THiFly_Queens at SemEval-2021 Task 9: Two-stage Statement Verification with Adaptive Ensembling and Slot-based Operation

Yuxuan Zhou[1], **Kaiyin Zhou**[2,3], **Xien Liu**[1], **Ji Wu**[1], **Xiaodan Zhu**[4]
[1]Department of Electronic Engineering, Tsinghua University, Beijing 100084, China
[2]THiFLY Research, Tsinghua University, Beijing 100084, China
[3]State Key Laboratory of Cognitive Intelligence, Hefei, Anhui 230088, China
[4]ECE & Ingenuity Labs Research Institute, Queen's University
zhouyx17@mails.tsinghua.edu.cn kyzhou3@iflytek.com
{xeliu, wuji_ee}@mail.tsinghua.edu.cn
xiaodan.zhu@queensu.ca

Abstract

This paper describes our system for verifying statements with tables at SemEval-2021 Task 9. We developed a two-stage verifying system based on the latest table-based pre-trained model GraPPa. Multiple networks are devised to verify different types of statements in the competition dataset and an adaptive model ensembling technique is applied to ensemble models in both stages. A statement-slot-based symbolic operation module is also used in our system to further improve the performance and stability of the system. Our model achieves second place in the 3-way classification and fourth place in the 2-way classification evaluation. Several ablation experiments show the effectiveness of different modules proposed in this paper.

1 Introduction

Verifying whether a statement is entailed, refuted, or unknown with the given table is a challenging task, which requires the system to understand the statement and table jointly and reasons from the two information resources. The studies on this task could benefit several downstream applications, e.g. fake news detection. Recently, with the release of a large-scale dataset for table-based fact verification named TABFACT (Chen et al., 2019), this task received several studies. Verifying statements based on tables is a challenging task since the researches on understanding tables are not enough compared with works on free-texts, and the methods to train models understanding free-text and table jointly need further studies as well.

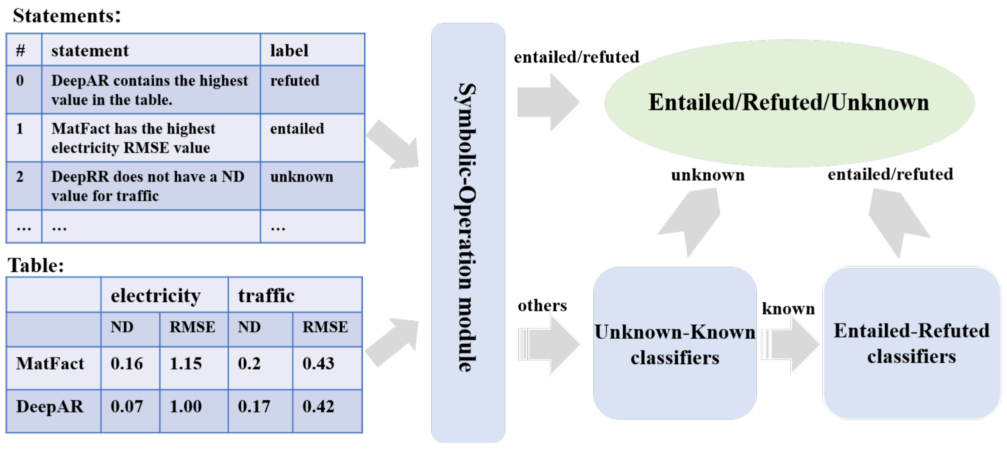
While TABFACT contains a huge number of statements and tables, the tables in the TABFACT dataset are relatively simple since they do not have hierarchical column heads as tables in scientific papers do and the contents in the tables are easier to understand compared to the tables in scientific papers as well. In the SemEval task 9 (Wang et al., 2021), the goal is to develop a system that can verify statements (**subtask A**) and find evidence (**subtask B**) based on the tables extracted from scientific tables. Our team is more interested in the verifying task and only participated in subtask A. Different from data in TABFACT, in subtask A, we are also required to classify a new type of statement that cannot be entailed or refuted based on the given table, named "unknown" type. Since no statements of this type are given in the training data, the classification of this type of statement becomes a core difficulty of the subtask.

This paper describes our two-stage table-based verifying system based on the latest table-based pre-trained language model GraPPa (Yu et al., 2020). The system leverages the model ensembling technique to ensemble different verifying models which are designed to solve different types of statements in the dataset in both two stages of the system. The statement-slot technique is also used to capture and solve a small part of the data by symbol calculation, which helps to increase the performance and stability of our system. Our system achieves a two-way score of 84.55 and a three-way score of 83.76 in subtask A respectively.

