Table-based Fact Verification with Self-adaptive Mixture of Experts

Anonymous ACL submission

Abstract

The table-based fact verification task has recently gained widespread attention and yet remains to be a very challenging problem. It inherently requires informative reasoning over natural language together with different numerical and logical reasoning on tables (e.g., count, superlative, comparative). In this paper, we present a **S**elf-**a**daptive **M**ixture-**o**f-**E**xperts Network (**SaMoE**), a novel framework built on this fundamental property. Specifically, we have developed a mixture-of-experts neural network to recognize and execute different types of reasoning—the network is composed of multiple experts, each handling a specific part of the semantics for reasoning, whereas a management module is applied to decide the contribution of each expert network to the verification result. A self-adaptive method is developed to teach the management module combining results of different experts more efficiently without external knowledge. The experimental results illustrate that our framework achieves **85.1%** accuracy on the benchmark dataset TABFACT, comparable with the previous state-of-the-art models. We hope our framework can serve as a new baseline for table-based verification. Our code will be available at (URL to be released here).

1 Introduction

Fact Verification, aiming to determine the consistency between a statement and given evidence, has become a crucial part of various applications such as fake news detection, rumor detection (Rashkin et al., 2017; Thorne et al., 2018; Goodrich et al., 2019; Vaibhav et al., 2019; Kryscinski et al., 2020). While most existing research focuses on verification based on unstructured text, a new trend is extending the scope to structured evidence (e.g., tables), which is informative and ubiquitous in our daily lives. Table-based verification is more challenging than unstructured-text-based due to the complexity of the requirements, including sophisticated textual, numerical, and logical reasoning across evidence tables; even for some statements, multiple types of reasoning are indispensable to complete the verification. An example is presented in Figure 1.

rank	player	country	earnings	wins
1	don january	united states	1338791	21
2	miller barber	united states	1166970	18
3	peter thomson	australia	838535	11
4	gene littler	united states	749216	5
5	lee elder	united states	720164	7

Statement The player with rank 5 have least earnings.
Label ENTAILED

Figure 1: An Example of table-based fact verification.

To tackle the challenges above, previous work established two kinds of methods: (1) *program-enhanced methods* (Chen et al., 2020; Zhong et al., 2020; Shi et al., 2020; Yang et al., 2020) and (2) *table-based pre-trained models* (Eisenschlos et al., 2020; Liu et al., 2021). The program-enhanced methods mainly leverage programs generated by the semantic parser. Specifically, statements are parsed into executable programs to extract the logical/numerical semantics, which is further be leveraged together with contextual semantics learned by a language model (e.g., BERT) in inference. However, the semantic parsers that generate semantic-consistent programs must be trained in a weak supervision setting, which brings difficulties in training. Furthermore, generalizing this method to other datasets is almost impossible without the API set modification according to the reasoning requirements on the new datasets.

The table-based pre-trained models leverage elaborate model structure (Herzig et al., 2020) and pre-training tasks (Eisenschlos et al., 2020; Liu et al., 2021) to enhance the reasoning skills on structured data. Nevertheless, two significant shortcomings remain. Firstly, the process is demanding due

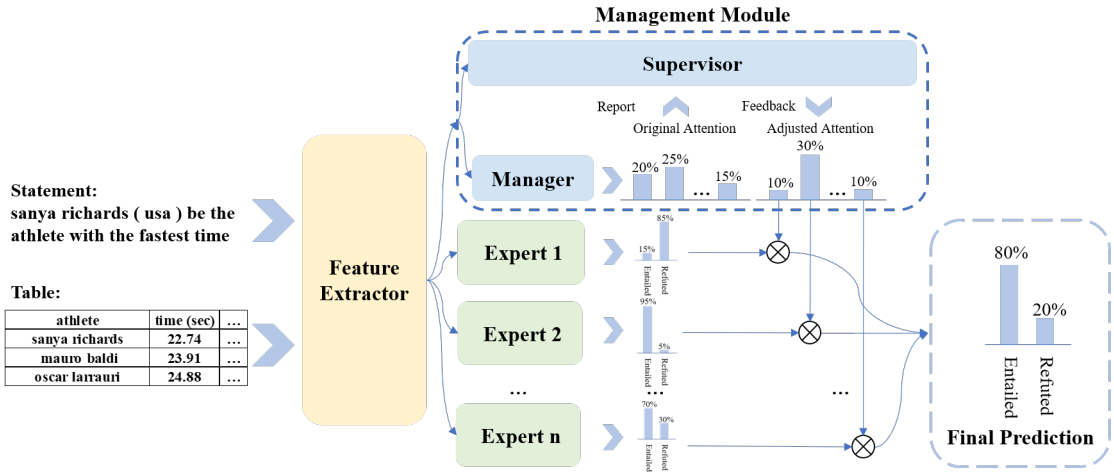


Figure 2: An overview of SaMoE.

to the tremendous computing resources required by pre-training. Moreover, the effectiveness of pre-training to its downstream tasks mainly depends on the adaptability between these two tasks. Therefore, implementing pre-training tasks may fail to meet the requirements when facing the unseen reasoning types demanded by new datasets.

