Tracy: Tracing 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
Bosch Center for Artificial Intelligence
Renningen, Germany
daria.stepanova@de.bosch.com

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

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

ABSTRACT

In order to accurately populate and curate Knowledge Graphs (KGs), it is important to distinguish $\langle s\ p\ o\rangle$ facts that can be traced back to sources from facts that cannot be verified. Manually validating each fact is time-consuming. Prior work on automating this task relied on numerical confidence scores which might not be easily interpreted. To overcome this limitation, we present Tracy, a novel tool that generates human-comprehensible explanations for candidate facts. Our tool relies on background knowledge in the form of rules to rewrite the fact in question into other easier-to-spot facts. These rewritings are then used to reason over the candidate fact creating semantic traces that can aid KG curators. The goal of our demonstration is to illustrate the main features of our system and to show how the semantic traces can be computed over both text and knowledge graphs with a simple and intuitive user interface.

KEYWORDS

Knowledge Graph; Fact-checking; Explainable Evidence; Reasoning

ACM Reference Format:
Mohamed H. Gad-Elrab, Daria Stepanova, Jacopo Urbani, and Gerhard Weikum. 2019. Tracy: Tracing Facts over Knowledge Graphs and Text. In *Proceedings of the 2019 World Wide Web Conference (WWW '19), May 13–17, 2019, San Francisco, CA, USA.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3308558.3314126

1 INTRODUCTION

Motivation and Problem. Knowledge Graphs (KGs) are repositories of factual knowledge in the form of $\langle s\ p\ o\rangle$ facts where s, o are entities and p specifies a semantic relation between them, e.g., $\langle$*London capitalOf UK*$\rangle$. Currently, a large number of KGs are publicly available on the Web (e.g., Wikidata [16], YAGO [15]) but some of them might contain doubtful if not incorrect facts as they are partly built by automatic information extraction, crowd-sourcing, or other noisy methods [12, 18]. Due to the increasing usage of KGs for tasks like query-answering, dialog systems, etc., it is important to validate each candidate fact to ensure the correctness of the KG.

WWW '19, May 13–17, 2019, San Francisco, CA, USA

ACM ISBN 978-1-4503-6674-8/19/05.
https://doi.org/10.1145/3308558.3314126

This task, which is often referred to as *fact-checking* or *truth discovery* [8], can be performed manually by KG curators but this is time-consuming. Therefore, the automation of this process is gaining more attention. Existing methods for automatic fact-checking (e.g., [5, 8, 9, 11, 13]) usually proceed in two steps as follows: First, they search for explicit mentions of the fact in the Web sources like news or other textual corpora. For example, for a fact such as $\langle$*Sadiq_Khan citizenOf UK*$\rangle$, they search for *"khan is a citizen of UK"* or *"Khan's nationality is British"*. Then, the extracted evidence is used to infer whether the candidate fact can indeed be verified.

Existing approaches have two main limitations: First, they quantify the confidence using numerical scores, which is not adequate in case the final decision is made by KG curators. Indeed, such scores are hard to understand or justify without explanations. Only few approaches (*e.g.*, [1, 5]) attempt to explain the results. For example, Defacto [5] shows the sources used in computing the scores as an explanation and [1] reports a comparison between computed scores to explain its final decision for humans. Second, searching for explicit mentions is 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. Moreover, some predicates (*e.g.*, *influencedBy*) are ambiguous and their interpretation is domain-specific.

Proposed Approach. We introduce Tracy, a tool designed to support KG curators in deciding the correctness of the candidate facts. The main novelty of our tool consists of finding *semantically related evidence* in textual sources and the underlying KG, and providing *human-comprehensible explanations* for the facts. In Tracy, the semantically related evidence is extracted according to intentional background knowledge given as rules of the form $H \leftarrow B_1, B_2, \ldots, B_n$ which can be either specified by humans or automatically extracted using rule mining methods [14] By utilizing rules, Tracy enables users to combine clues from different resources (structured and unstructured) thus overcoming the problem that arises when the fact is never explicitly mentioned. For example, consider the rule

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

which intuitively states that mayors of cities are normally citizens of countries where these cities are located. This rule can be used by our tool to verify the fact $\langle$*Sadiq_Khan citizenOf UK*$\rangle$ by searching in news articles whether he is a major of some city and then looking up the city's corresponding country in the KG. The combination

