# TENET: Joint Entity and Relation Linking with Coherence Relaxation

Xueling Lin
The Hong Kong University of Science and Technology
xlinai@cse.ust.hk

Lei Chen
The Hong Kong University of Science and Technology
leichen@cse.ust.hk

Chaorui Zhang
Theory Lab, Huawei Technologies
zhang.chaorui@huawei.com

## ABSTRACT

The joint entity and relation linking task aims to connect the noun phrases (resp., relational phrases) extracted from natural language documents to the entities (resp., predicates) in general knowledge bases (KBs). This task benefits numerous downstream systems, such as question answering and KB population. Previous works on entity and relation linking rely on the *global coherence* assumption, i.e., entities and predicates within the same document are highly correlated with each other. However, this assumption is not valid in many real-world scenarios *sparse coherence* among the entities and predicates within the same document is common. Moreover, there may exist isolated entities or predicates that are not related to any other linked concepts. In this paper, we propose TENET, a joint entity and relation linking technique, which *relaxes the coherence assumption* in an unsupervised manner. Specifically, we formulate the joint entity and relation linking task as a minimum-cost rooted tree cover problem on the knowledge coherence graph constructed based on the document. We then propose effective approximation algorithms with pruning strategies to solve this problem and derive the linking results. Extensive experiments on real-world datasets demonstrate the superior effectiveness and efficiency of our method against the state-of-the-art techniques.

## CCS CONCEPTS

• **Information systems** → *Information extraction*;

## KEYWORDS

knowledge base; entity linking; relation linking

**ACM Reference Format:**
Xueling Lin, Lei Chen, and Chaorui Zhang. 2021. TENET: Joint Entity and Relation Linking with Coherence Relaxation. In *Proceedings of the 2021 International Conference on Management of Data (SIGMOD '21), June 20–25, 2021, Virtual Event, China.* ACM, Xi'an, China, 14 pages. https://doi.org/10.1145/3448016.3457280


*SIGMOD '21, June 20–25, 2021, Virtual Event, China*

ACM ISBN 978-1-4503-8343-1/21/06…$15.00
https://doi.org/10.1145/3448016.3457280

## 1 INTRODUCTION

Nowadays, numerous information extraction techniques have been proposed to extract knowledge from natural language documents to the structural format employed by Knowledge Bases (KBs). Such techniques support a broad range of downstream applications, including question answering [19, 46, 56, 57, 60] and KB population [18, 33, 38]. Among them, the task of *joint entity and relation linking* has drawn significant attention in both industrial and academic communities.

Formally, this task can be described as follows. Given an existing KB and a document as input, our goal is to determine a set of noun phrases and a set of relational phrases from the document, with each noun phrase linked to a correct entity, and each relational phrase linked to a correct predicate in the given KB. We illustrated a real-world scenario in Figure 1. Given the input document, the noun phrase `Michael Jordan` is linked to the entity *M. Jordan (professor)* [1] in Wikidata. Similarly, the relational phrase *studies* is linked to the predicate *field of study* [2] in Wikidata.

There are many existing techniques that address the entity linking and relation linking tasks independently [16, 24, 42, 44] or jointly [56, 57]. However, most techniques disambiguate each noun phrase (resp., relational phrase) in a document **separately**. Recall the example in Figure 1. Without considering the context of the document, the noun phrase `Michael Jordan` is more likely to be linked to the most popular entity *M. Jordan (basketball player)* [3].

To address this issue, some research works, either focusing solely on entity disambiguation task [11, 30, 32, 47, 50, 55] or targeting on the joint entity and relation linking task [19, 38, 46], propose to utilize the **global coherence** among candidate concepts extracted from the same document. These works assume that all the concepts mentioned in the same document are **densely connected** in the KB. Take Figure 1 as an example. If `artificial intelligence` refers to *AI (CS topic)*, then `Michael Jordan` extracted from the same document is more likely to be *M. Jordan (professor)*. Nevertheless, there are still two major challenges that remain unsolved for the joint entity and relation linking task.

The first challenge is **the sparse coherence problem.** In real-world scenarios, especially in long-text documents, not every pair of entities or predicates is strongly related to each other. Phan et al.[51] conduct an analysis of coherence denseness on 7 benchmark datasets. The results indicate that in long-text documents such as MSNBC [15] and AQUAINT [43] (with more than 13 entities per document), each linked entity is highly related to less than 5 other entities averagely. It is even

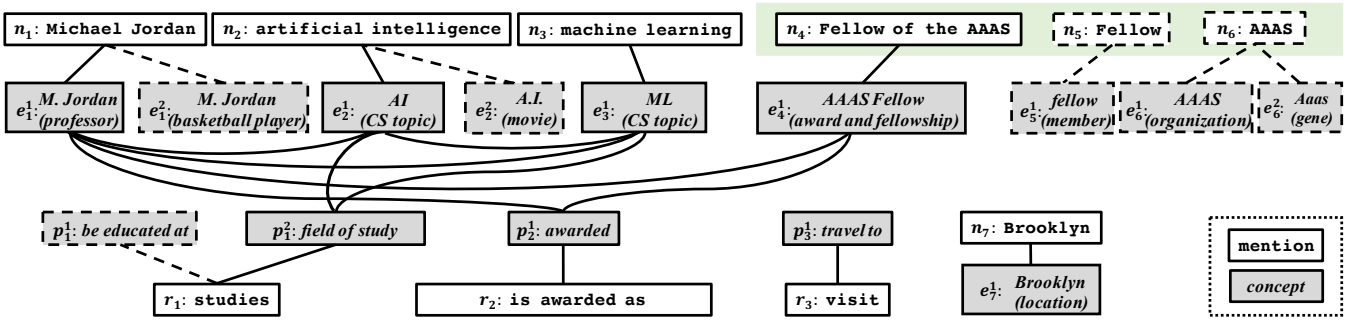



**Figure 1: The semantic coherence among the concepts in the input document "`Michael Jordan studies artificial intelligence and machine learning. He was awarded as the Fellow of the AAAS. He visited Brooklyn in April 2019.`" A solid edge between a pair of linked entities or predicates indicates that there is a strong semantic relatedness (small semantic distance) between them. The dashed edges denote incorrect selections of mentions or concepts.**

more common to encounter **isolated** entities (or predicates) that are not related to any other linked entities or predicates. For example, in Figure 1, neither *Brooklyn (location)* nor *travel to* shares strong semantic relatedness with any other linked concept. This may be attributed to several reasons, such as KB incompleteness, noisy data, or newly emerged phrases in fresh articles. In fact, more than 30% of the noun phrases in ClueWeb09 [14] extracted by Reverb cannot be linked to any Wikipedia entity [37]. Similarly, in the NYT2018 dataset [38], more than 20% noun phrases and 36% relational phrases are non-linkable. Forcing the dense connection of such isolated phrases to the other phrases hurts the disambiguation. Therefore, considering global coherence between every pair of candidate entities (or predicates) is **not necessary** in practice, in terms of both effectiveness and efficiency.

Recently, there are two types of techniques proposed to relax such global coherence to a sparser coherence. The first types of solutions [19, 51] assume that each entity has strong semantic relatedness with **at least one** of the other entities within the same document. However, isolated concepts still cannot be recognized. The second solution employs reinforcement learning [45, 62] to learn the relatedness between a given entity and the dynamic context of the document. However, they assume that the phrases are given as input, and ignore the relation linking task. Therefore, they are not applicable to the end-to-end joint entity and relation linking scenario. How to apply a robust and unsupervised method to recognize the isolated candidate entities (or predicates) while maintaining the sparse coherence of other linked entities (or predicates) is still an unsolved challenge.

The second challenge is **the end-to-end extraction problem.** In real-word scenario, it is common to have multiple overlapped mentions in one document, such as `Fellow`, `AAAS`, and `Fellow of the AAAS` in Figure 1. Therefore, we need to determine which ones are best to describe the semantic concepts expressed in the document. Most common entity linking systems consider mention detection (MD) and entity disambiguation (ED) as two separate tasks, without leveraging their mutual dependency [25, 52, 59]. However, wrong MD decisions can be avoided by utilizing global context coherence. For example, in Figure 1, understanding the presence of the entity *M. Jordan (professor)* helps to determine the correct mention `Fellow of the AAAS`, as opposed to separating `Fellow` and `AAAS` into less informative concepts. Another example is the document "`Mary and Max is a 2009 movie directed by Adam Elliot`". Knowing the presence of the entity *Adam Elliot (director)* [4] helps to deduce the correct mention `Mary and Max`, instead of two mentions `Mary` and `Max` which represent incorrect entities.

Recently, works that jointly conduct MD and ED have been proposed [34, 41, 48]. Nevertheless, most of these works require extensive data annotation to conduct supervised learning. Moreover, none of them studies the mutual dependency between mention detection and disambiguation to assist relation linking. How to incorporate the robust and unsupervised selection of candidate mentions into the end-to-end joint entity and relation linking task is still an unexplored issue.

In this paper, we introduce TENET, a novel Tree cover-based joint ENtity and rElaTion linking technique, to address the above challenges effectively and efficiently. Given a document and a target KB as input, we first construct a knowledge coherence graph to capture the coherence among the semantic concepts in the document. We then disambiguate the knowledge in the document. This is the first end-to-end joint entity and relation linking work that guarantees sparse coherence of the linked concepts while recognizing isolated concepts. We summarize our novel contributions as follows.

- To address the sparse coherence problem, we investigate the coherence among the concepts in the input document. We formulate a minimum cost rooted tree cover problem based on the knowledge coherence graph. We prove the NP hardness of the problem, and introduce an approximation algorithm as solution, which locates isolated concepts while preserving the sparse coherence of the linked concepts.
- To address the end-to-end extraction problem, we incorporate mention selection into the end-to-end joint entity and relation linking task. Specifically, we consider all possible spans as potential mentions, and organize the overlapped and correlated mentions as different canopies. We further propose a greedy algorithm to disambiguate the candidate entities and predicates of these mention canopies, with novel pruning strategies to enhance the efficiency.

- Extensive experiments validate the effectiveness and efficiency of TENET against the state-of-the-art methods.

In the rest of this paper, we introduce the preliminaries in Sec. 2, and define the knowledge coherence graph in Sec. 3. Sec. 4 and Sec. 5 formulate the joint entity and relation linking problem and propose approximation algorithms as solutions. We show the experimental results in Sec. 6 , discuss the related works in Sec. 7, and conclude in Sec. 8.

## 2 PRELIMINARIES

We introduce the preliminaries used in this paper as follows.

DEFINITION 1. ***Knowledge Base (KB).*** *A KB $\Psi$ is a collection of facts which represent real-word concepts and events. A fact is stored as a triple (subject, predicate, object). In the KB $\Psi$, we denote the set of entities as $\mathcal{E}_\Psi$, the set of predicates as $\mathcal{P}_\Psi$, and the collection of literals as $\mathcal{L}_\Psi$. In a triple, we have subject $\in \mathcal{E}_\Psi$, predicate $\in \mathcal{P}_\Psi$, and object $\in \mathcal{E}_\Psi \cup \mathcal{L}_\Psi$.*

DEFINITION 2. ***Noun Phrases and Candidate Entities.*** *A document $d$ contains a set of noun phrases $\mathcal{N} = \{n_1, n_2, ...\}$. For each noun phrases $n_i$, there are a set of candidate entities $\{e_i^1, e_i^2, ...\} \subset \mathcal{E}_\Psi$ from the target KB $\Psi$.*

For example, in Figure 1, we have noun phrase $n_1$: `Michael Jordan`, while its candidate entities include $e_1^1$: *M. Jordan (professor)* and $e_1^2$: *M. Jordan (basketball player)*.

