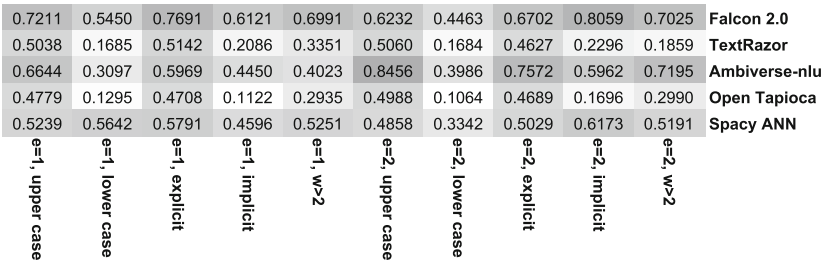


(c) F1 score heatmap of the CR task

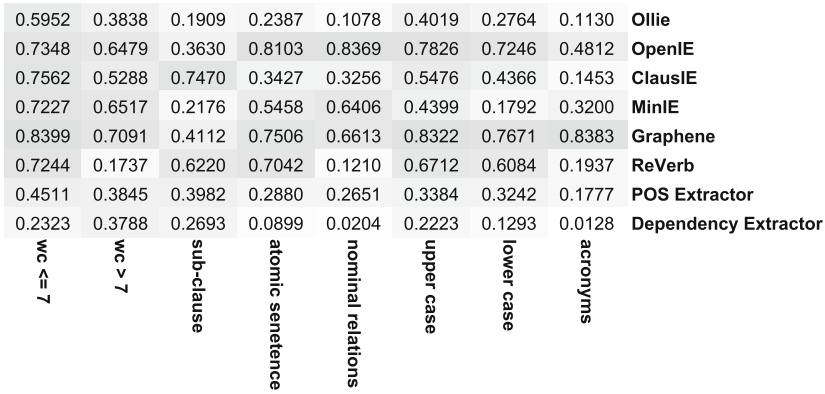


(d) F1 score heatmap of the RL task

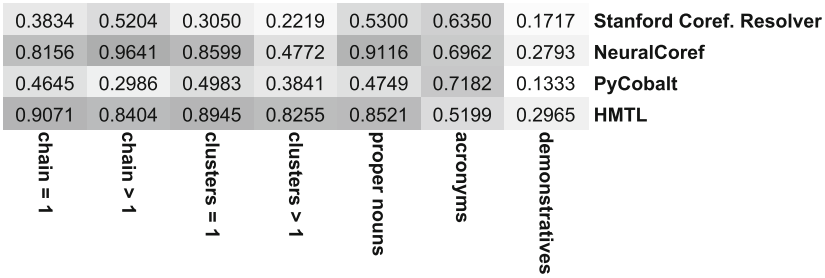
Fig. 5 Comparison of F1 scores per component for different IE tasks based on the various linguistic features of an input sentence (number of entities, word count in a sentence, implicit vs. explicit relation, etc.). Darker shades indicate a higher F1 score



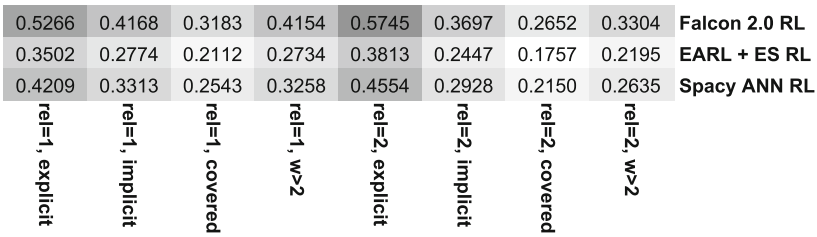
(a) F1 score heatmap of the EL task



(b) F1 score heatmap of the Text TE task



(c) F1 score heatmap of the CR task



(d) F1 score heatmap of the RL task

Fig. 6 Comparison of F1 scores per component for different IE tasks (on the T-Rex dataset). Darker shades indicate a higher F1 score. We observe that components for Wikidata show a similar trend in the performance as in Fig. 5 where test sentences having linguistic features such as implicit entities, word count in a sentence, capitalization of entity surface form, etc. negatively impact performance

