Research Study: Option A — CS 421

FACTKG: Fact Verification via Reasoning on Knowledge Graphs

Sarthak Jain[1, ID]

[1]University of Illinois Chicago

Reproducibility Summary

Scope of Reproducibility — The paper[1] aims to show that incorporating a Knowledge Graph (KG) as the evidence in the claim verification process improves the accuracy as compared to when no external knowledge is considered for fact verification. For this purpose, following two claims are formulated:

1. Claim 1: Incorporating a KG improves overall accuracy of the claim verification.
2. Claim 2: Incorporating a KG improves the accuracy of each claim type as well besides the overall accuracy.

Methodology — The code for the paper is published by the authors on their GitHub repository. There were some parts of the code that needed modifications and improvements. The improvements were conducted to accurately reproduce the results. The experiments were performed using a T4 GPU(16 GB RAM) of Google Colab. The training took an overall of 14 GPU hours.

Results — The claims stated in the paper hold in this reproducibility study. Specifically, for Claim 1, the obtained overall accuracy of the claims with KG included is better and within 2 % of the reported value. Also, the accuracy value is better than the best baseline that considers no KG in claim verification by 17 %. For Claim 2, the results are clearly in favour of including KG: for all the 5 claim types, when KG is included, the obtained accuracy is better than the best baseline.

What was easy — The author's code was readily available on Github. The instructions to run the code were also available and clearly written in the readme file on GitHub. Besides this, the folder structure on GitHub was clear so as to understand how the code files are organized. Finally, a description of the data used and the link to data was provided on Github.

What was difficult — Although the code was provided on GitHub and the instructions to run it were clear, there were a few issues with the code. First, there were a few libraries that were used in the code, but they were not mentioned in the requirements. Second, the code had some missing pipelines that had to be implemented,and bugs that had to be corrected. A section of the experiments also had missing data, due to which the results for that section could not be verified. A detailed discussion of these issues is presented in Section 5.2

Communication with original authors — There were a few results that were off from the reported results, and some experiments that we were unable to conduct due to missing data. Thus, we emailed the authors for their comments. The authors replied with their insights and their comments are discussed in Section 5.3.

1 Introduction

The paper[1] presents the idea that incorporating KG for fact verification improves the accuracy of verification as compared to the case when no external knowledge is included. To prove the hypothesis that KG improves fact verification accuracy, the authors first construct a dataset of claims and their labels(true/untrue). There are five different types of claims in the dataset. More details on the dataset are presented in Section 3.2. Having constructed a dataset, the authors divide the models into 2 categories: One category does not use KG in fact verification, whereas the other category uses KG in fact verification. It is then shown that models using KG for fact verification achieve better accuracy.
The baseline models which do not include KG on fact verification are BERT[2], BlueBERT[3], and Flan-T5[4]. We run each of these three models on the dataset constructed by the authors. We report both the overall accuracy and the accuracy of each of the five claim types. The model that includes the KG in claim verification employs GEAR[5] framework. For this model as well, we report both the overall accuracy and the accuracy for each of the five claim types. Finally, we compare the overall accuracy and accuracy by claim type of the model using GEAR framework against the three baseline models.

2 Scope of reproducibility

As stated in Section 1, the paper aims to show that incorporating KG improves the claim verification accuracy. The objective is also to show that KG can aid in verifying claim of any type (the details on claim types are presented in Section 3.2, but there are 5 claim types). Concretely, the following two claims are presented:

1. Claim 1: Incorporating a KG in claim verification using the GEAR framework gives better overall accuracy than the baseline models of BERT, BlueBERT, and Flan-T5.
2. Claim 2: Incorporating a KG in claim verification using the GEAR framework gives better accuracy for each claim type(the details of five claim types are presented in section 3.2) when compared to the baseline models of BERT, BlueBERT, and Flan-T5.

The dataset for the two claims is the custom dataset created by the authors and the details on the dataset are presented in section 3.2.

3 Methodology

The code for the paper was available publicly on GitHub, and thus we directly used their code to report the results. There were, however, some sections in the code that were incomplete or had bugs. We debugged those sections and added missing pipelines to obtain accurate results. A detailed discussion of the issues faced with the code is presented in Section 5.2. The experiments were conducted on Google Colab Pro+, with T4 GPU(16 GB RAM).

3.1 Model descriptions

The three baseline models: BERT, BlueBERT and Flan-T5 are transformer based text classification models. We use all the baseline models from the HuggingFace library. Additional details of each model are provided below.

