Collaborative Project

FROCKG - Fact Checking for Large Enterprise Knowledge Graphs

Project Number: E! 113314 Start Date of Project: 2020/01/01 Duration: 36 months

Deliverable 2.1: First version of the fact checking algorithms including initial benchmarking results

Dissemination Level	Public
Due Date of Deliverable	Month 18, 2021/06/30
Actual Submission Date	Month 18, 2021/06/30
Work Package	WP2
Tasks	T2.1, T2.2
Type	Report
Approval Status	Work in progress
Version	1.0
Number of Pages	17

Abstract:
This report presents the first prototype of the Fact Checking components of the FROCKG platform. This includes approaches for checking a single fact as well as an approach to check a complete knowledge graph. The report covers a description of the approaches as well as an evaluation.

Project by Eurostars.

History

Version	Date	Reason	Revised by
0.1	2021/05/06	final TOC	Michael Röder
0.2	2021/05/20	Drafted content	Michael Röder, Farshad Afshari
0.3	2021/06/09	Evaluation results added	Farshad Afshari
0.4	2021/06/14	First draft finished	Michael Röder, Farshad Afshari
1.0	2021/06/29	Final version	Michael Röder, Farshad Afshari

Author List

Organization	Name	Contact Information
Paderborn University	Michael Röder	michael.roeder@upb.de
Paderborn University	Farshad Afshari	farshad.afshari@uni-paderborn.de

Contents

Introduction

The main goals of this work package are twofold. First, we develop an approach to compute the veracity of single facts based on both textual evidence and corroborating facts found in a reference knowledge graph. To achieve this goal, the consortium develops novel solutions based on supervised explainable machine learning algorithms (T2.1).
The second goal is to check not only single facts but a set of facts (ranging from small sub-graphs to large knowledge bases) for their veracity. A simple approach relying on the check of every single fact in the knowledge base is not feasible for large knowledge bases. Hence, we develop a framework to select a subset of the knowledge base. The facts within this subset are then checked for their veracity and used to approximate the total veracity (T2.2).
This deliverable describes the first prototypes that were implemented to achieve the aforementioned goals. The University Paderborn (UPB) leads the development. It should be mentioned that because of the pandemic situation the position of a researcher working for the FROCKG project at the UPB couldn't be staffed before mid of August 2020 while the project already started in January 2020. To ensure that this won't delay the project, the work package lead decided to focus on the main functionalities of the two prototypes. This ensures that the prototypes fulfill the requirements defined in D1.2 [14] and D1.3 [15], and are ready to be used by the other work package.
The next section describes the developments that have been done within T2.1. This includes a description of the different Fact Checking services and their evaluation. The third section describes the knowledge graph veracity framework that has been developed within T2.2, and its evaluation. The last section concludes the deliverable and briefly summarizes the remaining targets within the second work package.

T2.1 Fact Checking single Facts

The task of checking a single fact can be defined as follows: "Given a fact, compute the likelihood that the given fact is true" [11]. In the context of the FROCKG project, the facts originate from a knowledge base, i.e., they can be represented as a triple (s, p, o) with a subject s, a predicate p and an object o. The triple represents a relation between s and o. The type of relation is expressed by p. For example, the triple ("Barack Obama", "nationality", "United States of America") represents the fact that Barack Obama is a citizen of the United States of America. We will use this example throughout this section.
Fact Checking approaches for single facts of the form (s, p, o) can be separated into two groups—text-based approaches and knowöedge-graph-based approaches. The FROCKG project makes use of both approaches. Text-based approaches rely on a reference corpus and use it to identify evidence for the given fact. Knowledge-graph-based approaches use knowledge graphs to derive knowledge that either supports or refutes the given fact. The techniques used for that can rely on the identification of paths between the given subject and object that support or refute the triple. Other approaches mine graph patterns which are used as evidence. However, it has been shown that path-based approaches outperform the

pattern-based approaches [11]. Hence, we prefer the path-based approaches when we work with knowledge graphs as reference knowledge graphs.
Within the FROCKG project, both groups of approaches are utilized. The interfaces of the FROCKG platform defined in D1.2 [14] have been defined in a way that one or several fact checking algorithms can be used. These algorithms are implemented as single services and are hidden using a facade.
Within T2.1, two services—a text-based and a knowledge-graph-based approach, respectively—are developed. Both are integrated into the facade service which will be the main service used by other components of the FROCKG platform.

