Notable Characteristics Search through Knowledge Graphs

Davide Mottin[1], Bastian Grasnick[2], Axel Kroschk[2], Patrick Siegler[2], Emmanuel Müller[1]
Hasso Plattner Institute
[1]first.last@hpi.de [2]first.last@student.hpi.de

ABSTRACT

Query answering routinely employs knowledge graphs to assist the user in the search process. Given a knowledge graph that represents entities and relationships among them, one aims at complementing the search with intuitive but effective mechanisms. In particular, we focus on the comparison of two or more entities and the detection of unexpected, surprising properties, called *notable characteristics*. Such characteristics provide intuitive explanations of the peculiarities of the selected entities with respect to similar entities. We propose a solid probabilistic approach that first retrieves entity nodes similar to the query nodes provided by the user, and then exploits distributional properties to understand whether a certain attribute is interesting or not. Our preliminary experiments demonstrate the solidity of our approach and show that we are able to discover notable characteristics that are indeed interesting and relevant for the user.

1 INTRODUCTION

Search engines have greatly evolved from simple indexes of pages to complex systems that are able to predict user intentions and answer queries on a variety of data sources. One way to improve the search quality is by using a knowledge graph that represents entities (e.g., Angela Merkel, Germany) as nodes and relationships between them (e.g., leaderOf) as edges in a graph. The great expressiveness of knowledge graphs can complement the search with more flexible search paradigms. Assume for instance a scholar who requires to know some non-trivial facts about Angela Merkel and Emmanuel Macron with respect to other country leaders. It would be interesting to discover for instance that Angela Merkel studied Physics as opposed to most of the other leaders, and that she has no children. We call this fact a *notable characteristic*, to remark the unexpected and non-trivial aspect of the discovery. To this end, we propose a novel type of search called *notable characteristics search* that allows the retrieval of such facts from a set of input query entities. Discovering notable characteristics constitutes a ground for targeted analyses of products (e.g., comparing two cameras effectively) in electronic commerce or microorganisms in biological networks (e.g., two influence bacteria) with respect to a set of similars. As a consequence, in all the cases in which a knowledge graph is available, the discovery of notable characteristics becomes an expressive and powerful search type for any user, from experts and practitioners to novice users.

In our setting, we assume the user provides a set of *query* nodes to be compared and the algorithm finds a set of notable characteristics of these nodes. Given a node, a property is a relationship with other nodes (e.g., leaderOf). A characteristic or property is notable, if it deviates from what one would expect for the kind of nodes (e.g., presidents) into consideration. To the best of our knowledge, this is the first study of automatic discovery of notable characteristics (or properties).

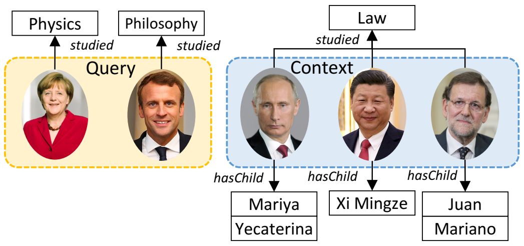


Figure 1: An example knowledge graph, the query (Merkel and Macron), and the discovered context nodes (Putin, Xi Jinping, and Rajoy). The fact that Merkel and Macron do not have children is a notable characteristic.

The discovery of notable characteristics entails two challenges. First, given the set of query nodes we need to compare them to only those nodes that are similar to some extent. Second, we need to select only those properties that are significantly different from the one expressed in the query. Note that tackling the first challenge is very important, as the comparison of the query nodes has to be performed with a set of similar nodes, which we call the context of the query. Consider the naïve approach that returns notable characteristics simply by comparing the query nodes and assume that the user provides "Angela Merkel" and "Theresa May" as query. This is a counter example for the naïve direct comparison, as it will not return the gender as a notable characteristic. Both query nodes are female, however only in comparison with other presidents this becomes an interesting fact. On the other extreme, selecting all the nodes in the graph as context will mislead the analysis towards non-relevant nodes. Take our example of "Angela Merkel" and "Emmanuel Macron". A naïve selection of all humans will not work as context, since the gender characteristic is not notable among all persons.

