Automatic Fact Checking for Knowledge Graphs: An Experimental Study

(unpublished draft)

Peng Lin[1], Yinghui Wu[1,2], Hanchao Ma[1], Jialiang Shen[3]
[1]Washington State University [2]Pacific Northwest National Laboratory
[3]Beijing University of Posts and Telecommunications
{plin1@eecs., yinghui@eecs.,hanchao.ma@}wsu.edu, shenjialiang@bupt.edu.cn

Abstract

Fact checking is a cornerstone task for knowledge base management. Given a knowledge graph G and a statement of fact t not in G, it is to decide whether t belongs to a missing part of G. A number of algorithms have been developed for fact checking in knowledge graphs. To gain an understanding of the performance of these methods, we conduct a study to experimentally compare state-of-the-art fact checking models. We categorize fact checking methods into four categories (*rule-based*, *path-based*, *subgraph-based*, and *embedding-based*), and investigate representative models from each category. Using real-world knowledge bases, we evaluate the models intensively in their accuracy and training cost. (1) We show that there is "no single winner" among these methods for diversified fact checking tasks. (2) To understand the performance of these methods, we provide in-depth case analysis to identify and demonstrate the impact of key factors. We also discuss the cases that are not well addressed by any of the methods we evaluated. (3) Moreover, we propose and evaluate two pragmatic strategies: (a) to enhance fact checking with interpretable graph patterns, and (b) to leverage ensemble-based approaches to improve the robustness of fact checking. We summarize the characteristics of different methods, and provide guidelines on selecting appropriate models for various fact checking scenarios.

1. Introduction

Knowledge graphs have been widely adopted to support information systems in *e.g.,* web search [8], recommendation [31] and decision making [15]. A knowledge graph consists of a set of edges, and each edge encodes a fact as a triple statement $<v_x$, r, $v_y>$, where v_x and v_y denote a *subject* entity and an *object* entity, respectively, and r refers to a *predicate* (a relation) that connects v_x and v_y. Existing knowledge graphs include Knowledge Vault [8], YAGO [38], DBpedia [22], Wikidata [40], Freebase [2], NELL [4], and ontologies such as WordNet [27].

There have been a host of work for knowledge base construction [30, 4, 22, 9, 8], refinement [41, 14, 25, 5], error detection [10, 16, 35] (see [32, 29] for survey), and querying [7]. One of the cornerstone tasks for all these work is *fact checking*. Given a knowledge graph G, and a fact t, it is to decide whether t belongs to the missing part of G. The verified facts can be used to (1) directly enrich or refine an incomplete knowledge base, (2) provide cleaned evidences for error detection in dirty knowledge bases, and (3) improve answer quality for query processing in knowledge graphs.

A number of methods have been developed for fact checking in knowledge graphs [12, 21, 14, 18, 36]. These methods learn models that either tell whether a new fact is true or not, or return possible values of missing facts that can be applied to fact checking. The relative performance on the accuracy and training cost of these methods are nevertheless unclear. Moreover, one need to try multiple fact checking tools in practice to find the best solution for checking the possible large amount of facts, which is not feasible. Hence, it is important to study the performance of available solutions for real-world fact checking tasks.

Despite that all methods report accuracy results, it is not easy to compare their performance directly. The principles of the existing models diverse significantly, and moreover, (1) most of existing work lack a systematic experimental study of the targeted methods and their peers in training cost and scalability; and (2) it is not clear how well these methods work over different types of real world facts, and what are critical factors that may impact their performance. Another challenge is that different methods report their performance with quite diversified metrics. It is difficult for practitioners to evaluate their performance in a principled manner.

To understand the performance of state-of-the-art fact checking methods for knowledge graphs, this paper conducts an experimental comparison of automatic fact checking methods using real-world datasets. Although an exhaustive comparison is beyond the scope, we organize and analyze these techniques in four categories by their design principle, and make a first step to compare representative ones from each category as "yardstick" solutions, over a set of intuitive metrics that are applicable to all the methods. Another objective of our work is to make a first step towards benchmarking fact checking tasks with data-driven characteristics and scenarios.

Our goals and contributions are summarized as follows.

(1) We categorize state-of-the-art fact checking methods to four classes: *rule-based*, *path-based*, *subgraph-based*, and *embedding-based*, and provide a review of the design principles for each category. From this analysis, we identify 5 "yardstick" methods, and derive critical factors and configurations for a fair comparison.

(2) We provide a systematic comparison of the fact checking methods, to study (a) how their design principles impact their performance, including both accuracy and efficiency, (b) the impact of critical factors from underlying facts and knowledge graphs, (c) case analysis on different types of facts, to show and explain when they outperform others. We analyze the strength and possible disadvantages of each method. In addition, we provide an analysis for cases that cannot be coped well with all or majority of the methods.

(3) Based on our observations, we provide two strategies that may improve these methods. (a) We perform *localized graph mining* to find small graph patterns and their matches as relevant information "around" true facts, and learn models with features extracted from the matches. The goal is to trade training cost with model accuracy. (b) We investigate *ensemble* methods to combine multiple models,

to improve the robustness of fact checking, measured by the variance of model accuracy over diversified facts. We show that both strategies can improve existing methods, with little tuning effort.

We envision that our observation can provide system designers and users with useful information that can promote the more effective usage and improvement of existing fact checking tools, and the development of new fact checking tools for knowledge base construction and quality management.

The rest of the paper is organized as follows. In Section 2, we give the background of fact checking problem and a categorization of state-of-the-art solutions. We also provide a summary of our evaluation methodology and major observations. In Section 3, we describe the "yardstick" solutions selected from each category. We detail our evaluation methodology, including datasets, evaluation metrics and critical factors in Section 4. We present our experimental results and analysis in Section 5. We summarize our findings before concluding in Section 6.

2. Background and Overview

2.1 Fact Checking in Knowledge Graphs

We start with the formulation of the fact checking problem.

Knowledge graphs. We consider directed graphs G=(V, E, L), which consists of a finite set of nodes V, a set of edges $E \subseteq V \times V$, and for each node $v \in V$ (resp. edge $e \in E$), $L(v)$ (resp. $L(e)$) is a label (value) from a finite alphabet carrying the content of v (resp. e) such as properties, relations or names. An entity $v_x \in V$ may also carry a specific type x.

A fact (triple) $<v_x, r, v_y>$ in a knowledge base [8] can be encoded as an edge e with label $L(e)$=r between a subject v_x of label $L(v_x)$ and type x, and an object v_y of label $L(v_y)$ and type y. We say $<v_x, r, v_y>$ pertains to a triple pattern $r(x, y)$.

Problem statement. For a knowledge graph G and a new fact t, the fact checking problem can be formulated as follows.

- Input: a knowledge graph G, a fact $t = <v_x, r, v_y>$
- Output: a flag "true" if t belongs to the missing part of G; "false" otherwise.

Two variants of the problem have been studied.

(1) Depending on the "missing" fraction of the facts, the task can be further specialized to decide whether v_x, r, or v_y exists given the existence of the rest in G. The task can thus be represented as a query q in one of the following forms: $<v_x?, r, v_y>$, $<v_x, r?, v_y>$ or $<v_x, r, v_y?>$, which asks if v_x, r or v_y should exist in G, given the other two validated entities in G, respectively.

(2) A more general form of fact checking constructs a query q in the form of $<x?, r, v_y>$, or $<v_x, r, y?>$, which is to find a set of labels for a variable with type x or y that most likely holds, given the other two validated entities in G, respectively.

Other forms of the fact checking problem include: (1) link prediction [29], which aims to develop predictive models that predict the existence (or probability of correctness) of (typed) facts in G; and (2) knowledge graph completion [32], which is to add missing knowledge (as triples) to an underlying knowledge graph.

Remarks. We distinguish fact checking in knowledge graphs with the tasks below. (1) Fact checking for unstructured Web data [34] and structured (relational) data [17, 45]. The former mostly relies on text analysis and crowd sourcing, and typically requires human effort. In this work, we consider methods that only use knowledge graph (with possible labeled facts) as input for fact checking. Facts in the latter task are often modeled as parameterized (SQL) queries [45]. The quality of the facts is evaluated by perturbing the parameters and observing the positive evidence from the query results. None of these work addresses fact checking in semi-structured (graph) data. (2) Error detection. From the data quality perspective, fact checking focuses more on *completeness*, and error detection aims at a different goal of "free-of-error" [32].

2.2 Current Solutions

Fact checking approaches as a knowledge base construction task can be categorized to two general groups [29]: *curated/collaborative*, where facts are manually created and verified, and *automated semi- or un-structured*, where facts are automatically verified by learned models (rules) from semi-structured data or unstructured data. We are interested in comparing automatic approaches that may scale to Web-scale data with support of database techniques. We consider four representative classes of methods in this category.

Rule based methods [13, 12, 42, 19, 33]. These work infer the correctness of a fact by testing if it matches one or more of the rules. These rules may be in the form of ontological First-order logic rules [19], physical rules (to specify semantic constraints) [42], association rules extended with Horn clauses [13, 12], or probabilistic logical rules [33]. For automatic methods, the rules are mined from knowledge graphs and a set of validated true facts Γ^+.

Path based methods [21, 41, 6, 36]. A common assumption of path-based methods is that a fact $<v_x, r, v_y>$ can be usually "explained" by the validated paths in G that connects v_x and v_y. The fact checking problem is usually treated as supervised link prediction problem. Given a set of validate true facts Γ^+, path-based methods extracts features from sampled paths between v_x and v_y in the facts $<v_x, r, v_y>$ from Γ^+, and trains prediction models (*e.g.,* classifiers) to decide if a new predicate r holds between v_x and v_y. While these work use similar inference process and learning models, the major difference is the path model. For example, one may make use of sampled paths induced from sampled nodes and their neighbors via random walk [21], shortest paths [6], or discriminant (meta) paths, where the same type of nodes are grouped to generate positive and negative examples [36].

Subgraph based methods [14, 46, 11]. The assumption for these work is that the existence of a fact can be predicted by a subgraph and its (topological) features that contain v_x and v_y. Subgraphs can be sampled via random walk [14], derived from positive and negative links to train neural networks [46], mined as graph patterns to extend the consequent in association rules [11], or be derived as dense summaries [39]. Subgraph-based fact checking extends path-based methods, but with enriched feature sets.

