NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS

SCHOOL OF SCIENCE
DEPARTMENT OF INFORMATICS AND TELECOMMUNICATION

BSc THESIS

Finding Exceptional Facts in Knowledge Graphs

Georgios E. Chalepoudis

Supervisor : **Dimitrios Gunopulos,** Professor

ATHENS

April 2021

ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Εύρεση Εξαιρετικών Γεγονότων σε Γράφους Γνώσης

Γεώργιος Ε. Χαλεπούδης

Επιβλέπων: **Δημήτριος Γουνόπουλος,** Καθηγητής

ΑΘΗΝΑ

ΑΠΡΙΛΙΟΣ 2021

BSc THESIS

Finding Exceptional Facts in Knowledge Graphs

Georgios E. Chalepoudis
S.N.: 1115201200192

SUPERVISOR: **Dimitrios Gunopulos,** Professor

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Εύρεση Εξαιρετικών Γεγονότων σε Γράφους Γνώσης

Γεώργιος Ε. Χαλεπούδης
Α.Μ.: 1115201200192

ΕΠΙΒΛΕΠΟΝΤΕΣ: **Δημήτριος Γουνόπουλος,** Καθηγητής

ABSTRACT

The wide use of the internet and its ability to introduce and amplify Fake News into the information stream demands to also make ways to detect these news, yet the immense amount of data makes it impossible to fact check every single thing and ways to figure out which facts take priority are needed. The study aims to explore the use of Knowledge Graphs in finding Exceptional Facts or in other words facts worth taking the time to fact check. In order to find these Exceptional Facts we make use of the algorithm outlined in the 2018 paper: Maverick Discovering Exceptional Facts from Knowledge Graphs.

The algorithm given an entity of interest will try to find exceptional facts about it. For the purposes of the study an interesting fact about an entity is when given a context that the entity is a part of it is in the minority of the entities for whom the fact applies. A scoring function is used to give a numerical value to the exceptionality . The algorithm works by constructing simple facts that are true for our entity of interest then finds contexts for which the entity of interest is exceptional given those facts, then it constructs more complex facts and repeats iteratively, until it has enough exceptional facts or it can’t find any more.

Results suggest that Knowledge Graphs can be used to find exceptional facts but are prone to showing a bias for facts that actually have no interest when interpreted to physical language and a need for better constructed Knowledge Graphs to avoid such pitfalls.

SUBJECT AREA: Data Analysis

KEYWORDS: Knowledge Graph, Fake News, Exceptional Facts

ΠΕΡΙΛΗΨΗ

Η ευρύς χρήσει του internet και της ιδιότητας του να εισάγει και να ενισχύει Ψευδείς Ειδήσεις στην ροή πληροφοριών απαιτεί την δημιουργία τρόπων για τον εντοπισμό τέτοιον ειδήσεων, όμως το τεράστιο μέγεθος δεδομένων καθιστά αδύνατο να κάνουμε έλεγχο σε κάθε ένα γεγονός και τρόποι για να βρούμε πια γεγονότα έχουν προτεραιότητα είναι αναγκαίοι. Η μελέτη στοχεύει να διερεύνηση την χρήσει γράφων γνώσης για την εύρεση εξαιρετικών γεγονότων ή με άλλα λόγια γεγονότα που τους αξίζει να ελεγχθούν. Για να βρούμε αυτά τα εξαιρετικά νέα κάνουμε χρήσει του αλγορίθμου που δόθηκε στην μελέτη του 2018 με όνομα 'Maverick Discovering Exceptional Facts from Knowledge Graphs'.

Ο αλγόριθμος, με εισαγωγή μιας οντότητας ενδιαφέροντος θα προσπαθήσει να βρει εξαιρετικά νέα για αυτή. Για τους σκοπούς της μελέτης ένα εξαιρετικό γεγονός για μια οντότητα είναι ότι αν έχουμε ένα σύνολο οντοτήτων στο οποίο η οντότητα μας ανήκει αν το γεγονός είναι αληθινό για την μειονότητα των οντοτήτων στο σύνολο. Μια συνάρτηση αποτίμησης χρησιμοποιείτε για να δώσουμε αριθμητική τιμή στην εξαιρετικότητα. Ο αλγόριθμος φτιάχνει απλά γεγονότα που είναι αληθινά για την οντότητα ενδιαφέροντος και βρίσκει σύνολα για τα οποία η οντότητα είναι εξαιρετική σε σχέση με τα δεδομένα, στη συνέχεια φτιάχνει πιο πολύπλοκα δεδομένα και συνεχίζει επαναληπτικά, μέχρι να βρει αρκετά ξεχωριστά δεδομένα ή δεν μπορεί να βρει άλλα.

