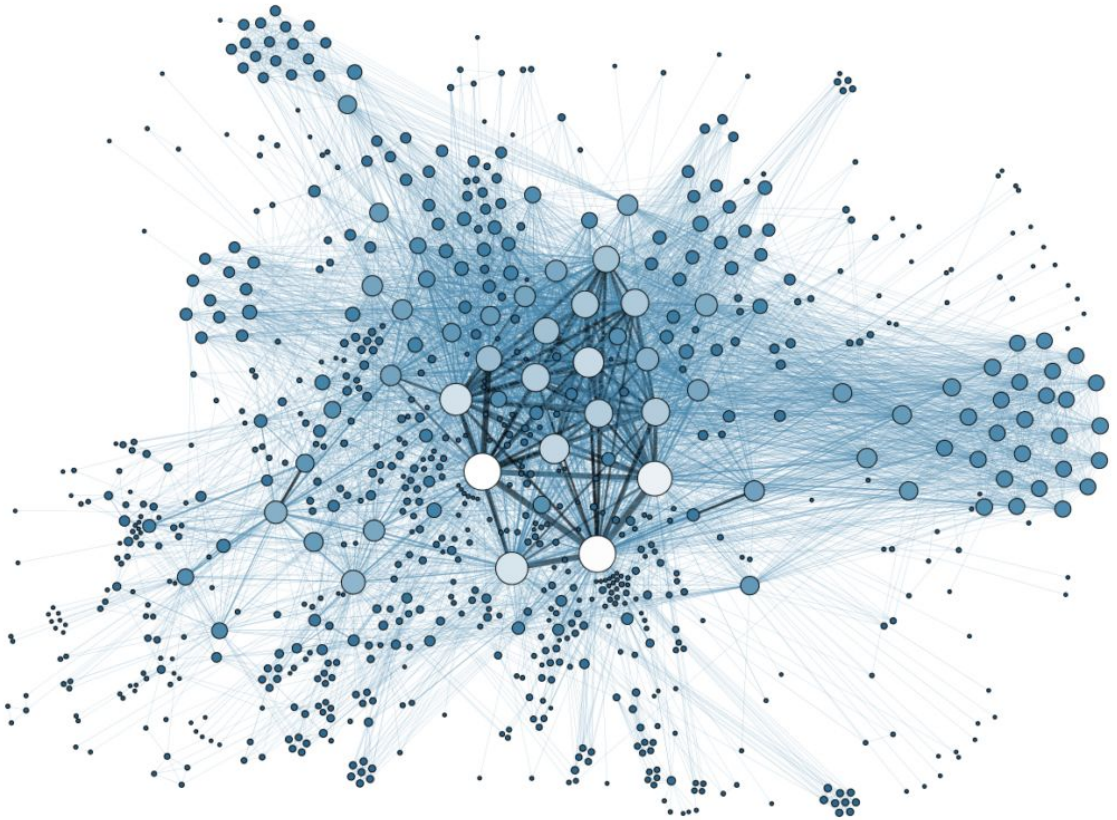
Maverick: Discovering Exceptional Facts from Knowledge Graphs

12/03/19

Paper published in Proc. ACM SIGMOD International Conference on Management of Data, 2018.

Presented by: Juan Carrillo
Candidate for MASc. in Computer Software
Department of Electrical & Computer Engineering
University of Waterloo