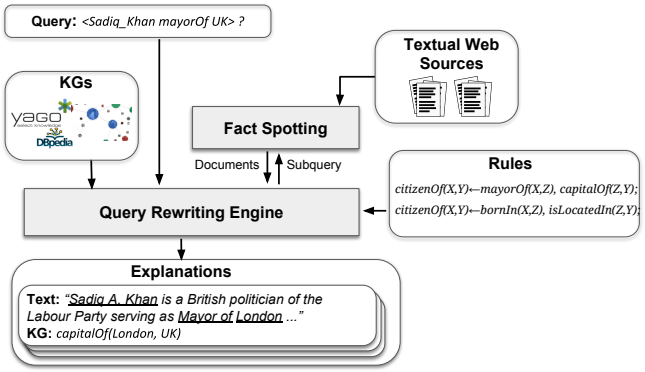

Figure 1: Tracy system overview.

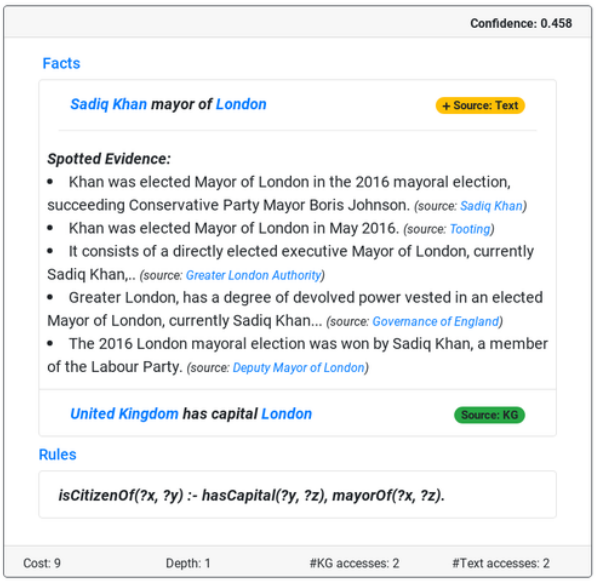

Figure 2: Example explanation returned by Tracy.

of these two semantically-related facts allows us to construct a human-readable explanation for the correctness of the fact.

As illustrated by the above example, Tracy uses the rules to decompose the spotting of an input fact into more frequently stated and thus easier-to-spot related facts. The search for evidence for the rule's body might trigger the execution of other rules, and this effectively creates semantic traces that *explain*, in a human-readable format, why a fact is likely to be true. These traces are presented to a KG curator in a comprehensible format in order to make a final decision about the truth value of the fact in question.

Outline. In the remaining, we briefly describe the framework underlying Tracy and the realization of the main components. Afterwards, we illustrate the features offered by the demonstrated graphical user interface, allowing users to experience different usage scenarios. Our demo is made available under the following link https://www.mpi-inf.mpg.de/impact/exfakt#Tracy.

2 SYSTEM OVERVIEW

Tracy follows ExFaKT framework introduced in our earlier work [3]. Tracy implements two main components: (i) *Query Rewriting Engine (QRE)*, which is responsible for rewriting facts and generating explanations, and (ii) *Fact Spotting (FS)*, which is the component that finds evidences in Web sources. Fig. 1 shows a graphical representation of this architecture.

Our framework receives as input a candidate fact (*i.e.*, the query), a set of rules, and two knowledge sources: (i) a KG, which we view as a *reliable* source; and (ii) an unstructured collection of text corpora, which we consider as an *unreliable* source, since its extractions by *FS* might be noisy.

Rules are Horn clauses of the form of $H \leftarrow B_1, \ldots, B_n$ where H is the rule's head and $B_1, \ldots, B_n$ is the body of the rule. In general, rules can be automatically extracted from KGs using systems such as [2, 4, 6, 17]. However, extracted rules are restricted to the predicates in the KG. Thus, Tracy also accepts manually specified rules with user-defined predicates, which later can be spotted in the text.

The workflow of our system is as follows: First, a user submits an input fact Q (*i.e.*, the query) to *QRE*. Then, this component possibly rewrites it into easier facts according to the logic specified by the rules contacting *FS* every time it cannot find evidences in the KG. If a sufficient number of evidences is found, then QRE uses them to return one or more explanations for Q.

2.1 Query Rewriting Engine

The process starts by considering rules, whose head predicates are the same as in the query. Then, it visits the rules' bodies and effectively rewrites the query into additional subqueries. The subqueries might trigger new rules and this results in a recursive process until all subqueries are answered. Such rule-based evaluation strategy is well-known in logic programming and commonly used for query answering. Indeed, our system implements an adaptation of a set-based version of standard SLD resolution [7].