- **BERT**: BERT[2] is a transformer architecture based model. The word embeddings are contextual in the sense that they vary based on the context within which the word appears, and the context on both the left and right directions is considered to generate the embeddings. BERT is pre-trained on Wikipedia dataset. Hence, it is expected that it will use the knowledge memorized from the Wikipedia dataset to classify claims in the FactKG dataset, and thus a good performance of BERT is expected. For the purpose of experiments, bert-base-uncased version of BERT is used which has 110M parameters and trained on lower case English text.

- **BlueBERT[3]**: This is also a BERT model, but it is pre-trained on Pubmed abstracts and MIMIC-III clinical notes. It is used in the paper as a comparator to BERT since it has never seen Wikipedia during training, and hence is expected to perform worse than BERT.
- **Flan-T5**: Flan-T5[4] is an enhanced version of T5 model. T5 is an encoder-decoder based model that converts every NLP task to text-to-text format, with a task specific prefix. Flan-T5 is better than T5 as it has been trained on an additional 1.8K tasks and also covers more languages. There are different checkpoints of Flan-T5, but we use Flan-T5 large checkpoint with 770M parameters. For the fact verification task, the authors use the prefix: ”Is this claim True or False? Claim: ” for each claim. Then they measure the probability that the tokens True and False appear in the output, and chose the token with the higher probability. Flan-T5 is used in a zero-shot setting.

The model that incorporates KG into fact verification process is a modified version of GEAR framework. GEAR[5] was originally designed to incorporate multiple piece of text evidence in the verification pipeline. The authors modify GEAR to incorporate KG evidence. Specifically, this is how GEAR is modified:

1. Subgraph Retrieval: The first step is to retrieve a subgraph from the given claim and the entities in the claim. The subgraph retrieval step is divided into 2 phases: (a) Relation prediction and (b) hop prediction.
    - **Relation prediction:** In this step, a claim is concatenated with a given entity in the claim to predict the relations with which other entities in the claim are linked to the given entity. A BERT[2] model(bert-base-uncased) is trained for relation prediction task.
    - **Hop prediction:** In this step, a claim is concatenated with a given entity in the claim to predict the maximum no of hops that need to be traversed from the given entity to any other entity in the claim in the given Knowledge Graph. Similar to the relation prediction step, a BERT model[2] (bert-base-uncased) is trained for this step.

    Using the predicted relations and maximum hops for each entity in the claim, a subgraph is retrieved from the Knowledge graph. The subgraph in this context means a set of paths. A path starts at an entity in the claim, and ends at another entity in the claim, which should be at a maximum distance of the predicted maximum hops.
2. Claim classification: In this step, for a given claim, all the paths retrieved from the knowledge graphs are appended to the given claim. Then using this concatenated input, the claim is classified as true or false. Again, a BERT[2] model(bert-base-uncased) is trained for this task.

3.2 Datasets

The dataset used for the paper is constructed by the authors themselves for the purpose of showing the efficacy of incorporating Knowledge Graph into claim verification. The dataset can be found here. The dataset consists of claims and their labels. Each claim is a single sentence created using triples from WebNLG. The claim type depends on the way in which the triples are combined to generate the claim. The dataset consists of five types of claims: (1) One-hop, (2) Conjunction, (3) Existence (4) Multi-hop, and (5) Negation. The claim label is a binary label: True(Claim is true) or False(Claim is false). The dataset is balanced: the ratio of true to false labels is 1:1 (0.97:1, to be precise). The number of different claim types is tabulated in Table 1.

Type	Count
One-hop	19530
Conjunction	37097
Existence	9172
Multi-hop	27262
Negation	15613
Total	108674

Table 1. Counts of claims types

The knowledge Graph used in the paper is the DBpedia knowledge graph, which extracts its relations and entities from Wikipedia. Authors provide two versions of the knowledge graph: (1) This version is the entire DBpedia KG, (2) This version contains only the triples that are present in the given claim dataset, and is sufficient for the experiments presented in the paper. **We use the second version, since using the entire knowledge graph requires considerable cloud space.**
The dataset is divided into train/validation/test split in the ratio of 8:1:1. The split is made such that the claim triples are disjoint across the splits. Each claim in training and validation split is represented as a dictionay, where key is the claim text, and value contains:

- The entities present in the claim.
- For each entity, the list of relation paths through which other entities in the claim are connected to the given entity.
- The label of the claim(True/False), and
- The type of the claim and other metadata in the claim.