DEFINITION 3. ***Relational Phrases and Candidate Predicates.*** *A document $d$ contains a set of relational phrase $\mathcal{R} = \{r_1, r_2, ...\}$. For each relational phrase $r_j$, there are a set of candidate predicates $\{p_j^1, p_j^2, ...\} \subset \mathcal{P}_\Psi$ from KB $\Psi$.*

For example, in Figure 1, we have relational phrase $r_1$: `studies`, while its candidate predicates include $p_1^1$: *be educated at* and $p_1^2$: *field of study*.

We then formally present the problem that we aim to solve.

PROBLEM 1. ***Joint Entity and Relation Linking Problem.*** *Given a document $d$ and a target KB $\Psi$, our goal is to (1) extract a set of noun phrases $\mathcal{N}$ and a set of relational phrases $\mathcal{R}$, and (2) determine a set of noun phrases $\mathcal{N}^\star \subseteq \mathcal{N}$, while each $n_i \in \mathcal{N}^\star$ is linked to a correct entity $e_i \in \mathcal{E}_\Psi$, as well as a set of relational phrases $\mathcal{R}^\star \subseteq \mathcal{R}$, while each $r_j \in \mathcal{R}^\star$ is linked to a correct predicate $p_j \in \mathcal{P}_\Psi$. The linking results describe the semantic concepts expressed in document $d$ correctly.*

Take the scenario in Figure 1 as an example. The white solid frames denote the noun phrases $\mathcal{N}^\star$ and relational phrases $\mathcal{R}^\star$ as mentions selected correctly, while the gray solid frames denote the correct linking of entities and predicates.

## 3 KNOWLEDGE COHERENCE GRAPH

This section discusses the construction of a knowledge coherence graph given the input document $d$.

**Step 1. Generate a Set of Noun Phrases and the Candidate Entities.** Given an input document $d$, we employ a pipeline of linguistic tools [5, 6, 39] to extract a set of noun phrases $\mathcal{N}$. Each noun phrase $n_i \in \mathcal{N}$ is associated with a type $t(n_i)$ produced by the linguistic tools. Following [38], given a KB $\Psi$, for each noun phrase $n_i \in \mathcal{N}$, we generate a set of candidate entities $\{e_i^1, e_i^2, ...\}$ where each of them (1) has $n_i$ as one of its aliases, and (2) matches the type information of $n_i$. For simplicity, $\mathcal{E}$ denotes the set of all the candidate entities of all the noun phrases $\mathcal{N}$ from document $d$. Note that $\mathcal{E} \subset \mathcal{E}_\Psi$.

**Step 2. Generate a Set of Relational Phrases and the Candidate Predicates.** Given an input document $d$, we employ a pipeline of linguistic tools [39] and Open Information Extraction tools [27] to generate a set of relational phrases $\mathcal{R}$. Similar to Step 1, given a KB $\Psi$, for each relational phrase $r_j \in \mathcal{R}$, we generate a set of candidate predicates $\{p_j^1, p_j^2, ...\}$ where each of them has $r_j$ as one of its aliases. For simplicity, $\mathcal{P}$ denotes the set of all the candidate predicates of all the relations phrases $\mathcal{R}$ from document $d$. Note that $\mathcal{P} \subset \mathcal{P}_\Psi$.

**Step 3. Construct a Knowledge Coherence Graph.** The knowledge coherence graph aims to capture the semantic distance among the concepts in the document $d$.

DEFINITION 4. ***Knowledge Coherence Graph.*** *Given a set of noun phrases $\mathcal{N}$ with the candidate entities $\mathcal{E}$, and a set of relational phrases $\mathcal{R}$ with the candidate predicates $\mathcal{P}$ from the input document $d$, the knowledge coherence graph is a weighted undirected graph $G = (V, E)$. Specifically, $V = \mathcal{N} \cup \mathcal{E} \cup \mathcal{R} \cup \mathcal{P}$, $E$ denotes the edges between every pair of nodes in $V$, which are weighted by the semantic distance between the nodes.*

We discuss the details of computing the weights of the edges in the knowledge coherence graph $G = (V, E)$ as follows.

**Edges Between Noun Phrases and Entities $d(n_i, e_i^x)$.** An edge connecting a noun phrases $n_i$ and its candidate entity $e_i^x$ is weighted by a score of *local semantic distance* $d(n_i, e_i^x)$. Following the current knowledge disambiguation approaches [38, 61], we estimate the prior matching probability that entity $e_i^x$ is the correct linking of the noun phrase $n_i$ as $P(e_i^x|n_i)$. As shown in Equation 1, a high matching probability implies a small local semantic distance.

$$d(n_i, e_i^x) = 1 - P(e_i^x|n_i) \tag{1}$$

For example, if the noun phrase $n_1$ : `Michael Jordan` refers to the candidate entity $e_1^2$: *M.Jordan (basketball player)* in 70% of its occurrences, then we have $P(e_1^2|n_1) = 0.7$. Therefore, $d(n_1, e_1^2) = 1 - 0.7 = 0.3$.

**Edges Between Relational Phrases and Predicates $d(r_i, p_i^x)$.** As shown in Equation 2, an edge between a relational phrases $r_i$ and its candidate predicate $p_i^x$ is weighted by a score of local semantic distance $d(r_i, p_i^x)$, which also relies on the matching probability that $p_i^x$ is the correct linking of $r_i$.

$$d(r_i, p_i^x) = 1 - P(p_i^x|r_i) \tag{2}$$

**Edges Between Entity Pairs $d(e_i^x, e_j^y)$.** There is an edge between the entities $e_i^x$ and $e_j^y$ only if their related noun phrases are different, i.e., $i \neq j$. For example, in Figure 1, *M. Jordan (professor)* is not linked to *M. Jordan (basketball player)*, since both entities are connected to the same noun phrase `Michael Jordan`. The edge between a pair of entities $e_i^x$ and $e_j^y$ is weighted by a score of *global semantic distance* $d(e_i^x, e_j^y)$.

Following the current knowledge disambiguation approaches [49, 51, 63], we employ the cosine similarity between the embedded representations of $e_i^x$ and $e_j^y$ as their pairwise relatedness, and further deduce $d(e_i^x, e_j^y)$ in Equation 3. A stronger relatedness implies a smaller global semantic distance.

$$d(e_i^x, e_j^y) = 1 - cos(embedding(e_i^x), embedding(e_j^y)) \quad (3)$$

For example, if the cosine similarity between the embedded representations of entity $e_1^1$: *M.Jordan (professor)* and $e_2^1$: *AI (CS topic)* is 0.35, then we have $d(e_1^1, e_2^1) = 1 - 0.35 = 0.65$.

**Edges Between Predicate Pairs $d(p_i^x, p_j^y)$.** There is an edge between the predicates $p_i^x$ and $p_j^y$ only if (1) their related relational phrases $r_i$ and $r_j$ are different, i.e., $i \neq j$, and (2) the related relational phrases $r_i$ and $r_j$ are in the same sentence. The reason is that in most natural language context, relational phrases in different sentences have very little semantic relatedness. For example, in Figure 1, the relational phrases `studies` and `visited` are in different sentences, while the predicates represented by them are not related.

Equation 4 computes their global semantic distance in a similar manner as the one of entities.

$$d(p_i^x, p_j^y) = 1 - cos(embedding(p_i^x), embedding(p_j^y)) \quad (4)$$

**Edges Between Entities and Predicates $d(e_i^x, p_j^y)$.** An edge between an entity $e_i^x$ and a predicate $p_j^y$ only exists when the corresponding noun phrase $n_i$ and the relational phrase $r_j$ are in the same sentence. The reason is that in most natural language contexts, noun phrases and relational phrases in different sentences do not share strong relatedness. For example, in Figure 1, the noun phrase `artificial intelligence` and the relational phrase `visited` are in different sentences, where there is no strong relatedness between the entity and predicate represented by both phrases.

Equation 5 computes their global semantic distance.

$$d(e_i^x, p_j^y) = 1 - cos(embedding(e_i^x), embedding(p_j^y)) \quad (5)$$

# 4 COHERENCE TREE COVER DERIVATION

In this section, we determine the related nodes for each noun phrase and each relational phrase. The input is a knowledge coherence graph $G = (V, E)$ constructed in Sec. 3, while $V = \mathcal{N} \cup \mathcal{E} \cup \mathcal{R} \cup \mathcal{P}$. For each $n_i \in \mathcal{N}$, we derive a set of nodes from $V$ which are coherent to $n_i$. Similarly, for each $r_j \in \mathcal{P}$, we derive a set of nodes from $V$ which are coherent to $r_j$.

## 4.1 The Coherence Tree Cover Problem

In this subsection, we introduce a novel *rooted tree cover*-based objective to effectively model our goal to select the related nodes of each noun phrase and each relational phrase. First, to simplify the discussion, we represent the nodes in the semantic coherence graph as follows.

Definition 5. ***Mention Set ($\mathcal{M}$) and Concept Set ($\mathcal{C}$).*** *We refer to the union of noun phrases $\mathcal{N}$ and relational phrases $\mathcal{R}$ as set of mentions $\mathcal{M}$, i.e., $\mathcal{M} = \mathcal{N} \cup \mathcal{R}$, the union of entities $\mathcal{E}$ and predicates $\mathcal{P}$ as set of concepts $\mathcal{C}$, i.e., $\mathcal{C} = \mathcal{E} \cup \mathcal{P}$. Specifically, for each mention $m_i \in \mathcal{M}$, the related set of concepts are denoted as $\{c_i^1, c_i^2, ...\}$.*

For each mention $m_i \in \mathcal{M}$, we aim to discover all the other concepts or mentions that are related to mention $m_i$. We present the formal definition of the rooted tree cover that manipulates our goal as follows.

Definition 6. ***$\mathcal{M}$-Rooted Coherence Tree Cover.*** *Given a knowledge coherence graph $G = (V, E)$ and a set of mentions $\mathcal{M} \subset V$, an $\mathcal{M}$-rooted coherence tree cover is a set of trees $\mathcal{T} = \{T_1, T_2, ..., T_{|\mathcal{M}|}\}$. Such a coherence tree cover satisfies the following conditions in our problem:*

- *Each tree $T_i$ contains a distinct mention $m_i \in \mathcal{M}$ as root;*
- *Every node $v \in V$ appears in at least one tree $T_i$, i.e., $V = \bigcup_{i=1}^{|\mathcal{M}|} V(T_i)$;*
- *The trees in $\mathcal{T}$ can share nodes and edges. The root of $T_i$, i.e., the mention $m_i$, may be in $T_j$ rooted at mention $m_j$.*

*Specifically, the cost of such a tree cover is defined as follows.*

- *The weight of a tree $T_i$ is defined as the sum of the weights of all the edges in $T_i$, i.e., $\omega(T_i) = \sum_{(v_i, v_j) \in T_i} d(v_i, v_j)$. Note that $d(v_i, v_j)$ denotes the semantic distance between nodes $v_i$ and $v_j$, which is discussed in Sec. 3.*
- *The cost of a tree cover $\mathcal{T}$ is defined as the weight of the tree with the maximum weight in $\mathcal{T}$, i.e., $\omega(\mathcal{T}) = \max_{T_i \in \mathcal{T}} \omega(T_i)$.*

We then formally introduce our goal, which is to determine an $\mathcal{M}$-rooted tree cover with the minimum cost.

