

Finding Streams in Knowledge Graphs to Support Fact Checking

Prashant Shiralkar*, Alessandro Flammini*†, Filippo Menczer*†, Giovanni Luca Ciampaglia†

*Center for Complex Networks and Systems Research, School of Informatics and Computing

†Network Science Institute

Indiana University, Bloomington (USA)

Abstract—The volume and velocity of information that gets generated online limits current journalistic practices to fact-check claims at the same rate. Computational approaches for fact checking may be the key to help mitigate the risks of massive misinformation spread. Such approaches can be designed to not only be scalable and effective at assessing veracity of dubious claims, but also to boost a human fact checker’s productivity by surfacing relevant facts and patterns to aid their analysis. To this end, we present a novel, unsupervised network-flow based approach to determine the truthfulness of a statement of fact expressed in the form of a (subject, predicate, object) triple. We view a knowledge graph of background information about real-world entities as a flow network, and knowledge as a fluid, abstract commodity. We show that computational fact checking of such a triple then amounts to finding a “knowledge stream” that emanates from the subject node and flows toward the object node through paths connecting them. Evaluation on a range of real-world and hand-crafted datasets of facts related to entertainment, business, sports, geography and more reveals that this network-flow model can be very effective in discerning true statements from false ones, outperforming existing algorithms on many test cases. Moreover, the model is expressive in its ability to automatically discover several useful path patterns and surface relevant facts that may help a human fact checker corroborate or refute a claim.

Index Terms—Knowledge Stream, Fact Checking, Knowledge Graph Completion, Unsupervised Learning, Relational Inference, Network Flow, Minimum Cost Maximum Flow, Successive Shortest Path

I. INTRODUCTION

Misinformation, unverified rumors, hoaxes, and lies have become rampant on the Internet nowadays, primarily due to the ability to quickly disseminate information at a large scale through the Web and social media. This phenomenon has led to many ill effects and, according to experts, poses a severe threat to society at large [1]. To address these problems, numerous approaches have been designed to study and mitigate the effects of misinformation spread (see Zubiaga *et al.* [2]). Most strategies rely on contextual indicators of rumors (e.g., number of inquiring tweets, reporting dynamics during breaking news, temporal patterns, or source credibility) for their detection and veracity assessment. To go beyond contextual approaches one would need to assess the truthfulness of claims by reasoning about their content and related facts. Moreover, a fact-checking system would ideally need to operate in near real time, to match the rate at which false or misleading claims are made.

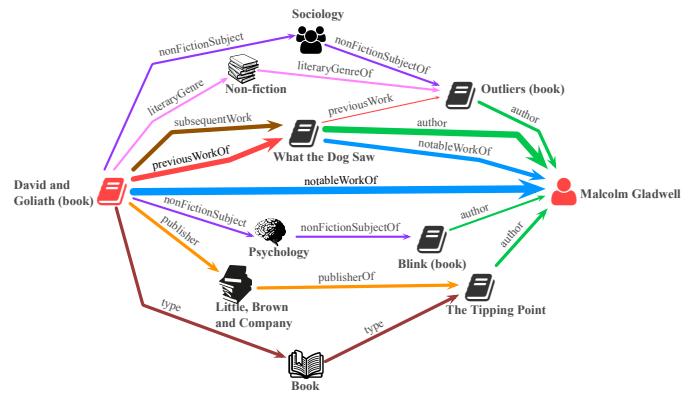


Fig. 1. The best paths identified by Knowledge Stream for the triple (*David and Goliath (book)*, *author*, *Malcolm Gladwell*). The width of an edge is roughly proportional to the flow of knowledge through it.

With advances in information extraction and in the adoption of semantic web standards, large quantities of structured knowledge have recently become available in the form of knowledge graphs (KGs). Nodes in a KG represent entities, and edges correspond to facts about them, as specified by semantic predicates, or relations. A wide class of empirical facts can be thus represented by a triple (s, p, o) , where the subject entity s is related to the object entity o by the predicate relation p . For example, (*Joe*, *spouse*, *Jane*) indicates that Jane is the spouse of Joe. DBpedia [3], YAGO2 [4] and Wikidata [5] are examples of publicly available KGs. These KGs contain vast amounts of high-quality knowledge about real-world entities, events, and their relations, and thus could be at least in principle harnessed by fact-checking agents.

Insofar as claims as simple as a triple are of concern, how can we automatically assess their truthfulness, given a large amount of prior knowledge structured as a KG? A few recent attempts have shown that this is possible via traversal of the graph. Traversal can take many forms, for example random walks (PRA [6]), path enumeration (PredPath [7]), or shortest paths (Knowledge Linker [8]). Other approaches have been proposed too, such as those designed for learning from multi-relational data (e.g., RESCAL [9], TransE [10] and their extensions), or those performing link prediction in social and collaboration networks [11].

However, as we discuss in Section IV, none of these approaches offers at once all the qualities that a desirable fact-checking system ought to have — namely accuracy, interpretability, simplicity, scalability, and the ability to take the greater context of a claim into account while evaluating it.

In this paper we propose Knowledge Stream (KS), an unsupervised approach for fact-checking triples based on the idea of treating a KG as a flow network. There are three motivations for this idea: (1) multiple paths may provide greater semantic context than a single path; (2) because the paths are in general non-disjoint, the method reuses edges participating in diverse overlapping chains of relationships by sending additional flow; and (3) the limited capacities of edges limit the number of paths in which they can participate, constraining the path search space.

Our approach not only delivers performance comparable to state-of-the-art methods like PredPath; it also produces more meaningful explanations of its predictions. It does so by automatically discovering in the KG useful patterns and contextual facts in the form of paths. The model we propose is conceptually simple, intuitive, and uses the broader structural and semantic context of the triple under evaluation.

As an example, Fig. 1 shows the paths computed for a true fact (*David and Goliath* (book), *author*, *Malcolm Gladwell*). We call this set of paths a “stream” of knowledge. A stream can thus be seen as the best form of evidence in support of the triple that the KG is able to offer. One can note from Fig. 1 that some paths give more evidence than others (wider edges in the figure). For example, the fact that Malcolm Gladwell is the author of the book *What the Dog Saw*, which followed *David and Goliath*, is a stronger form of evidence than the fact that another book authored by Gladwell, *The Tipping Point*, was published by the same company (Little, Brown and Company) as *David and Goliath*. Knowledge Stream correctly assigns a larger flow to the former path than the latter.

For a given triple (s, p, o) , we view knowledge as a certain amount of an abstract commodity that needs to be moved from the subject entity s to the object entity o across the network. Each edge of the network is associated with two quantities: a *capacity* to carry knowledge related to (s, p, o) across its two endpoints, and a *cost* of usage. We want to identify the set of paths responsible for the maximum flow of knowledge between s and o at the minimum cost. We give some definitions to make these statements more formal and explain the intuition behind our approach.