Text-based Fact Checking

A text-based Fact Checking approach relies on a reference corpus to search for pieces of evidence that support the given fact. For the FROCKG platform, we decided to use the state-of-the-art approach FactCheck [1].
FactCheck uses a local corpus by utilizing Elasticsearch[1] to index the documents of the corpus. When it is used to check a given fact, the fact is transformed into keyword queries. For the running example, the following queries could be generated:

“Barack Obama nationality United States of America”
“Barack Hussein Obama II nationality U.S.A.”
“Barack Obama born in United States”
...

For generating queries, information about the three parts of a fact are combined. This includes the different labels of the subject and object as well as different formulations for the predicate. In the example above, FactCheck already knows that the nationality property is closely bound to the place of birth. Hence, the generated queries include this information.
These queries are used to search for relevant documents in the reference corpus. The retrieved documents are further analysed in two ways. First, parts of the text in which the searched keywords occur are extracted. In our running example, these pieces of text could look as follows:

“During Obama's terms in office, the United States' reputation abroad, as well as the American economy, significantly improved.”

“Obama was born in Honolulu (U.S.A.).”

Each extracted piece of text is deeply analysed and rated by a machine learning algorithm with respect to its quality. This includes the distance between the subject and the object of the given fact within the extracted text. In contrast to previous approaches (e.g., Gerber et al. [12]), FactCheck additionally makes use of the sentence structure instead of relying on simple string matching algorithms. In the example above, the analysis would reveal that Obama and the United States are not well connected within the text although their distance

[1] https://www.elastic.co/

is small. In the second piece of text, the two terms are connected via a verb that represents the predicate FactCheck searched for. Hence, the second piece of text would receive a better score by the machine learning model.

A second analysis that is run by FactCheck is the measurement of trustworthiness of the identified documents. To this end, FactCheck relies on the trust model presented by Nakamura et al. [13]. It uses articles that describe the subject and object (Nakamura et al. [13] suggest Wikipedia articles) and compares the content of the articles with the content of the retrieved document. This ensures that documents with a high topical overlap with the subject or object article get a higher trust score than documents with a different topic.

In its final step, FactCheck relies on a trained classifier that uses all identified pieces of evidence to decide whether the given fact is true or false. The result is a truth value that is returned as result.

We adapted the open-source implementation to the needs of the platform. This included several changes:[2]

- FactCheck has been mainly used for checking facts of the DBpedia. We removed this dependency and generalized its workflow to work with facts of any knowledge base.
- FactCheck solely returned the truth value. However, since the explanation of the Fact Checking process is of central importance for the FROCKG platform, we extended the service API of FactCheck according to D1.2 [14] to return the identified pieces of evidence and their detailed scores. This additional information is crucial for the implementation of the generation of explanations in WP3 of the FROCKG project.
- We created a Docker image to integrate the service into the FROCKG platform.
- We set up an environment that eases the evaluation of FactCheck's performance based on a given benchmark dataset. This allows an easier adaptation of FactCheck's configuration based on a given dataset.
- We extended a module within FactCheck that allows an easier configuration of additional property verbalizations. FactCheck mainly relied on pregenerated patterns to generate more diverse queries (e.g., "nationality" led to the usage of "born in" in the example above). Since these patterns won't be available for all RDF properties that will be used within the FROCKG project, we assured that manually created patterns can be added easily.

Knowledge-Graph-based Fact Checking

A knowledge-graph-based Fact Checking approach makes use of structured information of a reference knowledge graph to identify pieces of evidence that support or refute the given fact. To enhance the performance of the Fact Checking service implemented in the FROCKG platform, we decided to offer the usage of such a Fact Checking algorithm in addition to the text-based Fact Checking. To this end, we utilized the open-source approach COPAAL [11] and extended it to implement a reliable service that can be used in the platform. In addition, we extended its path search approach by implementing ESTHER—a search for corroborative paths that relies on knowledge graph embedding spaces. Both are described in more detail in the following.