Problem 2. ***The Minimum-Cost $\mathcal{M}$-Rooted Coherence Tree Cover Problem.*** *Given a knowledge coherence graph $G = (V, E)$ and a set of mentions $\mathcal{M} \subset V$, we aim to determine the $\mathcal{M}$-rooted coherence tree cover of $G$ with the minimum cost, denoted as $\mathcal{T}^\star$.*

For example, Figure 1 presents a satisfactory tree cover that meets our goal. There are 5 disjoint trees. $\{n_5, e_5^1\}$, $\{n_6, e_6^1, e_6^2\}$, $\{n_7, e_7^1\}$ and $\{r_3, p_3^1\}$ are 4 disjoint trees, while each of them is rooted at $n_5$, $n_6$, $n_7$ and $r_3$. The trees that rooted at all the other mentions, i.e., $n_1$, $n_2$, $n_3$, $n_4$, $r_1$ and $r_2$, share the same nodes and edges.

Specifically, we explain the intuition to formulate the Joint Entity and Relation Linking Problem (Problem 1) as a Minimum-Cost $\mathcal{M}$-Rooted Coherence Tree Cover Problem (Problem 2) as follows. A minimum-cost $\mathcal{M}$-rooted coherence tree cover satisfies *all* the following requirements.

- Every mention $m_i \in \mathcal{M}$ is contained in the final derived tree cover, since each tree $T_i \in \mathcal{T}^\star$ is rooted from a mention $m_i \in \mathcal{M}$. In this way, we will not exclude any mention.
- All the nodes within a small semantic distance to a specific mention $m_i$ are connected in a tree $T_i$ rooted at $m_i$. In this way, the coherence in each tree $T_i$ is guaranteed.
- The tree cover $\mathcal{T} = \{T_1, T_2, ..., T_{|\mathcal{M}|}\}$ is a union of trees that may be *disjoint*. It satisfies our target to relax the densely-connection coherence to sparse coherence, which allows the existence of isolated concepts, such as *travel to* and *Brooklyn (location)* in Figure 1.
- By setting the cost of a tree cover as the maximum weight (instead of the average weight) of the trees, we avoid the domination of some light-weight edges over the other edges. The reason is that a light-weight edge only indicates a small

semantic distance between a pair of nodes in a local view, instead of most of the nodes in a global view. For example, in Figure 1, the mention `Michael Jordan` is connected to *M. Jordan (basketball player)* by a much lighter edge than the other concepts, since *M. Jordan (basketball player)* is more popular. However, *M. Jordan (basketball player)* is not the most suitable candidate in a global view. If we employ average edge weights as the cost of tree cover, then some tree covers with a few light edges may easily dominate the other selections of tree covers where most nodes are more closely connected, and lead to unsatisfactory results.

THEOREM 4.1. *Problem 2 is NP-hard.*

PROOF. Problem 2 can be proven to be NP-hard by a reduction to the BIN-PACK problem [12, 22, 26]. We describe a BIN-PACK instance as follows. Given a set of elements $U = \{u_1, u_2, ...\}$ where which of the element $u_j \in U$ is weighted as $\omega(u_j)$, and a set of $|\mathcal{M}|$ bins while each of the bin is of size $B$. In the BIN-PACK problem, we determine whether there exists $\mathcal{U} = \{U_1, U_2, ..., U_{|\mathcal{M}|}\}$, while for each $U_i$, $\sum_{u_j \in U_i} \omega(u_j) \leq B$, so that these elements can be packed to the $|\mathcal{M}|$ bins.

We cover this BIN-PACK instance to an instance of Problem 2 as follows. First, we construct a bipartite graph with $\mathcal{M} \cup U$ as the vertex sets, where $\mathcal{M} = \{m_1, m_2, ...\}$ represents the bins. For each $m_i \in \mathcal{M}$ and each $u_j \in U$, we build an edge weighted as $\omega(m_i, u_j) = \omega(u_j)$. We then designate $\mathcal{M}$ as the set of mentions in Problem 2, and decide whether there is an $\mathcal{M}$-rooted tree cover in the bipartite graph of cost no more than $B$, which is a special case of Problem 2. Obviously, each BIN-PACK instance generates an $\mathcal{M}$-rooted tree cover of the same cost. Moreover, each $\mathcal{M}$-rooted tree cover can be transferred into a solution of a BIN-PACK instance with the same cost in polynomial time. Specifically, in each $T_i \in \mathcal{T}$ rooted at mention $m_i$, for each node $u \in T_i$, if there exists an edge $(u, m_{j|j \neq i})$, we replace it with $(u, m_i)$. In this way, each element in $u \in U$ is packed to a unique mention $m_i \in \mathcal{M}$. This induces a BIN-PACK solution with the same cost. To summarize, the BIN-PACK instance can be solved if and only if the tree cover instance can be solved, and hence we can reduce the BIN-PACK problem to Problem 2, which completes our proof. □

## 4.2 The Approximation Algorithm to Derive $\mathcal{T}^\star$

To determine a satisfactory coherence tree cover, we propose an approximation algorithm based on [22, 29]. Specifically, we design novel tree splitting and pruning strategies which are more applicable to the joint entity and relation linking task. Algorithm 1 derives the coherence tree cover, while Algorithm 2 and Algorithm 3 show the tree splitting strategies.

The inputs of Algorithm 1 are the knowledge coherence graph $G = (V, E)$ with a set of mention nodes $\mathcal{M}$ specified, and a given constraint $B$ on the weight of each tree. Our output is (1) a failure warning that $B$ is too small ($B$ is smaller than $B^*$, which is the minimal cost of the tree cover), or (2) a $\mathcal{M}$-rooted tree cover of cost at most $4B$.

**Algorithm 1:** TREECOVERDETERMINATION

**Input:** (1) $G = (V, E)$ where $V = \mathcal{M} \cup \mathcal{C}$; (2) $B$.
**Output:** (1) Failure warning: $B$ is too small, or (2) Successful case: $\mathcal{T} = \{T_1, T_2, ..., T_{|\mathcal{M}|}\}$ (an $\mathcal{M}$-rooted tree cover of $G$).

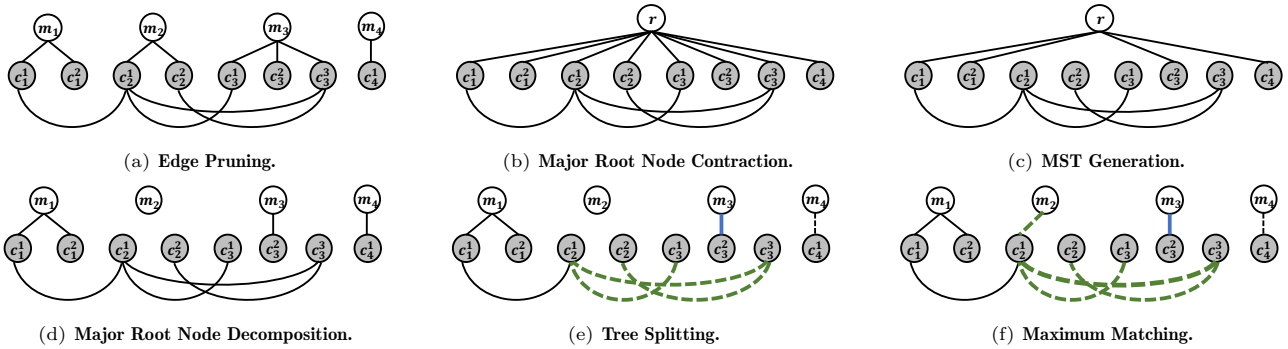
1 Prune the edges of weight greater than $B$;
2 $G' \leftarrow (r \cup \mathcal{C}, E')$; // $E'$ denotes the set of edges between $r$ and each concept node in $\mathcal{C}$.
3 $edge\_list \leftarrow sort(E')$, $MST \leftarrow \emptyset$, $index \leftarrow 0$;
4 **while** $|MST| < |\mathcal{C}|$ **do**
5  **if** $index >= len(edge\_list)$ **then**
6  **return** Failure warning; // $G'$ is not connected.
7  $index+ = 1$;
8  **if** $edge\_list.get(index)$ *will not cause cycle in* $MST$ **then**
9  $MST.add(edge\_list.get(index))$;
10 Decompose the major node $r$ in $MST$ back to $\mathcal{M} = \{m_1, m_2, ...\}$, and obtain $\{T_1, T_2, ..., T_{|\mathcal{M}|}\}$;
11 $leftover\_list \leftarrow \emptyset$, $subtree\_list \leftarrow \emptyset$, $\mathcal{T} \leftarrow \emptyset$;
12 **for** $T_i \in \{T_1, T_2, ..., T_{|\mathcal{M}|}\}$ **do**
13  $(L_i, \{S_i^j\}_j) \leftarrow$ TREESPLITTING$(T_i, B)$; // Algorithm 2.
14  $leftover\_list.add(L_i)$, $subtree\_list.add(\{S_i^j\}_j)$;
15 **if** $subtree\_list == \emptyset$ **then**
16  **return** $leftover\_list$; // $\mathcal{T} = \{L_1, L_2, ..., L_{|\mathcal{M}|}\}$
17 Construct a bipartite graph $bGraph$ with $\mathcal{M}$ as one side and $subtree\_list$ as the other side ;
18 $Mapping \leftarrow$ HOPCROFTKARP$(bGraph)$;
19 **if** $len(Mapping) < len(subtree\_list)$ **then**
20  **return** Failure warning;
21 **for** $(L_i, S_i') \in Mapping$ **do**
22  $\mathcal{T}.add(L_i, S_i'$, SHORTESTPATH$(L_i, S_i'))$;
23 **return** $\mathcal{T}$;

**Step (a): Edge Pruning (line 1).** Given the graph $G = (V, E)$, we prune the edges of weight larger than $B$. Hence, every pair of nodes with semantic distance larger than $B$ is disconnected.

**Step (b): Major Root Node Contraction (line 2).** We then contract all the mention nodes in $\mathcal{M}$ to a major root node $r$. Specifically, we (1) add a new node $r$; (2) for each mention node $m_i \in \mathcal{M}$ and every related concept nodes $c_i^x \in \mathcal{C}$, induce an edge $(r, c_i^x)$ between the new node $r$ and the concept node $c_i^x$, which is weighted as $\omega(r, c_i^x) = d(m_i, c_i^x)$; (3) remove each mention node $m_i \in \mathcal{M}$. We denote the current graph as $G'$.