Embedding based methods [3, 18, 43, 25]. Embedding based methods (also called latent distance models; cf. [20]) such as TransE [3] and its variants [18, 43, 25] learn embeddings (latent representations) of entities. In these methods, entities are represented as vectors of real numbers (embeddings), and relationships are represented as "translations" in the embedding space: if $<v_x, r, v_y>$ holds, then the embedding of v_y should be close to the embedding of v_x if transformed via vectors of r. Given true facts Γ^+, the goal is then to learn a transformation model that minimizes a ranking loss (quantified by *e.g.,* energy [3]), such that pairs of entities in Γ^+ are closer to each other than those in non-existing relationships.

We are also aware of several other methods that train predictive models with latent feature models. These methods make use of statistical properties of knowledge graphs and typically rely on specific probabilistic models [29]. The path-based and subgraph-based methods are discussed as graph features models in [29]. For the purpose of fair comparison, we do not cover other SRL-based methods (such as tensor factorization and multiway neural networks) due to quite different principles and assumptions.

2.3 Evaluation Methodology

While all the methods remarked earlier can report performance over knowledge graphs, to perform a systematic comparison is nontrivial. We identify several challenges.

(1) *Effectiveness measures*. The effectiveness of existing methods are evaluated with various measures, such as (a) precision and recall for predictive models as binary classifiers [13], (b) cross-validation [36, 21], and (c)ROC and precision-recall curve [6, 46, 36], for models that produce probabilistic scores. These results alone do not tell the relative performances of different methods. This requires a set of effectiveness metrics that are applicable for all methods, from the eye of end users.

(2) *Efficiency measures*. Large-scale and online knowledge base management requires fast model learning. The efficiency analysis should also identify critical factors that impact the learning cost. Prior work lack a systematic evaluation of the scalability and efficiency of the learning cost over large benchmark datasets.

(3) *Diversity of facts*. It is not clear how different types of facts impact the performance of each individual method. A reasonable categorization of the test cases should be prepared to understand the advantages and disadvantages of the methods, and their ability to cope with various fact checking tasks.

We address all the three challenges (see details in Section 4).

Fact-checking methods. We have selected 5 representative methods that cover all four categories. These methods include (a) a rule-based model AMIE+ [13, 12], (b) two path-based methods PRA [21] and KGMiner [24], (c) a subgraph-based method SFE [14] and (d) an embedding-based method TransD [18]. Besides these existing methods, we also evaluate two extensions. One is based on a hybrid model of rule-based and subgraph-based method. The other makes use of an ensemble-based method that aggregates the results of individual methods for fact checking. The goal is to investigate reasonable combination approaches for standalone methods.

These methods can be tuned to be tested under the same experimental configurations (inputs, outputs, parameters and data). Their performance are also comparable based on the same set of evaluation metrics and factors. These ensure a fair comparison of response time and accuracy of these methods.

We review these methods and their parameters in Section 3.

Datasets and test cases. We have selected four datasets as our benchmark datasets, including three popular knowledge bases, and one academic graph. There are three ways to generate training facts [32]: (a) *partial gold standard*, which assumes abundant manually labeled facts with high quality; (b) *silver standard*, which assumes that the given knowledge graphs are of reasonable quality, and samples true facts from the graphs; and (c) *retrospective evaluation*, that mainly relies on human feedback. As both (a) and (c) requires great manual effort, we adopt the silver standard evaluation that is more suitable, and commonly adopted in knowledge base completion [32]. We generate various types of testing cases to understand the pros and cons of the selected methods.

We report details in Section 4.1.

Performance metrics. We evaluate the fact checking methods in terms of five aspects: (1) *effectiveness*, which measures the accuracy of the predictive models; (2) *robustness*, which quantifies the ability that the models cope with various type of facts; (3) *efficiency*, *i.e.,* the learning cost of the models; (4) *interpretability* that measures how one may generate "explanation" of the models, and (5) *usability*, which indicates the tuning effort of the methods. To further understand the effectiveness of the models, we also identify (a) critical factors for each method and investigate their impact to the performance of the models; and (b) specific test cases that a particular method demonstrates the best performance, via case studies.

We ensure a fair comparison by using the same set of training facts, performance metrics and test cases for all methods. We report the details of evaluation metrics in Section 4.3.

Goals. With the above evaluation methodology, we want to answer the following questions.

(1) What is the (relative) performance of the selected methods?

(2) What are critical factors that may impact the performance of these methods and how?

(3) What are the test cases that each method works best and may not work well, and why?

(4) How may the hybrid strategies (graph pattern mining and ensemble methods) improve existing methods?

2.4 Main conclusions

Based on our experimental study, we draw several conclusions.

No Single Winner. In general, there exists no single method that dominates other methods. Different methods demonstrate different strengths for fact checking tasks. For example, path- and subgraph-based models are good at predicting facts with dense and well-connected neighbors. Rule-based and embedding-based methods report high recall and are more stable in coping with various fact checking tasks. Our case studies have identified different cases that each method performs best. We also found that there exists "no-winner" case with true facts that no method works well.

Relative Performance. We also find that the methods from different classes can predict true facts with a small overlap, as verified by our statistical tests and case studies. This indicates that these methods are complementary to each other. Our observation further suggests potentials for ensemble methods. We found that simple combination strategies such as majority voting with selection criteria already improves the effectiveness of the tested base models.

Another opportunity we identified is the hybrid models that combine graph analytical queries supported by scalable database techniques and statistical learning. Our results verify that a simple strategy that enhances path-based model PRA with graph patterns and their matches can already improve its performance in both precision and recall, incurring little additional learning cost.

Robustness and Factors. Factors from both model parameters and underlying knowledge graphs may have a significant impact on the usability of the fact checking models. Methods that rely on topological features such as path- and subgraph- based methods, as well as rule-based AMIE are more sensitive to graph data properties and diversity. Embedding-based models, on the other hand, are quite stable and report good accuracy in most cases.

Interestingly, a common observation that fact checking for functional relations has higher accuracy than non-functional counterparts, reported by rule- and path-based models, may not hold for embedding-based approaches. The embedding model may work better for non-functional predicates. This also suggests that the two classes of methods may complement each other.

Interpretability. We show that some fact checking models are easy to be interpreted, such as rule- and path-based models. Even

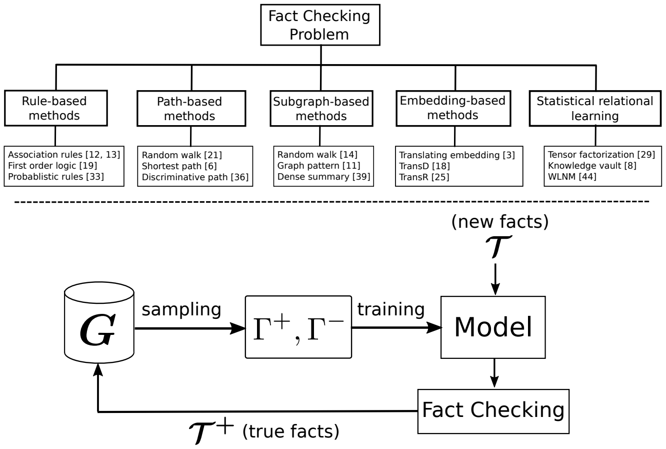


Figure 1: Taxonomy and workflow.

for such models, there is a need to interpret the results at instance level. This calls for new ways that can rank and produce discriminant "explanations" from *e.g.,* path and graph features in path- and subgraph-based models, and for embedding-based approaches.

We present the details of our findings in Section 5.

3. Fact checking methods

We avoid the selection of methods that have been extensively compared from the same category (*e.g.,* the family of embedding-based algorithms), but choose the methods that are shown to be the most effective among their peers from each category. We select five representative methods from the four categories in Section 2.2.

Overview. A general workflow of fact checking methods is illustrated in Figure. 1. It takes a set of positive training facts Γ^+ (facts known to be true), and optionally, a set of negative training facts Γ^- as input. There are two tasks: (1) *Model learning*, which learns a predictive model M for given training facts Γ^+ and Γ^-, and (2) *Fact checking*, which applies the learned model to decide a label "true" or "false" for each new fact.

We next review each method with specified models and workflows, followed by two generic hybrid methods.

3.1 Rule-based Fact Checking

Rule-based methods explain facts in graphs with a set of rules. (1) A simple type of rules are the (manually defined) semantic constraints on the relations, such as "If entity x has relation *isSonOf* an entity y, then y has relation *isFatherOf* to x" [42]. Other methods apply rules extended from conventional association rules with more involved semantics [13]. (2) The model training task typically requires the input of significant measures for rule models (*e.g.,* support, confidence and lift), and applies variants of association rule mining frameworks to discover significant rules. (3) Given a new fact t, and a set of rules Σ, the method checks if t "matches" some rule in Σ, such that both its consequent and antecedent are satisfied. If so, the rule "hit" the fact and t is accepted.

AMIE+. AMIE [13] is a rule-based method for knowledge base construction. It uses (Horn) rules in the form of

$$B_1 \wedge \ldots B_n \Rightarrow r(x, y)$$

where each B_i is an atom (a fact with variables at the subject and/or the object position). (1) The mining of rules extends the significance measures of conventional *support-confidence* framework [1] of association rules for incomplete knowledge bases. The support is defined as head coverage, which addresses rules with a smaller number of matches but are likely to hold, and confidence is enhanced with Partial Completeness Assumption (PCA). It states that if a knowledge graph G contains at least one node v_x that connects to a given node v_y via relation r, then it contains all possible values of y given v_x and r. (2) As a rule-based model, AMIE interprets the facts with the rules that "hit" the facts via variable assignment.

We adopt AMIE+ [12] to implement the mining of AMIE rules. AMIE+ reduces the cost of rule mining by approximating confidence and the rule body size.

3.2 Path-based Fact Checking

Path-based fact checking trains predictive models based on features extracted from sampled paths that connect two entities in training facts. We choose two path-based methods in this category: PRA [21] and KGMiner [36].