2 Related Work

2.1 GraPPa

GraPPa is a pre-trained model for table-based semantic parsing task, proposed by Yu et al. (2020). It is pre-trained on the synthetic question-SQL pairs with a novel text-schema linking objective, where the SQL queries are generated by a synchronous context-free grammar(SCFG). The text-schema linking objective makes the model predict the syntactic roles of the columns in the SQL to encourage the model to notice the link between



#	statement	label
0	DeepAR contains the highest value in the table.	refuted
1	MatFact has the highest electricity RMSE value	entailed
2	DeepRR does not have a ND value for traffic	unknown
…	…	…

	electricity		traffic	
	ND	RMSE	ND	RMSE
MatFact	0.16	1.15	0.2	0.43
DeepAR	0.07	1.00	0.17	0.42

Figure 1: The flowchart of the system. Statements and tables are first sent into the symbolic-calculation module, where statements with simple sentence structures are verified by symbol calculation. Statements that are not processed by the calculator are then sent into the two-stage deep-learning-based verifying system, where the first stage's models verify whether the statement cannot be verified by the given table, and the second stage's models verify whether the statement can be entailed by the given table.

natural language phrases and the corresponding logical form constituents.
The authors of GraPPa use 475k synthetic examples to pre-train GraPPa. On four popular semantic parsing benchmarks, GraPPa consistently achieves state-of-the-art results, which shows the powerful table understanding ability of GraPPa. In this work, we find that GraPPa's ability of understanding tables can also be migrated to the table-based fact verification task and achieves great performance on this task.

2.2 Mixture-of-Experts layer

In Shazeer et al. (2017) the authors proposed a sparsely mixture-of experts layer applied on the stacked LSTM model which achieves new state-of-the-art results on language modeling and machine translation benchmarks with lower computational cost and larger model capacity. In this work, we applied this MoE layer on the top of the GraPPa model as a part of our verifying system.

3 System Description

This section mainly describes the details of the two-stage table-based verifying system. The first stage classifies unknown type statements from the other two types of statements, while the second stage further classifies the entailed type statements from refuted type statements. We find that the proposed two-stage system works better than a direct three-way classification system, because 1) no unknown type statements are provided in the train set, which brings difficulty to the training process of the three-way classification system; 2) the two-stage system is closer to the processing procedure of human since you have to decide whether the table knowledge is enough to entail or refute the given statement before the further reasoning. Before the two-stage deep-learning-based system, a symbolic-calculation module is added to capture and process some statements with simple sentence structures. The symbolic-calculation module has the features of low recall and high precision and is used to process some numerical type statements (the verifying process involves numerical operations) since we find that the deep-learning-based system is unstable when processing such type of statements. The flowchart of the whole system is displayed in Figure 1.

3.1 Statement-table joint encoder

In both stages of our system, multiple binary classifiers are ensembled together to verify statements. All of the binary classifiers used in the system are constructed based on the table-based pre-trained model GraPPa. In the system, the GraPPa model works in two way: 1) encoding the statement and the pruned table by following the table-BERT method and the table pruning algorithm proposed in Chen et al. (2019); 2) encoding the statement and each row of the table separately by the same method. While the first way is the standard encoding method of NLI tasks, we found the second way also has its advantages and will explain later. After encoding the table and statement by GraPPa, multiple networks are devised to do reasoning based on the encoded representations. The following sections provide further details of these models, named pruned-table-based models and whole-table-based models separately.

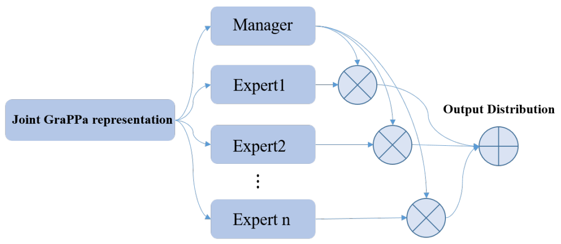


Figure 2: The structure of verifying model is based on the types of statements. An MLP manager activated by the softmax function takes the joint representation as input and outputs the probability distribution that the statement belongs to different types of statements. Multiple MLP experts process the joint representation and output the classification logits, which are further averaged by the statement type probability from the manager to achieve the final classification logits.