Each edge $e \in E$ of the KG has an intrinsic *capacity*, which depends on the triple under consideration. Recall that an edge is labeled with a predicate p' possibly different from the target predicate p . Intuitively, the more similar, or relevant, p' is to p , the higher the capacity of e ought to be. If we are to ascertain whether Jane is indeed the spouse of Joe, facts about the realm of, say, geology are in general less pertinent than facts about ancestry or family history. We use a data-driven approach, mining the structure of the KG itself, to define the similarity between predicates. To do so, we employ the graph-theoretic

concept of *line graph* of the KG. The full details are described in Section II-A.

The maximum knowledge flow carried by a path is the minimum capacity of its edges. The edge at which the minimum is found is known as the *bottleneck* of the path [12]. In our approach, the bottleneck corresponds to the least relevant triple along the path. In general, there are many paths connecting s to o , and the total knowledge that can flow through them is bounded by the sum of their bottlenecks.

To each edge $e \in E$ we also associate a *cost* for sending a unit of flow across its two endpoints. This ensures that the paths discovered by KS are short. Previous work has directly or indirectly confirmed the intuition that the structures (walks, paths, etc.) that best explain whether a triple is true are short [6]–[8].

Our definition of path length differs from the traditional number of hops: a short path involves not only few entities but also entities with few connections to other entities in the graph [8]. We say that such entities are “specific,” and the paths containing them are “specific paths.”

As mentioned earlier, one of the components of KS is the method to compute similarity between relations. This method can also be applied to shortest-path approaches, such as Knowledge Linker. The resulting algorithm, which we call “Relational Knowledge Linker” (KL-REL), assigns a truth score to (s, p, o) by biasing the search for the shortest path toward predicates related to p .

In summary, this paper makes the following contributions:

- We propose a novel method called Knowledge Stream to perform computational fact checking using large knowledge graphs such as DBpedia [3]. To our knowledge, this is the first instance of applying flow network to the problem of soft reasoning with knowledge graphs.
- We introduce a novel approach to gauge similarity between a pair of relations in the KG. This similarity can be used for many other tasks, e.g. analogical reasoning.
- We propose a fact-checking algorithm called Relational Knowledge Linker that verifies a claim based on the single shortest, semantically related path in the KG.
- We experimentally compare our approaches of Knowledge Stream and Relational Knowledge Linker to a number of existing algorithms designed for fact checking, knowledge graph completion, and link prediction. We show that both KS and KL-REL offer high interpretability and performance comparable to the state of the art.

II. METHODS

In this section we describe Knowledge Stream and Relational Knowledge Linker, the two methods we propose to perform fact checking using a KG. Formally, a KG is a directed graph $G = (V, E, \mathcal{R}, g)$, where V , E , and \mathcal{R} denote the node, edge, and relation sets, respectively, and $g : E \rightarrow \mathcal{R}$ is a function labeling each edge with a semantic relation or predicate. Even though G is a directed network, in practice most existing methods for fact checking, including ours, view

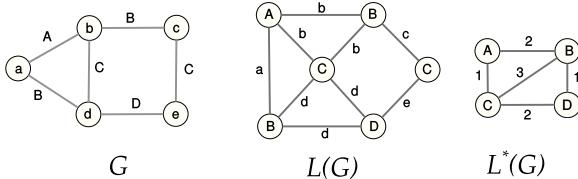


Fig. 2. Example of the line graph $L(G)$ and the contracted line graph $L^*(G)$ of a simple knowledge graph G with four relations (denoted by uppercase letters) and five nodes (lowercase letters). The edge weights in $L^*(G)$ represent how often each relation is co-incident to its neighbors in G .

it as an undirected one by discarding the directionality of edges.

Since both methods presented here rely on the ability to gauge the similarity, or relevance, of any pair of elements of \mathcal{R} , we start by explaining our data-driven approach to relational similarity.

A. Relational Similarity via the Line Graph of a KG

In graph theory, the *line graph* $L(G) = (V', E')$ of an undirected graph $G = (V, E)$ is the graph whose nodes set is $V' = E$ and in which two nodes are adjacent iff the corresponding edges of G are incident on the same node in G , that is, $E' = \{(e_1, e_2) : e_1, e_2 \in E \wedge e_1 \cap e_2 \neq \emptyset\}$. Line graphs are also sometimes known as *dual graphs*. The KG being an edge-labeled graph makes the $L(G)$ a node-labeled graph. However, even though $L(G)$ encodes information about the adjacency between relations of the KG, it is not suited to define a similarity metric on \mathcal{R} because it includes duplicate labels. We overcome this problem by contracting duplicate nodes until there is exactly one node for each element of \mathcal{R} . A graph can be *contracted* by replacing two nodes with a new node whose set of neighbors is the union of their neighbors. Rather than duplicating edges, the contracted graph is edge-weighted; the weight of a new edge reflects the number of old edges that are merged in the contraction. We thus start from G , then build $L(G)$ setting all edge weights to 1, and finally we iteratively contract pairs of nodes labeled with the same relation, until there are no duplicate labels. We call the resulting graph the *contracted line graph*, denoted by $L^*(G)$. See Fig. 2 for an example of a small KG with four relations and five nodes.

Let us denote with $C \in \mathbb{N}^{R \times R}$, where $R = |\mathcal{R}|$, the adjacency matrix of the contracted line graph. By construction, C is the co-occurrence matrix of \mathcal{R} . One could estimate the similarity between two relationships by computing the cosine between the row vectors corresponding to the relationships in C . However, the raw co-occurrence counts in C are dominated by the most common relationships. Therefore, as customary in information retrieval, we apply TF-IDF weighting to C :

$$\begin{aligned} \text{TF}(r_i, r_j) &= \log(1 + C_{ij}), \\ \text{IDF}(r_j, \mathcal{R}) &= \log \frac{R}{|\{r_i | C_{ij} > 0\}|}, \\ C'(r_i, r_j, \mathcal{R}) &= \text{TF}(r_i, r_j) \cdot \text{IDF}(r_j, \mathcal{R}) \end{aligned} \quad (1)$$



Fig. 3. Top 20 most similar relations for a few predicates in DBpedia. The font size is proportional to the relational similarity.

where C_{ij} is the co-occurrence count between $r_i \in \mathcal{R}$ and $r_j \in \mathcal{R}$. We define the *relational similarity* $u(r_i, r_j)$ as the cosine similarity of the i -th and j -th rows of C' . We found that this approach yields meaningful results; a few examples are shown in Fig. 3.

B. Fact checking as a Minimum Cost Maximum Flow Problem

As stated in the introduction, with Knowledge Stream we view fact checking as the problem of finding an optimal way to transfer, across the KG, knowledge from the source entity to the target entity under a set of constraints. These constraints depend both on the KG itself and on the given relation that we are trying to check.

The first set of constraints on the edges dictate that the amount of flow that can be pushed across an edge is bounded. In Knowledge Stream, we take the lower bound on this flow to be zero, and we define the upper bound or *capacity* of $e = (v_i, v_j) \in E$ with respect the triple to be fact-checked, (s, p, o) , as

$$U_{s,p,o}(e) = \frac{u(g(e), p)}{1 + \log k(v_j)}, \quad (2)$$

which is the product of the similarity u between the edge label $g(e)$ and the predicate p of the target triple (see Section II-A), and a quantity that represents the *specificity* of the node to which e is incident. The specificity is based on the assumption that the larger the degree k of a node — the more facts in the KG about it — the more *general* the concept is. Our use of the logarithm of the degree is based on information-theoretic arguments [8]. Alternative choices could of course be explored.