PRA [21]. Path Ranking Algorithm (PRA) is an established path-based model. It considers fact checking as a supervised link prediction problem, where a fact checking task is in the form of $<v_x$, r, $v_y?>$. Given the value of v_x and a predicate r, the model predicts the possible values of v_y, ranked by a likelihood score. To check whether a specific fact $<v_x$, r, $v_y>$ holds, it verifies whether the value of v_y has a high ranking score.

PRA is a two-step process for generating a feature matrix over node pairs in a graph. (1) For each training fact $<v_x$, r, $v_y>$, PRA constructs a feature vector, where each feature corresponds to a path between v_x and v_y. It samples potentially useful paths between v_x and v_y via random walk, up to a bounded length of sampled paths. Once a set of path features are constructed, it computes a value for each feature as the probability of arriving at v_y, given that a random walk began at v_x. For all training facts, this produces a feature matrix. (2) The feature matrix is then fed to a classification model (which is typically logistic regression). One can use other classifiers desired to learn a model and make predictions on test data.

KGMiner [36] trains classifiers that check if a specific fact $<v_x$, r, $v_y>$ holds. In contrast to PRA's random walk, it explicitly incorporates node labels, and extract features from meta-paths (a sequence of node and edge types on a path) and predicate paths (meta-paths that "describe" the targeted facts). It seeks the discriminative predicate paths which can alternatively represent the fact $<v_x$, r, $v_y>$. To train the classifier, KGMiner enumerates all paths via DFS constrained by a closure property which guarantees no repeated nodes visited twice. To reduce the enumeration cost and select the most discriminative paths, KGMiner uses three strategies: (1) It groups similar paths by Jaccard coefficient, and avoids selecting similar paths. (2) It selects paths with the entropy values above a threshold. (3) It selects the paths which have frequencies above a threshold.

3.3 Subgraph-based Fact Checking

Besides paths, subgraphs are also considered to be helpful to explain the existence of facts and their predictions. Subgraph-based methods aim to extract both semantic and topological features from sampled subgraphs "around" the targeted facts to train predictive models. One intuition is that the more densely connected the two entities are, the more likely a relationship exists between the two entities. We select SFE [14] as the representative method.

SFE [14]. Subgraph Feature Extraction (SFE) learns Logistic Regression models to predict facts, which has also been used by the two path-based methods PRA and KGMiner. In contrast to PRA and KGMiner, the model learning process extracts the subgraphs

Table 1: An overview of datasets.

Dataset	# entities	# triples	# entity labels	# relations	# patterns
YAGO	2.1M	4.0M	2273	33	15.5K
DBpedia	2.2M	7.4M	73	584	8240
Wikidata	10.8M	41.4M	18383	693	209K
MAG	0.6M	1.71M	8565	6	11742

“around” the training facts, induced by Breadth First Search (BFS) traversals. It selects three types of features from the subgraphs to train predictive models: (1) sampled paths induced by random walks as in PRA; (2) “one-sided” features from paths that start from a source but not necessarily ends at a target, and vice versa; and (3) “any-relation” features that enhance sampled paths by replacing some relations with similar ones in vector space. The goal is to enrich feature space to improve the accuracy of the models.

3.4 Embedding-based Fact Checking

Embedding-based methods cope with queries $(v_x, r, v_y?)$ and report possible values of v_y. The key idea is to learn a model that “translates” vector representations of subject entities to that of object entities in the true facts. (1) Entities and relations are mapped to their vector representations (*i.e.,* word embedding [23]). The distance of embeddings of two entities quantifies their difference. (2) True facts $<v_x$, r, $v_y>$ are interpreted by good “translations” in the form of $\mathbf{v_x} + \mathbf{r} \approx \mathbf{v_y}$, where $\mathbf{v_x}$, $\mathbf{r}$ and $\mathbf{v_y}$ refers to the embedding of v_x, r and v_y, respectively. Given a set of true facts Γ^+, the goal is to learn a model that can rank the values of v_y from Γ^+ as high as possible for the corresponding value of v_x and r, by minimizing a loss function $f_r(\mathbf{v_x}, \mathbf{v_y}) = -\|\mathbf{v_x} + \mathbf{r} - \mathbf{v_y}\|$. The idea is first implemented by TransE [3], followed by several other methods that extends TransE.

TransD [18]. While TransE adopts one embedding space for both entities and relations, it may not work well for *e.g.,* non-functional relations [18]. For such relations, the embeddings of corresponding subject and object entities may vary a lot in the vector space, which leads to low scores computed by $f_r(\mathbf{v_x}, \mathbf{v_y})$ even for true facts. TransD generalizes TransE [3] by distinguishing embeddings for entities and relations. This allows more accurate models that cope with both functional and non-functional relations.

3.5 Hybrid Models

We further investigate two hybrid methods. The first aims to improve the interpretability and accuracy of existing models by incorporating fast graph pattern mining. The second aims to improve the usability, by leveraging ensemble methods that combine the results of models and reduce tuning effort.

Enhanced fact checking with graph patterns. Beyond path features, graph patterns may suggest unique topological features that are more discriminant to describe true and false facts, as indicated by recent study on association rules with graph patterns for social recommendation [11]. We are interested in finding how graph patterns can be used to improve fact checking in knowledge graphs.

We develop a hybrid model called GFC, which combines graph mining and supervised link prediction. (1) Given a triple pattern $r(x, y)$ and its true instances from the training facts Γ^+, GFC samples subgraphs of G that contains each true fact, and mines frequent graph patterns from these subgraphs. This creates a set of associated graph patterns $\mathcal{P}$ of $r(x, y)$. (2) It then traces the subgraphs that match the patterns in $\mathcal{P}$, and extracts graph features from these subgraphs to train classifiers for each $r(x, y)$. The features can be fed to any supervised link prediction algorithm. We instantiate GFC with logistic regression, which is used by PRA and SFE.

An Ensemble method. We also want to understand how ensemble approaches can be used to combine the results of standalone methods, to achieve better results with little manual tuning effort. Such methods are desirable, as users often have little prior knowledge of the performance of existing methods.

We developed a second model called Ensemble, which selects and combines the results of a subset of the 5 yardstick methods as base classifiers. We test two combination strategy. (1) *All-voting* (Ensemble-I). This method invokes each standalone method, and assembles the results with majority voting. (2) *Top-k* (Ensemble-II). This method first finds the top-k best standalone methods with a quality measure, *e.g.,* F_1 score, over a set of sampled test cases. It then takes the majority voting results from only these methods. We evaluate the performance of both Ensemble-I and Ensemble-II in Section 4.

4. Experimental Settings

4.1 Datasets

We use 4 real-world knowledge graphs, including three knowledge bases YAGO [38], DBpedia [22] and Wikidata [40], and a citation network MAG [37]. (1) For the three knowledge bases, we extract terms from the content of an entity as its label, and used their types from the knowledge bases. We extract the name or verb from the edge content as the value of predicates. (2) We extracted a fraction of the dataset [37], which consists of 4 types of entities (papers, authors, venues, and affiliations) and in total 8565 labels.

An overview of datasets is shown in Table 1. The “# patterns” refer to total distinct triple patterns in each dataset, which are used to generate training and testing cases (to be discussed). Among all knowledge bases, Wikidata is the largest and most diversified dataset. While YAGO and DBpedia has comparable size, YAGO is more diversified in terms of entity types, and DBpedia has much more different relations.

4.2 Training Facts and Test Cases

Training facts. We follow the principles of existing fact checking methods to generate training facts Γ, as instances of a set of triple patterns. Given a triple pattern $r(x, y)$ and a knowledge graph G, the true facts are sampled from the instances of $r(x, y)$ in G. We follow *local closed world assumption* [8, 13] to construct false facts. It states that if G contains at least one node v_y that connects to a given node v_x via relation r, then it contains all possible values of type y given v_x of type x and r. Under this assumption, the false facts of $r(x, y)$ include pairs (v_x, v_y) with one of the three cases: (a) (v_x, r', v_y) and $r' \neq r$, (b) (v_x, r, v') in G with v' ranges over the nodes connected to v_x via r, none with a label that equals to v_y's; or (c) (v'', r, v_y) in G with v'' ranges over the nodes connected to v_y via r, none with a label that equals to v_x's.

Test cases. A test case consists of a triple pattern $r(x, y)$, and a set of test facts that are instances of $r(x, y)$. We consider four cases.

(a) *Functional* cases refer to test facts that pertain to a functional predicate (a “one-to-one” mapping). For example, a relation *capitalOf* is a one-to-one mapping between two locations (*e.g.,* <London, *capitalOf*, UK>).

(b) *Pseudo-functional cases* carry predicates that can be “one-to-many” but have a high functionality (“usually” functional). For example, relations *graduatedFrom* and *isCitizenOf* are not necessarily functional, but are functional for most *person* entities.

(c) *Reversed pseudo-functional cases* These include facts with inversed predicates ("many-to-one") as a pseudo-functional predicate, such as *graduatedFrom*$^{-1}$.

(d) *Non-functional* cases pertain to predicates that allow many-to-many mapping, *e.g., ActIn* between type *actor* and *movie*.

For each category, we generate a set of triple patterns. We then samples a group of facts from knowledge graphs that pertain to each triple pattern. We generate diversified test cases by mixing facts from different groups.

4.3 Evaluation metrics

We measure the performance of the selected methods from five aspects: effectiveness, robustness, efficiency, interpretability, and usability, defined as follows.

Effectiveness. We consider accuracy and relative effectiveness.

Accuracy. We adopt standard accuracy measures for each standalone method. These include prediction rate (Pred), precision (Prec), recall (Rec), and F1 score (F_1). Given a set of test facts denoted by $\mathcal{T}$, and a fact checking model M, the true positive TP (resp. false negative FN) of M is defined as the number of true facts in $\mathcal{T}$ that are correctly predicted (resp. are predicted to be false), and the true negative TN (resp. false positive FP) is the number of false facts in $\mathcal{T}$ that are correctly predicted (resp. are predicted to be true) by M. Based on these, (1) prediction rate is the ratio of facts that can be predicted correctly by model M, *i.e.,* $\mathsf{Pred} = \frac{\mathsf{TP}+\mathsf{TN}}{|\mathcal{T}|}$. (2) We adopt standard precision, recall and F_1 score, defined as follows.

$$\mathsf{Prec} = \frac{\mathsf{TP}}{\mathsf{TP}+\mathsf{FP}} \quad \mathsf{Rec} = \frac{\mathsf{TP}}{\mathsf{TP}+\mathsf{FN}} \quad F_1 = 2 \cdot \frac{\mathsf{Prec}\cdot\mathsf{Rec}}{\mathsf{Prec}+\mathsf{Rec}}$$

Relative effectiveness. A direct comparison of standard accuracy does not tell us how significantly one method outperforms another. We also compare the relative performance of the methods against the same set of training and testing facts with p-values [44]. Given two models M_1 and M_2, and their predicted results, a p-value that rejects the hypothesis of "M_1 and M_2 demonstrate statistically indistinguishable performance" indicates a significant winner, rather than "beat by chance".

