Context-aware Outstanding Fact Mining from Knowledge Graphs

Yueji Yang, Yuchen Li, Panagiotis Karras, Anthony K. H. Tung
National University of Singapore, Singapore Management University, Aarhus University
{yueji,atung}@comp.nus.edu.sg,yuchenli@smu.edu.sg,piekarras@gmail.com

ABSTRACT

An **O**utstanding **F**act (OF) is an attribute that makes a target entity stand out from its peers. The mining of OFs has important applications, especially in Computational Journalism, such as news promotion, fact-checking, and news story finding. However, existing approaches to OF mining: *(i)* disregard the context in which the target entity appears, hence may report facts irrelevant to that context; and *(ii)* require relational data, which are often unavailable or incomplete in many application domains.

In this paper, we introduce the novel problem of mining **C**ontext-aware **O**utstanding **F**acts (COFs) for a *target entity* under a given context specified by a *context entity*. We propose FMiner, a context-aware mining framework that leverages knowledge graphs (KGs) for COF mining. FMiner generates COFs in two steps. First, it discovers top-k relevant relationships between the target and the context entity from a KG. We propose novel optimizations and pruning techniques to expedite this operation, as this process is very expensive on large KGs due to its exponential complexity. Second, for each derived relationship, we find the attributes of the target entity that distinguish it from *peer* entities that have the same relationship with the context entity, yielding the top-l COFs. As such, the mining process is modeled as a top-(k, l) search problem. Context-awareness is ensured by relying on the *relevant* relationships with the context entity to derive peer entities for COF extraction. Consequently, FMiner can effectively navigate the search to obtain context-aware OFs by incorporating a context entity. We conduct extensive experiments, including a user study, to validate the efficiency and the effectiveness of FMiner.

CCS CONCEPTS

• **Information systems** → **Data mining**; **Collaborative and social computing systems and tools**.

KEYWORDS

Outstanding Fact Mining; Knowledge Graph

ACM Reference Format:
Yueji Yang, Yuchen Li, Panagiotis Karras, Anthony K. H. Tung. 2021. Context-aware Outstanding Fact Mining from Knowledge Graphs. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '21), August 14–18, 2021, Virtual Event, Singapore.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3447548.3467272

KDD '21, August 14–18, 2021, Virtual Event, Singapore.

ACM ISBN 978-1-4503-8332-5/21/08…$15.00
https://doi.org/10.1145/3447548.3467272

1 INTRODUCTION

Automatic **O**utstanding **F**act (OF) mining is important in a wide range of application domains, including news story discovery [2, 6], fact-checking [10, 11] and recommendation [36]. An OF is a true statement to the effect that an attribute of a *target entity* stands out in comparison to a group of *peer entities*. For instance, the news media often uses OFs as news lead to draft news stories, or to attract readers: *"Kamala Harris is the first woman to serve as vice president"*. This statement reveals an OF about the gender (*attribute*) of Harris (*target entity*) over all people (*peer entities*) who have served as United States vice president.

Existing approaches [3, 12, 23, 29, 30, 35] mainly focus on quantifying the strikingness of OFs and extracting the top ones. Whereas, these approaches are deficient when it comes to mining context-aware OFs in a cross-domain setting. Besides, most conventional approaches [3, 12, 23, 29] require the availability of *relational data* on the entities under consideration. Such data are often unavailable; even when available, they only cover basic information, lacking the richness necessary to discover OFs in context, since their acquisition, processing, and maintenance incurs prohibitive costs. A recent approach [35] mitigates this issue by leveraging open KGs. However, like conventional approaches, [35] does not support *context-aware* OF mining. This can cause inconvenience when searching for OFs in a certain context. For example, given the news of Kamala Harris becoming vice president, the previous mentioned OF serves as a good fit. Nevertheless, existing approaches may generate an OF: *"Kamala Harris is one of the only* 0.15% *Indian Americans educated at Howard University"*. This would be an interesting OF in a context focusing on Harris's education. However, in a political context involving Harris and her predecessors in office, such an education-related OF is less relevant, and may even cause disorientation.

In this paper, we propose a novel framework, FMiner, for discovering Context-aware Outstanding Facts (COFs) for a *target entity* under a context specified by a *context entity*. The specification of a context entity allows users to guide the mining process, so as to derive the desired COFs. This feature is crucial for fact-checkers and fact-finders like journalists. In a two-step process, FMiner generates the top-l COFs based on the top-k relevant relationships anchored at the context entity. In the first step, we discover the top-k relevant relationships between the target and the context entity, and also extract *peer* entities having the same relationships to the context entity. In the second step, we discover the attributes that distinguish the target from its peer entities in each relationship. The *context*-awareness of the resulting COFs follows from the use of peer entities with respect to relationships that are *relevant* to

the *context* entity. Following up on the previous example, Harris is the target entity and her predecessor in office, Mike Pence, the context entity. In the first step, FMiner generates a pattern (i.e., relationship) as in Expression 1. Then, by matching the pattern to the KG, we collect peer entities as all people who have been US vice presidents.

$$Human \xrightarrow{\text{position held}} Vice\ President \xleftarrow{\text{position held}} Pence \quad (1)$$

In the second step, the gender of Harris is extracted as the OF based on an existing strikingness measure [30, 35]. We only consider path patterns, rather than general graph patterns, because path patterns translate easily and intuitively to natural language claims.

The proposed COF mining workflow generates two chief technical challenges: First, we need to discover patterns that represent *context-relevant* relationships between the target and the context entity. For example, another pattern for Harris and Pence is shown in Expression 2.

$$Human \xrightarrow{\text{speak}} English \xleftarrow{\text{speak}} Pence \quad (2)$$

This relationship would lead to a set of peer entities consisting of all people speaking English. Consequently, both the relationship and the produced OFs are less relevant. Hence, we need to properly evaluate the patterns in terms of their relevance to the input context entity. Second, the number of different paths and patterns between two entities can be prohibitively large, as today's KGs often contain hundreds of millions of edges. Enumerating and evaluating all the patterns is of exponential complexity. In many application scenarios, such as online news feeds and live commentary, it would be unacceptable to spend hours or even minutes for COF mining.

To confront the challenges mentioned above, we devise a novel ranking model that quantifies the context-relevance of a path pattern. The intuition behind our technique is that a pattern is relevant only if the entities appearing in the corresponding matching instances are strongly related to the context entity. For example, a person speaking the same language (English) as Pence is less relevant to Pence, as compared with a person who has also been vice president. We measure the relatedness of two entities, hence the relevance of the associated patterns, via the proximity of corresponding nodes in the KG. Furthermore, to accelerate search, we adopt a bi-directional search paradigm to enumerate paths and patterns. In addition, we develop several novel optimizations to prune unqualified patterns and avoid redundant computation by reusing intermediate results. Thereafter, to extract COFs for the target entity among the collected peer entities, we adopt the strikingness measure from existing works [3, 18, 30, 35], which is already proven to be effective. We summarize our **contributions** as follows.

- We formalize the problem of mining Context-aware Outstanding Facts (COFs) from an open KG. To the best of our knowledge, this is the first work to study the mining of context-aware OFs from KGs, a task with important applications in Computational Journalism.
- We propose FMiner, a COF mining framework that discovers OFs by considering the path patterns between a target and a context entity, employing a novel relevance model to identify relevant patterns and a search algorithm with novel optimizations to mine OFs in real-time.
- In extensive experiments, including a user study, on queries from real-world news content, we demonstrate that FMiner discovers attention-seizing COFs from a large open KG with over a half billion edges in less than a second in most cases.

Organization. In Section 2, we formalize the COF mining problem and present the ranking model. Subsequently, we introduce the FMiner algorithm with novel optimizations in Section 3. We present the experimental evaluations in Section 4 and discuss related work in Section 5. Lastly, we conclude our work in Section 6. All proofs are in supplementary materials.

2 PROBLEM DEFINITION

This section starts out with some preliminaries (Section 2.1). Next, we introduce a ranking model for discovering relevant path patterns (Section 2.2), and the strikingness measure for OFs (Section 2.3). Lastly, we formalize the top-(k, l) COF problem (Section 2.4).

2.1 Preliminaries

Let $\mathcal{G}(\mathcal{V}, \mathcal{E}, \mathcal{M})$ denote a knowledge graph (KG), where $\mathcal{V}$ and $\mathcal{E}$ represent the sets of entity nodes and edges, respectively. We use v and e to denote an instance node and an edge, s.t. $v \in \mathcal{V}$ and $e \in \mathcal{E}$. Their types, or, equivalently, labels, are denoted by V_v and E_e, respectively. $\mathcal{M}$ defines a map function on instance nodes and edges: $\mathcal{M}(v) = \tilde{v}$, where $\tilde{v} \in \{v, V_v\}$ is called a node variable, and $\mathcal{M}(e) = E_e$. We may simply use $\mathcal{G}(\mathcal{V}, \mathcal{E})$ or $\mathcal{G}$ to refer to the KG. Given a target entity t and a context entity c, we assume their corresponding nodes in the KG are known; such knowledge can be gained using the query service of the KG (e.g., Wikidata Query Service[1] and Freebase Search API[2]) or by adopting existing entity linking tools [22]. In the following, we first give the formal definitions of *path pattern* and *matching instance*. Then, we introduce *context-anchored pattern* and *context-aware peer entity*.

