ExFaKT: A Framework for Explaining Facts
over Knowledge Graphs and Text

Mohamed H. Gad-Elrab
Max-Planck Institute for Informatics
Saarland Informatics Campus , Germany
gadelrab@mpi-inf.mpg.de

Daria Stepanova[a]
Bosch Center for Artificial Intelligence
Renningen, Germany
daria.stepanova@de.bosch.com

Jacopo Urbani
Vrije Universiteit Amsterdam
Amsterdam, Netherlands
jacopo@cs.vu.nl

Gerhard Weikum
Max-Planck Institute for Informatics
Saarland Informatics Campus, Germany
weikum@mpi-inf.mpg.de

ABSTRACT

Fact-checking is a crucial task for accurately populating, updating and curating knowledge graphs. Manually validating candidate facts is time-consuming. Prior work on automating this task focuses on estimating truthfulness using numerical scores which are not human-interpretable. Others extract explicit mentions of the candidate fact in the text as an evidence for the candidate fact, which can be hard to directly spot. In our work, we introduce ExFaKT, a framework focused on generating human-comprehensible explanations for candidate facts. ExFaKT uses background knowledge encoded in the form of Horn clauses to rewrite the fact in question into a set of other easier-to-spot facts. The final output of our framework is a set of semantic traces for the candidate fact from both text and knowledge graphs. The experiments demonstrate that our rewritings significantly increase the recall of fact-spotting while preserving high precision. Moreover, we show that the explanations effectively help humans to perform fact-checking and can also be exploited for automating this task.

KEYWORDS

Knowledge Graph; Fact-checking; Explainable Evidence; Reasoning

1 INTRODUCTION

Motivation and Problem. Knowledge Graphs (KGs) are large collections of factual triples of the form ⟨*subject predicate object*⟩ (SPO) about people, companies, places, etc. Projects like BabelNet [28], DBpedia [5], Wikidata [38] and YAGO [36] have constructed KGs with millions of entities and billions of facts. However, KGs also contain doubtful if not incorrect SPO triples, as they are partly built by automatic information extraction, crowd-sourcing, or methods for KG completion using rules [16] or embeddings [29, 31, 40]. This incurs the problem of validating if an SPO triple is correct or not, a task that is often referred to as *fact-checking* or *truth discovery* [24].

Traditionally, fact-checking has been performed manually by human reviewers but this is time-consuming. Therefore, with the increase of false facts on the Web, the automation of fact-checking is gaining more attention. Methods for automatic fact-checking (e.g., [18, 24, 26, 30, 32]) proceed in two steps. First, they perform *fact-spotting* by searching for occurrences of a fact candidate, such as ⟨*Sadiq_Khan citizenOf UK*⟩, and possible alternatives, such as ⟨*Sadiq_Khan citizenOf Pakistan*⟩, in the Web sources. This is done by expanding the predicate into paraphrases (*e.g.*, "has nationality", "has passport") and searching for it jointly with the alias names of the S and O arguments. Then, the extracted evidence (or counter-evidence) is used to infer the truth value of the candidate fact.

Numerical scores produced by fully automated methods are not adequate whenever the final decision is made by KG curators. For humans, such scores are hard to understand or justify without explanations. Some approaches (*e.g.*, [12, 18]) attempt to show the sources used in computing the scores as an explanation. Yet, the collected syntactic clues using fact-spotting are often not sufficient since textual sources are *incomplete* and *biased* in what is stated explicitly. For instance, the citizenship of London's mayor *Sadiq Khan* would rarely be mentioned. In addition, some predicates (*e.g.*, *influencedBy*) are *ambiguous*, and their interpretation is domain-specific.

Proposed Approach. To better support KG curators in deciding the correctness of the candidate facts, we propose a novel framework for finding *semantically related evidence* in Web sources and the underlying KG, and for computing *human-comprehensible explanations* for facts. We refer to our framework, as ExFaKT (**Ex**plaining **Fa**cts over **K**Gs and **T**ext resources).

The key for detecting semantic evidence is intensional background knowledge in the form of *rules*, specifically, Horn rules of the form $H \leftarrow B_1, B_2, \ldots, B_n$. For example,

$$citizenOf(X, Y) \leftarrow mayorOf(X, Z), locatedIn(Z, Y)$$

intuitively states that mayors of cities are normally citizens of countries where these cities are located. Such rules can be specified by humans or automatically extracted from KGs using rule mining methods (*e.g.*, [17, 39]). As the latter may fall short of covering all interesting situations, hand-crafted rules are a valuable asset. We performed a user study with university students who were novices to KGs, and obtained a good number of rules with low error rate in less than an hour (Section 4.6).

We utilize rules to decompose a fact-spotting query into more frequently stated and thus easier-to-spot related facts. This way, we counter the reporting sparseness and bias. Moreover, rules can encode domain-specific knowledge to better cope with ambiguous predicates. Finally, rules combine knowledge from both textual Web sources and the KG. For example, a rule could find the mayors of cities in news articles and look up the countries of cities in the KG.

Given a set of rules and a query for a fact candidate, ExFaKT rewrites the query into a set of subqueries. Whenever we find evidence in the KG or text that the body of the rule holds, the credibility of the head increases. This process creates semantic traces that *explain*, in a human-readable format, why a fact is likely to be true (or false).

A key difference between our setting and existing applications of query rewriting (*e.g.*, [34]) is that the latter assume that the data is contained exclusively in an indexed KG. This is in contrast to our scenario of interest, where some information partially exists in the KG, while the remaining pieces of knowledge are to be extracted from large and noisy text sources on-the-fly. Accounting for this, our framework is particularly tailored towards reducing the cost and uncertainty of the retrieval procedures.

Contributions. Our contributions are summarized as follows:

- We introduce ExFaKT, a framework for computing semantic traces for facts in question from both KG and an implicit external source in the form of a text corpora by utilizing Horn rules.

- We develop optimization strategies, whose target is an automatic search for an effective rewriting plan based on our cost model.

- We evaluate ExFaKT over real-world KGs and rules from various sources to illustrate the effectiveness of our rewriting strategy. We also show the benefits of the computed explanations in supporting human fact-checkers, and the viability of exploiting explanations in automated fact-checking.

ExFaKT is available at https://www.mpi-inf.mpg.de/impact/exfakt.

2 PROBLEM STATEMENT

We start with providing background on KGs, rules, and fact-spotting. Then, we describe the problem of using them along with text in computing human-comprehensible explanations for query facts.

Knowledge Graphs. Let $\mathcal{E}$ and $\mathcal{R}$ be fixed sets of entities and relations respectively. A knowledge graph (KG) $\mathcal{G}$ is a repository of unary and binary facts (i.e., $p(s)$ and $p(s, o)$) encoding domain knowledge (e.g., $director(nolan)$, $influencedBy(nolan, lucas)$), where $p \in \mathcal{R}$ is a relation (*predicate*) and $s, o \in \mathcal{E}$ are entities (*constants*).

Rules. To enable constructing explanations from KGs and other resources, we utilize sets of Horn rules, which we define below by borrowing the concepts from the Datalog language [1].