Efficiency. We compare the time cost and scalability of model learning for all methods. For rule-based model AMIE, we report the rule mining cost of AMIE+. For PRA, SFE and TransD, the total cost includes the time of feature extraction and model training.

Besides the above objective measures, we also evaluate two subjective measures via pragmatic case analysis.

Robustness. Given a model M and a set of test cases Σ, where each test case is a group of test facts that pertain to a triple pattern $r(x, y)$, we measure the robustness of M over Σ as the standard deviation of the F_1 score over test case; the smaller the value is, the more robust M is.

Interpretability. For rule-based model AMIE, we investigate AMIE rules used to make the prediction. For path-based models PRA and KGMiner, we examine the path features with high weights. For subgraph-based SFE and hybrid model GFC, we extract subgraphs that contribute to features with high weight in the model. We manually verify these structures as "explanations" to the facts.

Usability. We measure the usability in terms of the number of parameters the method requires to learn the model, and the effort to tune the model to its best performance over given test cases.

4.4 Configuration of Methods

We make effort to calibrate the parameters of all the methods, and find the best configuration that achieves their best results.

<u>AMIE+</u>. We adopt the source code of AMIE+ [1]. We adapted the code to enable (1) the training of rules over a set of given true facts Γ^+; and (2) a fact checking component, following the evaluation in [12]. Given a fact $<v_x, r, v_y>$, it accepts the fact if there exists a rule, such that the fact matches its consequent and antecedent. Otherwise, the fact is labeled "false".

The method requires three parameters: maximum size of rules b_1 (*i.e.,* the number of Horn clauses in a rule), support threshold and confidence threshold. By default, we set b_1 as 3, support as 0.0001 and confidence as 0.01, which achieves best performance on average over our datasets.

<u>PRA.</u> We use the source code of PRA [2]. The method outputs a ranked list of the labels for the missing fact (*i.e.,* v_y). Given a fact $<v_x, r, v_y>$, the method verifies if the label of v_y exists in the top-ranked list. If so, it accepts the fact. Otherwise, it returns false.

The method requires three parameters: maximum path length b_2 to be sampled, number of random walks per source node, and number of paths to be kept for each source. We set these parameters as 3, 100 and 1000 to achieve the best performance over our datasets.

<u>KGMiner.</u> We use the source code of KGMiner [3]. The original code directly output the result of cross-validation test. For a fair comparison, we slightly modified the tool to let it output binary labels in a train-test pipeline, to be consistent with other methods. The tool requires a parameter b_3 of maximum length of predicate paths, which is set to 3 by default.

We were not able to evaluate KGMiner for all cases, as it incurs a high cost by enumerating all paths for each training fact, and performing pairwise comparison of the paths to select features. The code adopts a server-client mode which cannot be easily deployed on our system. Our smaller scale tests find that its learning time is not very predictable, varying from a few minutes to 8 hours. Thus, we only report accuracy and case studies for KGMiner.

<u>SFE.</u> SFE [2] applies the same implementation of PRA (with enhanced features) and outputs binary labels. It takes three parameters: the radius b of the subgraphs to be sampled that contain training facts, the type of path extraction method, and the feature types. We set path extraction method as BFS, a bi-directed breadth-first traversal from v_x and v_y, and extract both PRA features and enriched features (*e.g.,* "AnyRelation").

We consider the radius b as a critical parameter for SFE. To evaluate its impact, we evaluate SFE under two settings, where b is set as 1 (SFE (b=1)) and 2 (SFE (b=2)), respectively.

<u>TransD.</u> We use the source code of a graph embedding library that implements TransD, available at [4]. This implementation answers fact checking queries in the form of $(v_x, r?v_y)$. Given a pair of entities (v_x, v_y), it returns a list of top $\alpha\%$ predicates that are likely to hold between v_x and v_y. It requires three parameters: threshold $\alpha\%$, dimension of vector space and number of training iterations. We set them as 10%, 100 and 1000, respectively, to ensure the training converges with a good performance.

<u>*Multi-threaded implementation.*</u> All the code base we used for the models AMIE+, PRA, SFE and TransD support multi-threaded

[1] https://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/yago-naga/amie/

[2] https://github.com/matt-gardner/pra

[3] https://github.com/nddsg/KGMiner

[4] https://github.com/LIBBLE/LIBBLE-MultiThread/tree/master/ParaGraphE

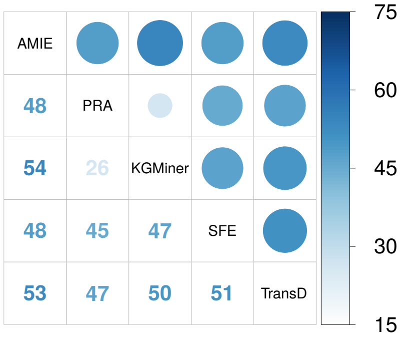


Figure 2: McNemar's test: pairwise comparison.

implementation for faster model learning. For a fair comparison, we set the number of threads as 4 for all the methods.

All of the tests are conducted on a Linux server with 4 Intel Xeon CPUs at 2.30GHz and 50GB memory. Each test was repeated 5 times and the average is reported.

5. Experimental Analysis

We analyze the pros and cons of each method with extensive experiments. Each method is evaluated in terms of effectiveness (accuracy), efficiency/scalability, robustness, interpretability and usability, as defined in Section 2.3.

5.1 Effectiveness

Accuracy. Given a knowledge base, we select 5 triple patterns for each of the 4 fact categories (functional, non-functional, pseudo-functional and inverse pseudo-functional). This generates in total 80 test cases over all datasets. We use stratified sampling to fix the ratio of true and false training facts as 1:4, and fix the ratio of training and testing facts as 4:1. We generate training and testing facts for all test cases, and reported the average accuracy results in Table 2. We find the following.

(1) In general, path-based PRA and subgraph-based SFE have higher prediction rate and precision than their peers AMIE+ and TransD. For example, on YAGO, SFE (b=1) has the highest prediction rate (0.90), followed by PRA. PRA has the highest precision (0.69), followed by SFE (b=1) (0.68). SFE (b=2) has the highest recall (0.89). The path-based model KGMiner has comparable Pred and Prec with PRA, and can sometimes improve the precision and recall of PRA (*e.g.,* on DBpedia), due to its selection of more discriminant paths.

(2) Rule-based AMIE+ and embedding-based TransD in general have a higher recall than PRA and SFE (b=1), especially over MAG. For AMIE+, a fact $<v_x, r, v_y>$is true as long as it matches one of the AMIE rules. For TransD, a predicate r holds as long as it encodes a good translation between subject and object entities with similar vector embeddings. Both have relatively smaller FN but larger FP compared with their peers, thus larger recall. Although PRA and SFE return ranked list of labels for v_y, their prediction models rely on existed topological features, thus report higher FP but smaller FN.

We also found that TransD and AMIE have quite "stable" performance. While all other methods have a worst result for some case, AMIE reports reasonable results for all metrics and in all cases. This is due to that rule models finds statistically significant type-level rules, which are less sensitive to the diversity of facts compared with others that rely on instance-level features.

(3) Notably, SFE (b=2) achieves the highest recall in all cases, but at a cost of lower accuracy and prediction rate. The effectiveness of SFE is sensitive to the radius b. In most cases (except for MAG), larger b decreases its precision, but improves its recall. For example, when b varies from 1 to 2 in Wikidata, the precision changes from 0.74 to 0.38, and recall increases from 0.58 to 0.96. We analyze the impact of b in a separate test, to be discussed soon.

(4) The hybrid model GFC that extends PRA with graph pattern mining reports better precision, prediction rate and recall than PRA in almost all the cases. This verifies the potential of using graph patterns as discriminant features, to improve precision without losing much recall.

For Ensemble strategies, we found that top-k ensemble (Ensemble-II) reports better results in most cases, due to a more careful selection of base methods with better effectiveness. We thus consider Ensemble-II a better strategy in practice.

Relative effectiveness. We perform two tests to evaluate the relative performance of the methods.

"Better than expected". We first investigate the following question: *For imbalanced test facts (a class of 20% true facts, and a class of 80% false facts), how statistically significant it is that a model reports accuracy better than the proportion of the majority class* (i.e., *a random guess)?* We apply binomial test [28] at a significant level (set as 0.05). Given a model M and a test case (with a set of test facts), if the binomial test yields a p-value smaller than 0.05, then it is different than a random guess. We count the number of such p-values. The more, the better.

Table 3 tells us that KGMiner and GFC have more cases that are distinguishable from random models. PRA, SFE, AMIE and TransD has relatively more indistinguishable cases. A possible reason is KGMiner and GFC trains binary classifiers that directly answer fact checking queries (v_x, r, v_y), while others outputs ranked lists to catch true labels. Finally, Ensemble methods have most distinguishable cases, as expected.

Pairwise comparison. We answer the second question: *Given models M_1 and M_2, where M_1 reports better accuracy than M_2, whether M_1 significantly outperforms M_2 or "by chance"?* We apply McNemar's test [26], which computes a score c defined as

$$c = \frac{(|c_{01} - c_{10}| - 1)^2}{c_{01} + c_{10}}$$

where c_{00} (resp. c_{01}) refers to the number of facts are correctly predicated by both M_1 and M_2 (by M_1 but not M_2). Setting a default threshold for c as 3.84, we perform McNemar's test over all 80 test cases, and report the number of test cases with c<3.84. Each such test case suggests that the two models are significantly distinguishable.

