Rule-based Fact Verification Utilizing Knowledge Graphs

Yuki Momii[1], Tetsuya Takiguchi[1] and Yasuo Ariki[1]

[1]*Graduate School of System Informatics, Kobe University, Japan*

Abstract

In this paper, we propose a method for performing Fact Verification using the structure of a knowledge graph and pre-defined rules. We focus on the classification of claims into five reasoning types based on the knowledge graph in FACTKG and manually create classification-specific rules. This method allows us to create rules that are independent of entities and relations. Experimental results with FACTKG show that rule-based judgments achieve a sufficient level of label accuracy.

Keywords

Knowledge Graph for Explainable Inference, Rule-based Fact Verification, Manually Created Rule

1. Introduction

Nowadays, with the widespread of misinformation, there is a growing demand for Fact Verification to determine the veracity of claims. In conventional Fact Verification, evidence is retrieved from external sources (Retriever), and the judgment is predicted by LLM (Reader) based on the evidence[1]. However, when LLM is used as the Reader, the inference process that leads to the final judgment becomes a black box, lacking explainability. On the other hand, a method has been proposed to perform rule-based judgments incorporating logical symbols as a Reader, rather than using LLM for judgment[2]. This method has high accuracy on FEVER[3], a representative dataset of Fact Verification, while enhancing explainability. In this paper, we propose a rule-based approach similar to [2], but utilizes a knowledge graph. We created rules to deal with the five reasoning types of Fact Verification defined in FACTKG[4]. A key difference from [2] is that external sources are interpretable knowledge graphs, not text. Experimental results with FACTKG show that rule-based judgments achieve a sufficient level of accuracy.

2. FACTKG

FACTKG (Fact Verification via Reasoning on Knowledge Graphs)[4] is a dataset for Fact Verification that leverages knowledge graph. It classifies claims into four reasoning types based on the structure of the knowledge graph (Table 1). For each type, Negation is also created by adding negations (e.g. ‘not’ or ‘never’), resulting in a total of five types.

- One-Hop: Claims can be verified with a single knowledge triple. In Table 1, if triple (*AIDAStella*, *Builder*, *Meyer Werft*) exists, it is labeled as SUPPORTED.

IJCKG 2023 Poster and Demo track

235x075x@gsuite.kobe-u.ac.jp (Y. Momii); takigu@kobe-u.ac.jp (T. Takiguchi); ariki@kobe-u.ac.jp (Y. Ariki)

- Conjunction: Claims can be verified with multiple knowledge triples. Both (*AIDAstella*, *Operator*, *AIDA Cruises*) and (*AIDAstella*, *Builder*, *Meyer Werft*) need to exist for it to be labeled as SUPPORTED.
- Existence: Claims can be determined based on whether a specific relation exists. If *Meyer Werft* has the *parentCompany*, it is labeled as SUPPORTED.
- Multi-Hop: Entities in Conjunction are transformed into common nouns. If both (AIDAstella, Builder, x) and (x, location, Papenburg) exist, it is labeled as SUPPORTED.

Table 1
Four different reasoning types of FACTKG cited from [4].r1: parentCompany, r2: Builder, r3: Operator, r4: location, m: Meyer Werft, s: AIDAstella, c: AIDA Cruise, p: Papenburg.

Reasoning type	Claim Example	Graph
One-Hop	AIDAstella was built by Meyer Werft.	$s \xrightarrow{r_2} m$
Conjunction	AIDA Cruise line operated the AIDAstella which was built by Meyer Werft.	$c \xleftarrow{r_3} s \xrightarrow{r_2} m$
Existence	Meyer Werft had a parent company.	$m \xrightarrow{r_1}$
Multi-Hop	AIDAstella was built by a company in Papenburg.	$s \xrightarrow{r_2} x \xrightarrow{r_4} p$

All claims are labeled as SUPPORTED or REFUTED. REFUTED claims are created by replacing an entity in SUPPORTED claims with another entity. In the case of One-Hop, relation substitution can also be done. Negation creation depends on the type. For One-Hop and Existence, they are created by adding negations into the claim and inverting the label. In the case of Conjunction, it is labeled as SUPPORTED only if negations are added to all parts with substituted entities. In Multi-Hop, the label is based on the actual knowledge graph. For example,"*AIDAstella was built by a company, not in Papenburg*" is labeled as SUPPORTED if (*AIDAstella*, *Builder*, x) and (x, *location*, y) exist, and y is not *Papenburg*. In Negation with Conjunction (or Multi-Hop), the number of entities are fixed at 3 (or 2), and the number of hops is 2. The rules are applied under those conditions. However, extensions are possible with sentence parsing.