UNIVERSITY OF WATERLOO

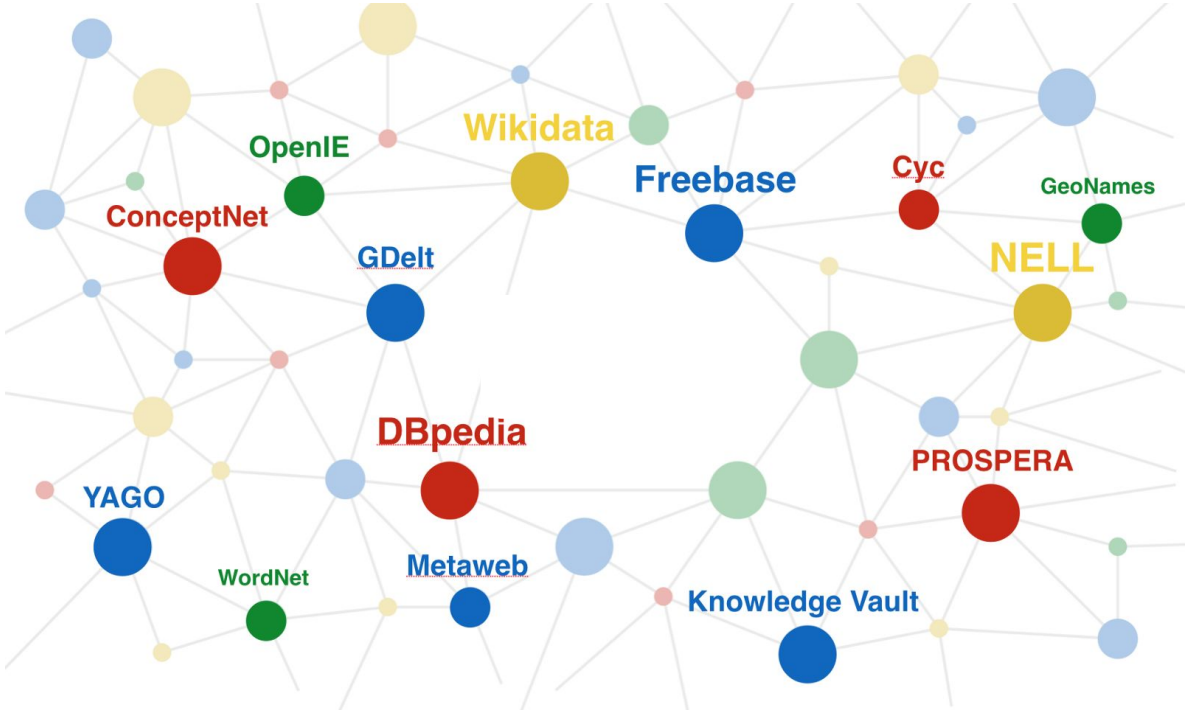
Agenda

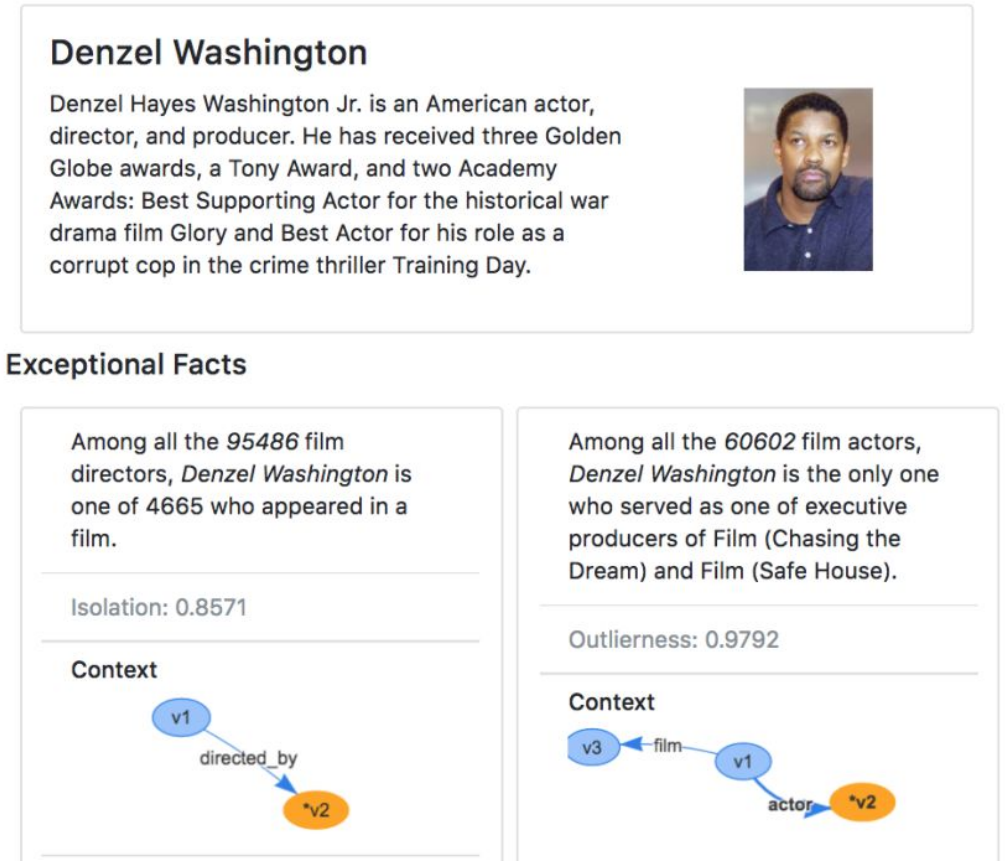
1. Introduction
2. Maverick core features
3. Experiments
4. Conclusions
5. Discussion