Table 7 Average runtime on all datasets in seconds

System	WebNLG	T-Rex	COV-triples
T2KG [41]	2.8	–	–
KnowledgeNet [49]	–	3.4	–
Frankenstein (static) [58]	2.4	2.5	–
Frankenstein (dynamic) [58]	10.1	3.9	2.9
PLUMBER (static)	1.8	1.9	1.2
PLUMBER (dynamic)	12.3	3.9	2.7

The dynamic pipelines on WebNLG report the slowest runtime because few components have a high avg. runtime (up to 65 sec)

For example, since its first release in 2015, the research community performed over 50,000 experiments[9] to improve EL components using the Gerbil framework [62]. Similarly, in 2018 Frankenstein reported the best dynamic question answering pipeline with F1 score 0.20. Within 2 years, the Semantic Web research community had released several components dedicated to solving entity linking and relation linking [26, 50, 54], which were two weaknesses identified by [58] for the QA task. At present, the QA system [43] reuses components from Frankenstein and is a new state of the art on standard complex QA dataset [61] with an F1 score of 0.68. We also calculated the average runtime of PLUMBER and baselines on all three datasets (cf. Table 7). The PLUMBER static pipeline was the fastest; however, the dynamic pipelines on DBpedia were the slowest. The main reason for the slow dynamic pipeline was the high runtime of DBpedia-based relation linking components. For example, Relmatch [57] has an average runtime of 65 s, thus negatively impacting the overall dynamic pipeline runtime. Due to the direct impact of a component's runtime on the overall efficiency (runtime and memory consumption), improving the runtime efficiency of the PLUMBER pipelines was out of scope for this work. However, including Table 7 provides insight into how different baselines and systems perform with various datasets and data domains.

We observe that state-of-the-art components for KG completion still have much potential to improve their performance (both in terms of runtime and F1 score). It is essential to highlight that some of the issues observed in our ablation study basic and have been repeatedly pointed out by researchers in the community. For instance, Derczynski et al. [21] in 2015, followed by Singh et al. [58] in 2018, showed that case-sensitivity is a main challenge for EL tools. Our observations in Figs. 5 and 6 again confirm that case-sensitivity of entity surface forms remains an open issue even for newly released components. In contrast, on specific datasets such as CoNLL-AIDA, several EL approaches reported F1 scores higher than 0.90 [65], showing that EL tools are highly customized to particular datasets. In a real-world scenario like ours, the underlying limitations of approaches are uncovered. We also found that relation linking and text triple extractor components contributed caused significant errors in PLUMBER performance. Based on our findings, we identified the following open challenges for KG completion tasks, which we deem crucial to guide future research:

- The text triple extractor quality is low across KGs. We need to incorporate the KG's underlying schema to guide the triple extraction.
- Case-sensitivity needs to be improved in entity linking approaches. Implicit entities also challenge the entity linking performance. Yang et al. [65] introduced entity descriptions

[9] http://aksw.org/Projects/GERBIL.html.

