Maverick: A System for Discovering Exceptional Facts from Knowledge Graphs

Gensheng Zhang*
gezhang@google.com
Google, Inc.

Chengkai Li
cli@uta.edu
The University of Texas at Arlington

ABSTRACT

This paper presents Maverick, a system for discovering exceptional facts about entities in knowledge graphs. Maverick is built upon a beam-search based algorithmic framework which we proposed in a research paper that will appear in SIGMOD 2018 [7]. In this demonstration proposal, we showcase an end-to-end system that also includes a user-facing portal and a cache server. The portal allows users to search entities in a knowledge graph and explores exceptional facts about the entities of interest. The cache server is built for sharing computation across entities. In Maverick, an exceptional fact about an entity of interest is modeled as a context-subspace pair, in which a subspace is a set of attributes and a context is defined by a graph query pattern of which the entity is a match. The entity is exceptional among the entities in the context, with regard to the subspace. The search spaces of both patterns and subspaces are exponentially large. Maverick conducts beam search on the patterns which uses a match-based pattern construction method to evade the evaluation of invalid patterns. It applies two heuristics to select promising patterns to form the beam in each iteration. Maverick traverses and prunes the subspaces organized as a set enumeration tree by exploiting the upper bound properties of exceptionality scoring functions. Furthermore, in order to avoid repetitive computation across entities, a cache server is introduced to store intermediate results, such as pattern evaluations, exceptionality calculations, and candidate patterns. In the user-facing portal, Maverick presents exceptional facts to users in forms of natural language sentences and illustration charts, for better interpretability of the discovered exceptional facts.

1 INTRODUCTION

Knowledge graphs such as DBpedia, Freebase, Wikidata, and YAGO record properties of and relationships between real-world entities. These data are used in numerous applications, including search, recommendation, and business intelligence. Given an entity in a knowledge graph, we propose a system that discovers *exceptional facts* about the entity. Informally, such exceptional facts separate the entity from many other entities. We have witnessed a lot of such facts in published news articles:

(1) "Denzel Washington followed Sidney Poitier as only the second black to win the Best Actor award." (*abcnews.go.com*)
(2) "This was Brazil's first own goal in World Cup history ..." (*yahoo.com*)
(3) "Hillary Clinton becomes first female presidential nominee." (*chicagotribune.com*)

An exceptional fact consists of three components: an *entity of interest*, a *context*, and a set of *qualifying attributes*. In each exceptional fact, among all entities in the context, the entity of interest is one of the few or even the only one that bears a particular value combination on the qualifying attributes. For example, in the above statement 1, the entity of interest is Denzel Washington, the context is the Academy Award Best Actor winners, and the qualifying attribute is ethnicity.

Discovery of exceptional facts is useful to important applications such as computational journalism [1], recommendation systems, and data cleaning. a) In *fact-finding* [4, 6], journalists are interested in monitoring data and discovering attention-seizing factual statements such as the aforementioned examples. These facts help make news stories substantiated and interesting, and they may even become leads to news stories. b) In *fact-checking* [3], for vetting the statements made by humans, fact-checkers at news organizations such as The Washington Post, CNN, and PolitiFact can compare the statements with automatically-discovered facts. For example, an algorithm may find that Hillary Clinton is the second female presidential nominee, which contradicts with the statement 3 above. [1] c) Exceptional facts can help promote friends, news, products, and search results in various recommendation systems. d) When the discovered facts are inconsistent with known truth or apparent common sense, it reveals incomplete data or data errors. Such insights aid knowledge base cleaning and completion. For example, the above statement 3 may be generated using an incomplete source that misses the nomination of Victoria Woodhull.