It is crucial to provide a thorough context selection to prevent the above cases. Therefore, we introduce the discovery of *context nodes*, i.e., nodes similar to the query nodes. An example of the proposed approach is depicted in in Figure 1. To this end, we devise a method that exploits metapaths [12] and random walks for context discovery. We also propose a generic framework that efficiently discover notable characteristics through a novel probabilistic approach based on distribution comparison.

Our contributions are summarized as follows: **(1)** We formalize the problem of *notable characteristics search* given a set of query nodes as input. **(2)** We show how to effectively compute metapaths to find the context nodes in knowledge graphs. **(3)** We introduce a probabilistic approach to discover notable characteristics given a query node set. **(4)** We experimentally evaluate our context selection approach through a user study, and show evidence of our discovered notable characteristics and the real time performance of the proposed algorithms.

2 NOTABLE CHARACTERISTICS SEARCH

We are given a set $\mathcal{A}$ of node labels and a set $\mathcal{L}$ of edge labels. A *knowledge graph* is a directed graph $G : \langle \mathcal{V}, \mathcal{E}, \phi, \psi \rangle$, where $\mathcal{V}$ is a set of nodes, $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is a set of edges, $\phi : \mathcal{V} \mapsto \mathcal{A}$, $\psi : \mathcal{E} \mapsto \mathcal{L}$ are node and edge labeling functions, respectively.

For simplicity, we assume that everything is modeled as relationships and nodes. This is the case for attributes such as birth date: we assume that the date itself is a node connected with a birthdate relationship. Additionally, we assume that for every edge $e \in \mathcal{E}$ with type $\psi(e) = l$ exists a reverse edge e^{-1} with $\psi(e^{-1}) = l^{-1}$ to model cases such as presidentOf and hasPresident. The above assumptions do not change the generality of the methods but simplify the notation and the analysis.

We aim at discovering *notable characteristics* expressed as a set of input query nodes (entities) in relation to their similars. This intuitive definition entails two questions: **(1)** what is the set of similars? **(2)** what are the notable characteristics?

Given a knowledge graph $G : \langle \mathcal{V}, \mathcal{E}, \phi, \psi \rangle$, the set of input nodes, referred to as *query set* or query in short, is any set $Q \subseteq \mathcal{V}$. The query is manually provided by the user and therefore considered reasonably small (i.e., ≤10 elements). The first question concerns the definition of a set of similars referred in this work as *context nodes.* We assume the existence of a *similarity function* $\sigma : \mathcal{V} \times 2^{\mathcal{V}} \mapsto \mathbb{R}$ that assigns a high score to nodes that are similar to those in the query set and low otherwise. Given such similarity, the context are the top-k most similar nodes.

Definition 2.1 (Context set). Given a knowledge graph $G : \langle \mathcal{V}, \mathcal{E}, \phi, \psi \rangle$, a query set $Q \subseteq \mathcal{V}$, a similarity function $\sigma : \mathcal{V} \times 2^{\mathcal{V}} \mapsto \mathbb{R}$, and a parameter k, the *context set* (or simply context) is a set $C \subseteq \mathcal{V}$ such that $Q \cap C = \emptyset, |C| = k$, and for each $n_c \in C \wedge n \in \mathcal{V} \setminus (Q \cup C), \sigma(n, Q) \leq \sigma(n_c, Q)$.

The second question concerns the notable characteristics. The characteristics are attributes or relationships of a specific node since they implicitly represent a signature of the node itself. We assume the existence of a generic *discrimination function* $\delta : \mathcal{L} \times 2^{\mathcal{V}} \times 2^{\mathcal{V}} \mapsto \mathbb{R}_0^+$, which represents how a specific characteristic is discriminative or unexpected comparing two set of nodes. The discrimination function returns 0 if the value is not discriminative. We are now ready to define a notable characteristic.

Definition 2.2 (Notable characteristic). Given a knowledge graph $G : \langle \mathcal{V}, \mathcal{E}, \phi, \psi \rangle$, a query $Q \subseteq \mathcal{V}$, a context $C \subseteq \mathcal{V}$, and a discrimination function $\delta : \mathcal{L} \times 2^{\mathcal{V}} \times 2^{\mathcal{V}} \mapsto \mathbb{R}_0^+$ a *notable characteristic* is a relationship $l \in \mathcal{L}|_{Q \cup C}$ such that $\delta(l, Q, C) \neq 0$.