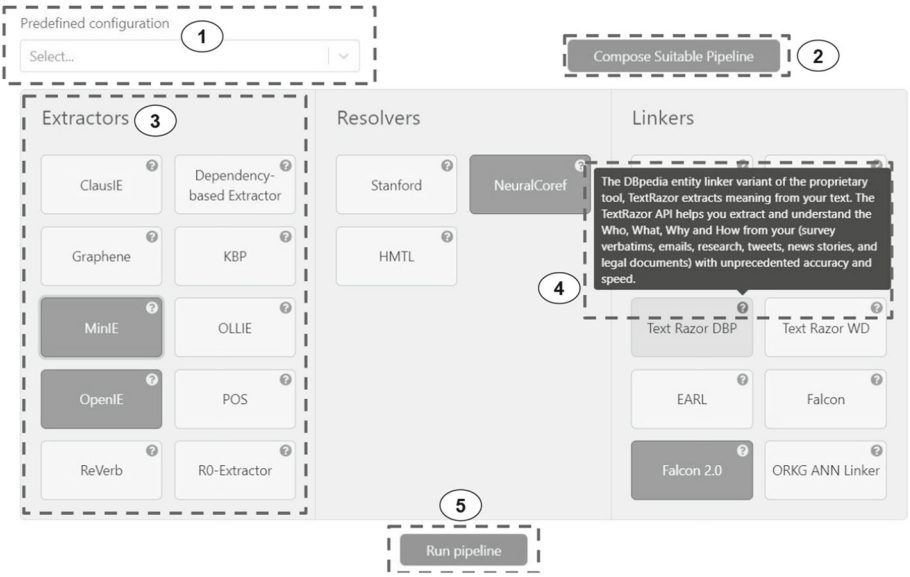


Fig. 7 Overview of the user interface of PLUMBER in the ORKG infrastructure: (1) Predefined pipeline selector: used for easy access to generally stable information extraction pipeline, (2) invoke the framework to create a dynamic pipeline on-the-fly based on the input, (3) collection of IE components that can be used in conjunction manually or automatically, (4) additional information from components to better help the user interact with the system, (5) pipeline runner to display the results and get feedback

as additional context to support implicit entity linking, and we deem such approaches to be beneficial for entity linking tools dedicated for an end-to-end KG completion.
- Relation linking accuracy is limited for the KG completion task across all micro-benchmarking features (cf. Figs. 5, 6). Handling implicit relations and covered relations are the primary source of errors, and we expect that our findings will motivate research to build dedicated relation linking components for KG completion.
- Overall, improving the component's accuracy is the first viable next step for collaborative KG completion.

Furthermore, from a human in the loop approach, we envision employing this framework in conjunction with other knowledge management systems to allow users the possibility to leverage automated extraction from natural language text. With the possibility for user feedback on structured triple level, this information can be used in an active-learning style to improve the underlying pipeline selection model or even propagate these comments and feedback into the individual IE components for further improvement. In this use case, we integrate PLUMBER within the ORKG infrastructure[10] providing an access point to researchers to convert textual descriptions into structured and linked triples.

Figure 7 depicts how the PLUMBER can be integrated within other systems (here the ORKG). Moreover, such an integration allows for user feedback and comments to be fed into the system [39].

[10] Demo video: http://www.youtube.com/watch?v=XC9rJNIUv8g.

8 Conclusion and future work

In this paper, we presented the PLUMBER approach and framework for KG completion. PLUMBER effectively selects the best possible pipeline for a given input sentence using the sentential contextual features and a state-of-the-art transformer-based classification model. PLUMBER has a service-oriented architecture which is scalable, extensible, reusable, and agnostic of the underlying KG. The core idea of PLUMBER is to combine the strengths of already existing disjoint research for KG completion and build a foundation for a platform to promote reusability for the construction of large-scale and semantically structured KGs. Our empirical results suggest that the performance of the individual components directly impacts the end-to-end KG completion accuracy.

This article does not focus on internal system architecture or employed algorithms in a particular IE component to analyze the failures. The focus of the ablation studies is to holistically study the collective success and failure cases for the various tasks. Our studies provide the research community with insightful results over three knowledge graphs, 40 components, 432 pipelines, and test datasets collectively containing over 2.2 M triples extracted from approximately 1.2 M sentences. We release all the experiment results publicly for reproducibility and continued research. Our work is a step in the larger research agenda of offering the research community an effective way for effective to synergistically combine and orchestrate various focused IE approaches balancing their strengths and weaknesses taking different application domains into account, applying their research to a domain driven by many different fields, consequently requiring a collaborative approach to achieve significant progress. We plan to extend our work in the following directions: (1) extending PLUMBER to other KGs such as UMLS [9] and AI-KG [22]. AI-KG also employs a single pipeline comprising community-released components for KG completion on their proposed scholarly KG. (2) adding multilingual components to PLUMBER, considering that existing components focus primarily on English, and (3) creating high performing relation linking components for KG completion.