The second set of constraints relate to conservation of flow across nodes: except for the nodes corresponding to the subject

s and object o , the amount of flow entering a node must be equal to that leaving the node. We associate with s (resp. o) a fixed *supply* (*demand*) of knowledge, γ , which is the maximum feasible flow through the network.

In network flow problems, costs map to quantities to be minimized, like the distance of a road or amount of gas spent carrying goods over it. In the KG context, we employ again the idea that the degree of a node is a measure of generality, to be minimized. We therefore set the cost of edge $e = (v_i, v_j) \in E$ to $c_e = \log k(v_j)$. Note that although the KG is undirected, for each edge along a path, the capacity and cost functions consider the degree of the incident node v_j in the direction from s to o .

Having defined the main constraints, we solve a *minimum cost maximum flow* problem [12, Ch. 1, 9, 10]. The flow assignment to the edges of the KG is a non-negative real-valued mapping $f : E \rightarrow \mathbb{R}^+$, that maximizes the total flow γ pushed from s to o while minimizing the total cost $\sum_{e \in E} c_e f(e)$ subject to the edge capacity constraints:

$$0 \leq f(e) \leq \mathcal{U}_{s,p,o}(e)$$

and to the node conservation constraints:

$$b(v) = \begin{cases} \gamma & v = s \\ -\gamma & v = o \\ 0 & \text{otherwise} \end{cases}$$

where $b(v_i) = \sum_{v_j \in V} f(v_i, v_j)$ is the net flow outgoing from node v_i .

We are interested in finding the set of paths along which the maximum flow γ is pushed from s to o . In practice we solve the minimum cost maximum flow problem using an algorithm that computes such a set of paths. We denote this set of paths the *stream of knowledge* $\mathcal{P}_{s,p,o}$. Each path in the stream carries knowledge at its full capacity. The maximum knowledge a path $P_{s,p,o}$ can carry is the minimum of the capacities of its edges, also called its *bottleneck* $\beta(P_{s,p,o})$. It can be shown that the maximum flow is the sum of the bottlenecks of the paths that are part of the stream:

$$\gamma = \sum_{P_{s,p,o} \in \mathcal{P}_{s,p,o}} \beta(P_{s,p,o}). \quad (3)$$

Having determined the maximum flow and knowing the exact contribution of each individual path in a stream, we need to specify how to use the stream for fact checking. The flow through a path captures the relational similarity and specificity of its bottleneck, as well as the specificity of the intermediate nodes. Nevertheless, long chains of specific relationships could lead us astray. Therefore Knowledge Stream should favor *specific paths* involving few specific entities. We define the specificity $\mathcal{S}(P_{s,p,o})$ of a path $P_{s,p,o}$ with n nodes as inversely proportional to the sum of logarithms of the degrees of its intermediate nodes:

$$\mathcal{S}(P_{s,p,o}) = \frac{1}{1 + \sum_{i=2}^{n-1} \log k(v_i)}. \quad (4)$$

We say that the net flow $\mathcal{W}(P_{s,p,o})$ in a path $P_{s,p,o}$ is the product of its bottleneck $\beta(P_{s,p,o})$ and specificity $\mathcal{S}(P_{s,p,o})$:

$$\mathcal{W}(P_{s,p,o}) = \beta(P_{s,p,o}) \cdot \mathcal{S}(P_{s,p,o}). \quad (5)$$

Fact checking a triple (s, p, o) then reduces to computing a *truth score* $\tau^{\text{KS}}(s, p, o)$ as the sum of the net flow across all paths in the stream:

$$\begin{aligned} \tau^{\text{KS}}(s, p, o) &= \sum_{P_{s,p,o} \in \mathcal{P}_{s,p,o}} \mathcal{W}(P_{s,p,o}) \\ &= \sum_{P_{s,p,o} \in \mathcal{P}_{s,p,o}} \beta(P_{s,p,o}) \cdot \mathcal{S}(P_{s,p,o}). \end{aligned} \quad (6)$$

C. Computing the Knowledge Stream

Let us now discuss how to solve our optimization problem and compute the truth score of a triple in practice. A well-known algorithm called Successive Shortest Path (SSP) provides a solution to the optimization problem and also returns the sequence of paths. The idea is to push the maximum flow γ from s to o by iteratively finding a shortest path in a residual network, along which we can push some flow. The *residual network* $G(f)$ of G w.r.t flow f has the same set of nodes V as G , but has two kinds of edges: (1) *forward edges* with some “leftover capacity” over which one can push additional flow, and (2) *backward edges* that represents edges already allocated, over which one can push reverse flow in order to undo flow in forward edges. At each step in the iteration we compute the bottleneck of the shortest path, given by

$$\beta(P_{s,p,o}) = \min \{x_e | e \in P_{s,p,o}\}, \quad (7)$$

where $x_e \leq \mathcal{U}_e$ represents the residual capacity of edge e in the residual network. Our extended version of SSP to compute the stream of knowledge and the truth score $\tau^{\text{KS}}(s, p, o)$ for a given triple is shown in Algorithm 1.

We associate a real number $\pi(v_i)$ (Line 3) with each node $v_i \in V$, called its *node potential*. A vector of such node potentials π serves two important purposes: (1) it allows us to keep track of the *reduced cost* c^π (Line 5) of an edge at each step of the algorithm, which makes successive path-finding efficient; and (2) it serves as an ingredient of the *reduced cost optimality conditions* that ensure the achievement of maximum flow upon termination [12, Ch. 9].

The complexity bounds for the SSP algorithm assume that all edge weights are integral, which does not hold for our capacities ($\mathcal{U}_{s,p,o} \in [0, 1]$). This is not a problem however, since capacities are rational numbers and can therefore be easily converted to integers.

If the maximum flow γ is an integer, the Knowledge Stream algorithm takes at most γ iterations. Since each shortest path computation can be performed in $O(|E| \log |V|)$ time using Dijkstra’s algorithm [13] with a binary heap implementation, the overall complexity of the algorithm is $O(\gamma |E| \log |V|)$. In practice, γ is not an integer, and is computed by the algorithm; this makes Knowledge Stream a pseudo-polynomial time algorithm. In practice we find acceptable performance: for large-scale KGs such as DBpedia, our implementation takes an average of 356 seconds per triple on a laptop.