Τα αποτελέσματα δείχνουν ότι Γράφοι Γνώσεις μπορούν να χρησιμοποιηθούν για την εύρεση εξαιρετικών γεγονότων αλλά έχουν την συνήθεια να δείχνουν εύνοια σε γεγονότα που δεν είναι ενδιαφέροντα όταν μεταφερθούν σε φυσική γλώσσα και χρειάζονται καλύτερα δημιουργημένοι Γράφοι Γνώσης για να αποφευχθούν τέτοια προβλήματα.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Ανάλυση Δεδομένων

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: Γράφοι Γνώσης, Ψευδείς Ειδήσεις, Εξαιρετικά Γεγονότα

CONTENTS

LIST OF FIGURES

LIST OF IMAGES

LIST OF TABLES

PREFACE

“A lie can make the round of the world before the truth can put it's pants on” is a quote attributed to Winston Churchil regarding the ability of false facts or fake news as they are called now, to spread fast before they can be verified and corrected. That was the reality when the press was the main medium of spreading news, on the information age the problem has increased in proportion.
The ease of access to the internet has created a never seen before amount of sources of information with global reach. The data are simply too big to go through and verify every single piece of information that is introduced to the endless stream that is the internet, effectively making it a perfect mechanism for the spread of misinformation or ‘fake news’ as is a now popular term.
Because of this before we can employ strategies for verification we have to filter through the data and pick which information should be verified.

1. INTRODUCTION

News aren't equal and we should remember that unusual or infrequent events in effect exceptional facts are more likely to be broadcasted in the news and more likely to spread fast through the internet. As such If we are to put effort into verifying news we should be interested in verifying those that we could describe as exceptional facts, for they are the ones that will be circulated around the most and in the most incredible speeds that the internet allows. For this purpose I believe we can make use of Knowledge Graphs and techniques to find exceptional facts ranked by a numerical value to decide If a piece of news is worthy of having resources used to verify it. First there is a need to explain what an exceptional fact is. Take for example the following two pieces of news: "A man has bitten a dog" and "A dog has bitten a man", the first is far more eye catching than the second, it is exceptional. From these two examples we can see that a fact consists of three parts, an *entity of interest*, a *context* that the entity is a part of and a *set of attributes* that apply to the entity of interest. In order for the fact to be exceptional the value combination of the entity of interest on the attributes must be rare among the value combinations of the other entities in the context. So what we are looking to do is given an entity of interest in a Knowledge Graph to find the contexts and attribute subsets for which the entity of interest is exceptional. For this I have run experiments on a C implementation of the Maverick Framework as outlined in the Maverick: Finding Exceptional facts from knowledge graphs paper, published 2018.

2. DEFINITIONS

2.1 Knowledge Graphs

A knowledge graph G(VG,EG), is a directed graph made up by a set of nodes VG that represent entities and a set edges EG. Every element in EG is in the form of (entity1,attribute,entity2), with entity1 and entity2, belonging to the set of nodes VG, while an attribute is the label of the edge.
As an example consider the following simple graph

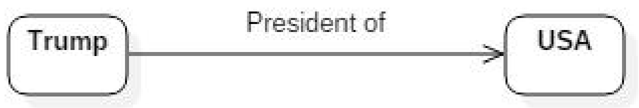

Image 1: Simple Knowledge Graph 1

In this example the set of nodes VG, has two entities, Trump and USA. VG = {Trump,USA}
While the set of edges EG, one element that being,(Trump,President of, USA), Trump and USA being the entities and ‘President of’ being the attribute or the label of this edge.

2.2 Patterns

A pattern P(V_P,E_P), is a weakly directed graph, where it’s set of nodes V_P can be entities or variables. We should also now define $X_{P,}$ as the as the variables that occur in a pattern P.

As an example consider the following knowledge graph and pattern graph.