1 Introduction

1. Introduction

From knowledge graphs to exceptional facts

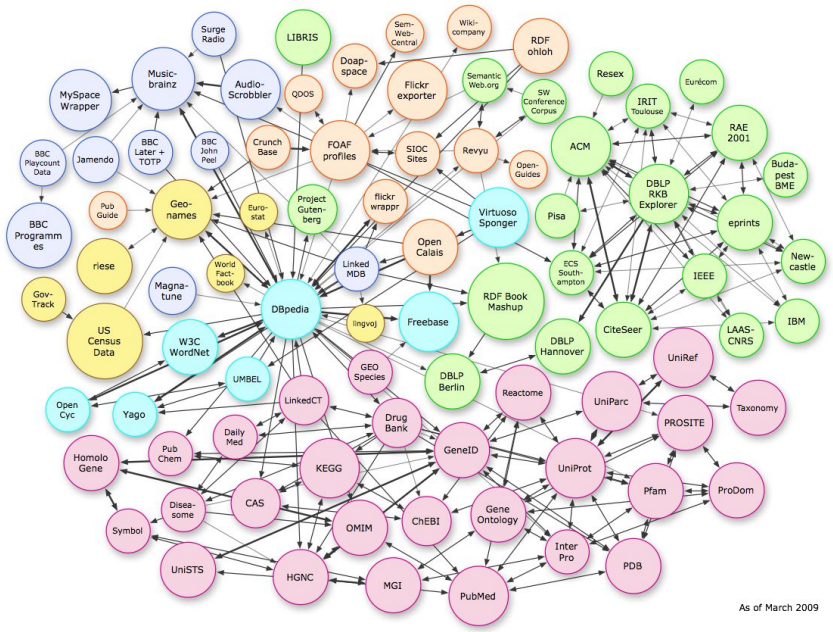




1. Introduction

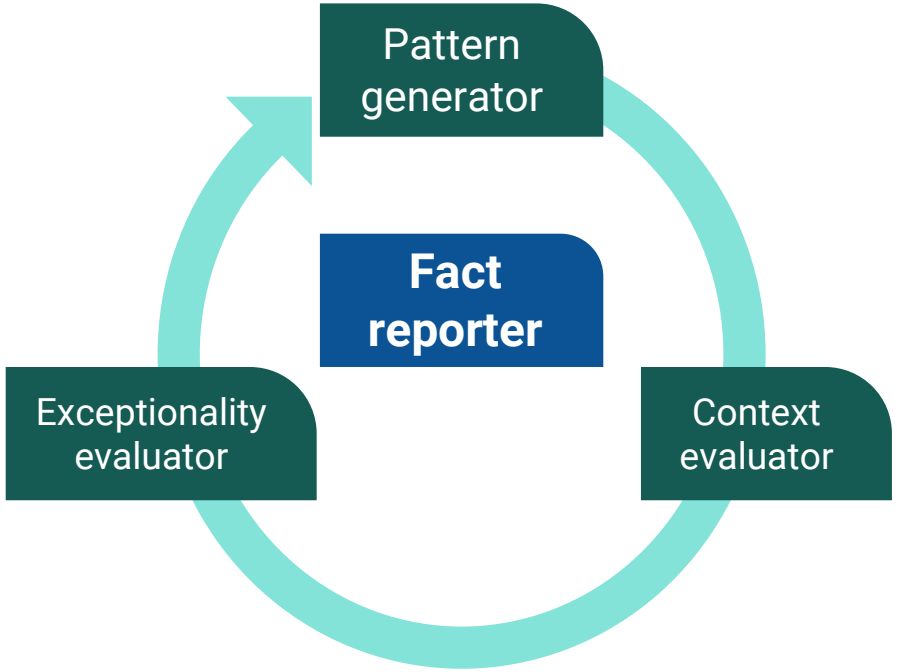
The problem, and the Maverick approach

Knowledge graphs (Linked Data)



Manually designed queries

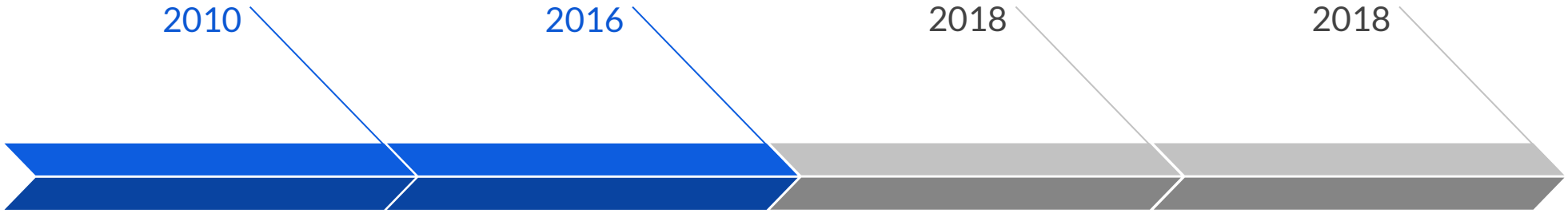
Maverick approach



Automated detection of exceptional facts

1. Introduction

Related background



Outlier detection

J Gao - 2010

On community outliers and their efficient detection in information networks

Outlying aspect mining

F Angiulli - 2016

Outlying property detection with numerical attributes

Maverick: Discovering Exceptional Facts from Knowledge Graphs

SIGMOD'18

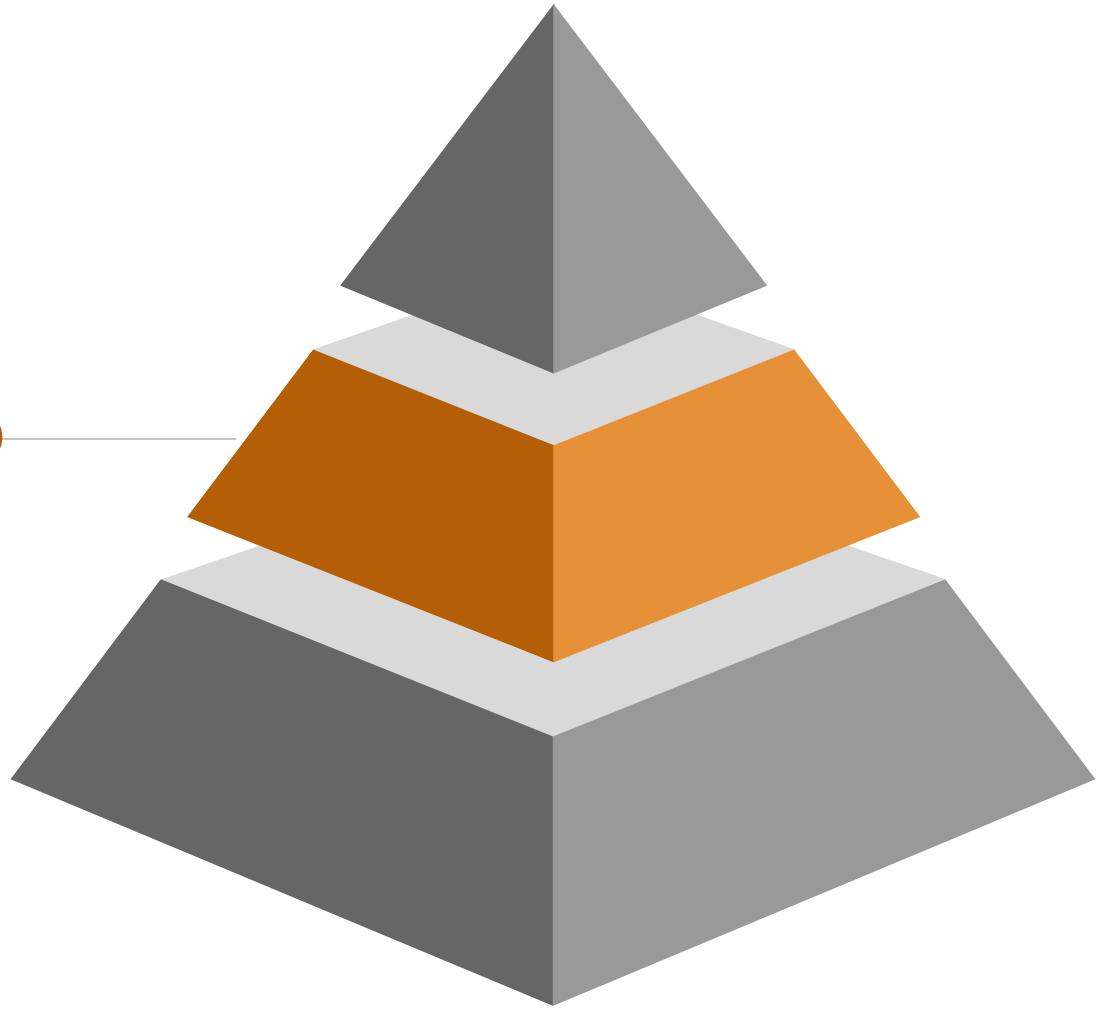
Comprehensive description and math basis