**Step (c): MST Generation (lines 3-9).** Next, we compute a minimum spanning tree (MST) on the graph. To summarize, we first sort all the edges in $G'$ in non-decreasing order of their weights (line 3). After that, we generate the MST by interactively picking the smallest edges which will not form a cycle with the spanning tree so far (lines 4-9). We conduct the edge picking until we form a spanning tree with $|\mathcal{C}|$ edges. If the input $B$ is set to be too small, too many edges will be pruned in Step (a). This makes $G'$ a disconnected graph (lines 5-6) which leads to a failure warning.

**Step (d): Major Root Node Decomposition (line 10).** We decompose the major node $r$ back to $\mathcal{M} = \{m_1, m_2, ...\}$, and obtain a set of trees $\{T_1, T_2, ..., T_{|\mathcal{M}|}\}$. The edge between each mention $m_i \in \mathcal{M}$ decomposed from the major node $r$ and a concept node $c_i^x \in \mathcal{C}$ is weighted as $\omega(m_i, c_i^x) = \omega(r, c_i^x)$.



(a) **Edge Pruning.** (b) **Major Root Node Contraction.** (c) **MST Generation.**

(d) **Major Root Node Decomposition.** (e) **Tree Splitting.** (f) **Maximum Matching.**

**Figure 2: Tree splitting example.**

**Step (e): Tree Splitting (lines 11-16).** In this step, we propose to split each tree $T_i$ into a set of subtrees. The intuition is to constrain the total weight of each subtree, so as to bound the maximum semantic distance between any two nodes in a subtree. In this way, we ensure that all the nodes in a subtree are correlated. Specifically, a tree $T_i$ will be decomposed into a set of subtrees $\{S_i^j\}_j$ and a leftover tree $L_i$ by Algorithm 2 (line 13). For the weight of the subtrees and leftover trees, we have $\omega(S_i^j) \in (B, 2B]$ and $\omega(L_i) \in (0, B]$. Note that $L_i$ contains the mention node $m_i$ of the tree $T_i$, and therefore $L_i$ must exist for every $m_i$. Specifically, if there is no subtree recorded, we directly return all the leftover trees as the $\mathcal{M}$-rooted tree cover (lines 15-16).

**Step (f): Maximum Matching (lines 17-23).** In this step, we propose to connect as many subtrees produced in Step (e) as possible to each mention node, while each connection produces a tree in the final tree cover. Since maximum matching only maps the subtrees to the mentions within a pre-specified distance, this ensures that the weight of each tree in the final derived tree cover is also bounded. To summarize, Step (e) and Step (f) guarantee that all the nodes in a tree $T_i$ are correlated to the mention node $m_i$. Specifically, we first construct a bipartite graph *bGraph*, with one side as the mention nodes $m_i \in \mathcal{M}$, and the other side as all the subtrees generate in Step (e) (line 17). We then determine a maximum matching on the graph *bGraph* based on HopcroftKarp algorithm [10](line 18). Specifically, an edge exists between $m_i \in \mathcal{M}$ and a subtree if and only if their distance is in the range $(0, B]$. If the maximum matching cannot match all the subtrees, we return a failure warning (lines 19-20). Otherwise, the algorithm returns a tree cover $\mathcal{T} = \{T_1, T_2, ..., T_{|\mathcal{M}|}\}$ (lines 21-23), where each tree $T_i$ contains (1) a leftover tree $L_i$ that contains the mention $m_i$, (2) a subtree $S_i'$ (if any), and (3) a shortest path from $m_i$ to $S_i'$ (if any).

**The Tree Splitting Strategies.** Algorithm 2 and Algorithm 3 present the tree splitting in Step (e). In Algorithm 2, given a tree $T_i$ rooted with mention $m_i$ and the bound $B$, if its cost is not larger than $B$, then we return $T_i$ as the leftover tree $L_i$ (lines 1-2). We then conduct the tree splitting task (lines 3-17). Specifically, we pick each edge following the post-order traversal order (line 4), and determine the tree $T'$ rooted with the closest ancestor of the nodes in the current edge (line 7). If $T'$ contains the mention $m_i$ and the cost of $T'$ is larger than $2B$, we regard it as a heavy tree and calls for Algorithm 3 for further splitting (lines 9-11). Otherwise, $T'$ is assigned as $L_i$ directly (lines 12-13). On the other hand, if $T'$ does not contain the mention $m_i$, we continue the tree splitting (lines 15-17). Algorithm 2 returns a leftover tree $L_i$ rooted with mention $m_i$ and a set of subtrees (line 18).

Algorithm 3 employs a recursion to decompose a tree $T'$ where $\omega(T') > 2B$. This algorithm accumulates the edges following post-order traversal (lines 4-12), and once the total weight of the tree composed by the accumulated edges is larger than $B$, the tree will be split away (lines 9-12). The recursion terminates when the weight of the remaining tree is not larger than $B$ (lines 1-3). Algorithm 3 also returns a leftover tree $L_i$ along with a set of subtrees.

**Running Example.** Figure 2(a)-(f) depicts a sample semantic coherence graph after conducting Step (a)-(f) in Algorithm 1.

**Algorithm 2:** TreeSplitting

**Input:** Tree $T_i$ with the mention $m_i$ as the root, the bound $B$.
**Output:** $(L_i, \{S_i^j\}_j)$, where $\omega(L_i) \leq B$ and for each $S_i^j \in \{S_i^j\}_j$, $\omega(S_i^j) \in (B, 2B]$. Specifically, $m_i \in L_i$.

1 **if** $\omega(T_i) \leq B$ **then**
2  **return** $(T_i, \emptyset)$;
3 $stack \leftarrow \emptyset$; // *stack* records the proceeded edges.
4 $L_i \leftarrow \emptyset, Q \leftarrow \emptyset$; // $Q$ is a queue to record every $S_i^j \in \{S_i^j\}_j$.
5 **for** $edge \in$ PostOrderTraverse$(T_i)$ **do**
6  $stack.push(edge)$;
7  $T' \leftarrow$ getCompleteTree$(stack, edge)$; // Find the tree rooted with the closest ancestor of $edge$.
8  **if** $m_i \in T'$ **then**
9  **if** $\omega(T') > 2B$ **then**
10  $(L_i, Q) \leftarrow$ HeavyTreeSplitting$(T', B, L_i, Q)$;
11  Remove all the edges in $L_i$ and $Q$ from *stack*;
12  **else**
13  $L_i \leftarrow T'$;
14  **break**;
15  **if** $\omega(T') \leq 2B$ *and* $\omega(T') > B$ **then**
16  $Q.enqueue(T')$;
17  Remove all the edges in $T'$ from *stack*;
18 **return** $(L_i, Q)$; // $Q$ records every $S_i^j \in \{S_i^j\}_j$.

**Discussion of Algorithm Design.** To determine the minimum spanning tree, our design is more similar to Kruskal's [35]

**Table 1: Sample documents with the mention groups and canopies.**

| Documents (with the short-text mentions underlined) | Mention Groups | Canopies |
|---|---|---|
| Rembrandt painted The Storm on the Sea of Galilee. | $G_1$={Rembrandt} | $A_1^1$ = {Rembrandt} |
| | $G_2$={The Storm, Sea, Galilee} | $A_2^1$ = {The Storm, Sea, Galilee}<br>$A_2^2$ = {The Storm on the Sea, Galilee}<br>$A_2^3$ = {The Storm, the Sea of Galilee}<br>$A_2^4$ = {The Storm on the Sea of Galilee} |

instead of Prim's [54]. The reason is that Prim's grows a tree from the root, which is not applicable in our scenario where the disambiguation is performed by the order of the semantic distance between the node pairs in the graph. In other words, we propose to enforce the less confidence choice to be consistent with the more confident decisions made previously. Hence, Kruskal's is more applicable. This strategy is also utilized by other disambiguation methods [51, 63].

**Time Complexity of Tree Splitting (Algorithm 2 and Algorithm 3).** Given a tree $T_i$ to be split, It takes $O(|V|)$ to conduct the post-order traversal, and $O(|E|)$ to enumerate the edges in the post order traversal order to decide whether tree splitting operation is necessary for each subtree with the help of a stack. Therefore, it requires $O(|V| + |E|)$ time to conduct tree splitting for each tree $T_i$.

**Time Complexity of Tree Cover Determination (Algorithm 1).** Given a semantic coherence graph $G = (V, E)$, Algorithm 1 takes $O(|E|)$ for edge pruning (line 1), $O(|E|)$ for major root node contraction (line 2), and $O(|E|log(|E|))$ for edge sorting (line 3). To construct the minimum spanning tree, the iteration through all edges and the union of edges without forming a cycle (based on Kruskal's) takes $O(|E|log(|V|))$ time (the value of $|E|$ can be at most $O(|V|^2)$, so $log(|V|)$ can be regarded to be the same as $log(|E|)$) (lines 4-10). The decomposition of edges takes $O(|E|)$ time (line 11). Since we already know that it takes $O(|V| + |E|)$ to conduct tree splitting for each tree $T_i$ after the decomposition step, the time complexity to conduct tree splitting on all the tree $T_i$ in the forest takes $O(|\mathcal{M}|(|V|+|E|))$ time (lines 12-18). Moreover, as for the maximum matching (lines 19-25), supposed that the number of subtrees in the bipartite graph is $\beta$, the number of nodes in the bipartite graph will be $\beta+|\mathcal{M}|$, and the number of edges is at most $(\beta + |\mathcal{M}|)^2$. The time complexity of this step will be $O(\beta^2\sqrt{\beta})$. To summarize, the total time complexity is $O(|E|)+O(|E|)+O(|E|log(|E|))+O(|E|log(|V|))+O(|\mathcal{M}|(|V|+|E|)) + O(\beta^2\sqrt{\beta}) \leq 2O(|E|) + 2O(|E|log(|E|)) + 2O(|V|^2) = O(|E|log(|E|) + |V|^2) \leq O(|V|^2log(|V|^2) + |V|^2) = O(|V|^2)$ in the worst case. However, the number of mention nodes $|\mathcal{M}|$ and the number of subtrees generated by tree splitting algorithms is far less than $(|V|)$. Therefore, the algorithm takes less time to finish.