3. Proposed Method

Our proposed method consists of three predictors(Relation, Reasoning type, Negation) and manually created rules. Appropriate rules are selected based on predictors' result.
All predictors employ BERT(bert-base-uncased)[5]. Relation Predictor takes"Claim[SEP]Entity" as input and predicts relations as multi-labels for all entities in the claim. Then they are combined to form Relation Set *R*. Reasoning type Predictor classifies the claim based on the categories in Table 1. Negation Predictor predicts whether negations are added.

Rule-based judgment

- One-Hop (Figure 1(a)): For this type, we verify whether the entities in the claim are directly connected each other. If not, we determine that the claim is based on incorrect knowledge triples and predict it as REFUTED. Otherwise, we verify if they're connected by the relations included in *R*, and if so, we predict it as SUPPORTED. This decision allows for handling cases where the claim incorrectly represents entities relationship.

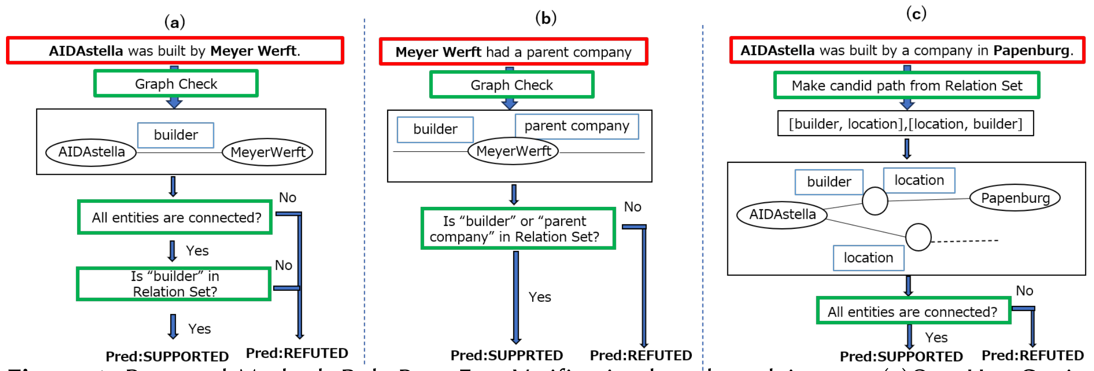
(a)
AIDAstella was built by Meyer Werft.
Graph Check
builder
AIDAstella
MeyerWerft
All entities are connected?
No
Yes
Is "builder" in Relation Set?
No
Yes
Pred:SUPPORTED
Pred:REFUTED
(b)
Meyer Werft had a parent company
Graph Check
builder
parent company
MeyerWerft
Is "builder" or "parent company" in Relation Set?
No
Yes
Pred:SUPPRTED
Pred:REFUTED
(c)
AIDAstella was built by a company in Papenburg.
Make candid path from Relation Set
[builder, location],[location, builder]
builder
location
AIDAstella
Papenburg
location
All entities are connected?
No
Yes
Pred:SUPPORTED
Pred:REFUTED

Figure 1: Proposed Method: Rule Base Fact Verification based on claim type (a)One-Hop,Conjunction(b)Existence(c)Multi-Hop

- Conjunction (Figure 1(a)): Like One-Hop, we check if all entities are connected or not.
- Existence (Figure 1(b)): In this type, we retrieve all relations possessed by the entity in the claim, and if there is any common relation with R, we predict it as SUPPORTED.
- Multi-Hop (Figure 1(c)): We arrange R to generate candidate paths. Then, we search for subgraph by exploring from one entity in the claim along candidate paths, and if all entities appear in subgraph, we predict it as SUPPORTED. Note that the hop count of subgraph is fixed at 3. This is because the maximum hop count in the dataset is 3, and the final judgment is not based on LLM. We need not to consider the size of subgraph.