Acknowledgements We thank anonymous reviewers for their very useful comments and suggestions. This work was co-funded by the European Research Council for the project ScienceGRAPH (Grant agreement ID: 819536) and the TIB Leibniz Information Centre for Science and Technology. We also thank Allard Oelen and Vitalis Wiens for their valuable feedback.

Funding Open Access funding enabled and organized by Projekt DEAL. Open Access funding enabled and organized by Projekt DEAL.

References

1. Alobaid A, Corcho O (2018) Fuzzy semantic labeling of semi-structured numerical datasets. In: Faron Zucker C, Ghidini C, Napoli A, Toussaint Y (eds) Knowledge engineering and knowledge management. Springer, Cham, pp 19–33

2. Anand R, Mehrotra K, Mohan CK, Ranka S (1995) Efficient classification for multiclass problems using modular neural networks. IEEE Trans Neural Netw 6:117–124
3. Angeli G, Johnson Premkumar MJ, Manning CD (2015) Leveraging linguistic structure for open domain information extraction. In: ACL, pp 344–354
4. Auer S, Bizer C, Kobilarov G, Lehmann J, Cyganiak R, Ives Z (2007) Dbpedia: a nucleus for a web of open data. In: The semantic web, pp 722–735
5. Balog K (2018) Entity linking. In: Entity-oriented search, Springer, pp 147–188
6. Bastos A, Nadgeri A, Singh K, Mulang IO, Shekarpour S, Hoffart J, Kaul M (2021) Recon: relation extraction using knowledge graph context in a graph neural network, In: Proceedings of the web conference (WWW), p N/A
7. Berners-Lee T (n.d.) Linked data. https://www.w3.org/DesignIssues/LinkedData.html. Accessed on 10 June 2020
8. Berners-Lee T, Hendler J, Lassila O (2001) The semantic web. Sci Am 284(5):34–43
9. Bodenreider O (2004) The unified medical language system (umls): integrating biomedical terminology. Nucleic Acids Res 32:D267–D270
10. Both A, Diefenbach D, Singh K, Shekarpour S, Cherix D, Lange C (2016) Qanary: a methodology for vocabulary-driven open question answering systems, vol 9678, pp 625–641
11. Cetto M, Niklaus C, Freitas A, Handschuh S (2018) Graphene: semantically-linked propositions in open information extraction. In: Proceedings of the 27th COLING, pp 2300–2311
12. Chaganty AT, Paranjape A, Bolton J et al (n.d.) Stanford at tac kbp 2017: building a trilingual relational knowledge graph
13. CHAI Y, (2020) Evaluation metrics of name entity recognition systems. https://ychai.uk/notes/2018/11/21/NLP/NER/Evaluation-metrics-of-Name-Entity-Recognition-systems/
14. Chen C, You G (1993) Class sensitive neural networks. Neural Parallel Sci Comput 1:93–96
15. Clark K, Manning CD (2016) Deep reinforcement learning for mention-ranking coreference models. In: Proceedings of the 2016 EMNLP, pp 2256–2262
16. Cui W, Liu S, Tan L, Shi C, Song Y, Gao Z, Qu H, Tong X (2011) Textflow: towards better understanding of evolving topics in text. IEEE TVCG 17(12):2412–2421
17. Cui W, Liu S, Wu Z, Wei H (2014) How hierarchical topics evolve in large text corpora. IEEE TVCG 20(12):2281–2290
18. Daiber J, Jakob M, Hokamp C, Mendes PN (2013) Improving efficiency and accuracy in multilingual entity extraction. In: Proceedings of the 9th I-semantics
19. Del Corro L, Gemulla R (2013) Clausie: clause-based open information extraction. In: Proceedings of the 22nd international conference on world wide web, WWW '13, ACM, pp 355–366
20. Delpeuch A (2019) Opentapioca: lightweight entity linking for wikidata
21. Derczynski L, Maynard D, Rizzo G, Van Erp M, Gorrell G, Troncy R, Petrak J, Bontcheva K (2015) Analysis of named entity recognition and linking for tweets. Inf Process Manag 51:32–49
22. Dessi D, Osborne F, Reforgiato Recupero D, Buscaldi D, Motta E, Sack H (2020) Ai-kg: an automatically generated knowledge graph of artificial intelligence. In: International semantic web conference
23. Devlin J, Chang M-W, Lee K, Toutanova K (2019) BERT: pre-training of deep bidirectional transformers for language understanding. In: NAACL, pp 4171–4186
24. Diefenbach D, Giménez-García J, Both A, Singh K, Maret P (2020) Qanswer kg: designing a portable question answering system over rdf data. In: Harth A, Kirrane S, Ngonga Ngomo AC, Paulheim H, Rula A, Gentile AL, Haase P, Cochez M (eds) The semantic web. Springer, Cham, pp 429–445
25. Dong T, Wang Z, Li J, Bauckhage C, Cremers AB (2019) Triple classification using regions and fine-grained entity typing. In: Proceedings of the AAAI conference on artificial intelligence, vol 33, pp 77–85
26. Dubey M, Banerjee D, Chaudhuri D, Lehmann J (2018) EARL: joint entity and relation linking for question answering over knowledge graphs. In: Lecture notes in computer science, Springer, pp 108–126
27. ElSahar H, Vougiouklis P, Remaci A, Gravier C, Hare JS, Laforest F, Simperl E (2018) T-rex: a large scale alignment of natural language with knowledge base triples. In: Proceedings of the eleventh international conference on language resources and evaluation, LREC 2018, Miyazaki, Japan, May 7–12, 2018
28. Fabian M, Gjergji K, Gerhard W et al (2007) Yago: a core of semantic knowledge unifying wordnet and wikipedia. In: WWW, pp 697–706
29. Fader A, Soderland S, Etzioni O (2011) Identifying relations for open information extraction. In: Proceedings of the 2011 EMNLP, pp 1535–1545
30. Ferragina P, Scaiella U (2010) TAGME: on-the-fly annotation of short text fragments (by wikipedia entities), pp 1625–1628
31. Fredrickson S, Tarassenko L (1995) Text-independent speaker recognition using neural network techniques