In this paper, we introduce an innovative framework, **S**elf-**a**daptive **M**ixture **o**f **E**xperts (SaMoE), to address the previously mentioned problems. The entire framework is illustrated in Figure 2. SaMoE consists of 3 components: **feature extractor**, **experts**, and **management module**, which is the combination of **manager** and **supervisor** networks. Each expert initially takes the same feature as input and then learns to deal with different parts of the reasoning types (e.g., contextual/logical/numerical) required by table-based verification. A management module is designed to guide the training of experts and combine experts' verification results effectively. The manager network in this module assigns each expert a unique attention score, allowing each individual to focus on different kinds of reasoning types and summarizes experts' entire outputs as the final verification result. However, managers failed to allocate the highest attention score to the expert who performs best on the current reasoning type in most circumstances. To alleviate this problem, we introduce a supervisor network to adjust the attention score given by the manager. The supervisor network is trained **self-adaptively** (i.e., it learns directly from experts' performance on the train set) without prior knowledge of the task or dataset. Extensive experiments are conducted to show that our proposed framework, implemented with a general pre-trained language model RoBERTa (Liu et al., 2019), outperforms previous state-of-the-art methods, including table-based pre-trained models. The main contributions of this work are as follows:

- We innovatively implement mixture-of-experts for table-based verification, aiming to arrange each expert to different types of reasoning. This method can also be easily generalized to other datasets.
- We investigate a self-adaptive method to adjust suitable attention score to each expert according to its performance on different reasoning types, achieving more efficient cooperation across experts.
- Our framework achieves better performance on the TABFACT dataset without the assistance of table-based pre-trained models.

2 Task Formulation

The table-based verification task expects one to determine whether a statement S is entailed or refuted by an evidence table T. The process above can be regarded as a binary classification task and thus denoted as $f(S, T) = \hat{y}$, where f is the verification model and $\hat{y} \in \{0, 1\}$ its prediction.

3 Methods

We present the proposed framework (SaMoE), which leverages a set of experts to deal with different parts of the reasoning types involved in table-based verification. This section is organized as follows. Sec.3.1 introduces the feature extractor that

extracts the joint semantics of the table-statement pair. Sec.3.2 describes experts that verify the statements separately based on the same extracted semantics. Sec.3.3 describes the management module that guides the experts' training and combines their verification results effectively; two components of this module, the manager and the supervisor, are introduced in this section individually.

3.1 Feature Extractor

Feature extractor parses the statement-table pair and learns the joint table-statement semantics. Tables are initially pruned and serialized into a sequence. Subsequently, the serialized tables are transmitted into the language model together with the statements for joint representation learning. These two processes will be further interpreted in the following subsections.

3.1.1 Table Pre-processing

As for Tables, the pre-processing (pruning and serializing) before the joint representation learning provides convenience for subsequent processing of the existing language model.

Table Pruning Table pruning discards some parts of the table that do not participate in the verification, according to the input size limit of the language model. We take advantage of the table-pruning algorithm proposed in Chen et al. (2020) and further enhance its performance. The original algorithm matches the entities in statements with cells in tables by a heuristic method and selects the columns that include matched cells to form the pruned table. Noticed that the algorithm always fails to select the critical columns of verification while there is still room left for the input sequence of the language model, we further add a greedy strategy on the algorithm that keeps adding columns that are not selected to the pruned table until reaching the maximum input size of the downstream model to make the best use of its capacity.

Table Serializing Tables are further serialized to a 1-D sequence after pruning to be compatible with the input format of the language model. We follow the serializing method used in TABLE-BERT (Chen et al., 2020) that paraphrases tables with a natural language template. Specifically, a table with m rows and n columns is paraphrased as "row 1 is: h_1 is T_{11}; ... ; h_n is T_{1n}. row 2 is: ... row m is: h_1 is T_{m1}; ... ; h_n is T_{mn}.", where h_i refers the i^{th} header and T_{ij} the value in the $(i,j)-th$ cell of table T. We find that such template-serialized tables are more suitable for language models pre-trained on unstructured text to process.

3.1.2 Joint Representation Learning

After the table pre-processing, the serialized table and the statement are further passed to a language model to learn the joint contextual representation of each token. The learned representation vectors are then transmitted to the experts and the management module for inference and management. Specifically, the serialized table and the statement are initially tokenized into two token sequences $\tilde{\mathbf{T}}$ and $\mathbf{S}$. Then the joint token sequence $\mathbf{X}$ is formed as $\mathbf{X} = [\langle s\rangle, \mathbf{S}, \langle /s\rangle, \tilde{\mathbf{T}}, \langle /s\rangle]$, where $\langle s\rangle$ and $\langle /s\rangle$ are the separators that identify the beginning and the end of each token sequence. The token sequence $\mathbf{X}$ will be padded or truncated to fit the maximum input length of the language model. Finally, a transformer model is applied to learn the contextual representation of $\mathbf{X}$:

$$\mathbf{H} = f_{LM}(\mathbf{X}) \tag{1}$$

where $\mathbf{H} \in \mathbb{R}^{n\times d}$ refers to the learned joint representation, n is the maximum input length and d the dimension of the representation vector. f_{LM} denotes the contextual representation learning process of the language model. In this paper, we implement it with *transformer* (Vaswani et al., 2017), the most popular contextual representation model in recent years.