Algorithm 1 Knowledge Stream Algorithm

```

1: procedure KNOWLEDGESTREAM( $G, s, p, o$ )
2:    $\tau \leftarrow 0, \mathcal{P} \leftarrow \emptyset, f \leftarrow 0$ 
3:    $\pi \leftarrow 0$ 
4:    $c_{v_i, r_m, v_j} = \log(v_j), \forall (v_i, r_m, v_j) \in E$ 
5:    $c_{v_i, r_m, v_j}^\pi = c_{v_i, r_m, v_j} - \pi(v_i) + \pi(v_j)$ 
6:    $d \leftarrow$  compute shortest path distances from  $s$  to all
      other nodes in  $G(f)$  w. r. t.  $c^\pi$ 
7:    $P \leftarrow$  a shortest path from  $s$  to  $o$  in  $G(f)$ 
8:   while  $P$  exists do
9:      $\mathcal{P} \leftarrow \mathcal{P} \cup \{P\}$ 
10:     $\pi \leftarrow \pi - d$ 
11:     $\beta(P) \leftarrow \min \{x_{v_i, r_m, v_j} | (v_i, r_m, v_j) \in P\}$ 
12:    Push  $\beta(P)$  units of flow along  $P$ 
13:     $\mathcal{S}(P) \leftarrow \frac{1}{1 + \sum_{i=2}^{n-1} \log k(v_i)}$  for  $v_i \in P$ 
14:     $\mathcal{W}(P) \leftarrow \beta(P) \cdot \mathcal{S}(P)$ 
15:     $\tau \leftarrow \tau + \mathcal{W}(P)$ 
16:    update  $f, G(f)$  and reduced edge lengths  $c^\pi$ 
17:     $d \leftarrow$  compute shortest path distances from  $s$  to
      all other nodes in  $G(f)$  w. r. t.  $c^\pi$ 
18:     $P \leftarrow$  a shortest path from  $s$  to  $o$  in  $G(f)$ 
19:   end while
20:   return  $\tau, \mathcal{P}$ 
21: end procedure

```

D. Relational Knowledge Linker

Our measure of relational similarity defined in Section II-A can also be used to extend existing KG-based fact-checking methods. One such method is Knowledge Linker (KL) [8]. The approach used by KL for fact checking a triple (s, p, o) is to find the path between entities s and o that maximizes specificity (Eq. (4)). This approach ignores the semantics of the target predicate p . We hypothesize that biasing the search for specific paths to favor edges that are semantically related to p should improve KL. We therefore replace the definition of path specificity in Eq. (4) by

$$\mathcal{S}'(P_{s,p,o}) = \left[\sum_{i=2}^{n-1} \frac{\log k(v_i)}{u(r_{i-1}, p)} + \frac{1}{u(r_{n-1}, p)} \right]^{-1}. \quad (8)$$

This formulation maximizes the relational similarity between each edge and the target predicate, in addition to the specificity of the intermediate nodes. The last term allows to consider the relation of the last edge without penalizing the generality of the object o . The truth score of triple (s, p, o) is just $\tau^{\text{KL-REL}}(s, p, o) = \max_{P_{s,p,o} \in \mathcal{P}_{s,p,o}} \mathcal{S}'(P_{s,p,o})$.

The truth score and the associated path can be computed efficiently using Dijkstra's algorithm [13]. We call this extended approach the Relational Knowledge Linker (KL-REL).

III. EVALUATION

In this section we present the results of an evaluation of our two methods, Knowledge Stream (KS) and Relational Knowledge Linker (KL-REL), on a range of datasets. To make the evaluation meaningful, we pit these algorithms against a

number of existing approaches from the literature on computational fact checking and related problems, namely automated knowledge base construction (KBC) and link prediction. We start by describing the experimental setup.

A. Setup

1) Knowledge Graph: We select DBpedia, a popular knowledge base derived from Wikipedia, as the KG for all evaluations. DBpedia is a large community effort with the goal of extracting structured data from the body and infobox of each Wikipedia article. It is freely available in a serialized form, split across a number of RDF data dumps. In particular, to build the KG we used in the evaluation, we downloaded and merged together the following dumps: ontology, instance-types, and mapping-based properties. We use the most recent distribution at the time of evaluation.¹ We apply the following filtering to the dumps: (1) from the instance-types dump we remove all subsumption triples (i.e., triples of the form $x \xrightarrow{\text{is a}} T$, where x is an entity and T is a class in the DBpedia ontology) that are the result of transitive closure, as they shortcut the ontological hierarchy in an undesired way; and (2) we discard all triples whose object is an RDF literal (e.g., dates, numerical values, text labels), as they do not correspond to any KG entity. The undirected graph we obtain as a result has the following characteristics: $|V| = 6M$ nodes, $|E| = 24M$ triples, and $|\mathcal{R}| = 663$ relations.

2) Labeled Datasets: We evaluate all methods on two classes of datasets. The first class includes *synthetic* corpora that have been created for evaluation purposes by our team and others. These datasets mix *a priori* known true and false facts and are drawn from the domains of entertainment, business, geography, literature, sports, etc. Additional datasets include triples extracted in the wild, whose ground truth covers the full spectrum of truth scores, ranging from completely true to completely false. Several *real-world* datasets in the second class are derived from the Google Relation Extraction Corpora (GREC),² which contain information about birth place, death place, alma mater, and educational degree of notable people. Two more datasets about professions and nationalities are derived from the corpus of the WSDM Cup 2017 Triple Scoring challenge.³ Table I summarizes all the datasets. Those marked with an asterisk were first used in prior work [7]. We report the average number of facts per subject in the last column.

The ground truth in both the GREC and the WSDM Cup corpora was obtained via crowdsourcing. In the GREC, each triple was evaluated by five human raters. We use only triples whose rating was unanimous, i.e., either all true or all false. In the WSDM Cup corpus each triple was scored by seven raters, but the corpus contains only true triples, by design. We consider only true triples with a unanimous score, and we

¹wiki.dbpedia.org/downloads-2016-04

²research.googleblog.com/2013/04/50000-lessons-on-how-to-read-relation.html

³www.wsdm-cup-2017.org/triple-scoring.html

generate false facts by randomly drawing from professions or nationalities that individuals are not known to hold. This approach amounts to making a *local closed-world assumption* (LCWA).

3) *Benchmark & Metric:* We compare our approaches to three existing algorithms designed for fact checking (Knowledge Linker [8], PredPath [7], and PRA [6]), one algorithm for knowledge graph completion (TransE [10]), and four link prediction algorithms (Katz [14], Adamic & Adar [15], Jaccard coefficient [11], and Degree Product [7]). We use the area under the Receiver Operating Characteristic curve (AUROC) as a metric to evaluate algorithms; it allows us to compare the accuracy across datasets with different ratios of true and false facts. Each method emits a list of probabilistic scores, one for each triple, and the AUROC expresses the probability that a true triple receives a higher score than a false one.

4) *Implementation & Configuration:* All algorithms have been implemented in Python 2.7, and we use Cython 0.22 to efficiently compute single-source shortest paths and distances as required for KS (Line 17 and 18 in Algorithm 1). The source code for our methods can be found at <https://github.com/shiralkarprashant/knowledgestream>. For Katz, PRA and PredPath, we use up to 200 paths for every value of path length $l = 1, 2, 3$. In the case of TransE, we create 100-dimensional embeddings using a margin of one and a learning rate of 0.01 for 1,000 epochs.

B. Results

1) *Fact checking:* Table II and Table III give a comparison of fact-checking performance between our approaches and other algorithms on several synthetic and real-world datasets. We report average performance and standard error across datasets for each method in the last column. Although statistical significance tests do not reveal a clear overall winner, we can make a few observations. KL-REL performs better than the original KL, TransE, and all link prediction algorithms. In fact, it outperforms all other algorithms on real-world datasets and has comparable performance to PredPath on synthetic data.