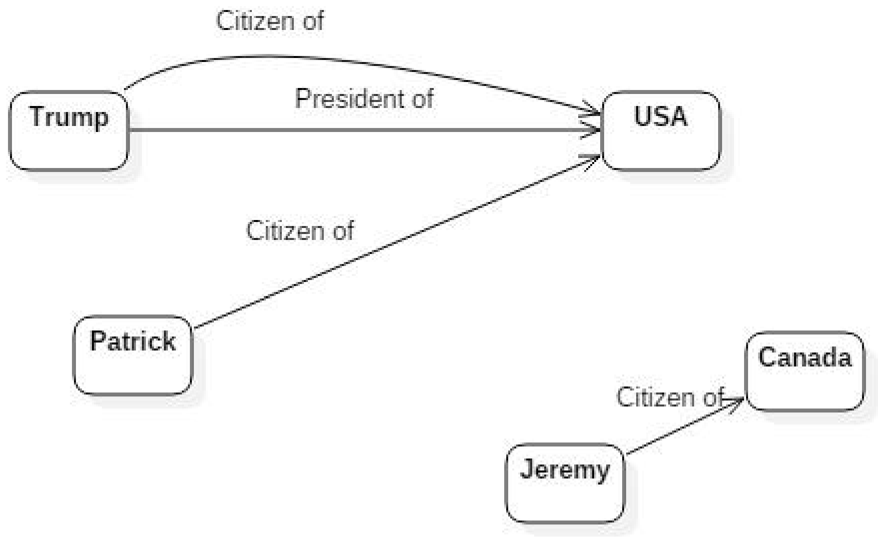

Image 2: Simple Knowledge Graph 2

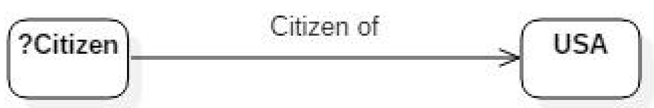

Image 3: Pattern

From the knowledge graph we can see that V_G = {Trump,Parick,USA,Jeremy,Canada}.

Also from the knowledge graph we can distinguish the different types of variables.
Those being citizens and countries.
In the pattern graph we can see that the set of nodes for the pattern are V_P = {Citizen,USA} while the set of variables occurring in P, X_P = {Citizen}.

2.3 Matches

A match M(VM,EM) to apatterm P, is a subgraph of G so that if you replace all variables occurring in P with fitting values from the set of nodes of G, you will have a subgraph of G. To make an example, in the definition of Pattern, we distinguished two different variables, now let us list them along with their potential values, values being the entities fitting said variables.
Citizens = {Trump,Patrick,Jeremy}, Countries = {USA, Canada}

Now let us consider the Graph and Pattern from the previous example the Pattern only has one variable Citizen, which can take the values {Trump,Patrick,Jeremy}, by replacing the node citizen with the nodes Trump or Patrick on the Pattern graph we get a subgraph of G.

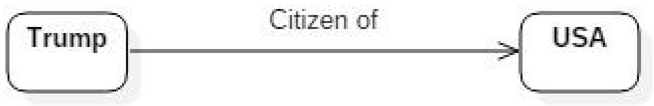

Image 4: Correct Match 1

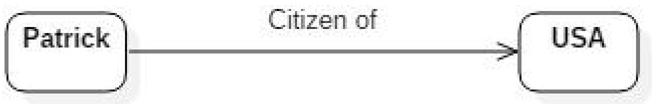

Image 5: Correct Match 2

Thus those two subgraphs constitute a Match for P. While should we replace the node Citizen with Jeremy we would have

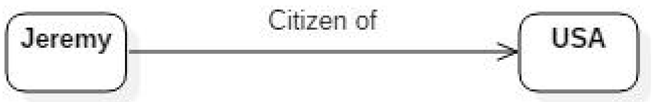

Image 6: Incorrect Match

Which isn't a subgraph of G, thus it isn't a Match for P.

2.4 Range

We will define as Range of a variable x given a pattern P, R_x^P as the set that includes all the entities that are possible values for the variable x while being a match to the knowledge graph G.