Lemma 4.2. *In the successful case, Algorithm 1 derives a $\mathcal{M}$-rooted tree cover of cost bounded by $4B$.*

Proof. Each tree $T_i \in \mathcal{T}$ contains three major parts (as shown in line 25 of Algorithm 1): (1) a leftover tree $L_i$ that contains the mention $m_i$, where $\omega(L_i) \in (0, B]$, (2) a subtree $S_i^k$ (if any) where $\omega(S_i^k) \in (B, 2B]$, and (3) a shortest path from $m_i$ to $S_i^k$ (if any) which is constrained by $B$. Therefore, the weight of each tree $T_i$ is constrained by $4B$. □

**Algorithm 3:** HeavyTreeSplitting

**Input:** The tree $T'$ to be splitted, the bound $B$, $L_i$ to record the leftover tree, and *queue* to record the subtrees.
**Output:** $(L_i, Q)$, where $\omega(L_i) \leq B$, and for each $S_i^j$ in $Q$, $\omega(S_i^j) \in (B, 2B]$.

```
1  if ω(T') ≤ B then
2      L_i ← T';
3      return (L_i, Q);
4  stack ← ∅;
5  for edge ∈ PostOrderTraverse(T') do
6      T'_new ← getCompleteTree(stack, edge);
7      if ω(T'_new) ≤ B then
8          stack.push(edge);
9      else
10         Q ← a new tree containing all the edges in stack;
11         T'_next ← T' removes all the edges in stack ;
12         (L_i, Q) ← HeavyTreeSplitting(T'_next, B, L_i, Q);
13 return (L_i, Q);
```

## 5 KNOWLEDGE DISAMBIGUATION

In this section, we introduce the knowledge disambiguation process to address the joint entity and relation linking problem. Given the coherence tree cover $\mathcal{T} = \{T_1, T_2, ..., T_{|\mathcal{M}|}\}$ obtained in Sec. 4 as input, where each $T_i \in \mathcal{T}$ contains all the nodes related to mention $m_i \in \mathcal{M}$, our major goal is to select a subset of mentions $\mathcal{M}^\star \in \mathcal{M}$, and produce a mapping $\mathcal{M}^\star \mapsto \mathcal{C}$, such that every mention $m_i' \in \mathcal{M}^\star$ is linked to a correct concept $c_i' \in \mathcal{C}^\star$ in the target KB.

### 5.1 Generation of Mention Groups and Canopies

In this subsection, we discuss the manipulation of the *overlapped mentions* in the document. The mentions `Fellow of the AAAS`, `Fellow` and `AAAS` in Figure 1 are a vivid example of overlapped mentions. Our goal in the joint entity and relation linking task is to incorporate the mention selection task into the decision of satisfactory linking to preserve sparse coherence. Hence, it is necessary to design selection strategies based on different groups of mentions.

**Step 1. Select the short-text mentions.** Given a list of mentions $\mathcal{M} = \{m_1, m_2, ...\}$, we first employ pre-specified linguistic features [48] to select a set of *short-text mentions* $\hat{\mathcal{M}}$ from $\mathcal{M}$. Specifically, the sample linguistic features include:

- Coordinating conjunctions between noun phrases, e.g., `Romeo and Juliet`, `Lord of the Ring`;
- Preposition or subordinating conjunction between noun phrases, e.g., `Storm on the Island`;
- Numbers between noun phrases, e.g., `Apollo 11 mission`;

- Punctuation marks between noun phrases, e.g., `Jurassic World: Fallen Kingdom`.

We then give the formal definition of short-term mentions.

DEFINITION 7. ***Short-text mentions ($\hat{m}$).*** *Given a set of mentions $\mathcal{M}$ and a set of pre-specified linguistic features $\mathcal{F}$, a short-text mention $\hat{m} \in \mathcal{M}$ denotes a mention that does not contain any linguistic feature $f \in \mathcal{F}$.*

For example, in Table 1, given the sample document and the sample linguistic features listed in Step 1, the extracted short-text mentions are {`The Storm, Sea, Galilee`}. Similarly, a *long-text mention* refers to a mention that contains at least one of the pre-specified linguistic features. For example, `The Storm on the Sea of Galilee` is a long-text mention.

**Step 2. Generate the mention groups.** We then partition the short-text mentions into different groups, such that the short-text mention(s) in one group shares at least one of the pre-specified linguistic features.

DEFINITION 8. ***Mention Groups ($\mathcal{G}$).*** *Given a set of short-text mentions $\hat{\mathcal{M}} = \{\hat{m_1}, \hat{m_2}, ...\}$ and a set of linguistic features $\mathcal{F}$, we partition them into different mention groups $\mathcal{G} = \{G_1, G_2, ...\}$ with the following properties:*

- *Each short-text mention $\hat{m_i}$ belongs to only one mention group, i.e., $G_j \cap G_k = \emptyset$.*
- *In a mention group $G_j = \{\hat{m_1}, \hat{m_2}, ...\}$, $\hat{m_i}$ is connect to $\hat{m}_{i+1}$ (if any) by a linguistic feature $f \in \mathcal{F}$.*

Take the scenario given in Table 1 as an example. There are two mention groups generated for the first document, i.e., $G_1$ = {`Rembrandt`} and $G_2$ = {`The Storm, Sea, Galilee`}.

**Step 3. Generate the mention canopies.** Given a mention group $G_j$ and a set of linguistic features $\mathcal{F}$, we then produce a set of canopies $\mathcal{A}_j = \{A_j^1, A_j^2, ...\}$. Here is the formal definition.

DEFINITION 9. ***Canopies ($\mathcal{A}$).*** *Given a mention group $G_j$ and a set of linguistic features $\mathcal{F}$, we produce different canopies $\mathcal{A}_j = \{A_j^1, A_j^2, ...\}$ which hold the following properties:*

- *Each short-text mention $\hat{m} \in G_j$ is assigned to every canopy $A_j^k \in \mathcal{A}_j$.*
- *Each canopy $A_j^k \in \mathcal{A}_j$ contains different combinations of the short-text mentions in $G_j$, which generate different long-text mentions based on the set of linguistic features $\mathcal{F}$.*

For example, in Table 1, for the mention group $G_2$ = {`Storm, Sea, Galilee`} of the second document, there are four distinct canopies.

We present the generation of mention groups and canopies in Algorithm 4, where we employ a queue (line 2) to manipulate the generation. We iteratively proceed the mentions in $\hat{\mathcal{M}}$ according to the sequence they appear in the document (lines 3-9). We utilize a recursion (lines 13-20) to generate the canopies for each mention group $G_j$ (lines 10-11).

## 5.2 Disambiguation

In this section, we discuss the disambiguation of entities and predicates, based on the tree cover $\mathcal{T}$ we obtained for the knowledge coherence graph $G$ from Sec. 4, and the mention canopies discussed in Sec. 5.1.

**Algorithm 4:** MENTIONGROUPSWITHCANOPIES

**Input:** A set of mentions $\mathcal{M}$ and a set of linguistic features $\mathcal{F}$.
**Output:** A set of mention groups $\mathcal{G} = \{G_1, G_2, ...\}$, with the canopy $\mathcal{A}_j$ of each mention group $G_j \in \mathcal{G}$.

```
1  Generate a set of short-text mentions M̂ from M;
2  G ← ∅; queue ← ∅; // G records the mention groups.
3  for m̂_i ∈ M̂ do
4      if i > 1 then
5          if ObtainFeature(m̂_{i-1}, m̂_i) ∉ F then
6              G.add(quque.dequeue_all());
7      queue.enqueue(m_i);
8      if i == |M̂| then
9          G.add(quque.dequeue_all());
10 for G_j ∈ G do
11     A_j ←CanopyGeneration(G_j);
12 return {G_1, G_2, ..., G_|G|},{A_1, A_2, ..., A_|G|}.;
13 Function CanopyGeneration(G'):
14     temp_set ← G';
15     for m̂_i ∈ G' do
16         front = {m̂_1, m̂_2, ..., m̂_{i-1}};
17         end = {m̂_i, m̂_{i+1}, ..., m̂_|G'|};
18         for G' ∈CanopyGeneration(end) do
19             temp_set.add(ConnectWithFeature(front, end));
20     return temp_set;
```

For one mention group $G_j$, the final disambiguation results should contain the mentions in only one canopy to avoid potential conflicts. Take Table 1 as an example. In $G_2$, we produce the linking results for one of $A_2^1$, $A_2^2$, $A_2^3$ and $A_2^4$. Here is the formal definition of the Entity and Predicate Disambiguation Problem.

PROBLEM 3. ***Entity and Predicate Disambiguation*** *The inputs of this problem include (1) an $\mathcal{M}$-rooted coherence tree cover $\mathcal{T} = \{T_1, T_2, ..., T_{|\mathcal{M}|}\}$ where each $T_i$ is rooted at the mention $m_i$, and (2) a set of mention groups $\mathcal{G} = \{G_1, G_2, ...\}$ and the related set of canopies $\mathcal{A}_j$ for each $G_j \in \mathcal{G}$. Our goal is to select a set of mention nodes $\mathcal{M}^\star$, and a concept node $c_i^x \in \mathcal{T}$ for each mention $m_i \in \mathcal{M}^\star$, which satisfies:*

- *For one mention group $G_j$, only the mentions in one canopy $A_j^k \in \mathcal{A}_j$ are selected.*
- *The type of each selected mention $m_i$ and the concept $c_i^x$ selected for $m_i$ should be constrained. In other words, if $m_i$ is a noun phrase (resp., relational phrase), then $c_i^x$ should be an entity (resp., predicate).*

We propose a greedy algorithm to solve the knowledge disambiguation problem. The major idea is to compute a *minimum spanning forest* employing the strategy of Kruskal's algorithm on the tree cover $\mathcal{T}$, which iteratively picks an edge with the smallest semantic distance to form the forest. Note that computing a minimum spanning tree on each $T_i \in \mathcal{T}$ is not applicable. The reason is that if we iteratively process each $T_i$ and select the optimal linking for $m_i$, the sequence of processing the trees will affect the final results.

**Pruning Strategies and Pseudo-Code.** In Algorithm 5, we first sort each edge in the tree cover $\mathcal{T}$ in non-decreasing

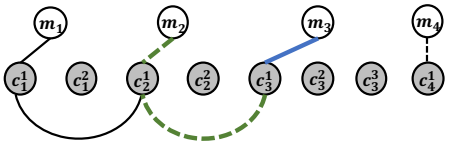



**Figure 3: The sample disambiguation results.**

order by the weight (line 1). For each mention group $G_j \in \mathcal{G}$, we maintain a mapping $\mathbb{M}$ to record the linking results of each canopy $A_j^k \in \mathcal{A}_j$ (lines 2-4). We then construct the minimum spanning forest, by iteratively picking an edge of the least possible distance (lines 5-30). Figure 3 shows a sample knowledge coherence graph after conducting the disambiguation procedure in Algorithm 5. Specifically, we follow these major pruning strategies:

- After a concept $c_i^x$ is selected for a mention $m_i$, we will not consider any other concept $c_i^{y|y\neq x}$ which is also candidate of $m_i$. This is to ensure that only one concept is selected for one mention (lines 7-18).
- If one of the concepts in an edge has a related mention already added to the final disambiguation results, then we will discard this edge. This is to ensure the correctness of edge picking (lines 13-18).
- For a mention group $G_j$, if all the mentions in a canopy $A_j^x \in \mathcal{A}_j$ have been already disambiguated, we will not consider any other mention in $A_j^{k|k\neq x}$ (other canopies for $\mathcal{A}_j$). This is to ensure that for one mention group, only the mentions in one canopy is selected (lines 20-28).
- The algorithm terminates when all the mention groups have one canopy covered in the final disambiguation result. This is to cut off the non-necessary iterations and ensure the efficiency of the algorithm (lines 29-30).