KS lags behind KL. A possible explanation could be that the extra signal provided by the additional paths found by KS may not always be beneficial. To shed more light into this issue, we analyzed the average performance as a function of the number of paths in the stream. Fig. 4 shows that the overall optimum is attained when exactly two paths are considered. On the one hand, this confirms the value of considering multiple paths. On the other hand, this suggests that too many paths hinder performance, and thus the number of paths should be tuned.

Based on this insight, we include in our evaluation two variants of Knowledge Stream. KS-AVG uses the number of paths (two) resulting in the best performance on average. KS-CV uses cross-validation to tune the optimal number of paths for each dataset; this makes KS-CV a supervised approach. As we see from both tables, KS-AVG and KS-CV have a better performance on average than KS, and even better than KL-REL on synthetic datasets. This confirms our intuition of focusing only on a few paths in a stream.

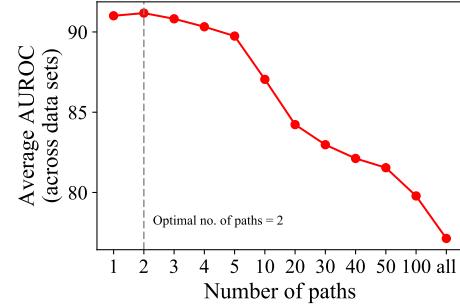


Fig. 4. Average performance of Knowledge Stream across datasets as a function of the number of paths used in the stream.

We observe that our algorithms (KL-REL, KS, KS-AVG and KS-CV) often outperform existing fact-checking methods (PredPath, KL and PRA). We emphasize that KL-REL and KS-AVG are purely unsupervised algorithms, whereas PredPath and PRA require supervision for both feature selection and model training.

Finally, link prediction algorithms (Adamic & Adar, Jaccard coefficient and Degree Product) tend to perform poorly. Katz is the exception in this category. On real-world datasets, its performance is comparable to that of KL-REL. However, both KS or KL-REL are computationally efficient compared to Katz. In the case of KL-REL this is because of its focus on a single path. As for KS, it uses multiple paths and penalizes longer paths just like Katz, but is more efficient thanks to the capacity constraints.

2) *Discovery of relational patterns:* For each triple, the paths discovered by algorithms like KS, KL-REL, PRA, and PredPath can be seen as the evidence used by the algorithm in deciding whether the fact is true. By pooling together evidence from many triples, we can discover data-driven patterns that define a relation, based on the prior knowledge in the KG. It is natural to ask whether the patterns discovered by our methods conform to common-sense understanding of these relations. To do so, we perform the following simple exercise. For each relation, we define the two sets A and B of all paths discovered from either true or false triples, respectively. We then rank the paths in decreasing order of their frequency of occurrence in the set difference $A - B$.

Table IV shows a few top patterns discovered by KS for a few relations. The patterns are highly relevant. We omit many other interesting examples due to space constraints. This characteristic of KS has a wider applicability — with only a few true and false examples, these patterns can be discovered in an unsupervised fashion, and used either as a seed set of rules in information extraction projects, or as features for learning other concepts.

3) *Surfacing facts relevant to a target claim:* The workflow of a human fact checker begins by gathering facts that are relevant to the claim being checked. Possible sources are background information, interview transcripts, etc. [16]. We find that KS can assist in this task by identifying the general context of a triple. As an illustration, Fig. 5 shows the set

TABLE I
SUMMARY OF DATASETS USED IN THE EVALUATION.

| | Dataset | Example Fact | True / Total | Facts / Subject |
|------------|------------------|---|---------------------|------------------------|
| Synthetic | NYT-Bestseller* | Flash Boys, author, M. Lewis | 93/558 | 7.5 |
| | NBA-Team | K. Bryant, team, LA Lakers | 41/164 | 9 |
| | Oscars | Gravity, director, A. Cuarón | 78/4680 | 13 |
| | CEO* | Best Buy, keyPerson, H. Joly | 201/1208 | 107 |
| | US War* | First Battle of Bull Run, battle, I. McDowell | 126/710 | 150 |
| | US-V. President* | B. Obama, vicePresident, J. Biden | 47/274 | 169 |
| | FLOTUS | B. Obama, spouse, M. Obama | 16/256 | 298 |
| | US-Capital #2* | Alabama, capital, Montgomery | 50/300 | 4214 |
| Real-World | GREC-Birthplace | D. Snow, birthPlace, Windermere, CA | 273/1092 | 8 |
| | GREC-Deathplace | N. Tate, deathPlace, Southwark | 126/504 | 8 |
| | GREC-Education | J. Warga, education, Bach. of Science | 466/1861 | 9 |
| | GREC-Institution | A. Mirsky, almaMater, Harvard College | 1546/6184 | 11 |
| | WSDM-Nationality | A. Einstein, nationality, Germany | 50/200 | 97 |
| | WSDM-Profession | A. Sandler, profession, Comedian | 110/440 | 220 |

TABLE II
FACT-CHECKING PERFORMANCE (AUROC) ON SYNTHETIC DATA. BEST SCORES FOR EACH DATASET ARE SHOWN IN BOLD.

| Method | NYT-Bestseller | NBA-Team | Oscars | CEO | US-War | US-V. President | FLOTUS | Capital #2 | Avg. (S.E.) |
|--------------------|-----------------------|-----------------|---------------|--------------|---------------|------------------------|---------------|-------------------|--------------------|
| KL-REL | 96.32 | 99.94 | 97.67 | 89.88 | 86.34 | 87.29 | 98.32 | 100.00 | 94.47 (2.0) |
| KS | 89.72 | 99.96 | 95.00 | 81.19 | 72.11 | 77.80 | 98.05 | 100.00 | 89.23 (3.9) |
| KS-AVG | 91.95 | 99.01 | 98.13 | 80.96 | 99.98 | 99.53 | 99.09 | 99.76 | 96.05 (2.3) |
| KS-CV | 93.63 | 99.29 | 97.72 | 80.52 | 99.98 | 99.47 | 99.27 | 99.28 | 96.14 (2.3) |
| PredPath [7] | 99.80 | 92.31 | 99.97 | 88.67 | 99.51 | 94.40 | 100.00 | 99.68 | 96.79 (1.6) |
| KL [8] | 94.99 | 99.94 | 97.56 | 89.77 | 63.55 | 74.62 | 98.59 | 99.42 | 89.80 (4.8) |
| PRA [6] | 96.24 | 91.26 | 99.54 | 87.73 | 99.96 | 50.00 | 60.48 | 98.88 | 85.51 (6.8) |
| TransE [10] | 80.99 | 56.71 | 82.66 | 82.68 | 53.22 | 72.50 | 84.82 | 85.31 | 74.86 (4.6) |
| Katz [14] | 96.52 | 98.50 | 98.98 | 87.53 | 57.80 | 72.92 | 97.42 | 99.97 | 88.70 (5.5) |
| Adamic & Adar [15] | 95.84 | 99.73 | 56.54 | 84.97 | 54.98 | 81.06 | 99.40 | 100.00 | 84.06 (6.7) |
| Jaccard [11] | 92.64 | 99.42 | 53.35 | 78.74 | 49.68 | 70.79 | 97.89 | 100.00 | 80.31 (7.3) |
| Degree Product [7] | 56.52 | 53.21 | 54.42 | 49.17 | 64.08 | 49.55 | 50.00 | 52.10 | 53.63 (1.7) |