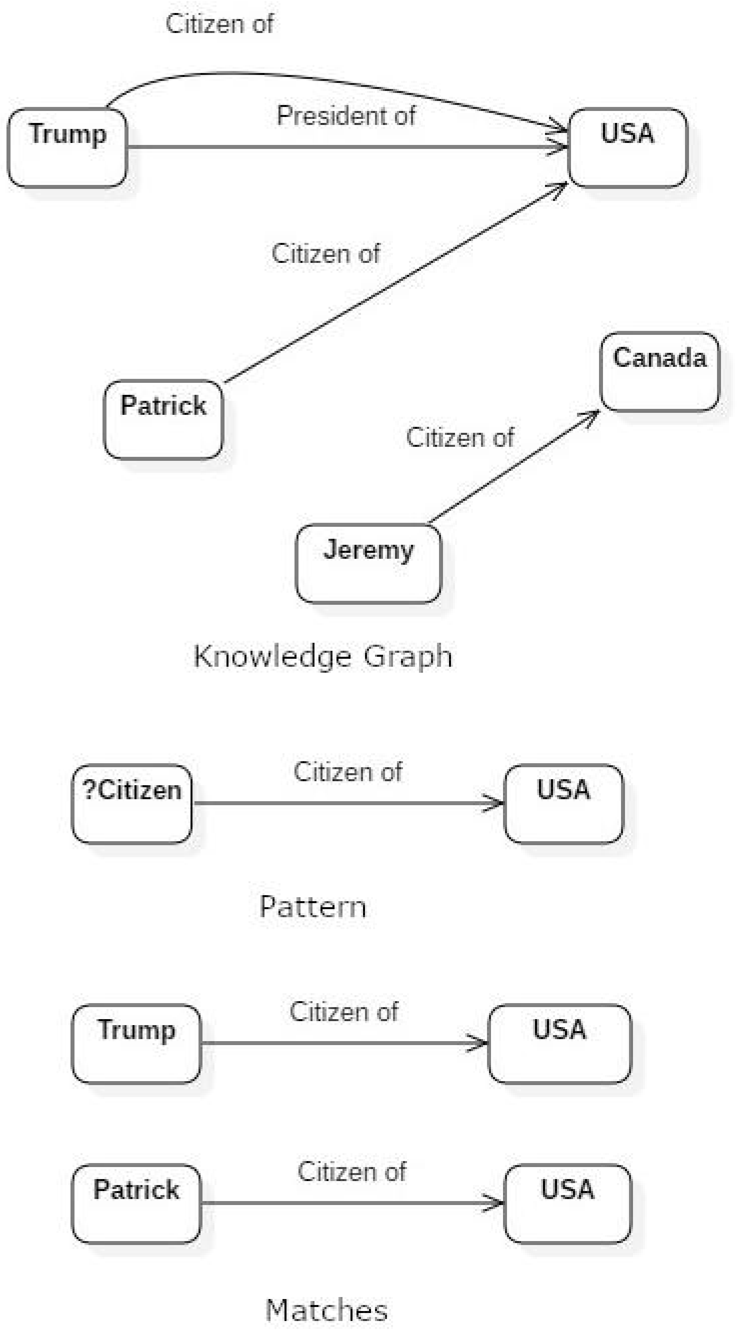

Image 7: KG, Pattern, Matches

Image 7 shows a Knowledge graph, a Pattern and it's Matches, we will refer to the variable of citizens as Ci, thus the range of Ci given the pattern P and knowledge graph G we have established above, R^{P}_{Ci}= {Trump, Patrick}.

2.5 Context

Given an entity u, a pattern P and a variable x belonging in XP, such that u belongs to R^{P}_{x}, the context of u as defined by P and x is $C^{P,x}_{u}$ and $C^{P,x}_{u}$= R^{P}_{x}.
For example using the knowledge graph and pattern we defined as an example in patterns given the entity Trump and variable Citizen, which we will refer to as T and Ci respectively The context is $C^{P,Ci}_{T}$= R^{P}_{Ci}= {Trump, Patrick}.

2.6 Entity Attributes

Given an entity u, the attributes of u, A_u is the set of pairs {(edge lebel,direction) }, direction showing if the edge is outgoing or incoming.
For example the entity Trump has A_T = {(citizen of,->),(president of,->)} both being outgoing.

2.7 Subspace

A subspace A of an entity u, is a subset of the set of attributes A_u.

2.8 Exceptionality Scoring Function

We will call *x*(u,A,C) the exceptionality scoring function which measures the degree of exceptionality of an entity with regard to a subspace A and context C. The larger the value the greater the exceptionality. Different exceptionality scoring functions can be used, but for the purposes of testing a "One out of few" scoring function has been used, that works as follows. Let v be an entity within the contexts C and v.A the value of v on the attributes then for every S = v.A, we define the probability of s occuring as P_S = the number of entities h in C so that h.A = S / the number of entities in the context.
The scoring function x(u,A,C) = The number of entities v in the contexts so that
$P_{v.A} > P_{u.A}$ / the number of entities in the context C.

2.9 Upper Function

An upper function calculates the upper bounds of the scoring function x(u,A,C), upper(u,A,C) ≥ x(u,A,C) and given a subspace B that is a superset of subspace A upper(u,A,C) ≥ upper(u,B,C).The upper function should be made with the Exceptionality Scoring Function in mind. In the case of the "One out of few" the upper used for tests it works as follows. upper(u,A,C) = (The number of entities in C except entity of interest u with a probability $P_{v.A}$ > 1/C)/ the number of entities in the context C.

3. IMPLEMENTATION

3.1 Inputs

KG: A knowledge graph

k: Max number of Context-Subspace pairs to be printed as output

i: Max number of iterations

w: Max number of patterns for every iteration.