A Horn rule is an expression of the form

$$H \leftarrow B_1, \ldots, B_n \tag{1}$$

where $H, B_1, \ldots, B_n$ are the *atoms* of the rule, i.e., expressions of the form $p(X)$ or $p(X, Y)$, where $p \in \mathcal{R}$ and X, Y are either entities or variables. We refer to $head(r) = H$ and $body(r)=\{B_1, ..., B_n\}$ respectively as the *head* and *body* of r.

Example 2.1. For r from Section 1, $head(r) = \{citizenOf(X, Y)\}$, and $body(r) = \{mayorOf(X, Z), locatedIn(Z, Y)\}$. □

Rules encode commonsense knowledge, *e.g.*, capitals are located in countries or constraints, *e.g.*, a person cannot be born in two places.

Given a rule r of the aforementioned form (1) and a set I of facts, we define the set of facts inferred by r from I as

$$r(I) = \{H\theta \mid B_1\theta, \ldots, B_n\theta \in I\}, \tag{2}$$

where θ is a postfix operator which substitutes variables with constants. We denote by ϵ empty substitutions, i.e., $q\epsilon = q$ for any q. Moreover, $\Pi(I) = \bigcup_{r \in \Pi} r(I)$ is the extension to inferences produced by all rules in Π. The output of multiple rule executions is recursively defined by setting $\Pi^0(I) = I$ and $\Pi^{i+1}(I) = \Pi(\bigcup_{j \in \{0, \ldots, i\}} \Pi^j(I))$. The set $\Pi^\infty(I)$ (called *closure*) contains all possible inferences derived using Π on I.

Fact Spotting. Let $textspot$ be a textual fact-spotting procedure, which gets as input an atom q and a set $\mathcal{T}$ of textual documents such as Wikipedia ($textspot(q, \mathcal{T})$), and outputs a set of tuples of the form $\langle \theta, s \rangle$, where $q\theta$ is a fact spotted in the text and s is a textual string containing this fact.

Example 2.2. For the query $q = directed(lucas, star\_wars)$ and the text $\mathcal{T} = \{$"G. Lucas, the director of Star Wars, signs..", "Nolan got inspired by Star Wars", "Star Wars 1977 directed by Lucas.."$\}$, $textspot(q, \mathcal{T})$ returns $\{\langle \theta, s = $"G. Lucas, the director of Star Wars.."$\rangle$, $\langle \theta, s =$"Star Wars 1977 directed by Lucas.."$\rangle\}$, where $\theta = \epsilon$. □

We define the set of all facts involving entities from $\mathcal{E}$ and relations from $\mathcal{R}$ which can be extracted from the text $\mathcal{T}$ using the syntactic fact-spotting procedure $textspot$ by $I_{\mathcal{T}} = \{p(s, o) \mid s, o \in \mathcal{E}, p \in \mathcal{R} \wedge textspot(p(s, o), \mathcal{T}) \neq \emptyset\}$.

Fact Explanations. Given a fact in question, a KG, text, and rules, our goal is to compute a set of semantic traces or *explanations* as we call them for a given fact, which are formally defined as follows:

Definition 2.3 (Explanation). Given a fact q, a KG $\mathcal{G}$, text corpus $\mathcal{T}$ and a ruleset Π, a set $E \subseteq \mathcal{G} \cup I_{\mathcal{T}}$ of facts is an *explanation* for q w.r.t. $\Pi, \mathcal{G}, \mathcal{T}$ if $q \in \Pi^\infty(E)$.

The presence of unstructured textual resources in the input makes the problem of computing explanations particularly challenging. Naturally, a given fact q might have multiple explanations or a single *trivial* one, i.e., q itself. Obviously, explanations as defined above may be subsumed by others. Among all explanations, ideally, we aim at computing non-trivial ones that are

(D1) concise, i.e., contain a small number of atoms;

(D2) close to the query, i.e., obtained by using few rules;

(D3) reliable, i.e., contain as many facts from the KG as possible, since KGs are usually more reliable than text.

3 EXFAKT FRAMEWORK

ExFaKT utilizes rules to compute explanations over the content of KGs and textual resources. We first start with an overview of how these explanations are computed. Then, we illustrate the technical details underlying our framework.

3.1 Computing Explanations

Computing the entire KG closure is not feasible as it requires the extraction of *all* possible facts from $\mathcal{T}$. Thus, we proceed *backwards*, *i.e.*, we start from the input fact, and check whether any of the rules can potentially produce such derivation. If so, then we move to the body of the rule, and search for possible instantiations for each body atom. This triggers a recursive process, where the new input is constituted by the body atoms. The recursion stops in case no rules can be found or all atoms are instantiated either by facts in the KG or by text. In this last case, the rules produce new derivations which are returned to earlier recursive calls.

We illustrate the overview of computing explanations using the following example:

Example 3.1. Consider the following input

- a query $q = influencedBy(nolan, lucas)$
- a text corpus $\mathcal{T}$ composed of Wikipedia articles
- a KG $\mathcal{G} = \{directed(lucas, star\_wars), isDirector(nolan),$
  $directed(lucas, amer\_graffiti)\}$
- a ruleset $\Pi = \{r_1, r_2\}$, where
  $r_1: influencedBy(X, Y) \leftarrow isDirector(X), directed(Y, Z),$
  $inspiredBy(X, Z);$
  $r_2: inspiredBy(X, Y) \leftarrow liked(X, Y), isArtist(X).$

As $q \notin \mathcal{G}$, and assuming that $textspot(q, \mathcal{T}) = \emptyset$, ExFaKT utilizes those rules in Π, such that $Head(r) = influencedBy$ to explore the potential explanations resulting in the set:
$P = \{\{isDirector(nolan), directed(lucas, Z), inspiredBy(nolan, Z)\}\}$.
For this candidate, it holds that $isDirector(nolan) \in \mathcal{G}$, hence, we move to the second atom $directed(lucas, Z)$. Grounding this atom from the KG results in the updated set of candidate explanations

$P = \{\{isDirector(nolan), directed(lucas, star\_wars),$
$inspiredBy(nolan, star\_wars)\},$

$\{isDirector(nolan), directed(lucas, amer\_graffiti),$
$inspiredBy(nolan, amer\_graffiti)\}\}$.

We start processing the first explanation, and since the first two atoms are in $\mathcal{G}$, they are marked as *found*, and we move to the third atom $inspiredBy(nolan, star\_wars)$. As this third atom is spotted in the text, we get $E_1 = \{isDirector(nolan), directed(lucas, star\_wars), inspiredBy(nolan, star\_wars)\}$. Since all atoms in E_1 were *found*, it is added to the output set.

However, since the last atom in E_1 was found in text which is a noisy resource, we would still rewrite it, seeking for more reliable evidences. In this case, r_2 is used for rewriting leading to $E_2 = \{isDirector(nolan), directed(lucas, star\_wars), isArtist(nolan), liked(nolan, star\_wars)\}$. Assuming that the last two atoms of E_2 are spotted in $\mathcal{T}$, we get E_2 to be in the output set.

Analogously, we process the second explanation candidate in P, *i.e.*, $\{isDirector(nolan), directed(lucas, amer\_graffiti), \dots\}$. □

ExFaKT explanations are human-interpretable; allowing KG curators to judge the validity of the fact. They can also be used as input features to automatically assess the candidate fact truthfulness.