TABLE III
FACT-CHECKING PERFORMANCE (AUROC) ON REAL TEST DATASETS. BEST SCORES FOR EACH DATASET ARE SHOWN IN BOLD.

| Method | GREC Birthplace | GREC Deathplace | GREC Education | GREC Institution | WSDM Nationality | WSDM Profession | Avg. (S. E.) |
|--------------------|------------------------|------------------------|-----------------------|-------------------------|-------------------------|------------------------|---------------------|
| KL-REL | 92.54 | 90.91 | 86.44 | 85.64 | 96.92 | 97.32 | 91.63 (2.0) |
| KS | 72.92 | 80.02 | 89.03 | 78.62 | 97.92 | 98.66 | 86.20 (4.4) |
| KS-AVG | 81.38 | 83.58 | 75.46 | 81.31 | 93.37 | 92.93 | 84.67 (2.9) |
| KS-CV | 82.28 | 82.57 | 75.23 | 81.33 | 94.20 | 95.84 | 85.24 (3.3) |
| PredPath [7] | 84.64 | 76.54 | 83.21 | 80.14 | 95.20 | 92.71 | 85.41 (2.9) |
| KL [8] | 92.10 | 90.49 | 62.32 | 87.61 | 96.05 | 91.36 | 86.65 (5.0) |
| PRA [6] | 74.34 | 75.58 | 70.51 | 63.95 | 83.87 | 50.00 | 69.71 (4.8) |
| TransE [10] | 54.88 | 56.47 | 66.32 | 44.99 | 77.09 | 82.91 | 63.78 (5.9) |
| Katz [14] | 88.46 | 84.07 | 89.55 | 82.99 | 99.23 | 98.84 | 90.52 (2.9) |
| Adamic & Adar [15] | 82.79 | 79.13 | 50.00 | 74.58 | 97.21 | 95.07 | 79.80 (7.0) |
| Jaccard [11] | 80.39 | 75.99 | 49.95 | 69.88 | 95.93 | 90.01 | 77.02 (6.6) |
| Degree Product [7] | 52.82 | 50.86 | 91.51 | 64.56 | 84.38 | 86.36 | 71.75 (7.3) |

TABLE IV
RELATIONAL PATTERNS DISCOVERED BY KNOWLEDGE STREAM.

| Relation | Pattern | Freq. | Example |
|------------|--------------------------------------|-------|---|
| Spouse | (child, childOf) | 34 | J. F. Kennedy $\xrightarrow{\text{child}}$ Patrick Kennedy $\xrightarrow{\text{childOf}}$ Jacqueline Kennedy Onassis |
| | (parentOf, parent) | 20 | J. F. Kennedy $\xrightarrow{\text{parentOf}}$ Patrick Kennedy $\xrightarrow{\text{parent}}$ Jacqueline Kennedy Onassis |
| | (child, parent) | 19 | J. F. Kennedy $\xrightarrow{\text{child}}$ Patrick Kennedy $\xrightarrow{\text{parent}}$ Jacqueline Kennedy Onassis |
| | (predecessor, spouse, predecessorOf) | 6 | R. Reagan $\xrightarrow{\text{predecessor}}$ P. Brown $\xrightarrow{\text{spouse}}$ B. Brown $\xrightarrow{\text{predecessorOf}}$ N. Reagan |
| CEO | (parentCompanyOf, keyPerson) | 32 | News Corporation $\xrightarrow{\text{parentCompanyOf}}$ Sky TV plc $\xrightarrow{\text{keyPerson}}$ Rupert Murdoch |
| | (employerOf) | 24 | Twitter $\xrightarrow{\text{employerOf}}$ Dick Costolo |
| | (foundedBy) | 24 | Foxconn $\xrightarrow{\text{foundedBy}}$ Terry Gou |
| | (subsidiary, keyPerson) | 20 | Samsung $\xrightarrow{\text{subsidiary}}$ Samsung Electronics $\xrightarrow{\text{keyPerson}}$ Lee Kun-hee |
| US-Capital | (deathPlaceOf, deathPlace) | 491 | Delaware $\xrightarrow{\text{deathPlaceOf}}$ Nathaniel B. Smithers $\xrightarrow{\text{deathPlace}}$ Dover, Delaware |
| | (part, isPartOf) | 123 | Delaware $\xrightarrow{\text{part}}$ Delaware Valley $\xrightarrow{\text{isPartOf}}$ Dover, Delaware |
| | (headquarterOf, location) | 112 | Kansas $\xrightarrow{\text{headquarterOf}}$ State Library of Kansas $\xrightarrow{\text{location}}$ Topeka, Kansas |
| | (jurisdictionOf, location) | 104 | Kansas $\xrightarrow{\text{jurisdictionOf}}$ Kansas Department of Revenue $\xrightarrow{\text{location}}$ Topeka, Kansas |

of most relevant facts (as indicated by the paths) for the triple (*Berkshire Hathaway*, *keyPerson*, *Warren Buffett*), with the width of edges roughly proportional to their net flow $\mathcal{W}(P_{s,p,o})$. See Fig. 1 for another example. Notice the diversity in the set of facts that support these triples. Also note how Knowledge Stream is able to “bubble up” the most intuitively relevant facts by channeling a large flow through their corresponding paths (indicated by their wider edges). Other approaches rely on the availability of path patterns that are either curated by knowledge engineers or mined using a large number of labeled examples. KS automatically surfaces relevant ground facts in an unsupervised way. We believe that it is the first computational fact-checking approach featuring such an expressive power.

IV. RELATED WORK

Fact checking is an important activity to prevent dubious claims and unverified rumors from spreading. Preliminary computational approaches have employed metadata and other contextual indicators around entities of interest, e.g., characteristic features in user account metadata, unexpected shifts in temporal signals, credibility, and so on. For example, Truthy [17], Rumorlens [18], TweetCred [19], and Claim-Buster [20] are systems whose aim is to study the spread of misinformation and rumors, and identify interesting claims to check. The Hoaxy system [21] tracks claims and fact checks to study their interplay. By design, these systems do not attempt to understand the actual contents of claims, which limits their applicability.

Other approaches focus on checking the content of a claim based on prior knowledge, typically found in a knowledge base or knowledge graph. We can distinguish two broad classes of

methods based on how easy it is to interpret their results. On the one hand, we have approaches inspired by logical reasoning (e.g., ILP [22] and AMIE [23]), which mine first-order Horn clauses and are thus easy to interpret. On the other, there are statistical learning models (e.g., RESCAL [24], TransE [10], TransH [25], TransR [26], and ProjE [27]) that create vector embeddings for entities and relations, which can be used to assign similarity scores. Statistical approaches can be particularly hard to interpret, but they are great at handling uncertainties and capturing global regularities in the KG. Unfortunately, scalability is an issue for both types of approaches, as many of the algorithms mentioned above struggle to perform in the face of large-scale KGs, due to either large search spaces or high model complexity. Nickel *et al.* [24] review a number of these models.