Our adaption consists of three modifications which are crucial for our scenario of interest: First, we impose a maximum depth onto the recursive process to ensure termination. Second, the system attempts to answer the query consulting *FS* whenever it cannot be answered relying on the KG. Third, we care about the quality of the sources; and hence, we still rewrite the fact if the evidence source is not trustful seeking more reliable evidences. These two last modifications are particularly important: In fact, it is well-known that KGs are highly incomplete and a conventional query answering engine would simply fail when evidence is not found. In contrast, the ability of our system to also include extractions from additional sources overcomes this problem and enables the construction of explanations even when the content of the KG is not sufficient.

Once this process is finished, the answers (either from the KG or Web sources) are used to compute the explanations. This is done by revisiting the rules bottom-up. Fig. 2 shows an extract of an example explanation provided by Tracy. In this case, the explanation illustrates that the fact that *Sadiq Khan is a citizen of UK* is likely to be true according to the rule

$$isCitizenOf(X, Y) \leftarrow mayorOf(X, Z),\ hasCapital(Y, Z)$$

and additional evidence found both in the KG and textual corpora.

Note that a given query might have multiple complex explanations or a single *trivial* one, i.e., the fact itself. Moreover, some explanations may be subsumed by others. Ideally, we aim at computing non-trivial explanations that are (i) **concise** with a small number of atoms; (ii) **close to the query**, i.e., obtained by using few rules; (iii) **reliable**, i.e., contain as many facts from the KG as possible, since KGs are usually more reliable than text.

In order to recognize these explanations, we compute a confidence score for each explanation E, defined as:

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

where a is the atom of the explanation E. This confidence score is directly correlated with the quality of the sources containing a ($trust(sources[a])$) and inversely correlated with the depth of the rewriting performed to reach the atom a (*i.e.*, the number of used rewriting rules) and the number of distinct atoms in the explanation.

2.2 Fact Spotting

Tracy follows a modular design allowing integrating any fact-spotting method. As default, we implemented a dictionary based fact-spotting procedure similar to [9, 13]. It is a simple and highly scalable method that does not require training. The main idea consists of first converting the SPO query into a textual representation (*i.e.*, verbalization) using a paraphrasing dictionary. Then, queries with the paraphrases are issued to a spotting engine to retrieve documents that mention them.

We used two types of dictionaries, depending on whether the used KG is YAGO or Wikidata (these are the only two which are currently supported). With YAGO, we use the paraphrases learned by PATTY [10] after some manual filtering. With Wikidata, we exploit the name aliases included in the KG.

We also implemented two types of spotting engines, one that uses a *local* text corpus and another one that consults a *remote* one:

- **Local corpus:** We used Elasticsearch [1] to index all Wikipedia articles. As a prepossessing step, we filtered the textual parts in the articles by removing semi-structured parts such as tables and info-boxes. Then, we index the article sentences separately along with the title. For spotting an SPO query, we issue a boolean query with the paraphrases of each part of the SPO separately. Then, we collect the *top-5* matching sentences as evidence for the query.
- **Remote corpus:** We also provide the option for searching the Web using Bing Web search API [2]. To query the API, we compose a string query containing all possible paraphrases of SPO and use the *top-5* search results as evidence.

2.3 Experimental Results

To illustrate the potential of our system, we report the results of an experiment using the YAGO KG. We simulated the case where we need to verify 300 unseen candidate facts, uniformly distributed over six different relations *influences*, *isPoliticianOf*, *wroteMusicFor*, *mayorOf*, *actedWith*, and *countryWonPrize*. A rule set was compiled for each predicate by selecting the top-ranked rules mined by AMIE [4] and adding other manually created rules involving new predicates that do not exist in YAGO.

[1] https://www.elastic.co/

[2] http://azure.microsoft.com/en-us/services/cognitive-services/bing-web-search-api

Table 1: Direct fact-spotting vs Tracy explanations

	Textual Source	Recall	Prec@5	F1@5
Direct-spotting	Wikipedia	0.25	0.85	0.40
Tracy		0.50	0.87	0.64
Direct-spotting	Web Search	0.41	0.85	0.55
Tracy		0.90	0.97	0.93

Then, we used our system to extract explanations for the candidates over YAGO using either the Wikipedia articles (local fact-spotting) or Web search (remote fact-spotting). Later, we asked MTurk workers to judge the correctness of each candidate fact based on the provided explanations. Based on these annotations, we computed *recall*, *precision@5 (Prec@5)*, and F_1*@5* score.