The results in Figure 2 justify our selection of methods. It demonstrates that AMIE, PRA, SFE and TransD reports reasonable accuracy with small overlap of correctly predicated test cases (more than half of 80 cases suggest distinguishable performance), thus are complements of each other. This also justifies the performance of Ensemble methods. PRA and KGMiner are more alike due to their similar path-based principle.

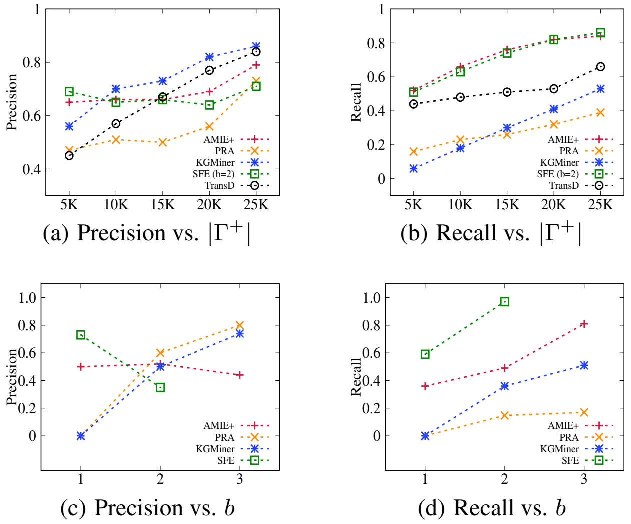
Impact of factors. Using Wikidata, we next evaluate the impact of critical factors to the effectiveness of the models.

Varying $|\Gamma^+|$. We first evaluate the impact of the size of true facts for the five standalone models. We use a fixed set of 15 triple patterns, and varied $|\Gamma^+|$ from $5K$ to $25K$, by increasing the true instances of each triple pattern in Wikidata.

The precision and recall of the methods are reported in Fig-

Table 2: Effectiveness: average accuracy

	YAGO				DBpedia				Wikidata				MAG			
Model	Pred	Prec	Rec	F_1	Pred	Prec	Rec	F_1	Pred	Prec	Rec	F_1	Pred	Prec	Rec	F_1
AMIE+	0.71	0.44	0.76	0.51	0.69	0.50	0.85	0.58	0.64	0.42	0.78	0.48	0.70	0.53	0.62	0.52
PRA	0.87	**0.69**	0.34	0.37	0.88	0.60	0.41	0.45	**0.90**	0.65	0.51	0.53	**0.77**	**0.88**	0.21	0.32
KGMiner	0.87	0.62	0.36	0.40	0.88	0.75	0.60	0.63	0.90	0.63	0.49	0.52	0.76	0.74	0.17	0.27
SFE (b=1)	**0.90**	0.68	0.53	0.57	**0.93**	**0.83**	0.70	**0.73**	0.89	**0.74**	0.58	**0.57**	0.74	0.55	0.36	0.39
SFE (b=2)	0.80	0.56	**0.89**	**0.66**	0.69	0.50	**0.93**	0.60	0.58	0.38	**0.96**	0.52	**0.77**	0.56	**0.83**	**0.63**
TransD	0.76	0.41	0.65	0.48	0.78	0.37	0.56	0.44	0.70	0.40	0.76	0.51	0.63	0.38	0.76	0.48
GFC	0.88	**0.76**	0.46	0.53	0.89	0.80	0.56	0.61	0.87	**0.82**	0.37	0.46	**0.90**	**0.86**	0.62	**0.71**
Ensemble-I	**0.92**	**0.76**	0.71	0.72	0.92	0.79	0.74	0.74	0.86	0.72	0.77	0.67	0.83	0.83	0.52	0.62
Ensemble-II	0.90	0.74	**0.86**	**0.77**	**0.93**	**0.83**	**0.78**	**0.79**	**0.87**	0.76	**0.86**	**0.77**	0.79	0.63	**0.76**	0.65



(a) Precision vs. $|\Gamma^+|$

(b) Recall vs. $|\Gamma^+|$

(c) Precision vs. b

(d) Recall vs. b

Figure 3: Effectiveness: Impact of factors (Wikidata)

Table 3: Binomial test with p-values under 0.05

Model	AMIE+	PRA	KGMiner	SFE
# $p < 0.05$	19	38	42	20
Model	TransD	GFC	Ensemble-I	Ensemble-II
# $p < 0.05$	8	43	54	54

ures 3(a) and 3(b), respectively. More training facts improve the accuracy of most models, as expected. Interestingly, the precision of SFE (b=2) decreases as $|\Gamma^+|$ grows from $5K$ to $20K$, and increases as $|\Gamma^+|$ grows from $20K$. We observe that the number of (enriched) path features significantly increases due to more induced subgraphs from the facts, which leads to "temporal" overfitting. This is later mitigated as more redundant features are merged by SFE, and more training facts are used. The recall of all methods are improved with more true training facts, as expected.

Varying b. We evaluate the impact of bounds b for SFE, as well as a set of comparable bound parameters $b_1 = b$ (rule-size bound) for AMIE, $b_2 = 2 * b$ (path length bound) for PRA, and $b_3 = 2 * b$ (discriminant path length bound) for KGMiner. We varied b from 1 to 3, and varied b_1-b_3 accordingly. Intuitively, larger b indicate more fraction of knowledge graphs is visited to train the models. We do not report the performance of TransD due to embedding-based models do not consider such parameters. Figures 3(c) and 3(d) report the performance of the models.

(1) Larger bound b_2 (resp. b_3) improve the precision of PRA (resp. KGMiner), as larger path length bounds allows more meaningful and discriminant paths to be identified to improve the precision.

(2) The precision of both AMIE and SFE decreases due to larger bound parameters. For AMIE, larger b_1 allows more rules to be discovered, and more false facts are likely to be "hit" by a rule as true, which decreases precision.

For SFE, larger radius b introduces significantly more features due to larger induced subgraphs. As b varies from 1 to 3, the total number of features SFE checks varies from a few hundred to more than a million. With a fixed size of training facts, the model tends to be overfitting over larger b. Indeed, the true positive TP is more likely to decrease and the false positive FP is increasing, reducing the overall precision. It is not very easy to balance SFE between underfitting and overfitting by tuning b alone.

(3) All methods have higher recall when the bounds increases. This is because larger bounds allow more meaningful features (resp. rules) to be discovered for PRA, SFE and KGMiner (resp. AMIE), which lead to models that predicate more true positive facts.

We also observe that the number of path features of SFE significantly increases with larger b. When b=3, more than 1 million features are sampled. The algorithm does not run to completion after 10 hours. We thus omit the results.

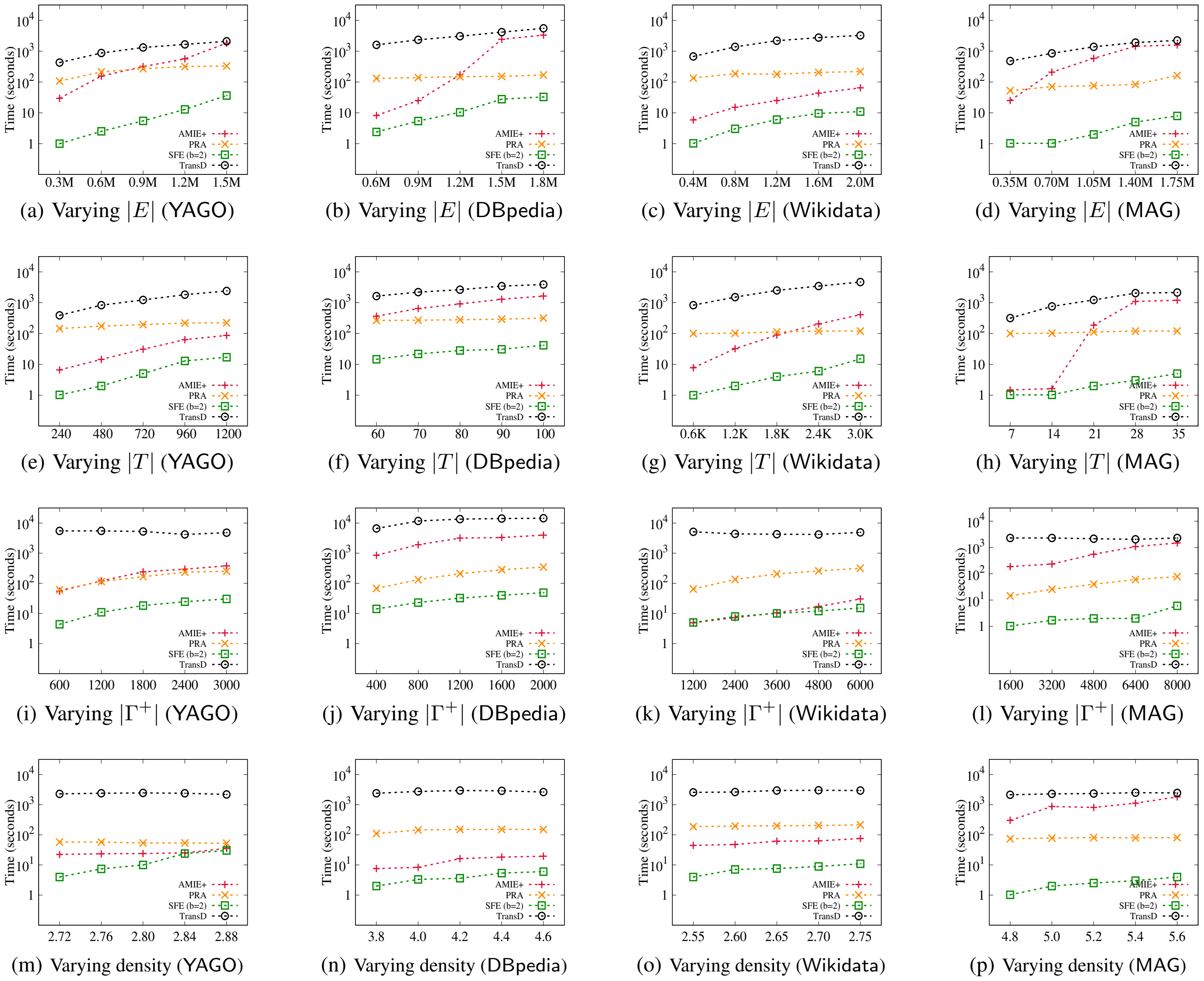
Summary. We observed the following. (1) Path- and subgraph-based methods PRA,KGMiner and SFE often report higher prediction rate and precision, due to that path features can be quite discriminant in explaining true and false facts. (2) Rule- and embedding-based models AMIE and TransD report higher recall, and are less sensitive to the diversity of instance-level features. (3) Adding more true facts in general increases precision and recall of majority of models. Increasing the bound parameters, on the other hand, has a negative impact on the precision of AMIE+ and SFE due to the inclusion of possible useless rules and path features, respectively. (4) Both hybrid models GFC and Ensemble can improve the base methods. Top-k based ensemble outperforms simple majority voting due to a more careful selection criteria.

