KBPearl: A Knowledge Base Population System Supported by Joint Entity and Relation Linking

Xueling Lin, Haoyang Li, Hao Xin, Zijian Li, Lei Chen
Department of Computer Science and Engineering
The Hong Kong University of Science and Technology, Hong Kong, China
{xlinai, hlicg, hxinaa, zlicb, leichen}@cse.ust.hk

ABSTRACT

Nowadays, most openly available knowledge bases (KBs) are incomplete, since they are not synchronized with the emerging facts happening in the real world. Therefore, knowledge base population (KBP) from external data sources, which extracts knowledge from unstructured text to populate KBs, becomes a vital task. Recent research proposes two types of solutions that partially address this problem, but the performance of these solutions is limited. The first solution, dynamic KB construction from unstructured text, requires specifications of which predicates are of interest to the KB, which needs preliminary setups and is not suitable for an in-time population scenario. The second solution, Open Information Extraction (Open IE) from unstructured text, has limitations in producing facts that can be directly linked to the target KB without redundancy and ambiguity. In this paper, we present an end-to-end system, KBPearl, for KBP, which takes an incomplete KB and a large corpus of text as input, to (1) organize the noisy extraction from Open IE into canonicalized facts; and (2) populate the KB by joint entity and relation linking, utilizing the context knowledge of the facts and the side information inferred from the source text. We demonstrate the effectiveness and efficiency of KBPearl against the state-of-the-art techniques, through extensive experiments on real-world datasets.

PVLDB Reference Format:
Xueling Lin, Haoyang Li, Hao Xin, Zijian Li and Lei Chen. KBPearl: A Knowledge Base Population System Supported by Joint Entity and Relation Linking. *PVLDB*, 13(7): 1035-1049, 2020.
DOI: https://doi.org/10.14778/3384345.3384352

1. INTRODUCTION

With the recent development of information extraction techniques, numerous large-scale knowledge bases (KBs), such as Wikidata [63] and DBpedia [3], have been constructed. In these KBs, ontological knowledge is stored in the form of (`subject, predicate, object`) triples, representing millions of real-world semantic concepts, including entities, relations, specified types, and categories. These KBs provide back-end support for various knowledge-centric services of real-world applications, such as question answering [68] and online recommendations [7].

However, most KBs are not complete [44, 46], since they have limited coverage of what happens in the real world and are not synchronized with the emerging entities and their relations. Rerunning the KB construction process to keep these KBs up-to-date with real-time knowledge is too expensive, since most KBs are built in a batch-oriented manner. Therefore, knowledge base population (KBP) from external data sources has been proposed [25, 29]. Specifically, this is the task that takes an incomplete KB and a large corpus of text as input, then extracts knowledge from the text corpus to complete the incomplete elements in the KB.

Currently, there are two major types of research works that address the KBP issue. The first type of works, such as DeepDive [12, 45, 57] and SystemT [10], aim for a dynamic and broader construction of KBs. However, these prior works require a specification that which predicates are of interest to the KBP task. In other words, unless predicates such as *birthplace* have been specified preliminarily by the systems, the knowledge regarding these predicates will never be discovered automatically. This limits the ability of these systems to extract knowledge from unstructured text for KBP without any preliminary setups.

The second type of works, Open Information Extraction (Open IE) [13, 47, 53], partially addresses the task of KBP. The approaches based on Open IE allow the subjects and objects in the triples to be arbitrary noun phrases extracted from external sources, while predicates will be arbitrary verb phrases. Such Open IE approaches not only improve the efficiency of KBP in an unsupervised manner, but also recognize entities and predicates that are not recorded in the current KBs. Nevertheless, there are two major problems that these Open IE approaches suffer from:

- *The canonicalization problem*: the triples extracted by Open IE tools (referred to as *Open triples*) that represent the same semantic meaning are not grouped. For instance, the Open triples (`Michael Jordan`, `was born in`, `California`), (`Professor Jordan`, `has birthplace`, `CA`), and (`M. Jordan`, `was born in`, `CA`) will all be presented even if they represent an equivalent statement, which leads to *redundancy and ambiguity*. Recently, some techniques [21,32,61,64] have been proposed to canonicalize Open triples, by clustering those with the same semantic meaning in one group. However, such approaches have high overheads and are not geared up for in-time KBP.

Proceedings of the VLDB Endowment, Vol. 13, No. 7
ISSN 2150-8097.
DOI: https://doi.org/10.14778/3384345.3384352

- *The linking problem*: the knowledge in the Open triples is not mapped to the ontological structure of the current KBs. In other words, Open IE techniques only extract (`Michael Jordan, was born in, California`), without indicating which entity the noun phrase `Michael Jordan` refers to. Recent techniques [16, 52] have been proposed to address both entity linking and relation linking tasks in *short text*, by linking the noun phrases (resp., relation phrases) detected in the text to the entities (resp., predicates) in the KB. However, there are still three drawbacks. First, if a noun phrase cannot be linked to any existing entity in the KB, then the knowledge regarding this noun phrase cannot be populated to the KB. Second, most of the existing works disambiguate each phrase in a document *separately*. Nevertheless, it is beneficial to consider entity and predicate candidates for the input document in combination (which we call "*global coherence*"), to maximize the usable evidence for the candidate selection process. For example, if a noun phrase `artificial intelligence` refers to *AI (branch of computer science)* rather than *A.I. (movie)*, then we would expect the noun phrase `Michael Jordan` in the same document to be *M. Jordan (computer scientist and professor)* rather than *M. Jordan (basketball player)*. Third, none of the existing works utilizes such global coherence for joint entity and relation linking in *long-text documents*.

In this paper, we propose *KBPearl* (KB Population supported by joint entity and relation linking), a novel end-to-end system which takes an incomplete KB and a set of documents as input, to populate the KB by utilizing the knowledge extracted and inferred from the source documents. Initially, for each document, KBPearl conducts knowledge extraction on the document to retrieve *canonicalized facts* and *side information*. Specifically, the side information includes (1) all the named entity mentions in the document; (2) the types of these named entity mentions; and (3) the timestamps in the document. After that, to capture the global coherence of concepts in the documents, KBPearl conducts KBP in the following steps. First, KBPearl constructs a *semantic graph* based on the knowledge extracted from the canonicalized facts and the side information. Second, KBPearl disambiguates the noun phrases and relation phrases in the facts, by determining a dense subgraph from the semantic graph in an efficient and effective manner. Finally, KBPearl links the canonicalized facts to the KB and creates new entities for non-linkable noun phrases.

Specifically, compared to the mainstream dynamic KB construction, we acquire facts for non-predefined predicates. Compared to the Open IE methods, we canonicalize the arguments of facts and link them to the target KB. Compared to the current techniques aiming for joint entity and relation linking, we conduct the joint linking task on long-text documents efficiently. To the best of our knowledge, this is the first work that conducts joint entity and relation linking and reports new entities for KBP leveraging the knowledge inferred from long-text documents. We summarize the novel contributions of this paper as follows.

- We present an end-to-end system KBPearl to populate KB based on natural-language source documents with the support of Open IE techniques.
- We employ a *semantic graph-based approach* to represent the facts and side information in the source documents. In this way, we capture the concepts mentioned in the documents with *global coherence*. Specifically, we propose a novel similarity design schema to formulate the similarity between entity (resp., predicate) pairs based on their shared *keyphrases* stored in the target KB.
- We perform a joint entity and relation linking task on the semantic graph, aiming to disambiguate the entities and predicates *jointly* for KBP. Specifically, we formulate this task as a dense subgraph detection problem. We prove the NP-hardness of finding such a dense subgraph and propose a greedy graph densification algorithm as solution. We also present two variants to employ the graph densification algorithm in the KBPearl system in an effective and efficient manner. Moreover, we locate the noun phrases representing new semantic concepts which do not exist in the target KB, and create new entities for them.
- We conduct extensive experiments to demonstrate the viability of our system on several real-world datasets. Experimental results prove the effectiveness and efficiency of KBPearl, compared with the state-of-the-art techniques.

The rest of the paper is organized as follows. We introduce the important definitions in Section 2, and present the system overview in Section 3. We illustrate the construction of the semantic graph in Section 4. We formulate the dense subgraph problem, and present the graph densification algorithm to perform joint entity and relation linking in Section 5. Section 6 presents our experimental results. We discuss the related works in Section 7 and conclude our paper in Section 8.

2. PROBLEM DEFINITION

We first introduce the important definitions. We then define the problem that our system addresses in Problem 1.

DEFINITION 1. ***Knowledge Base (KB).*** *A KB Ψ can be considered as a set of facts, where each fact is stored in the form of a triple (s, p, o), representing the subject, predicate, and object of a fact. In the KB Ψ, we denote the set of entities as $\mathcal{E}_\Psi$, the set of predicates as $\mathcal{P}_\Psi$, and the collection of literals as L_Ψ. For a triple (s, p, o), we have $s \in \mathcal{E}_\Psi$, $p \in \mathcal{P}_\Psi$ and $o \in \mathcal{E}_\Psi \cup L_\Psi$.*

DEFINITION 2. ***Open Triples.*** *The Open triples are extracted by the Open IE techniques from a given document. An Open triple is denoted as $t = (n^{sub}, r, n^{obj})$, where the noun phrase n^{sub} is the subject, r stands for the relation, and noun phrase n^{obj} represents the object.*

DEFINITION 3. ***Side Information.*** *The side information of a given document d, denoted by $S(d)$, includes (1) all the named entity mentions detected in document d; (2) the types of these named entity mentions; and (3) the timestamp information in the document d.*

PROBLEM 1. ***KB Population From External Unstructured Text.*** *Given an incomplete KB Ψ and a document d, our task is to (1) extract the side information $S(d)$ and a set of canonicalized Open triples $T = \{t_1, t_2, ...\}$ from the document d; (2) link each triple $t_i = (n^{sub}, r, n^{obj})$ in T to the KB Ψ in order to populate the knowledge in Ψ based on the side information $S(d)$. Specifically, in each $t_i \in T$, we decide whether n^{sub} is linked to an entity $e \in \mathcal{E}_\Psi$, r is linked to a predicate $p \in \mathcal{P}_\Psi$, and n^{obj} is linked to an entity $e' \in \mathcal{E}_\Psi$ or referred to as a literal $l \in \mathcal{L}_\Psi$.*

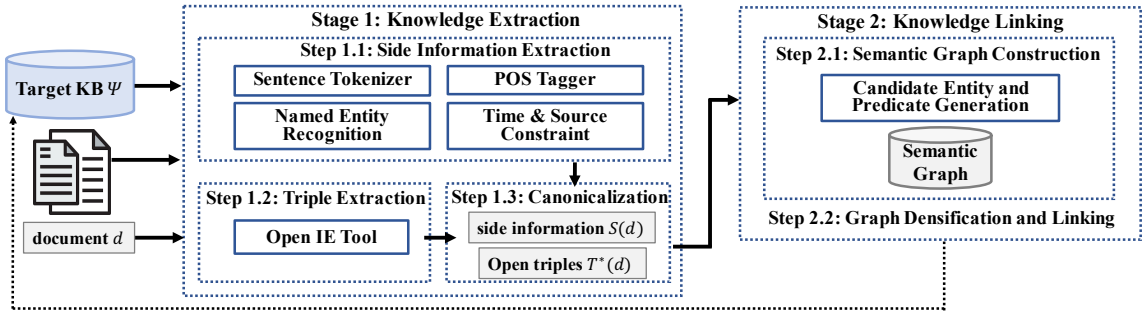


Figure 1: Framework overview.