To the best of our knowledge, there is no previous system for discovering exceptional facts about entities in knowledge graphs. The two most related areas are *outlier detection* in graphs and *outlying aspect mining*. Duan et al. [2] and Vinh et al. [5] discussed the differences between these two areas. They achieve different goals. Outlier detection searches for all outlying objects among a set of objects. Outlying aspect mining, however, focuses on only one given object and returns the subspaces of attributes in which the object is relatively outlying, regardless of its true degree of outlyingness. In terms of objectives and problem modeling, the exceptional fact discovery problem is closer to outlying aspect mining than outlier detection. However, it focuses on graph data. In contrast, existing outlying aspect mining methods assume a single relational table

*The bulk of the work was done while the author was at UT-Arlington.

[1] The first female presidential nominee was Victoria Woodhull, according to *http://www.snopes.com/victoria-woodhull-hillary-clinton/*.

SIGMOD'18, June 2018, Houston, Texas USA

ACM ISBN 123-4567-24-567/08/06…$15.00
https://doi.org/10.475/123_4

and cannot be effectively applied to knowledge graphs, as explained in [7].

This paper presents Maverick, a system for exceptional fact discovery for entities in knowledge graphs. Maverick is built upon the beam-search based framework we proposed in [7]. In this demonstration proposal, we showcase an end-to-end system that includes a user-facing portal and a cache server, in addition to the algorithmic framework in [7]. The portal allows users to search entities in a knowledge graph and explores exceptional facts about the entities of interest. The cache server is built for sharing computation across entities. In Maverick, an exceptional fact about an entity of interest is modeled as a context-subspace pair, in which the subspace is a set of attributes and the context is defined by a graph query pattern of which the entity is a match, together with other matching entities. The entity is exceptional among the entities in the context, with regard to the subspace. The search spaces of both patterns and subspaces are exponentially large. Maverick conducts beam search on the patterns which uses a match-based pattern construction method to evade the evaluation of invalid patterns. It applies two heuristics to select promising patterns to form the beam in each iteration. Maverick traverses and prunes the subspaces organized as a set enumeration tree by exploiting the upper bound properties of exceptionality scoring functions. Furthermore, in order to avoid repetitive computation across entities, a cache server is introduced to store intermediate results, such as pattern evaluations, exceptionality calculations, and candidate patterns. In the user-facing portal, Maverick presents exceptional facts to users in the forms of natural language sentences and illustration charts, for better interpretability of the discovered facts.

2 CONCEPTS

Knowledge graph, Patterns, Matches A *knowledge graph* $G(V_G, E_G)$ is a set of RDF triples with node set $V_G \subseteq \mathcal{I}$ and edge set $E_G \subseteq V_G \times \mathcal{I} \times V_G$, where $\mathcal{I}$ is the universe of Internationalized Resource Identifiers (IRIs). A *pattern* is a weakly connected graph $P(V_P, E_P)$, where $V_P \subseteq \mathcal{I} \cup \mathcal{V}$, $E_P \subseteq V_P \times \mathcal{I} \times V_P$, and $\mathcal{V}$ is the universe of variables. A subgraph M of G is a *match* to pattern P if M is edge-isomorphic to P and, for each non-variable node v in P, its counterpart in the match has the same identifier.

Context A *context* is a set of entities sharing some common characteristics defined in a pattern and a variable in the pattern. The set of entities are those in the matching patterns that correspond to the variable. An implication of this definition is that the entities in a context have similar characteristics, e.g., "players who play for Brazil".

Attributes Given an entity v, its *attributes* A_v is the union of its incoming and outgoing edge labels: $A_v = \{(l, \leftarrow) \mid \exists(x, l, v) \in E_G\} \cup \{(l, \rightarrow) \mid \exists(v, l, x) \in E_G\}$. Given an attribute a, the *attribute value* $v.a$ of an entity is the set of the other incidents of the related edges. A *subspace* A is a subset of A_v.

Exceptionality function Given a context C of an entity v and a subspace A, an exceptionality scoring function $\chi(v, A, C) \in \mathbb{R}$ measures entity v's degree of exceptionality with regard to A in comparison with other entities in C. Without loss of generality, we assume the range of χ is $[0, 1]$, with larger χ implying greater exceptionality.