Definition 1 (Path Pattern). *A path pattern refers to an ordered sequence of node variables and edge types. Let $P(\tilde{v_0}, E_0, \tilde{v_1}, \ldots, \tilde{v_{\ell-1}}, E_{\ell-1}, \tilde{v_\ell})$, or $P(\tilde{v_0}, \tilde{v_\ell})$ for short, to denote an ℓ-hop path pattern. E_i represents the type of the i^{th} edge where $i \in \{0, \ldots, \ell-1\}$.*

Definition 2 (Matching Instance). *A matching instance to a path pattern $P(\tilde{v_0}, \tilde{v_\ell})$ is a path instance in $\mathcal{G}(\mathcal{V}, \mathcal{E}, \mathcal{M})$, referred to as $p(v_0, e_0, v_1, \ldots, v_{\ell-1}, e_{\ell-1}, v_\ell)$ or $p(v_0, v_\ell)$, such that the following conditions are satisfied simultaneously (i.e. p is isomorphic to P).*

(1) $\forall i \in \{0, \ldots, \ell\}$, $\mathcal{M}(v_i) = \tilde{v_i}$, where $\tilde{v_i}$ is either the instance node itself v_i or its type V_i;

(2) $\forall i \in \{0, \ldots, \ell-1\}, \mathcal{M}(e_i) = E_i$.

We use $p \triangleright P$ to denote that p is a matching instance of P.

Given a pair of target and context nodes $\langle t, c \rangle$, we first identify a path instance $p(t, c)$ connecting t and c in the KG. Then, we derive a *context-anchored pattern* P which has $p(t, c)$ as a matching instance and ends at the context node c at the same time (Definition 3). The context-anchored pattern further leads to *context-aware peer entities* (Definition 4).

Definition 3 (Context-anchored Pattern). *Given the target t and the context c, a path pattern $P(\tilde{v_0}, \tilde{v_\ell})$ is called context-anchored*

[1] https://query.wikidata.org/

[2] https://developers.google.com/freebase/v1/search-overview

pattern, if and only if $\tilde{v_\ell} = c$ *and* $\exists p(t, c) \triangleright P(\tilde{v_0}, \tilde{v_\ell})$. *In particular, we denote a context-anchored pattern as* $P(\tilde{v_0}, \tilde{v_\ell} = c)$ *or simply* $P(\tilde{v_0}, c)$.

Definition 4 (Context-aware Peer Entity). *Given a pair t and c, and a context-anchored pattern* $P(\tilde{v_0}, \tilde{v_\ell} = c)$, *we define* v_0 *as a context-aware peer entity of t if* $p(v_0, c) \triangleright P(\tilde{v_0}, \tilde{v_\ell} = c)$.

Several benefits arise from admitting a context node c and confining the search to context-anchored patterns. First, this confinement helps us find *context-aware peer entities* (Definition 4), which are essential to context-aware OF mining: such entities connect to the context node c with the same relationship as the target entity t. In effect, we can compare the attributes of t with those of its peer entities under the same context, and extract the OFs for t. Second, this design choice allows for users to effectively guide the search process by merely specifying the context entity. Existing approaches mainly output OFs ranked by their strikingness scores, overlooking the relevance factor. Third, this design choice is elegant and easy to deploy in many different application scenarios. For example, apart from requiring users to specify queries, FMiner can be used to automatically process the streams of news feed or tweets, extract entities from text content, and pair such content with a relevant and attention-seizing COFs discovered from an open KG. This task would otherwise require extensive human efforts. Example 1 illustrates the above definitions.

Example 1. *Given a path instance p(Harris, VicePresident, Pence), two context-anchored patterns can be derived:* P_1*(Human, VicePresident, Pence) and* P_2*(Human, Position, Pence). Note that for simplicity, we have omitted the edge positionHeld. p is a matching instance of both* P_1 *and* P_2. *In particular,* P_2 *has turned VicePresident into its node type Position. On the other hand,* P_3*(Human, Position, Human) is not a valid context-anchored pattern because it does not end at the context node c. As a consequence, the matching instances of* P_3 *can deviate a lot from the context, leading to irrelevant OFs.*

2.2 Pattern Relevance Model

A context-anchored pattern, may still produce irrelevant peer entities and OFs. For example, Expression 2 in Section 1 is a context-anchored pattern. Nevertheless, the resulting peer entities are all people speaking the same language (i.e. English) as Pence. These people are only weakly related to Pence. This is because "English" (matching *Language* type) is a shortcut node with a high degree in the KG. The path connections via such shortcut nodes can be weak since they express less informative relationships [1, 16]. This observation inspires us to incorporate node proximity [20, 28, 31], to rank the relevance of context-anchored patterns. The intuition is that a pattern is relevant if its matching instances are relevant, and a matching instance is relevant to the extent that the matching entity nodes in the instance have high proximity scores to the context node. Thus, we ground our relevance ranking of patterns on the following node proximity measure.

Definition 5. *(Node Proximity Measure). A node proximity measure* $S(v, u)$ *returns a score that reflects the proximity of u to v in* $\mathcal{G}(\mathcal{V}, \mathcal{E})$, *normalized so that* $\sum_{u \in \mathcal{V}} S(v, u) = 1$.

We may instantiate a node proximity measure by many popular approaches, such as Personalized PageRank (PPR) [13], SimRank [15] or even weighted-shortest-path scores [1, 14]. In our implementation, we have opted for PPR, as it is widely used and there exist efficient algorithms, e.g., [8, 19, 26]. Moreover, PPR takes into consideration the structural relevance between two nodes, which makes it more robust than weighted-shortest-path scores, as mentioned in [19]. Note that node proximity measure is a plug-in component of FMiner. It is possible and easy to use any different measures in FMiner without the need to change the search algorithm. Next, we present the definitions of *path relevance score* and *matching node sets*, which lay the basis for *pattern relevance model*.

Definition 6 (Path Relevance Score). *Given the context node c, the path relevance score of* $p(v_0, v_\ell)$ *is defined as*

$$PR(p) = \min_{i \in \{0,\ldots,\ell\}} \beta^{-i} \cdot S(c, v_i),$$

where β *is a length decay factor.*

Definition 6 is based on two aspects. First, as long as there is an irrelevant node within a path, i.e., with low proximity score, then the whole path connecting via that node immediately makes a weak relationship. Second, a decay factor β is used to progressively penalize paths along their length, as the associated OFs are bound to be less interesting with long and complex relationships.

Definition 7 (Matching Node Set). *Given a context-aware path pattern* $P(\tilde{v_0}, \tilde{v_\ell} = c)$, *we denote* N_i *as the matching node set of* $\tilde{v_i}$ *where* $i \in \{0, \ldots, \ell\}$, *s.t.* $\forall u \in N_i$ *there exists a matching instance* $p(v_0, c)$ *of P with u substituting for the* i^{th} *node variable* $\tilde{v_i}$ *within P. In particular,* $N_\ell = \{c\}$ *and* N_0 *corresponds to the set of all context-aware peer entities. The proximity of* N_i *to c is defined as the average node proximity of* $\{u \in N_i\}$ *to c by* $S(c, N_i) = \frac{1}{|N_i|} \sum_{u \in N_i} S(c, u)$.

Based on the matching node sets and their proximity to the context node, we next formalize *pattern relevance model* for a context-anchored pattern regarding $\langle t, c \rangle$.

Definition 8 (Pattern Relevance Model). *Given a context-anchored pattern* $P(\tilde{v_0}, \tilde{v_\ell} = c)$, *and a length decay factor* β, *the relevance score of* $P(\tilde{v_0}, \tilde{v_\ell} = c)$ *is defined as* $\mathcal{R}(P) = \min\{HR(P), VR(P)\}$ *where* $HR(P)$ *and* $VR(P)$ *are defined as follows:*

- ***Horizontal Relevance (HR)***:
  $HR(P) = \max_{p} PR(p)$, *where* $p \triangleright P$ *starts from t and ends at c;*
- ***Vertical Relevance (VR)***:
  $VR(P) = \min_{i \in \{0,\ldots,\ell\}} \beta^{-i} \cdot S(c, N_i)$, *where* N_i *is the matching node set for* $\tilde{v_i}$ *of P, and* β *is the same as in Definition 6.*

With Definition 8, the relevance of a context-anchored pattern P is evaluated in two aspects: *(i)* in terms of HR, it invokes at least one relevant matching instance $p(t, c)$ that strongly connects the target node t to the context node c, and *(ii)* in terms of VR, its matching instances should be relevant on average, as measured by the proximity of its matching node sets, N_i to c. We use the same decay factor β as in Definition 6 to progressively penalize long matching instances. The min function ensures that every node variable associated with P is relevant, similar to Definition 6. We illustrate these concepts with an example below.

Example 2. *In Figure 1, a path pattern P(Human, position, Pence) is obtained given t=Harris and c=Pence. We have three matching node*

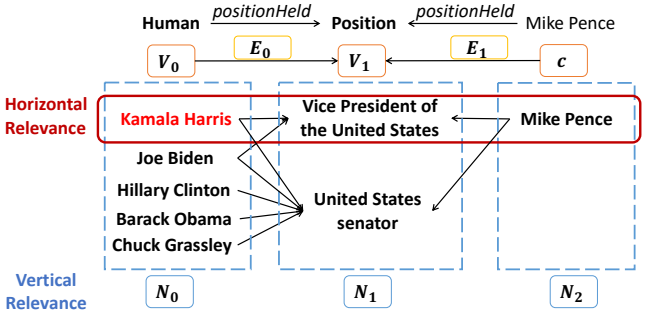

Figure 1: An example for definitions on relevance.

sets N_0, N_1 and N_2 as depicted in Figure 1. Taking N_1 as an example, its proximity to Pence is calculated as:

$$S(Pence, N_1) = \frac{1}{2}[S(Pence, VicePresident) + S(Pence, senator)].$$

The HR score is taken as the maximum PR (in red box), which is further calculated as the minimum among: $S(Pence, Harris)$, $\beta^{-1} \cdot S(Pence, VicePresident)$ and $\beta^{-2} \cdot S(Pence, Pence)$. The VR score is calculated as the minimum among: $S(Pence, N_0)$, $\beta^{-1} \cdot S(Pence, N_1)$ and $\beta^{-2} \cdot S(Pence, N_2)$.