Note that for a triple $t_i = (n^{sub}, r, n^{obj})$, there are two possible situations:

- The first situation is that n^{sub} is linked to an entity $e \in \mathcal{E}_\Psi$, r is linked to a predicate $p \in \mathcal{P}_\Psi$, and n^{obj} is linked to an entity $e' \in \mathcal{E}_\Psi$ or referred to as a literal $l \in \mathcal{L}_\Psi$. If (e, p, e') or (e, p, l) does not exist in the KB Ψ, we add it as a new fact to KB Ψ;
- The second situation is that at least one of n^{sub} and n^{obj} is determined to be non-linkable to the KB Ψ. If n^{sub} is non-linkable, we report a new entity for it. If n^{obj} is non-linkable and it is detected as a named entity mention in $S(d)$, we also report a new entity for it. Finally, we add the new fact to the KB Ψ as well.

3. SYSTEM FRAMEWORK OVERVIEW

Figure 1 depicts the framework overview of our system. Given a KB Ψ and an input document d, our framework works in two major stages.

3.1 Stage 1: Knowledge Extraction

In this stage, we conduct pre-processing and knowledge extraction from the input document d.

Step 1.1: Side Information Extraction. The document d is pre-processed by a pipeline of linguistic tools, including (1) sentence tokenization, (2) part-of-speech (POS) tagging, (3) noun-phrase chunking and named entity recognition (NER) with entity typing, and (4) Time Tagger tools which extract time information from d. The output of this step is a set of side information $S(d)$ of document d (defined in Definition 3).

Step 1.2: Triple Extraction Supported by Open IE. In this step, we employ an existing Open IE tool to extract the triples in the form of $t = (n^{sub}, r, n^{obj})$. Specifically, we choose several popular Open IE tools in the experiments to evaluate the performance of our system (the details are shown in Section 6). The output of this step is a set of Open triples $T(d) = \{t_1, t_2, ...\}$.

Step 1.3: Triple Canonicalization and Noise Reduction. Step 1.2 produces a large set of candidate Open triples, which may still contain noise and redundancy. In this step, to reconcile the semantic redundancy and group the triples with the same semantic meaning, we perform triple canonicalization in three major tasks. First, we preprocess the triples, by filtering those that only contain pronouns, stopwords, and interrogative words as the subjects and objects. Second, we conduct basic canonicalization on the triples based on their noun phrases, by utilizing the side information $S(d)$ obtained in Step 1.1. Specifically, we have two assumptions: (1) One particular surname or first name in one article refers to the same `PERSON`-type entity. For example, `Jordan` in an article refers to `Michael Jordan`, which appears in the former part of the same article; (2) The pronouns in one sentence refer to the first subject of the triple extracted from the former sentences for co-reference [5]. Third, following QKBfly [44], we employ a pattern dictionary [43] to conduct basic canonicalization on the triples based on their relation phrases. The output of this step is a set of canonicalized Open triples $T^\star(d)$.

3.2 Stage 2: Knowledge Linking

In this stage, we conduct knowledge linking from the output of Stage 1 to the target KB.

Step 2.1: Semantic Graph Construction. For each document d, we build a semantic graph based on the side information $S(d)$ (produced in Step 1.1) and the extracted canonicalized Open triples $T^\star(d)$ (produced in Step 1.3). The semantic graph captures all the noun phrases (resp., relation phrases) mentioned in document d, as well as all the candidate entities (resp., predicates) to each noun phrase (resp., relation phrase) (see Section 4).

Step 2.2: Graph Densification and Linking. In this step, given the graph constructed in Step 2.1, our task is to wisely select the optimal entity for each noun phrase, and the optimal predicate for each relation phrase. The optimal linking result is expected to represent global coherence among these entities and predicates. We formulate the task of joint entity and relation linking as an efficient graph densification task (See Section 5).

4. SEMANTIC GRAPH CONSTRUCTION

Given a set of canonicalized Open triples $T^\star(d)$ extracted from natural-language document d (produced in Step 1.3), and a set of side information $S(d)$ extracted from d (produced in Step 1.1), the KBPearl system builds a weighted undirected semantic graph to represent all these knowledge in an effective manner. An example of a semantic graph is demonstrated in Figure 2.

We refer to the semantic graph built in this section as $G = (V, E)$. Specifically, V denotes the set of vertices (i.e., nodes), as introduced in Section 4.1, while E denotes the set of edges among the nodes in G, as presented in Section 4.2.

4.1 Graph Nodes

A node in the semantic graph is a container for the following semantic knowledge extracted from the document. In the semantic graph $G = (V, E)$, there are four different types of nodes in V, i.e., $V = \mathcal{N} \cup \mathcal{E} \cup \mathcal{R} \cup \mathcal{P}$.

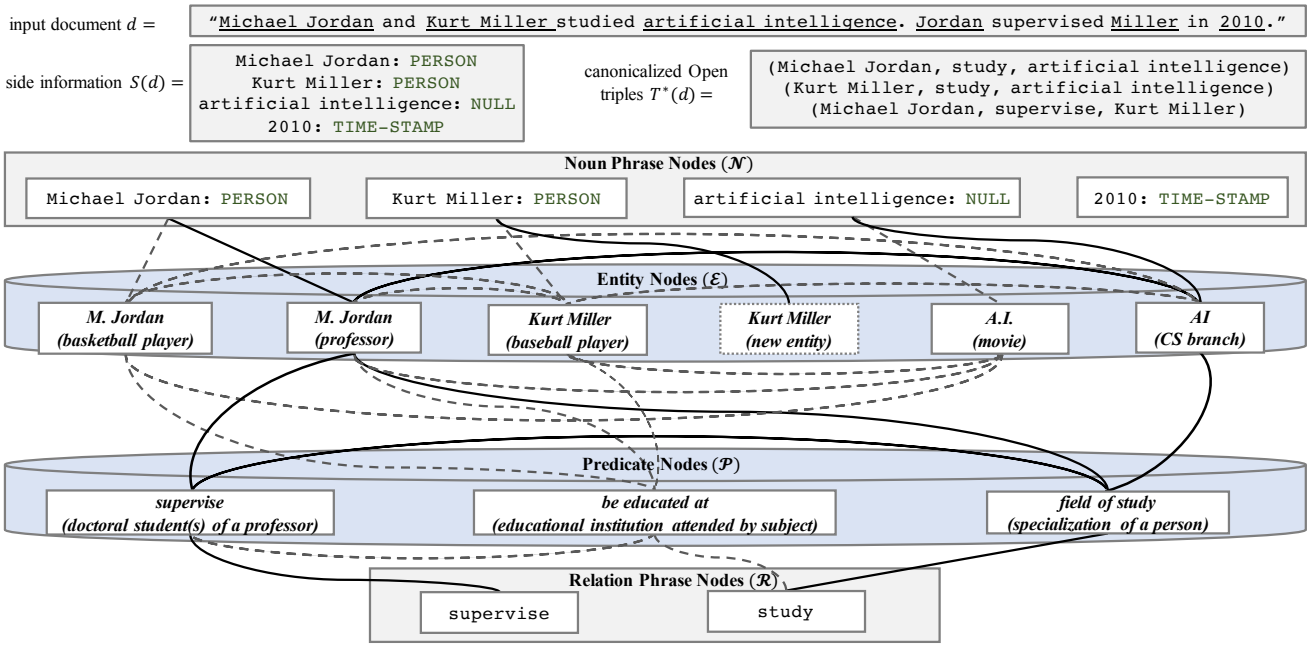


Figure 2: Example of a semantic graph. The solid edges are the potential edges that comprise the dense subgraph to capture the best coherence among the noun phrase nodes, entity nodes, relation phrase nodes, and predicate nodes. "*Kurt Miller*" is recognized as a new entity.

Noun Phrase Nodes ($\mathcal{N}$). A noun phrase node $n \in \mathcal{N}$ represents a subject or object noun phrase in an Open triple $t \in T^{\star}(d)$, or a noun phrase in the side information $S(d)$. Specifically, each noun phrase is assigned with a candidate type from $S(d)$. For example, in Figure 2, the noun phrase node `Michael Jordan` is typed as `PERSON`. Moreover, a noun phrase without type information in $S(d)$ is assigned with the type `NULL`.

Entity Nodes ($\mathcal{E}$). An entity node $e \in \mathcal{E}$ denotes an existing entity in the target KB Ψ. Note that $\mathcal{E} \subset \mathcal{E}_{\Psi}$. Specifically, for each noun phrase node n, we generate a set of candidate entity nodes that (1) have n as one of their aliases, and (2) match the types of noun phrases provided by the side information. For example, for the noun phrase node `Michael Jordan` typed as `PERSON` in Figure 2, we involve all the entities that have `Michael Jordan` as one of their aliases and typed as `PERSON` in Ψ as the candidate entity nodes, including *M. Jordan (basketball player)* and *M. Jordan (professor)*. For a noun phrase node typed as `NULL`, we involve all the entities with this noun phrase as their alias name as the candidate entity nodes, without any type constraint.

Relation Phrase Nodes ($\mathcal{R}$). A relation phrase node $r \in \mathcal{R}$ represents a relation phrase in an Open triple $t \in T^{\star}(d)$. An example is the relation phrase node `study` in Figure 2.

Predicate Nodes ($\mathcal{P}$). A predicate node $p \in \mathcal{P}$ denotes an existing predicate in the target KB Ψ. Note that $\mathcal{P} \subset \mathcal{P}_{\Psi}$. Specifically, for each relation phrase node p, we generate a set of predicate nodes that have p as one of their aliases. For example, for the relation phrase node `study` in Figure 2, we involve all the predicates that have `study` as the alias names in Ψ, including *field of study* and *be educated at*.

4.2 Graph Edges

An edge between two nodes in the semantic graph represents the *similarity measure* between the two nodes. In the semantic graph $G = (V, E)$, there are five different types of edges in E, where the details are listed as follows.

Noun Phrase-Entity Edges. Let $\phi_e(n_i, e_j)$ denote the weight of the noun phrase-entity edge between a noun phrase node $n_i \in \mathcal{N}$ and an entity node $e_j \in \mathcal{E}$. Specifically, $\phi_e(n_i, e_j)$ reflects the likelihood that entity e_j is the correct linking for the noun phrase n_i.

Following the recent entity disambiguation approaches [28, 41, 44], we compute $\phi_e(n_i, e_j) = pop(n_i, e_j)$, which is the relative frequency under which a query of n_i points to e_j in the target KB Ψ. For example, `Michael Jordan` refers to *M. Jordan (basketball player)* in 75% of all its occurrences, while only 15% to *M. Jordan (professor)*. Specifically, $pop(n_i, e_j)$ can be easily obtained as the number of sitelinks of e_j regarding n_i in KB Ψ as the popularity of e_j. Note that the textual similarity between n_i and e_j is not considered, since we only choose the entities that share the same alias name with n_i as its candidate entities (see Section 4.1).

Relation Phrase-Predicate Edges. Let $\phi_p(r_i, p_j)$ denote the weight of the relation phrase-predicate edge between a relation phrase node $r_i \in \mathcal{R}$ and a predicate node $p_j \in \mathcal{P}$. Specifically, $\phi_p(r_i, p_j)$ reflects the likelihood that predicate p_j is the true linking for the relation phrase r_i.