3.2 Prerequisites

Besides the KG, ExFaKT requires two key resources, rules and fact-spotting engine over textual sources, which are obtained as follows:

Ruleset Acquisition. In general, rules can be either automatically extracted using tools such as [17, 37, 39], or manually specified by domain experts as in the context of ontology engineering [4]. We performed experiments to evaluate the effectiveness of using rulesets from both sources. We have also conducted a pilot experiment to ensure the feasibility of manual rule construction where we asked non-experienced participants to create useful rules. From these experiments, we observed that adequate rules can be produced not only by KG curators (which are our target group), but also by non-experts (more details in Section 4).

Fact-spotting Realization. ExFaKT utilizes the syntactic fact-spotting subroutine *textspot*, defined above. In practice, similar to [26, 32], it is implemented relying on a textual-search-based method, in which the SPO query q is converted to textual representation (*i.e.*, verbalization) using paraphrasing dictionaries for relations such as PATTY [27] and entity-mentions dictionaries for entity name aliases. Then, q with its paraphrasing is issued to get the documents containing it. Finally, a named entity recognizer, *e.g.*, [14], is utilized to collect entities from the documents and compute the substitution θ. This versatile approach is easily scalable without extensive training, but any other fact-spotting method can easily be plugged in ExFaKT.

3.3 ExFaKT Algorithm

We identify two main operations in ExFaKT's recursive process, namely, *bind* and *rewrite*. The first retrieves answers for a given query from the underlying data sources, while the second rewrites a query into subqueries. We formally specify them below.

Bind. Prior to defining *bind*, we introduce some formal notations. Let g be an atom, $\mathcal{G}$ a KG, and $\mathcal{T}$ a text corpus. We define $\Sigma_{\mathcal{G}}(g) = \{\sigma \mid g\sigma \in \mathcal{G}\}$ and $\Sigma_{\mathcal{T}}(g) = \{\sigma \mid textspot(g\sigma, \mathcal{T}) \neq \emptyset\}$ as the sets of substitutions that result in answers to the query g from the KG and text respectively. Every substitution $\sigma \in \Sigma_{\mathcal{G}}(g)$ is annotated with the metadata $source[\sigma] = \text{KG}$, while every $\sigma' \in \Sigma_{\mathcal{T}}(g)$ is annotated with $source[\sigma'] = \text{TEXT}$; moreover, each σ' is annotated with another metadata $text[\sigma_{\mathcal{T}}]$ which contains the string that mentions g in $\mathcal{T}$ (as returned by *textspot*).

Metadata of substitutions is passed to atoms they substitute, e.g., if $text[\sigma] = \text{X}$ then $text[g\sigma] = \text{X}$. In our procedures, we use another two types of metadata, $status[.]$, to mark atoms which are already processed, and $depth[.]$, to record the number of rewritings that led to the atom at hand. Finally, metadata is not considered for the equality of substitutions and atoms, i.e. two atoms can be equivalent even if they are annotated with different metadata.

Definition 3.2. The function $bind(g, \mathcal{G}, \mathcal{T})$ receives as input an atom g, a KG $\mathcal{G}$, and a text corpus $\mathcal{T}$. It returns in output the set $\Sigma = \{\Sigma_{\mathcal{G}}(g) \cup \{\Sigma_{\mathcal{T}}(g) \setminus \Sigma_{\mathcal{G}}(g)\}\}$.

Algorithm 1: Algorithm for computing explanations.

Input: fact q, KG $\mathcal{G}$, text corpus $\mathcal{T}$, ruleset Π, nonegative parameter
$\quad max\_depth$ for ensuring termination
Output: set of explanations O
1 **function** $explain(q, \mathcal{G}, \mathcal{T}, \Pi)$
2 $\quad depth[q] \leftarrow 0; status[q] \leftarrow$ TODO; $P \leftarrow \{\{q\}\}; O \leftarrow \{\}$
3 $\quad$ **while** $P \neq \emptyset$ **do**
4 $\quad\quad$ Pick explanation E from P (i.e., $E \in P$)
5 $\quad\quad P \leftarrow P \setminus \{E\}$
6 $\quad\quad$ **if** $status[g] =$FOUND *for all* $g \in E$ **then**
7 $\quad\quad\quad O \leftarrow O \cup \{E\}$ ▷We found a valid explanation.
8 $\quad\quad$ **else**
9 $\quad\quad\quad$ Pick an atom g from E s.t. $status[g] =$TODO
10 $\quad\quad\quad NT \leftarrow process\_goal(g, E \setminus \{g\}, \mathcal{G}, \mathcal{T}, \Pi)$
11 $\quad\quad\quad P \leftarrow P \cup NT$
12 $\quad$ **return** O
13 **function** $process\_goal(g, E, \mathcal{G}, \mathcal{T}, \Pi)$
14 $\quad O \leftarrow \emptyset$
15 $\quad \Sigma \leftarrow bind(g, \mathcal{G}, \mathcal{T})$
16 $\quad TR \leftarrow \{g\}$
17 $\quad$ **for** $\sigma \in \Sigma$ **do**
18 $\quad\quad a \leftarrow g\sigma$
19 $\quad\quad depth[a] \leftarrow depth[g]; status[a] \leftarrow$ FOUND
20 $\quad\quad$ **if** $source[\sigma] =$KG **then** $TR \leftarrow TR \setminus \{a\}$
21 $\quad\quad O \leftarrow O \cup \{E\sigma \cup \{a\}\}$
22 $\quad$ **for** $gr \in TR$ s.t. $depth[gr] < max\_depth$ **do**
23 $\quad\quad O \leftarrow O \cup rewrite(gr, E, \Pi)$
24 $\quad$ **return** O

Example 3.3. Let us assume that $g = directed(lucas, Z)$, $\mathcal{G} = \{directed(lucas, star\_wars)\}$ and $\mathcal{T} =$"Along with *Star Wars,* Lucas has directed short documentaries *Herbie,* ...". Then, $\Sigma_\mathcal{G} = \{\sigma_1\}$, $\Sigma_\mathcal{T} = \{\sigma_2\}$, where $\sigma_1 = \{Z \rightarrow star\_wars\}$, $\sigma_2 = \{Z \rightarrow herbie\}$, Also, $source[\sigma_1] =$KG, $source[\sigma_2] =$ TEXT, and $text[\sigma_2]$, store string coordinates of films in $\mathcal{T}$. □

Rewrite. The function *rewrite* moves the search space of the answers for a given query from the rule's head to its body by rewriting an input query into a respective set of subqueries.

Definition 3.4. The function *rewrite(g,E,Π)* receives as input an atom g, a set of atoms E and a program Π. It computes the set $B_g = \{body(r)\sigma_g\sigma_E | r \in \Pi \wedge head(r)\sigma_g = g\}$ where σ_g is a substitution that unifies the rule head with g and propagates the substituted terms to the rule body, while σ_E renames among the rest of the body variables those that appear in E into fresh ones. Moreover, for each atom a in B_g the function sets $status[a] =$TODO, $depth[a] = depth[g] + 1$ and returns the set $\{E \cup a \mid a \in B_g\}$.