VLDB'18

High-level description and demo

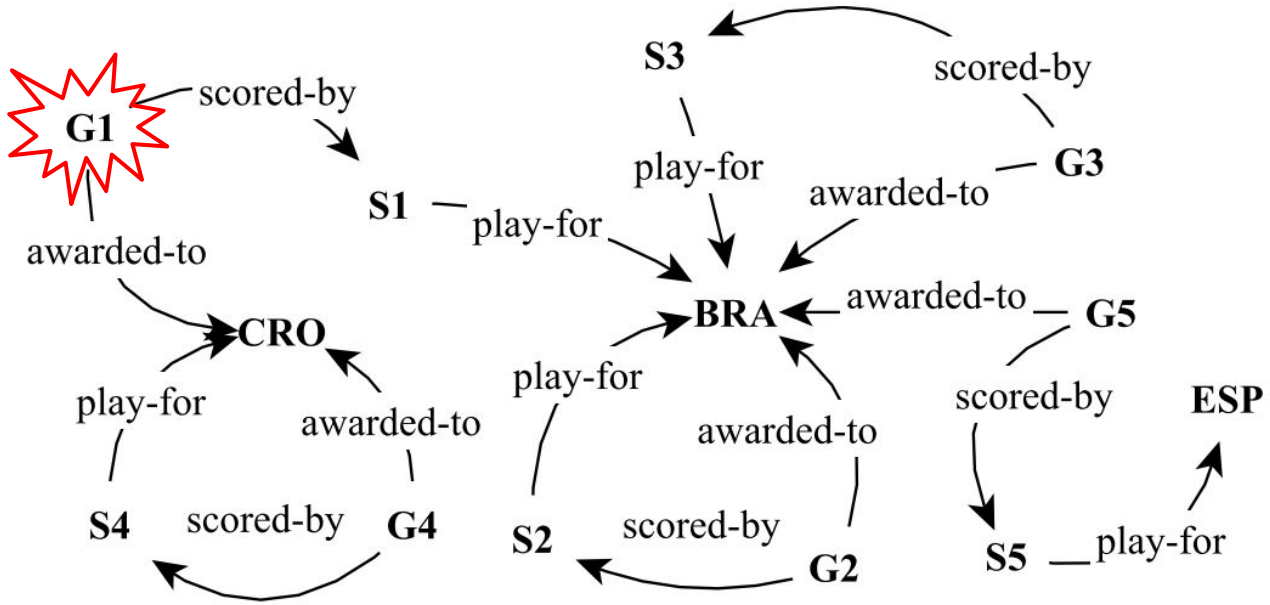
Maverick core features

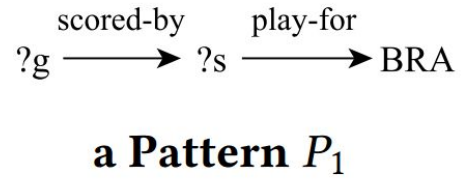
2



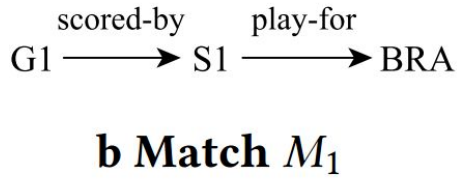
2. Maverick core features

Entity, context, pattern

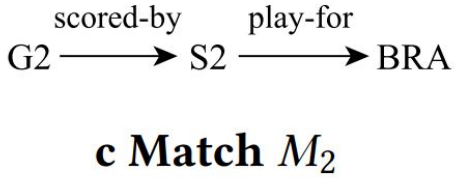




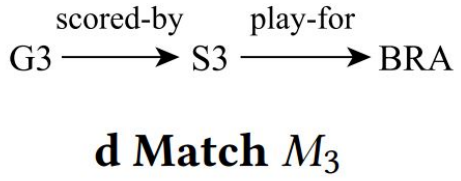
a Pattern P_1



b Match M_1



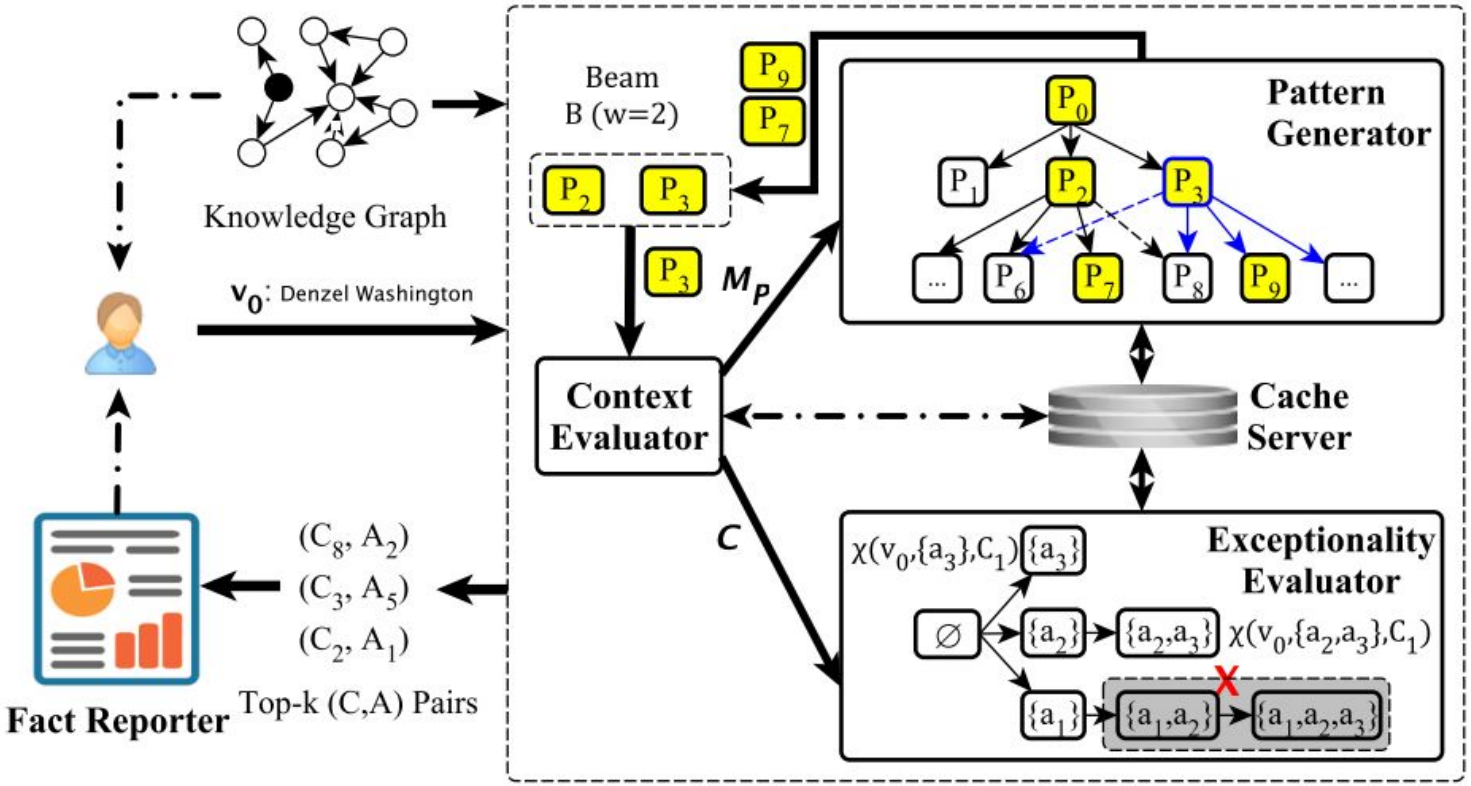
c Match M_2



d Match M_3

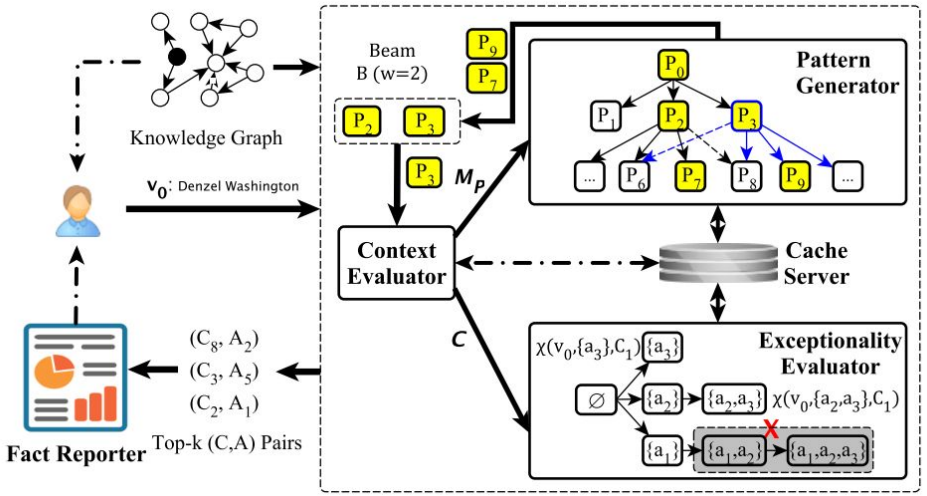
2. Maverick core features

The overall framework



2. Maverick core features

Main Algorithm



FACT-DISCOVER (G, v_0, χ, k, w)

Input: G : the knowledge graph; $v_0 \in V_G$: the entity of interest; χ : the exceptionality scoring function; k : the size of output; w : the beam width

Output: H : k most exceptional context-subspace pairs

$P_0 \leftarrow (V_{P_0} = \{x_0\}, E_{P_0} = \emptyset)$; // Initial state. x_0 is a variable.

$B \leftarrow \{P_0\}$; // Beam.