We collect the synonymous phrases of the relation phrase r_i from pattern repositories such as PATTY [43], denoted as $\mathbb{P}(r_i)$. Moreover, we harness the target KB Ψ for labels and aliases of the predicate p_j, denoted as $\mathbb{P}_{\Psi}(p_j)$. We formulate $\phi_p(r_i, p_j)$ as the *overlap coefficient* [62] between $\mathbb{P}(r_i)$ and $\mathbb{P}_{\Psi}(p_j)$, as shown in Equation 1.

$$\phi_p(r_i, p_j) = \frac{|\mathbb{P}(r_i) \cap \mathbb{P}_{\Psi}(p_j)|}{min(|\mathbb{P}(r_i)|, |\mathbb{P}_{\Psi}(p_j)|)} \quad (1)$$

Entity-Entity Edges. Let $\rho(e_i, e_j)$ denote the weight of the entity-entity edge between a pair of entity nodes $e_i, e_j \in \mathcal{E}$. Specifically, $\rho(e_i, e_j)$ represents the pairwise relatedness

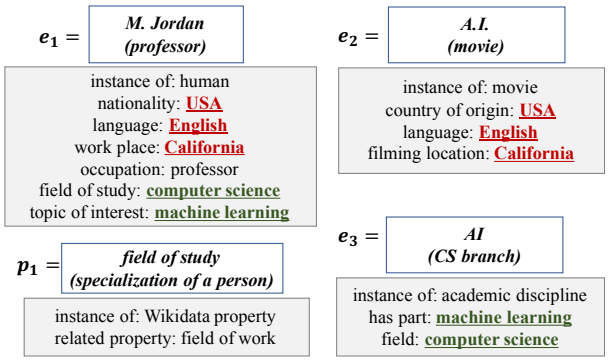


Figure 3: Keyphrases in entities and predicates.

between e_i and e_j. An entity-entity edge only exists between a pair of entity nodes connected to *different* noun phrase nodes. The reason is that under the global coherence assumption, to select the optimal entity for a noun phrase, we only consider whether it is closely related to the candidate entities of *other* noun phrases in the same document. For example, in Figure 3, there is no edge between *M. Jordan (basketball player)* and *M. Jordan (professor)*, since they are connected to the same noun phrase node `Michael Jordan`.

We propose a novel solution to formulate the pairwise relatedness between entities e_i and e_j in a target KB Ψ, by utilizing their *keyphrases* extracted from Ψ. We first formally define the keyphrases for an entity e in a KB Ψ.

DEFINITION 4. ***Keyphrases of Entities.*** *The keyphrases of an entity e in the KB Ψ, denoted as $\mathcal{K}(e)$, is the set of predicate-object pairs of e stored in Ψ. Particularly, suppose that there are m facts stored in Ψ where all of them have entity e as subjects, i.e, $\{(e, p_1, o_1), (e, p_2, o_2), ..., (e, p_m, o_m)\}$, we have $\mathcal{K}(e) = \{(p_1, o_1), (p_2, o_2), ..., (p_m, o_m)\}$. Furthermore, we denote $\mathcal{K}_p(e) = \{p_1, p_2, ..., p_m\}$ as the predicate-keyphrases of entity e, and $\mathcal{K}_o(e) = \{o_1, o_2, ..., o_m\}$ as the object-keyphrases of entity e.*

For instance, in Figure 3, $\mathcal{K}(e_3) = \{$*(instance of, academic discipline), (has part, machine learning), (field, computer science)*$\}$, while $\mathcal{K}_p(e_3) = \{$*instance of, has part, field*$\}$.

To measure the relatedness of entity pairs, we first assign weights to the predicate-keyphrases and object-keyphrases of the entities based on the *Inverse Document Frequency (IDF)*, which measures how important each keyphrase is in a global view. The keyphrases with higher IDF weights will be considered as more uncommon and important. For example, in Figure 3, although e_1 shares more object-keyphrases with e_2 (*USA*, *English*, and *California*) than e_3 (*machine learning* and *computer science*), e_1 is more related to e_3 since the object-keyphrases they share are more uncommon.

Specifically, for a predicate-keyphrase p_m, we compute its weight as $\omega(p_m) = log_2 \frac{|\mathcal{E}_\Psi|}{|\mathcal{E}_{p_m}|}$, where $\mathcal{E}_{p_m}$ denotes the set of entities that contain predicate p_m in its keyphrases, while $|\mathcal{E}_\Psi|$ is the total number of entities in the target KB Ψ. Similarly, for an object-keyphrase o_m, we have $\omega(o_m) = log_2 \frac{|\mathcal{E}_\Psi|}{|\mathcal{E}_{o_m}|}$. Let (p_m, o_m) denote a keyphrase tuple, we define the pairwise relatedness between entity e_i and e_j as follows.

$$\rho(e_i, e_j) = \frac{\sum_{o_m \in \mathcal{K}_o(e_i) \cap \mathcal{K}_o(e_j)} \omega(p_m) \cdot \omega(o_m)}{\sum_{o_m \in \mathcal{K}_o(e_i) \cap \mathcal{K}_o(e_j)} \omega(p_m)} \quad (2)$$

Predicate-Predicate Edges. We use $\rho(p_i, p_j)$ to denote the weight of the predicate-predicate edge between a pair of predicate nodes $p_i, p_j \in \mathcal{P}$. Specifically, $\rho(p_i, p_j)$ represents the pairwise relatedness between p_i and p_j. Note that a predicate-predicate edge only exists between a pair of predicate nodes connected to *different* relation phrase nodes.

Similar to the entities, the keyphrases of a predicate p in the target KB is denoted as $\mathcal{K}(p)$, which is a set of predicate-object pairs that describe p in the KB. Take Figure 3 as an example, where we have $\mathcal{K}(p_1) = \{$*(instance of, Wikidata property), (related property, field of work)*$\}$.

Let (p_m, o_m) denote a keyphrase tuple, we define $\rho(p_i, p_j)$ as the pairwise relatedness between p_i and p_j as follows.

$$\rho(p_i, p_j) = \frac{\sum_{o_m \in \mathcal{K}_o(p_i) \cap \mathcal{K}_o(p_j)} \omega(p_m) \cdot \omega(o_m)}{\sum_{o_m \in \mathcal{K}_o(p_i) \cap \mathcal{K}_o(p_j)} \omega(p_m)} \quad (3)$$

Entity-Predicate Edges. Let $\rho(e_i, p_j)$ denote the weight of the entity-predicate edge between an entity node $e_i \in \mathcal{E}$ and a predicate node $p_j \in \mathcal{P}$. Specifically, $\rho(e_i, p_j)$ represents the pairwise relatedness between e_i and p_j.

We determine that $\rho(e_i, p_j) = \omega(p_j)$ if both conditions are satisfied: (1) the noun phrase node n_i linked to e_i and the relation phrase node r_i linked to p_i are in the same triple $t \in T^*(d)$; and (2) a fact (e_i, p_j, x) exists in the KB Ψ and $x \neq NULL$ (x can be either an entity or a literal). Otherwise, $\rho(e_i, p_j) = 0$. Specifically, we use $\omega(p_j)$ as the value of $\rho(e_i, p_j)$, because we need to avoid the cases where popular predicates are linked to too many entities and hence dominate the other predicates.

5. GRAPH DENSIFICATION & LINKING

In this section, our task is to wisely conduct the joint entity and relation disambiguation and linking. Formally, given a semantic graph $G = (V, E)$, where $V = \mathcal{N} \cup \mathcal{E} \cup \mathcal{R} \cup \mathcal{P}$, we select one optimal entity node $e \in \mathcal{E}$ for each noun phrase node $n \in \mathcal{N}$, and one optimal predicate node $p \in \mathcal{P}$ for each relation phrase node $r \in \mathcal{R}$ (as shown in Figure 2).

Specifically, the optimal linking result is expected to represent global coherence among these entities and predicates, where each entity and predicate is related with at least one of other entities and predicates in the result. Following the community-search problem [59], we seek to transform this candidate selection problem into a dense subgraph problem. In this way, the subgraph that contains the optimal candidate entities and predicates is *densely connected*.

We discuss the definition of subgraph density and the definition of our dense subgraph problem in Section 5.1. We propose a greedy algorithm to address the dense subgraph problem in Section 5.2. In Section 5.3, we deploy the greedy algorithm on the KBPearl system effectively and efficiently.

5.1 The Dense Subgraph Problem

5.1.1 Definition of Subgraph Density

In this step, we discuss the best definition of subgraph density, so that the global coherence of the candidate entities and predicates is captured.

First, we determine the formulation of the node degrees in the semantic graph. To capture global coherence among the entity nodes and predicate nodes in the desired subgraph, we need to examine whether a given node in the graph has a strong connection with *the other nodes*. Hence, we define

the *weighted degree* of a node v in the graph to be the total weight of its incident edges. Formally, in the semantic graph $G = (V, E)$, let $\epsilon(v)$ denote the set of incident edges of the node v, and $w(\mu)$ represent the weight of an edge $\mu \in E$, the weighted degree of v is defined as: $w(v) = \sum_{\mu \in \epsilon(v)} w(\mu)$.

Second, we determine how to measure the density of the desired subgraph H. There are two methods for density measurement in terms of the weighted degrees of its nodes.

The first method regards the *average degree of the nodes* in $H = (V_H, E_H)$ as its density, i.e., $d(H) = \frac{\sum_{v \in V_H} w(v)}{|V_H|}$, as the density of H. This measurement has been extensively studied in the dense subgraph area [1, 59]. However, one drawback of this method is that an unrelated but densely connected community can be easily added to the graph to increase the average density. In other words, in our semantic graph, an entity node (or predicate node) that share heavy weighted edges with half of the other nodes can easily dominate an entity node (or predicate node) that connects with every other node. Take the entity node *M. Jordan (basketball player)* in Figure 2 as an example. The total weight of this node will be high, just because it has rather high popularity and has a heavy edge linked to the noun phrase `Michael Jordan`, but not because of its dense connection to other entity nodes. Therefore, popular nodes such as *M. Jordan (basketball player)* may easily dominate other nodes. Hence, this measurement may cause failure to select a group of nodes where global coherence is expected to be achieved.

The second method, which is used in this paper, is to regard the *minimum degree of the nodes* in H as the density of H, i.e., $d(H) = \min_{v \in V_H} w(v)$. This measurement does not suffer from the failure cases that some popular nodes dominate the others and form a non-related dense community. Maximizing the minimum degree of the nodes in H can guarantee the *global coherence goal* such that the coherence among all the entities and predicates are taken into account.

5.1.2 Definition of the Dense Subgraph Problem

Our goal is to compute a subgraph with maximum density, while observing constraints on the subgraph structure.