[2] The changes are open-source and can be found at https://github.com/dice-group/FactCheck

COPAAL

COPAAL [11] is an open-source knowledge-graph-based Fact Checking algorithm. For a given fact, COPAAL searches for paths between the subject and object of the fact within the reference knowledge graph. Figure 1 shows some paths between the resources representing Barack Obama and the United States of America for our example. It can be seen that the paths can have different lengths (2 or 3 triples in the figure).

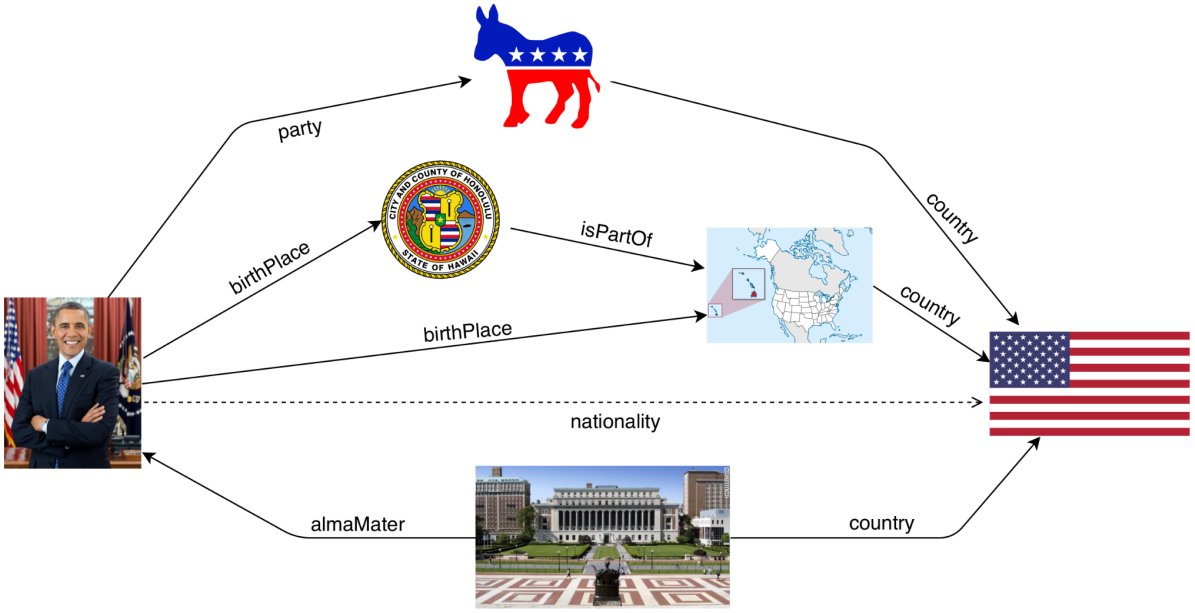

Figure 1: The fact that should be checked (the dotted line) and additional paths between the resources Barack Obama and the United States of America extracted from the DBpedia.

These paths are scored whether they corroborate the existence of the fact using an NPMI-based heuristic. To make COPAAL applicable for the FROCKG platform, we extended the existing implementation in the following ways:[3]

- We identified several problems in the existing prototypical implementation of COPAAL. We changed the structure of the project to make it extendible for future changes. During this project, several flaws have been fixed, e.g., the new implementation is able to rely on paths with an arbitrary size while the original implementation was limited to paths of the size 2.
- COPAAL makes use of domain and range information of the given fact's predicate. However, such information is not available for all RDF predicates.
- The summarization of the scores for the paths to form a final truth value for the given fact has been improved.
- We added the possibility to rely on exact counts instead of the heuristics suggested by Syed et al. [11]. This can be very helpful in scenarios in which the heuristic leads to bad results and the size of the graph allows the determination of the exact number of connected pairs.
- We created a Docker image to integrate the service into the FROCKG platform.