32. Freitas A, Bermeitinger B, Handschuh S (n.d.) Lambda-3/pycobalt: coreference resolution in python. https://github.com/Lambda-3/PyCobalt
33. Gardent C, Shimorina A, Narayan S, Perez-Beltrachini L (2017) Creating training corpora for NLG micro-planners, pp 179–188
34. Gashteovski K, Gemulla R, del Corro L (2017) MinIE: minimizing facts in open information extraction. In: Proceedings of the 2017 EMNLP, pp 2630–2640
35. Hoffart J, Yosef MA, Bordino I, Fürstenau H, Pinkal M, Spaniol M, Taneva B, Thater S, Weikum G (2011) Robust disambiguation of named entities in text, pp 782–792
36. Hou Y, Jochim C, Gleize M, Bonin F, Ganguly D (2019) Identification of tasks, datasets, evaluation metrics, and numeric scores for scientific leaderboards construction. In: Proceedings of the 57th ACL, pp 5203–5213
37. Ibrahim Y, Riedewald M, Weikum G, Zeinalipour-Yazti D (2019) Bridging quantities in tables and text. In: 2019 IEEE 35th ICDE, pp 1010–1021
38. Jaradeh MY, Oelen A, Farfar KE, Prinz M, D'Souza J, Kismihók G, Stocker M, Auer S (2019) Open research knowledge graph: next generation infrastructure for semantic scholarly knowledge, Marina Del K-CAP, 19
39. Jaradeh MY, Singh K, Stocker M, Auer S (2021) Plumber: a modular framework to create information extraction pipelines, Association for Computing Machinery, New York, pp 678–679. https://doi.org/10.1145/3442442.3458603
40. Jaradeh MY, Singh K, Stocker M, Both A, Auer S (2021) Better call the plumber: orchestrating dynamic information extraction pipelines. In: Brambilla M, Chbeir R, Frasincar F, Manolescu I (eds) Web engineering. Springer, Cham, pp 240–254
41. Kertkeidkachorn N, Ichise R (2017) T2kg: an end-to-end system for creating knowledge graph from unstructured text. In: AAAI workshops, vol WS-17
42. Kim J-D, Unger C, Ngomo A-CN, Freitas A, Hahm Y-g, Kim J, Nam S, Choi G-H, Kim J-u, Usbeck R et al (2017) OKBQA framework for collaboration on developing natural language question answering systems
43. Liang S, Stockinger K, de Farias TM, Anisimova M, Gil M (2020) Querying knowledge graphs in natural language
44. Liu Y, Ott M, Goyal N, Du J, Joshi M, Chen D, Zettlemoyer L, Stoyanov V (2019) Roberta: a robustly optimized bert pretraining approach
45. Liu Y, Zhang T, Liang Z, Ji H, McGuinness D (2018) Seq2rdf: an end-to-end application for deriving triples from natural language text
46. Lu B-L, Ito M (1997) Task decomposition based on class relations: a modular neural network architecture for pattern classification, pp 330–339
47. Malyshev S, Krötzsch M, González L, Gonsior J, Bielefeldt A (n.d.) Getting the most out of wikidata
48. Mausam, Schmitz M, Soderland S, Bart R, Etzioni O (2012) Open language learning for information extraction. In: Proceedings of the 2012 joint conference on empirical methods in natural language processing and computational natural language learning, ACL, pp 523–534
49. Mesquita F, Cannaviccio M, Schmidek J, Mirza P, Barbosa D (2019) KnowledgeNet: a benchmark dataset for knowledge base population, 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, pp 749–758
50. Mihindukulasooriya N, Rossiello G, Kapanipathi P, Abdelaziz I, Ravishankar S, Yu M, Gliozzo A, Roukos S, Gray A (2020) Leveraging semantic parsing for relation linking over knowledge bases, ISWC
51. Niklaus C, Cetto M, Freitas A, Handschuh S (2018) A survey on open information extraction. In: Proceedings of the 27th COLING, pp 3866–3878
52. Ponza M, Del Corro L, Weikum G (2018) Facts that matter. In: Proceedings of the 2018 EMNLP, ACL, pp 1043–1048
53. Raghunathan K, Lee H, Rangarajan S, Chambers N, Surdeanu M, Jurafsky D, Manning C (2010) A multi-pass sieve for coreference resolution. In: EMNLP
54. Sakor A, Onando Mulang I, Singh K, Shekarpour S, Esther Vidal M, Lehmann J, Auer S (2019) Old is gold: linguistic driven approach for entity and relation linking of short text, ACL, pp 2336–2346
55. Sakor A, Singh K, Patel A, Vidal M-E (2020) Falcon 2.0: an entity and relation linking tool over wikidata. In: CIKM
56. Sanh V, Wolf T, Ruder S (2019) A hierarchical multi-task approach for learning embeddings from semantic tasks. Proc AAAI 33:6949–6956
57. Singh K, Mulang IO, Lytra I, Jaradeh MY, Sakor A, Vidal M, Lange C, Auer S (2017) Capturing knowledge in semantically-typed relational patterns to enhance relation linking. In: Proceedings of the knowledge capture conference, K-CAP 2017, Austin, TX, USA, December 4–6, 2017, pp 31:1–31:8