PROBLEM 2. ***The Dense Subgraph Problem.*** *Given an undirected connected weighted semantic graph $G = (V, E)$, where $V = \mathcal{N} \cup \mathcal{E} \cup \mathcal{R} \cup \mathcal{P}$, our target is to find an induced subgraph $H = (V_H, E_H)$ of G, where $V_H = \mathcal{N} \cup \mathcal{E}_H \cup \mathcal{R} \cup \mathcal{P}_H$. Specifically, H satisfies all the following constraints:*

- *H is connected;*
- *H contains every noun phrase node and every relation phrase node, i.e., $\mathcal{N} \subseteq V_H$ and $\mathcal{R} \subseteq V_H$;*
- *Every noun phrase node in H is connected to one entity node, and every relation phrase node in H is connected to one predicate node. A noun phrase node and a relation phrase node cannot be linked to each other directly. Moreover, an entity node can be connected to more than one noun phrase node, and a predicate node can be connected to more than one relation phrase node. Therefore, H has at most $2 \cdot (|\mathcal{N}| + |\mathcal{R}|)$ nodes, i.e., $|V_H| \leq 2 \cdot (|\mathcal{N}| + |\mathcal{R}|)$;*
- *The minimum degree of H, i.e., $\min_{v \in V_H} w(v)$, is maximized among all feasible choices for H.*

THEOREM 1. *Problem 2 is NP-hard.*

PROOF. To prove that Problem 2 is NP-hard, we give a unite-weight instance of Problem 2, as well as a simple reduction of this instance to the Steiner tree problem with unit weights. It is known that the Steiner tree problem is NP-hard [31] [6](Theorem 20.2). In the decision version of the Steiner tree problem with unit weights, given a graph $G = (V, E)$, a set of nodes $\mathcal{T} \subseteq V$(usually referred to as *terminals*), and an integer k, we need to determine whether there is a subtree of G which contains all nodes in $\mathcal{T}$ with at most k edges. Specifically, we set $k = 2 \cdot |\mathcal{T}| - 1$ in our problem.

Given an instance of the Steiner tree problem, with G as the input graph, $\mathcal{T}$ as the set of terminal nodes, and $2 \cdot |\mathcal{T}| - 1$ an upper bound on the number of edges, we define a unit-weight instance of the decision version of our problem as follows. Given G as the input graph, $\mathcal{T} = \mathcal{N} \cup \mathcal{R}$ as the union set of the noun phrase nodes and relation phrase nodes, the upper bound on the number of nodes is $2 \cdot |\mathcal{T}|$, and we need to find a graph with a minimum degree of at least 1.

We show that there is a solution for the Steiner tree problem *if and only if* there is a solution for the unit-weight instance of our problem. First, any Steiner tree using at most $2 \cdot |\mathcal{T}| - 1$ edges is also a solution for our problem using at most $2 \cdot |\mathcal{T}|$ nodes. Second, given a solution H for our problem containing at most $2 \cdot |\mathcal{T}|$ nodes, we can compute a Steiner tree with at most $2 \cdot |\mathcal{T}| - 1$ edges by simply taking any spanning tree of H. □

Algorithm 1 GraphDensification

Input: The semantic graph $G = (V, E)$, where $V = \mathcal{N} \cup \mathcal{E} \cup \mathcal{R} \cup \mathcal{P}$; Specifically, $\eta_G(v)$ denotes the neighbour nodes of $v \in V$ in graph G, and $\epsilon_G(v)$ denotes the incident edges of $v \in V$ in graph G.
Output: The dense subgraph $H = (V_H, E_H)$ which satisfies the constraints in Problem 2, where $V_H = \mathcal{N} \cup \mathcal{E}_H \cup \mathcal{R} \cup \mathcal{P}_H$.

```
1: H ← G
2: V' ← sorted(V);        ▷ Sort all v ∈ V based on their weights from least to greatest.
3: for each v ∈ list(V') do
4:     if v ∈ N or v ∈ R then continue;
5:     else
6:         for each n ∈ N do
7:             if v ∈ η_G(n) and |(η_G(n)| > 1 then
8:                 H ← (V_H − v, E_H − ε_G(v));
9:         for each r ∈ R do
10:            if v ∈ η_G(r) and |(η_G(r)| > 1 then
11:                H ← (V_H − v, E_H − ε_G(v));
12: for each e' ∈ E_H do
13:     if Σ_{x∈η_H(e')∧x∉N_H} ρ(x, e') < θ then
14:         Reports a new entity for n ∈ N linked to e';
15: return H;
```

5.2 Greedy Algorithm

As discussed above, finding a dense subgraph in the semantic graph G with bounded size (Problem 2) is an NP-hard problem. To address this problem, we extend the greedy algorithm for finding strongly interconnected and size-limited groups in social networks [28, 59], while also locating the new entities.

The pseudo-code of the greedy algorithm is presented in Algorithm 1. We start with the full semantic graph $G = (V, E)$. We first sort all the nodes $v \in V$ in G, based on their weights from least to greatest (line 2). We then iteratively remove the entity nodes and predicate nodes with the

Algorithm 2 KBPearl-Pipeline

Input: The sentence group of the input document $\mathcal{G} = \{g_1, g_2, ..., g_{|\mathcal{G}|}\}$, where each of $\{g_1, g_2, ..., g_{|\mathcal{G}|-1}\}$ contains k sentences, and $g_{|\mathcal{G}|}$ contains ($|\mathcal{G}|$ mod k) sentences.
Output: $F(d)$, the final linking results of the input documents. Note that $F(d)$ contains all the noun phrase-entity edges and relation phrase-predicate edges in each H_i for $g_i \in \mathcal{G}$.

1: $\psi'_e \leftarrow \emptyset$; ▷ Record set of the noun phrase-entity edges.
2: $\psi'_p \leftarrow \emptyset$; ▷ Record set of the relation phrase-predicate edges.
3: $F(d) \leftarrow \emptyset$; ▷ The final linking results.
4: **for** each $g_i \in \mathcal{G}$ **do**
5: Build semantic graph G_i for the document based on g_i;
6: $G_i \leftarrow \psi'_e, \psi'_p$;
7: $H_i \leftarrow GraphDensification(G_i)$; ▷ Employ Algorithm 1.
8: Obtain $\psi_e(H_i)$ and $\psi_p(H_i)$ from H_i as the linking results;
9: $\psi_e^{temp} \leftarrow \psi_e(H_i) - \psi'_e$, $\psi_p^{temp} \leftarrow \psi_p(H_i) - \psi'_p$;
10: $\psi'_e \leftarrow \psi_e^{temp}$, $\psi'_p \leftarrow \psi_p^{temp}$;
11: $F(d) \leftarrow \psi_e(H_i), \psi_p(H_i)$;
12: **return** $F(d)$;

smallest weighted degree (lines 3 - 11). Specifically, to guarantee that we capture the coherence of the linkings for all the entity nodes and predicate nodes, we enforce each noun phrase node to remain connected with one entity node (lines 6 - 8), and each relation phrase node to remain connected with one predicate node (lines 9 - 11). The algorithm terminates when (1) at least one of the noun phrase nodes or relation phrase nodes has the minimum degree in H, or (2) a noun phrase node $n \in \mathcal{N}$ or a relation phrase node $r \in \mathcal{R}$ will be disconnected in the next loop.

To recognize the new entities, we further conduct a post-processing task. For each entity node $e' \in \mathcal{E}_H$ in the dense subgraph H, we compute the sum of the weights of the incident edges of e' in H (except for the noun phrase-entity edges). If it is smaller than a given threshold θ, we report a new entity for the noun phrase linked to e' (lines 12 - 14). We discuss the learning of the threshold θ in Section 6.1.

The time complexity of Algorithm 1 is analyzed as follows. We record the weights of each node during the graph construction process and use a list to record the neighbour nodes of each node. The loop in lines 3 to 11 takes $O(|V'|) = O(|V|)$ time to traverse all the node $v \in V'$. Lines 6 to 8 search for every noun phrase node $n \in \mathcal{N}$, which takes $O(|\mathcal{N}|)$ time. Similarly, lines 9 to 11 take $O(|\mathcal{R}|)$ time. To locate new entities, lines 12 to 14 need $O(|\mathcal{E}_H|)$ time. To summarize, let $\beta = |\mathcal{N}|+|\mathcal{R}|$, the time complexity of Algorithm 1 is $O(|V|\cdot(|\mathcal{N}|+|\mathcal{R}|)+|\mathcal{E}_H|) \leq O(|V|\cdot(|\mathcal{N}|+|\mathcal{R}|)+|\mathcal{N}|+|\mathcal{R}|) = O(\beta \cdot |V| + \beta) \approx O(\beta \cdot |V|)$. Specifically, since we reduce the size of the semantic graph via the triple canonicalization in Step 1.3 of Stage 1, β will be a relatively small number for one document.

5.3 Efficient and Effective Linking

However, the assumption that all entities mentioned in a document are densely connected with each other in a KB is not always correct. This assumption will hurt the performance in both effectiveness and efficiency.

In the aspect of effectiveness, in a long-text document, not every pair of entities or predicates shares strong relatedness. Take the case illustrated in Figure 4 as an example (we omit the predicates for simplicity). Among the five entities, only two pairs of them are closely related. This indicates that the sparse coherence between the entities in documents is

Algorithm 3 KBPearl-NearNeighbour

Input: The input document $d = \{a_1, a_2, ..., a_{|d|}\}$, where each a_i is the sentence in d; side information $S(d)$; canonicalized triples $T^\star(d)$; maximum number of near neighbour of noun phrase distance k.
Output: $F(d)$, the final linking results of the input documents, which contains all the noun phrase-entity edges and relation phrase-predicate edges in the dense subgraph H of G.

1: Generate $V = \mathcal{N} \cup \mathcal{E} \cup \mathcal{R} \cup \mathcal{P}$ from $S(d)$ and $T^\star(d)$ (Section 4.1), E includes all the noun phrase-entity edges for each $n \in \mathcal{N}$, and all the relation phrase-predicate edges for each $r \in \mathcal{R}$; we use $\eta_k(n_i)$ to record the k noun phrase as near neighbours of the noun phase $n_i \in \mathcal{N}$, $\mathcal{E}(n_i)$ to denote the candidate entities of $n_i \in \mathcal{N}$, $\mathcal{P}(r_i)$ to denote the candidate predicate of $r_i \in \mathcal{R}$, $sen(a_i)$ to record the noun phrases and relation phrases in the sentence a_i in d.
2: **for** each $n_i \in \mathcal{N}$ **do**
3: **for** each $n_j \in \eta_k(n_i)$ **do**
4: **for** each $e_i \in \mathcal{E}(n_i)$ AND each $e_j \in \mathcal{E}(n_j)$ **do**
5: **if** $e_i \neq e_j$ **then** $E \leftarrow \rho(e_i, e_j)$;
6: **for** each $a_i \in d$ **do**
7: **for** each $r_i \in sen(a_i)$ **do**
8: **for** each $r_j \in sen(a_j)$ **do**
9: **for** each $p_i \in \mathcal{P}(r_i)$ AND each $p_j \in \mathcal{P}(r_i)$ **do**
10: **if** $p_i \neq p_j$ **then** $E \leftarrow \rho(p_i, p_j)$;
11: **for** each $n_i \in sen(a_i)$ **do**
12: **for** each $r_j \in sen(a_j)$ **do**
13: **for** each $e_i \in \mathcal{E}(n_i)$ AND each $p_j \in \mathcal{P}(r_i)$ **do**
14: $E \leftarrow \rho(e_i, p_j)$;
15: $G \leftarrow (V, E)$
16: $H \leftarrow GraphDensification(G)$; ▷ Employ Algorithm 1.
17: $F(d) \leftarrow \psi_e(H), \psi_p(H)$;
18: **return** $F(d)$;

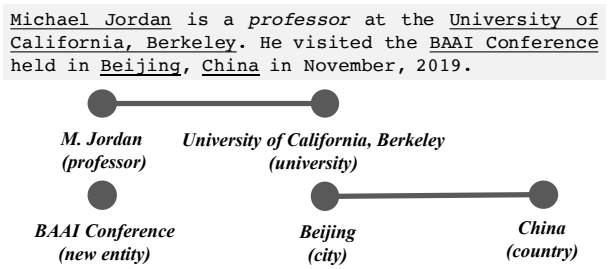


Figure 4: The coherence of entities in the sample document where entity mentions are underlined. The edge represents real semantic relatedness between entities recorded in KB.

common, especially when there are new entities that cannot be connected to any other nodes. Current researches that propose to find the top-k densest subgraphs in a large graph [4, 23] and overlapping community detection [65] partially address this issue. However, all these methods require a large graph that contains the weight information of every edge as input. This requires the pre-computation of each entity-entity edge, predicate-predicate edge and entity-predicate edge.