3.2 Experts

A group of experts is applied to verify the statements separately based on the same statement-table joint semantics extracted by the feature extractor module. Experts share the same model structure, while the parameter learning strategy of SaMoE gives expert differentiation. Specifically, each expert is implemented with a stack of transformer encoding layers. An MLP classifier that calculates the probability of the statement is entailed by the evidence table based on the encoded semantics. We implement experts with the same general structure rather than different structures specially designed for certain reasoning types since we anticipate that the proposed framework can be smoothly generalized to other datasets. The process above can be formulated as follows:

$$\mathbf{h}_i = f_{Enc_i}(\mathbf{H}) \tag{2}$$

$$\mathbf{p}_i = softmax(tanh(\mathbf{h}_i\mathbf{W}_1^i)\mathbf{W}_2^i) \quad (3)$$

where $\mathbf{h}_i \in \mathbb{R}^d$ is the token $\langle s \rangle$'s final representation vector encoded by the i^{th} expert's encoder Enc_i. It implies the i^{th} expert's whole understanding to the statement-table pair. $\mathbf{W}_1^i \in \mathbb{R}^{d\times d}$ and $\mathbf{W}_2^i \in \mathbb{R}^{d\times 2}$ are the trainable parameters of i^{th} expert's classifier, which projects $\mathbf{h}_i$ to the probabilities $\mathbf{p}_i \in \mathbb{R}^2$ that the statement is entailed/refuted by the table. $tanh$ and $softmax$ are activation functions. n_e refers to the number of experts.

3.3 Management Module

Learning the joint semantics parsed in Sec.3.1, the management module intends to generate attention scores to bias experts' training and combine experts' results efficiently. The module consists of two components: manager and supervisor, both of them are implemented based on *transformer* model. The manager is mainly designed to guide experts' training, while the supervisor is applied to combine experts' results efficiently.

Manager The manager guides the training of experts and forms a preliminary assumption to the expert combination. It encodes the joint representation matrix and generates attention scores $\mathbf{a}_M$ to guide the experts' training process:

$$\mathbf{h}_M = f_{Enc_M}(\mathbf{H}) \quad (4)$$

$$\mathbf{e}_M = tanh(\mathbf{h}_M\mathbf{W}_1^M)\mathbf{W}_2^M \quad (5)$$

$$\mathbf{a}_M = softmax(\mathbf{e}_M) \quad (6)$$

where Enc_M denotes the manager's encoder, $\mathbf{W}_1^M \in \mathbb{R}^{d\times d}$ and $\mathbf{W}_2^M \in \mathbb{R}^{d\times n_e}$ are trainable parameters. The network structures of the manager and experts are basically the same, only different in the layers of the encoder and the output dimension.

After preceding calculation, the normalized attention scores $\mathbf{a}_M$ are used to guide the training of experts by a specially designed verification loss, which will be introduced in Sec.4.1.1.

Supervisor The supervisor adjusts the attention scores submitted by the manager to improve the cooperative efficiency among experts (i.e. assigning higher weights to experts who perform better on the current input pair). The network predicts the deviation between the preliminary assumption (i.e., the attention) and the ideal combination weights based on the knowledge encoded in the joint representation matrix $\mathbf{H}$:

$$\mathbf{h}_S = f_{Enc_S}(\mathbf{H}) \quad (7)$$

$$\mathbf{e}_S = tanh(\mathbf{h}_S\mathbf{W}_1^S)\mathbf{W}_2^S \quad (8)$$

$$\mathbf{a}_S = softmax(\mathbf{e}_M + \mathbf{e}_S) \quad (9)$$

where $\mathbf{W}_1^S \in \mathbb{R}^{d\times d}$, $\mathbf{W}_2^S \in \mathbb{R}^{d\times n_e}$ are trainable parameters and Enc_S refers to the encoder of the supervisor. Parameters of the supervisor are optimized self-adaptively based on experts' performance on the train set. More details of this learning strategy will be presented in Sec.4.2.

4 Parameter Learning

Parameters in SaMoE are learned in two consecutive stages: 1) Supervised learning: parameters in the feature extractor, experts and the manager are end-to-end optimized under the supervision of labels; 2) Self-adaptive learning: parameters in the supervisor are self-optimized by observing experts' performance on the train set (other parameters are fixed simultaneously). A weighted sum of two losses is minimized in the first stage to achieve diverse and balanced training of experts. For the second stage, we minimize a self-adaptive loss calculated based on the experts' classification loss. Subsequent sections introduce these two learning stages in detail.