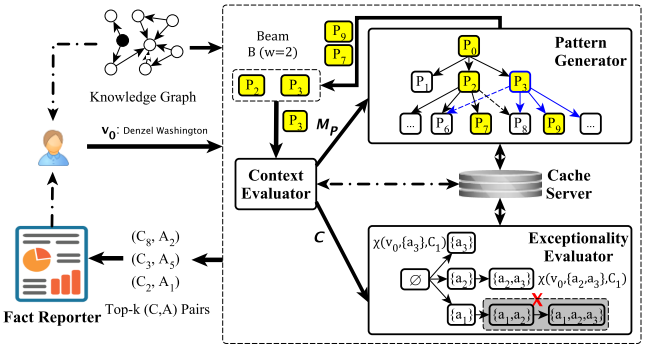


Figure 1: The framework of Maverick.

We will demonstrate in the system the three exceptionality functions as described in [7]: Isolation, Outlierness, and One-of-the-Few.

Exceptional Fact Discovery Problem Given a knowledge graph G, an entity of interest v_0, an integer k, and an exceptionality scoring function χ, the problem of *exceptional fact discovery* is to find the top-k exceptional facts about v_0.

3 SYSTEM OVERVIEW

Figure 1 illustrates the components of Maverick. While the three components *Context Evaluator (CE)*, *Exceptionality Evaluator (EE)*, and *Pattern Generator (PG)* work together in an iterative fashion of beam search to discover exceptional facts, which are highly-scored context-subspace pairs, component *Cache Server (CS)* caches (intermediate) results of PG, CE and EE to improve efficiency by avoiding repetitive computation, and component *Fact Reporter* reports highly-scored context-subspace pairs to users in natural language sentences as well as charts. Intuitively, the process can be viewed as nested loops. The outer loop enumerates contexts, while the inner loop enumerates subspaces for each context. Given the entity of interest v_0, while subspace enumeration in the inner loop enumerates the subsets of A_{v_0}, the outer loop enumerates contexts by patterns. Conceptually, Maverick organizes all the possible contexts as a partial order on patterns, i.e., a Hasse diagram, in which each node is a pattern and each edge represents the subsumption (subgraph-supergraph) relationship between the two patterns. The essence of the outer loop is thus a traversal of the search space of patterns.

At each iteration, Maverick maintains a beam B of a fixed size w. The beam consists of heuristically the best w patterns (e.g., P_2, P_3 in Fig. 1 where $w = 2$). For each pattern P in B, component CE obtains the matches $\mathcal{M}_P$ to the pattern and the corresponding contexts C of v_0. For each context C in C (e.g. C_1 in Fig. 1), component EE finds the top-k scored subspaces according to a given exceptionality scoring function χ. Component PG finds the children of the visited pattern based on its matches. Since there are usually much more children than what the beam size w allows, PG applies a set of heuristics to prune the child patterns. Each child pattern is given a score that measures how promising it is according to the heuristics. The best w patterns among all the children of patterns in B will become the new beam B, which is the input to the next iteration, e.g., $\{P_7, P_9\}$ in Fig. 1. When the algorithm terminates, Maverick returns the k context-subspace pairs with the highest exceptionality scores. Below we explain these components in more detail.

Context Evaluator (CE) This component is responsible for obtaining the matches to a given pattern as well as the corresponding contexts. EE employs a graph query system to take a pattern as the input and return all the matches to the pattern. Although EE in our implementation uses Neo4j as the graph query system, EE is agnostic to the choice of the specific query processing system.

Exceptionality Evaluator (EE) For each context C of the entity of interest v_0, this component finds the k subspaces A with the highest $\chi(v, A, C)$ scores. Section 4.1 briefly discusses how EE uses a set enumeration tree to avoid exhaustively enumerating exponential number of possible subspaces. Specifically, Maverick exploits the upper bound properties of exceptionality scoring functions to guide the traversal of the set enumeration tree.