3.2 General Outline

The process of finding exceptional facts or Context-Subspace pairs is an iterative process with four main distinct parts, Find Matches, Find Contexts, Exceptionality Evaluation, Pattern Generation. Given a pattern it's matches are found and then the contexts that include the entity of interest, for every context we find the best ranked context-subspace pairs and finally we expand the pattern. Due to the time complexity issues that would arise from running an iteration for every child pattern, expanding it and repeating, children patterns that are expected to have likely to find low ranking context-subspace pairs are pruned and a maximum number of patterns is set to be explored on each iteration. The starting pattern is a graph with only one node that is a variable that has the entity of interest as a possible value.

3.3 Find Matches

Given the knowledge and a pattern the algorithm finds all matches. The algorithm is iterative, in the first step it will generate as many possible matches as possibly by giving one variable node on the pattern a possible value for it and in the following steps continue generating more matches by giving another of the variable nodes on the possible matches a possible value until all matches have no variable nodes.

At the end of it's iteration an elimination of possible matches that already can't be a match for the graph takes place.

3.4 Find Contexts

Given the knowledge graph, a pattern, the matches to the graph of the same pattern and the entity of interest as input, the algorithm finds all contexts that include the entity of interest that the pattern can produce. The number of variable nodes in the pattern are the algorithm's number of iterations and in every step it will create a context by filling it with the unique entities that can be found matches of the pattern when the variable node was replaced. If a context doesn't include the entity of interest it is deleted.

It should be noted that a context in the implementation includes the pattern that created it, along with which variable node it maps to as those information are critical for appraising the results of the implementation.

3.5 Exceptionality Evaluator

For every context found the subspaces with which the context gets the top k scores are found. To do so an enumeration tree is used and the children with the highest upper function scores are visited first, when k subspaces have been found children and their children with lower upper function score than the lowest scored subspace are pruned.

3.6 Pattern Generator

The pattern generator expands into children by adding one edge from the knowledge graph that doesn't exist in any of the pattern's matches and has a node adjacent to one of them. Because going through every pattern is costly a heuristic is used to find the w most promising patterns. While there are different approaches for tests an optimistic approach is used by which the most promising patterns are the ones that create the largest sized contexts as a larger context can produce a higher score.

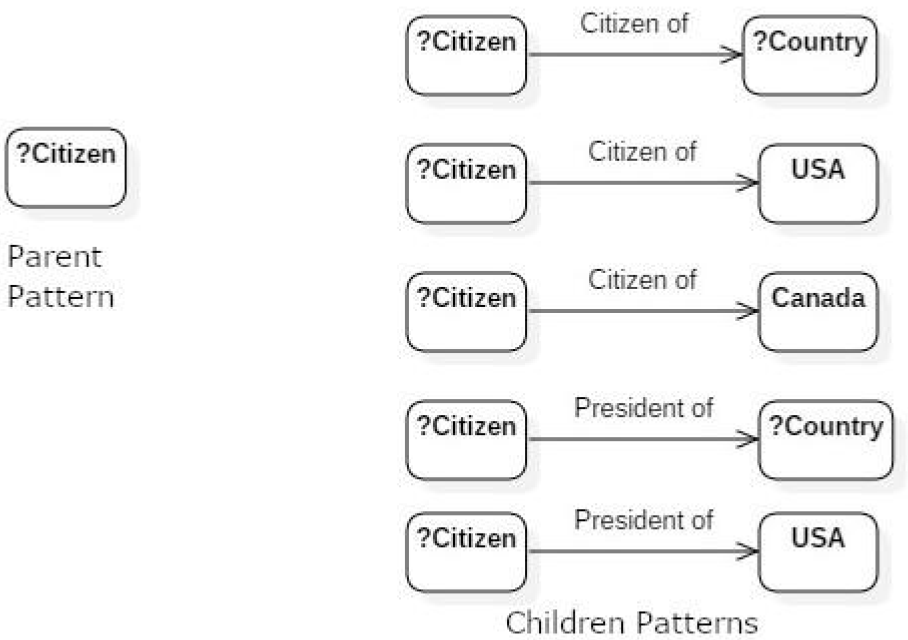

Image 8: Pattern Generator Example

In the pattern Pattern Generator Example the parent pattern composed of a single node, the variable Citizen has five children patterns (based on the knowledge graph of image 2). The first two patterns define contexts of size 2, while the rest define contexts of size 1. If w = 2, the heuristic would pick the two first children patterns and the rest would be deleted.

4. TESTS - RESULTS - OBSERVATIONS

4.1 Inputs used

The dataset used for the tests was created and based on information of a collection of articles. Specifically, the sources of the articles and the targets of the articles were added as nodes and made up two distinct variables. The event text of the articles was used to form the edges between articles.

Size of output k = 5

Beam Width w = 3

Max Number of Iterations = 3

Knowledge graph of 1000 edges.