$i \leftarrow 1$; // Iteration number.

while $B \neq \emptyset$ *and* $i \leq$ MAX_ITERATION **do**

  $i \leftarrow i + 1$; $B_{tmp} \leftarrow \emptyset$;

  foreach $P \in B$ **do**

    // Obtain contexts of v_0 and matches to P. (Section 3.1)

    $C^P_{v_0}, \mathcal{M}_P \leftarrow$ CONTEXT-EVALUATOR(P, v_0, G);

    foreach $C \in C^P_{v_0}$ **do**

      // Exceptionality Evaluation. (Section 4)

      $\mathcal{A} \leftarrow$ EXCEPTIONALITY-EVALUATOR(v_0, C, k, χ);

      foreach $A \in \mathcal{A}$ **do** $H \leftarrow H \cup \{(C, A)\}$;

    // Find $\mathcal{Y}$ – the children of P. (Section 5)

    $\mathcal{Y} \leftarrow$ PATTERN-GENERATOR$(v_0, P, \mathcal{M}_P, w, G)$;

    $B_{tmp} \leftarrow B_{tmp} \cup \mathcal{Y}$;

  $B \leftarrow$ top-w of B_{tmp} based on heuristics h ; // Section 5.4

return top-k pairs in H based on exceptionality scores;

Description of components

Context Evaluator

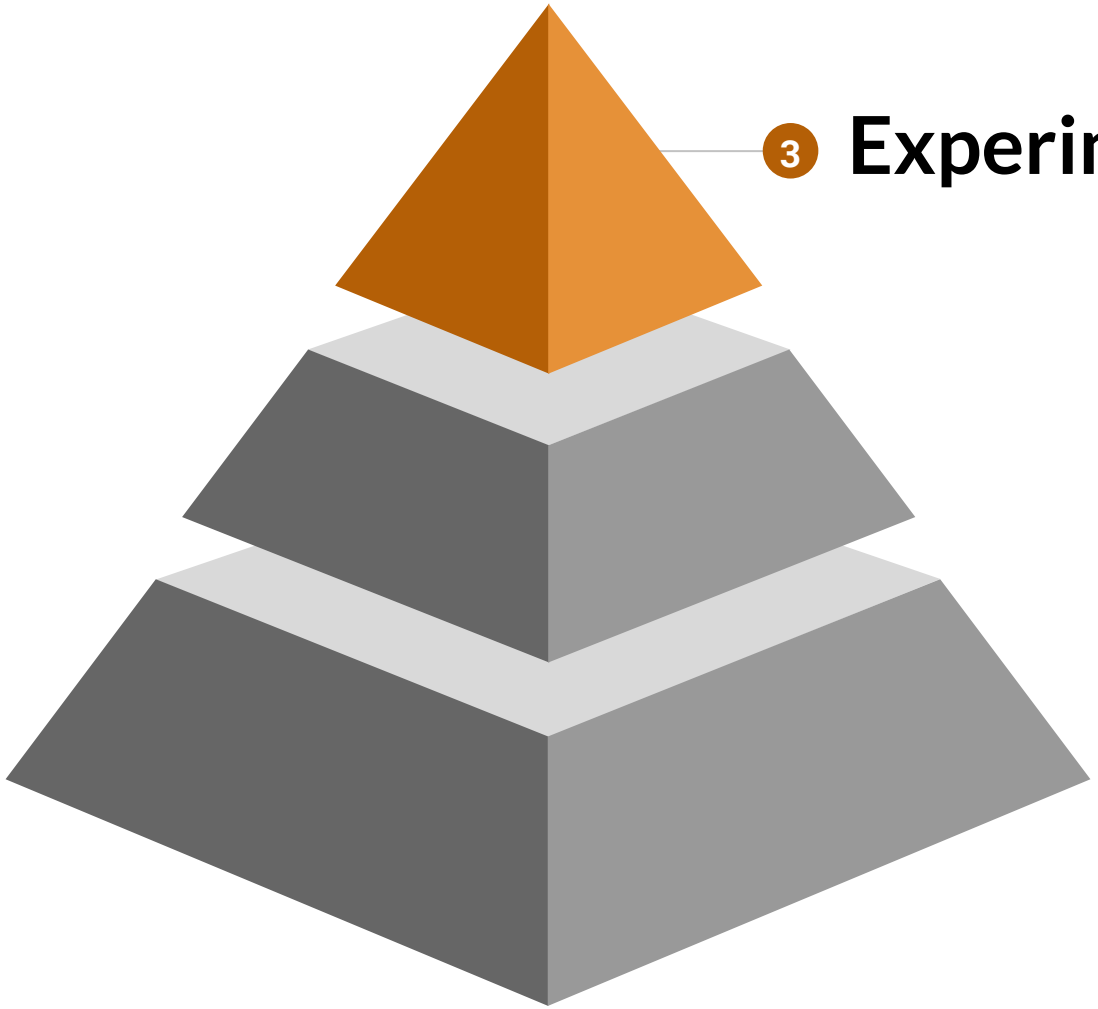
- Uses a graph query system (Neo4j)
- Takes a pattern as input and returns the matches
- Agnostic to query processing system

Exceptionality Evaluator

- Takes the entity of interest and its contexts
- Looks for the k subspaces with highest scores
- Implements scoring functions

Pattern generator

- Uses beam search to look for promising patterns
- Implements domain specific heuristics
- Beam width can be tuned to requirements


3
Experiments

Experimental setup

Single node: 16-core, 32GB RAM

Methods compared

- Beam-Rdm
- Beam-Opt
- Beam-Conv
- Breadth-First

Datasets

WCGoals

49.078 nodes, 158.114 edges, 13 different edge labels, and 11 entity types.

OscarWinners

42.148 nodes, 63.187 edges, 24 distinct edge labels, and 13 entity types.