Moreover, in the aspect of efficiency, suppose that the total number of entity nodes (resp., predicate nodes) in the semantic graph is $|\mathcal{E}|$ (resp., $|\mathcal{P}|$), computing the weights of every entity-entity edges requires $\binom{|\mathcal{E}|}{2}$ (resp., $\binom{|\mathcal{P}|}{2}$) computations. Even if the task is parallelizable, overcoming such complexity is necessary to achieve satisfactory scalability, especially for the on-the-fly tasks such as real-time KBP.

In order to address the issue mentioned above, we propose two variants of the KBPearl system as solutions.

KBPearl in Pipeline Mode (KBPearl-PP). We show the pseudo-code of the pipeline mode in Algorithm 2. Given a document as input, we first separate it into sentence groups $\mathcal{G} = \{g_1, g_2, .., g_{|\mathcal{G}|}\}$ (the sentence tokenization can be obtained in Step 1.1), where each group $\{g_1, g_2, .., g_{|\mathcal{G}|-1}\}$ contains k sentences and the last group $g_{|\mathcal{G}|}$ contains ($|\mathcal{G}|$ mod k) sentences. For each document composed of $g_i \in \mathcal{G}$, we build semantic graph G_i (line 5), and perform Algorithm 1 to obtain a dense subgraph H_i (line 7). Specifically, this task is processed in sequential order, and the linking results in H_i will be utilized to derive H_{i+1} (lines 6 - 10).

We analyze the time complexity of the pipeline mode in Algorithm 2 as follows. For a document with γ sentences, there will be $|\mathcal{G}| = \frac{\gamma}{k} + 1$ iterations. Since the complexity of generating H_i based on G_i in each iteration is $O(\beta_i \cdot |V_i| + \beta_i)$ (see Section 5.2), where V_i is the total number of nodes in G_i, the total time complexity is $O((\frac{\gamma}{k}+1) \cdot (\beta_i \cdot |V_i| + \beta_i)) = O((\frac{\gamma}{k}+1) \cdot \beta_i \cdot (|V_i|+1)) \approx O(\frac{\gamma}{k}\beta_i \cdot |V_i|)$. Note that β_i is the total number of noun phrases and relation phrases in the sentence group where g_i is built on, which is relatively small because of the canonicalization in Step 1.3.

KBPearl in Near-Neighbour Mode (KBPearl-NN). We show the pseudo-code of the near-neighbour mode in Algorithm 3. Specifically, to construct the edges of a semantic graph (lines 2 - 14), the entity-entity edges are computed only if the pair of the related noun phrases are within k distance in the document d (lines 2 - 5). For example, in Figure 4, the distance between `Michael Jordan` and `Beijing` is 3, while the distance between `BAAAI Conference` and `Beijing` is 1. The predicate-predicate edges are computed only if the pair of the related relation phrases are in the same sentence (lines 6 - 10). The entity-predicate edges are computed only if the related noun phrase and relation phrase are in the same sentence (lines 11 - 14).

Suppose that α is the average number of candidates for a noun phrase or relation phrase. Originally, constructing the edges among all entity nodes and predicate nodes in the semantic graph G take $O(|\mathcal{E}|^2 + |\mathcal{P}|^2 + |\mathcal{E}||\mathcal{P}|) = O(\alpha^2|\mathcal{N}|^2 + \alpha^2|\mathcal{R}|^2 + \alpha^2|\mathcal{N}||\mathcal{R}|) = O(\alpha^2(|\mathcal{N}|^2 + |\mathcal{R}|^2 + |\mathcal{N}||\mathcal{R}|))$ time. Suppose that the average number of noun phrases and relation phrases in a sentence is w, and let $|d|$ denote the average number of sentences in one document. Algorithm 3 reduces this time complexity to $O(\alpha^2 k|\mathcal{N}| + |d|(w^2\alpha^2 + w^2\alpha^2)) = O(\alpha^2(k|\mathcal{N}| + 2|d|w^2))$. To achieve promising efficiency, k is set as a small number compared with $|\mathcal{N}|$ and $|\mathcal{R}|$.

6. EXPERIMENTS

In our experiments, we first compare the performance of different variants of KBPearl on the joint entity and relation linking task against the state-of-the-art techniques. We then demonstrate the ability of KBPearl on determining new entities. Third, we validate the effectiveness and efficiency of KBPearl based on the length of the source document.

6.1 Experiment Settings

Benchmarks. We validate the performance of KBPearl and the baselines on several real-world datasets.

- **ReVerb38** is a dataset proposed for Open KB canonicalization [61] with 45K Freebase triples extracted from Clueweb09 [8] corpus. We manually annotate facts in 38 documents based on Wikidata and DBpedia as ground truths. The average document length is 18.34 words.
- **NYT2018** contains news articles collected from the New York Times (*nytimes.com*) in 2018 from various domains [32]. We select 30 documents and manually annotate the facts based on Wikidata and DBpedia as ground truths. The average document length is 206.20 words.
- **LC-QuAD2.0** [15] comprises 6046 complex questions for Wikidata (80% questions are with more than one entity and relation). We select 1942 questions for testing, where the average question length is 14.13 words.
- **QALD-7-Wiki** [60] is one of the most popular benchmark datasets for question answering. This dataset is mapped over Wikidata, comprising 100 questions. The average question length is 6.87 words and over 50% of the questions include a single entity and relation.
- **T-REx** [18] is a benchmark dataset which evaluates KBP, relation extraction, and question answering. We select 1815 documents with Wikidata triples aligned as ground truths. The average document length is 116.14 words.
- **Knowledge Net** [39] is a benchmark dataset for automatic KBP with facts extracted from natural language text on the web. It contains 3977 documents, and 6920 triples linked to Wikidata. The average document length is 157.22 words.
- **CC-DBP** [26] is a benchmark dataset for KBP based on Common Crawl and DBpedia. We select 969 documents with non-empty triples as ground truths. The average document length is 18.39 words.

Specifically, the original CC-DBP datasets only provide ground truths that linked to DBpedia, while T-REx, Knowledge Net, LC-QuAD2.0, and QALD-7-Wiki provide ground truths linked to Wikidata. In order to utilize all the datasets for evaluation, we employ SPARQL query[1]. We produce the entity mapping between DBpedia and Wikidata based on the relation *owl:sameAs* in DBpedia, and the relation mapping based on the property *owl: equivalentProperty* in DBpedia as well as the predicate *P1628 ("equivalent property")* in Wikidata. To ensure fairness, we only evaluate the documents where both the DBpedia and Wikidata ground truths are not null.

Baselines. We compare the performance of our system with several state-of-the-art techniques listed as follows.

- **DBpedia Spotlight** [11,38] is a popular baseline for Entity Linking in TAC KBP [35, 37, 51] based on DBpedia. We adopt the best configurations suggested in [38], where annotation confidence is set as 0.6 and support to be 20.
- **TagMe** [20] is a popular baseline for Entity Linking in TAC KBP datasets [30, 49] and Semantic Evaluation [50] based on DBpedia. We follow all the suggested configurations, and filter the linking results with a confidence lower than 0.3.
- **ReMatch** [42] is one of the top-performing tools used for the Relation Linking task, especially in the question-answering community [16, 52, 58]. We follow all the suggested configurations. Specifically, we set the minimum (resp., maximum) length of combinatorials of input text to be 2 (resp., 3), and the winner threshold to be 0.6.
- **Falcon** [52] performs joint entity and relation linking of short text, leveraging several fundamental principles of English morphology and an extended KB which merges entities and relations from DBpedia and Wikidata. We

[1]https://www.w3.org/TR/rdf-sparql-query/

adopt all the default configurations. The maximum length of combinatorials of input text is set to be 3.

- **EARL** [16] also performs entity linking and relation linking as a joint task, by formulating and solving a Generalized Traveling Salesman Problem among the candidate nodes in the knowledge graph. We adopt all the default configurations suggested by the API of EARL.
- **QKBfly** [44] is proposed to dynamically conduct information extraction tasks based on ClausIE [13] and produce canonicalized triples linked to KB in an on-the-fly manner. We obtain the code of QKBfly from the authors, and adopt all the default configurations suggested in the paper. Note that QKBfly does not perform the relation linking task, but only canonicalizes the relation phrases based on PATTY [43], a relational pattern dictionary. Hence, we do not compare it in our relation linking task.
- **KBPearl-PP** and **KBPearl-NN** are our system that execute in the pipeline mode and the near-neighbour mode, respectively.

Specifically, some of the tools (DBpedia Spotlight, TagMe, Falcon, and EARL) are designed to process short text. To make them feasible on long-text documents, we follow the setting in TagMe [20] to conduct sentence tokenization on the documents with more than 30 words. One document is divided into a list of sentences to pass to these tools for further processing.

Implementation Details of KBPearl. We follow the Opentapioca project [14] to index 64,126,653 entities and predicates from the Wikidata JSON dump of 2018-07-22 with Solr (Lucene)[2]. We list the implementation details of our system as follows.

- In Side information Extraction (Step 1.1 in Stage 1), we employ the Washington KnowItAll project for sentence tokenization[3], the NLTK Toolkit [34] for part-of-speech (POS) tagging, the Stanford CoreNLP toolkit [36] for noun-phrase chunking and named entity recognition, and SUTIME [9] for Time Tagger.
- In Triple Extraction (Step 1.2 in Stage 1), we employ four different Open IE tools for performance evaluation, including ReVerb [19], MinIE (in *safe* mode) [24], ClausIE [13] and Stanford Open IE Tool [2]. Specifically, we choose one of the four tools to execute our system each time. The evaluation details are presented in Section 6.2.
- In Semantic Graph Construction (Step 2.1 in Stage 2), we obtain the aliases of each entity and predicate item from Wikidata by directly querying the values of its *aliases* property. Moreover, to compute the weight of keyphrases, we select the top 5% of entities with the largest number of sitelinks, and 5% of predicates with the largest number of statements as our samples for calculation. The rest of the keyphrases are assigned the maximum weight in the sample set of keyphrases during the computation.
- For the hyper-parameter training of θ in the greedy algorithm to derive the dense subgraph (Step 2.2 in Stage 2, presented in Section 5.2), we utilize the ground truths labeled for the NYT2018 dataset. For the noun phrases detected as named entity mentions and are reported to be non-linkable, we manually label whether they are emerging entities (i.e., entities not contained in Wikidata). We label 127 noun phrases as emerging entities. We train θ based on LBFGS optimization. Specifically, we exclude all the validation data where the ground truths are used to train the hyper-parameters in evaluation.