**Time Complexity.** Let $\tau$ denote the number of edges in the tree cover $\mathcal{T}$, and $\alpha$ denote the average number of canopies in the long-text phrases. Algorithm 5 takes $O(\tau log(\tau))$ for edge sorting (line 1), $O(\alpha|\mathcal{G}|)$ for organizing the mapping $\mathbb{M}$ (lines 2-4) and $O(\tau\alpha)$ for the disambiguation (lines 5-24). Therefore, the total time complexity is $O(\tau log(\tau)) + O(\alpha|\mathcal{G}|) + O(\tau\alpha) \leq O(\tau(log(\tau)+\alpha)) + O(\alpha|\mathcal{M}|) = O(\tau log(\tau)) + O(|\mathcal{M}|)$. Moreover, the pruning strategies for early stop helps the overall execution of to take less time to finish.

**Algorithm 5:** DISAMBIGUATION

**Input:** An $\mathcal{M}$-rooted coherence tree cover $\mathcal{T} = \{T_1, T_2, ..., T_{|\mathcal{M}|}\}$, a set of mention groups $\mathcal{G} = \{G_1, G_2, ...\}$ and the related set of canopies $\mathcal{A}_j = \{A_j^1, A_j^2, ...\}$ for each $G_j \in \mathcal{G}$.
**Output:** The dismbiguation result $\Gamma = \{m_1 : c_1, ..., m_i : c_i\}$.

1 $\mathcal{T}' \leftarrow sort(\mathcal{T})$, $\Gamma \leftarrow \emptyset$, $\mathcal{G}' \leftarrow \mathcal{G}$;
2 **for** $G_j \in \mathcal{G}$ **do**
3  **for** $A_j^k \in \mathcal{A}_j$ **do**
4  $\mathbb{M}_{G_j, A_j^k} \leftarrow \emptyset$;
5 **for** $(u, v) \in \mathcal{T}'$ **do**
6  $Q \leftarrow \emptyset$;// $Q$ is the queue to record the $(m_i, c_i^x)$ pairs.
7  **if** $u \in \mathcal{M}$ **then**
8  **if** $u \notin \Gamma.keys()$ **then**
9  $Q.enqueue((u, v))$;
10  **else**
11  **continue**;
12  **else**
13  **if** $\mathcal{M}(u) \notin \Gamma.keys() \wedge \mathcal{M}(v) \notin \Gamma.keys()$ **then**
14  $Q.enqueue((\mathcal{M}(u), u))$, $Q.enqueue((\mathcal{M}(v), v))$;
15  **if** $u \in \Gamma.values() \wedge \mathcal{M}(v) \notin \Gamma.keys()$ **then**
16  $Q.enqueue((\mathcal{M}(v), v))$;
17  **if** $v \in \Gamma.values() \wedge \mathcal{M}(u) \notin \Gamma.keys()$ **then**
18  $Q.enqueue((\mathcal{M}(u), u))$;
19  **for** $(m, c) \in Q$ **do**
20  **if** $G(m) \notin \mathcal{G}'$ **then**
21  **continue**;
22  **for** $A' \in G(m)$ **do**
23  **if** $m \in A'$ **then**
24  $\mathbb{M}_{G(m),A'}.add((m, c))$;
25  **if** $len(A') == len(\mathbb{M}_{G(m),A'})$ **then**
26  Update $\Gamma$ with all pairs in $\mathbb{M}_{G(m),A'}$;
27  $\mathcal{G}'.remove(G(m))$;
28  **break**;
29  **if** $\mathcal{G}' == \emptyset$ **then**
30  **break**;
31 **return** $\Gamma$;

## 6 EXPERIMENTS

In this section, we compare the performance of TENET on the joint entity and relation linking task against the state-of-the-arts methods. We also investigate the robustness and efficiency of TENET. All the experiments are conducted on a server with 128GB RAM, 2.4GHz CPU, with Ubuntu 18.04.1 LTS installed.

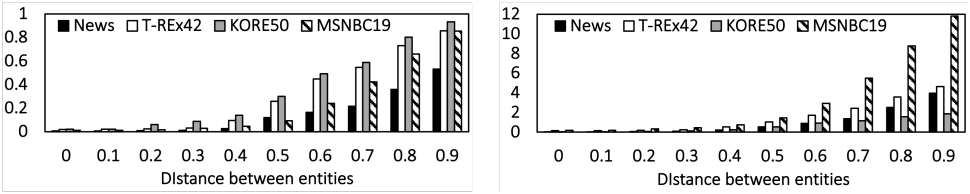
### 6.1 Experimental Settings

**Implementation Details of TENET.** We use the 2021-02-08 version of Wikidata dump, which contains 91,981,935 entities and predicates, as the target KB. We set the weight constraint of the cost of each tree $B$ to be $|\mathcal{M}|$ in each document. We list the details of the pre-processing tasks in TENET as follows.

- **Recognizing the Noun Phrases.** We employ a pipeline of linguistic tools, including the NLTK toolkit [39], the spaCy library [5], and the TAGME project [6, 24] for Named Entity Recognition (NER) and Entity Typing. We also utilize the linguistic features proposed in [48] to generate complex candidate named entity mentions, i.e., long-text phrases (Sec. 5.1). We further conduct a canonicalization for all the noun phrases based on their co-references [13].
- **Recognizing the Relational Phrases.** We employ an Open Information Extraction tool, MinIE (*safe* mode) [27] to extract the relational phrases, and conduct lemmatization on them by employing NLTK toolkit [39].
- **Indexing the Candidate Entities and Predicates.** To facilitate the fast query of candidates and predicates of a given phrase, we build index on the JSON dump of Wikidata. Following the Opentapioca project [17] and KBPearl [38], we add the labels and aliases of all the entities and predicates to a case-insensitive index via Solr (Lucene) [7].
- **Generating Embeddings for the Entities and Predicates.** Following the PyTorch-BigGraph project [36], we produce the embeddings of the entities and predicates. Specifically, we use all the distinct strings that appeared as either source

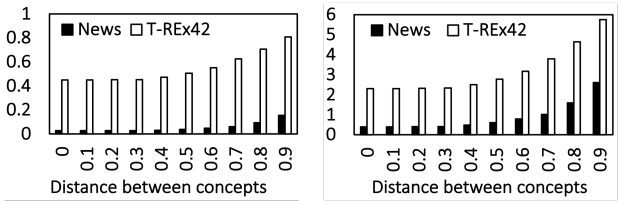
Table 2: Statistical analysis of the non-linkable noun phrases (n.) and relational phrases (re.) in all the datasets.

| Dataset | # of n./document | # of n. | # of non-linkable n. | % of non-linkable n. | # of re./document | # of re. | # of non-linkable re. | % of non-linkable re. |
|---|---|---|---|---|---|---|---|---|
| **News** | 7.69 | 138 | 29 | 21.01% | 4.75 | 76 | 48 | 63.16% |
| **KORE50** | 2.96 | 148 | 1 | 0.68% | N.A. | N.A. | N.A. | N.A. |
| **MSNBC19** | 22.32 | 424 | 64 | 15.09% | N.A. | N.A. | N.A. | N.A. |
| **T-REx42** | 7.79 | 327 | 24 | 7.34% | 5.17 | 217 | 98 | 45.16% |




(a) **Density of the entities in one document.** (b) **Average degree of the entities in one document.**

**Figure 4: Sparsity of Entities.**




(a) **Density of concepts/doc.** (b) **Average degree of concepts/doc.**

**Figure 5: Sparsity of Concepts.**

or target nodes in Wikidata dump in the RDF NTriples format as entities, and all the distinct strings that appeared as properties as relation types. We store the pre-trained embeddings in a memory-mapped array, so obtaining the embeddings from memory during the execution is fast.

**Baselines.** We have the following methods as baselines.

- **Falcon [56] (without coherence assumption)** investigates the joint entity and relation linking task. We adopt all the default configurations in its live API [8].
- **EARL [19] (with relaxed coherence)** performs joint entity linking and relation linking. We adopt all the default configurations in its live API [9].
- **KBPearl [38] (with relaxed coherence)** conducts joint entity and relation linking. We obtain the code of KBPearl (near-neighbour mode, MinIE based) from the authors with all configurations suggested in the paper.
- **MINTREE [51] (with relaxed coherence)** conducts entity disambiguation by taking both extracted mentions and documents as input. We apply the knowledge coherence graph generation part of TENET (discussed in Sec. 3) to help the extraction of MINTREE in our experiments.
- **QKBfly [46] (with global coherence)** studies Open triple canonicalization and linking in an on-the-fly manner. We adopt all the default configurations suggested in the paper. Specifically, QKBfly canonicalizes the relation phrases based on PATTY [45] without linking them to predicates. Hence, we do not compare with QKBfly in relation linking.

**Datasets.** We conduct experiments on four real-world datasets.

- **News [38] (long-text, 170.88 words/document)** contains news articles collected from NewYorkTimes in 2018. We pick 10 documents from normal domains and 6 documents in advertisement domain (with more fresh phrases) with facts annotated on Wikidata and DBpedia as ground truths.
- **T-REx42 [21] (long-text, 179.17 words/document)** is a benchmark dataset which evaluates KB population, relation extraction, and question answering. We select and further annotate 42 documents as ground truths.
- **KORE50 [31] (short-text, 12.84 words/document)** contains 50 hand-crafted sentences with a large number of very ambiguous mentions for the evaluation of entity linking.
- **MSNBC19 [15] (long-text, 562.00 words/document)** was constructed from news and an automatic disambiguation named entities process. Since the original MSNBC dataset only provides part of the linkable noun phrases as ground truths for entity linking, we manually label more noun phrases based on Wikidata and DBpedia in 19 documents.

We also conduct a statistical analysis of the percentage of the *non-linkable mentions* in the documents of all these datasets. We manually label the noun phrases and relational phrases that cannot be linked to either DBpedia or Wikidata as non-linkable. Specifically, we only count the relational phrases that connect two noun phrases in a triple. As shown in Table 2, there are a considerable amount of noun-linkable noun phrases and relational phrases, especially in the long-text datasets such as News, MSNBC19 and T-Rex42.

We further investigate the *sparse coherence problem* in these datasets based on two metrics. **(1) The density of the concepts in each document**, which is defined as $Den(\mathcal{C}) = \frac{2*|E|}{|\mathcal{C}|*(|\mathcal{C}|-1)}$, while $\mathcal{C}$ is the set of concepts, and $E$ is the set of coherence edges that connect the concepts. **(2) The average degree of each concept in each document**, which is defined as $Avg\_degree(\mathcal{C}) = \frac{2*|E|}{|\mathcal{C}|}$. This is proposed by Phan et al. [51] to evaluate the denseness of a concept graph. Obviously, for both metrics, higher values indicate that the concepts are densely connected, while lower values reflect sparse coherence.