The test set claims are also represented in the same way, with the only difference being that the second point above is not present in the test set. The number of claims in each type in each of the train, validation, and test split is presented in Table 2

Type	Train	Validation	Test	Total
One-hop	15069	2547	1914	19530
Conjunction	29711	4317	3069	37097
Existence	7372	930	870	9172
Multi-hop	21833	3555	1874	27262
Negation	12382	1917	1314	15613
Total	86367	13266	9041	108674

Table 2. Counts of claim types in each of train, validation, and test splits

As stated in Section 3.1, for incorporating Knowledge Graph into claim verification, BERT based relation and hop predictors need to be trained. Data preprocessing is done for these tasks. Specifically, for these tasks, in the training and validation set, the claim is concatenated with an entity in the claim, and this is treated as the BERT input. ***During test phase, for a given claim, all the entities in the claim are concatenated to the claim and***

this concatenated text is fed to the model input. This way of preprocessing the data is an issue and a discusion on this is presented in Section 5.2
For the claim classification step presented in Section 3.1, a different data preprocessing is performed. In the training and validation set, each claim is concatenated with the relation paths(already given with the dataset) for all the entities in the claim. For the test set, each claim is concatenated with the relation paths (generated using the predicted relations and hops) for all the entities in the claim. For each of the three splits, the preprocessed claim is treated as the BERT model input.

3.3 Experimental setup and code

There are three baseline models that we evaludate, and the GEAR framework based method of incorporating KG for fact verification. The code was published by the authors and was adapted to rectify errors and implement pipelines. The rectified code used for the study is available here. All the experiments use accuracy as the evaluation metric since the dataset is balanced. The setup for each of them is presented below:

- BERT[2]: This uses the bert-base-uncased version of BERT from the HuggingFace transformers library. The model is trained on the claims in the training split, and finally tested on the test split. The loss function used to train the model is CrossEntropyLoss function from torch.nn module, and it is minimized using the Adam[6] optimizer
- BlueBERT[3]: This uses the bionlp/bluebert_pubmed_mimic_uncased_L-12_H-768_A-12 model from the HuggingFace Transformers library. The model is trained on the training split, and evaluated on the test split. Like BERT, the loss function used to train the model is CrossEntropyLoss function from torch.nn module, and it is minimized using the Adam[6] optimizer.
- Flan-T5[4]: This is used in a zero-shot setting, so no training is performed, and the model is directly evaluated on the test set.
- GEAR[5]: This consists of the relation prediction step, hop prediction step, and the classification step. The setup of each of them is presented below:
  - Relation prediction: The preprocessing of train, validation, and test set is done as mentioned in Section 3.2. The task is treated as a multi-label prediction task since there can be multiple relations for an entity in the claim. AdamW[7] is utilized to minimize the loss function. The BERT model is trained on the preprocessed training set, validated on the preprocesses validation set,and finally tested on the preprocessed test set.
  - Hop prediction: The setup is exactly the same as for relation prediction, with the only difference being that the task is no more a multi-label classification task since there can be only one maximum hop value for an entity in the claim. The task is a multi-class classification task. The loss function used to train the model is CrossEntropyLoss function from torch.nn module, and it is minimized using the AdamW[7] optimizer.
  - Classification step: As mentioned in section 3.2, the training, validation, and test sets are generated after preprocessing. A BERT model is used on these preprocessed splits to predict the label of each claim. The loss function used to train the model is CrossEntropyLoss function from torch.nn module, and it is minimized using the Adam[6] optimizer .

3.4 Hyperparameters

There are four major models trained and evaluated in this reproducibility study. Neither the paper, nor the code documentation specifies how the hyperparameter tuning was performed for the datasets, thus we directly uses the hyperparameter values specified in the code. The hyperparameters associated with each model are presented below:

- BERT: The training, validation, and test batch size is 64. Model is trained on three epochs, with a learning rate of 1e-4.
- BlueBERT: The hyperparameter values are exactly the same as for BERT
- Flan-T5: Since this is used in a zero-shot setting, the only hyperparameter is the test set batch size which is set to 64.
- GEAR: The hyperparameter values for each of the three steps involved are presented below:
  - **Relation prediction:** The BERT model is trained for 10 epochs with training, validation, and test batch size of 125. The no of relations predicted for each claim in test set is three. The learning rate of the AdamW optimizer is 5e-5.
  - **Hop prediction:** The BERT model is trained for one epoch with training and validation batch size of 32. The test batch size is 1. The learning rate of AdamW optimizer is 5e-5.
  - **Claim Classification:** The BERT model is trained for 10 epochs (with early stopping) with training and validation batch size of 16. The test batch size is also 16. The learning rate is 5e-5.