2.3 Strikingness Measure

Given a context-anchored pattern, we can derive the context-aware peer entities. Furthermore, we extract OFs that distinguish the target entity from its peer entities. In this section, we present the definitions of *candidate OF* and *strikingness measure* for the OF extraction.

Definition 9 (Candidate OF). *A candidate OF is defined as a quadruple $Q = (t, \mathcal{A}, \mathcal{X}, P(\tilde{v_0}, \tilde{v_\ell} = c))$ with $|N_0| \geq w$, where the symbol meanings are listed as follows.*

- *t, c: The target and context nodes in $\mathcal{G}(\mathcal{V}, \mathcal{E})$.*
- *$\mathcal{A}$: An attribute of t.*
- *$\mathcal{X}$: The value for attribute $\mathcal{A}$ of t.*
- *$P(\tilde{v_0}, \tilde{v_\ell} = c)$: An ℓ-hop context-anchored pattern, which describes the relationship under consideration.*
- *w: A significance threshold to ensure that the number of peer entities is large enough for an interesting OF.*

A candidate OF comprises an attribute-value pair of the target entity t, which is striking if it makes the target stand out compared to its peer entities. To identify an OF, we need to rank the attribute-value pairs of the target against its peer entities according to a strikingness measure. We adopt a strikingness measure from [30, 35], which identifies outstanding attribute-value pairs utilizing statistics on the attribute-value frequency distribution. Given a candidate OF $Q = (t, \mathcal{A}, \mathcal{X}, P(\tilde{v_0}, \tilde{v_\ell} = c))$, $F(\mathcal{A}, \mathcal{X}', N_0)$ denotes the **frequency** (percentage) of a value $\mathcal{X}'$ among its peer entities in N_0, i.e., the matching node set of $\tilde{v_0}$.

Definition 10 (Strikingness Measure). *The strikingness score of a candidate OF $Q(t, \mathcal{A}, \mathcal{X}, P(\tilde{v_0}, \tilde{v_\ell} = c))$ is measured as below:*

$$\mathcal{I}(Q) = \sum_{\mathcal{X}' \in \overline{\mathcal{X}}} F(\mathcal{A}, \mathcal{X}', N_0),$$

where $\overline{\mathcal{X}} = \{\mathcal{X}' | F(\mathcal{A}, \mathcal{X}', N_0) > F(\mathcal{A}, \mathcal{X}, N_0)\}$.

Table 1: An example to illustrate the calculation of the strikingness score for Figure 1.

Attribute	Value	Entity	Frequency
Gender	Female	Kamala Harris	0.4
Gender	Female	Hillary Clinton	
Gender	Male	Joe Biden	0.6
Gender	Male	Barack Obama	
Gender	Male	Chuck Grassley	

By Definition 10, a fact is more striking if its value for the attribute $\mathcal{A}$ of the target entity t is *rarer*, i.e. has lower frequency, than *most* others. While alternative strikingness measures [3, 18, 25, 35] exist and could be used with our design. We have adopted an existing measure that has been shown to be effective in finding OFs. Example 3 provides a toy example for the strikingness scoring.

Example 3. *From Figure 1, a candidate OF is Q = (Harris, gender, Female, P(Human, position, Pence)). In Table 1, the value **Male** for attribute **gender** has a strictly higher frequency than **Female**. Thus, $\mathcal{I}(Q)$ sums to 0.6, as there are no other values to be considered.*

2.4 The COF Mining Problem

We formalize the mining process as the top-(k, l) COF problem in Definition 11. The context-awareness of a COF is achieved by ensuring the relevance of *context-aware peer entities* as well as their relationships, i.e., *context-anchored patterns*, associated with the context entity.

Definition 11 (Top-(k,l) COF Problem). *Given the target and context $\langle t, c \rangle$, and $\mathcal{G}(\mathcal{V}, \mathcal{E})$, we find the top-$l$ COFs with the highest strikingness scores from the top-k relevant context-anchored patterns.*

To solve the top-(k, l) COF problem, the processing divides into two steps. In the first step, we find the top-k relevant context-anchored patterns ranked by $\mathcal{R}(\cdot)$ in Definition 8. In the second step, we extract the top-l COFs from the collected k patterns. The major challenge lies in the first step, as the second step can be based on existing well-developed approaches [30, 35]. In the first step, in order to identify context-anchored patterns, we need to first conduct path enumeration to find connecting paths between the target and the context nodes. Then, for each connecting path, we need to enumerate all possible patterns, and for each enumerated pattern, we further find all its matching instances in the graph for calculating the relevance score. Every procedure involved here is of exponential complexity. Even worse, the scale of today's open KGs renders the problem more difficult. To mitigate the efficiency issue, we propose optimizations to quickly prune unpromising patterns and avoid any redundant computation to speed up the search. As a result, we reduce the search time from more than tens of seconds to sub-seconds on average over hundreds of queries, running on a KG with more than a half billion edges. Before moving on, we refer readers to Table 2 for the frequently used notations.

3 ALGORITHMS AND OPTIMIZATIONS

In this section, we first introduce the baseline algorithm. Then, we present several optimizations that speed up the processing.

3.1 Overview and Baseline Algorithm

Given $\langle t, c \rangle$, the baseline algorithm runs in two steps to produce the top-(k, l) COFs. In the **first step**, we traverse the graph with a

Table 2: Frequently Used Notations.

Notation	Meaning
$\mathcal{G}(\mathcal{V}, \mathcal{E}, \mathcal{M})$	The Knowledge Graph (KG).
$\tilde{v}$	A node variable, instantiated to either a node or a node type.
V, E	Node type and edge type.
$\langle t, c \rangle$	A pair of the target node and the context node.
$P(\tilde{v_0}, \tilde{v_\ell})$	An ℓ-hop path pattern from $\tilde{v_0}$ to $\tilde{v_\ell}$
$p(v_0, v_\ell)$	An ℓ-hop path or matching instance from v_0 to v_ℓ.
$S(v, u)$	Node proximity measure of u to v.
N_i	The matching node set w.r.t. $\tilde{v_i}$ of a pattern.
$S(c, N_i)$	Proximity of a matching node set N_i w.r.t. c.
$\mathcal{R}(P)$	Relevance score of a context-anchored pattern P.
$\mathcal{R}_k$	The k^{th} largest pattern relevance score.
$PR(p)$	The path relevance score of p.
$HR(P), VR(P)$	Horizontal and Vertical relevance scores of pattern P.
Q	A candidate COF.
w	The significance threshold of peer entity number.

bi-directional Breadth-First Search (bi-BFS) to find path instances $p(t, c)$ connecting t and c. For each $p(t, c)$, we enumerate the possible patterns $P(\tilde{v_0}, c)$ where $p \triangleright P$ and then evaluate the relevance score $\mathcal{R}(P)$. The bi-BFS includes a forward-BFS from t and a backward-BFS from c. The first step terminates after collecting the exact top-k context-anchored patterns. In the **second step**, we extract top-l COFs with the existing approach [30, 35] from the collected patterns. In this section, we focus on the first step.

The details are shown in Algorithm 1. The first step corresponds to line 1 to 19. During search, we keep two priority queues, i.e., q_f and q_b for forward-BFS and backward-BFS, respectively. The priority queues pop out the path instance with the highest PR score. Specifically, at line 7, we enumerate a prefix path $p(t, v_i)$, and check if it can be concatenated with some postfix path $p(v_i, c)$ to form a complete path $p(t, c)$. The postfix path $p(v_i, c)$ was obtained during backward-BFS traversal. Once a complete path is found, it is passed to the pattern evaluation procedure, introduced shortly, for pattern relevance score calculation. After the pattern evaluation, we enumerate new path instances from $p(t, v_i)$ by appending to p the out-neighbors of v_i. A path p can be inserted into the priority queue q_f only if its relevance score $PR(p)$ is larger than the k^{th} score of the collected patterns, denoted by $\mathcal{R}_k$. The same procedure is done for backward-BFS, but expands with the reversed directions of edges. The bi-BFS paradigm effectively speeds up the search by cutting down the complexity from $O(d^\ell)$ of uni-direction expansion to $O(d^{\frac{\ell}{2}})$, where d is the average degree and ℓ is the path length. The correctness of the algorithm is guaranteed by Theorem 1, which can be implied from Lemma 1.

Lemma 1 (Monotonicity of PR scores). *The path instances enumerated during forward-BFS and backward-BFS produce non-increasing PR scores.*

Theorem 1 (Correctness). *Algorithm 1 correctly returns the top-k relevant context-anchored patterns.*

The pattern evaluation procedure consumes a complete path $p(t, c) = p(t, v_i) + p(v_i, c)$, then enumerates all possible context-anchored patterns, and finally calculate the relevance score of each pattern by extracting all the matching node sets. The pattern enumeration maps each node v_i of $p(t, c)$ to either the node type V_i or itself v_i. Hence, there are $2^{\ell-2}$ patterns for an ℓ-node path. Note that t should always be mapped to its type and c remains an instance node. Given a pattern $P(\tilde{v_0}, c)$, the extraction of matching node sets starts backtrack reversely from c in a Depth First Search (DFS) manner, so as to recover every matching instance. Due to the space limit, we refer readers to supplementary materials for details.