4.2 Qualitative Results

Given how the scoring function operates the larger the size of the context the better the potential score, thus larger contexts are more likely to find themselves rising to the top. This fact along with only two distinct variables has as a result of contexts that are unsuitable for the purpose of interpreting them as exceptional facts.

To give an example consider the top result for entity of interest ‘Japan’ with a good score of 0.89.

Variable ?1 and pattern ?2 - Express intent to meet or negotiate -> ?1 define the context that includes 19 entities

{Russia, Japan, Warren Christopher, Ernesto Zedillo, Other Authorities / Officials (Chile), Turkey, Oscar Luigi Scalfaro,International North Atlantic Treaty Organization, Greece , Cyprus, Italy, Foreign Affairs (Russia), László Kovács, Head of Government (United States), Mircea Snegur, Willy Claes, Cambodia, Ung Huot}

The subspace: {(Express intent to provide economic aid,->), (Accuse,<-),(Provide economic aid,->)}

With the projection of Japan being

{Japan - Express intent to provide economic aid -> Haiti,
Japan - Express intent to provide economic aid -> Iran,
Japan - Provide economic aid -> Iran,
Japan - Provide economic aid -> Iran, North Korea - Accuse -> Japan}

Trying to interpret this results it would be
“Among all the entities that another entity has expressed intent to meet or negotiate, Japan is one of the few and possibly the only one that has been accused by North Korea and has expressed intent to provide economic aid to Haiti and Iran and has provided economic aid to Haiti and Iran”

It’s not very interesting, especially because the first part “Among all the entities that another entity has expressed intent to meet or negotiate” defines no interesting group to any human for which exceptionality of one of its members would have a meaning, yet it’s score is high, because the context size is large. To add to that, the context includes entities we can recognize as Countries, individuals, ‘Officials’, an Organisation. Such a context can create high ranking context-subspace pairs that are in truth meaningless.

4.3 Quantitative Results

To evaluate the execution time the times for finding matches, finding contexts, evaluating contexts and generating patterns (combines children generator and heuristic), were taken separately (bar graphs for the execution times of three different entities of interest can be found below). The total time execution time with the stated inputs for different entities could range from 2 minutes to 40 minutes.

Consistently the Total Time spent for finding contexts and evaluating them was under 1 second. The Total Time spent for finding matches never went further than 10 seconds, leaving the context generator as the most costly and unpredictable part of the algorithm.

It should be noted that finding matches and by extension, generating children and running heuristics on them is more costly on patterns with more variables as nodes instead of entities and the optimistic approach of the heuristic used means those patterns are going to be more plentiful.

As a consequence larger Knowledge Graphs are bound to also have increasingly more time spent on the context generator and heuristic as the larger the Knowledge Graph the patterns will likely have a lot more children that will need to be generated and will run a lot more heuristics to find the top w.

To overcome this problem implementing parallelism for the pattern generator becomes important. The time spent for heuristics on one child is bound to be similar to the time spent for the heuristics on the other children of the same parent pattern, as such finding and running the heuristic of children in parallel is likely to be strongly scalable.