Table 1 shows a comparison between the results of using Tracy against the baseline method which consists of using only *direct fact-spotting* (thus without any rewriting). We can observe that Tracy doubles the recall while slightly increasing the precision@5. This leads to significant enhancement in the F_1 score. These results are encouraging, as they show that our system is 1) indeed capable of finding additional evidence that might not be directly mentioned and that 2) the explanations are actually valuable for human annotators for judging the correctness of candidate facts. More detailed experimental results can be found in [3].

3 DEMONSTRATION

We built a Web interface to allow an easy interaction with the system. Fig. 3 shows a screenshot illustrating its main components.

System options. The left-hand side of Fig. 3 shows the *system options menu*, which is the part where the user can select from a predefined list of KGs and textual sources to work with. Moreover, the interface allows the user to tune the QRE procedure by specifying a *trust* value for each textual source and the limit on the number of used rules and output explanations.

Input. The top part in Fig. 3 contains the *input form* with three fields for the SPO *query* and a text area for the *rules*. In the *query* fields, the user can select the subject and the object of the fact in question from a set of KG entities and specify the predicate, which might be out of KG. In the *rules* text area, the user is expected to input the rules using the standard logic programming syntax. For example, a rule expressing that "*a person is a citizen of a country if he was born in one of its cities*" is written as:

$$isCitizenOf(?x, ?y) \text{ :- } bornIn(?x, ?z),\ isA(?z, \text{`}City\text{'}),\ in(?z, ?y)$$

where $?x, ?y, ?z$ are variables, '$City$' is a constant, *isCitizenOf* is a head predicate and the rest are body predicates. These rules can also involve out-of-KG predicates, which Tracy spots in the text.

Output. Below the rules text area, Tracy returns the list of explanations for the input fact. Each *explanation* card contains:

- A set of facts that support the correctness of the query;