Algorithm 1: FMiner

Input: $\langle t, c \rangle$, $\mathcal{G}(\mathcal{V}, \mathcal{E})$, and β.
Output: Top-(k, l) COFs.

```
1  NodeProximityCalculation(G, c);
2  Initialize a fwd-queue q_f and a bwd-queue q_b;
3  q_f.push(p(t, t));
4  q_b.push(p(c, c));
5  while q_f not empty or q_b not empty do
6      if q_f not empty then
7          p(t, v_i) ← q_f.pop();
8          foreach visited p(v_i, c) from bwd-BFS do
9              patternEvaluation(p(t, v_i), p(v_i, c));
10         foreach v_{i+1} ∈ OutNeighbor(v_i) do
11             if PR(p(t, v_{i+1})) > R_k then
12                 q_f.push(p(t, v_{i+1}));
13     if q_b not empty then
14         p(v_j, c) ← q_b.pop();
15         foreach visited p(t, v_j) from fwd-BFS do
16             patternEvaluation(p(t, v_j), p(v_j, c));
17         foreach v_{j-1} ∈ InNeighbor(v_j) do
18             if PR(p(v_{j-1}, c)) > R_k then
19                 q_b.push(p(v_{j-1}, c));
20 top-l COFs ← COFExtraction(top-k relevant patterns);
21 return Top-(k, l) COFs;
```

The pattern evaluation procedure, unfortunately, turns out to be a bottleneck, due to the inherent exponential complexity. In the next section, we introduce optimizations that accelerate this procedure while retrieving the exact top-k context-anchored patterns.

Discussion. There are many keyword search approaches [1, 14, 16] that involve graph traversal and path enumeration processes. However, our scenario differs in two aspects, rendering their technique not applicable. *First*, the keyword search approaches aim at finding a concise subtree, with one path from each *keyword cluster* (i.e., all nodes containing one keyword) to the root. Therefore, their techniques and optimizations mainly focus on quickly discovering the optimal answer subtree. To speed up the process, they build indexes [14], such as *keyword-node lists* and *node-keyword map*, to maintain shortest paths between keywords and nodes. Whereas, in our case, there are only two nodes under consideration. Those indexes do not bring any benefit since we need not consider the combinations of paths from different *keyword clusters*. *Second*, keyword search approaches only favor the shortest path between any two nodes [14]. Nonetheless, we need to evaluate the patterns derived from *all* connecting paths for top-k relevant pattern search.

3.2 Optimizations

In this section, we present three major optimizations for accelerating the top-k pattern search step. **First**, we propose an upper bound $\widehat{N_i}$ on $|N_i|$. $\widehat{N_i}$ is used during the extraction of matching node sets. Once the number of the collected matching nodes for $\tilde{v_i}$ exceeds $\widehat{N_i}$, the corresponding pattern can be safely pruned. **Second**, we avoid redundant processing during pattern evaluation by reusing the intermediate results. Moreover, we introduce the *pattern subsumption* relation (Definition 12), which further helps early terminate the evaluation for unpromising patterns. **Third**, the algorithm mainly relies on the k^{th} pattern relevance score, i.e. $\mathcal{R}_k$, for pruning. Therefore, we enhance the $\mathcal{R}_k$ by changing Algorithm 1 to a two-pass

algorithm. We delay the expansion via high-degree nodes until the second pass. This is based on the observation that high-degree nodes often result in a large number of patterns, of which many have low relevance scores. The details are introduced as follows.

Upper bound $\widehat{N_i}$ of $|N_i|$. Given a path pattern, we aim to early terminate the *pattern evaluation* procedure for patterns guaranteed to fall under $\mathcal{R}_k$. According to Definition 8, in order to obtain a pattern $P(\tilde{v_0}, \tilde{v_\ell} = c)$ with a score no smaller than $\mathcal{R}_k$, Equation 3 must hold for every matching node set N_i of P.

$$\forall i \in \{0, \ldots, \ell\}, \mathcal{R}_k \leq \beta^{-i} \cdot S(c, N_i) = \beta^{-i} \cdot \frac{\sum_{n \in N_i} S(c, n)}{|N_i|} \quad (3)$$

where $\mathcal{R}_k$ denotes the k^{th} pattern score and β is the length decay factor. A direct transformation of Equation 3 leads to an upper bound of $|N_i|$ as $|N_i| \leq \beta^{-i} \cdot \frac{\sum_{n \in N_i} S(c,n)}{\mathcal{R}_k}$. However, the numerator $\sum_{n \in N_i} S(c, n)$ is unknown until the exact N_i is found, which is the expensive part we want to avoid. Note that $\sum_{n \in N_i} S(c, n)$ can be bounded by $\sum_{u \in U_i} S(c, u)$ for any super set U_i of N_i. For instance, a trivial and useless bound can be obtained by setting $U_i = \mathcal{V}$. Thus, to derive a tight bound with cheap computational costs, we propose a *double-type constrained upper bound* of $\sum_{n \in N_i} S(c, n)$. Specifically, we introduce a super set U_i of N_i such that $\forall u \in U_i$ must have *(i)* the same type as V_i (w.r.t. to nodes in N_i) and *(ii)* at least one out-going edge with type E_i, given the pattern $P(\tilde{v_0}, \tilde{v_\ell} = c)$ being evaluated. Note that the collection of U_i is easy and does not involve any expensive graph traversal or join operations. Moreover, the *double-type constraint* greatly reduces the possible size of U_i, i.e., $|U_i| \ll |\mathcal{V}|$, leading to a much tighter upper bound with effective pruning power. This is formalized in Theorem 2. Note that the upper bound only applies to node types of a pattern.

Theorem 2. *Given a path pattern $P(\tilde{v_0}, \tilde{v_\ell} = c)$, the size of N_i (i.e. $|N_i|$) w.r.t. V_i is bounded by $\widehat{N_i}$, defined as below:*

$$\widehat{N_i} = \beta^{-i} \cdot \frac{\sum_{u \in U_i} S(c, u)}{\mathcal{R}_k},$$

where $U_i = \{u | \mathcal{M}(u) = V_i \wedge E_i \in outType(u)\}$, $outType(u)$ denotes the set of types of edges going out of u, $\mathcal{R}_k$ is the k^{th} pattern score, and β is the length decay factor.

To incorporate this optimization into the pattern evaluation procedure, we calculate the $\widehat{N_i}$ before extracting the matching node sets. Then, during the matching processes, once N_i is modified s.t. $|N_i| > \widehat{N_i}$, we can immediately terminate the processing.

Avoiding redundant processing. If two patterns share a common prefix pattern, then the matching instances for that prefix pattern can be extracted once and reused for the second pattern. Motivated by this observation, we can keep a dynamic map, called **Dmap**, which records the set of matching instances for each processed prefix pattern ending at some node v_i, i.e., $P(\tilde{v_0}, v_i)$. The Dmap is defined as below:

$$Dmap[P(\tilde{v_0}, v_i)] \mapsto \{N_0, \ldots, N_i\}.$$

Given Dmap, every time the matching process hits a visited prefix pattern $P(\tilde{v_0}, v_i)$, we can avoid repeatedly extracting the matching node sets of P by reusing the kept intermediate results in Dmap.

Furthermore, Dmap can be combined with the first optimization using $\widehat{N_i}$ to avoid redundant processing of patterns that are guaranteed to exceed $\widehat{N_i}$ for some N_i and thus fail below $\mathcal{R}_k$. Before introducing the details, we first present the definition of the **pattern subsumption** relation and Corollary 1.

Definition 12 (Pattern subsumption). *Pattern $P_1(\tilde{v_0}, \tilde{v_\ell} = v_\ell)$ subsumes $P_2(\tilde{u_0}, \tilde{u_m} = u_m)$, denoted by $P_1 \sqsupseteq P_2$, if and only if two conditions are satisfied: (1) $\ell \geq m$; (2) $\forall i \in \{0, \ldots, m\}$, if $\tilde{u_i} = U_i$ (a type), then $\tilde{v_i} = \tilde{u_i}$, or if $\tilde{u_i} = u_i$ (an instance node), then $\mathcal{M}(\tilde{u_i}) = \tilde{v_i}$.*

Corollary 1. *If $P_1(\tilde{v_0}, \tilde{v_\ell} = v_\ell) \sqsupseteq P_2(\tilde{u_0}, \tilde{u_m} = u_m)$, then $\forall i \in \{0, \ldots, m\}, N_{\tilde{v_i}} \supseteq N_{\tilde{u_i}}$.*

Based on Corollary 1, if a prefix pattern $P(\tilde{v_0}, v_i)$ has a matching node set N_j exceeding $\widehat{N_j}$ ($j < i$), then any pattern that subsumes P can be safely pruned. This is because they share the same $\widehat{N_i}$ and a pattern subsuming P contains at least N_i for $\tilde{v_i}$. We can extend Dmap to record a failure flag for a prefix pattern $P(\tilde{v_0}, v_i)$, i.e., $Dmap[P(\tilde{v_0}, v_i)] = FAIL$, if $P(\tilde{v_0}, v_i)$ results in $|N_j| > \widehat{N_j}, \exists j < i$.