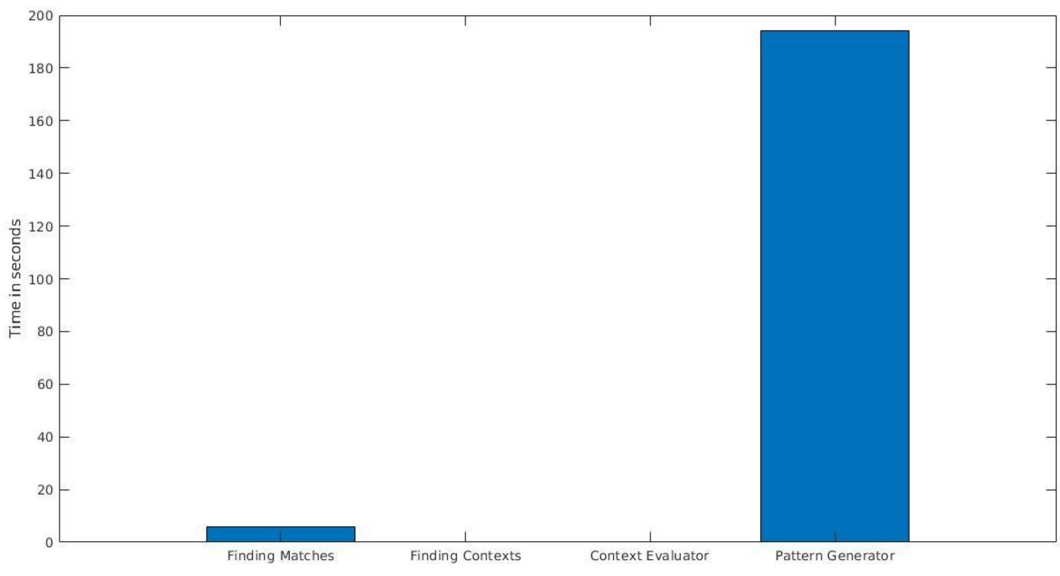

Figure 1: Execution time for Entity of Interest Greece

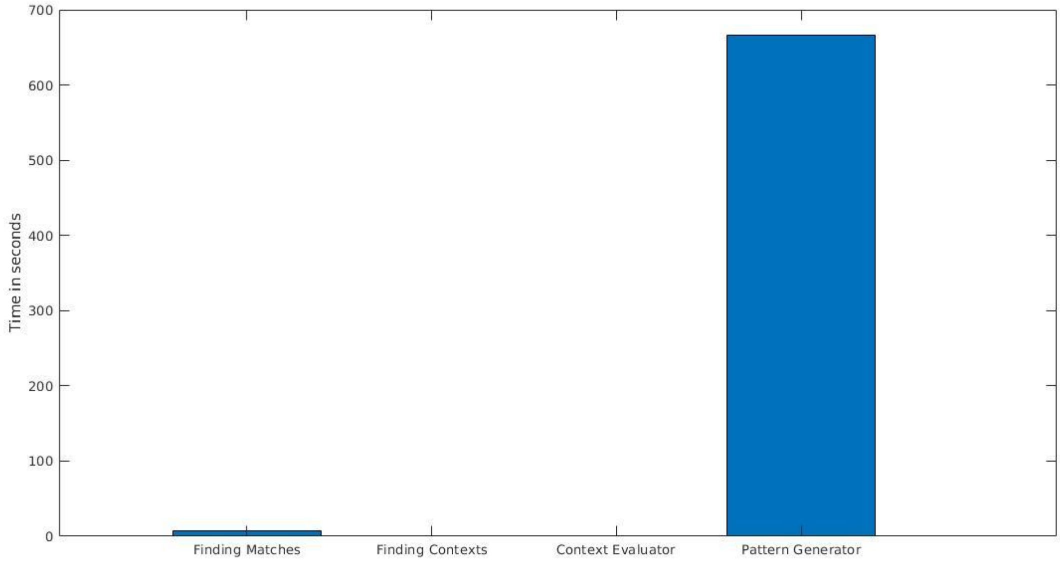

Figure 2: Execution time for Entity of Interest United Nations

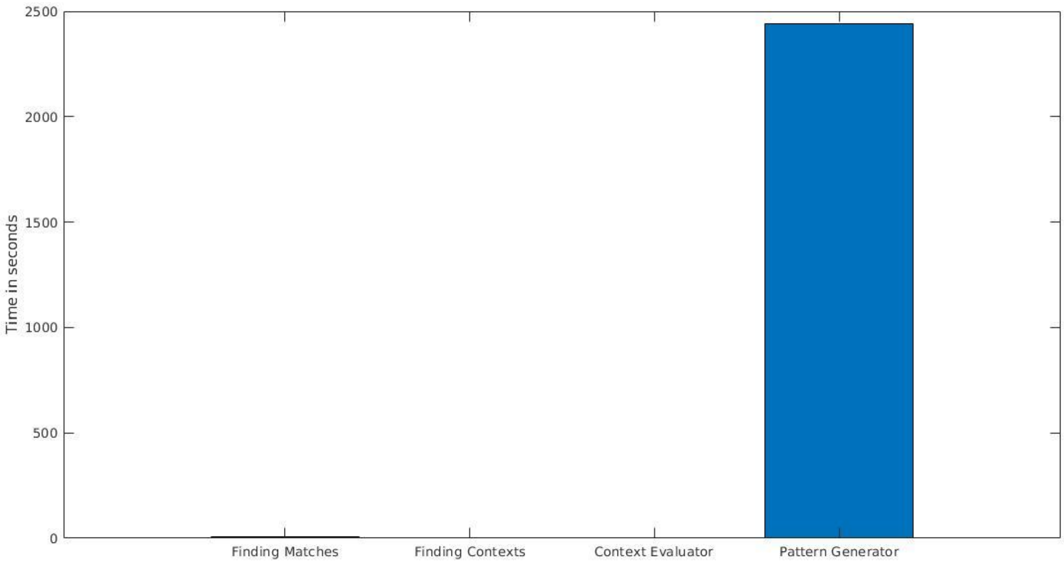

Figure 3: Execution time for Entity of Interest Japan

4.4 Conclusion

For Knowledge Graphs to be used in order to find exceptional facts to use as a filter for facts we really wish to verify there is a clear need for the Knowledge Graphs to be well constructed so that they themselves put restrictions on contexts, to avoid the occurrence of bad contexts rising to the top as high scoring context-subspace pairs and resulting in useless results.