3.2 Pruned-table-based models

As introduced before, we simply substitute BERT by GraPPa in the Table-BERT method to get the joint representation of the statement and table. Multiple networks are devised to process the joint representation: 1) a simple linear layer, same as Table-BERT; 2) three MLP processing different types of statements (statements that need to count rows/columns, statements related with the superlative operation, other statements) with a softmax activated MLP manager outputs the probability that the statement belongs to each type of statements. The final output distribution is the weighted average of different MLP experts' output distribution; 3) same structure as 2), but the statements are separated based on the language style (natural/artificial) and the MLP manager is trained based on the natural statement ids provided in the train set; 4) sparsely MoE layer introduced in Shazeer et al. (2017) with applying the implementation on `https://github.com/davidmrau/mixture-of-experts`. Figure 2 shows the basic model structure of networks 2) and 3).

3.3 Whole-table-based models

As serialized tables obtained by language template are usually too long to be encoded by transformer-based model, the above methods use a table pruning algorithm to choose statement-related columns from the table, while the algorithm is not perfect and it may discard some useful columns in tables. Even if the pruning algorithm chooses all the related columns, the pruned serialized table may still be too long. The whole-table-based models regard the table as the set of rows and serialize each row by the same method. Then each row is concatenated with the statement and sent in GraPPa to get the representation of tokens in the statement and each row. Two different reasoning networks are applied after the encoding process. The first one is a simple attention network across all tokens in the statement and table, the final output distribution is produced by the following:

$$Q = RR^T, Q' = softmax(Q), F = Q'R \quad (1)$$

$$\mathbf{att} = RW_a + \mathbf{b} \quad (2)$$

$$\mathbf{out} = MLP(F^T\mathbf{att}) \quad (3)$$

where $R \in \mathbb{R}^{n \times e}$ is the representation matrix of all the tokens in the table and statements, n is the product of row numbers and the maximum token numbers of the concatenation between the statement and different rows, e is the embedding size. Softmax is applied to the attention matrix, where Q'_{ij} represents the j^{th} token's contribution to the i^{th} token. F is the attended representation matrix, and **att** is the importance score of tokens in statement and table achieved by a linear layer where $W_a \in \mathbb{R}^{e \times 1}$ and $\mathbf{b} \in \mathbb{R}^{e \times 1}$ are trainable parameters. Follow the equation 3, the representations are further aggregated by the importance score and sent into an MLP classifier to get the output distribution.

Another choice of reasoning network is the GAT network proposed in Liu et al. (2019) to aggregate evidence from different sources. GAT hierarchically aggregates information (first in token-level and then in sentence-level) which is more reasonable compared to the simple attention network mentioned before. For the convenience of presentation, all the models mentioned in this section are named and listed in Table 1.

3.4 Adaptive model ensembling

An adaptive model ensembling technique is applied to both stages of our verifying system, where

Model	Description
Table-GraPPa	GraPPa with a linear layer
NumGuide	GraPPa with three MLP experts for verifying different numerical type statements
StmtGuide	GraPPa with two MLP experts for verifying statements with different language styles
MoE	GraPPa with sparsely MoE layer (8 MLP experts)
RowAtt	GraPPa with attention layer across all tokens
RowGAT	GraPPa with GAT aggregation module

Table 1: Name and description of models used in the system

different weights of models are searched in different subsets of the development set. Such ensembling technique works well in our system since models work differently on different types of data (e.g. A model may perform better on shorter statements while B model may perform better on longer statements). In the first stage, the weight of each model is searched on two subsets of development set based on the length of the statement and serialized table (we found that some models have a better performance on longer input sequences than others, and set 300 words as the threshold).
In the second stage, the type of each statement is first recognized by some language features and then the model weights are searched on each subset of the development set. Three types of statements are defined: 1) “count” type statements, in which the model needs to count the specific rows or columns in the table; 2) ”superlative” type statements, in which the model need to find the maximum or the minimum value of one row/column; 3) “same/different” type statements, which the model need to decide whether some cells’ value are the same or different. The first two types of statements are recognized by statement slots, while the third type is recognized by trigger words (“same”, “different”, “equal”). We apply the weights searched from the development set on these three statement types and simply do averaging ensemble to the rest of the statements. The ensembling weights are searched on the development set without the participation of models trained with train and development set, while we replaced some models with the version that retrained on both train and development set in the evaluation period and direct applied the weights searched on the development set. More details of the models involved in ensembling are presented in the appendix.