Example 3.5. For the input predicate $g = inspiredBy(nolan, Z)$, the intermediate explanation $E = \{isDirector(nolan), directed(Y, Z)\}$ and the ruleset $\Pi = \{r_2: inspiredBy(X, Y) \leftarrow liked(X, Y), isArtist(X)\}$, the function $rewrite(g, E, \Pi)$ first computes $B_g = body(r_2)\sigma_g\sigma_E$, where $\sigma_g = \{X \rightarrow nolan, Y \rightarrow Z\}$ and $\sigma_E = \epsilon$, since σ_g already handled all variables appearing in $body(r_2)$. Finally, in the output we obtain $E = \{isDirector(nolan), directed(Y, Z), liked(nolan, Z), isArtist(nolan)\}$, where for every atom $a \in B_g$, the status is assigned as $status[a]=$TODO, and the depth as $depth[a] = depth[g] + 1$. □

Main Procedure. Algorithm 1 shows the main procedure underlying ExFaKT. Our procedure takes as input a query q, a KG $\mathcal{G}$, a text corpus $\mathcal{T}$, a ruleset Π and a global parameter max_depth specifying a maximum number of allowed rewritings to ensure termination, and as output provides a set O of explanations for q. Initially, the

status of the input query is set to TODO, and the set P of potential explanations is initialized with $\{q\}$. In (3) the algorithm iterates over the set P of potential explanations. Explanations $E \in P$, all of whose atoms have status FOUND, are moved to the output set O in (7). For other candidate explanations, atoms with the status TODO are processed by the procedure *process_goal*.

Procedure *process_goal* takes as input an atom g and first retrieves all substitutions of variables to constants that result in answers to g in the KG and text using *bind* function and copies g into the set TR (to be rewritten). Then, it iterates over the retrieved substitutions and sets the depth of every obtained answer (a) to the one of g and the status to FOUND in line (19). After that, the answer is added to the existing atom in the current explanation (E), which is included in O (line 21).

Lastly, *process_goal* rewrites the input query using the rules in the program by invoking the function *rewrite*. Notice that TR might be empty: This occurs if g is a fact, which was verified in the KG. In this case, we do not need to rewrite it, since the fact is already explicitly stated; thus we remove it from TR in line (20).

Note that since no restriction is put on the form of allowed rules, Algorithm 1 might not terminate. To avoid this, we bound the number of rewritings using the *max_depth* parameter, so that the algorithm always terminates. As output, Algorithm 1 returns explanations for the input candidate, which is a property formally stated in the following theorem:

THEOREM 3.6. *Let* $\mathcal{G}$ *be a KG,* $\mathcal{T}$ *a text corpus,* Π *a program, q an input fact, and* $O = explain(q, \mathcal{G}, \mathcal{T}, \Pi)$. *If* $E \in O$ *then* E *is an explanation of q w.r.t.* $\Pi, \mathcal{G}, \mathcal{T}$.

Optimizations. Often, we do not need to calculate all explanations; few relevant ones are sufficient for establishing the truth value of a fact in question. To improve the performance, we introduce anytime behavior and incrementally collect new explanations as they are added to the output set (line 7 of Algorithm 1). In this new setting, we can stop the algorithm whenever it has returned satisfactory explanations, but it is crucial that the most relevant explanations based on the criteria **(D1)-(D3)** from Section 2 are computed first.

Example 3.7. If $influencedBy(nolan, lucas)$ from Ex. 3.1 was found in text, then $E_0 = \{influencedBy(nolan, lucas)\}$ would be the most *concise* explanation, since it contains only a single atom. Moreover, for E_1 and E_2 from the same example we have that E_1 is *closer to the query* than E_2, since less rules were used to produce E_1. If there is another explanation E_3 which equals to E_2 but with *star_wars* substituted by *Herbie* and $directed(lucas, herbie)$ is found in text, it holds that E_2 is more *reliable* than E_3, since E_2 contains fewer atoms from text sources. □

Our optimizations along these lines affect two operations: the *explanation selection* and the *atom selection criteria*.

Explanation Selection Criterion. In line 4, Algorithm 1 selects one explanation to be processed from those in P. To prioritize the processing of promising explanations we change the order in which they are picked based on the following criteria: (i) we favor *shorter explanations*, i.e., explanations with the lowest number of atoms **(D1)**; (ii) we favor *explanations produced with fewer rewritings* by picking the query with the smallest *depth* value **(D2)**.

Atom Selection Criterion. In line 9 of Algorithm 1, a naive strategy would always pick the first atom in E; resulting in a huge search space. To counter this, we first process atoms without variables. Then, we pick atoms with some constants and keep those with only variables till the end. Moreover, to favor explanations that can be proven from the KG **(D3)**, we prefer atoms with KG substitutions.

4 EVALUATION

We start in Section 4.2 with evaluating the effectiveness of ExFaKT with respect to the increase in the coverage of the collected pieces of evidence while preserving their precision (as shown later in Section 4.3). Additionally, we present a user-study which targets assessing the quality of the explanations and their readability and usefulness for human-reviewers in Section 4.3. Later in Section 4.4, we demonstrate the effectiveness of ExFaKT using automatically mined rules as input. Moreover, a showcase for integrating ExFaKT into the pipeline of a fully automated fact-checking system is given in Section 4.5. Finally, Section 4.6 reports the results of our study on the feasibility of manual rule construction.

4.1 Experimental Setup

Datasets. We conducted our experiments on two datasets:

- **YAGO-based** benchmark, which consists of 300 true candidate facts, uniformly distributed over six relations (listed in Table 1). The instances of the first three relations were randomly selected from YAGO [36]. The other three relations do not appear in the KG; hence, their instances were semi-automatically curated using simple logical rules. Then, 50% of the facts used during their creation were removed at random; thus, intentionally introducing incompleteness in the KG and hence the need for textual sources.

- **DBpedia-based** benchmark is a subset of the dataset proposed by [35] containing facts over the predicates, for which AMIE, a state-of-the-art rule mining system [16], managed to learn at least 5 rules having them in the head. This benchmark contains four predicates with a total of 1763 correct facts.

Rules. We constructed a ruleset for each benchmark as follows: For the *YAGO-based* benchmark, we selected rules from the top-ranked ones mined by AMIE from YAGO, and added further rules with new predicates that do not exist in the KG. We refer to these new predicates as *text-based*, as they need to be verified from the text. This way we obtained 20 rules on average for each head predicate. For the *DBpedia-based* benchmark, we used the rules mined by AMIE from DBpedia without altering them. Each head predicate in the dataset obtained a set of 100 related rules on average. This setup is designed to study the case of fully relying on automatically learned rules.

Knowledge Graph. We used YAGO3 [36] as *KG* in all experiments. YAGO3 contains around 5.5M facts and 35 relations. This KG is geared towards precision rather than recall, allowing us to treat it as a trusted resource.

Text Corpora. As for text, we experimented with two different sources: (i) *Wiki* which contains 5.5M Wikipedia articles, whose textual parts were split into sentences and indexed as separate documents using Elasticsearch [19] and (ii) *Web* constructed relying on the Bing API for searching in Web pages.