[3] The changes are open-source and can be found at https://github.com/dice-group/COPAAL

ESTHER

We extended the concept of the search for corroborative paths, i.e., the search implemented in COPAAL, to knowledge graph embedding spaces. Instead of the costly search within the knowledge graph based on individual given facts, ESTHER searches for corroborative paths within a knowledge graph embedding space. For a given property, ESTHER applies an A* search algorithm to generate candidates for corroborative paths. Identified paths are scored similar to the paths identified by COPAAL. Paths with a high score are stored for later reusage.
An advantage of ESTHER is that the search for corroborative paths can be applied as a preprocessing step. For checking a given fact, the algorithm is reduced to check the existence of the previously identified corroborative paths between the fact's subject and object. This reduces the runtime to check single facts.
ESTHER is integrated into the FROCKG platform by extending the COPAAL service to make use of both path search approaches—the original COPAAL search algorithm and the ESTHER algorithm—to identify corroborative paths.[4]

Fact Checking Facade

Following D1.2 [14], the FROCKG platform provides a Fact Checking facade service. This facade ensures that the usage of the Fact Checking components is eased by providing a single service that encapsulates all functionalities. This facade service includes the following functionalities:[5]

- Given a fact to check, it calls one or several Fact Checking services. Both types of Fact Checking services are integrated, i.e., text-based as well as knowledge-graph-based Fact Checking services.
- If multiple services are available, their results are merged. In the current version, the maximum score is used, i.e., if one of the services returns a high truth value for the given fact this value will be returned.
- The facade handles the pieces of evidence returned by the services and forwards them to the integrated explanation component which is developed with WP3. The result (i.e., the explanation of the Fact Checking) is added to the Fact Checking result that is sent to the user.
- In addition to an API to check the fact, it contains an API endpoint that returns the human-readable result of the fact and the explanation.

Evaluation

For our evaluation we focus on the evaluation of FactCheck, COPAAL and the facade service. In a second experiment, we evaluated ESTHER.

[4] The source code is open-source and available at https://github.com/dice-group/esther
[5] The source code is open-source and available at https://github.com/dice-group/FROCKG

Datasets

For the first experiment, we use the FactBench dataset created by Gerber et al. [12]. We use the property subset, since it is known to be the hardest dataset within FactBench. The dataset comprises 750 true and 750 false facts. The true facts are gathered from DBpedia while the false facts are generated. The generation is done by randomly choosing a triple (s,p,o) and replacing the property with p' to form the false fact (s,p',o). During that, it is ensured that the domain and range restrictions of p' are fulfilled [12].
For the second experiment, we use the datasets FB15k-237 and WN18RR. Both datasets come with a small knowledge graph and a set of true facts. The main features of the datasets are listed in Table 1. We randomly generated wrong facts as suggested by Gerber et al. [12]. We configure ESTHER to work with three different embedding algorithms: TransE [17], RotatE [18] and DensE [19]. In the experiment, we compare ESTHER to 10 other graph-based approaches. In the first run, we benchmark all 11 approaches on their own. In the second run, we combine each approach with ESTHER using a decision tree. Since this transforms the approach into a supervised approach, we use a 10 fold cross validation.

	FB15k-237	WN18RR
Entities	14541	40943
Relations	237	11
Triples	289650	89869
True facts	750	750

Table 1: Features of the two datasets used for evaluating ESTHER.

Results

We measure the performance of a Fact Checking approach using the area under the receiver operating characteristic curve (AUC-ROC) [1]. We report the results of experiment 1 in Table 2.[6] It can be seen that different configurations of COPAAL lead to a different performance. This shows that for the use cases of the FROCKG project different configurations of COPAAL should be tested as a single configuration may not always

[6] The experiment have been run on GERBIL and are available at http://w3id.org/gerbil/kbc/experiment?id=202106040005, http://w3id.org/gerbil/kbc/experiment?id=202106040001, http://w3id.org/gerbil/kbc/experiment?id=202106040002, http://w3id.org/gerbil/kbc/experiment?id=202106040003, http://w3id.org/gerbil/kbc/experiment?id=202106100001, http://w3id.org/gerbil/kbc/experiment?id=202106040009, http://w3id.org/gerbil/kbc/experiment?id=202106040007, http://w3id.org/gerbil/kbc/experiment?id=202106040010, http://w3id.org/gerbil/kbc/experiment?id=202106100003 .

perform best. The best performing configuration for FactBench's property subset defines the maximum path length to be 3 and uses virtual types. The results show that at least in this experiment longer paths lead to lower results. This is surprising since more information should lead to better results. However, we see it as a hint that these results point to further improvements that can be achieved within the months.
FactCheck achieves a lower AUC-ROC value than the best COPAAL configuration. However, the results of the Facade are all better than their corresponding COPAAL instances. Hence, we can conclude that FactCheck and COPAAL have different views on the given facts and that their combination leads to better results.