3.5 Computational requirements

The experiments were run on a single Tesla T4 GPU(16 GB RAM). The training of the models took a total of 14 GPU hours. With the fact that the test set size is 9041, the breakdown of training time and the test time for all the models is presented in Table 3

Model	Training Time(GPU hours)	Test Time(s)
BERT	2	61
BlueBERT	2	68
Flan-T5	NA	151
BERT:Relation Prediction	2	NA
BERT: Hop Prediction	2.5	NA
GEAR	5.5	NA

Table 3. Training and validation times for different models. Flan-T5 is used in a zero-shot setting, hence there is no training time. For relation, hop prediction and GEAR, test time was not recorded

4 Results

The results of this study support the claims made in the paper. Incorporating KG gave both better overall accuracy and better accuracy by claim type as well. The results are presented in the following sections.

Model	One-hop	Conjunction	Existence	Multi-hop	Negation	Total
BERT	64.10	59.79	65.17	56.99	61.56	60.90
BlueBERT	44.93	55.94	50.11	48.82	54.79	51.41
Flan-T5	63.89	67.28	53.56	62.43	53.57	62.24

Table 4. Obtained Results for baseline models

Model	One-hop	Conjunction	Existence	Multi-hop	Negation	Total
BERT	69.64	63.31	61.84	70.06	63.62	65.20
BlueBERT	60.03	60.15	59.89	57.79	58.90	59.93
Flan-T5	62.17	69.66	55.29	60.67	55.02	62.70

Table 5. Reported Results for baseline models

4.1 Baseline Results

The results of this study for baselines are presented in Table 4. The baseline results reported in the paper are presented in Table 5
From Table 4 and Table 5, we see that:

- For BERT, the obtained result for total accuracy is less than the reported value in the paper and within 5% of the reported value. If we consider the claims by types, then all types except existence have lower obtained accuracy values.
- For BlueBERT, the obtained result for total accuracy is less than the reported value in the paper and within 9% of the reported value. If we consider the claims by types, then all types have lower obtained accuracy values.
- For Flan-T5, the obtained overall accuracy value is approximately equal to the reported value. If we consider accuracy by claim types, then the results are mixed but within 2% of reported value.

Thus, we can conclude that BERT and BlueBERT results are not reproducible as they are quite far away from the reported values. However, Flan-T5 results are reproducible.

4.2 GEAR results

The results of this study for GEAR are presented in Table 6. The results reported in the paper for GEAR are presented in Table 7

One-hop	Conjunction	Existence	Multi-hop	Negation	Total
81.82	84.13	91.95	61.26	81.42	79.26

Table 6. Obtained results for GEAR

From Table 6 and Table 7, we see that the obtained overall accuracy of GEAR is higher than the reported value and is within 2% of the reported value. However, for claim types Conjunction and Existence, the obtained accuracy is more by 7% and 10% respectively. For multi-hop claims, the obtained accuracy is lower than the reported value by 7 %.

4.3 Overall Results

For all reasoning types, the obtained overall accuracy value of GEAR is better than the next best baseline(Flan-T5) by 17%. Also, for each claim type, the obtained accuracy of GEAR is higher than the baseline accuracies.

One-hop	Conjunction	Existence	Multi-hop	Negation	Total
83.23	77.68	81.61	68.84	79.41	77.65

Table 7. Reported Results for GEAR

5 Discussion

From Section 4.3 we see that the obtained overall accuracy of GEAR is higher than the next best baseline(Flan-T5) by 17%. **This supports Claim 1**. The obtained accuracy by claim type is also higher than the baseline accuracies. **This supports Claim 2**. *Thus, the results of this study support both the claims made in the paper[1].*
Although the claims made in the paper are supported by this study, there are a few observations in order. First, as shown in Section 4.1 the results for BERT and BlueBERT are not reproducible. The difference in accuracy is high for multi-hop claim type(for BERT, the difference between the reported and obtained accuracy value for multi-hop claim type is 14%, and for BlueBERT the difference is 9 %). For GEAR also, the obtained accuracy for claim type multi-hop is much lower(7%) than the reported value. The reason for this anomaly for GEAR is discussed in Section 5.2
Second, although the accuracy of multi-hop claim type is not reproducible for baselines or for GEAR, we see that the accuracy of GEAR for multi-hop claim type is higher than that of baselines.

5.1 What was easy

The author's code was readily available on Github. The instructions to run the code were also available and clearly written in the Readme file on GitHub. Besides this, the folder structure on GitHub was clear so as to understand how the code files are organized. Finally, a description of the data used and the link to data was provided on Github. Considering that there were an overall 6 training steps, the training GPU hours(14) that were spent in this study is quite reasonable. The GPU requirements were also not very extensive. We used a T4 GPU(16 GB RAM), but much better GPUs such as A100 and V100 GPU are available, but they were not required.