Table 1: Recall of baselines vs. ExFaKT configurations

	B-Wiki	B-Web	KG	Wiki	Web	KG+Wiki	KG+Web
influences	0.30	0.24	0.00	0.38	0.88	0.42	0.92
isPolitOf	0.02	0.16	0.26	0.18	0.88	0.42	0.92
wroteMusic	0.08	0.28	0.00	0.10	0.72	0.24	0.78
mayorOf	0.66	0.9	0.00	0.66	0.90	0.66	0.90
actedWith	0.26	0.52	0.18	0.26	0.60	0.54	0.94
countryWon	0.18	0.38	0.00	0.18	0.38	0.70	0.92
Total	**0.25**	**0.41**	**0.07**	**0.29**	**0.73**	**0.50**	**0.90**

Baselines. We compared against three baselines:

- **B-Wiki**: a method that syntactically spots candidate facts and their paraphrases in the *Wiki* corpus.

- **B-Web**: a method that retrieves fact occurrences in the Web using the Bing search API and post-filters the results to obtain the relevant text snippets (this baseline was extracted from [32]).

- **B-Search**: a simulation for a user issuing verbalized versions of candidate facts to commercial search engine and retrieving the top-5 results.

Configurations. To analyze the influence of the various sources, we ran ExFaKT with the following

- **KG**: Rules are used over the KG facts only.

- **Wiki**: Rules are used only with syntactic fact-spotting over the *Wiki* corpus, i.e., no KG facts are exploited.

- **Web**: Rules are used only with fact-spotting over the *Web*, i.e., no KG facts are used.

- **KG+Wiki**: Rules used together with both *Wiki* corpus and KG.

- **KG+Web**: Both *Web* corpus and KG are used with the rules.

In all of the experiments, we set the parameter *max_depth* to 5 unless otherwise stated.

4.2 Explanations Coverage

Experimental Details. In this experiment, we show the effectiveness of ExFaKT in retrieving more explanations for candidate facts. We used the *YAGO-based* dataset and compared all five configurations of our method against the baselines. We computed the **recall** of each configuration as the ratio of the queries for which at least one explanation was retrieved.

Results. Table 1 reports the recall for each predicate in the dataset and the recall of the whole dataset as the total. These results show that ExFaKT configured with *KG+Wiki* or *KG+Web* almost doubled the recall of the baselines *B-Wiki* and *B-Web* respectively, whereas rules on the KG alone (*KG*) have the worst recall. Similarly, configurations over text corpora alone, namely *Wiki* or *Web* could not compensate the absence of the KG.

Observe that since *mayorOf* is prominent enough to be easily spotted in the text, there is no increase in the recall for this predicate. In contrast, our method is particularly successful for *countryWon*, since the baselines fail to spot facts over this newly created predicate. Overall, the results demonstrate that while neither the KG nor textual sources alone are sufficient to collect strong evidence, their combination doubles the recall, and is doubtlessly the best configuration. Moreover, comparing *KG+Wiki* with *KG+Web* shows that increasing the size of the textual corpus enhances the results.

Table 2: Examples of explanations produced by ExFaKT

Fact candidates	Explanations
countryWon(*guatemala*, *nobel*)	*isCitizenOf*(*miguel_asturias*, *guatemala*), *source* = TEXT, text="Asturias is a Nobel Prize-winning Guatemalan poet…" *hasWonPrize*(*miguel_asturias*, *nobel*), *source* = KG
influences(*s_fitzgerald*, *r_yates*)	*wrote*(*s_fitzgerald*, *the_great_gatsby*), *source* = KG *read*(*r_yates*, *the_great_gatsby*), *source* = TEXT, text="Yates, called "The Great Gatsby" the most nourishing novel he read."

Table 3: Mechanical Turk task statistics

Config	Candid.	Explan.	Question 1			Question 2	
			Yes	No	Cannot	Yes	No
B-Wiki	75	75	0.87	0.04	0.10	0.90	0.10
B-Web	122	122	0.85	0.0	0.15	0.80	0.20
B-Search	228	228	0.58	0.01	0.42	0.55	0.45
KG+Wiki	159	311	0.64	0.35	0.02	0.63	0.37
KG+Web	267	1021	0.82	0.01	0.17	0.74	0.26

Examples. As anecdotal evidence, Table 2 shows two examples of candidate facts for which our approach managed to compute supporting explanations even though fact-spotting failed to find direct mentions for them. The first fact to be spotted is about the country *Guatemala* and the predicate *country has won prize*. Our method was able to find positive evidence for Guatemala winning the Nobel prize by spotting the Nobel laureate Miguel Asturias, and combining this information with the fact that he is a citizen of Guatemala.

In the second example, we were able to extract an evidence about the influence of a writer on another one by spotting the fact that the latter read books written by the former. Note that here the relation *read* is not present in the KG but mentioned in the Wikipedia text.

4.3 Explanations Quality and Usefulness

In this experiment, we evaluate the quality of the retrieved explanations by estimating the precision of the results and their readability based on human judgment.

Experimental Details. We designed a Mechanical Turk [6] task to collect human judgments on the quality of the computed explanations. To facilitate readability, we translated our task into a natural language question. The participants were shown a candidate fact and some extracted explanation (*i.e.*, in a human readable format), and their task was to judge the correctness of the fact relying only on the provided information.

Since it may be hard for non-experts to give a solid judgment, the participants were asked to choose one out of five answers: (i) *"For sure, yes"*, (ii) *"Probably, yes"*, (iii) *"Probably, no"*, (iv) *"For sure, no"*, or (v) *"Can not judge"*. As a confirmation, they were also asked for their explicit feedback on the usefulness of the provided information. Participants were provided with a set of instructions and clarification examples covering all possible cases. Each record was assigned to 5 different participants.

We evaluated the top-5 explanations produced by ExFaKT's configurations *KG+Wiki* and *KG+Web* on the *YAGO-based* dataset as reported in Section 4.2. We compared them to the results of the baselines *B-Wiki* and *B-Web* respectively. Additionally, we included the results of the third baseline *B-Search* for annotations.

Evaluation Metrics. The target of this experiment is to assess the relevance of the extracted traces for the query by relying on
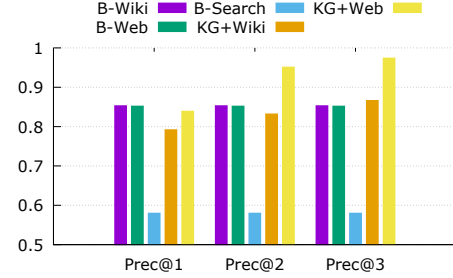

Figure 1: Precision@k computed using human annotations

whether participants were able to make a correct judgment. To this end, we used the annotations given by the participants to perform majority voting on three categories: *correct judgment* (*i.e.*, yes case) *incorrect judgment* (*i.e.*, no case), and *unable to judge*. Tie cases are randomly broken. Explanations with a majority of the *correct judgment* are counted as relevant.

We calculated **Precision@K**, which is defined as the ratio of candidate facts for which our method returned one or more *relevant* explanations to those that have at least one (relevant or irrelevant) explanation. Finally, we computed the f_1 **score** and **the mean average precision** at *top-5* (*MAP@5*).