The notation $\mathcal{L}|_{Q \cup C} = \{l \mid \exists x \in Q \cup C, y \in \mathcal{V} \text{ s.t. } (x, y) \in \mathcal{E} \wedge \psi(x, y) = l\}$ denotes the set of edge labels restricted to those found in the edges directly connected to $Q \cup C$.

The general problem we aim to solve is efficiently returning the notable characteristics, given a query, a similarity function and a discrimination function.

PROBLEM 1 (NOTABLE CHARACTERISTICS SEARCH). *Given a knowledge graph $G : \langle \mathcal{V}, \mathcal{E}, \phi, \psi \rangle$, a query $Q \subseteq V$, a similarity function $\sigma : \mathcal{V} \times 2^{\mathcal{V}} \rightarrow \mathbb{R}$ and a discrimination function $\delta : \mathcal{L} \times 2^{\mathcal{V}} \times 2^{\mathcal{V}} \mapsto \mathbb{R}_0^+$, find the set of notable characteristics.*

3 A PROBABILISTIC SOLUTION

The problem entails the definition of appropriate σ (similarity) and δ (discrimination) functions. Section 3.1 introduces a graph-principled solution based on random walks for retrieving context nodes, while Section 3.2 describes a probabilistic approach to effectively discover notable characteristics.

3.1 Finding the context

Given the query Q, we define a similarity function σ to retrieve a set of context nodes. Although many notions of similarity functions have been developed, such as SimRank [4], none seems suitable to our case since they compare two nodes at the time. We devise an algorithm that takes into account edge labels and combines the advantages of random walk and metapath approaches.

In the random walk model, a walker chooses one of the outgoing edges from a node with uniform probability. Motivated by information theoretic notions applied to graphs [10], instead of uniform probability, we favor edge labels with lower frequency. We define $\mathcal{E}_l = \{(i, j) \in \mathcal{E} | i, j \in \mathcal{V}, \psi(i, j) = l\}$, the set of edges having label $l \in \mathcal{L}$. The frequency of a label l is the fraction of l-labeled edges with respect to the total number of edges. The weighted adjacency matrix is a $|\mathcal{V}| \times |\mathcal{V}|$ matrix, where for each node i and j, $A_{ij} = 1 - |\mathcal{E}_l|/|\mathcal{E}|$ if $(i, j) \in \mathcal{E}$ and $A_{ij} = 0$ otherwise.

The Personalized PageRank is the vector $\mathbf{p}=c\tilde{A}\mathbf{p} + (1 - c)\mathbf{v}$, where $\tilde{A}_{ij}=A_{ji}/\sum_k A_{jk}$, c is the damping factor, and $\mathbf{v}$ is vector called personalization vector. In our experiments the damping factor is 0.8, in line with previous works. We compute $\mathbf{p}$ starting from each node in the query to retrieve the k nodes with the highest score. This is done by setting $\mathbf{v}_n=1/|Q|$ for each $n \in Q$. We refer to this baseline as RANDOMWALK.

The PageRank disregards which type of relationships are involved in the random walk, discarding the valuable information encoded in the surrounding of the query nodes. To this end, we adopt the notion of metapath [8, 12] which generalizes the concept of path. A metapath for a path $\langle n_1, ..., n_t \rangle, n_i \in \mathcal{V}, 1 \leq i \leq t$ is a sequence $\langle \phi(n_1), \psi(n_1, n_2), ..., \psi(n_{t-1}, n_t), \phi(n_t) \rangle$ that alternates node and edge labels along the path.

We mine metapaths as follows. We sample a node in $\mathcal{V} \setminus Q$ uniformly and run a random walk until a query node is reached. The sequence of edge labels m encountered in the random walk is added to the set of metapaths M along with the number of times $c(m)$ the same metapath has been found so far. It has been proved that random walks are effective in mining metapaths [7].