Enhancing the k^{th} score ($\mathcal{R}_k$) for pruning. The algorithm relies on the $\mathcal{R}_k$ to prune paths (line 11 and 18 in Algorithm 1) as well as derive a tight bound $\widehat{N_i}$ of matching node sets. An insight is that a high-degree node can lead to many patterns being enumerated, evaluated, and finally inserted into the top-k result heap. There can be many of these patterns that tend to be less relevant, because the connection via a high-degree node can be weak. Consequently, those patterns will be later replaced by other more relevant ones. The downside is that the $\mathcal{R}_k$ generated from them is relatively low and does not provide a good pruning power. Motivated by this, we propose to defer the traversing through the high-degree nodes until the processing of other nodes finishes. More specifically, we classify all nodes whose degree is larger than a threshold r as high-degree nodes. The original Algorithm 1 is changed to a two-pass algorithm. In the first pass, it processes all nodes with degree no more than r. Meanwhile, any path hitting a node with degree larger than r is kept separately without further expansion. In the second pass, we resume the expansion and processing of all paths hitting high-degree nodes. To guarantee the correctness, the prefix (resp. postfix) paths generated in the second pass are also concatenated with those postfix (reps. prefix) paths from the first pass. In the experiment section, we show that the optimizations introduced in this section have effectively helped reduce the search time from tens of seconds to sub-seconds given a pair of target and context entities on a KG with more than a half billion edges.

4 EXPERIMENTS

In this section, we verify the efficiency and effectiveness of FMiner in Section 4.1 and Section 4.2, respectively.

Knowledge Graph and Query Pairs. We use Wikidata KG [5] with 39.9M nodes and 592.8M edges. We randomly collect 200 query pairs of co-occurring target and context entities from a public news corpus [3]. We refer readers to the supplementary materials for information of the running platform and the implementation details.

[3] https://cs.nyu.edu/~kcho/DMQA/

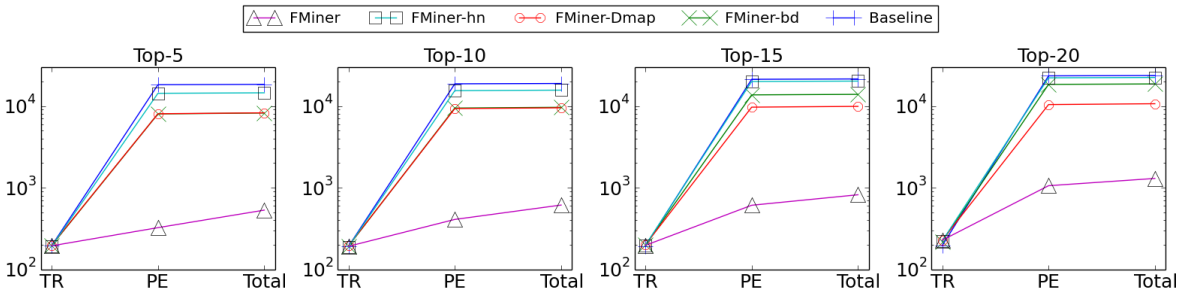

Figure 2: Vary top-k values. There are three time components of the algorithm: Traversal Time (TR), Pattern Evaluation Time (PE) and COF Extraction Time (CE). Note that Total = TR + PE + CE. Time is measured in millisecond.

4.1 Efficiency Study

Algorithms to Compare. To our best knowledge, there are no existing studies for COF mining. A closely related one, Maverick [35], only finds OFs for one entity at a time without considering any context. Maverick and FMiner have different query formats and objectives. In addition, according to our experiments, Maverick can only support small KGs reported in their paper [35] and cannot terminate in reasonable time for Wikidata KG used in our paper. Thus, we focus on studying the efficiency of FMiner and its variants with the proposed optimizations, as listed below.

- **Baseline**: Algorithm 1 without any optimizations.
- **FMiner-bd**: The algorithm that only uses $\widehat{N_i}$ for pruning.
- **FMiner-Dmap**: The algorithm that only uses Dmap for avoiding redundant processing of prefix patterns. This version does not combine $\widehat{N_i}$ with Dmap.
- **FMiner-hn**: The algorithm that only delays the processing of high-degree nodes.
- **FMiner**: The ultimate version integrating all optimizations.

Parameters to Study. There are three parameters to be studied for efficiency. First, the top-k value for obtaining k patterns. We set $k = 10$ by default. The corresponding experiment result is reported in Figure 2. Second, the top-l value for extracting l COFs from the collected patterns. We set $l = 10$ by default. The result is reported in Table 3. Third, we vary the degree threshold r which is used for delaying the processing of high-degree nodes. We set $r = 10000$ by default. We report the result in Figure 3. When studying one parameter, the others are set to the default values. In all experiments, we do not include the time profiling for node proximity score calculation since we adopt PPR (Personalized PageRank) with the existing approach [8]. The time for proximity score computation is negligible. *All reported running time are in millisecond and averaged across all queries.* We set time limit to 180 seconds (3 minutes). If such a limit is reached, we return whatever is collected. In addition, we set $\beta = \mathbf{e}$ by default. The significance threshold w (Definition 9) is set to 20. A large w can lead to missing of interesting groups of peer entities. For instance, the number of peer entities comprising all the US presidents is only 46 (at the time the paper was written). Note that we only consider paths with fewer than four nodes since longer paths can only lead to complex patterns, which is less meaningful and can hardly be understood by users.

Result and analysis. In Figure 2, we vary top-k with four different values, 5, 10, 15 and 20. In each sub-figure, we show time components including Traversal time (TR) for expanding and enumerating paths, Pattern Evaluation time (PE) for evaluating pattern relevance, and Total time. The COF Extraction time (CE) is the same for all versions of algorithms (COF extraction starts after top-k patterns are collected), which is separately reported in Table 3. Note that Total time = TR + PE + CE.

From the results shown in Figure 2, FMiner with all optimizations has reduced the search time to sub-second. Whereas, solely applying any individual optimization cannot achieve the same efficiency result. Specifically, simply delaying high-degree nodes (FMiner-hn) does not help much. Although it helps to discover more relevant patterns in an early stage, there is no effective pruning utilizing the improved k^{th} score. The other two optimizations improved the efficiency from different aspects, i.e., by pruning (FMiner-bd) and avoiding redundant processing (FMiner-Dmap). The ultimate version (FMiner) achieved the best performance by combining all optimizations. FMiner-hn improved the k^{th} score which further strengthens FMiner-bd by allowing for a tighter bound $\widehat{N_i}$. Moreover, FMiner-bd is further combined with FMiner-Dmap so that the evaluation of patterns failing $\widehat{N_i}$ is terminated quickly. The traversal process is not a bottleneck due to the effectiveness of bi-BFS paradigm. As the top-k value increases, the time cost slightly increases because a larger k leads to a lower k^{th} score. Hence, it can weaken the pruning power of the optimizations. The result of varying top-l for COF extraction is reported in Table 3.

Table 3: Vary top-l for COF extraction.

top-l	top-5	top-10	top-15	top-20
CE time (ms)	12.459	12.774	13.389	14.467

Table 4: Degree distribution.

node degree	≥ 100	≥ 1000	$\geq 10,000$	$\geq 10,000$
node number	617,941	14,787	1,087	63

In Figure 3, we have investigated the influence of degree threshold r by varying it from 1,000 to 100,000. Only FMiner (the ultimate version) and FMiner-hn (delaying high-degree node) are affected by different r and thus plotted. The Baseline is also shown for comparison. The node-degree distribution, as in Table 4, corresponds to a power law distribution. As the degree threshold r increases, r becomes less sensitive since only very few nodes are affected (there are totally 39M nodes). The best result is achieved by setting $r = 10000$. Smaller r has caused the running time to increase. This is because smaller r delays too many nodes, some of which may result in very relevant patterns. Consequently, good k^{th} scores may be missed in the first pass of the algorithm. In addition, raising the degree threshold to 100,000 slightly increases the running time by

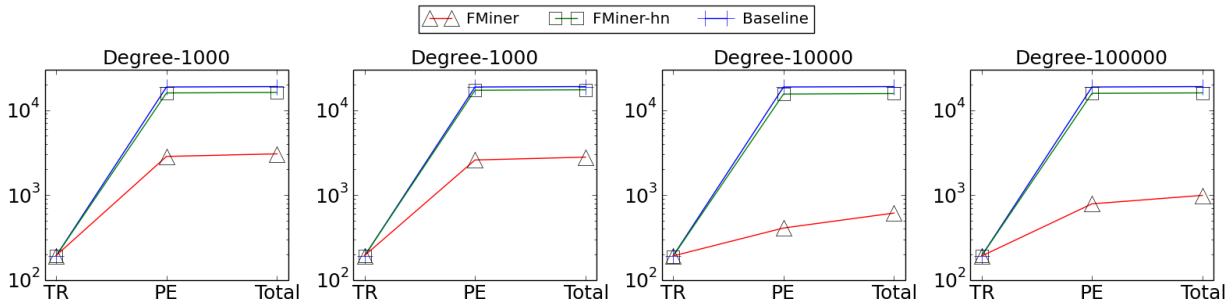

Figure 3: Vary degree threshold. Note that Total = TR + PE + CE. Time is measured in millisecond.

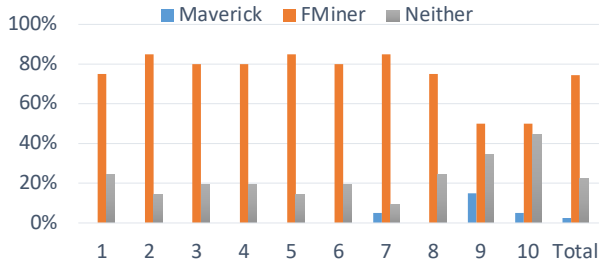

Figure 4: Relevance Study.