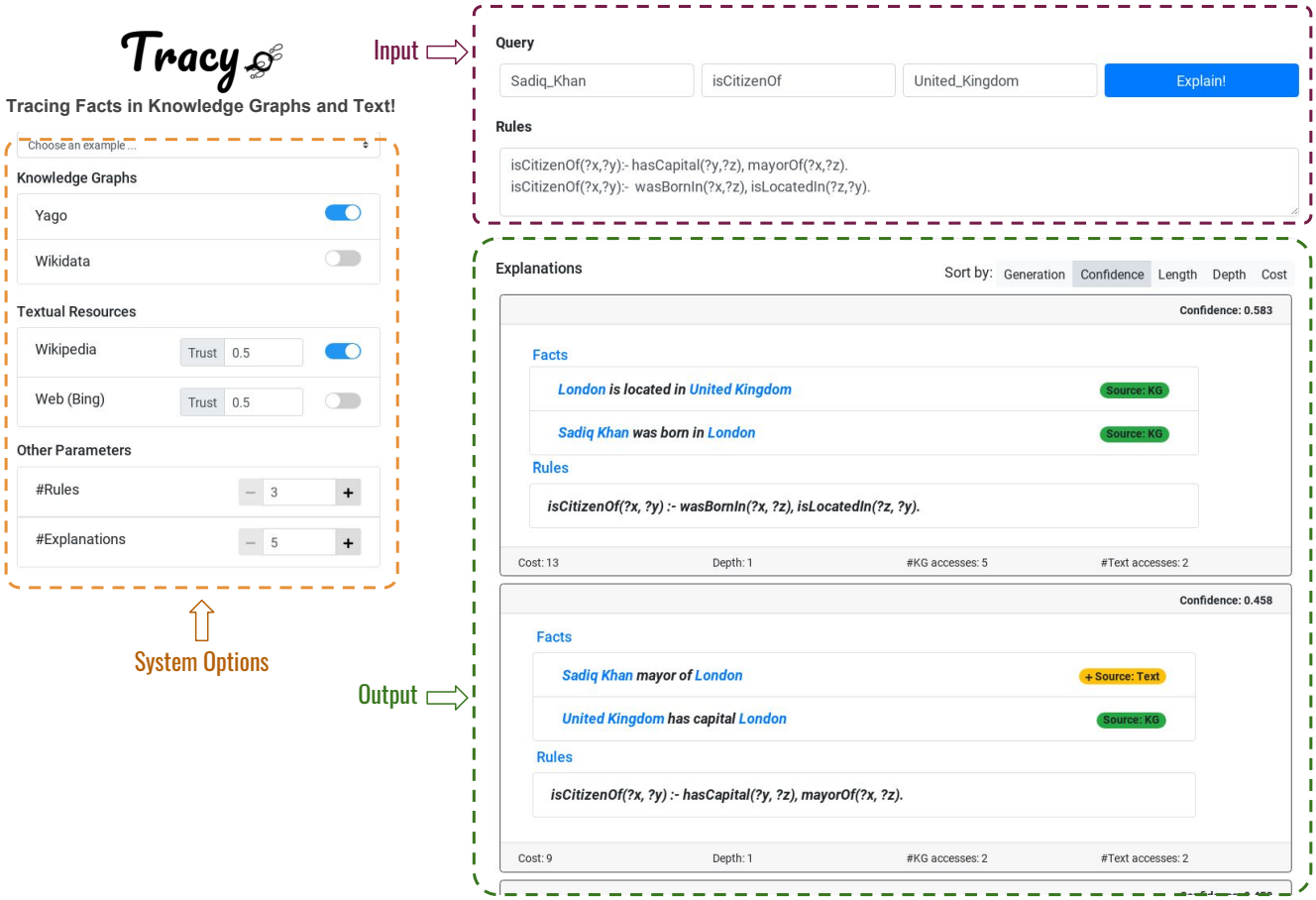

Figure 3: Tracy Web interface.

- In case the explanation includes facts with textual evidence, the user can view top-5 matching sentences with the links to the Web-pages where they have been found;
- The set of rules used by Tracy to find the evidence;
- Explanation confidence score computed as described in Section 2.1 and some execution insights (*e.g.*, resources accesses).

Explanations can be sorted based on their generation order, quality, length, rewriting depth, or cost.

During the demonstration session, We will showcase how Tracy can be used to extract non-trivial explanations for several hard to spot queries in different domains (*e.g.*, art, sport, and politicians). Users will have the chance to observe the results of changing the ruleset and other parameters to experience the performance of Tracy in different scenarios.

Users will also have the possibility to test Tracy with some arbitrary queries and/or other rule sets. We will encourage them to first try to collect related evidence manually before using Tracy with the same query. This illustrates how Tracy facilitates the discovery of related evidence required for judging the correctness of the query.

4 CONCLUSION

Our tool represents a first step towards producing more human understandable evidence for unverified facts. Tracy can help curators judging the correctness of new facts faster and more accurately. As future work, it is interesting to enhance our tool to support more complex rules, *e.g.*, with negations. Such rules are useful to represent additional background knowledge. Moreover, developing a more robust spotting engine that is capable of recognizing negative mentions of facts would allow more complex reasoning. Finally, we plan to further enhance the interface to make our system easier to use for KG curators.

ACKNOWLEDGMENTS

This work was partially supported by the ERC Synergy Grant 610150 (imPACT).

REFERENCES

[1] 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.

[2] Mohamed H Gad-Elrab, Daria Stepanova, Jacopo Urbani, and Gerhard Weikum. 2016. Exception-enriched rule learning from knowledge graphs. In *Proceedings of ISWC*. 234–251.

[3] Mohamed H. Gad-Elrab, Daria Stepanova, Jacopo Urbani, and Gerhard Weikum. 2019. ExFaKT: A Framework for Explaining Facts over Knowledge Graphs and Text. In *Proceedings of WSDM*. 87–95.

[4] 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.

[5] 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.

[6] Vinh Thinh Ho, Daria Stepanova, Mohamed H. Gad-Elrab, Evgeny Kharlamov, and Gerhard Weikum. 2018. Rule Learning from Knowledge Graphs Guided by Embedding Models. In *The Semantic Web - ISWC 2018, Proceedings, Part I*. 72–90.

[7] Robert Kowalski and Donald Kuehner. 1971. Linear resolution with selection function. *Artificial Intelligence* 2, 3 (1971), 227 – 260.

[8] 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 (2015), 1–16.

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

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

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

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

[13] 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.

[14] Daria Stepanova, Mohamed H. Gad-Elrab, and Vinh Thinh Ho. 2018. Rule Induction and Reasoning over Knowledge Graphs. In *Reasoning Web* (2018) *(Lecture Notes in Computer Science)*, Vol. 11078. Springer, 142–172.

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

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

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

[18] 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.