Pattern Generator (PG) At each iteration of the beam search on patterns, this component finds the children of each visited pattern P in the current beam. A child pattern, if not pruned (see Section 4.3), is given a score that measures how promising it is according to a few heuristics. Among all the children of the patterns in the current beam, the w children with the highest scores are returned to form the new beam, where w is the predefined beam width. The new beam becomes the input to the next iteration.

Cache Server (CS) This component stores and reuses the intermediate results of the aforementioned components, to avoid repetitive computation in discovering facts about different entities. Entities usually share common characteristics, for example, entities of the same type usually have the same set of attributes and share a lot of common patterns. As a result, CS can improve the efficiency of the system by caching the pattern evaluation results in component CE, the exceptionality calculations in component EE, and the search of candidate patterns in component PG.

Fact Reporter (FR) This component presents exceptional facts of the given entity to users in form of natural language sentences as well as charts. In order to help users interpret the exceptionality of each fact, FR collects statistics about the qualifying attribute values as well as the context, such as the attribute value distributions, context size and so on. Furthermore, FR provides summaries of the statistics in natural language sentences to improve the interpretibility of the exceptional facts.

4 ALGORITHMS

To discover the exceptional facts about an entity, we must explore two extremely large search spaces, one of attribute subspaces and the other of patterns. In this section, we briefly explain how we approach the challenges.

4.1 Finding Top-k Subspaces

EE applies a *set enumeration tree* to avoid exhaustively enumerating subspaces. Each node in the tree is a subspace—a subset of v's attributes A_v. The children of a node correspond to various supersets of the node's associated attributes. The gist is to explore the set enumeration tree using heuristic search methods such as best-first search and to prune branches that are guaranteed to not contain highly-scored subspaces. What is particularly challenging is that an exceptionality scoring function χ usually does not have the *downward closure property* with respect to subspace inclusion, i.e., $\chi(v, A, C)$ can be greater than, less than, or equal to $\chi(v, A', C)$ for any $A' \supseteq A$. EE uses upper bounds on the exceptionality scoring function χ to allow for pruning of the set enumeration tree.The set enumeration tree nodes (i.e., subspaces) are visited in the descending order of their upper bounds. If the upper bound score of a node is not greater than the score of the current k-th ranked subspace, the node and all its children are pruned.

4.2 Match-based Construction of Patterns

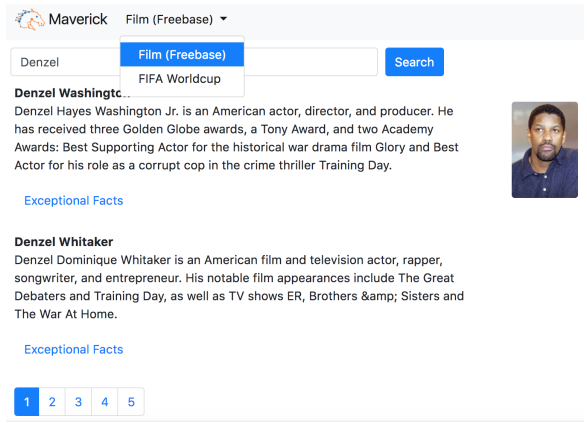
In the search of context, Maverick organizes the candidate patterns as a Hasse diagram, in which each node is a pattern and each edge represents the subgraph-supergraph relationship between the two nodes. To construct the child patterns of a pattern P, a simple approach is to enumerate all possible ways of expanding P by adding one more edge. A major drawback of this approach is it may construct many invalid patterns that do not have any match. Some invalid patterns can be easily recognized by referring to the schema graph of the data. However, chances are most of the schema-abiding patterns are still invalid because they do not have matching instances in the data graph, given the sheer diversity of a knowledge graph. The system will evaluate such patterns in vain to get empty results in order to realize they are invalid.