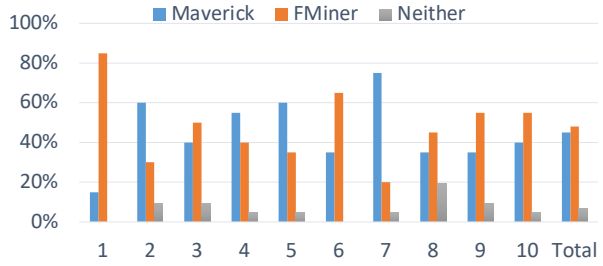

Figure 5: Strikingness Study.

around a hundred milliseconds. This indicates larger r can lead to more high-degree nodes being processed in the first pass of the algorithm. This makes the derived k^{th} score less effective for pruning. However, this effect is not as strong as decreasing the threshold. Our results suggest a larger r is preferred since such choice does not affect running time in a sensitive way, and can effectively accelerate the search process.

4.2 Effectiveness Study

In this section, we conduct a user study to validate two aspects of FMiner: *(i)* the resulting COFs are *relevant* regarding the context entity; *(ii)* the resulting COFs are *interesting facts* regarding the target entity. We compare our results with the OFs produced by Maverick [35] which discovers exceptional facts from KGs without considering any context. However, the search space of Maverick is intractable given the size of Wikidata, we thus use Maverick to discover all path patterns from the target entity. The results from both methods are indeed interesting, but with different relevance regarding a given context.

In the user study, we collect 10 pairs of entities that are well-known and relevant to each other. For each entity pair, we fix one entity as the target, the other as the context. Note that Maverick and FMiner adopt the same strikingness measure in Definition 10. We use the top OFs from each approach. *Only the resulting two OFs (in random order) are presented to participants per user study case. Then, two questions are asked: (1) Which fact is more relevant regarding the [context entity]? (2) Which fact sounds more interesting or striking regarding the [target entity]? For each of the two questions, we ask participants to select one out of three choices:* ***Fact 1***, ***Fact 2*** *and* ***Neither***. We convert OFs into natural languages for ease of understanding of the participants. We have collected the results from 20 students who are unaware of our research. Two output samples are shown in Table 5 for discussion. The full details of all user study cases are shown and further discussed in supplementary materials, including the used entity pairs, the strikingness scores and the discovered patterns of each study case.

In the **relevance study** (Figure 4), the results verify that FMiner can consistently outperform Maverick in retrieving relevant facts. FMiner extracts COFs from peer entities with *relevant* relationships to the context entity. For instance, in Table 5, the peer entities retrieved by FMiner consist of all the first ladies of the US including Hillary Clinton (Case 1). In contrast, Maverick focuses on finding global optimal OFs with highest striking scores while ignoring the context. In the **strikingness study** (Figure 5), the results validate that while achieving context-awareness, FMiner can still produce competitive OFs as compared with Maverick. Maverick tend to produce OFs with a large number of peer entities, which can render a fact more striking. For example, in Case 7 (Table 5), although the two approaches obtained a very similar fact (i.e. Taylor Swift plays Banjo), the OF statement by Maverick tend to sound more striking. This is because Maverick considers the peer entities of all actors (190470 recorded in the KG), whereas FMiner only considers those with the same music genre as Lady Gaga (220 recorded). However, the OF produced by FMiner is more suitable to be embedded into the context specified by Lady Gaga.

In sum, FMiner can produce striking and context-aware OFs. Such OFs have important applications especially in news and social media, where the contents often have a clear context involving real-world entities.

5 RELATED WORK

In this section, we discuss two closely related research areas.

Outlier Detection and Outlying Aspect Detection. Although similar, these two tasks have very different objectives. Outlier detection [9, 21, 24, 27] aims at finding an object that is different from objects in a cluster, whereas outlying aspect detection [3, 12, 23, 29, 35] is to find the most interesting attribute (i.e. the outlying aspect) of a given object compared with a set of peer objects. There is a thorough discussion on the differences of the two tasks in [4, 25, 35].

Table 5: Sample Facts.

Entity Pair	Maverick	FMiner
Case 1: t=Michelle Obama, c=Hillary Clinton	Among all people who have family name Robinson, Michelle also has Obama as her family name.	Among the first ladies of the US, Michelle is the only one who has been Dean in any university.
Case 7: t=Taylor Swift, c=Lady Gaga	Among all actors, Taylor is one of a few banjoists.	Among people with the same music genre as Lady Gaga, Taylor is the only one whose instrument includes banjo.

Both the attributes and the set of peer entities need to be mined and extracted from the dataset for outlying aspects detection. Thus, our work is more close to outlying aspect detection. However, most of the existing approaches [3, 12, 23, 29] of outlying aspect detection focus on relational data and assume a single table as pointed out by [35]. Maverick [35] is a recent work that proposes an approach for mining exceptional facts on knowledge graphs. In this paper, we adopt the same strikingness measure proposed in [30, 35] due to its proved effectiveness. There are several alternative ones [3, 18, 25] to choose from. Nonetheless, there are few outlying aspect detection methods aimed at mining context-aware OFs where the context is specified by an entity. Thus, they are not applicable to our scenario.

Entity Relationship Discovery on Graphs. One popular method of relationship discovery is keyword search on graphs, to mention a few [1, 14, 16, 17, 33, 34]. These methods return concise subgraphs that connect multiple input keywords. However, these approaches do not consider any patterns which are required for generating OF. In addition, [32] proposes to find tree patterns and output table answers. Furthermore, REX [7] proposes to find path patterns that connect two entities. It focuses on combining the path patterns to form a graph pattern that explains the relationship between the entity pair. Neither of the two works [7, 32] consider context relevance ranking, peer entity finding nor OF extraction.

6 CONCLUSION

We proposed FMiner, a novel framework that automatically mines context-aware outstanding facts (COFs) from open KGs, given a target entity and a context entity. The resulting COFs can serve as lead to interesting news materials and aid tasks like fact-checking and news promotion. We proposed several optimizations to speed up the search processes. Through extensive experiments, we validated the advantages of our approach in terms of both efficiency and effectiveness.

Acknowledgement. Yuchen Li's work was partially supported by Lee Kong Chian fellowship. This work was also supported in part by the Danish Council for Independent Research (Research Project 9041-00382B).