Once the metapaths are retrieved, we compute a score for each node based on the probability that some metapath starting from a query node ends in such node. Given the set of metapaths M, $\{n \overset{m}{\rightsquigarrow} n'\}$ is the set of paths from node n to n' matching metapath $m \in M$. The score of a node $n' \in \mathcal{V} \setminus Q$ with respect to $n \in Q$ is

$$\sigma(n', Q) = \sum_{m \in M, n \in Q} \frac{|\{n \overset{m}{\rightsquigarrow} n'\}|}{|\{n \overset{m}{\rightsquigarrow} n'' | n'' \in \mathcal{V} \setminus Q\}|} \Pr(m) \qquad (1)$$

$\Pr(m) = c(m)/\sum_{m \in M} c(m)$ is the probability of choosing metapath m. Intuitively, σ gives a higher score to nodes that are reachable through frequent metapaths connecting the query nodes or connected through many of these metapaths. Hence, nodes that are reached from infrequent metapaths will have a low score. We refer to this method as CONTEXTRW.

3.2 Comparing the distributions

We revise the definition of notable characteristics in probabilistic terms. Assume we have computed the distribution of values for each characteristic (i.e., edge label) for both query and context nodes found with the method in Section 3.1. Such distribution of the context represents the expected, or *normal* behavior, to be evaluated against the *notable* behaviour of the query set.

Formally, for each characteristic $l \in \mathcal{L}$, we consider two vectors in order to evaluate its notability. The first represents the count of the node labels (e.g., France) connected to a specific edge label (e.g., bornIn). This expresses information about the values in the nodes and can be used to identify cases where different attribute values are relevant. For instance, in the query in Figure 1 all people are European, while in the context half are Europeans and half Asian. We refer to these vectors as *instance vectors*

$$\mathcal{I}_q(l, C, Q) = (x_1, x_2, ..., x_t), \mathcal{I}_c(l, C) = (y_1, y_2, ..., y_t)$$

where x_i and y_i are the number of occurrences of node i at the end of an edge labeled l from a node in Q and C, respectively. In the example in Figure 1, $\mathcal{I}_q(studied, C, Q) = (1, 1, 0)$,

$\mathcal{I}_c(studied, C) = (0, 0, 3)$, where the positions in the vector indicate (Physics, Philosophy, Law). Note that both vectors have the same size, so x_i is zero if i appears only in the context.

Similarly, the second vector represents aggregates over the number of occurrences of a specific edge label in the context, which are useful to represent the characteristic "Angela Merkel" has no child, instead of listing the children names. We refer to these vectors as *cardinality vectors.*

$$\mathcal{K}_q(l, C, Q) = (x_1, x_2, ..., x_t), \mathcal{K}_c(l, C) = (y_1, y_2, ..., y_t)$$

where x_i and y_i are the number of times a node in Q and C respectively has i edges labeled l.

Both vectors can be built by iterating through the nodes in each set and counting the respective occurrences. For a given $l \in \mathcal{L}$, this results in two scores $\delta_{\mathcal{I}}$ and $\delta_{\mathcal{K}}$. The final score δ is the maximum score between $\delta_{\mathcal{I}}$ and $\delta_{\mathcal{K}}$.

$$\delta(l, C, Q) = \max(\delta_{\mathcal{I}}(l, C, Q), \delta_{\mathcal{K}}(l, C, Q)) \quad (2)$$

Many measures have been proposed in statistics to compare two vectors in terms of distributions, such as the Kullback-Leibler (KL) divergence, the χ^2 test, and Earth Mover's Distance (EMD). However, most of them draw specific assumptions, such as non-zero probabilities, or normality, that are not fulfilled in our case since $\mathcal{I}$ and $\mathcal{K}$ have no natural ordering and no distance-function between the values. Therefore, we resort to a more natural multinomial test that better expresses the relationship between our distributions. The multinomial test assumes that a set of observations (the query) is drawn from a multinomial distribution (the context). If the values observed in the query are drawn from the multinomial, than the hypothesis cannot be rejected and the characteristic is marked as non-notable; otherwise, the success of the test denotes that the characteristic is notable.

Assume we have a random variable $X_{N,\pi} \sim Mult(N, \pi)$, with parameters N and distribution π. We normalize $\mathcal{I}_c$ and $\mathcal{K}_c$ to express the probability distributions $\widehat{\mathcal{I}_c} = \mathcal{I}_c / ||\mathcal{I}_c||_1$ and $\widehat{\mathcal{K}_c} = \mathcal{K}_c / ||\widehat{\mathcal{K}_c}||_1$. The significance probability is

$$\Pr_s(X_{N,\pi} = x) = \sum_{y:\Pr(X_{N,\pi}=y) \le \Pr(X_{N,\pi}=x)} \Pr(X_{N,\pi} = y)$$

where $\Pr_s(\pi, x)$ is the probability of x or any equally or less likely outcome being drawn from the probability distribution[1]. A difference in distributions is considered significant if the hypothesis is rejected with probability $p > 0.95$.