4.1 Supervised Learning

Supervised learning guides each expert on dealing with different reasoning types while maintaining balanced training across experts. To achieve the goals above, we develop two loss functions: 1) verification loss $\mathcal{L}_V$ that measures the weighted sum of each expert's classification loss, differentiating experts' learning with different attention scores assigned by the manager; 2) manager assumption loss $\mathcal{L}_M$ that is applied to prevent the occurrence of imbalanced training across experts. The overall loss of this state is calculated by a weighted sum of these two terms: $\mathcal{L}_1 = \mathcal{L}_V + \lambda\mathcal{L}_M$, where λ is a hyperparameter that controls the ratio of $\mathcal{L}_M$. Subsequent sections provide detailed introduction to these two terms.

4.1.1 Verification Loss

The verification loss $\mathcal{L}_V$ is designed based on the loss function proposed in Jacobs et al. (1991). It

is calculated by the weighted sum of each expert's cross-entropy:

$$\mathcal{L}_V = \sum_{i=1}^{n_e} (\mathbf{a}_M)_i \cdot CE(\mathbf{p}_i, l) \tag{10}$$

where $(\mathbf{a}_M)_i$ is the i^{th} element of the attention scores $\mathbf{a}_M$, $l \in \{0, 1\}$ is the label of the statement-table pair and $CE(\cdot, \cdot)$ the cross-entropy function. Note that it is necessary to calculate each expert's cross-entropy independently. We want each expert to behave like an independent expert (i.e., complete the verification without the help of other experts). The attention vector $\mathbf{a}_M$ acts as a "training scheduler" in this loss function: experts that are assigned with larger attention scores receive a larger gradient than other experts on the current input, resulting in diverse experts' performance.

4.1.2 Manager Assumption Loss

We have trained the MoE with only the verification loss $\mathcal{L}_V$ and observe a severe "imbalanced experts" phenomenon that only one expert is well-trained (i.e., the expert performance is improved by training) and the manager keeps assigning a close-to-1 attention score to this expert, which is also reported in previous research (Eigen et al., 2013; Shazeer et al., 2017). To avoid this problem, we develop another loss function that forces the manager to assign reasonable attention scores to experts:

$$\mathcal{L}_M = D(\mathbf{a}_P || \mathbf{a}_M) \tag{11}$$

where $D(\cdot||\cdot)$ denotes the Kullback–Leibler divergence and $\mathbf{a}_P$ a prior assumption that is generated with a simple heuristic algorithm (to be introduced in the next paragraph) which requires limited prior knowledge of the reasoning types. By minimizing $\mathcal{L}_M$, the manager learns to assign each expert with a reasonable attention score, resulting in a balanced training across experts.

Prior Assumption Generation The prior assumption $\mathbf{a}_P$ is generated to represent the probabilities that the statement involves different reasoning types that we are interested in. Specifically, we develop a trigger-word-based heuristic algorithm to form the prior assumption for each statement automatically:

1. Initialize the prior assumption with $\mathbf{e}_0 \in \mathbb{R}^{n_e}$, which is empirically set as $(0.1, 0.1, ..., 0.6)^T$. The $(\mathbf{e}_0)_{n_e}$ represents the probability that the statement does not involve any predefined reasoning types and thus is set higher than other values in advance.

2. Traverse the trigger-word set of each reasoning type ($n_e - 1$ types in total). If a trigger word/pattern w that belongs to i^{th} reasoning type is detected in the statement, the trigger's weight s_w (set empirically) is accumulated to the i^{th} dimension of a zero-initialized bias vector $\delta \in \mathbb{R}^{n_e}$: $\delta_i \leftarrow \delta_i + s_w$.

3. Add the bias vector δ to the prior assumption $\mathbf{e}_0$ and normalize to get the prior assumption: $\mathbf{a}_P = softmax(\mathbf{e}_0 + \delta)$.

Figure 3 presents an example of this process. Learning to imitate the prior assumption, the manager guides each expert to focus on different reasoning types and thus achieves diverse experts. We implement a relatively small trigger-word pool in experiments and find the method works effectively, indicating that the method can be smoothly generalized to other datasets with little modification to the predefined reasoning types and trigger-word pool.

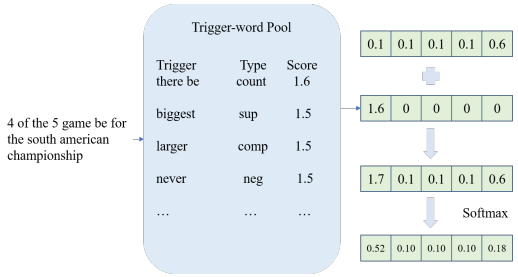


Figure 3: An example of prior assumption generation with $n_e = 5$ and 4 predefined reasoning types.