To avoid evaluating invalid patterns, we propose a *match-based pattern construction method*. Instead of constructing the child patterns by directly expanding pattern P, this method expands the matches of P and constructs the child patterns from the expanded matches. It guarantees to construct only valid patterns and evade the evaluation of invalid patterns. Specifically, given a match M of P, it finds each of its weakly connected supergraphs by adding an edge that exists in the data graph G and is adjacent to a node in M. The supergraphs are then used to generates P's child patterns by optionally replacing the newly added node with a variable.

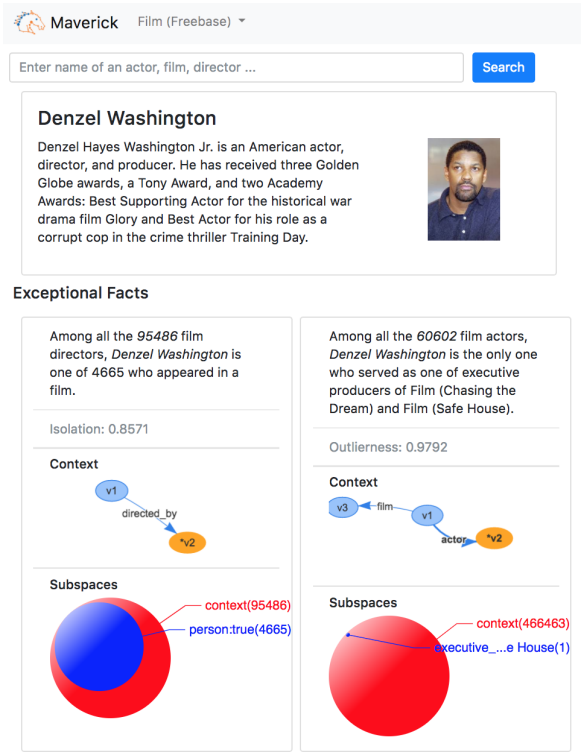
4.3 Candidate Pruning and Candidate Ranking

To ensure efficiency, PG employs two pruning rules to exclude irrelevant patterns and to avoid repeated constructions of patterns from certain type of parent patterns: 1) exclude a pattern if it does not define any context for the entity of interest v_0; 2) expand a pattern only if the new edge has at least one variable. The rational behind the first rule is, for discovering exceptional facts about v_0, a pattern is relevant only if it defines a context for v_0. Enforcing the second rule reduces the number of candidate patterns by eliminating the patterns that define same contexts of v_0.

To further limited the number of candidates, PG applies two scoring heuristics for selecting promising patterns to visit: Optimistic h_{opt} and Convergent h_{conv}. h_{opt} simply uses the exceptionality score upper bound of P, which essentially selects the patterns that define large contexts. The h_{conv} score of P is a weighted sum of the upper bound of P (for any subspace) and the best score of the parent pattern P'. Compared with h_{opt}, h_{conv} is potentially both more efficient and more effective. It can be more efficient since it may favor child patterns that define smaller contexts. Such child patterns usually can be evaluated more efficiently since they have less matches. It can be more effective since it discards child patterns that define contexts where the entity of interest may not be exceptional, based on the highest score of the context-subspace pairs for the parent pattern.



a Entity search page



b Exceptional fact page

Figure 2: The main user interface of Maverick**.**

5 USER INTERFACE AND DEMONSTRATION PLAN

The Maverick system is hosted at *http://idir-server4.uta.edu/maverick*. We will demonstrate Maverick using two knowledge graphs: *Film* and *FIFA World Cup*. *Film*, a subgraph of Freebase, contains two Freebase domains: Film and Award. The knowledge graph *FIFA World Cup* is constructed from *https://www.fifa.com/worldcup/*. Figure 2 shows the main user interface of Maverick when it is applied on Denzel Washington in *Film*. Some functions in the user interface are still under development, e.g., the natural language interpretation of exceptional facts.