Only a few approaches fall somewhere in the middle of this interpretability spectrum. Ciampaglia *et al.* [8] propose an approach that relies on a single short, specific path to differentiate a true fact from a false one. Although intuitive, their algorithm fails to account for the semantics of the target predicate.

PRA [6] and PredPath [7] mine the KG in search of paths connecting the subject to the object of a triple, and use the predicate labels found along these paths to identify features for a supervised learning framework. Labeled examples of true and false triples are therefore needed at the stage of feature selection and during model training. Both approaches spend significant computational resources on feature generation and selection. And even though they rely on massive amounts of features, most provide only a very weak signal. Nevertheless, they have been shown to be very effective on fact-checking

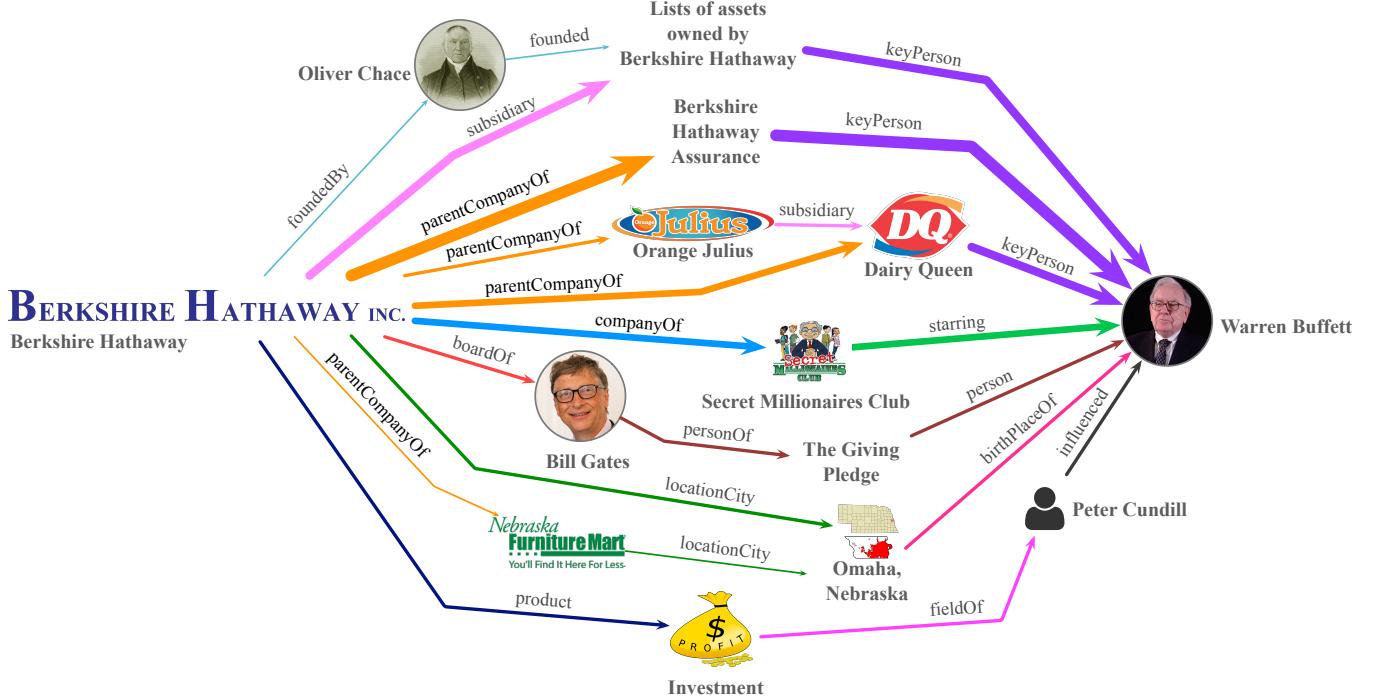


Fig. 5. Relevant facts about a target claim as surfaced by Knowledge Stream.

test cases [7], and in large-scale machine reading projects [28], [29]. They also offer some interpretability due to the features (or rules) they learn. Our methods achieve comparable or better performance while offering greater interpretability and expressiveness in terms of supporting facts, and without training — except for “learning” the edge capacities (see Section II-A).

Most of the approaches (including ours) described above have focused on checking claims as simple as a triple. Since a triple is a link in the graph, an impressive array of link prediction algorithms in dynamic networks [11], [30], [31] can be applied to the task of fact checking. However, these approaches mainly rely on elementary structural cues, leading to poor performance on many fact-checking test sets [7] (see also Table III).

V. DISCUSSION AND FUTURE WORK

Network flow theory [12] has guided the design of many applications in engineering, logistics, manufacturing, and so on. In this paper, we have shown that it can also serve as a useful toolbox for reasoning about facts, and for fact checking in particular.

We presented two novel, unsupervised approaches to assess and explain the truthfulness of a statement of fact by leveraging its semantic context in a knowledge graph. Knowledge Stream is based on network flow and employs multiple short paths; Relational Knowledge Linker finds a single shortest path. In pursuing both approaches, we also proposed a novel

method to measure the similarity of any two relations purely based on their co-occurrence in the KG.

We evaluated both approaches on a diverse set of real-world and synthetic test cases, and found that their performance is on par with the state of the art. Moreover, we saw that, in many cases, multiple paths can provide additional evidence to support fact checking. Our Knowledge Stream model offers high expressive power by its ability to automatically surface useful path patterns and relevant facts about a claim. Based on this experience we believe that network flow techniques are particularly promising for fact checking.

Knowledge Stream is still a preliminary approach for computational fact checking, leaving much room for improvement to address complex test cases. For example, the success of KS hinges on the appropriate design of edge capacities in the graph. We have explored the use of relational similarity for this purpose. The development and evaluation of effective relational similarity metrics is an important avenue of future work. The capacities could also incorporate metadata from the KG itself, for example confidence scores from the information extraction phase (see, e.g., YAGO [32]).

In surfacing facts relevant to a triple, we ranked the set of paths in a stream based on their flow values. Devising alternative ways to rank such facts, reflecting their novelty, diversity, or serendipity (as is done in evaluating recommender systems) is another interesting thread of future research.

Many KGs (e.g., YAGO2 [4] and Wikidata [5]) now contain facts augmented with spatio-temporal details. Checking facts

that may be true only during a certain time frame or at a certain location is another important challenge. One way to extend KS to handle such facts could be to bias the search toward those areas of the KG that may contain facts valid during that interval or near that place.

Lastly, our version of KS relies on successive path-finding, which can be slow for triples involving subjects with a large search space. Our implementation takes a few minutes to check each triple with DBpedia. Other approaches could be explored in the future. For example, the network simplex algorithm [12] has better theoretical and run-time behavior.