Evaluation Metrics. We use precision (P), recall (R) and F1-score (F) for performance evaluation. Precision is defined as the number of correct linkings provided by the system, divided by the total number of linkings provided by the system. Recall is defined as the number of correct linkings provided by the system, divided by the total number of ground truths. F1-score computes the harmonic mean of precision and recall, i.e., $F1 = \frac{2*P*R}{P+R}$. Please refer to [48] for a more detailed report and examples of calculation.

Experiment Setting. All the experiments presented are conducted on a server with 128GB RAM, 2.4GHz CPU, with Ubuntu 18.04.1 LTS installed.

6.2 Performance Evaluation

Performance of KBPearl and Baselines on the Joint Entity and Relation Linking Task. We first evaluate the results of the entity linking of our system and the state-of-the-art techniques. As shown in Table 1, we conduct the experiments on 7 datasets, and compare the performance of our system with 5 other competitors, including Falcon, EARL, DBpedia Spotlight (referred to as Spotlight for simplicity), TagMe, and QKBfly. On the short-text datasets (LC-QuAD2.0, QALD-7-Wiki, CC-DBP, and ReVerb38), both variants of KBPearl achieve satisfactory performance. Moreover, on the long-text datasets (T-REx, Knowledge Net, and NYT2018), both variants of KBPearl significantly outperform the baselines in the precision and F1-score. Specifically, KBPearl-NN obtains the best performance of all long-text datasets. This is because that long-text documents contain more valuable evidence to capture the coherence among the knowledge in the source text.

On the other hand, both variants of KBPearl achieve the second-best precision on LC-QuAD2.0 and QALD-7-Wiki, while Spotlight and Falcon obtain better results. One reason is that both datasets provide interrogative sentences as text. The quality of Open IE results on interrogative sentences may not be as satisfactory as on declarative sentences. Moreover, KBPearl tends to extract more knowledge from the text. For example, for the short-text input "`Is the wife of Obama called Michelle?`", apart from the entities for `Obama` and `Michelle`, KBPearl also provides entity *Q188830 (wife)* based on `wife`, even though it also provides predicate *P26 (spouse)* based on `wife`. Therefore, the precision of KBPearl is affected due to such false positives.

Note that KBPearl, TagMe, and Spotlight always achieve relatively high recall. The reason is that they provide more linking results than the other tools. Therefore, although some of their results are not correct (low precision), they have covered most of the truths (high recall).

We then evaluate the relation linking task. We conduct the experiments on 5 datasets, and compare the performance of our system with 3 other competitors, including Falcon, EARL, and ReMatch. Specifically, two datasets are excluded. One is Knowledge Net, which does not provide the relation linking results as standard Wikidata predicates. The other is CC-DBP, where most of the triples (over 80%) are not labeled with relations. As shown in Table 2, KBPearl significantly outperforms all the other baselines in the

[2]https://lucene.apache.org/solr/
[3]https://github.com/knowitall

Table 1: Performance of the entity linking task. Both KBPearl-PP and KBPearl-NN employ MinIE for Open triple extraction. The best and the second-best performance values are in bold.

Dataset (# of words per document)	System	Precision	Recall	F1
ReVerb38 (18.34)	Falcon	0.498	0.464	0.480
	EARL	0.496	0.535	0.515
	Spotlight	0.642	0.605	0.623
	TagMe	0.512	**0.661**	0.577
	QKBfly	**0.724**	0.528	0.611
	KBPearl-PP	**0.643**	**0.664**	**0.653**
	KBPearl-NN	**0.643**	**0.664**	**0.653**
NYT2018 (206.20)	Falcon	0.275	0.470	0.347
	EARL	0.282	0.526	0.367
	Spotlight	0.393	0.524	0.449
	TagMe	0.114	0.528	0.188
	QKBfly	0.410	0.477	0.441
	KBPearl-PP	**0.422**	**0.744**	**0.539**
	KBPearl-NN	**0.491**	**0.694**	**0.575**
LC-QuAD2.0 (14.13)	Falcon	0.533	0.598	0.564
	EARL	0.403	0.498	0.445
	Spotlight	**0.585**	**0.657**	**0.619**
	TagMe	0.352	**0.864**	0.500
	QKBfly	0.518	0.479	0.498
	KBPearl-PP	**0.561**	0.647	**0.601**
	KBPearl-NN	**0.561**	0.647	**0.601**
QALD-7-Wiki (6.87)	Falcon	**0.708**	0.651	0.678
	EARL	0.516	0.460	0.486
	Spotlight	0.619	0.634	0.626
	TagMe	0.349	0.661	0.457
	QKBfly	0.592	0.510	0.548
	KBPearl-PP	**0.647**	**0.715**	**0.679**
	KBPearl-NN	**0.647**	**0.715**	**0.679**
T-REx (116.14)	Falcon	0.141	0.203	0.167
	EARL	0.305	0.505	0.380
	Spotlight	0.328	0.472	0.387
	TagMe	0.057	0.064	0.060
	QKBfly	0.248	0.271	0.259
	KBPearl-PP	**0.329**	**0.513**	**0.401**
	KBPearl-NN	**0.340**	**0.554**	**0.421**
Knowledge Net (157.22)	Falcon	0.253	0.380	0.304
	EARL	0.234	0.388	0.292
	Spotlight	0.255	0.382	0.306
	TagMe	0.254	0.362	0.299
	QKBfly	**0.322**	**0.472**	**0.382**
	KBPearl-PP	0.297	0.446	0.356
	KBPearl-NN	**0.327**	**0.466**	**0.384**
CC-DBP (18.39)	Falcon	0.244	0.303	0.270
	EARL	0.277	0.605	0.380
	Spotlight	0.382	0.604	0.468
	TagMe	0.231	0.615	0.336
	QKBfly	0.339	0.433	0.380
	KBPearl-PP	**0.415**	**0.620**	**0.497**
	KBPearl-NN	**0.418**	**0.620**	**0.499**

relation linking task in both short-text data and long-text data. Specifically, the performance of all systems on T-REx is relatively low. The reason is that T-REx provides less number of relation linking ground truths for long-text documents that actually contain more relations.

As presented in Table 1 and Table 2, the performance of all the systems on the entity linking task is better than the relation linking task. The reason is that some of the relations expressed in the natural language text have various representations. For example, in the sentence "`Barack Obama was the US President`", the correct extraction should be (*Q76, P39, Q11696*) (*Barack Obama, political office held, President of the United States*). However, all the systems recognize the more general predicate *P31* (*instance of*) instead of *P39*(*political office held*) from the original sentence.

Table 2: Performance of the relation linking task. Both KBPearl-PP and KBPearl-NN employ MinIE for Open triple extraction. The best and the second-best performance values are in bold.

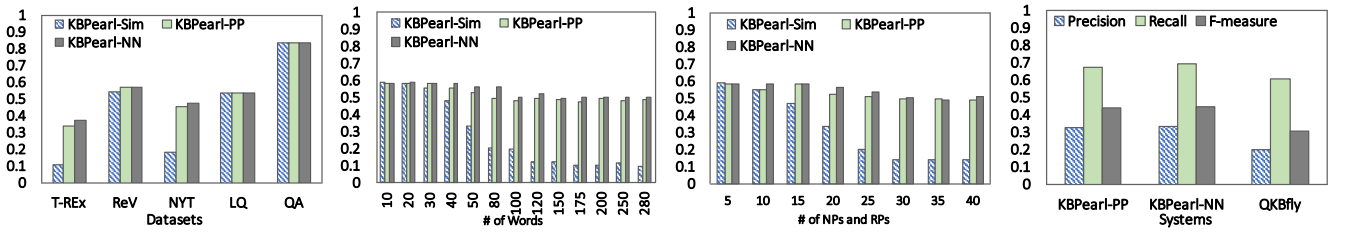
Dataset (# of words per document)	System	Precision	Recall	F1
ReVerb38 (18.34)	Falcon	0.137	0.220	0.169
	EARL	0.280	0.431	0.339
	ReMatch	0.292	0.223	0.253
	KBPearl-PP	**0.403**	**0.452**	**0.426**
	KBPearl-NN	**0.403**	**0.452**	**0.426**
NYT2018 (206.20)	Falcon	0.151	0.203	0.173
	EARL	0.201	0.245	0.221
	ReMatch	0.218	0.297	0.251
	KBPearl-PP	**0.302**	**0.405**	**0.346**
	KBPearl-NN	**0.313**	**0.494**	**0.383**
LC-QuAD2.0 (14.13)	Falcon	0.302	0.325	0.313
	EARL	0.259	0.251	0.255
	ReMatch	0.201	0.214	0.207
	KBPearl-PP	**0.566**	**0.479**	**0.410**
	KBPearl-NN	**0.566**	**0.479**	**0.410**
QALD-7-Wiki (6.87)	Falcon	0.337	0.329	0.333
	EARL	0.214	0.222	0.218
	ReMatch	0.208	0.203	0.205
	KBPearl-PP	**0.482**	**0.399**	**0.437**
	KBPearl-NN	**0.482**	**0.399**	**0.437**
T-REx (116.14)	Falcon	0.026	0.072	0.038
	EARL	0.126	0.157	0.140
	ReMatch	0.125	0.151	0.137
	KBPearl-PP	**0.136**	**0.188**	**0.157**
	KBPearl-NN	**0.139**	**0.187**	**0.159**

Performance of KBPearl based on different Open IE tools. We also conduct experiments to evaluate the performance of KBPearl when employing different Open IE tools to extract the knowledge from natural language text in Step 1.2 of Stage 1. Specifically, MinIE (in *safe* mode) [24], ReVerb [19], Stanford Open IE Tool [2] (referred to as Standford for simplicity), and ClausIE [13] are studied. Note that we only involve Falcon and EARL as baselines for comparison, because they are the only tools that perform both entity linking task and relation linking task.

As listed in Table 3, all the variants of KBPearl based on different Open IE tools outperform the baselines in 4 datasets. MinIE achieves the best and second-best F1 score of all the time. The reason is that MinIE provides more compact extractions, and therefore gives most of the correct linkings, which leads to high recall. Specifically, the experimental results indicates that there will not be a very huge difference in the performance of KBPearl when employing different Open IE tools to extract knowledge. The reason is that we have conducted noise filtering, redundancy filtering, and canonicalization on the Open triples at Step 1.3, to reduce the redundant triples.

Table 3: Performance of joint entity and relation linking of KBPearl-NN based on different Open IE tools employed in Step 1.2, Stage 1. "*#Extr.*" denotes the total number of triples after the noise filtering and canonicalization in Step 1.3, Stage 1. The precision (P) and recall (R) are the average value of those for the entity linking task and relation linking task. The best and the second-best performance values are in bold.