Results. Table 3 reports the sizes of each dataset and the distribution of the majority voting over the classes. The second column shows the number of candidate facts involved in the evaluation, while the third column indicates the total number of explanations produced for all of the candidates. The annotations demonstrate a fair Fleiss Kappa [20] agreement of 0.35 for the Wiki-based configurations and a slight agreement of 0.03 for Web-based configurations, reflecting the difficulty and subjectivity of the task in the presence of noisy results. Deeper analysis showed that the lower the *ratio of atoms binded from text*, the easier it is to judge the explanation correctly. While reducing the number of rules has a similar effect, increasing the the number of atoms in the explanation increases human-comprehensibility of explanations.

Observe that the average time required for the annotator to judge the truthfulness of a fact candidate based on explanations retrieved by our method (27 seconds) was almost half the time required to judge it based on standard Web search results (51 seconds), yet the quality of the judgments in the former case is higher. This illustrates the benefits of using our method for increasing the productivity and accuracy of human fact-checkers.

Figure 1 shows the results for *Prec@k* for $k = 1, 2, 3$. First, we observe that the Prec@1 of ExFaKT's configurations *KG+Wiki* and *KG+Web* are 6% and 2% lower than for the respective baselines *B-Wiki* and *B-Web* which both have precision of 0.85, yet all configurations have significantly higher precision compared to the case when a traditional commercial search engine was exploited as for

Table 4: Recall with automatically mined rules

	B-Wiki	KG	Wiki	KG+Wiki
vicePresident	0.96	1.00	0.96	1.00
diedIn	0.46	0.11	0.51	0.56
nationality	0.64	0.08	0.68	0.68
graduatedFrom	0.14	0.11	0.14	0.30
Total	**0.20**	**0.14**	**0.21**	**0.35**

B-Search. From the figure we can observe that *KG+Web* exceeds the baselines starting from top-2 with $Prec@2 = 0.95$ and *KG+Wiki* overcomes the baselines for the top-3 explanations. The results also indicate that the precision for the both configurations *KG+Wiki* and *KG+Web* consistently increase till they achieve $Prec@5$ of 0.87 and 0.97 respectively. Furthermore, $MAP@5$ for *KG+Web* is 0.84, which exceeds $MAP@5 = 0.77$ of *KG+Wiki*.

Since our method doubles the recall (Table 1), it also significantly enhances the F_1 score achieving 0.64 and 0.93 with *KG+Wiki* and *KG+Web* compared to 0.4 and 0.55 for the corresponding baselines.

4.4 Extracting Explanations using Mined Rules

So far we have considered manually refined rules. We now evaluate ExFaKT using automatically learned, hence noisier rules.

Experimental Details. We used the *DBpedia-based* dataset with the ruleset automatically mined by AMIE from DBpedia. These rules contain many predicates that are not in YAGO; thus, textual sources are vital. We compare the recall of *KG+Wiki* against the configurations *B-Wiki*, *KG*, and *Wiki*. For this experiment, we set the rewriting depth *max_depth* to 2 to reduce propagation of noisy rule rewritings. We computed $Prec@5$, $MAP@5$ and $f_1@5$ scores in the same way as in Section 4.3.

Results. Table 4 reports the recall for the fact candidates over four different predicates. We observe that our method improves the recall for all predicates, with the best result obtained for *graduatedFrom*, where the recall has even doubled. Based on human annotations, we obtained $Prec@5 = 0.84$, $MAP@5 = 0.84$ for *B-Wiki* and $Prec@5 = 0.64$, $MAP@5 = 0.56$ for *KG+Wiki*. Despite that, *KG+Wiki* configuration achieved a better f_1 with $f_1@5 = 0.45$ compared to a lower value $f_1@5 = 0.32$ obtained by *B-Wiki*. These results demonstrate the effectiveness of ExFaKT also when automatically mined rules are used as input.

4.5 Rule-based Automatic Fact-checking

In this experiment, we focus on the viability of exploiting extracted explanations in automated fact-checking. To this end, we implemented a simple automated fact-checker that uses the explanations to determine whether a given fact should be supported or rejected.

For the sake of computing numerical scores over explanations, we define the notion of explanation confidence as

$$confidence(E) = \frac{1}{|E|} \sum_{a \in E} \frac{trust(source[a])}{depth[a]}$$

where E is an explanation, $trust(\cdot)$ is 1 if $source[a]$ is KG, or 0.5 otherwise, representing the trust in the source, and $depth[a]$ is number of rewritings performed to obtain the atom as in Section 3. Then, for each fact candidate $f = p(a, b)$, we exploit two rules sets Π_+ for supporting it and Π_- for refuting it. Then, we compute the supporting explanations set $O_+ = explain(p(a, b), \mathcal{G}, \mathcal{T}, \Pi_+)$, and another set

Table 5: Rule-based fact checking vs prior methods

Method	DBpedia-Based		Politicians	
	Recall	Accu.	Recall	Accu.
ExFaKT (*KG+Wiki*)	0.68	0.93	0.83	0.81
TruthFinder [41]	0.66	0.97	0.73	0.79
Lang.-Stance-Credibility (LSC) [32]	0.99	0.59	1.00	0.55

of refutation evidence $O_- = explain(not\_p(a, b), \mathcal{G}, \mathcal{T}, \Pi_-)$ where not_p is newly introduced predicate representing the negation of p. Then, we compute the *truthfulness score* for f as:

$$truth\_score(f) = quality(O_+) - quality(O_-)$$

where $quality(\cdot)$ is the average confidence of explanations belonging to this set. The truth score acts as a threshold value which determines whether the fact should be accepted or rejected.

Dataset. We applied the automated fact-checking on two datasets containing both correct and erroneous candidate facts.

- *Politicians benchmark* [26] with 275 candidates, which contains for each true fact a set of its alternatives. For fairness, we removed the existing candidates from the KG which would trivially support the input facts. For this dataset, we used a ruleset of both manually specified and automatically mined rules by AMIE.

- The previously used *DBpedia-based* dataset enriched by adding a set of erroneous alternative instances from [35] for each true fact. In this case, we used the set of automatically learned rules from Section 4.4 extended with rules for not_p for each predicate p in the given dataset.

Experimental Details. The explanations were computed using the *KG+Wiki* configuration. We compare the results of our simple ExFaKT-based fact checker to two state-of-the-art approaches:

- *TruthFinder* [41] which uses a voting approach over the retrieved documents weighted based on their sources.

- *Language-Stance-Credibility (LSC)* [32] which decides the truthfulness of a claim based on language, stance, and reliability of the evidence sources.

We considered the first 5 explanations retrieved by the *KG+Wiki* configuration per supporting O_+ and refuting O_- sets respectively. Note that *LSC* collects clues from the whole Web, while *TruthFinder* and ExFaKT rely only on the Wikipedia text corpus.