5.2 What was difficult

Although the instructions to run the code were clearly available in the author's GitHub Readme, there were a few issues with the code which are summarized below:

- The code to report accuracy by claim type was not implemented for baseline models. Thus, that part of the code had to be implemented. Furthermore, there was an issue with the code for Flan-T5 model wherein the last test batch was not getting evaluated resulting in inaccurate accuracy results. That part of the code had to be rectified
- For BERT, all the layers were being unfrozen, which is not typical in finetuning. This is probably the reason why the reported and obtained results for BERT are relatively lower considering the fact that BERT is trained on the Wikipedia dataset. However, we contacted the authors regarding this and they replied with a different reason for lower obtained accuracy. Their comments are discussed in 5.3.
- For the claim classification part of the GEAR framework, for a given entity, the path to any entity in the claim that is within the predicted maximum hops of the given entity should be included. However, this is not the case in the code implemented by the authors. Only the paths to the entities in the claim that are at a distance of predicted maximum hops(in the KG) from the given entity in the claim

are included. This means that relation paths that are being extracted from KG is only a subset of the actual paths in the KG. This is probably the reason for relatively lower obtained accuracy of multi-hop claims as compared to other claim types for GEAR.

- The train and test splits for relation and hop prediction have different data distributions. In the train split the concatenated input contains the claim and a single entity in the claim, whereas in the test set, the concatenated input to the BERT model contains the claim and all the entities in the claim. This results in different distributions of train and test set.
- For hop prediction, the maximum number of hops for any entity in any claim is three. However, this is not stated in the paper or the author's GitHub repo. This is an issue because, this value is being set manually in the code when specifying the no of labels during BERT model initialization. Ideally, this should have been done programmatically.
- For baseline models, libraries like sentencepiece and wandb are being imported but they are not mentioned in the requirements file present in the GitHub repo of the Authors. This resulted in code failing in the beginning.
- When training the hop prediction model and the final classification model in GEAR, we ran out of GPU RAM. So we reduced the batch size(from 64 to 32 for hop prediction and from 32 to 16 for GEAR claim classification) to overcome the issue.

Finally, the format of the data used in the code was slightly different than the format presented in the GitHub repo of the authors. In the code, a particular claim could have two claim types: negation and any of the other four types. This was because negation claims are generated by negating other claim types. However, if this is the case, then the claim is counted as only a negation type and not the other claim type. This was an issue because this was not stated in the GitHub repo which led us to assume that a claim can have only a single type in the code, thus leading to incorrect dataset statistics when analyzing the dataset.

5.3 Communication with original authors

The BERT and BlueBERT results were off from the reported results, and there were some experiments that we were unable to conduct due to missing data. Thus, we emailed the authors for their comments. For BERT and BlueBERT, the authors stated that they obtained highest validation accuracy in the initial stages of the training, and thus used the model checkpoint at that point to obtain BERT results. But when we ran the BERT experiment, we ran the model up to last epoch and used that model to evaluate on the test dataset. This is the reason the obtained accuracy of BERT is lower than the reported accuracy.
The authors also provided insights into steps for the experiments that we were unable to conduct. Implementing those insights is left for future work.

References

1. J. Kim, S. Park, Y. Kwon, Y. Jo, J. Thorne, and E. Choi. **FactKG: Fact Verification via Reasoning on Knowledge Graphs**. 2023. arXiv: 2305.06590 [cs.CL].
2. J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. **BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding**. 2019. arXiv: 1810.04805 [cs.CL].
3. Y. Peng, S. Yan, and Z. Lu. "Transfer Learning in Biomedical Natural Language Processing: An Evaluation of BERT and ELMo on Ten Benchmarking Datasets." In: **Proceedings of the 2019 Workshop on Biomedical Natural Language Processing (BioNLP 2019)**. 2019, pp. 58–65.

4. H. W. Chung et al. **Scaling Instruction-Finetuned Language Models**. 2022. arXiv: 2210.11416 [cs.LG].
5. J. Zhou, X. Han, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun. “GEAR: Graph-based Evidence Aggregating and Reasoning for Fact Verification.” In: **Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics**. Ed. by A. Korhonen, D. Traum, and L. Màrquez. Florence, Italy: Association for Computational Linguistics, July 2019, pp. 892–901.
6. D. P. Kingma and J. Ba. **Adam: A Method for Stochastic Optimization**. 2017. arXiv: 1412.6980 [cs.LG].
7. I. Loshchilov and F. Hutter. **Decoupled Weight Decay Regularization**. 2019. arXiv: 1711.05101 [cs.LG].