Approach	AUC-ROC
COPAAL (length 2)	0.6331
COPAAL (length 3)	0.4806
COPAAL (length 2, vt)	0.6112
COPAAL (length 3, vt)	0.6441
FactCheck	0.5872
Facade (length 2)	0.6896
Facade (length 3)	0.5702
Facade (length 2, vt)	0.6706
Facade (length 3, vt)	0.6988

Table 2: AUC-ROC results for COPAAL, FactCheck and the Facade on FactBench's property subset. Higher values are better.

We tested different configurations of ESTHER. The approach achieved an AUC-ROC of 83.07 for FB15k-237 when using RotatE embeddings and 77.55 on WN18RR with TransE embeddings. Table 3 shows that ESTHER achieves a better performance than most other approaches. However, some approaches (especially KS) perform better. However, ESTHER was able to improve the performance of all other approaches by 20% on average as shown in Table 3.

Approach	Approach only		With ESTHER	
	FB15k-237	WN18RR	FB15k-237	WN18RR

COPAAL [1]	77.42	68.11	87.12 (+9.70)	79.38 (+11.27)
KS [2]	87.59	86.44	89.97 (+2.38)	94.92 (+8.48)
Katz [3]	82.8	69.96	86.30 (+3.50)	86.22 (+16.26)
Pathent [4]	73.46	79.98	84.75 (+11.29)	86.94 (+6.96)
Simrank [5]	40.07	44.15	81.60 (+41.53)	82.47 (+38.32)
Adamic Adar [6]	72.12	59.86	85.36 (+13.24)	84.22 (+24.36)
Jaccard [7]	38.56	42.34	82.91 (+44.35)	87.18 (+44.84)
Degree Product [8]	77.11	65.57	83.28 (+6.17)	87.43 (+21.86)
PredPath [9]	69.87	80.2	83.76 (+13.89)	82.20 (+2.00)
PRA [10]	8.53	71.8	97.44 (+88.91)	75.35 (+3.55)

Table 3: AUC-ROC values achieved by different approaches with and without ESTHER. The values in parentheses show the difference. Higher values are better.

T2.2 Knowledge Graph Veracity

The goal of the T2.2 is the development of an algorithm that can measure the veracity of a given knowledge graph.

Approach

Given the size of knowledge graphs, it is impracticable to check every single fact. Hence, developing an algorithm that approximates the overall veracity of a knowledge graph by checking only a small portion of the graph is needed.
In the first part of this task, we started by defining a framework that comprises the following steps:

1. Fact selection
2. Fact conversion
3. Fact Checking
4. Summary generation

In the following, the single steps will be explained in more detail.[7]

[7] The implementation of the framework is open-source and can be found at https://github.com/dice-group/KGV

Fact Selection

The goal of this first step is to select a subset of triples. This is necessary since not all triples of a graph can be checked within a reasonable amount of time. The set of triples should be representative for the graph. There are several methods to select such a subset. However, since the focus of the first part of the project was to implement the framework itself, we focussed on a baseline method: the random selection of triples. We relied on an even distribution, i.e., each triple had the same chance to be chosen.

Fact Conversion

As described in the previous sections, Fact Checking algorithms typically rely on reference knowledge. This reference knowledge can be textual knowledge or structured knowledge. In the latter case, the knowledge graph could be checked with itself as reference. However, in many situations, it makes sense to use another knowledge graph as reference. To this end, it is necessary to convert the chosen facts into facts that match the URIs in the reference knowledge base.

Fact Checking

This step uses the Fact Checking services that are developed within T2.1. The result of this step is a veracity value for each fact.

Summary Generation

The last step comprises the generation of a summary that describes the overall veracity of the knowledge base. To this end, a final score is generated based on the calculated veracity scores. In our current implementation, we rely on the accuracy of the knowledge graph, i.e., the percentage of true facts. To this end, we define a threshold θ and count the number of facts that have a veracity value that is larger than θ.

Evaluation

For the evaluation, we use the Facade service with COPAAL and FactCheck as Fact Checking services. We evaluate our knowledge graph veracity approach by using a knowledge graph that should be checked and a reference knowledge graph that is used to evaluate single triples.