The analysis of sparsity under both metrics are shown in Figure 4 (for entities) and Figure 5 (for concepts). Since there exists no standard to define the value of the *semantic distance* of a pair of entities that shall be considered as *correlated and connected*, we study all the extent of semantic distance (from range 0 to 0.9). For example, when x-axis is 0.1 (distance =

**Table 3: Performance of end-to-end entity linking.**

| Systems | News | | | T-REx42 | | | KORE50 | | | MSNBC19 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | *P* | *R* | *F* | *P* | *R* | *F* | *P* | *R* | *F* | *P* | *R* | *F* |
| Falcon | 0.456 | 0.107 | 0.173 | 0.504 | 0.184 | 0.270 | 0.137 | 0.125 | 0.130 | 0.386 | 0.137 | 0.202 |
| QKBfly | **0.731** | 0.146 | 0.243 | 0.438 | 0.248 | 0.317 | 0.539 | 0.420 | 0.472 | 0.640 | 0.403 | 0.495 |
| KBPearl | 0.578 | 0.368 | 0.450 | 0.748 | 0.545 | 0.631 | 0.281 | 0.255 | 0.267 | 0.713 | 0.522 | 0.602 |
| EARL | 0.459 | 0.201 | 0.280 | 0.533 | 0.337 | 0.413 | 0.122 | 0.141 | 0.131 | 0.287 | 0.257 | 0.271 |
| MINTREE | 0.453 | 0.375 | 0.410 | 0.289 | 0.243 | 0.264 | 0.355 | 0.352 | 0.353 | 0.394 | 0.423 | 0.408 |
| TENET | 0.503 | **0.414** | **0.454** | **0.752** | **0.603** | **0.669** | **0.558** | **0.508** | **0.532** | **0.644** | **0.596** | **0.619** |

**Table 4: Performance of end-to-end relation linking.**

| Systems | News | | | T-REx42 | | |
|---|---|---|---|---|---|---|
| | *P* | *R* | *F* | *P* | *R* | *F* |
| Falcon | 0.133 | 0.154 | 0.143 | 0.132 | 0.188 | 0.155 |
| KBPearl | 0.261 | **0.443** | 0.329 | 0.243 | 0.241 | 0.242 |
| EARL | 0.136 | 0.154 | 0.145 | 0.075 | 0.156 | 0.101 |
| TENET | **0.489** | 0.373 | **0.423** | **0.452** | **0.259** | **0.330** |

0.1), we study the two metrics of the filtered concept graph where only nodes within 0.1 distance are connected. Figure 4 and Figure 5 depict the sparse coherence of the entities and concepts in both metrics. Specifically, in the long-text datasets such as MSNBC19 (over 22 entities/document), each entity is connected to less than 6 other entities with closer than 0.7 distance averagely.

**Evaluation Metrics.** Precision (P) is the percentage of the correct linkings provided by a system over all the linkings provided by the system. Recall (R) is the percentage of the correct linkings provided by a system over all the linkings in ground truths. F1-score (F) is their harmonic mean [53].

## 6.2 Performance Evaluation

**The End-to-End Entity Linking Task.** In this task, the input is a document, and the output includes (1) the noun phrases determined to be named entity mentions, and (2) the linking results of each noun phrase. This is different from most of the current entity disambiguation techniques [11, 30, 32, 47, 50, 51, 55], where the noun phrases to be disambiguated are also given as input. Therefore, the performance of the end-to-end linking systems will not be as satisfactory as those techniques, because the mention detection part affects the accuracy. Specifically, all the datasets only provide part of the linkable mentions as ground truths (e.g., mentions that represent `PERSON`, `ORGANIZATION` or `LOCATION` such as `David Beckham`), while all the systems produce linking results for more noun phrases (e.g., `economy` and `career`). Therefore, the evaluation is only conducted on the entities with which the related noun phrases are provided in the ground truths.

As shown in Table 3, compared with the systems that disambiguate each mention separately (Falcon), the systems that preserve global or partial coherence achieve better performance on F1 scores. We observe that TENET achieves the best F1 score on all the datasets. This indicates the effectiveness of coherence relaxation. Specifically, QKBfly and KBPearl obtain better precision on the News dataset, which contains more fresh concepts. The reason is that both system tend to provide less entities as the linking results, while they report more noun phrases as new concepts compared with TENET. Therefore, both system obtain relatively much higher precision than recall in this dataset.

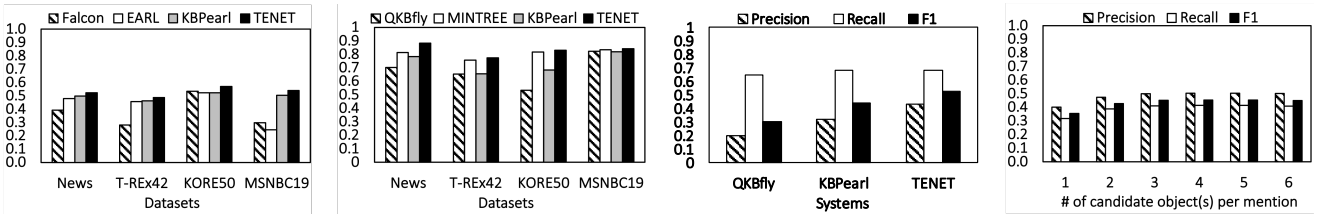
Furthermore, QKBfly, KBPearl, MINTREE, and TENET obtain better results on the short-text dataset KORE50, while TENET achieves the best performance on all the measurements. The reason is that KORE50 contains very ambiguous mentions, where the related entities need to be inferred from the context. Therefore, the techniques that emphasize the coherence of concepts in the context are more applicable.

**The Mention Detection and Entity Disambiguation Task.** We also investigate the mention detection task and the entity disambiguation task separately. The results of mention detection are illustrated in Figure 6(a). All the systems achieve satisfactory performance on the short-text dataset (KORE50). For the long-text datasets (News, T-REx42, and MSNBC19), TENET performs better than the competitors. The major reason is that long-text data have more overlapped and ambiguous mentions. The integration of the mention canopy selection part and the disambiguation part in TENET improves the accuracy of both tasks.

On the other hand, for the entity disambiguation task shown in Figure 6(b), we take the documents and noun phrases as input and evaluate the disambiguation algorithms of each system. Specifically, we do not compare with Falcon and EARL which do not have specific disambiguation parts. TENET outperforms the competitors in all long-text datasets and the ambiguous dataset (KORE50), where the disambiguation relies more on the relatedness discovery.

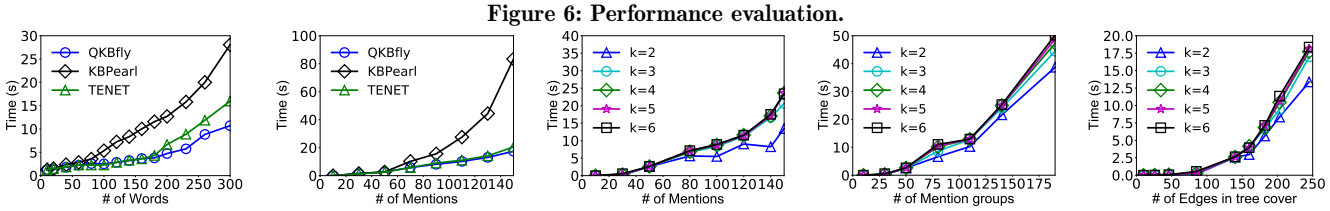
**The End-to-End Relation Linking Task.** We conduct evaluations on the 2 datasets that provide annotated facts with predicates, and compare TENET with 3 other competitors that conduct the relation linking task. As shown in Table 4, TENET outperforms other baselines in all datasets in terms of F1 score.

As shown in Table 3 and Table 4, the performance of all systems on the relation linking task is not as satisfactory as on the entity linking task. There are two major reasons. First, all the systems tend to recognize the phrases representing relations as entities. For example, given the sentence "`Rome is the sister city of Paris`", all the systems recognize `is` as a relational phrase while `sister city` as a separate noun phrase. However, the correct predicate expressed by this sentence should be *P190 (twinned administrative body)*. This indicates the importance of a rich pattern dictionary for relational phrases in the relation linking task. Second, some




(a) **F1 of mention detection.** (b) **F1 of entity disambiguation.** (c) **Isolated concept detection.** (d) **Different number of candidates.**

**Figure 6: Performance evaluation.**




(a) **Time v. words (all).** (b) **Time v. $|\mathcal{M}|$ (all).** (c) **Time v. $|\mathcal{M}|$ (TENET).** (d) **Time v. groups (TENET).** (e) **Time v. tree size (TENET).**

**Figure 7: Efficiency evaluation.**

hidden relations are ambiguous and difficult to be identified. For example, given the sentence "`More than 30 billion people live in Poland`", both KBPearl and TENET recognize the relational phrase `live in` and link it to the predicate *P551 (residence in)*, while the correct predicate should be *P1082 (population)*. Therefore, the precision is affected.

**Detection of Isolated Concepts.** We select 6 advertisement articles from the News dataset with more non-linkable noun phrases, to validate the performance of QKBfly, KBPearl, and TENET on the detection of isolated concepts. Figure 6(c) shows that TENET achieves better precision. There are two major reasons. First, both QKBfly and KBPearl rely on Open IE tools for noun phrases extraction, and hence less informative noun phrases are generated. Second, preserving global coherence hurts the precision of the cases with isolated concepts. TENET reduces the error rate by both mention canopy selection and global assumption relaxation.

**Parameter Sensitivity.** We study the setting of the average number of candidate objects per mention in TENET on the News dataset in Figure 6(d). The best performance is achieved with 3-4 candidate objects per mention, because less candidates cannot provide sufficient hints to coherence learning while more candidates will involve noises.

**Evaluation of Efficiency.** Specifically, in efficiency analysis,we do not involve Falcon and EARL for comparison, since both systems conduct on-line queries of candidate concepts and hence the computation time will be affected by network latency. We first study the execution time of TENET, OKBfly, and KBPearl in terms of (1) number of words as input, and (2) number of mentions as input. As shown in Figure 7(a) and Figure 7(b), compared with QKBfly and TENET, KBPearl is more sensitive to the length of the document and the number of mentions. There are two major reasons. First, since the semantic relatedness between each pair of concepts is independent, QKBfly and TENET pre-compute the relatedness/semantic distance between every pair of concepts while building index on them. Therefore, retrieving one edge actually costs $O(1)$ time complexity in the implementation. Moreover, during the construction of coherence graph, we retrieve the weights of the edges between each concept pair in a parallel manner based on multi-threading implementation. Therefore, the construction of coherence graph $G = (V, E)$ actually takes less than $O(|E|)$ time. Second, the pruning strategies in TENET contribute to the concept disambiguation and hence save a lot of time.

We then study the efficiency of TENET in terms of different number of candidate concepts per mention ($k$), regarding (1) different number of mentions as input, i.e., $|\mathcal{M}|$, which is the same number of the trees in the $\mathcal{M}$-rooted tree cover, (2) different number of mention groups, and (3) different size of the trees, i.e., total number of edges in the tree cover. The results are illustrated in Figure 7(c), 7(d), and 7(e). Specifically, since most mention has 3-4 disambiguates candidate objects in the KB, the running time of TENET will not be affected when $k \geq 4$. All the studies indicate that the running time of TENET is roughly linear to the amount of data processed and prove the scalability of TENET.