Systems	QALD-7-Wiki				T-REx				ReVerb38				NYT2018			
	#Extr.	P	R	F1	*#Extr.*	P	R	F1	*#Extr.*	P	R	F1	*#Extr.*	P	R	F1
KBP-MinIE	200	**0.565**	**0.557**	**0.561**	373,553	0.239	**0.370**	**0.291**	183	**0.523**	**0.558**	**0.540**	402	0.402	**0.594**	**0.479**
KBP-ReVerb	59	**0.631**	0.481	0.546	100,063	**0.353**	0.243	0.288	67	**0.606**	0.448	0.515	158	**0.516**	0.330	0.403
KBP-Stanford	189	0.560	0.539	0.549	306,659	0.251	0.330	0.285	146	0.517	0.545	**0.531**	350	0.481	**0.459**	0.470
KBP-ClausIE	175	0.554	**0.549**	**0.551**	299,582	**0.287**	**0.395**	**0.332**	132	0.503	**0.563**	**0.531**	325	**0.502**	0.443	**0.471**
Falcon	N.A	0.523	0.490	0.506	N.A	0.084	0.138	0.104	N.A	0.318	0.342	0.329	N.A	0.163	0.337	0.220
EARL	N.A	0.365	0.341	0.353	N.A	0.216	0.331	0.261	N.A	0.388	0.483	0.430	N.A	0.192	0.385	0.256



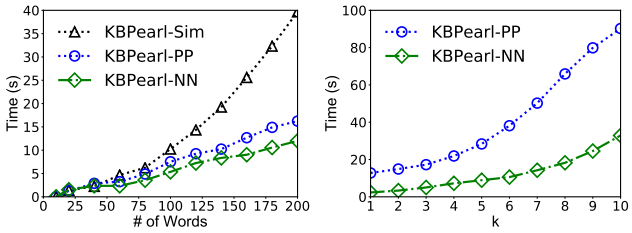
(a) F1-measure of different KBPearl variants on different datasets.

(b) F1-measure of different KBPearl variants with different # of words.

(c) F1-measure of different KBPearl variants with different # of phrases.

(d) Discovery of new entities and predicates.

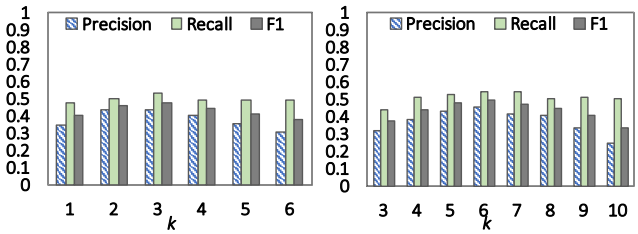
Figure 5: Performance Evaluation.



(a) Computation Time

(b) Computation Time

Figure 6: Efficiency evaluation regarding the length of the documents and different values of parameter k of the KBPearl variants.



(a) KBPearl-PP

(b) KBPearl-NN

Figure 7: Parameter Sensitivity in KBPearl-Pipeline (KBPearl-PP) and KBPearl-Near Neighbour (KBPearl-NN).

Performance of Different Variants of KBPearl. We also evaluate the performance of KBPearl in pipeline mode (KBPearl-PP), KBPearl in near-neighbour mode (KBPearl-NN), and KBPearl which directly constructs the whole semantic graph and conduct graph densification without filtering sparse coherence (KBPearl-Sim). Figure 5(a) illustrates the F-measure of the three variants in joint entity and relation linking task, on T-REx, ReVerb38 (ReV), NYT2018 (NYT), LC-QuAD2.0(LQ), and QALD-7-Wiki (QA). We also present the detailed performance of KBPearl-PP and KBPearl-NN in Table 1 and Table 2. For short-text datasets, such as LC-QuAD2.0, QALD-7-Wiki, and ReVerb38, there is very little difference between the performance of all variants of KBPearl. The reason is that the numbers of candidate entities and predicates are always limited in short text, while the number of sentences is also limited. Therefore, limiting the number of neighbours or the number of sentence groups will not have different performances. On the other hand, for long-text datasets such as T-REx and NYT2018, KBPearl-PP and KBpearl-NN perform better than KBPearl-Sim. This proves the capability of our system to handle the sparse coherence (illustrated in Figure 4) which is common in the real-world scenarios for most long-text documents.

We further investigate the performance of different variants of KBPearl in terms of the length of the document. Specifically, the length of the documents is determined based on the number of words (Figure 5(b)), and the number of noun phrases and relation phrases in the canonicalized Open triples (Figure 5(c)). The performance of KBPearl is based on MinIE. KBPearl-PP and KBPearl-NN achieve satisfactory performance with both short-text and long-text documents, while the F1-measure of KBPearl-Sim decreases with long documents. This also demonstrates the effectiveness of the pipeline mode and near-neighbour mode of KBPearl.

Moreover, Figure 6(a) presents the efficiency evaluation of KBPearl regarding the lengths of the sentences. We can observe that with more than 75 words, the time cost of KBPearl-Sim increases in an exponential manner, while the time cost of both KBPearl-PP and KBPearl-NN is still linear to the number of words.

Parameter Sensitivity in Different Variants of KBPearls. We first study the time cost of KBPearl-PP and KBPearl-NN based on different k. As shown in Figure 6(b), the time cost of KBPearl-NN is linear to k when it is smaller than 8, while the time cost of KBPearl-PP is linear to k when it is smaller than 5. We also investigate the performance

of both variants based on different k. As demonstrated in Figure 7(a), the precision of KBPearl-PP is affected with k larger than 4. Moreover, Figure 7(a) indicates that the precision of KBPearl-NN drops with k larger than 8. This suggests that a wise selection of k will be around 3 to 4 for KBPearl-PP, and 6 to 7 for KBPearl-NN. The result indicates that in most cases, the information in 3 to 4 sentences in a document is valuable enough for knowledge inference and linking with global coherence. On the other hand, when conducting entity disambiguation, knowledge of the 6 to 7 nearest noun phrases can give valuable support.

Results of Detecting New Entities. As demonstrated in Figure 5(d), we evaluate the performance of KBPearl-PP, KBPearl-NN, and QKBfly on new entity discovery on NYT2018. We do not involve other baselines for comparison, since they do not create new entities for non-linkable facts. Specifically, both variants of KBPearl achieve better performance than QKBfly. The reason that all systems have relatively low precision and high recall is that it is not trivial for them to recognize entities containing long phrases. For example, in a sentence containing "`Girl from the North Country`" (a song), both QKBfly and our system will extract `Girl` and `the North Country` separately and report new entities, which introduces false positives.

Summary. In conclusion, the joint entity and relation linking task in KBPearl performs better than the state-of-the-art techniques, especially in long-text datasets. KBPearl relies on Open IE tools for knowledge extraction, but with side information preparation, the performance of KBPearl will not be affected by the output quality of the Open IE tools. Moreover, the two variants of KBPearl enhance the effectiveness and efficiency of our method on long-text documents. KBpearl is also capable of discovering new entities.

7. RELATED WORKS

Knowledge Base Population (KBP). Recently, a lot of research in the text analysis community focus on KBP from external data sources [25, 29, 30, 46]. There are two major tasks: (1) *Entity Discovery and Linking*, which extracts the entities from the unstructured text and link them to an existing KB, and (2) *Slot Filling*, i.e., Relation Discovery and Linking, which conduct information mining about relations and types of the detected entities. However, most of the techniques proposed for KBP treat them as separate tasks [25,29], and therefore the global coherence among the candidate entities and predicates cannot be utilized. Moreover, declarative methods to IE and KBP, such as Deepdive [12,45,57] and SystemT [10], rely on predefined sets of predicates and rules. Hence, these techniques are not suitable for in-time KBP. QKBfly [44] is proposed as a solution for dynamic KB construction and KBP. Nevertheless, it also relies on a pre-generated dictionary of relational patterns to generate canonicalized facts.

Entity Linking and Relation Linking. Most of the Entity Linking (EL) works focus on three tasks: candidate entity generation [27, 55, 66, 67], candidate entity ranking [33, 54] and non-linkable mention prediction [55, 56]. However, all these works only assign the non-linkable mentions to a *NIL* entity, without further clustering and processing on the *NIL* entities. Furthermore, linking relation phrases to predicates in KBs is a relatively new field of study. Rematch [42] is the first attempt in this direction, which measures the semantic similarity between the relation phrases and the predicates in KBs based on graph distances from Wordnet [40]. Moreover, SIBKB [58], AskNow [17], and QKBfly [44] utilize large dictionary resources for textual patterns that denote binary relations between entities (e.g., PATTY [43]), to discover synonymous relation phrases in unstructured text and assist relation linking. Particularly, EARL [16] and Falcon [52] are the first two works that perform both the entity linking task and relation linking task. Specifically, EARL exploits the connection density between entity nodes in KBs, while Falcon leverages the fundamental principles of English morphology and extended self-integrated KBs without considering the coherence among the entities and predicates. However, neither of them processes the non-linkable noun phrases.

Open IE. The Open IE methods [13, 47, 53] overcome the limitations of KBP, but face issues that neither entities nor predicates in the Open triples are canonicalized. There are some recent research works that focus on the canonicalization of Open triples by clustering the triples with the same semantic meaning. *Galárraga* [21] performs clustering on noun phrases over manually-defined feature spaces to obtain equivalent entities, and clusters relation phrases based on rules discovered by AMIE [22]. FAC [64] solves *Galárraga*'s problem in a more efficient graph-based clustering method with pruning and bounding techniques. Furthermore, CESI [61] and SIST [32] conduct the canonicalization of entities and relations jointly based on the external and internal side knowledge. However, all these works only improve the output of Open IE techniques. None of them investigates the method to map and link the canonicalized facts to the existing KBs for enhancement and population.

8. CONCLUSION

In this paper, we present an end-to-end system, KBPearl, for KB population. KBPearl takes an incomplete KB and a large corpus of text as input, to (1) organize the noisy extraction from Open IE into canonicalized facts; and (2) populate the KB by joint entity and relation linking, utilizing the facts and the side information extracted from the source documents. Specifically, we employ a semantic graph-based approach to capture the knowledge in the source document in a global coherence, and to determine the best linking results by finding the densest subgraph effectively and efficiently. Our system is also able to locate the new entities mentioned in the source document. We demonstrate the effectiveness and efficiency of KBPearl against the state-of-the-art techniques, through extensive experiments on real-world datasets.

Acknowledgments

We acknowledge all our lovely teammates and reviewers for their valuable advice on this paper. This work is partially supported by the Hong Kong RGC GRF Project 16202218, CRF project C6030-18G, AOE project AoE/E-603/18, the National Science Foundation of China (NSFC) under Grant No. 61729201, Science and Technology Planning Project of Guangdong Province, China, No. 2015B010110006, Hong Kong ITC grants ITS/044/18FX and ITS/470/18FX, Didi-HKUST Joint Research Lab Grant, Microsoft Research Asia Collaborative Research Grant, WeChat Research Grant, WeBank Research Grant, and Huawei PhD Fellowship.

9. REFERENCES

[1] R. Andersen and K. Chellapilla. Finding dense subgraphs with size bounds. In *International Workshop on Algorithms and Models for the Web-Graph*, pages 25–37. Springer, 2009.

[2] G. Angeli, M. J. J. Premkumar, and C. D. Manning. Leveraging linguistic structure for open domain information extraction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 344–354, 2015.

[3] 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*, pages 722–735. 2007.

[4] O. D. Balalau, F. Bonchi, T. Chan, F. Gullo, and M. Sozio. Finding subgraphs with maximum total density and limited overlap. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*, pages 379–388. ACM, 2015.