5.2 Efficiency and Scalability

We next report the efficiency and scalability of model learning for AMIE+, PRA, SFE (with b set to 2) and TransD over all the datasets. We fix the number of threads as 4 for all methods, as remarked earlier.

Average learning cost. Using the same setting as in Table 2, we report the cost of learning each model (not shown). We found that SFE ($b = 2$) outperforms other three methods by 26, 12 and 225 times on average. Among all the models, TransD takes the longest time to train the transformation model. The learning cost of PRA and AMIE+ are comparable.

Impact of factors. We evaluate the impact of four factors: $|E|$ (the number of triples in knowledge graph G), $|T|$ (the distinct triple patterns in G), $|\Gamma^+|$, and a node density measure D defined as the average node degree of G. By default, we fix $|T|$ as 1244, 100, 1621 and 34, $|\Gamma|$ = 20K, 15K, 30K, 50K, D as 2.88, 4.6, 2.75 and 5.6, and $|E|$ as $2.7M$, $3.6M$, $9M$ and $1.7M$, for YAGO, DBpedia,



(a) Varying $|E|$ (YAGO) (b) Varying $|E|$ (DBpedia) (c) Varying $|E|$ (Wikidata) (d) Varying $|E|$ (MAG)

(e) Varying $|T|$ (YAGO) (f) Varying $|T|$ (DBpedia) (g) Varying $|T|$ (Wikidata) (h) Varying $|T|$ (MAG)

(i) Varying $|\Gamma^+|$ (YAGO) (j) Varying $|\Gamma^+|$ (DBpedia) (k) Varying $|\Gamma^+|$ (Wikidata) (l) Varying $|\Gamma^+|$ (MAG)

(m) Varying density (YAGO) (n) Varying density (DBpedia) (o) Varying density (Wikidata) (p) Varying density (MAG)

Figure 4: Efficiency and scalability

Wikidata and MAG, respectively, unless otherwise specified. We report all learning time in log scale.

Varying $|E|$. We created fractions of knowledge graphs by sampling the facts around the training facts, with the total number of facts controlled by $|E|$. The sampling strategy removes 20% edges for each triple pattern in 4 iterations to generate 5 graphs, from the full-sized graph. We report the performance in Figures 4(a)- 4(d), respectively. We found the following.

(1) For all cases, all the methods take more time to learn models over graphs with larger $|E|$. SFE takes least cost, and TransD incurs the highest cost. For example, it takes 11 seconds to learn the classifier over graphs with 2 million facts in Wikidata. With the same setting, TransD takes 3230 seconds to run to completion. The major cost of TransD is incurred by its construction of entity and relation embeddings. With more facts, it takes longer time to construct the embeddings. On the other hand, SFE is quite sensitive to the growth of $|E|$. For example, it becomes 2.8 times slower over DBpedia when $|E|$ is 1.25 times larger. Indeed, more triples lead to more enriched path features for SFE, hence a higher cost.

(2) Among all the methods, AMIE+ is most sensitive to the growth of edges. For example, when $|E|$ becomes 15 times larger in DBpedia, the learning cost is 1.25 times higher. This is because more triple facts in knowledge bases lead to more rule candidates and rule set. For example, the number of rules reported by AMIE varies from 233 to 1421 as $|E|$ varies from 0.3M to 1.5M in YAGO.

(3) The learning cost of PRA, while not the cheapest, is least sensitive as $|E|$ grows. This is because PRA samples up to a bounded number of paths for each training facts, regardless of $|E|$. While AMIE+ take less time over smaller graphs, PRA outperforms AMIE for larger ones (*e.g.,* when $|E| \geq 1.2M$ in DBpedia).

Varying $|T|$. We next evaluate the impact of total distinct triple patterns in knowledge bases, which indicate how "diverse" the graphs are. Fixing $|E|$, D and $|\Gamma^+|$, we sampled graphs with facts that are instances of a set of triple patterns with size controlled by $|T|$.

Figures 4(e)-4(h) tells us that all methods take longer time to learn the models as the graphs become more diversified. AMIE+ is most sensitive, due to more diversified triple patterns lead to larger amount of rules to be verified. SFE is also quite sensitive. We found that its higher cost is mainly due to a significant growth of path feature size, as more types of relations and node labels are introduced in the subgraphs it induces. The learning cost of PRA is again the least sensitive, due to that the path features are extracted

Table 4: Robustness

Model	YAGO	DBpedia	Wikidata	MAG
AMIE+	0.30	0.31	0.25	0.15
PRA	0.43	0.45	0.47	0.18
KGMiner	0.42	0.39	0.47	0.18
SFE	0.18	0.15	0.20	0.04
TransD	0.07	0.09	0.02	0.02
GFC	0.12	0.12	0.09	0.01
Ensemble-I	0.12	0.09	0.08	0.02
Ensemble-II	0.06	0.09	0.07	0.01

from a bounded number of paths for each training facts.

The embedding based model TransD is also sensitive. As remarked earlier, a dominating cost of TransD is the factorization of all the entities and relationships into the vector space. As more types of triple patterns are introduced, the cost of constructing the embeddings also becomes larger.

Varying $|\Gamma^+|$. We next evaluate the learning cost with more true facts. With fixed $|\mathcal{T}|$, $|E|$ and D, we increase $|\Gamma^+|$ by populating it with more distinct triple patterns and more true instances for each triple pattern. We sampled negative facts $|\Gamma^-|$ with a fixed positive/negative fact ratio, *i.e.,* the number of total training facts grows proportionally with $|\Gamma^+|$.

As shown in Figures 4(i)- 4(l), all methods take longer time to learn the models over larger number of training facts. When $|\Gamma^+|$ is larger, PRA is no longer insensitive, and takes more time due to more path features are extracted for more training facts. TransD, on the other hand, are less sensitive. Indeed, with fixed $|E|$ and $|T|$, the cost of embedding construction of TransD is not sensitive to the change of $|\Gamma^+|$.

Varying D. We also investigate the impact of node density D to understand the performance of these models over scale-free knowledge graphs. Nodes with high degrees (*e.g.,* "hub" or "authorities") are commonly seen in such networks whose degree distribution follows a power law. Fixing the number of entities $|V|$ as $1.2M$, $1M$, $1.5M$ and $0.6M$ for YAGO, DBpedia, Wikidata and MAG, respectively, we insert edges to nodes with a higher degree with a higher probability, following "rich gets richer" principle, controlled by average degree D.

Figures 4(m)- 4(p) tell us the following. (1) TransD and PRA are in general not sensitive to the node density. PRA samples a bounded number of paths with a cost that are not sensitive to node degrees. TransD solves an optimization problem in a vector space of entities and relations, and is also not affected by D. In most cases, AMIE+ is not sensitive to D except for MAG. A closer inspection shows that this is due to a set of "super nodes" in MAG such as "venue" and "affiliation" entities, which have large degrees. These super nodes lead to a significant growth of rule candidates and the verification cost for AMIE+. (2) SFE is most sensitive to the change of D. For example, its cost becomes 8 times larger when D changes from 2.72 to 2.88 in YAGO. The reason is larger average degree in induced subgraphs causes more path features to be extracted, increasing the model training cost.

Summary. In general, SFE is the most efficient in learning the model, and TransD takes the longest time. Among the factors we evaluated, (1) PRA is not sensitive to triple size, diversity of graphs and density, and only takes more time over more training facts. (2) SFE is quite sensitive to all the factors, especially $|E|$ and D, due to the growth of path features from its induced subgraphs. (3) AMIE is sensitive to triple size, diversity of graphs and size of training facts. It only takes more time if there are high degree nodes in the training facts that contribute to more rule candidates. (4) The major cost of TransD, typically dominated by the cost of embedding construction, is affected by the number of facts and the number of triple patterns. It is less sensitive to the training fact and density.

We next evaluate the robustness, interpretability and usability of the models. For these subjective measures, we adopt pragmatic case studies.

5.3 Robustness

We selected 20 triple patterns for each category from the tests in Table 2, and derive a set Σ of 80 test cases. We report the standard deviation of its F_1 scores over Σ. Table 4 tells us the following. (1) Both TransD and SFE have lower standard deviation than other methods. This indicate that their performance are more "stable" over various test cases. (2) AMIE+ demonstrates reasonable robustness over all cases. (3) Path-based methods PRA and KGMiner on the other hand have comparable but relatively higher deviation. (4) Compared with SFE and PRA, GFC is more robust over diversified cases. Both Ensemble-I and Ensemble-II have better robustness, due to their combination strategies.

Case analysis. To further understand the pros and cons of each method, we inspect the result for representative test cases.

The good, the bad and the ugly. A majority of the methods report results with highest accuracy over most functional cases, lower accuracy over (reversed) pseudo-functional facts, and lowest accuracy over most of the non-functional facts. For example, the average F_1 score of SFE is 0.8 for functional cases, 0.75 for pseudo-functional cases and 0.42 for non-functional cases, in the 80 test cases. This observation is consistent with the design of path- and rule-based methods, where most of the models use (partial) closed world assumption to create negative training examples. Such examples are indeed false for functional facts, but not necessarily for the rest types of test cases. Thus, a common observation we make is that non-functional facts are in general "harder" to be classified. The results depend on a proper selection of training facts.