ACKNOWLEDGMENTS

The authors would like to thank B. Shi and T. Weninger for sharing their evaluation data. This work was supported in part by NSF (Award CCF-1101743) and DARPA (grant W911NF-12-1-0037). Funding agencies had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

REFERENCES

- [1] L. Howell, “Digital wildfires in a hyperconnected world,” World Economic Forum, Tech. Rep., 2013, online; accessed 19-August-2015. [Online]. Available: <http://reports.weforum.org/global-risks-2013/risk-case-1/digital-wildfires-in-a-hyperconnected-world/>
- [2] A. Zubiaaga, A. Aker, K. Bontcheva, M. Liakata, and R. Procter, “Detection and resolution of rumours in social media: A survey,” *arXiv preprint arXiv:1704.00656*, 2017.
- [3] C. Bizer, J. Lehmann, G. Kobilarov, S. Auer, C. Becker, R. Cyganiak, and S. Hellmann, “Dbpedia-a crystallization point for the web of data,” *Web Semantics: science, services and agents on the world wide web*, vol. 7, no. 3, pp. 154–165, 2009.
- [4] J. Hoffart, F. M. Suchanek, K. Berberich, and G. Weikum, “Yago2: A spatially and temporally enhanced knowledge base from wikipedia,” *Artificial Intelligence*, vol. 194, pp. 28–61, 2013.
- [5] F. Erxleben, M. Günther, M. Krötzsch, J. Mendez, and D. Vrandečić, “Introducing wikidata to the linked data web,” in *International Semantic Web Conference*. Springer, 2014, pp. 50–65.
- [6] N. Lao and W. W. Cohen, “Relational retrieval using a combination of path-constrained random walks,” *Machine Learning*, vol. 81, no. 1, pp. 53–67, 2010.
- [7] B. Shi and T. Weninger, “Discriminative predicate path mining for fact checking in knowledge graphs,” *Knowledge-Based Systems*, vol. 104, pp. 123–133, 2016.
- [8] G. L. Ciampaglia, P. Shiralkar, L. M. Rocha, J. Bollen, F. Menczer, and A. Flammini, “Computational fact checking from knowledge networks,” *PIOS ONE*, vol. 10, no. 6, p. e0128193, 2015.
- [9] M. Nickel, V. Tresp, and H.-P. Kriegel, “A three-way model for collective learning on multi-relational data,” in *Proc. of the 28th International Conference on Machine Learning (ICML-11)*, 2011, pp. 809–816.
- [10] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, “Translating embeddings for modeling multi-relational data,” in *Advances in Neural Information Processing Systems*, 2013, pp. 2787–2795.
- [11] D. Liben-Nowell and J. Kleinberg, “The link-prediction problem for social networks,” *Journal of the American society for Information Science and Technology*, vol. 58, no. 7, pp. 1019–1031, 2007.
- [12] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network Flows: Theory, Algorithms, and Applications*. Prentice hall, 1993.
- [13] E. W. Dijkstra, “A note on two problems in connexion with graphs,” *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.
- [14] L. Katz, “A new status index derived from sociometric analysis,” *Psychometrika*, vol. 18, no. 1, pp. 39–43, 1953.
- [15] L. A. Adamic and E. Adar, “Friends and neighbors on the web,” *Social networks*, vol. 25, no. 3, pp. 211–230, 2003.
- [16] B. Borel, *The Chicago Guide to Fact-Checking*. University of Chicago Press, 2016.
- [17] J. Ratkiewicz, M. Conover, M. Meiss, B. Gonçalves, S. Patil, A. Flammini, and F. Menczer, “Truthy: Mapping the spread of astroturf in microblog streams,” in *Proc. of the 20th International Conference Companion on World wide web*. ACM, 2011, pp. 249–252.
- [18] P. Resnick, S. Carton, S. Park, Y. Shen, and N. Zeffner, “Rumorlens: A system for analyzing the impact of rumors and corrections in social media,” in *Proc. Computational Journalism Conference*, 2014.
- [19] A. Gupta, P. Kumaraguru, C. Castillo, and P. Meier, *TweetCred: Real-Time Credibility Assessment of Content on Twitter*. Springer International Publishing, 2014, pp. 228–243.
- [20] N. Hassan, B. Adair, J. T. Hamilton, C. Li, M. Tremayne, J. Yang, and C. Yu, “The quest to automate fact-checking,” *world*, 2015.
- [21] C. Shao, G. L. Ciampaglia, A. Flammini, and F. Menczer, “Hoaxy: A platform for tracking online misinformation,” in *Proc. of the 25th International Conference Companion on World Wide Web*. International World Wide Web Conferences Steering Committee, 2016, pp. 745–750.
- [22] S. Muggleton, R. Otero, and A. Tamaddoni-Nezhad, *Inductive Logic Programming*. Springer, 1992, vol. 38.
- [23] L. A. Galárraga, C. Teflioudi, K. Hose, and F. Suchanek, “AMIE: Association rule mining under incomplete evidence in ontological knowledge bases,” in *Proc. of the 22nd International Conference on World Wide Web*. ACM, 2013, pp. 413–422.
- [24] M. Nickel, K. Murphy, V. Tresp, and E. Gabrilovich, “A review of relational machine learning for knowledge graphs,” *Proc. of the IEEE*, vol. 104, no. 1, pp. 11–33, 2016.
- [25] Z. Wang, J. Zhang, J. Feng, and Z. Chen, “Knowledge graph embedding by translating on hyperplanes,” in *AAAI*. Citeseer, 2014, pp. 1112–1119.
- [26] Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu, “Learning entity and relation embeddings for knowledge graph completion,” in *AAAI*, 2015, pp. 2181–2187.
- [27] B. Shi and T. Weninger, “ProjE: Embedding projection for knowledge graph completion,” *CoRR*, vol. abs/1611.05425, 2017.
- [28] N. Lao, T. Mitchell, and W. W. Cohen, “Random walk inference and learning in a large scale knowledge base,” in *Proc. of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2011, pp. 529–539.
- [29] X. Dong, E. Gabrilovich, G. Heitz, W. Horn, N. Lao, K. Murphy, T. Strohmann, S. Sun, and W. Zhang, “Knowledge vault: A web-scale approach to probabilistic knowledge fusion,” in *Proc. of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2014, pp. 601–610.
- [30] A. G. Maguitman, F. Menczer, F. Erdinc, H. Roinestad, and A. Vespiagnani, “Algorithmic computation and approximation of semantic similarity,” *World Wide Web*, vol. 9, no. 4, pp. 431–456, 2006.
- [31] L. Lü and T. Zhou, “Link prediction in complex networks: A survey,” *Physica A: Statistical Mechanics and its Applications*, vol. 390, no. 6, pp. 1150–1170, 2011.
- [32] F. M. Suchanek, G. Kasneci, and G. Weikum, “Yago: A core of semantic knowledge,” in *Proc. of the 16th International Conference on World Wide Web*, ser. WWW ’07. New York, NY, USA: ACM, 2007, pp. 697–706. [Online]. Available: <http://doi.acm.org/10.1145/1242572.1242667>