[5] D. Bamman, T. Underwood, and N. A. Smith. A bayesian mixed effects model of literary character. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 370–379, 2014.

[6] K. Bernhard and J. Vygen. Combinatorial optimization: Theory and algorithms. *Springer, Third Edition, 2005.*, 2008.

[7] R. Blanco, B. B. Cambazoglu, P. Mika, and N. Torzec. Entity recommendations in web search. In *International Semantic Web Conference*, pages 33–48. Springer, 2013.

[8] J. Callan, M. Hoy, C. Yoo, and L. Zhao. Clueweb09 data set, 2009.

[9] A. X. Chang and C. D. Manning. Sutime: A library for recognizing and normalizing time expressions. In *Lrec*, volume 2012, pages 3735–3740, 2012.

[10] L. Chiticariu, R. Krishnamurthy, Y. Li, S. Raghavan, F. R. Reiss, and S. Vaithyanathan. Systemt: an algebraic approach to declarative information extraction. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 128–137. Association for Computational Linguistics, 2010.

[11] J. Daiber, M. Jakob, C. Hokamp, and P. N. Mendes. Improving efficiency and accuracy in multilingual entity extraction. In *Proceedings of the 9th International Conference on Semantic Systems*, pages 121–124. ACM, 2013.

[12] C. De Sa, A. Ratner, C. Ré, J. Shin, F. Wang, S. Wu, and C. Zhang. Deepdive: Declarative knowledge base construction. *ACM SIGMOD Record*, 45(1):60–67, 2016.

[13] L. Del Corro and R. Gemulla. Clausie: clause-based open information extraction. In *Proceedings of the 22nd international conference on World Wide Web*, pages 355–366. ACM, 2013.

[14] A. Delpeuch. Opentapioca: Lightweight entity linking for wikidata. *arXiv preprint arXiv:1904.09131*, 2019.

[15] M. Dubey. test set for lcquad 2.0. 7 2019.

[16] M. Dubey, D. Banerjee, D. Chaudhuri, and J. Lehmann. Earl: Joint entity and relation linking for question answering over knowledge graphs. In *International Semantic Web Conference*, pages 108–126. Springer, 2018.

[17] M. Dubey, S. Dasgupta, A. Sharma, K. Höffner, and J. Lehmann. Asknow: A framework for natural language query formalization in sparql. In *European Semantic Web Conference*, pages 300–316. Springer, 2016.

[18] H. Elsahar, P. Vougiouklis, A. Remaci, C. Gravier, J. Hare, F. Laforest, and E. Simperl. 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)*, 2018.

[19] A. Fader, S. Soderland, and O. Etzioni. Identifying relations for open information extraction. In *Proceedings of the conference on empirical methods in natural language processing*, pages 1535–1545. Association for Computational Linguistics, 2011.

[20] P. Ferragina and U. Scaiella. Tagme: on-the-fly annotation of short text fragments (by wikipedia entities). In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 1625–1628. ACM, 2010.

[21] L. Galárraga, G. Heitz, K. Murphy, and F. M. Suchanek. Canonicalizing open knowledge bases. In *CIKM*, pages 1679–1688, 2014.

[22] L. A. Galárraga, C. Teflioudi, K. Hose, and F. Suchanek. Amie: association rule mining under incomplete evidence in ontological knowledge bases. In *WWW*, pages 413–422, 2013.

[23] E. Galbrun, A. Gionis, and N. Tatti. Top-k overlapping densest subgraphs. *Data Mining and Knowledge Discovery*, 30(5):1134–1165, 2016.

[24] K. Gashteovski, R. Gemulla, and L. Del Corro. Minie: minimizing facts in open information extraction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2630–2640, 2017.

[25] J. Getman, J. Ellis, S. Strassel, Z. Song, and J. Tracey. Laying the groundwork for knowledge base population: Nine years of linguistic resources for tac kbp. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.

[26] M. Glass and A. Gliozzo. A dataset for web-scale knowledge base population. In *European Semantic Web Conference*, pages 256–271. Springer, 2018.

[27] S. Guo, M.-W. Chang, and E. Kiciman. To link or not to link? a study on end-to-end tweet entity linking. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1020–1030, 2013.

[28] J. Hoffart, M. A. Yosef, I. Bordino, H. Fürstenau, M. Pinkal, M. Spaniol, B. Taneva, S. Thater, and G. Weikum. Robust disambiguation of named entities in text. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 782–792. Association for Computational Linguistics, 2011.

[29] H. Ji and R. Grishman. Knowledge base population: Successful approaches and challenges. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies-volume 1*, pages 1148–1158. Association for Computational Linguistics, 2011.

[30] H. Ji, J. Nothman, B. Hachey, et al. Overview of tac-kbp2014 entity discovery and linking tasks. In *Proc. Text Analysis Conference (TAC2014)*, pages 1333–1339, 2014.

[31] D. S. Johnson and M. R. Garey. *Computers and intractability: A guide to the theory of NP-completeness*, volume 1. WH Freeman San Francisco, 1979.

[32] X. Lin and L. Chen. Canonicalization of open knowledge bases with side information from the source text. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, pages 950–961. IEEE, 2019.

[33] X. Liu, Y. Li, H. Wu, M. Zhou, F. Wei, and Y. Lu. Entity linking for tweets. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1304–1311, 2013.

[34] E. Loper and S. Bird. Nltk: the natural language toolkit. *arXiv preprint cs/0205028*, 2002.

[35] W. Lu, Y. Zhou, H. Lu, P. Ma, Z. Zhang, and B. Wei. Boosting collective entity linking via type-guided semantic embedding. In *National CCF Conference on Natural Language Processing and Chinese Computing*, pages 541–553. Springer, 2017.

[36] 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*, pages 55–60, 2014.

[37] P. N. Mendes, J. Daiber, M. Jakob, and C. Bizer. Evaluating dbpedia spotlight for the tac-kbp entity linking task. In *Proceedings of the TAC-KBP 2011 Workshop*, pages 118–120, 2011.

[38] P. N. Mendes, M. Jakob, A. García-Silva, and C. Bizer. Dbpedia spotlight: shedding light on the web of documents. In *Proceedings of the 7th international conference on semantic systems*, pages 1–8. ACM, 2011.

[39] F. Mesquita, M. Cannaviccio, J. Schmidek, P. Mirza, and D. Barbosa. 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)*, pages 749–758, 2019.

[40] G. A. Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[41] A. Moro, A. Raganato, and R. Navigli. Entity linking meets word sense disambiguation: a unified approach. *Transactions of the Association for Computational Linguistics*, 2:231–244, 2014.

[42] I. O. Mulang, K. Singh, and F. Orlandi. Matching natural language relations to knowledge graph properties for question answering. In *Proceedings of the 13th International Conference on Semantic Systems*, pages 89–96. ACM, 2017.

[43] N. Nakashole, G. Weikum, and F. Suchanek. Patty: a taxonomy of relational patterns with semantic types. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1135–1145. Association for Computational Linguistics, 2012.

[44] D. B. Nguyen, A. Abujabal, N. K. Tran, M. Theobald, and G. Weikum. Query-driven on-the-fly knowledge base construction. *PVLDB*, 11(1):66–79, 2017.

[45] F. Niu, C. Zhang, C. Ré, and J. W. Shavlik. Deepdive: Web-scale knowledge-base construction using statistical learning and inference. *VLDS*, 12:25–28, 2012.

[46] H. Paulheim. Knowledge graph refinement: A survey of approaches and evaluation methods. *Semantic web*, 8(3):489–508, 2017.

[47] M. Ponza, L. Del Corro, and G. Weikum. Facts that matter. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1043–1048, 2018.

[48] D. M. Powers. Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. 2011.

[49] J. R. Raiman and O. M. Raiman. Deeptype: multilingual entity linking by neural type system evolution. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[50] H. Rosales-Méndez, B. Poblete, and A. Hogan. Multilingual entity linking: Comparing english and spanish. In *LD4IE@ ISWC*, pages 62–73, 2017.

[51] M. Rospocher and F. Corcoglioniti. Joint posterior revision of nlp annotations via ontological knowledge. In *IJCAI*, pages 4316–4322, 2018.

[52] A. Sakor, I. O. Mulang, K. Singh, S. Shekarpour, M. E. Vidal, J. Lehmann, and S. Auer. Old is gold: linguistic driven approach for entity and relation linking of short text. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2336–2346, 2019.

[53] M. Schmitz, R. Bart, S. Soderland, O. Etzioni, et al. 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*, pages 523–534. Association for Computational Linguistics, 2012.

[54] W. Shen, J. Wang, P. Luo, and M. Wang. Liege: link entities in web lists with knowledge base. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1424–1432. ACM, 2012.

[55] W. Shen, J. Wang, P. Luo, and M. Wang. Linden: linking named entities with knowledge base via semantic knowledge. In *Proceedings of the 21st international conference on World Wide Web*, pages 449–458. ACM, 2012.

[56] W. Shen, J. Wang, P. Luo, and M. Wang. Linking named entities in tweets with knowledge base via user

interest modeling. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 68–76. ACM, 2013.

[57] J. Shin, S. Wu, F. Wang, C. De Sa, C. Zhang, and C. Ré. Incremental knowledge base construction using deepdive. *PVLDB*, 8(11):1310–1321, 2015.

[58] K. Singh, A. S. Radhakrishna, A. Both, S. Shekarpour, I. Lytra, R. Usbeck, A. Vyas, A. Khikmatullaev, D. Punjani, C. Lange, et al. Why reinvent the wheel: Let's build question answering systems together. In *WWW*, pages 1247–1256, 2018.

[59] M. Sozio and A. Gionis. The community-search problem and how to plan a successful cocktail party. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 939–948. ACM, 2010.

[60] R. Usbeck, A.-C. N. Ngomo, B. Haarmann, A. Krithara, M. Röder, and G. Napolitano. 7th open challenge on question answering over linked data (qald-7). In *Semantic Web Evaluation Challenge*, pages 59–69. Springer, 2017.

[61] S. Vashishth, P. Jain, and P. Talukdar. Cesi: Canonicalizing open knowledge bases using embeddings and side information. In *WWW*, pages 1317–1327, 2018.

[62] M. Vijaymeena and K. Kavitha. A survey on similarity measures in text mining. *Machine Learning and Applications: An International Journal*, 3(2):19–28, 2016.

[63] D. Vrandečić. Wikidata: A new platform for collaborative data collection. In *WWW*, pages 1063–1064, 2012.

[64] T.-H. Wu, Z. Wu, B. Kao, and P. Yin. Towards practical open knowledge base canonicalization. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 883–892. ACM, 2018.

[65] J. Xie, S. Kelley, and B. K. Szymanski. Overlapping community detection in networks: The state-of-the-art and comparative study. *Acm computing surveys (csur)*, 45(4):43, 2013.

[66] W. Zhang, Y.-C. Sim, J. Su, and C.-L. Tan. Entity linking with effective acronym expansion, instance selection and topic modeling. In *Twenty-Second International Joint Conference on Artificial Intelligence*, 2011.

[67] W. Zhang, J. Su, C. L. Tan, and W. T. Wang. Entity linking leveraging: automatically generated annotation. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 1290–1298. Association for Computational Linguistics, 2010.

[68] Y. Zhang, H. Dai, Z. Kozareva, A. J. Smola, and L. Song. Variational reasoning for question answering with knowledge graph. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.