4.2 Self-adaptive Learning

Self-adaptive learning aims to enhance further the expert combining efficiency with only the knowledge of the expert's performance on the train set. Specifically, an "expert ability" vector $\mathbf{a}_E \in \mathbb{R}^{n_e}$ is calculated based on the "expert loss" vector $\mathbf{m} \in \mathbb{R}^{n_e}$, where $\mathbf{m}_i = CE(\mathbf{p}_i, l)$ is the cross-entropy loss of the i^{th} expert. Note that the cross-entropy of the expert is negatively correlated with its performance. Then the expert ability vector $\mathbf{a}_E$ is calculated as follows:

$$\mathbf{a}_E = softmax(-\alpha \cdot \mathbf{m}) \tag{12}$$

where $\alpha = \sqrt{\beta / Var(\mathbf{m})}$ is a variance-normalizing coefficient and β is a hyperparameter

that decides the variance of the expert ability vector before the activation (i.e., $Var(-\alpha \cdot \mathbf{m}) = \beta$). Such normalization is designed based on the observation that $\mathbf{m}$ tends to have a extreme small variance and $softmax(-\mathbf{m})$ often generates a close-to-uniform distribution. Note that the generated $\mathbf{a}_E$ is positively correlated with the experts' performance (e.g., if the i^{th} expert outperforms the j^{th} expert on the input pair then we have $(\mathbf{a}_E)_i > (\mathbf{a}_E)_j$).

Based on $\mathbf{a}_E$, we develop the loss function that has the same form with $\mathcal{L}_M$ in Sec 4.1.2:

$$\mathcal{L}_S = D(\mathbf{a}_E || \mathbf{a}_S) \tag{13}$$

By minimizing the loss above, the higher attention scores are assigned to the best-performed experts after the supervisor's adjustment, resulting in more efficient cooperation across experts.

5 Experiment Setup

5.1 Data and Metric

We conduct the experiments on TABFACT, a large scale benchmark dataset of the table-based fact verification task[1]. TABFACT contains a total of 117k statements and 16k Wikipedia tables. The test set is further divided into a simple and complex subset based on verification difficulty, for verifying some statements on TABFACT requires more logical/numerical reasoning skills. We choose accuracy as the evaluation metric following the existing work to make our experiment results comparable. More details of TABFACT are presented in Appendix A.

5.2 Implementation Details

Training Details We set $n_e = 5$ expert networks in our implementation of SaMoE. The transformer layers are 12 for encoders in the feature extractor and experts and 2 for encoders in the manager and supervisor. The hidden states' dimension d, the maximum input length n, the λ in Sec.4.1, and the β in Sec.4.2 are set to 1024, 512, 0.1 and 0.1 respectively. We applied RoBERTa-Large (Liu et al., 2019) to initialize the feature extractor and experts in our framework. The details of parameter initialization can be found in Appendix B.

We apply Adam optimizer (Kingma and Ba, 2015) in training with learning rate 2e-5, dropout rate 0.1, warmup step 17,304, and batch size 32. SaMoE is first trained in the supervised learning stage for 57,680 steps (20 epochs). Then the supervisor is trained in the self-adaptive learning stage for another 5,000 steps, while the best parameters of other parts in the framework are loaded and fixed. The model is evaluated every 1000 steps, and the model that achieves the highest performance on the development set is saved. All the codes are implemented with Pytorch (Paszke et al., 2019) and the transformers package (Wolf et al., 2020).

Settings of Prior Assumption We choose the top 4 types of reasoning types that appear most frequently in TABFACT[2] (count, comparative, superlative, negation). We apply a small trigger-word pool containing only 26 trigger words, injecting limited prior knowledge of the dataset. More details of this part are presented in Appendix C.

5.3 Baselines

We compared our proposed framework with different kinds of baselines on TABFACT: (1) Program-enhanced methods: LPA (Chen et al., 2020), LogicalFactChecker (Zhong et al., 2020), HeterTFV (Shi et al., 2020), ProgVGAT (Yang et al., 2020) and Decomp (Yang and Zhu, 2021); (2) Table-based pre-trained models: TAPAS (Eisenschlos et al., 2020) and TAPEX (Liu et al., 2021); (3) Other methods: Table-BERT (Chen et al., 2020) and SAT (Zhang et al., 2020).

6 Results

6.1 Overall Performance

We compare the proposed SaMoE with different kinds of baselines, and the results are listed in Table 1. Baselines are presented with the best performance reported in the corresponding papers. SaMoE obtains an accuracy of **85.1%** on the test set, achieving a new state-of-the-art on the dataset. Results show that our method consistently outperforms all the program-enhanced methods with a significant **2.4%** improvement compared with the Decomp method (the best performed program-enhanced method). Note that SaMoE performs similar with Decomp-LARGE on the simple subset of the test set (93.6% vs. 93.6%) while outperforms Decomp-LARGE with a remarkable **3.5%** on the complex subset (80.9% vs. 77.4%). Such analysis indicates that the performance improvement is mainly derived from successfully verifying

[1] We did not conduct experiments on other datasets such as SEM-TAB-FACTS (Wang et al., 2021) and InfoTabs (Gupta et al., 2020), since there is little work and comparisons have been made on these datasets.