3. Experiments

Efficiency

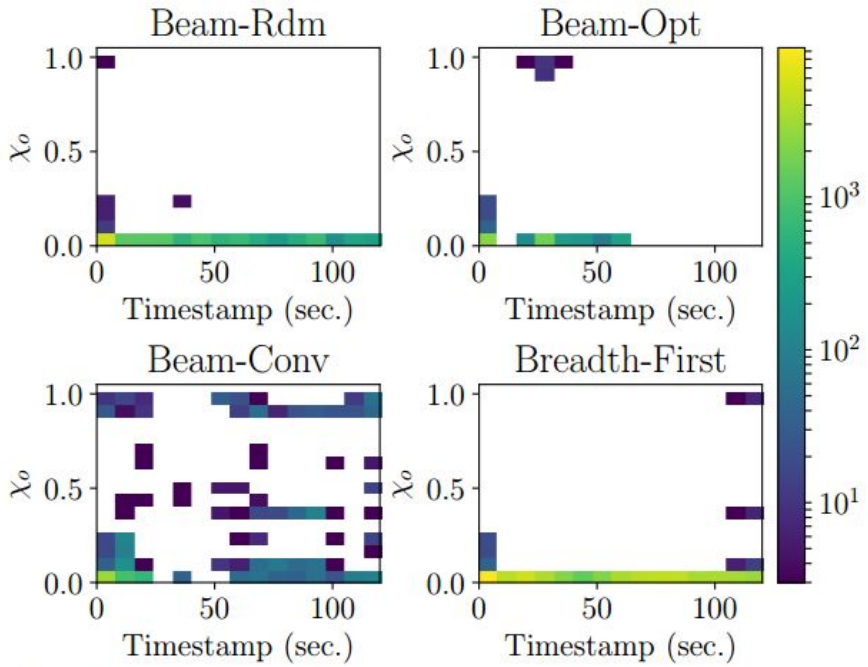
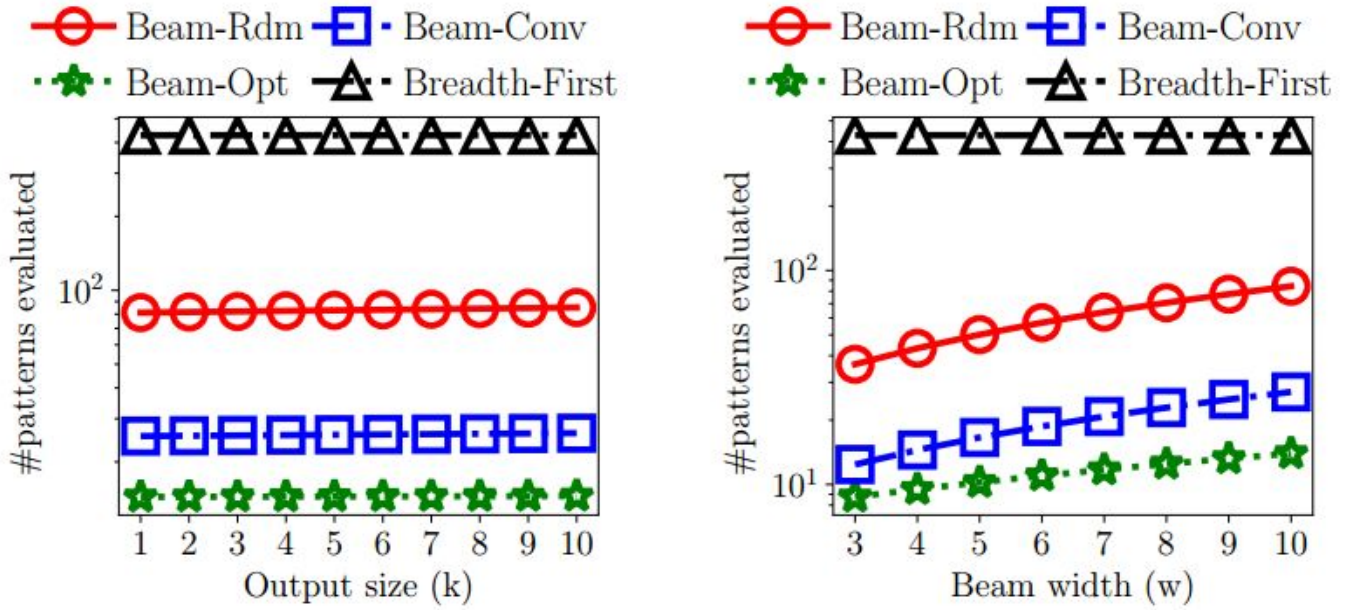


Figure 7: The heat map of exceptionality scores (χ_o) and timestamps of all the discovered context-subpsace pairs during 2-minute runs for 10 entities of interest (v_0) in WCGoals ($k = 10$, $w = 10$).

3. Experiments

Efficiency



a Varying k, fixing $w = 10$. **b Varying w, fixing $k = 10$.**

Figure 8: Effect of k and w on the number of evaluated patterns.

3. Experiments

Effectiveness

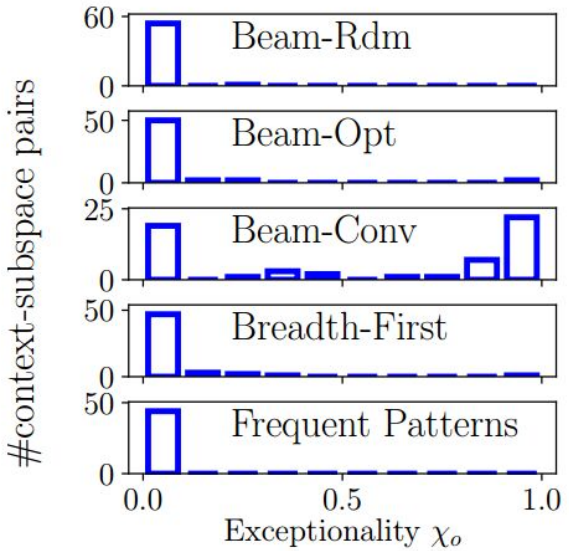


Figure 13: Score distributions of top-10 context-subspace pairs for 10 entities, 10 2-minute runs per entity.