Datasets

We use YAGO 3.0 [16] because it already has been checked by humans in a similar setup. They manually checked 2014 facts and calculated the accuracy as explained above. The manual tests show that YAGO 3.0 has an accuracy of 98.0%.[8]

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

We use DBpedia as a reference dataset and Wikipedia as a reference corpus. For the fact conversion, we created a mapping from Yago properties to DBpedia properties. The entities are already linked with owl:sameAs links.

Results

We sampled 1000 facts from the YAGO 3.0 dataset. The Fact Conversion step matched 993 facts to the DBpedia. 7 facts couldn't be matched.
The calculated accuracy of the YAGO knowledge graph is shown in Table 4. The Facade service gives the best result when COPAAL is configured to use paths with the length 2 and virtual types.

Approach	Accuracy
COPAAL (length 2)	28.90%
COPAAL (length 3)	26.59%
COPAAL (length 2, vt)	38.87%
COPAAL (length 3, vt)	32.83%
FactCheck	19.54%
Facade (length 2)	38.67%
Facade (length 3)	36.66%
Facade (length 2, vt)	47.33%
Facade (length 3, vt)	41.89%

Table 4: calculated accuracy of Yago 3.0 according to the Fact Checking pipeline. Values closer to the expected accuracy of 98% are better.

Table 5 lists the detailed results for the single properties. Some properties lead to a low accuracy value compared to the results of the manual check. The `gender` property as well as the `hasWebsite` property are object properties in YAGO while they are datatype properties in DBpedia. COPAAL is built to check facts with object properties and, hence, is not able to find any evidence for this kind of fact. At the same time, FactCheck could not find textual evidence for the most facts with these properties.

Property	Checked Facts	Accuracy (%)	Gold Standard (%)
location	336	57.74	100.00

gender	177	8.47	100.00
team	73	89.04	100.00
isAffiliatedTo	97	7.22	100.00
wasBornIn	54	87.04	100.00
created	49	42.86	100.00
hasWebsite	39	0.00	100.00
actedIn	29	86.21	98.68
hasWonPrize	27	85.19	100.00
graduatedFrom	16	93.75	100.00
diedIn	16	68.75	100.00
isMarriedTo	11	81.82	100.00
hasChild	10	70.00	95.89
isPoliticianOf	8	0.00	100.00
owns	8	75.00	98.21
hasMusicalRole	7	57.14	100.00
isCitizenOf	7	100.00	100.00
directed	6	83.33	98.44
influences	4	0.00	94.95
happenedIn	4	50.00	100.00
livesIn	4	25.00	98.41
wroteMusicFor	3	0.00	100.00
participatedIn	2	50.00	100.00

hasCapital	2	100.00	97.37
isLeaderOf	2	50.00	100.00
edited	1	100.00	89.28
hasAcademicAdvisor	1	100.00	100.00

Table 5: Accuracy per property. Values closer to the expected accuracy of 98% are better.

Summary

With the work described in this report, WP2 fulfilled all necessary steps to achieve the project's second milestone.

The evaluation results of T2.1 are promising. We showed that the single services with their different approaches to verify a given fact work within the FROCKG platform. At the same time, the evaluation points out several ways to improve the single services as well as the overall FROCKG platform. These points will be part of the remaining work in the second half of the FROCKG project.

The runtime of COPAAL will be improved. The implementation of ESTHER showed that knowledge-graph-based Fact Checking approaches can be very fast if the main work is moved into a preprocessing step. In the case of ESTHER, this step is the A* search that identifies the paths that are used for the verification. This reduces the verification of a given fact to a simple lookup whether previously identified paths exist between the given subject and object. At the moment, the path search of COPAAL relies on the given fact and, hence, is executed as part of the verification process. However, it is possible to adapt this search and base it on a sample of facts that can be gathered from the knowledge graph. This allows us to decouple the path search from the fact verification and enables an implementation of the search as a preprocessing step.

For FactCheck, we saw a high impact of patterns that are used to generate queries to get relevant documents. This is an important insight for WP6 as we will have to come up with patterns for the properties that we want to use in the single use cases.

Further, ESTHER showed that it can improve the performance of other fact checking approaches. Hence, we will integrate it into the FROCKG platform as an optional service that can be used if the necessary knowledge graphs embedding are available for the reference knowledge graph.