$$\mathrm{MT}(\pi, x) = \begin{cases} 1 - \Pr_s(X_{N,\pi} = x) & \text{if } \Pr_s(...) \le 0.05 \\ 0 & \text{otherwise} \end{cases}$$

Finally, δ id defined as $\delta_{\mathcal{I}}(l, C, Q) = \mathrm{MT}(\widehat{\mathcal{I}_c}(l, C), \mathcal{I}_q(l, C, Q))$ and $\delta_{\mathcal{K}}(l, C, Q) = \mathrm{MT}(\widehat{\mathcal{K}_c}(l, C), \mathcal{K}_q(l, C, Q))$.

4 EXPERIMENTAL EVALUATION

We experimentally evaluate our approach on different datasets and show the impact of the parameters on the final results.

Datasets: We perform experiments on two real datasets.

• YAGO is a large knowledge graph based on Wikipedia, Wordnet and Geonames, with 3.3M nodes, 27M edges, 366K node types and 38 edge labels. We downloaded YAGO 2.5[2] and converted node attributes to edges and attribute values to node labels.

• LMDB[3]: LinkedMDB is a knowledge graph for the movie domain, extracted from the Internet Movie Database (IMDB), with 739K nodes, 1.6M edges, and 18 edge types.

Experimental Setup: We implemented our solution in Java 1.8, and ran the experiments on a Intel i5-4210U 1.7 GHz machine with 12GB RAM. All datasets are loaded into Apache Jena triple store. Along our ContextRW described in Section 3.1 for context selection and FindNC for notable characteristics identification on top of ContextRW, we implement RandomWalk, a baseline for context selection based on Personalized PageRank (see Section 3.1) computed through the power iteration method with 10 iterations and $c = 0.8$.

4.1 Evaluating Context selection

We compare the effectiveness of ContextRW with the baseline RandomWalk within different topics. Since no ground truth for finding context nodes given a set of query nodes were available, we generated context nodes via CrowdFlower (https://www.crowdflower.com) for 15 query sets in three domains, namely *politicians*, *actors*, and *movie contributors*. For each domain we manually determined 6 entities belonging to the domain and generated queries of increasing size (up to 6 entities). We asked 34 workers to provide a ranked list of related entities given the query, resulting in 7'650 entities. From such entities we removed those mentioned only once and obtained 36 to 76 entities per query that are mapped into YAGO.

Context size $|C|$. Context size affects the quality of the results, since more context nodes potentially lead to better recall but worse precision. Figure 2a compares RandomWalk and our ContextRW in terms of F_1 score at different $|C|$.In all cases, ContextRW performs up to four times better than the RandomWalk, vindicating the effectiveness of our metapath constrained random walk in finding context nodes, while RandomWalk mostly returns nodes that are close to the query nodes, but semantically irrelevant. Quality does not improve for $|C| > 100$ due to a loss in precision. We also experience a lower variance for ContextRW that exploits metapaths for guiding the search.

Query size $|Q|$. We analyze the performance of the algorithms varying the query size $|Q|$. Figure 2 shows that ContextRW improves in result quality when more query nodes are considered, which means that our method capture semantic relationships between the nodes. On the contrary, RandomWalk is not affected by the size of the query disregarding metapaths.

Figure 3a reports the time to compute the context, showing that RandomWalk is on average up to two orders of magnitude slower than ContextRW, for $|Q| = 5$. Expectedly, ContextRW is faster with larger queries ($<20s$ comprising the database time), since the chance to end up in a query node is larger.

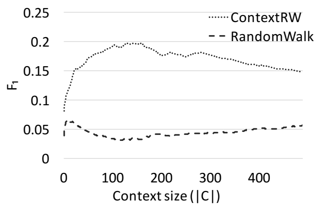
Figure 2c reports the maximum F_1 of ContextRW at increasing $|Q|$, comparing YAGO and LMDB datasets within the *actors* domain. Unsurprisingly, ContextRW performs moderately better than YAGO in LMDB due to the specificity of the dataset; however, YAGO result testifies the generalization ability of ContextRW in larger, more complex datasets.