5. ARTICLES DATASET

The Knowledge Graph is created from a tsv file that stores knowledge about articles published 1995 and are political in nature and total size 151311 entries.

The different knowledge about them include date, source name, source sector, source country, event text (what the article expresses, it's values include ('Praise or endorsement', 'Accuse of crime, corruption'), intensity by numerical value, target name, target sector, target country, Publisher, City, District, Province, Country, Latitude and Longitude.

Not all of the attributes have a value for every article, District for example is often blank.

For the construction of the knowledge graph the Source Name, Event Text and Target name of an article were taken, source name and target name used as nodes while the values of the event text as the edges of the graph, as such every row in the Dataset creates a new edge in the graph while adding none to two new nodes to it.

6. PUBLIC KNOWLEDGE GRAPHS

6.1 DBPedia

DBPedia is a knowledge graph created by extracting data from Wikipedia, creating an entity for every article on Wikipedia. Currently it has 4,828,418 entities. The Knowledge Graph is continuously enhanced by the contributions of the DBPedia community through the use of the DBpedia Mappings Wiki.

6.2 Yago

Like DBPedia, Yago is a knowledge graph that creates an entity for every article on Wikipedia but it also extracts data from WordNet and GeoName. It contains over 50 million entities. In Yago entities are arranged into classes for example, John Lennon belongs to the class people. Classes are arranged in a taxonomy, that is to say, classes can have subclasses, for example cities, a subclass of populated cities which is a subclass of geographical locations. In addition Yago defines which relations hold between which entities can hold between entities, an example would be the relation birthplace is between Person and place entities, this definitions and the taxonomy constitute what is called an ontology.

Data in Yago are subject to three logical constraints in order to keep the data clean, those three being Disjointness, Functionality, Domain and range.

Disjointness: Place, person, and creative works are disjoint classes.

Functionality: several relations (such as birthPlace) can have at most one object.

Domain and range: for every relation, we define which class the subject and the object belong to.

7. RELATED WORK

7.1 One graph to rule them all?

On the 2017 paper, they looked into different Public Knowledge Graphs to see which graph was better for which task. The Knowledge Graphs being DBPedia, Yago, Wikidata, OpenCynca and Nell

It shows that for person data Wikidata is the best KG, containing twice as many instances as DBPedia or Yago. For organizations like companies Yago was shown to be preferable. While DBPedia contained more places (villages,countries) than the other Knowledge graphs. It should be noted that despite that despite DBPedia having more places than Wikidata, Wikidata has more detailed information about them.

7.2 Content Based Fake News Detection Using Knowledge Graph

The 2018 paper makes use of Knowledge Graphs to test If incomplete Knowledge Graphs can be used for fake news detection, for this three Knowledge Graphs are built, one KG including only fake news that are manual extracted from confirmed fake news articles and two truth KG. One built by extracting data from articles confirmed to be true and the other by extracting data from DBPedia with taking care to remove repeated triples.

Using the three knowledge graphs three TransE models are generated and used for detecting fake news both on their own and in combinations of the Fake News model with a true one.

The results showed that using imprecise and incomplete Knowledge Graphs can be effective for Fake News detection and if Knowledge Graphs aren't available at hand but articles are they can be effective for constructing one for the purposes of Fake News detection.

TABLE OF TERMINOLOGY

Ξενόγλωσσος όρος	**Ελληνικός Όρος**
Knowledge Graph	Γράφος Γνώσης
Context	Συμφραζόμενα
Pattern	Μοτίβο
Exceptionality Scoring Function	Συνάρτηση Βαθμολόγησης Εξαιρετικότητας

ABBREVIATIONS - ACRONYMS

KG	Knowledge Graph

REFERENCES

[1] Gensheng Zhang, Damian Jimenez, Chengkai Li, Maverick: Discovering Exceptional Facts from Knowledge Graphs, 2018.
[2] Daniel Ringler, Heiko Paulheim, One Knowledge Graph to Rule them All? Analyzing the Differences between DBpedia, YAGO, Wikidata & co.
[3] Jeff Z. Pan, Siyana Pavlova, Chenxi Li, Ningxi Li, Yangmei Li, Jinshuo Liu, Content Based Fake News Detecting Using Knowledge Graphs
[4] https://yago-knowledge.org/
[5] https://www.dbpedia.org/