Finally, our results underline that the combination of several Fact Checking approaches leads to an improvement. Hence, the development of the facade as an extendable component was a good choice and allows the further addition of other Fact Checking approaches if necessary.

The evaluation results of T2.2 show the effectiveness of the knowledge graph veracity framework. The next steps will be the further improvement of the single FactChecking approaches within T2.1 and the extension of the framework based on the use cases in WP5.

References

[1] Zafar Habeeb Syed, Michael Röder, and Axel-Cyrille Ngonga Ngomo: “FactCheck: Validating RDF Triples using Textual Evidence”. In Proceedings of the International Conference on Information and Knowledge Management (CIKM), 2018.

[2] Shiralkar, P., Flammini, A., Menczer, F., Ciampaglia, G.L.: “Finding streams in knowledge graphs to support fact checking”. In: 2017 IEEE International Conference on Data Mining (ICDM). pp. 859–864. IEEE (2017)

[3] Katz, L.: “A new status index derived from sociometric analysis”. Psychometrika 18 (1953)

[4] Xu, Z., Pu, C., Yang, J.: “Link prediction based on path entropy”. Physica A: Statistical Mechanics and its Applications 456, 294–301 (2016)

[5] Jeh, G., Widom, J.: Simrank: “A measure of structural-context similarity”. In: Proceedings of the Eighth ACM SIGKDD Intern. Conf. on Knowledge Discovery and Data Mining (2002)

[6] Adamic, L.A., Adar, E.: “Friends and neighbors on the web”. Social Networks 25(3) (2003)

[7] Liben-Nowell, D., Kleinberg, J.: “The link prediction problem for social networks”. In: Proceedings of the Twelfth Intern. Conf. on Information and Knowledge Management (2003)

[8] Shi, B., Weninger, T.: “Fact checking in large knowledge graphs - A discriminative predicate path mining approach”. CoRR abs/1510.05911 (2015)

[9] Shi, B., Weninger, T.: “Discriminative predicate path mining for fact checking in knowledge graphs”. Knowledge-based systems 104, 123–133 (2016)

[10] Lao, N., Cohen, W.W.: “Relational retrieval using a combination of path-constrained random walks”. Machine learning 81(1), 53–67 (2010)

[11] Zafar Habeeb Syed, Michael Röder, and Axel-Cyrille Ngonga Ngomo: “Unsupervised Discovery of Corroborative Paths for Fact Validation”. The Semantic Web – ISWC 2019 , page 630–646. Cham, Springer International Publishing, (2019)

[12] D. Gerber, D. Esteves, J. Lehmann, L. Bühmann, R. Usbeck, A. Ngonga Ngomo, and R. Speck: “DeFacto - Temporal and Multilingual Deep Fact Validation”. Web Semantics: Science, Services and Agents on the World Wide Web (2015)

[13] Satoshi Nakamura, Shinji Konishi, Adam Jatowt, Hiroaki Ohshima, Hiroyuki Kondo, Taro Tezuka, Satoshi Oyama, and Katsumi Tanaka: “Trustworthiness analysis of web search results”. Research and advanced technology for digital libraries (2007), 38–49.

[14] Deliverable 1.2: Component architecture and interfaces. FROCKG project. (2020)

[15] Deliverable 1.3: Requirements Specification. FROCKG project. (2020)

[16] Farzaneh Mahdisoltani, Joanna Biega, Fabian M. Suchanek: “YAGO3: A Knowledge Base from Multilingual Wikipedias”. Conference on Innovative Data Systems Research (2015).

[17] Bordes, Antoine and Usunier, Nicolas and Garcia-Duran, Alberto and Weston, Jason and Yakhnenko, Oksana: “Translating embeddings for modeling multi-relational data”. In Advances in neural information processing systems. 2787–2795 (2013).

[18] Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie and Jian Tang. “RotatE: Knowledge Graph Embedding by Relational Rotation in Complex Space”. In CoRR, abs/1902.10197 (2019).

[19] Haonan Lu and Hailin Hu: “DensE: An Enhanced Non-Abelian Group Representation for Knowledge Graph Embedding”. In arXic, cs.AI, 2008.04548 (2020).