Number of paths $|M|$. The ContextRW algorithm depends on the number of paths. Figure 2d shows the F_1 score in relation to the context size and the number of paths. The number of paths does not affect the score; however, as shown in Figure 3b the time increases as the length of the metapaths (and also the number, not reported) increases. Therefore, a reasonable choice for the number of metapaths $|M|$ and maximum length is 5.

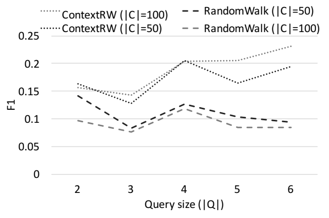
4.2 Distribution Comparison

Test cases. We show preliminary evidence of the effectiveness of FindNC with respect to the RandomWalk baseline for context retrieval with the multinomial test. The test case in Figure 4a shows the instance distribution of the query Q = {*George Clooney*,

[1]In case of large N, an approxiamte Montecarlo sampling is performed.
[2]http://resources.mpi-inf.mpg.de/yago-naga/yago2.5
[3]https://datahub.io/dataset/linkedmdb



(a) **Context size $|C|$**



(b) **Query size $|Q|$**

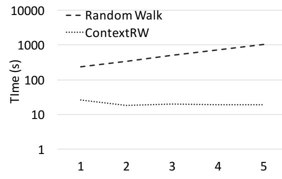
$\|Q\|$		max F_1	$\|C\|$
2	YAGO	0.23	23
	LMDB	0.30	101
3	YAGO	0.2	107
	LMDB	0.25	122
4	YAGO	0.19	130
	LMDB	0.24	124
5	YAGO	0.25	162
	LMDB	0.26	198
6	YAGO	0.22	285
	LMDB	0.25	139

(c) Dataset comparison

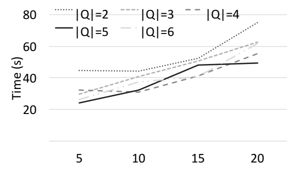
	Number of paths ($\|M\|$)			
$\|C\|$	5	10	15	20
50	0.15	0.16	0.13	0.15
100	0.22	0.21	0.21	0.21
150	0.22	0.23	0.23	0.23
200	0.22	0.22	0.22	0.22

(d) Number of paths $|M|$

Figure 2: Average quality in terms of F_1 varying parameters $|C|$, $|q|$, and $|M|$ in YAGO dataset.

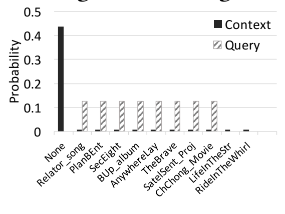


(a) Query size $|Q|$

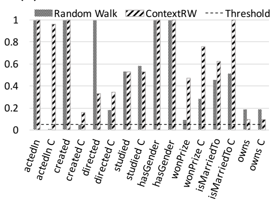


(b) Maximum metapath length

Figure 3: Average time (s) in YAGO dataset.



(a) *Created* label in YAGO



(b) Method comparison

Figure 4: Test cases for the actors domain and 5 query nodes. A "C" in the labels denotes cardinality distributions

Brad Pitt, *Leonardo DiCaprio*, *Scarlett Johansson*, *Johnny Depp*} over the top-100 context nodes for the *created* edge label. The *created* edge label is absent in 43% cases (represented as *None* instance), whereas all the other values are equally likely with 0.66% chances. The query presents a different distribution, with one actor without *created* labels and all the others with a different value. This clear deviation from the context is a notable characteristic by the multinomial test.

In the second test case, not reported due to space limits, we test the query {*Douglas Adams*, *Terry Pratchett*} against the top-30 nodes as context. Our solution identified the edge *influences* as a notable characteristic. This is because the two authors in the query influenced an actor that was influenced by only 3 in total.