Evaluation Metric. In order to compare the performance of the various approaches, we grouped the true facts with their alternatives and ranked the elements of the group using the scoring function of each method. For instance, let f_t be a true fact and $f_{f_1}, \ldots, f_{f_n}$ be alternative false facts. In this setting, each method assigns a score value to every fact thus producing a ranking. After the ranked list is computed, we compute the **recall** as the number of groups where the method at hand was capable of returning some truthfulness scores for the true fact. Then, for each given group $G_i = \{f_t, f_{f_1}, \ldots, f_{f_n}\}$, we compute the **accuracy** as $Accuracy(G_i) = \frac{1}{n} \sum_{f_{f_j} \in G} [score(f_t) \geqslant score(f_{f_j})]$ where $[\cdot]$ is the Iverson bracket and $score(f)$ is the truthfulness score computed by each method. This accuracy estimates the probability that the true fact f_t has the highest rank among its alternative facts as in [26].

Results. Table 5 shows the results of the average accuracy (*Accu.*) for all groups of facts and recall of our method compared to the

Table 6: Statistics for manually specifying rules experiment

	Strong	Valid	Invalid	Total
Supporting Rules	37	22	10	69
Refutation Rules	10	12	5	27
Total	47	34	15	96

competitors. According to the first two rows, we can observe that our method is on par with the accuracy of TruthFinder, despite the simplicity of our ranking function. Moreover, our method is advantageous, as it offers clear explanations of the results, and has a better recall compared to TruthFinder, especially for the politicians dataset. Moreover, importantly while LSC is utilizing the benefits of scrapping the whole Web, our system still achieves significantly higher accuracy. We analyzed the cases where our method failed, and observed that some of the relations in the test set require more complex rules to be properly supported or refuted. For instance, this dataset contains facts over such predicates as *isMarriedTo* or *holdsPosition*, which are better explained using temporal rules, as they change over time (e.g., one person can be married to multiple persons, but not at the same time). Extending our approach to support also rules of this kind is a promising future direction.

4.6 Rule Specification

In addition to automatically mined rules, our approach benefits from hand-crafted rules. In this experiment, we examine the feasibility and cost of this kind of rule specification.

Experimental Details. We asked 10 students from different fields to create rules. Most of these had no prior exposure to logical rules or KGs; so we gave them 20 minutes of explanation on KGs and Horn rules. Each participant picked 5 predicates from a list of KG predicates, and was asked to write at least one supporting rule and at least one refuting rule. The participants were given 30 minutes to create the rules.

Evaluation Metric. We classified the resulting rules into three categories: (i) *strong rules* that represent causality and generalize (*e.g.*, $politicianOf(X, Y) \leftarrow electedIn(X, Z), in(Z, Y).$); (ii) *valid rules* that capture typical correlations but may be tied to specific cases (*e.g.*, $citizenOf(X, Y) \leftarrow grewuUpIn(X, Y)$); (iii) *invalid rules* that are logically flawed or do not properly reflect typical situations (*e.g.*, $not\_memberOf(X, Y) \leftarrow leader(X, Y)$). We consider *strong* and *valid* rules as suitable.

Results. Within 30 minutes, the 10 participants created 96 rules for 23 different head predicates (*i.e.*, with average of 3 minutes per rule). Table 6 shows their distribution over the three categories. More than half of the rules were strong, and more than 80% were strong or valid. The remaining incorrect rules could be filtered out by having the same participants judge the validity of each others' rules, using a voting scheme. This study clearly demonstrates that manual construction of rules is not a bottleneck, and can be accomplished by informed crowdsourcing at fairly low cost.

5 RELATED WORK

Fact-checking. Starting with TruthFinder [41] and T-Verifier [23], text-based fact-checking has become an established research field [24]. The majority of the state-of-the-art methods (e.g. [11, 18, 22, 25, 26, 32]) perform joint estimation of the fact's truth value based on the credibility of the fact candidates syntactically spotted in text and the trustworthiness of the underlying sources. These approaches, unlike ours are purely syntactic, i.e., they ignore domain background knowledge often vital for inferring the fact in question.

A query language to assess the validity of claims in multiple contexts over structured information sources has been proposed in [21]. Other works (e.g., [2, 9, 33, 35]) estimate the support for factual statements by mining and ranking connectivity patterns in KGs. In contrast to ours, all of these approaches ignore textual sources. The inclusion of textual sources make the execution of bottom-up procedures (like the well-known chase in [21]) problematic, as they require the execution of fact-spotting for any possible fact, which is unfeasible due to the high cost of this procedure. This problem has been circumvented by our top-down approach. The same holds for less related works that explain semantic connections among given KG entities (*e.g.*, [3, 15]).

Checking facts jointly over KGs and textual resources has been explored in the context of natural language question answering in, e.g., [10], where given a relation and an entity (i.e., subject or object), the task of retrieving a set of other entities that form a true triple is solved using neural networks. Our work differs from [10] in several aspects. First, instead of retrieving entities we are given a fact to be checked. Second, we do not merely estimate the truthfulness of the fact, but also output human-readable semantic traces that support or refute it, while the output of [10] cannot be explained to humans.

Inductive and Deductive Reasoning over KGs. Mining rules for KG completion (e.g., [8, 16, 37]) is orthogonal to our work. Indeed, instead of inducing rules from KGs, we are rather interested in exploiting them effectively for fact-checking over both structured and unstructured (textual) resources.

Logical query rewriting is well-studied in databases, e.g., for the Datalog language. This line of research (see [7] for an overview) focused on structured databases, though. In contrast, we consider also external sources in text form, which introduces additional challenges and requires optimizations. Deductive reasoning over multiple external sources has been explored, e.g., in [13]. However, joint query rewriting over structured and textual sources has not been studied in this context.

6 CONCLUSION

Prior methods for automatic fact-checking focused on producing final truthfulness scores, which are hard to interpret for humans. Moreover, existing approaches spot solely explicit occurrences of facts in text. In this paper, we moved forward towards deriving more human understandable evidence based on background knowledge in the form of rules. ExFaKT utilizes these rules to collect pieces of evidence from both KGs and textual sources. The conducted experiments demonstrate the usefulness of our method for supporting human curators in making accurate decisions about the truthfulness of facts as well as the potential of our explanations for improving automated fact-checking systems.

REFERENCES

[1] Serge Abiteboul, Richard Hull, and Victor Vianu. 1994. *Foundations of Databases*. Addison Wesley.

[2] Nitish Aggarwal, Sumit Bhatia, and Vinith Misra. 2016. Connecting the Dots: Explaining Relationships Between Unconnected Entities in a Knowledge Graph. In *The Semantic Web - ESWC 2016 Satellite Events*. 35–39.

[3] Marcelo Arenas, Gonzalo I. Diaz, and Egor V. Kostylev. 2016. Reverse Engineering SPARQL Queries. In *Proceedings of WWW 2016*. 239–249.

[4] Oscar Corcho Asuncion Gomez-Perez, Mariano Fernández-López. 2005. *Ontological Engineering: with examples from the areas of Knowledge Management, e-Commerce and the Semantic Web*. Springer Science and Business Media.

[5] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary G. Ives. 2007. DBpedia: A Nucleus for a Web of Open Data. In *Proceedings of ISWC*. 722–735.

[6] Michael Buhrmester, Tracy Kwang, and Samuel D. Gosling. 2011. Amazon's Mechanical Turk: A New Source of Inexpensive, Yet High-Quality, Data? *Perspectives on Psychological Science* 6, 1 (2011), 3–5.