REFERENCES

[1] B. Aditya, Gaurav Bhalotia, Soumen Chakrabarti, Arvind Hulgeri, Charuta Nakhe, Parag Parag, and S. Sudarshan. 2002. BANKS: Browsing and Keyword Searching in Relational Databases. In *VLDB'02*.
[2] Albert Angel, Nick Koudas, Nikos Sarkas, Divesh Srivastava, Michael Svendsen, and Srikanta Tirthapura. 2012. Dense Subgraph Maintenance under Streaming Edge Weight Updates for Real-time Story Identification. In *VLDB '12*.
[3] Fabrizio Angiulli, Fabio Fassetti, and Luigi Palopoli. 2009. Detecting Outlying Properties of Exceptional Objects. *ACM Trans. Database Syst.* 34, 1 (April 2009).
[4] Lei Duan, Guanting Tang, Jian Pei, James Bailey, Akiko Campbell, and Changjie Tang. 2015. Mining Outlying Aspects on Numeric Data. *Data Min. Knowl. Discov.* 29, 5 (Sept. 2015).
[5] Fredo Erxleben, Michael Günther, Markus Krötzsch, Julian Mendez, and Denny Vrandeăić. 2014. Introducing Wikidata to the Linked Data Web. In *ISWC '14*.
[6] Qi Fan, Yuchen Li, Dongxiang Zhang, and Kian-Lee Tan. [n.d.]. Discovering newsworthy themes from sequenced data: A step towards computational journalism. In *TKDE '17*.
[7] Lujun Fang, Anish Das Sarma, Cong Yu, and Philip Bohannon. 2011. REX: Explaining Relationships Between Entity Pairs. In *VLDB '11*.
[8] Dániel Fogaras and Balázs Rácz. 2004. Towards Scaling Fully Personalized PageRank. In *Algorithms and Models for the Web-Graph*. Springer Berlin Heidelberg.
[9] Jing Gao, Feng Liang, Wei Fan, Chi Wang, Yizhou Sun, and Jiawei Han. 2010. On Community Outliers and Their Efficient Detection in Information Networks. In *KDD '10*.
[10] Naeemul Hassan, Fatma Arslan, Chengkai Li, and Mark Tremayne. 2017. Toward Automated Fact-Checking: Detecting Check-worthy Factual Claims by ClaimBuster. In *KDD '17*.
[11] Naeemul Hassan, Chengkai Li, and Mark Tremayne. 2015. Detecting Check-worthy Factual Claims in Presidential Debates. In *CIKM '15*.
[12] Naeemul Hassan, Afroza Sultana, You Wu, Gensheng Zhang, Chengkai Li, Jun Yang, and Cong Yu. 2014. Data in, Fact out: Automated Monitoring of Facts by FactWatcher. In *VLDB '14*.
[13] Taher H. Haveliwala. 2002. Topic-sensitive PageRank. In *WWW '02*.
[14] Hao He, Haixun Wang, Jun Yang, and Philip S. Yu. 2007. BLINKS: Ranked Keyword Searches on Graphs. In *SIGMOD'07*.
[15] Glen Jeh and Jennifer Widom. 2002. SimRank: A Measure of Structural-context Similarity. In *KDD '02*.
[16] Varun Kacholia, Shashank Pandit, Soumen Chakrabarti, S. Sudarshan, Rushi Desai, and Hrishikesh Karambelkar. 2005. Bidirectional Expansion for Keyword Search on Graph Databases. In *VLDB'05*.
[17] Rong-Hua Li, Lu Qin, Jeffrey Xu Yu, and Rui Mao. 2016. Efficient and Progressive Group Steiner Tree Search. In *SIGMOD'16*.
[18] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. 2008. Isolation Forest. In *ICDM '08*.
[19] Peter Lofgren, Siddhartha Banerjee, and Ashish Goel. 2016. Personalized PageRank Estimation and Search: A Bidirectional Approach. In *WSDM '16*.
[20] L. Page, S. Brin, R. Motwani, and T. Winograd. 1998. The PageRank citation ranking: Bringing order to the Web. In *WWW '07*.
[21] Bryan Perozzi, Leman Akoglu, Patricia Iglesias Sánchez, and Emmanuel Müller. 2014. Focused Clustering and Outlier Detection in Large Attributed Graphs. In *KDD '14*.
[22] W. Shen, J. Wang, and J. Han. 2015. Entity Linking with a Knowledge Base: Issues, Techniques, and Solutions. In *TKDE '15*.
[23] A. Sultana, N. Hassan, C. Li, J. Yang, and C. Yu. 2014. Incremental discovery of prominent situational facts. In *ICDE '14*.
[24] Hanghang Tong and Ching-Yung Lin. 2012. Non-negative Residual Matrix Factorization: Problem Definition, Fast Solutions, and Applications. *Stat. Anal. Data Min.* 5, 1 (Feb. 2012).
[25] Nguyen Xuan Vinh, Jeffrey Chan, Simone Romano, James Bailey, Christopher Leckie, Kotagiri Ramamohanarao, and Jian Pei. 2016. Discovering Outlying Aspects in Large Datasets. *Data Min. Knowl. Discov.* 30, 6 (Nov. 2016).
[26] Sibo Wang, Renchi Yang, Xiaokui Xiao, Zhewei Wei, and Yin Yang. 2017. FORA: Simple and Effective Approximate Single-Source Personalized PageRank. In *KDD '17*.
[27] Xiang Wang and Ian Davidson. 2009. Discovering Contexts and Contextual Outliers Using Random Walks in Graphs. In *ICDM '09*.
[28] Yanhao Wang, Yuchen Li, Ju Fan, Chang Ye, and Mingke Chai. 2021. A survey of typical attributed graph queries. *World Wide Web* 24, 1 (2021), 297–346.
[29] Tianyi Wu, Dong Xin, Qiaozhu Mei, and Jiawei Han. 2009. Promotion Analysis in Multi-dimensional Space. In *VLDB '09*.
[30] You Wu, Pankaj K. Agarwal, Chengkai Li, Jun Yang, and Cong Yu. 2012. On "One of the Few" Objects. In *KDD '12*.
[31] Yubao Wu, Ruoming Jin, and Xiang Zhang. 2014. Fast and Unified Local Search for Random Walk Based K-nearest-neighbor Query in Large Graphs. In *SIGMOD '14*.
[32] Mohan Yang, Bolin Ding, Surajit Chaudhuri, and Kaushik Chakrabarti. 2014. Finding Patterns in a Knowledge Base Using Keywords to Compose Table Answers. In *VLDB '14*.
[33] Y. Yang, D. Agrawal, H. V. Jagadish, A. K. H. Tung, and S. Wu. 2019. An Efficient Parallel Keyword Search Engine on Knowledge Graphs. In *ICDE '19*.
[34] Y. Yang, Y. Li, and A. K. H. Tung. 2021. NewsLink: Empowering Intuitive News Search with Knowledge Graphs. In *ICDE '21*.
[35] Gensheng Zhang, Damian Jimenez, and Chengkai Li. 2018. Maverick: Discovering Exceptional Facts from Knowledge Graphs. In *SIGMOD '18*.
[36] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. 2019. Deep Learning Based Recommender System: A Survey and New Perspectives. *ACM Comput. Surv.* 52, 1 (Feb. 2019).

APPENDIX: SUPPLEMENTARY MATERIALS

A DETAILS OF ALGORITHMS

In this section, we introduce the details of the pattern evaluation procedure (Algorithm 2) and the matching node set extraction procedure (Algorithm 3).

Algorithm 2: Pattern Evaluation

Input: A prefix path $p(v_0 = t, v_i)$ and a postfix path $p(v_i, v_\ell = c)$.
Output: Patterns with relevance scores.

1 **Procedure** patternEvaluation($p(v_0 = t, v_i), p(v_i, v_\ell = c)$)
2 $p(t, c) = p(v_0, v_i) + p(v_i, v_\ell)$;
3 Enumerate all possible $P(\tilde{v_0}, \tilde{v_\ell} = c)$, s.t., $p(t, c) \triangleright P(\tilde{v_0}, \tilde{v_\ell} = c)$;
4 **foreach** $P(\tilde{v_0}, c)$ **do**
5 **if** matchingNodeSetExtraction($P(\tilde{v_0}, v_\ell = c)$) **then**
6 Insert $P(\tilde{v_0}, v_\ell = c)$ into top-k pattern heap;

In Algorithm 2, the pattern evaluation procedure takes as input a prefix path $p_1(v_0 = t, v_i)$ and a postfix path $p_2(v_i, v_\ell = c)$. By concatenating the two paths, we derive a complete connecting path $p(t, c)$ (line 2). Then, all possible patterns are enumerated by turning the instance nodes of p to either types or the node itself (line 3). Subsequently, every pattern is passed to the matching node set extraction procedure for obtaining the matching node sets (line 5). The matching node sets are further used to calculate the VR score of the pattern.

Algorithm 3: Matching Node Set Extraction with Dmap

Input: $P(\tilde{v_0}, \tilde{v_\ell} = c)$, $\mathcal{G}(\mathcal{V}, \mathcal{E})$.
Output: $\{N_0, \ldots, N_\ell\}$ modified in place.

1 **Procedure** matchingNodeSetExtraction($P(\tilde{v_0}, v_\ell)$)
2 **if** $\ell == 0$ **then**
3 N_0.add(v_ℓ);
4 **return** true;
5 **if** *Dmap contains* $P(\tilde{v_0}, v_\ell)$ **then**
6 $\{N'_0, \ldots, N'_i\} \leftarrow Dmap[P(\tilde{v_0}, v_\ell)]$;
7 **foreach** $N'_j \in \{N'_0, \ldots, N'_i\}$ **do**
8 $N_j \leftarrow N_j \cup N'_j$;
9 **return** true;
10 hasMatch $\leftarrow$ false;
11 **foreach** $v_{\ell-1} \in InNeighbor(v_\ell)$ **do**
12 **if** $\mathcal{M}(e(v_{\ell-1}, v_\ell))! = E_{\ell-1}$ *or* $\mathcal{M}(v_{\ell-1})! = \tilde{v_{\ell-1}}$ **then**
13 continue;
14 **if** matchingNodeSetExtraction($P(\tilde{v_0}, v_{\ell-1})$) **then**
15 hasMatch = true;
16 **if** *hasMatch* **then**
17 $N_\ell.add(v_\ell)$;
18 $Dmap[P(\tilde{v_0}, v_\ell)] \leftarrow \{N_0, \ldots, N_i\}$;
19 **return** hasMatch;

Algorithm 3 shows the matching node set extraction integrated with Dmap optimization. The procedure starts from the context node c by following the reversed direction of the pattern $P(\tilde{v_0}, \tilde{v_\ell} = c)$. We search the graph in a DFS manner, recursively. Once reaching a node v_0 matching $\tilde{v_0}$ (line 2-3), a matching instance is found. The process returns true and inserts the nodes along the matching instance into the corresponding matching node set (line 3 and 17). The Dmap is integrated such that it stores any found matching node sets (line 18) and reuses the intermediate results (line 5-9) to avoid redundant processes.

B PROOFS

Lemma 1. We only need to show that given any path p' expanded from path p, $PR(p') \leq PR(p)$. For forward-BFS, suppose $p'(v_0, v_{i+1})$ is expanded from $p'(v_0, v_i)$, then we have $PR(p') = \min\{PR(p), \beta^{-(i+1)} \cdot S(c, v_{i+1})\}$. From this, we can obtain that $PR(p') \leq PR(p)$. For backward-BFS, suppose $p'(v_{i-1}, v_\ell)$ is expanded from $p'(v_i, v_\ell)$, then we have $PR(p') = \min\{\beta^{-1} \cdot S(c, v_{i+1}), \beta^{-1} \cdot PR(p)\}$. Hence, $PR(p') \leq \beta^{-1} \cdot PR(p) \leq PR(p)$. In sum, we have $PR(p') \leq PR(p)$ for both forward-BFS and backward-BFS. Thus, the PR scores of expanded paths are non-increasing. □

Theorem 1. We only need to show that every pattern P derived from a path p by the algorithm must have $\mathcal{R}(P) \leq PR(p)$. Thus, the algorithm can correctly terminates without the need to consider the paths whose PR scores are smaller than $\mathcal{R}_k$. We prove this by contradiction. Suppose P is derived from p' for the first time and $\mathcal{R}(P) > PR(p')$. However, from Definition 6, we know that $\mathcal{R}(P) \leq HR(P)$ where $HR(P) = \max\{PR(p)\}, \forall p \triangleright P$. This means there must exist a p s.t. $PR(p) > PR(p')$. Otherwise, $\mathcal{R}(P) \leq HR(P) = PR(p')$. Since $PR(p) > PR(p')$, based on Lemma 1, p must be enumerated before p' and is used for deriving P. This contradicts the assumption that p' is used to derive P for the first time. This concludes the proof. □

Theorem 2. The proof can be obtained from the construction of the upper bound $\widehat{N_i}$. We refer readers to Section 3.2. □

Corollary 1. Based on Definition 12, we know that if $P_1(\tilde{v_0}, \tilde{v_\ell} = v_\ell) \sqsupseteq P_2(\tilde{u_0}, \tilde{u_m} = u_m)$, then $p \triangleright P_1, \forall p \triangleright P_2$. Therefore, any matching node set of P_2 is a subset of that of P_1. □

C PLATFORM AND IMPLEMENTATION

All programs are run on a single machine with CentOS 7.0 and Intel(R) Xeon(R) Platinum 8170 CPU @ 2.1GHz. It has 1TB RAM. All algorithms are implemented using C++ 11, with -O3 flag turned on for compilation.