Type	Statement slot
Count	there be () value(s) in the table
Count	there be () different ()
Superlative	() lowest () in the table be ()
Superlative	() has highest value(s) of ()

Table 2: Examples of the statement slots used in the system

3.5 Statement-slot based symbolic-calculation module

To further increase the performance and the stability of the system, a statement-slot-based symbolic-calculation module is developed to solve some numerical type statements which are relatively difficult to deep models. The symbolic-calculation module is added before the two-stage system with a low recall and high precision. Several statement slots are devised to capture “count” and “superlative” type statements. After the capture, the symbolic-calculation module parses the table and extracts the corresponding rows/columns based on a designed entity linking algorithm, and then do the related logical calculation based on the category of the statement and the parsing result of the table. Table 2 shows some examples of the statement slots used in our system.

3.6 Training method

In the first stage, the unknown type training data are created from three source:1) from other tables’ statements under the same XML file, 1685 statements in total; 2) automatic generated statements by language templates, 1378 statements in total; 3) the statements from the related scientific papers (extracted from the related papers of training set tables, near the references of the tables by programming), 1272 statements in total. All of the statements in the train set are used as the known type data to train models. Training processes are stopped when models reach the highest accuracy on the development set.

In the second stage, we first use the train set to train models and stop the training process at the maximum development set accuracy checkpoint and the minimum development set loss checkpoint. Then we add the development set into the train set and retrain some of the models, stopping at a fixed epoch to make the most use of data.

Almost all of the models are trained with the cross-entropy loss function except the NumGuide, StmtGuide, and MoE. For the NumGuide, we use trigger words to recognize the "counting" and "superlative" types statements in the train set and use the recognization result as labels to calculate the cross-entropy between the manager's output and labels. For StmtGuide, we used the provided natural statement ids in the train set to calculate the cross-entropy between the manager's output and labels. For MoE, an extra loss mentioned in Shazeer et al. (2017) is applied to avoid the local optimum. The extra loss functions mentioned above are simply added on the origin cross-entropy loss with a weight of 0.01. Models in the second stage are pre-trained on the TABFACT dataset before further trained on the competition dataset. More details about model training can be found in the appendix.

4 Evaluation

To evaluate our proposed verifying system, we perform two ablation studies regarding the adaptive ensembling method and the statement-slot-based symbolic-calculation module used in our system on both the development set and test set.

Table 3 shows the evaluation of our system on both development set and test set, compared with different ways of ensembling. We found that the adaptive ensembling on both stages achieves the highest two-way and three-way scores on the development set, while it only improves the three-way classification performance on the test set and slightly regresses on the two-way classification. We assume that it may because of the difference between the data distributions of the development set and the test set.

Besides the ablation experiment on ensembling, we also perform an ablation experiment on the statement-slot-based symbolic-calculation module. Table 4 shows the result of the ablation experiment. The result of the experiment shows that the proposed symbolic-calculation module increases the performance on both the development set and test set by around 5 percent, which shows the benefit of the symbolic-calculation module on verifying numerical type statements.

Ensembling type	Dev set		Test set	
	2-way	3-way	2-way	3-way
W+W	**87.81**	**86.93**	84.55	**83.76**
W+A	86.77	86.09	84.92	82.45
A+W	86.50	84.31	**85.41**	81.58
A+A	85.46	83.47	85.22	81.41

Table 3: Ablation experiment regarding the ensembling in the two stages of the system, where **W** refers to the adaptive ensembling and **A** refers to the simple averaging ensembling. The sequence of the letter refers to the ensembling setting of two stages, e.g. W+A means the first stage applies adaptive ensembling and the second stage applies average ensembling.

Ensembling type	Dev set		Test set	
	2-way	3-way	2-way	3-way
w/ sym-cal	**87.81**	**86.93**	**84.55**	**83.76**
w/o sym-cal	82.31	82.36	78.94	77.79

Table 4: Ablation experiment regarding the symbolic-calculation module (sym-cal). The result of the experiment shows that the proposed symbolic-calculation module consistently improves the performance on both the development set and the test set.

5 Conclusion

This paper describes our two-stage verifying system developed for SemEval-2021 Task 9, which leverages the latest table-based pre-trained model GraPPa. The two-stage verifying structure allows us to develop more targeted models on both stages of the system. Multiple reasoning networks are applied behind the GraPPa model, and an adaptive model ensembling technique is used in both stages of the system. A statement-slot-based symbolic-calculation module is also added at the top of the whole system to further improves the performance and stability of the system. Ablation experiments show the effectiveness of the methods proposed in the paper.