Algorithm comparison. Figure 4b compares FindNC with RW-Mult, with query {George Clooney, Brad Pitt, Leonardo DiCaprio, Scarlett Johansson and Johnny Depp}. All items above the threshold, depicted as a dashed line, are considered not interesting ($\delta = 0$). The random walk selects mostly famous people in the movie business; hence, *actedIn* that connects actors with movies, is rare in the context but common in the query.However, this is clearly not correct and our FindNC algorithm marks *actedIn* as uninteresting. Similarly, *hasWonPrize* shows a significant difference between the two algorithms, as winning a prize is common for actors (75%), but not so in the rather mixed random walk context (only 25%). The chart also shows that the significance level of the multinomial test can be used as a parameter to obtain the desired "interestingness" level. Choosing 0.1 would include the *owns* relationship as a notable characteristic, revealing that Brad Pitt is (according to the dataset) the only relevant actor to own a company (Plan B Entertainment).

5 RELATED WORK

Finding notable characteristics reminisces the problem of anomaly detection in attributed graphs [1]; yet, it is fundamentally different, for it does not provide explanations on the nodes returned as anomalies, nor such approaches are query-driven.

Node comparison measures. Node similarities, typically defined in terms of neighbors, is a centerpiece for community detection, classification, and link prediction. Structural equivalence, such as SimRank [4] defines two nodes similar if the neighbors are similar. Random walk approaches, such as Personalized PageRank [2] and HITS [5] can also be used to find structurally similar nodes. However, node similarities cannot readily explain differences among query nodes and other similar nodes.

Seed set expansion. Seed set expansion, or example-based methods, refers to methods that ask the user to provide an initial set of entities or structures and retrieve similar nodes. Seed nodes are used to discover groups of nodes with similar characteristics [6] exploiting the specificity of each node in the seed set. Likewise, seed-based approaches are used to discover dense graph regions [3, 11]. Although these methods provide multiple groups of nodes they cannot properly explain the characteristics and the differences among them; in general, they do not directly compare the query nodes with the others.

Relevant path summarization. Our problem is connected to the discovery of metapaths between nodes [8, 12]. Methods have been proposed to automatically learn metapaths from a given seed set [9]. However, metapaths cannot express the lack of an edge (e.g., Angela Merkel has no children), nor they cannot detect notable characteristics: Being born in the same place is notable, only if similar people are born in different places.

REFERENCES

[1] Leman Akoglu, Hanghang Tong, and Danai Koutra. 2015. Graph based anomaly detection and description: a survey. *DAMI* 29, 3 (2015), 626–688.

[2] Soumen Chakrabarti. 2007. Dynamic personalized pagerank in entity-relation graphs. In *WWW*. 571–580.

[3] Aristides Gionis, Michael Mathioudakis, and Antti Ukkonen. 2015. Bump hunting in the dark: Local discrepancy maximization on graphs. In *ICDE*. 1155–1166.

[4] Glen Jeh and Jennifer Widom. 2002. SimRank: a measure of structural-context similarity. In *KDD*. 538–543.

[5] Jon M Kleinberg. 1999. Authoritative sources in a hyperlinked environment. *JACM* 46, 5 (1999), 604–632.

[6] Isabel M Kloumann and Jon M Kleinberg. 2014. Community membership identification from small seed sets. In *KDD*. 1366–1375.

[7] Sangkeun Lee, Sanghyeb Lee, and Byoung-Hoon Park. 2015. PathMining: A Path-Based User Profiling Algorithm for Heterogeneous Graph-Based Recommender Systems.. In *FLAIRS Conference*. 519–523.

[8] Sangkeun Lee, Sungchan Park, Minsuk Kahng, and Sang-goo Lee. 2012. Pathrank: a novel node ranking measure on a heterogeneous graph for recommender systems. In *CIKM*. 1637–1641.

[9] Changping Meng, Reynold Cheng, Silviu Maniu, Pierre Senellart, and Wangda Zhang. 2015. Discovering meta-paths in large heterogeneous information networks. In *WWW*. 754–764.

[10] Davide Mottin, Matteo Lissandrini, Yannis Velegrakis, and Themis Palpanas. 2016. Exemplar queries: a new way of searching. *VLDB J.* 25, 6 (2016), 741–765.

[11] Natali Ruchansky, Francesco Bonchi, David García-Soriano, Francesco Gullo, and Nicolas Kourtellis. 2015. The Minimum Wiener Connector Problem. In *SIGMOD*. 1587–1602.

[12] Yizhou Sun, Jiawei Han, Xifeng Yan, Philip S Yu, and Tianyi Wu. 2011. Pathsim: Meta path-based top-k similarity search in heterogeneous information networks. *PVLDB* 4, 11 (2011), 992–1003.