## 7 RELATED WORKS

**Joint Entity and Relation Linking.** Recently, a lot of systems have been proposed to handle the joint entity and relation linking task to assist two major tasks. The first task is knowledge base population, while QKBfly [46] and KBPearl [38] are the representative works. Both systems employ Open IE tools to extract canonicalized mentions from the input documents, then generate the related concepts. Nevertheless, QKBfly relies solely on the global coherence assumption. KBpearl proposes to relax the global coherence assumption by inferring the linking of each mention based on a fixed number of other mentions. However, choosing the number of attention mentions is not easy in practice. The second task is question answering, while Falcon [56, 57] and EARL [19] are the pilot works. Falcon leverages the principles of language morphology for mention extraction and concept linking, but does not consider the coherence among the concepts. EARL

exploits the connection density among the concepts, by formulating the disambiguation as a general travel salesman problem, but cannot capture isolated concepts.

**Entity Linking with Coherence.** A lot of conventional entity linking techniques [49, 55, 58, 63] assume that all entities mentioned in one document should be densely connected. MINTREE [51] is the first work to reveal that the sparse coherence among the entities in one document is not occasional. It constructs a minimum spanning tree based-objective to address the entity disambiguation problem. However, isolated concepts cannot be captured under this problem formulation. Similar to KBPearl [38], Globerson et al. [28] and Lin et al. [38] infer the linking of each mention based on a fixed number of other mentions. However, it is not trivial to choose the optimal value for this number. Moreover, DCA [62] and RLEL[23], which employs reinforcement learning models, are proposed to sequentially accumulate context information to make linking decisions. However, both methods require well-annotated training data with mappings between pre-extracted mentions and entities, and ignore the relation linking task. Therefore, they are not applicable to our problem.

**Joint Mention Detection and Entity Disambiguation.** With few exceptions, mention detection (MD) and entity disambiguation (ED) are treated separately in the vast entity linking literature. Sil et al. [59] is the first attempt to employ an MD model to assist the linking decision. However, this method relies on the performance of the mention spotter. More systems are proposed to utilize either handcrafted features [20, 40] or learned features [34, 41, 48] to conduct joint learning of entity typing, entity disambiguation, and co-reference. However, the mutual task dependency of MD and ED is not strong. Moreover, our work mainly investigates the incorporation of MD into both entity linking and relation linking as a joint task, which is different from their goal.

## 8 CONCLUSIONS

In this paper, we propose TENET, a joint entity and relation linking technique, which *relaxes the coherence assumption* in the collective entity and relation linking in an unsupervised manner. To determine the coherence among the concepts in the document, we formulate the joint entity and relation linking task as a minimum-cost coherence rooted tree cover problem. We propose approximation algorithms with pruning strategies as solutions. Extensive experiments on real-world datasets prove the effectiveness and efficiency of our method.

## ACKNOWLEDGMENTS

This work is conducted during the internship of Xueling Lin in Theory Lab, Huawei Technologies. Specifically, this work is sponsored in part by Theory Lab, Central Research Institute, 2012 Labs, Huawei Technologies Co., Ltd. This work is also partially supported by National Key Research and Development Program of China Grant No. 2018AAA0101100, the Hong Kong RGC GRF Project 16202218, CRF Project C6030-18G, C1031-18G, C5026-18G, AOE Project AoE/E-603/18, Theme-based project TRS T41-603/20R, China NSFC No. 61729201, Guangdong Basic and Applied Basic Research Foundation 2019B151530001, Hong Kong ITC ITF grants ITS/044/18FX and ITS/470/18FX, Microsoft Research Asia Collaborative Research Grant, HKUST-NAVER/LINE AI Lab, Didi-HKUST joint research lab, HKUST-Webank joint research lab grants.

## REFERENCES

[1] https://www.wikidata.org/wiki/Q3308285. [Online].
[2] https://www.wikidata.org/wiki/Property:P101. [Online].
[3] https://www.wikidata.org/wiki/Q41421. [Online].
[4] https://www.wikidata.org/wiki/Q349248. [Online].
[5] https://spacy.io. [Online].
[6] https://tagme.d4science.org/tagme. [Online].
[7] https://lucene.apache.org/solr. [Online].
[8] https://labs.tib.eu/falcon. [Online].
[9] https://github.com/AskNowQA/EARL. [Online].
[10] R. K. Ahyja, J. B. Orlin, and T. L. Magnanti. *Network flows: theory, algorithms, and applications*. Prentice-Hall, 1993.
[11] A. Alhelbawy and R. Gaizauskas. Graph ranking for collective named entity disambiguation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 75–80, 2014.
[12] E. M. Arkin, R. Hassin, and A. Levin. Approximations for minimum and min-max vehicle routing problems. *Journal of Algorithms*, 59(1):1–18, 2006.
[13] 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.
[14] J. Callan, M. Hoy, C. Yoo, and L. Zhao. Clueweb09 data set, 2009.
[15] S. Cucerzan. Large-scale named entity disambiguation based on wikipedia data. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 708–716, 2007.
[16] 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, 2013.
[17] A. Delpeuch. Opentapioca: Lightweight entity linking for wikidata. *arXiv preprint arXiv:1904.09131*, 2019.
[18] M. Dredze, P. McNamee, D. Rao, A. Gerber, and T. Finin. Entity disambiguation for knowledge base population. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 277–285. Association for Computational Linguistics, 2010.
[19] 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.
[20] G. Durrett and D. Klein. A joint model for entity analysis: Coreference, typing, and linking. *Transactions of the association for computational linguistics*, 2:477–490, 2014.
[21] 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)*, Miyazaki, Japan, May 2018. European Language Resources Association (ELRA).
[22] G. Even, N. Garg, J. Könemann, R. Ravi, and A. Sinha. Min–max tree covers of graphs. *Operations Research Letters*, 32(4):309–315, 2004.
[23] Z. Fang, Y. Cao, Q. Li, D. Zhang, Z. Zhang, and Y. Liu. Joint entity linking with deep reinforcement learning. In *The World Wide Web Conference*, pages 438–447. ACM, 2019.
[24] 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, 2010.
[25] O.-E. Ganea and T. Hofmann. Deep joint entity disambiguation with local neural attention. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2619–2629, 2017.

[26] M. R. Garey and D. S. Johnson. *Computers and intractability*, volume 174. freeman San Francisco, 1979.

[27] 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 2620–2630, 2017.

[28] A. Globerson, N. Lazic, S. Chakrabarti, A. Subramanya, M. Ringgaard, and F. Pereira. Collective entity resolution with multi-focal attention. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 621–631, 2016.

[29] B. Gorain, P. S. Mandal, and K. Mukhopadhyaya. Generalized bounded tree cover of a graph. *J. Graph Algorithms Appl.*, 21(3):265–280, 2017.

[30] Z. Guo and D. Barbosa. Robust named entity disambiguation with random walks. *Semantic Web*, 9(4):459–479, 2018.

[31] J. Hoffart, S. Seufert, D. B. Nguyen, M. Theobald, and G. Weikum. Kore: keyphrase overlap relatedness for entity disambiguation. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 545–554, 2012.

[32] 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.

[33] 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.

[34] N. Kolitsas, O.-E. Ganea, and T. Hofmann. End-to-end neural entity linking. In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, pages 519–529, 2018.

[35] J. B. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical society*, 7(1):48–50, 1956.

[36] A. Lerer, L. Wu, J. Shen, T. Lacroix, L. Wehrstedt, A. Bose, and A. Peysakhovich. Pytorch-biggraph: A large-scale graph embedding system. *arXiv preprint arXiv:1903.12287*, 2019.

[37] T. Lin, O. Etzioni, et al. Entity linking at web scale. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction*, pages 84–88. Association for Computational Linguistics, 2012.

[38] X. Lin, H. Li, H. Xin, Z. Li, and L. Chen. Kbpearl: a knowledge base population system supported by joint entity and relation linking. *Proceedings of the VLDB Endowment*, 13(7):1035–1049, 2020.

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

[40] G. Luo, X. Huang, C.-Y. Lin, and Z. Nie. Joint entity recognition and disambiguation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 879–888, 2015.

[41] P. H. Martins, Z. Marinho, and A. F. T. Martins. Joint learning of named entity recognition and entity linking. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Volume 2: Student Research Workshop*, pages 190–196. Association for Computational Linguistics, 2019.

[42] 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, 2011.

[43] D. Milne and I. H. Witten. Learning to link with wikipedia. In *Proceedings of the 17th ACM conference on Information and knowledge management*, pages 509–518, 2008.

[44] 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, 2017.

[45] 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, 2012.

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

[47] D. B. Nguyen, J. Hoffart, M. Theobald, and G. Weikum. Aida-light: High-throughput named-entity disambiguation. In *Proceedings of the Workshop on Linked Data on the Web co-located with the 23rd International World Wide Web Conference (WWW 2014)*, volume 1184 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2014.

[48] D. B. Nguyen, M. Theobald, and G. Weikum. J-nerd: joint named entity recognition and disambiguation with rich linguistic features. *Transactions of the Association for Computational Linguistics*, 4:215–229, 2016.

[49] A. Pappu, R. Blanco, Y. Mehdad, A. Stent, and K. Thadani. Lightweight multilingual entity extraction and linking. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, pages 365–374, 2017.

[50] M. Pershina, Y. He, and R. Grishman. Personalized page rank for named entity disambiguation. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 238–243, 2015.

[51] M. C. Phan, A. Sun, Y. Tay, J. Han, and C. Li. Pair-linking for collective entity disambiguation: Two could be better than all. *IEEE Transactions on Knowledge and Data Engineering*, 31(7):1383–1396, 2018.

[52] F. Piccinno and P. Ferragina. From tagme to wat: a new entity annotator. In *Proceedings of the first international workshop on Entity recognition & disambiguation*, pages 55–62, 2014.

[53] D. M. W. Powers. Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. *CoRR*, abs/2010.16061, 2020.

[54] R. C. Prim. Shortest connection networks and some generalizations. *The Bell System Technical Journal*, 36(6):1389–1401, 1957.

[55] L. Ratinov, D. Roth, D. Downey, and M. Anderson. Local and global algorithms for disambiguation to wikipedia. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 1375–1384. Association for Computational Linguistics, 2011.

[56] 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.

[57] A. Sakor, K. Singh, A. Patel, and M.-E. Vidal. Falcon 2.0: An entity and relation linking framework over wikidata. *arXiv preprint arXiv:1912.11270*, 2019.

[58] 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, 2012.

[59] A. Sil and A. Yates. Re-ranking for joint named-entity recognition and linking. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, pages 2369–2374, 2013.

[60] F. Wang, W. Wu, Z. Li, and M. Zhou. Named entity disambiguation for questions in community question answering. *Knowledge-Based Systems*, 126:68–77, 2017.

[61] I. Yamada, H. Shindo, H. Takeda, and Y. Takefuji. Joint learning of the embedding of words and entities for named entity disambiguation. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 250–259, 2016.

[62] X. Yang, X. Gu, S. Lin, S. Tang, Y. Zhuang, F. Wu, Z. Chen, G. Hu, and X. Ren. Learning dynamic context augmentation for global entity linking. 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 271–281, 2019.

[63] S. Zwicklbauer, C. Seifert, and M. Granitzer. Robust and collective entity disambiguation through semantic embeddings. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 425–434, 2016.