Interestingly, this observation does not hold for TransD. For example, the average F_1 score of TransD for functional, pseudo-functional and non-functional cases are 0.12, 0.60, 0.62, respectively. We found that this is due to different output of the optimization process of TransD. The target of TransD is to minimize a score function that quantifies $\mathbf{v_x} + \mathbf{r} - \mathbf{v_y}$. For functional cases, there exists one true fact connecting a given v_x via r to v_y. In this case, the learned translation is exact and quite "specific" from $\mathbf{x}$ to $\mathbf{y}$. For a test fact $<v'_x, r, v'_y>$, even $\mathbf{v'_x}$ and $\mathbf{v'_y}$ are not far away from $\mathbf{v_x}$ and $\mathbf{v_y}$, the embedding $\mathbf{r}$ can no longer yield a small score for $\mathbf{v'_x} + \mathbf{r} - \mathbf{v'_y}$. In other words, the translations learned from functional cases tend to be overfitted for test cases. By contrast, for pseudo-functional and non-functional cases, TransD learns averaged translations that fit to multiple v_x and v_y labels, mitigating the overfitting issue.

"No-single-winner". We selected 24 triple patterns that span four categories (functional, pseudo-functional, reversed pseudo-functional and non-functional), and generated 24 groups of test facts accordingly. We reported the average F_1 score of the methods (including hybrid models) over each test case. The result illustrated in Figure 5 verifies that no single method is dominant in all test cases. We use several cases to demonstrate the issues for each method.

(1)AMIE+ fails to predict any fact for test cases 2 and 8. We found that no (useful) rules are mined given the facts and the setting of default support and confidence thresholds. It takes a great effort to identify better thresholds for such cases.

(2) Both PRA and KGMiner fail to predict any fact $<v_x, r, v_y>$ from test cases 2, 8, 20 and 24. We identified two reasons: (a)

Table 5: Case study: F_1 score over 24 test cases

Type	ID	Test case $r(x, y)$	AMIE+	PRA	KGMiner	SFE	TransD	GFC	Ensemble-I	Ensemble-II
Functional	1	<prefecture, *hasCapital*, district> (YAGO)	0.75	**1.00**	**1.00**	0.67	0.29	0.86	1.00	1.00
	2	<armyOfficer, *wasBornIn*, district> (YAGO)	0.00	0.00	n/a	0.67	0.00	**1.00**	0.00	0.00
	3	<site, *hasCapital*, district> (YAGO)	**1.00**	**1.00**	**1.00**	0.57	0.29	0.89	1.00	1.00
	4	<Organisation, *owningCompany*, place> (DBpedia)	0.67	**1.00**	**1.00**	1.00	0.00	1.00	1.00	1.00
	5	<sportPlayer, *successor*, sportPlayer> (DBpedia)	**1.00**	**1.00**	**1.00**	**1.00**	0.00	1.00	1.00	1.00
	6	<railStation, *isLocatedAt*, village> (Wikidata)	0.67	0.77	**0.93**	0.52	0.63	0.88	0.88	0.93
Pseudo-Functional	7	<woman, *hasChild*, person> (YAGO)	0.43	**0.84**	0.08	**0.83**	0.21	0.61	0.84	0.85
	8	<team, *created*, song> (YAGO)	0.00	0.00	0.00	**0.96**	0.73	0.96	0.00	0.96
	9	<district, *participatedIn*, conflict> (YAGO)	0.52	0.60	**0.92**	0.58	0.67	0.73	1.00	0.93
	10	<Event, *maidenFlight*, transportation> (YAGO)	**1.00**	0.00	0.86	0.55	0.89	0.55	1.00	1.00
	11	<Music, *subsequentWork*, article> (DBpedia)	0.45	**0.93**	**0.93**	0.52	0.42	0.88	0.82	0.93
Pseudo-Functional (*inverse*)	12	<person, *worksAt*, school> (YAGO)	0.52	0.19	0.59	0.56	**0.70**	0.35	0.85	0.80
	13	<person, *hasAdvisor*, alumnus> (YAGO)	0.71	0.22	**0.86**	0.61	0.43	0.45	0.77	0.88
	14	<bridge, *carries*, road> (Wikidata)	0.50	**0.67**	0.00	0.35	0.00	0.44	0.29	0.67
	15	<article, *isOf*, genre> (Wikidata)	**0.83**	0.00	0.75	**0.83**	0.67	0.84	0.95	0.90
Non-Functional	16	<airport, *isConnectedTo*, airport> (YAGO)	0.54	0.35	0.00	**0.84**	0.72	0.43	0.95	0.81
	17	<Politician, *associate*, Politician> (DBpedia)	0.34	0.55	**0.67**	0.44	0.23	0.50	0.58	0.67
	18	<novels, *authoredBy*, human> (Wikidata)	0.36	**0.89**	0.24	0.60	0.56	0.33	0.88	0.88
	19	<film, *basedOn*, book> (Wikidata)	0.32	0.29	0.36	0.41	**0.43**	0.64	0.49	0.49
	20	<human, *employedBy*, college> (Wikidata)	**0.61**	0.00	0.00	0.52	0.44	0.10	0.52	0.71
	21	<AI, *referenceIs*, ML> (MAG)	**0.69**	0.37	0.29	0.65	0.45	0.68	0.74	0.71
	22	<ML, *reference*, ML> (MAG)	0.53	0.49	0.45	**0.67**	0.45	0.67	0.71	0.68
	23	<DB, *reference*, DB> (MAG)	**0.74**	0.41	0.49	0.71	0.40	0.62	0.74	0.82
	24	<author, *isAffiliatedTo*, institute> (MAG)	0.70	0.00	n/a	0.84	**0.88**	0.99	0.65	0.83

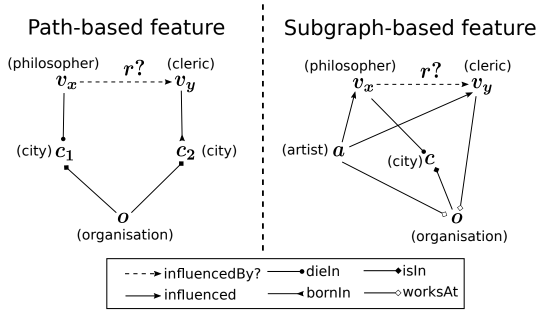


Figure 5: Interpretability: PRA and GFC

"*dangling facts*": there exists no path between the v_x and v_y in knowledge graph, thus no path feature can be extracted by PRA or KGMiner, or (b) v_x and v_y are connected by a set of paths with new features that are not used to train the classifiers (*e.g.,* case 10 and 14). For such facts, path-based models perform worse.

Another issue is the learning algorithm of KGMiner may fail to run to completion for facts that connect to "super nodes" with high degrees due to its high cost incurred by path enumeration. For case 2 and 24, it does not run to completion after 8 hours.

(3) While SFE can cope with dangling facts with its enhanced path features, the major issue is its sensitivity to the setting of radius b, as also indicated earlier. While smaller b seems to be helpful to improve precision, larger b improves recall, at a risk of high learning cost due to the explosion of the number of features.

(4) TransD does not detect cases 4 and 5. These two cases are both functional and have a low number of training examples, which lead to be overfitted models as discussed earlier.

The hybrid models GFC and Ensemble are quite effective over most cases. This is consistent with our observation in Table 2.

"No-winner" cases. Interestingly, we also identified in total 3437 facts that cannot be correctly validated by any of the selected methods. Most of such test facts $<v_x$, r, $v_y>$ are dangling facts that produce isolated entities, or entities with small degree and very sparse neighborhood, upon the removal of r to be predicated. For such paces, AMIE+ cannot mine rules because of sparse neighors, PRA, KGMiner and TransD has few useful features due to sparse topological structures, and TransD cannot find good vectors of r due to the large distance between the embeddings of v_x and v_y. One such example is the test case <person, *wroteMusicFor*, Film> (YAGO), where the true facts <Michal Lorenc, *wroteMusicFor*, Psy> and <Hossein Alizadeh, *wroteMusicFor*, Song of Sparrows> are categorized as false.

5.4 Interpretability

We evaluate the interpretability of each model by extracting significant features. For AMIE, we identify rules with high support, confidence and are often matched with test facts. For PRA, SFE and KGMiner, (discriminant) path features are extracted to explain the facts. We also extracted frequent graph patterns and their matches reported by GFC as "explanations" for the test cases. Interestingly, these substructures also provide insights that help understanding the accuracy of the models.

We demonstrate a test case <philosopher, *influencedBy*, Cleric> from DBpedia, which verifies whether a philosopher is influenced by a cleric. (1) A corresponding rule reported by AMIE+ is *influencedBy*(artist, cleric)$\wedge$ *influencedBy*(philosopher, artist), which states that a philosopher is influenced by a cleric if he is influenced by an artist who is influenced by the cleric. (2) A path feature highly weighted by PRA is shown in Figure 5, which suggests that such influence exists via a path of cities and organizations associated with their birth and death. The path can also be expressed by an AMIE+ rule. (3) We also show a subgraph that matches a pattern from GFC, which shows a more informative structure that combines paths and entities involved in (1) and (2). GFC reports highest F_1 score (0.45) with this specific structure alone, while AMIE+ and PRA reports 0.29 and 0.4, respectively.

A potential issue of the enhanced features from SFE is there may exist no actual paths in knowledge graphs, for some replaced path labels in *e.g.,* "AnyRelation" features. Thus a proper ranking function is needed to suggest meaningful interpretations. It is relatively harder to extract interpretations from embedding-based TransD. New methods are needed to extract good "explanations".

5.5 Usability

We consider the usability in two aspects: number of model parameters, and the effort of parameter tuning to achieve good performance over different test cases. (1) Both AMIE+ and TransD

Table 6: Overall ranking

Model	AMIE+	PRA	KGMiner	SFE	TransD
Effectiveness	Good	Good	Good	Excellent	Fair
Efficiency	Good	Good	Fair	Excellent	Fair
Robustness	Good	Fair	Fair	Fair	Excellent
Interpretability	Excellent	Excellent	Excellent	Good	Fair
usability	Good	Good	Fair	Good	Excellent

have 3 parameters to configure. Nevertheless, it is relatively harder to identify proper support and confidence of rules for AMIE+. (2) PRA and SFE require prameters that are more intuitive, which also indicate a controlling for learning cost. On the other hand, the performance of SFE is sensitive to the radius b, thus it requires more effort to properly set b. (3) KGMiner has only one parameter (path length). On the other hand, the learning performance can be expensive and less predictable. It also takes additional tuning effort to configure the client-server implementation.