D DETAILS OF USER STUDY

In this section, we introduce the details of user study and give more discussions on the results. The entity pairs used in the user study are shown in Table 6. These pairs are well-known so that the participants know about the entities. The entities are also relevant to each other within a pair so that it makes sense to extract a context-aware OF for that entity pair.

Table 6: Entity Pairs used for User Study.

Case No.	Target Entity	Context Entity
1	Michelle Obama	Hillary Clinton
2	Akon	Michael Jackson
3	Steve Jobs	Bill Gates
4	Lionel Messi	Neymar
5	Donald Trump	Joe Biden
6	Barack Obama	Donald Trump
7	Taylor Swift	Lady Gaga
8	Apple Inc.	Microsoft
9	Jason Statham	Vin Diesel
10	Lebron James	Michael Jordan

Table 7: Detailed output. $(\mathcal{A}, \mathcal{X}, \mathcal{I}, |N_0|)$ denotes the attribute, value, strikingness score, and the number of context-aware peer entities. NL is short for Natural Language. We use E^{-1} to denote a right-to-left edge direction.

	Maverick	FMiner
Case 1	⟨ **Michelle Obama**, **Hillary Clinton**⟩	
Path Pattern	P(**Human**, $familyName$, Robinson)	P(**Human**, $positionHeld$, **FirstLadyOfUS**, $positionHeld^{-1}$, Hillary Clinton)
$(\mathcal{A}, \mathcal{X}, \mathcal{I}, \lvert N_0\rvert)$	($familyName$, Obama, 0.999519, 2086)	($positionHeld$, Dean, 0.948276, 58)
NL	Among all people who have family name Robinson, Michelle is the only that also has Obama as her family name.	Among the first ladies of the US, Michelle is the only one who has been Dean in any university.
Case 2	⟨**Akon**, **Michael Jackson** ⟩	
Path Pattern	P(**Human**, $occupation$, actor)	P(**Human**, $genre$, **MusicGenre**, $influencedBy^{-1}$, Michael Jackson)
$(\mathcal{A}, \mathcal{X}, \mathcal{I}, \lvert N_0\rvert)$	($occupation$, philanthropist, 0.999616, 190470)	($countryOfCitizenship$, Senegal, 0.996012, 7812)
NL	Among all actors, Akon is one of a few philanthropists.	Among people whose music genre is influenced by Michael Jackson, Akon is the only Senegalese-American.
Case 3	⟨**Steve Jobs**, **Bill Gates** ⟩	
Path Pattern	P(**Human**, $occupation$, engineer)	P(**Human**, $occupation$, **inventor**, $occupation^{-1}$, Bill Gates)
$(\mathcal{A}, \mathcal{X}, \mathcal{I}, \lvert N_0\rvert)$	($occupation$, executive producer, 0.999475, 28622)	($occupation$, executive producer, 0.995956, 5193)
NL	Among all engineers, Steve Jobs is one of a few executive producers.	Among all inventors, Steve Jobs is the only executive producer.
Case 4	⟨**Lionel Messi**, **Neymar** ⟩	
Path Pattern	P(**Human**, $participateIn$, 2008 Summer Olympics)	P(**Human**, $memberOf$, **F.C. Barcelona**, $memberOf^{-1}$, Neymar)
$(\mathcal{A}, \mathcal{X}, \mathcal{I}, \lvert N_0\rvert)$	($participateIn$, 2019 Copa América, 0.999203, 10391)	($memberOf$, Argentina national football team, 0.986425, 1547)
NL	Among all participants of 2008 Summer Olympics, Messi is one of a few who also participate in 2019 Copa América.	Among all team members of F.C. Barcelona, Messi is one of a few that also play in Argentina national football team.
Case 5	⟨**Donald Trump**, **Joe Biden** ⟩	
Path Pattern	P(**Human**, $occupation$, politician)	P(**Human**, $awardReceived$, **Medal**, $awardReceived^{-1}$, Joe Biden)
$(\mathcal{A}, \mathcal{X}, \mathcal{I}, \lvert N_0\rvert)$	($occupation$, game show host, 0.999978, 462240)	($awardReceived$, WWE Hall of Fame, 0.966346, 623)
NL	Among all politicians, Trump is one of a few game show hosts.	Among all people receiving some same medal as Joe Biden, Trump is the only one who has received the award of WWE Hall of Fame.
Case 6	⟨**Barack Obama**, **Donald Trump**⟩	
Path Pattern	P(**Human**,$occupation$, politician)	P(**Human**, $positionHeld$, **USPresident**, $positionHeld^{-1}$, Donald Trump)
$(\mathcal{A}, \mathcal{X}, \mathcal{I}, \lvert N_0\rvert)$	($occupation$, Community Organizer, 0.999948, 462240)	($language$, Indonesian, 0.977273, 46)
NL	Among all politicians, Obama is one of a few community organizers.	Among all the presidents of US, Obama is the only one who can speak Indonesian.
Case 7	⟨**Taylor Swift**, **Lady Gaga**⟩	
Path Pattern	P(**Human**, $occupation$, actor)	P(**Human**, $Genre$, **MusicGenre**, $Genre^{-1}$, Lady Gaga)
$(\mathcal{A}, \mathcal{X}, \mathcal{I}, \lvert N_0\rvert)$	($occupation$, banjoist, 0.999685, 190470)	($instrument$, banjo, 0.994737, 220)
NL	Among all actors, Taylor is one of a few banjoists.	Among people with the same music genre as Lady Gaga, Taylor is the only one whose instrument includes banjo.
Case 8	⟨**Apple Inc.**, **Microsoft** ⟩	
Path Pattern	P(**Business**, $industry$, **Industry**, $industry^{-1}$, GOG.com)	P(**Business**, $industry$, **software industry**, $industry^{-1}$, Microsoft)
$(\mathcal{A}, \mathcal{X}, \mathcal{I}, \lvert N_0\rvert)$	($industry$, consumer electronics, 0.999206, 3778)	($industry$, consumer electronics, 0.956522, 92)
NL	Among all businesses that share a same industry as GOG.com, Apple Inc. is one of a few that belong to the consumer electronics industry.	Among all businesses that fall in software industry, Apple Inc. is one of few that also belong to the consumer electronics industry.
Case 9	⟨**Jason Statham**, **Vin Diesel** ⟩	
Path Pattern	P(**Human**, $castMemberOf$, **Film**, $distributor$, Universal Pictures)	P(**Human**, $genre$, action movie, $genre^{-1}$, Vin Diesel)
$(\mathcal{A}, \mathcal{X}, \mathcal{I}, \lvert N_0\rvert)$	($occupation$, competitive diver, 0.987717, 13523)	($industry$, kickboxer, 0.960000, 50)
NL	Among all people starring in any films of Universal Pictures, Jason Statham is the only competitive diver.	Among all people whose movie genre is action movie, Jason Statham is the only kickboxer.
Case 10	⟨**LeBron James**, **Michael Jordan** ⟩	
Path Pattern	P(**Human**, $positionPlayedOnTeam$, **BasketballPosition**, $sport$, basketball)	P(**Human**, $awardReceived$, **NBA RookieOfTheYearAward**, $awardReceived^{-1}$, Michael Jordan)
$(\mathcal{A}, \mathcal{X}, \mathcal{I}, \lvert N_0\rvert)$	($positionPlayedOnTeam$, point forward, 0.999752, 20191)	($positionPlayedOnTeam$, point forward, 0.985915, 71)
NL	Among all people playing in some basketball team, Lebron James is one of few who play point forward.	Among all people who won the NBA Rookie of the year award as Michael Jordan, Lebron James is one of few who play point forward.

In Table 7, we show all details of output facts used in the user study. Note that only the fact descriptions are shown to participates so that participants are not overwhelmed. The target and context entities are only mentioned in the questions associated with each user study case. From the details shown in Table 7, Maverick is capable of finding OFs with higher strikingness scores than FMiner. The strikingness measure favors OFs that are generated from a large number of peer entities, which can result in higher strikingness scores. This can make the OFs sound more striking on one hand, e.g. Case 2 and 7 where Maverick outperforms FMiner in Figure 5. On the other hand, it sometimes can lead to less meaningful OFs, e.g. Case 1 and 6 where FMiner performs better.

From the results, the peer entities generated by FMiner do not have a large cardinality as Maverick. FMiner effectively confines the OF extraction to the relevant context, though losing some strikingness scores. There are some very interesting results, such as Case 1 where FMiner finds all first ladies of US, Case 6 where FMiner identifies all the US presidents, and Case 9 where FMiner navigates to the action movie genre that is very relevant to the two involved actors.

We note that FMiner relies on the accuracy and completeness of the underlying KG. Fortunately, Wikidata is rapidly growing and its content is covering more and more entities as well as topics. This makes COF finding from such a KG even more attractive.