Rule-based judgment with negated claims

- One-Hop, Existence: We reverse the predicted labels without negation.
- Conjunction: First, we examine the #*number* of entities that don't connect with other entities. #0: We predict it as REFUTED. #1: In the case of third line of Table 1, either of the entities at both ends is replaced. #3:This occurs when the central entity is replaced. In cases #1 and #3, subsequently, the negations and the position of the entity are identified through keyword matching. We predict it as SUPPORTED only when negations are present in all positions related to the substituted entity.
- Multi-Hop: In this case, we search for edge entities with predicted relations of each entity. Then common entities are excluded. The position of the entities and negations are determined in the claim, and N_e, entities not related to replaced entity is identified. If N_e has one or more excluded edge entities, we predict it as SUPPORTED.

4. Experiment

The baseline (Model-Based) refers to FACTKG's baseline. Relation prediction is performed in a manner similar to the proposed method, and the number of hops required for graph traversal is also predicted. Using the acquired subgraph, the final judgment is predicted by classifier(BERT(bert-base-cased) with linear layer). FACTKG is used to train our predictors and baseline's classifier as the dataset, but the test data lacks the necessary subgraph for judgment, so we divided the validation data into two parts to create new validation and test data. Besides the simple accuracy (LA), we report the evaluation of Evidence Enhanced Label Accuracy

Table 2
Fact verification accuracy with and without negation.

		One-Hop		Conjunction		Existence		Multi-Hop	
		Nor	Neg	Nor	Neg	Nor	Neg	Nor	Neg
Model Based	LA	0.8242	0.6986	**0.8421**	**0.8726**	0.9268	0.9621	0.707	0.6368
	EELA	0.6586	0.5284	0.4646	**0.5802**	0.9247	0.9621	0.2947	**0.3582**
	OLA	**0.9231**	0.8668	**0.9393**	**0.8962**	0.9978	0.9962	0.7480	**0.7711**
Rule Based(Ours)	LA	**0.8375**	**0.8652**	0.7693	0.7925	**0.9398**	**0.9697**	**0.7238**	**0.6418**
	EELA	**0.7009**	**0.734**	**0.4844**	0.5708	**0.9376**	**0.9697**	**0.3757**	0.3483
	OLA	0.9027	**0.9787**	0.8814	0.8443	**1.000**	**1.000**	**0.9000**	0.7562

(EELA), which considers relation prediction errors as judgment errors. Furthermore, Oracle Label Accuracy (OLA), which indicates the accuracy when correct relations are provided. Table 2 shows the results. The proposed method performed better in many of the reasoning types.

5. Conclusion

We proposed a rule-based method for fact verification using a knowledge graph, aiming for better explainability and accuracy. Difference between LA and EELA of model-based method reveals that it may predict correct results with incorrect reasoning. Future work will focus on quantitative assessments of explainability. Furthermore, we will incorporate grammatical interpretation to avoid errors in determining relations related to negations.

Acknowledgements
This work was supported in part by JSPS KAKENHI (Grant No. JP21H00906).

References

[1] M. DeHaven, S. Scott, BEVERS: A general, simple, and performant framework for automatic fact verification, in: Proceedings of the Sixth Fact Extraction and VERification Workshop (FEVER), Association for Computational Linguistics, Dubrovnik, Croatia, 2023, pp. 58–65. doi:10.18653/v1/2023.fever-1.6.

[2] A. Krishna, S. Riedel, et al., Proofver: Natural logic theorem proving for fact verification, Transactions of the Association for Computational Linguistics 10 (2021) 1013–1030.

[3] J. Thorne, A. Vlachos, et al., FEVER: a large-scale dataset for fact extraction and VERification, in: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), Association for Computational Linguistics, New Orleans, Louisiana, 2018, pp. 809–819. doi:10.18653/v1/N18-1074.

[4] J. Kim, S. Park, et al., FactKG: Fact verification via reasoning on knowledge graphs, in: Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Association for Computational Linguistics, Toronto, Canada, 2023, pp. 16190–16206. doi:10.18653/v1/2023.acl-long.895.

[5] J. Devlin, M.-W. Chang, et al., BERT: Pre-training of deep bidirectional transformers for language understanding, in: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), Association for Computational Linguistics, Minneapolis, Minnesota, 2019, pp. 4171–4186. doi:10.18653/v1/N19-1423.