[2] We follow the statistics in Chen et al. (2020) for the frequency of different reasoning types.

Model	Val	Test	Test$_{simple}$	Test$_{complex}$	Small Test
TABLE-BERT	66.1	65.1	79.1	58.2	68.1
LPA	65.1	65.3	78.7	58.5	68.9
LogicalFactChecker	71.8	71.7	85.4	65.1	74.3
HeterTFV	72.5	72.3	85.9	65.7	74.2
SAT	73.3	73.2	85.5	67.2	-
ProgVGAT	74.9	74.4	88.3	67.6	76.2
Decomp-LARGE	82.7	82.7	93.6	77.4	84.7
TAPAS-LARGE	81.5	81.2	93.0	75.5	84.1
TAPEX	**84.6**	84.2	**93.9**	79.6	85.9
SaMoE	84.2	**85.1**	93.6	**80.9**	**86.7**
Human Performance	-	-	-	-	92.1

Table 1: Performance (accuracy) of models and human on TABFACT.

complex statements, which required more sophisticated reasoning than statements in the simple set. SaMoE even shows comparable performance with the previous SOTA TAPEX that is pre-trained to execute SQL queries on tables. Our method outperforms TAPEX with a 0.9% improvement on the test set and a further 1.3% improvement on the complex subset, indicating that SaMoE, based on a **text-based** pre-trained model, performs even better than **table-based** pre-trained models on a variety of complex reasoning types demanded by the table-based verification.

Model	Val	Test
SaMoE	**84.2**	**85.1**
SaMoE w/o Sa	84.0	84.7
SaMoE w/o Sa ($n_e = 1$)	83.6	84.0

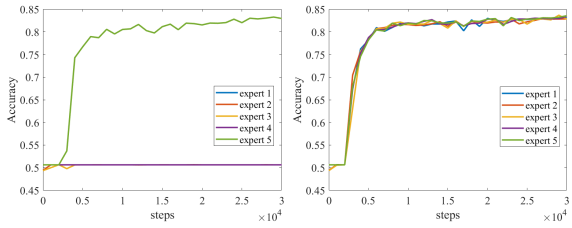
Table 2: Ablation results that shows the effectiveness of the proposed MoE and self-adaptive learning methods. SaMoE w/o Sa denotes that the framework without self-adaptive learning, and $n_e = 1$ denotes that the framework involves only one expert, where the management module does not work in this situation.

6.2 Ablation Study

We further investigate the effectiveness of the MoE structure and self-adaptive learning with an ablation study. We conduct two experiments: one reduces the number of experts to 1 to disable the contribution from the MoE structure (SaMoE w/o Sa ($n_e = 1$)); the other trains the proposed framework with only the supervised learning stage (SaMoE w/o Sa). Results are presented in Table 2. The MoE structure achieves a 0.7% improvement on the test set (84.7% vs. 84.0%), and self-adaptive learning further improves the performance slightly (85.1% vs. 84.7%). Note that the slight improvement of self-adaptive learning is expected since the experts and the feature extractor are fixed in this stage. The results demonstrate the effectiveness of both the MoE structure and the self-adaptive learning.

6.3 Effectiveness Analysis

We show in this section that the effectiveness of the proposed framework is derived from two aspects: the **differentiation of experts** (each expert outperforms others on a specific part of reasoning types) and the **effective attention assignment** by the management module (the best-performed experts are assigned with higher attention scores).



(a) Trained with $\mathcal{L}_V$ (b) Trained with $\mathcal{L}_V + \mathcal{L}_M$

Figure 4: Comparison of models trained with/without the manager assumption loss $\mathcal{L}_M$.

6.3.1 Expert Differentiation

We first investigate the proposed manager assumption loss $\mathcal{L}_M$ and find that it achieves balanced training across experts, which is the premise of expert differentiation. Figure 4 compares the two models trained with and without $\mathcal{L}_M$, with the performance curves of different experts on the development set presented in each sub-figure. Once $\mathcal{L}_M$ is applied, four experts that fail to be trained (the

performance stays around 50% as training steps increase) achieve comparable performance with the rest expert (expert 5 in sub-figure (a)). The result indicates that the proposed $\mathcal{L}_M$ leads balanced training across experts.

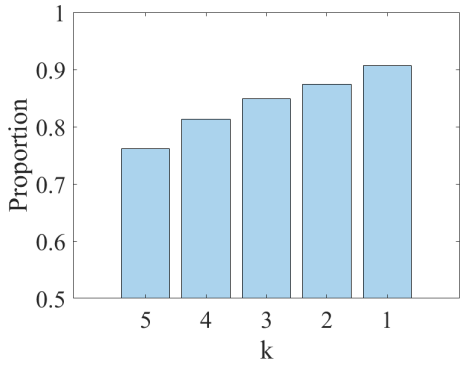


Figure 5: The proportion of statements in the test set that at least k experts verify them correctly ($k \in [1, 5]$).