When a user visits the Maverick system, it shows a landing page which has a banner and a search box. The landing page is identical to the top part of Figure 2a. The banner allows the user to choose a knowledge graph to explore. The search box allows the user to search entities by names. The search box is also equipped with auto-completion to help users search entities more efficiently. Figure 2a shows a screenshot when a user enters "Denzel" in the search box. The page shows the descriptions of a list of matching entities to help the user choose the entity of interest. Below the description of each entity, it provides a link to the page of exceptional facts of the entity, which will be described below. When there are many entities in the search result, the system provides a pagination bar at the bottom of the page for the user to navigate through the search results.

Figure 2b shows a screenshot of the exceptional fact page of entity Denzel Washington. The exceptional fact page of a given entity displays a list of exceptional facts about the entity as well as the entity's description. We will demonstrate exceptional facts based on 3 different exceptionality functions: Isolation, Outlierness, and One-of-the-Few. In the system, we will show top-5 exceptional facts from each exceptionality function.

Each exceptional fact is shown as a fact card in Figure 2b. There are four elements in a fact card: interpretation, exceptionality score, context graph, and subspace chart. The interpretation is usually a sentence or two describing the fact. The exceptionality score is a real value between 0 and 1 indicating how exceptional the fact is. The context graph shows a graph pattern. The graph pattern along with the orange node together define the context of the entity. The subspace chart is a set of tangent circles. The outermost circle represents the context. Each nested circle represents the entities satisfying the conjunctions of the qualifying attributes represented by outer circles. For example, let the set of qualifying attributes be $\{a_1, a_2, a_3\}$, and the entity of interest be v_0, then there are four circles in the subspace chart. From outermost to innermost, the circles represent entities $V_0 = C$, $V_1 = \{v \in C \mid v.a_1 = v_0.a_1\}$, $V_2 = \{v \in C \mid v.a_1 = v_0.a_1 \wedge v.a_2 = v_0.a_2\}$, and $V_3 = \{v \in C \mid v.a_1 = v_0.a_1 \wedge v.a_2 = v_0.a_2 \wedge v.a_3 = v_0.a_3\}$, respectively, where C is the context. The size of each circle reflexes the cardinality of the corresponding set of entities in logarithmic scale.

REFERENCES

[1] Sarah Cohen, Chengkai Li, Jun Yang, and Cong Yu. 2011. Computational Journalism: A Call to Arms to Database Researchers. In *CIDR*. 148–151.
[2] Lei Duan, Guanting Tang, Jian Pei, James Bailey, Akiko Campbell, and Changjie Tang. 2015. Mining outlying aspects on numeric data. *DMKD* 29, 5 (2015), 1116–1151.
[3] Naeemul Hassan, Bill Adair, James T. Hamilton, Chengkai Li, Mark Tremayne, Jun Yang, and Cong Yu. 2015. The Quest to Automate Fact-Checking. In *Proceedings of the 2015 Computation+Journalism Symposium*.
[4] Naeemul Hassan, Afroza Sultana, You Wu, Gensheng Zhang, Chengkai Li, Jun Yang, and Cong Yu. 2014. Data In, Fact Out: Automated Monitoring of Facts by FactWatcher. *PVLDB, demonstration description* 7, 13 (2014), 1557–1560.
[5] Nguyen Xuan Vinh, Jeffrey Chan, Simone Romano, James Bailey, Christopher Leckie, Kotagiri Ramamohanarao, and Jian Pei. 2016. Discovering outlying aspects in large datasets. *DMKD* 30, 6 (2016), 1520–1555.
[6] Gensheng Zhang, Xiao Jiang, Ping Luo, Min Wang, and Chengkai Li. 2014. Discovering General Prominent Streaks in Sequence Data. *TKDD* 8, 2 (June 2014), 9:1–9:37.
[7] Gensheng Zhang, Damian Jimenez, and Chengkai Li. 2018. Discovering Exceptional Facts from Knowledge Graphs.. In *Proceedings of the 2018 ACM SIGMOD International Conference on Management of Data (SIGMOD)*. To appear.