References

Wenhu Chen, Hongmin Wang, Jianshu Chen, Yunkai Zhang, Hong Wang, Shiyang Li, Xiyou Zhou, and

William Yang Wang. 2019. Tabfact: A large-scale dataset for table-based fact verification. *arXiv preprint arXiv:1909.02164*.

Zhenghao Liu, Chenyan Xiong, Maosong Sun, and Zhiyuan Liu. 2019. Fine-grained fact verification with kernel graph attention network. *arXiv preprint arXiv:1910.09796*.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*.

Nancy Xin Ru Wang, Diwakar Mahajan, Marina Danilevsky, and Sara Rosenthal. 2021. SemEval-2021 Task 9: A fact verification and evidence finding dataset for tabular data in scientific documents (SEM-TAB-FACTS). In *Proceedings of SemEval*.

Tao Yu, Chien-Sheng Wu, Xi Victoria Lin, Bailin Wang, Yi Chern Tan, Xinyi Yang, Dragomir Radev, Richard Socher, and Caiming Xiong. 2020. Grappa: Grammar-augmented pre-training for table semantic parsing. *arXiv preprint arXiv:2009.13845*.

A Weight searching in adaptive ensembling

We use a grid search method to search the weight of each model in adaptive ensembling. The searching algorithm can be expressed as follows:

1. Initialize total weight T=15 (larger T may cause overfitting on the dev set and larger searching cost as well).

2. Generate all possible weight combinations L, where every element $\mathbf{W}$ in L needs to satisfy the constraint: $\sum_{i=1}^{n} W_i = T$ where n is the number of models joining in the ensembling.

3. For each $\mathbf{W}$ in L, apply the normalized weights $\mathbf{W}/T$ on models' results on the dev set to get the accuracy of ensembling. Save the set of weights with the highest accuracy as the best weights. If multiple sets of weights achieve the same best accuracy, calculate the variance of each set of weights and choose the set with the lowest variance as the best weights.

B Models involved in adaptive ensembling

The following sections introduce the details of models involved in the two-stage ensembling. We revised the entity linking algorithm proposed in Chen et al. (2019) to get better-pruned tables. The new algorithm works better on some models, while we found some models with the origin entity linking algorithm also perform well and we keep them as a part of ensembling. In the following description, the model applies the revised entity linking algorithm if not mentioned specially.
For the simplicity of presentation, we define the following usage of symbols: a "+" symbol added behind the name of the model means the model is trained on train set and stopped when reaching the maximum accuracy on dev set; a "-" symbol added behind the name of the model means the model is trained on train set and stopped when reaching the minimum loss on dev set; no symbol added behind the name of the model means the model is trained on both train and dev set and stopped at fixed epochs.

B.1 Models involved in the first stage's ensembling

A total of 8 models participate in the first stage's ensembling, while some models are trained by biased CE loss (we adjusted the weight of different classes with 1.5:1 for unknown and other classes) to reach a more balance recall and precision. They are: 1) Table-GraPPa+ ;2) RowGAT+; 3) RowAtt+; 4) MoE+; 5) StmiGuide+; 6) RowAtt+ trained with weight added to the loss function; 7) Table-GraPPa+ trained with weight added to the loss function; 8) RowGAT+ trained with weight added to the loss function.

B.2 Models involved in the second stage's ensembling

A total of 9 models participate in the second stage's ensembling, they are: 1) Table-GraPPa; 2) StmtGuide; 3) Table-GraPPa+ with origin entity linking algorithm; 4) MoE; 5) RowGAT; 6) NumGuide; 7) RowAtt; 8) MoE-; 9) Table-GraPPa with origin entity linking algorithm.
When searching the ensembling weights on the dev set, we simply do the following replacements: change Table-GraPPa with Table-GraPPa+; change StmtGuide with StmtGuide+; change MoE with MoE+; change RowGAT with RowGAT+; change NumGuide with NumGuide+; change RowAtt with RowAtt+; change Table-GraPPa with origin entity linking algorithm with Table-GraPPa- with origin entity linking algorithm. Ensembling weights are searched on these models without training on the dev set and directly apply to the corresponding model trained on both train and dev sets.

C More details about model training

All models are trained with AdamW optimizer (implemented by Huggingface) with a warmup ratio be 0.3 and learning rate be 2e-5, and all models in the second stage are pre-trained first on the TABFACT dataset.