We further show that the proposed framework achieves differentiation across experts. Figure 5 presents the proportion of statements in the test set that are verified correctly by at least k experts (k varies from 1 to 5). Note that the proportion increases rapidly as k decreases (76.2% to 90.7% for k from 5 to 1), which illustrates that experts behave differently on a large proportion of statements. The results indicate that SaMoE successfully achieves expert differentiation, which expands the original performance upper bound considerably **(90.7%)**.

Model	Accuracy		
	Top 1	Top 2	Top 3
SaMoE	**32.0**	**59.0**	**76.0**
SaMoE w/o Sa	25.4	44.8	67.6

Table 3: The top-k accuracy of the management module that predicts the best-performed experts on the test set.

6.3.2 Effective Attention Assignment

We conduct a detailed analysis to investigate whether the management module assigns higher attention scores to experts with the best performance after self-adaptive learning. To achieve this goal, we regard the management module as a n_e-class classifier and calculate the top-k accuracy of predicting the best-performed expert (the one with the smallest cross-entropy) on the test set where k is chosen in [1, 2, 3]. The results of the analysis are presented in Table 3. The top-k accuracy is improved significantly after self-adaptive learning (**+6.6%, +14.2%, +8.4%** respectively), indicating that the management module successfully assigns higher attention scores to the best-performed experts by self-adaptive learning.

Based on the significant performance upper bound expanded by the expert differentiation, the effective attention assignment achieves more efficient cooperation across these diverse experts, thus improving the verification performance.

7 Related Works

Table-Based Fact Verification Most of the current models utilize programs to improve the model's ability to handle various types of numerical and logical reasoning (Chen et al., 2020; Zhong et al., 2020; Shi et al., 2020; Yang et al., 2020; Yang and Zhu, 2021), while Eisenschlos et al. (2020); Liu et al. (2021) leverage table-based pre-trained models to parse the structural and numerical semantics of tables better. Unlike previous works, we use a novel mixture-of-experts framework to handle different logical and numerical semantics without semantic parsing and table-based pre-training.

Mixture of Experts Mixture of experts is a special model combining method. Jacobs et al. (1991) first introduces this method and proposes a loss that encourages competitive learning across expert models. We develop a self-adapted mixture-of-experts framework that achieves a more effective combination of experts by learning from the experts' performance on the train set.

8 Conclusion

This paper proposes a framework that leverages the mixture of experts to recognize and execute different types of reasoning required for table-based fact verification. We propose an MoE model guided with limited prior knowledge to handle different parts of the reasoning types required by table-based verification with diverse experts. Moreover, we design a novel supervisor network to adjust the imprecise attention score and achieve a more efficient combination across experts. A self-adaptive learning strategy is further applied to train the proposed supervisor network without prior knowledge of the task or dataset. The experiments show that the proposed model achieves a new state-of-the-art performance of 85.1% accuracy on the benchmark dataset TABFACT. The ablation studies and analysis further indicate the effectiveness of the proposed MoE structure and self-adaptive learning strategy.

References

Wenhu Chen, Hongmin Wang, Jianshu Chen, Yunkai Zhang, Hong Wang, Shiyang Li, Xiyou Zhou, and William Yang Wang. 2020. Tabfact: A large-scale dataset for table-based fact verification. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

David Eigen, Marc'Aurelio Ranzato, and Ilya Sutskever. 2013. Learning factored representations in a deep mixture of experts. *arXiv preprint arXiv:1312.4314*.

Julian Eisenschlos, Syrine Krichene, and Thomas Müller. 2020. Understanding tables with intermediate pre-training. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 281–296, Online. Association for Computational Linguistics.

Ben Goodrich, Vinay Rao, Peter J Liu, and Mohammad Saleh. 2019. Assessing the factual accuracy of generated text. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 166–175.

Vivek Gupta, Maitrey Mehta, Pegah Nokhiz, and Vivek Srikumar. 2020. INFOTABS: Inference on tables as semi-structured data. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2309–2324, Online. Association for Computational Linguistics.

Jonathan Herzig, Pawel Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Eisenschlos. 2020. TaPas: Weakly supervised table parsing via pre-training. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4320–4333, Online. Association for Computational Linguistics.

Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. 1991. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Wojciech Kryscinski, Bryan McCann, Caiming Xiong, and Richard Socher. 2020. Evaluating the factual consistency of abstractive text summarization. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9332–9346, Online. Association for Computational Linguistics.

Qian Liu, Bei Chen, Jiaqi Guo, Zeqi Lin, and Jianguang Lou. 2021. Tapex: Table pre-training via learning a neural sql executor. *arXiv preprint arXiv:2107.07653*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 8024–8035.