[7] Stefano Ceri, Georg Gottlob, and Letizia Tanca. 1989. What you Always Wanted to Know About Datalog (And Never Dared to Ask). *IEEE Transactions on Knowledge and Data Engineering* 1, 1 (1989), 146–166.

[8] Yang Chen, Sean Goldberg, Daisy Zhe Wang, and Soumitra Siddharth Johri. 2016. Ontological Pathfinding: Mining First-Order Knowledge from Large Knowledge Bases. In *Proceedings of SIGMOD/PODS 2016*. ACM, 835–846.

[9] Giovanni Luca Ciampaglia, Prashant Shiralkar, Luis Mateus Rocha, Johan Bollen, Filippo Menczer, and Alessandro Flammini. 2015. Computational fact checking from knowledge networks. *CoRR* abs/1501.03471 (2015).

[10] Rajarshi Das, Manzil Zaheer, Siva Reddy, and Andrew McCallum. 2017. Question Answering on Knowledge Bases and Text using Universal Schema and Memory Networks. In *Proceedings of ACL 2017*. 358–365.

[11] Xin Luna Dong, Evgeniy Gabrilovich, Kevin Murphy, Van Dang, Wilko Horn, Camillo Lugaresi, Shaohua Sun, and Wei Zhang. 2015. Knowledge-Based Trust: Estimating the Trustworthiness of Web Sources. *PVLDB* 8, 9 (2015), 938–949.

[12] Xin Luna Dong and Divesh Srivastava. 2013. Compact Explanation of Data Fusion Decisions. In *Proceedings of the 22Nd International Conference on World Wide Web (WWW '13)*. ACM, 379–390.

[13] Thomas Eiter, Tobias Kaminski, Christoph Redl, Peter Schüller, and Antonius Weinzierl. 2017. Answer Set Programming with External Source Access. In *Reasoning Web. Semantic Interoperability on the Web*. 204–275.

[14] Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling. In *Proceedings of ACL*. 363–370.

[15] Valeria Fionda and Giuseppe Pirrò. 2017. Explaining and Querying Knowledge Graphs by Relatedness. *PVLDB* 10, 12 (2017), 1913–1916.

[16] Luis Galarraga, Christina Teflioudi, Katja Hose, and Fabian M. Suchanek. 2015. Fast rule mining in ontological knowledge bases with AMIE+. *VLDB J.* 24, 6 (2015), 707–730.

[17] Luis Antonio Galárraga, Christina Teflioudi, Katja Hose, and Fabian Suchanek. 2013. AMIE: Association Rule Mining Under Incomplete Evidence in Ontological Knowledge Bases. In *Proceedings of WWW*. 413–422.

[18] Daniel Gerber, Diego Esteves, Jens Lehmann, Lorenz Bühmann, Ricardo Usbeck, Axel-Cyrille Ngonga Ngomo, and René Speck. 2015. DeFacto-Temporal and Multilingual Deep Fact Validation. *Web Semant.* 35, P2 (Dec. 2015), 85–101.

[19] Clinton Gormley and Zachary Tong. 2015. *Elasticsearch: The Definitive Guide* (1st ed.). O'Reilly Media, Inc.

[20] Joseph L. Fleiss. 1971. Measuring Nominal Scale Agreement Among Many Raters. 76 (11 1971), 378–.

[21] Julien Leblay. 2017. A Declarative Approach to Data-Driven Fact Checking. In *AAAI*. AAAI Press, 147–153.

[22] Qi Li, Yaliang Li, Jing Gao, Bo Zhao, Wei Fan, and Jiawei Han. 2014. Resolving Conflicts in Heterogeneous Data by Truth Discovery and Source Reliability Estimation. In *Proceedings of SIGMOD*. 1187–1198.

[23] Xian Li, Weiyi Meng, and Clement Yu. 2011. T-verifier: Verifying Truthfulness of Fact Statements. In *Proceedings of ICDE*. IEEE, 63–74.

[24] Yaliang Li, Jing Gao, Chuishi Meng, Qi Li, Lu Su, Bo Zhao, Wei Fan, and Jiawei Han. 2015. A Survey on Truth Discovery. *SIGKDD Explorations* 17, 2 (2015), 1–16.

[25] Subhabrata Mukherjee, Gerhard Weikum, and Cristian Danescu-Niculescu-Mizil. 2014. People on drugs: credibility of user statements in health communities. In *Proceedings of KDD*. 65–74.

[26] Ndapandula Nakashole and Tom M. Mitchell. 2014. Language-Aware Truth Assessment of Fact Candidates. In *Proceedings of ACL*. 1009–1019.

[27] Ndapandula Nakashole, Gerhard Weikum, and Fabian Suchanek. 2012. PATTY: A Taxonomy of Relational Patterns with Semantic Types. In *Proceedings EMNLP*. 1135–1145.

[28] Roberto Navigli and Simone Paolo Ponzetto. 2012. BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence* 193 (2012), 217–250.

[29] Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. 2016. A Review of Relational Machine Learning for Knowledge Graphs. *Proc. IEEE* 104, 1 (2016), 11–33.

[30] Jeff Pasternack and Dan Roth. 2013. Latent credibility analysis. In *Proceedings of WWW*. 1009–1020.

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

[32] Kashyap Popat, Subhabrata Mukherjee, Jannik Strötgen, and Gerhard Weikum. 2017. Where the Truth Lies: Explaining the Credibility of Emerging Claims on the Web and Social Media. In *Proceedings of WWW*. 1003–1012.

[33] Baoxu Shi and Tim Weninger. 2016. Discriminative predicate path mining for fact checking in knowledge graphs. *Knowl.-Based Syst.* 104 (2016), 123–133.

[34] Hui Shi, Kurt Maly, and Steven Zeil. 2014. Optimized Backward Chaining Reasoning System for a Semantic Web. In *Proceedings of WIMS'14*. 34:1–34:6.

[35] Prashant Shiralkar, Alessandro Flammini, Filippo Menczer, and Giovanni Luca Ciampaglia. 2017. Finding Streams in Knowledge Graphs to Support Fact Checking. In *Proceedings of ICDM 2017*. 859–864.

[36] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: A Core of Semantic Knowledge. In *Proceedings of WWW*. 697–706.

[37] Hai Dang Tran, Daria Stepanova, Mohamed Gad-elrab, Francesca A Lisi, and Gerhard Weikum. 2016. Towards Nonmonotonic Relational Learning from Knowledge Graphs. *ILP* (2016).

[38] Denny Vrandecic and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Communications of ACM* 57, 10 (2014), 78–85.

[39] Zhichun Wang and Juan-Zi Li. 2015. RDF2Rules: Learning Rules from RDF Knowledge Bases by Mining Frequent Predicate Cycles. *CoRR* abs/1512.07734 (2015).

[40] Zhuoyu Wei, Jun Zhao, Kang Liu, Zhenyu Qi, Zhengya Sun, and Guanhua Tian. 2015. Large-scale Knowledge Base Completion: Inferring via Grounding Network Sampling over Selected Instances. In *Proceedings of CIKM '15*. 1331–1340.

[41] X. Yin, J. Han, and P. S. Yu. 2008. Truth Discovery with Multiple Conflicting Information Providers on the Web. *IEEE Transactions on Knowledge and Data Engineering* 20, 6 (2008), 796–808.