58. Singh K, Radhakrishna AS, Both A, Shekarpour S, Lytra I, Usbeck R, Vyas A, Khikmatullaev A, Punjani D, Lange C, Vidal ME, Lehmann J, Auer S (2018) Why reinvent the wheel: Let's build question answering systems together, WWW '18, pp 1247–1256
59. Singh K, Saleem M, Nadgeri A, Conrads F, Pan JZ, Ngomo A-CN, Lehmann J (2019) Qaldgen: towards microbenchmarking of question answering systems over knowledge graphs. In: ISWC, pp 277–292
60. Skoutas D, Simitsis A (2007) Ontology-based conceptual design of ETL processes for both structured and semi-structured data. Int J Semant Web Inf Syst 3(4):1–24. https://doi.org/10.4018/jswis.2007100101
61. Trivedi P, Maheshwari G, Dubey M, Lehmann J (2017) Lc-quad: a corpus for complex question answering over knowledge graphs. In: ISWC, pp 210–218
62. Usbeck R, Röder M NN et al (2015) Gerbil: general entity annotator benchmarking framework. In: Proceedings of the 24th WWW, pp 1133–1143
63. Vrandečić D, Krötzsch M (2014) Wikidata: a free collaborative knowledgebase. Commun ACM 57(10):78–85
64. Weikum G, Dong L, Razniewski S, Suchanek F (2020) Machine knowledge: creation and curation of comprehensive knowledge bases. arXiv preprint arXiv:2010.10156
65. Yang X, Gu X, Lin S, Tang S, Zhuang Y, Wu F, Chen Z, Hu G, Ren X (2019) Learning dynamic context augmentation for global entity linking. In: EMNLP-IJCNLP, pp 271–281
66. Yao L, Mao C, Luo Y (2019) Kg-bert: bert for knowledge graph completion
67. Yu W, Li Z, Zeng Q, Jiang M (n.d.) Tablepedia: automating pdf table reading in an experimental evidence exploration and analytic system, WWW '19, pp 3615–3619