Hannah Rashkin, Eunsol Choi, Jin Yea Jang, Svitlana Volkova, and Yejin Choi. 2017. Truth of varying shades: Analyzing language in fake news and political fact-checking. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2931–2937, Copenhagen, Denmark. Association for Computational Linguistics.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc V. Le, Geoffrey E. Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Qi Shi, Yu Zhang, Qingyu Yin, and Ting Liu. 2020. Learn to combine linguistic and symbolic information for table-based fact verification. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 5335–5346, Barcelona, Spain (Online). International Committee on Computational Linguistics.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. FEVER: a large-scale dataset for fact extraction and VERification. In *NAACL-HLT*.

Vaibhav Vaibhav, Raghuram Mandyam, and Eduard Hovy. 2019. Do sentence interactions matter? leveraging sentence level representations for fake news classification. In *Proceedings of the Thirteenth Workshop on Graph-Based Methods for Natural Language Processing (TextGraphs-13)*, pages 134–139, Hong Kong. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Nancy X. R. Wang, Diwakar Mahajan, Marina Danilevsky, and Sara Rosenthal. 2021. SemEval-2021 task 9: Fact verification and evidence finding for tabular data in scientific documents (SEM-TAB-FACTS). In *Proceedings of the 15th International Workshop on Semantic Evaluation (SemEval-2021)*, pages 317–326, Online. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Xiaoyu Yang, Feng Nie, Yufei Feng, Quan Liu, Zhigang Chen, and Xiaodan Zhu. 2020. Program enhanced fact verification with verbalization and graph attention network. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7810–7825, Online. Association for Computational Linguistics.

Xiaoyu Yang and Xiaodan Zhu. 2021. Exploring decomposition for table-based fact verification. *arXiv preprint arXiv:2109.11020*.

Hongzhi Zhang, Yingyao Wang, Sirui Wang, Xuezhi Cao, Fuzheng Zhang, and Zhongyuan Wang. 2020. Table fact verification with structure-aware transformer. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1624–1629, Online. Association for Computational Linguistics.

Wanjun Zhong, Duyu Tang, Zhangyin Feng, Nan Duan, Ming Zhou, Ming Gong, Linjun Shou, Daxin Jiang, Jiahai Wang, and Jian Yin. 2020. LogicalFactChecker: Leveraging logical operations for fact checking with graph module network. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6053–6065, Online. Association for Computational Linguistics.

A Statistics of TABFACT

Table 4 shows the basic statistics of TABFACT. As the table shows, the whole dataset is randomly divided into three subsets with the ratio be 8:1:1. The average numbers of rows and columns in tables keep approximately the same across three subsets, which reflects the consistency of data distribution.

Split	#Sentence	#Table	Avg.row	Avg.col
Train	92,283	13,182	14.1	5.5
Dev	12,792	1,696	14.0	5.4
Test	12,779	1,695	14.2	5.4

Table 4: Statistics of TABFACT, including the number of statements, tables, and the average number of rows and columns in tables.

B Parameter Initialization

For parameter initialization, We leverage RoBERTa-Large, a pre-trained language model that has 24 transformer encoding layers. We initial parameters of the feature extractor with the embedding layer and the bottom 12 encoding layers of RoBERTa-Large and each expert with the upper 12 encoding layers of RoBERTa-Large, respectively. We use PyTorch to initialize other parameters randomly.

C Specific Setting of Prior Assumption Generation

We choose four reasoning types that appear most frequently in TABFACT: count, comparative, superlative, and negation. The detailed definitions of four reasoning types chosen in our implementation are listed below:

1. Count: counting the number of specific rows in the table, such as “xxx be listed a total of 3 times", “xxx win only 1 time in ...", etc.

2. Comparative: comparing two values in the statement or cells, such as “xxx play in more than 1 game during ...", “xxx has a larger yyy than zzz", etc.

3. Superlative: finding the highest/lowest value of the specific column, such as “the longest xxx be yyy", “the lowest score at xxx be yyy", etc.

4. Negation: negating the original semantics of the statement, such as “xxx has never lost a game in ...", “xxx never score 0 points", etc.

Type	Trigger	Weight
Count	only+[number]	1.6
Count	[number]+times	2
Count	[number]+of	1.6
Count	there be+[number]	1.6
Negation	no	1.5
Negation	not	1.5
Negation	never	1.5
Negation	didn’t	1.5
Comparative	[JJS] or [RBS]	1.5
Superlative	[JJR] or [RBR]	1.5

Table 5: Some trigger words/patterns applied in the generation of the prior assumption on TABFACT.

A small trigger-word pool that contains only 26 trigger words/patterns is applied for the prior assumption generation: 11 triggers for the "count" type, 15 for "negation"; and for the rest types (i.e., "comparative" and "superlative" types), the NLTK package is employed to recognize the comparative and superlative words automatically. Such a small trigger-word pool injects limit prior knowledge of the dataset, indicating that the proposed method can be generalized to other datasets by simply modifying the pool of trigger words. Table 5 presents some words/patterns in the trigger-word pool applied in our experiments. x+[number] denotes a combination of a word and a number that is served as a trigger (e.g., for the statement “xxx win 3 times in ...", we match the phrase “3 times" with the trigger “[number]+times").