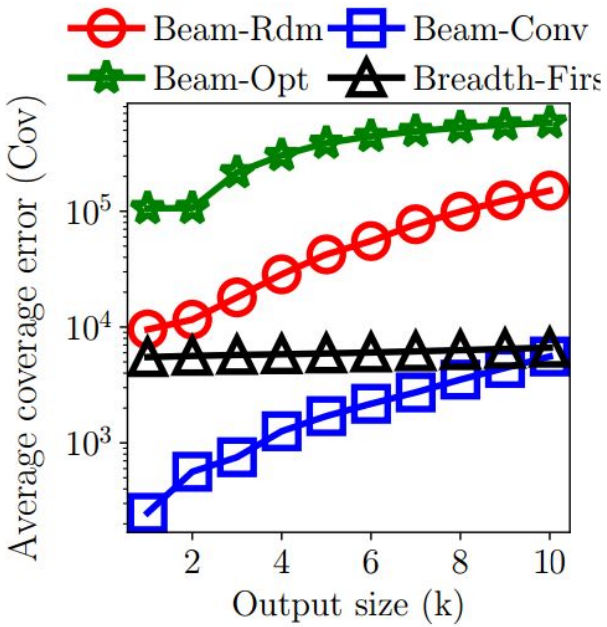


Figure 14: Average coverage error on 10 entities. Beam width 10.

Conclusions 4

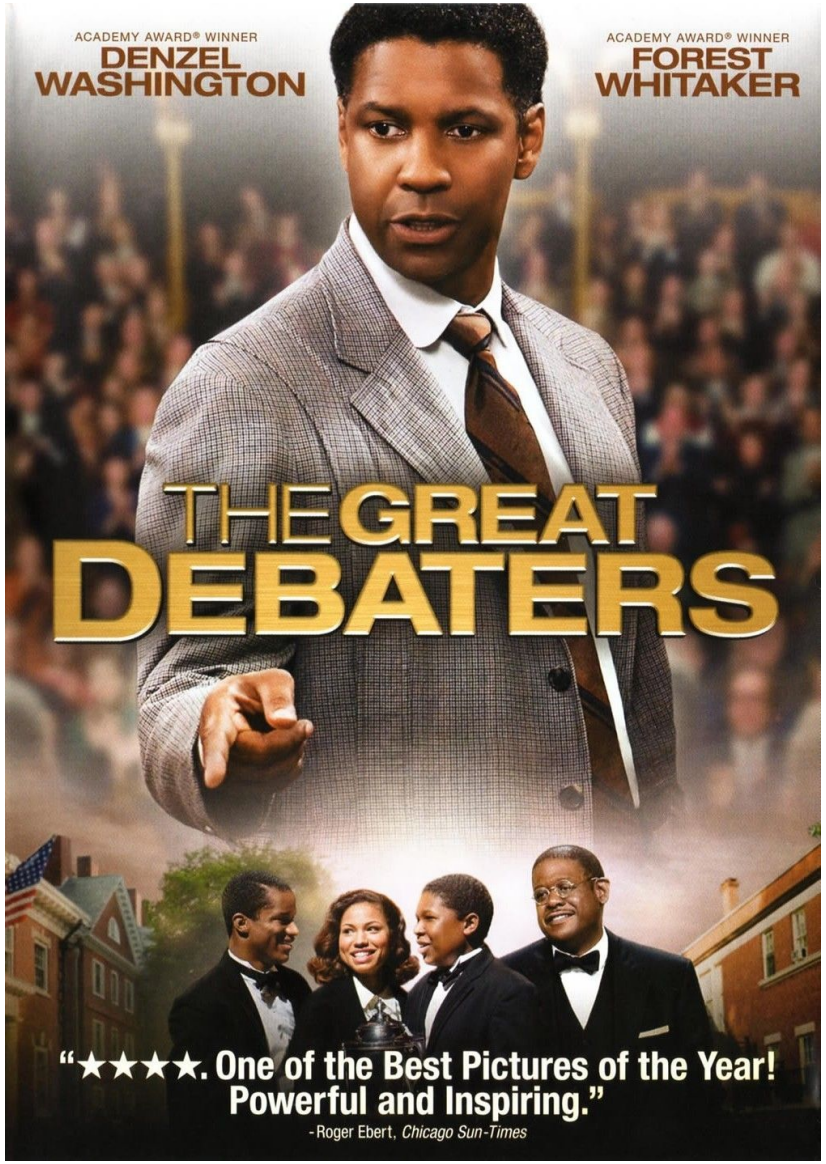
4. Conclusions

Takeaways and paper contributions

- ✓ The authors model an exceptional fact as a context-pattern pair on a knowledge graph
- ✓ Exponential complexity of search is handled using beam search
- ✓ The framework is adaptable to domain specific requirements

Thanks for your attention

Discussion 5



5. Discussion

Research

1. What other heuristics could be proposed in addition to the two presented in the paper? Design requirements for a third heuristic?
2. How Maverick would perform over a completely different dataset? Different proportions among nodes, edges, edge labels, and entity types.
3. What if we add attributes to the nodes and edges? Constraints
4. How to adapt Maverick to work over multiple/linked knowledge graphs?

Industry

5. What is an example of an application over Google knowledge graph?