Mohamad Yaser Jaradeh holds a MSc from Bonn University in Germany and received his PhD from Leibniz University Hannover. He is currently working as a postdoc researcher at the Data Science and Digital Libraries group at TIB in Hannover, Germany. He occupies the roles of a backend developer and a research fellow at his institute. His research interests are in the application of neural natural language processing techniques on free text to extract information for the use within knowledge graphs. Other research interests are in semantic web, gamification, and question answering.

Kuldeep Singh is currently working as a Principle Product Manager. He did his PhD thesis at the University of Bonn, Germany, focusing on answering questions over knowledge graphs. Kuldeep received the prestigious Marie Curie Fellowship from the European Union for his PhD studies. He obtained a double master's degree in Computer Science from the Technical University of Berlin, Germany, and Aalto University, Finland in computer science. He regularly publishes in top conferences such as The Web Conference, CIKM, ECML, EACL, ESWC, SIGIR and ISWC.

Markus Stocker leads the Knowledge Infrastructures research group at the TIB - Leibniz Information Centre for Science and Technology where he co-leads the Open Research Knowledge Graph research and development activities. His research interests lie at the intersection between research infrastructures and research communities, and how such infrastructures acquire, maintain, and share scientific knowledge about human and natural worlds. He got his doctorate in Environmental Sciences (Informatics) of the University of Eastern Finland, Kuopio. He publishes in top venues such as WWW, JCDL, K-CAP, ICWE, and IUI.

Andreas Both studied computer science at the Martin Luther University Halle-Wittenberg (Germany) until 2005 and received his doctorate in the research field of software engineering at the same university in 2010. He held leading positions in research and development units in companies in the fields of e-commerce and business software and was responsible for the area of technology-oriented innovation. Until 2018, he held the role of Head of Architecture, Web Technologies & IT Research at DATEV eG - a top-tier business software provider in Germany. Since 2018, he has been Head of Research at DATEV and a university professor: 2018–2022 at Anhalt University of Applied Sciences (Germany), and since 2022 at the Leipzig University of Applied Sciences (Germany). His research in the field of web and software engineering focuses on data-driven architectures, applied AI, and question answering over knowledge graphs.

Sören Auer is currently a professor of Data Science and Digital Libraries at Leibniz Universität Hannover and Director of the TIB. He has made important contributions to semantic technologies, knowledge engineering and information systems. He is the author (resp. co-author) of over 200 peer-reviewed scientific publications. He has received several awards, including an ERC Consolidator Grant from the European Research Council, a SWSA ten-year award, the ESWC 7-year Best Paper Award, and the OpenCourseware Innovation Award. He has led several large collaborative research projects, such as the EU H2020 flagship project BigDataEurope. He is co-founder of high potential research and community projects such as the Wikipedia semantification project DBpedia, the Open Research Knowledge Graph. He is an expert for industry, European Commission, W3C, the German National Research Data Infrastructure (NFDI) and the European Open Science Cloud (EOSC).