5.6 Overall Rating

Putting all our analysis together, we provide an overall ranking of all the methods in Table 6. We highlight the advantages of each algorithm and give an overall summary in the aspects of accuracy, efficiency, robustness, interpretability, usability.

Accuracy. We rank SFE as "Excellent" as it can achieve either higher precision or higher recall (although quite sensitive to the radius bound b). AMIE+ have good recall, and PRA can achieve better precision by adding more facts and increasing path length bound. KGMiner achieves good average accuracy. These three methods are ranked "Good".

Efficiency. The learning cost of PRA is not quite sensitive to graph size and density, and scale well with graph size, hence is ranked "Excellent". SFE is the fastest one in the five algorithms when its radius bound b is no larger than 2, but is not very practical for larger radius bound over large knowledge graphs. AMIE+ scales well with good accuracy results. We rank PRA and amiep as "Good". TransD takes the longest time to learn the translation, with the major cost on embedding construction. KGMiner requires an expensive path enumeration process. Both are ranked "Fair".

Robustness. From our robustness analysis and case studies, we observed that the performance of TransD and AMIE+ are the most stable among the 5 algorithms over all diversified test cases, hence are ranked as "Excellent" and "Good", respectively. The performance of SFE highly depends on its configuration, but achieves reasonable robustness for a fixed setting. KGMiner and PRA rely on the existence of path features. The models perform worse for some specific cases with no or little such path features.

Interpretability. AMIE+ is Horn rule-based models and is very easy to be interpreted. Path-based methods PRA and KGMiner can be interpreted as a set of Horn rules, or a set of discriminant path features. These three methods have "Excellent" interpretability. The interpretability of SFE depends on a proper ranking on possibly multiple grouped path features. In addition, GFC also demonstrates good interpretability using graph patterns (not shown).

Usability. From our experience, TransD is the easiest method to configure with stable performance, and is ranked as "Excellent". PRA, AMIE+ and SFE are also easy to use. Nevertheless, while PRA and SFE provide parameters that indicate learning cost and interpretability, it requires more effort to identify proper support and confidence thresholds to make the best use of AMIE+. Although KGMiner has only one parameter (the path bound), it takes more effort to make the best use of KGMiner. The major reason is the unpredictability of its learning cost and an underlying implementation based on server/client mode.

6. Conclusion and future work

We have systematically reviewed and evaluated five representative fact checking methods, including path-based models, rule-based models, subgraph-based models, and embedding based models, in terms of their accuracy, efficiency (learning cost), robustness, interpretability and usability. We also examined how graph patterns can improve the models that make use of topological features, and how combination strategies can improve the performance of existing models. Our experimental study verifies the following. (1) There is no single method that can dominate the rest in diversified fact checking. The models either demonstrate high precision and low recall and vice versa. (2) Methods from different classes complement others with small overlap of correct predictions. This provides chance for combination strategies such as ensemble methods. (3) The models are also very sensitive to particular parameters, and can be expensive over dense graphs. (4) There is need of new ways to generate explanations and interpretations for fact checking.

Based on these observations, we identify several future directions. (1) *Ensemble-based models*. We have shown that simple strategies such as majority voting already improves the model accuracy. Nevertheless, one needs to pursue prior knowledge of their performance. New holistic models that combine base models are required to dynamically select and combine best models upon request. (2) *Enhanced models with graph patterns*. Graph patterns can improve the performance of models using topological features. New models that integrate fast pattern discovery and statistical learning may perform better with small learning cost. (3) *Generating explanation for facts*. There is a need of tools to produce explanations for knowledge graph completion and exploration. Such tools will greatly help end users in knowledge graph exploration and sensemaking over real-world datasets.

7. References

[1] R. Agrawal, T. Imieliński, and A. Swami. Mining association rules between sets of items in large databases. In *SIGMOD*, 1993.

[2] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD*, 2008.

[3] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko. Translating embeddings for modeling multi-relational data. In *NIPS*, 2013.

[4] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. R. Hruschka Jr, and T. M. Mitchell. Toward an architecture for never-ending language learning. In *AAAI*, 2010.

[5] Y. Chen and D. Z. Wang. Knowledge expansion over probabilistic knowledge bases. In *SIGMOD*, 2014.

[6] G. L. Ciampaglia, P. Shiralkar, L. M. Rocha, J. Bollen, F. Menczer, and A. Flammini. Computational fact checking from knowledge networks. *PloS one*, 2015.

[7] O. Deshpande, D. S. Lamba, M. Tourn, S. Das, S. Subramaniam, A. Rajaraman, V. Harinarayan, and A. Doan. Building, maintaining, and using knowledge bases: a report from the trenches. In *SIGMOD*, 2013.

[8] 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 *KDD*, 2014.

[9] X. L. Dong, E. Gabrilovich, G. Heitz, W. Horn, K. Murphy, S. Sun, and W. Zhang. From data fusion to knowledge fusion. *VLDB*, 2014.

[10] M. Dredze, P. McNamee, D. Rao, A. Gerber, and T. Finin. Entity disambiguation for knowledge base population. In *COLING*, 2010.

[11] W. Fan, X. Wang, Y. Wu, and J. Xu. Association rules with graph patterns. *VLDB*, 2015.

[12] L. Galárraga, C. Teflioudi, K. Hose, and F. M. Suchanek. Fast rule mining in ontological knowledge bases with amie+. *VLDB*, 2015.

[13] L. A. Galárraga, C. Teflioudi, K. Hose, and F. Suchanek. Amie: association rule mining under incomplete evidence in ontological knowledge bases. In *WWW*, 2013.

[14] M. Gardner and T. M. Mitchell. Efficient and expressive knowledge base completion using subgraph feature extraction. In *EMNLP*, 2015.

[15] T. R. Goodwin and S. M. Harabagiu. Medical question answering for clinical decision support. In *CIKM*, 2016.

[16] S. Gottipati and J. Jiang. Linking entities to a knowledge base with query expansion. In *EMNLP*, 2011.

[17] N. Hassan, A. Sultana, Y. Wu, G. Zhang, C. Li, J. Yang, and C. Yu. Data in, fact out: automated monitoring of facts by factwatcher. *VLDB*, 2014.

[18] G. Ji, S. He, L. Xu, K. Liu, and J. Zhao. Knowledge graph embedding via dynamic mapping matrix. In *ACL*, 2015.

[19] S. Jiang, D. Lowd, and D. Dou. Learning to refine an automatically extracted knowledge base using markov logic. In *ICDM*, 2012.

[20] R. Kadlec, O. Bajgar, and J. Kleindienst. Knowledge base completion: Baselines strike back. In *RepL4NLP*, 2017.

[21] N. Lao, T. Mitchell, and W. W. Cohen. Random walk inference and learning in a large scale knowledge base. In *EMNLP*, 2011.

[22] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. Van Kleef, S. Auer, et al. Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web*, 2015.

[23] O. Levy, Y. Goldberg, and I. Dagan. Improving distributional similarity with lessons learned from word embeddings. *TACL*, 2015.

[24] D. Liben-Nowell and J. Kleinberg. The link-prediction problem for social networks. *JASIST*, 2007.

[25] Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu. Learning entity and relation embeddings for knowledge graph completion. In *AAAI*, 2015.

[26] Q. McNemar. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, 1947.

[27] G. A. Miller. Wordnet: a lexical database for english. *CACM*, 1995.

[28] D. S. Moore and G. P. McCabe. *Introduction to the Practice of Statistics.* WH Freeman/Times Books/Henry Holt & Co, 1989.

[29] M. Nickel, K. Murphy, V. Tresp, and E. Gabrilovich. A review of relational machine learning for knowledge graphs. *Proceedings of the IEEE*, 2016.

[30] F. Niu, C. Zhang, C. Ré, and J. W. Shavlik. Deepdive: Web-scale knowledge-base construction using statistical learning and inference. *VLDS*, 2012.

[31] A. Passant. dbrec-music recommendations using dbpedia. In *ISWC*, 2010.

[32] H. Paulheim. Knowledge graph refinement: A survey of approaches and evaluation methods. *Semantic web*, 2017.

[33] J. Pujara, H. Miao, L. Getoor, and W. Cohen. Knowledge graph identification. In *ISWC*, 2013.

[34] C. Shao, G. L. Ciampaglia, A. Flammini, and F. Menczer. Hoaxy: A platform for tracking online misinformation. In *WWW Companion*, 2016.

[35] W. Shen, J. Wang, P. Luo, and M. Wang. Linden: linking named entities with knowledge base via semantic knowledge. In *WWW*, 2012.

[36] B. Shi and T. Weninger. Discriminative predicate path mining for fact checking in knowledge graphs. *Knowledge-Based Systems*, 2016.

[37] A. Sinha, Z. Shen, Y. Song, H. Ma, D. Eide, B.-j. P. Hsu, and K. Wang. An overview of microsoft academic service (mas) and applications. In *WWW*, 2015.

[38] F. M. Suchanek, G. Kasneci, and G. Weikum. Yago: a core of semantic knowledge. In *WWW*, 2007.

[39] A. Thor, P. Anderson, L. Raschid, S. Navlakha, B. Saha, S. Khuller, and X.-N. Zhang. Link prediction for annotation graphs using graph summarization. *ISWC*, 2011.

[40] D. Vrandečić and M. Krötzsch. Wikidata: a free collaborative knowledgebase. *CACM*, 2014.

[41] Q. Wang, J. Liu, Y. Luo, B. Wang, and C.-Y. Lin. Knowledge base completion via coupled path ranking. In *ACL*, 2016.

[42] Q. Wang, B. Wang, and L. Guo. Knowledge base completion using embeddings and rules. In *IJCAI*, 2015.

[43] Z. Wang, J. Zhang, J. Feng, and Z. Chen. Knowledge graph embedding by translating on hyperplanes. In *AAAI*, 2014.

[44] R. L. Wasserstein and N. A. Lazar. The asa's statement on p-values: context, process, and purpose, 2016.

[45] Y. Wu, P. K. Agarwal, C. Li, J. Yang, and C. Yu. Toward computational fact-checking. *VLDB*, 2014.

[46] M. Zhang and Y. Chen. Weisfeiler-lehman neural machine for link prediction. In *KDD*, 2017.