TFV: A Framework for Table-Based Fact Verification

Mingke Chai, Zihui Gu, Xiaoman Zhao, Ju Fan[*], Xiaoyong Du
Renmin University of China
No. 59 Zhongguancun Street, Beijing 100872, China
{cmk,zihuig,xiaomanzhao,fanj,duyong}@ruc.edu.cn

Abstract

Table-based fact verification, which verifies whether a claim is supported or refuted by structured tables, is an important problem with many downstream applications, like misinformation identification and fake news detection. Most existing works solve the problem using a natural language inference approach, which may not be effective to capture structure of the tables. Despite some recent attempts of modeling table structures, the proposed methods are not compared under the same framework and thus it is hard for practitioners to understand their benefits and limitations. To bridge the gap, in this paper, we introduce a framework `TFV`*, including pre-trained language models, fine-tuning, intermediate pre-training and table serialization techniques. Based on the framework, we define a space of design solutions for each module in* `TFV`*, and conduct an empirical study to explore the design space. Through the experiments, we find that structure information is very crucial but yet under-explored. We point out the limitations of the current solutions and identify future research directions. Moreover, we also develop a python package that implements* `TFV` *and illustrate how it is used for table-based fact verification.*

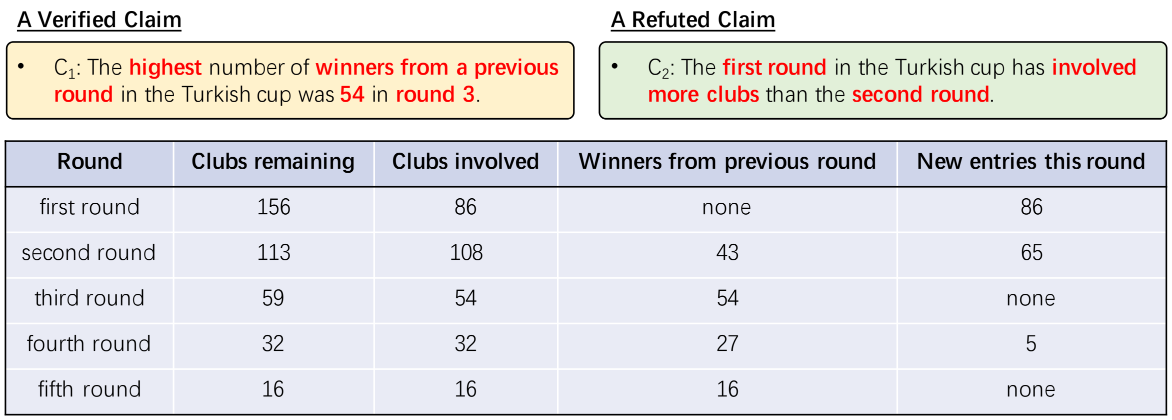
1 Introduction

Fact verification, which examines whether a textual hypothesis (or a *claim*) is supported or refuted by some given evidence, has many downstream applications, such as misinformation identification and fake news detection. As a fundamental task in natural language understanding, fact verification has been extensively studied [2,6,13,22,27], and most of the existing studies focus on *textual evidence*. For example, given a claim "*solar panels drain the sun's energy*", the approaches find relevant articles as evidence to support or refute the claim [22].

Recently, *table-based fact verification* has been introduced and attracted much attention [3,9,14,31]. Different from traditional fact verification over textual evidence, table-based fact verification aims to classify whether a claim is supported or refuted by a given table. Figure 1 illustrates an example about table-based fact verification. Given a table that contains structured facts of the Turkish cup as evidence, we can verify that the claim C_1 is correct by referring to the table. Similarly, it is obvious that claim C_2 can be refuted. Since tabular data are being created and curated at unprecedented pace and are made available in a variety of data sources, table-based fact verification may become more and more feasible in real-world applications.

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

[*]Ju Fan is the corresponding author.



A Verified Claim

- C_1: The highest number of winners from a previous round in the Turkish cup was 54 in round 3.

A Refuted Claim

- C_2: The first round in the Turkish cup has involved more clubs than the second round.

Round	Clubs remaining	Clubs involved	Winners from previous round	New entries this round
first round	156	86	none	86
second round	113	108	43	65
third round	59	54	54	none
fourth round	32	32	27	5
fifth round	16	16	16	none

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

However, compared with fact verification over textual evidence, table-based fact verification is still under-explored. Most recent studies solve the problem using a natural language inference approach [3, 9]. Specifically, the approach first serializes a table into a token sequence and then leverages a pre-trained language model (e.g., BERT [7]) to encode the serialized table and the claim into vectors. One straightforward approach to table serialization is to linearize the table content via horizontal scan, i.e., row-by-row. Unfortunately, the approach may have a limitation because it ignores the inherent *structure* of the tables. For example, the number "54" in Figure 1 is ambiguous if we do not consider the attributes to which it belongs. Similarly, claim C_2 is hard to verify if we ignore the structure of the first and second rows in the table. Although some recent studies consider structure-aware methods [14, 31], as far as we know, the proposed methods were not compared under the same framework. Thus, it is difficult for practitioners to understand the benefits and limitations of these methods.

To address the above problem, this paper introduces a framework TFV that unifies the existing solutions to table-based fact verification. Specially, TFV consists of four main modules: (1) *Table Serialization* converts a table into a sequence of tokens. (2) *Pre-trained Language Model (LM)* encodes the claim and table sequence into vectors. (3) *Intermediate Pre-training* utilizes masked language modeling (MLM) objective [7] to fine-tune the LM based on training dataset. (4) *Fine-tuning for Verification* takes as input the vectors and returns either support or refutation as the output. Based on the framework, we conduct an empirical study to systemically investigate how to effectively encode structure of tables. We review the existing solutions for each module in our framework, and define a design space by categorizing the solutions. Through exploring the design space, we provide experimental findings on modeling structure of tables. Also, we compare TFV with the existing solutions to table-based fact verification, and analyze benefits and limitations of TFV. Note that some previous studies have also investigated some individual modules considered in this paper, e.g., structure-aware pre-trained LMs [14] and fine-tuning methods [31], and they reached consistent conclusions with ours regarding the necessity of modeling structure of tables. However, compared with them, we not only further examine the trade-off between different modules, but also evaluate more modules, such as intermediate pre-training and table serialization.

To summarize, we make the following contributions in the paper.

(1) We conduct a comprehensive experimental study on table-based fact verification, with a special focus on investigating how to encode table structures. We formally define the problem and review existing literature (Section 2). We introduce a general framework TFV, and define a design space by categorizing the existing solutions in each module of the framework (Section 3).

(2) We empirically evaluate the design space in TFV (Section 4). We reveal the insights from our experimental findings to show that modeling table structures can significantly improve the accuracy of fact verification, and also point out research directions for better model design.

(3) We have implemented `TFV` as a python package (Section 5). We publish all the code at Github[1]. Fed with a collection of structured tables, the package is easy-to-use and offers table-based fact verification to users.

2 Table-Based Fact Verification

2.1 Problem Formalization

This paper considers a structured table T with n attributes (i.e., columns), denoted by $\{A_1, A_2, \ldots, A_n\}$, and m tuples (i.e., rows), denoted by $\{t_1, t_2, \ldots, t_m\}$. In particular, we denote the value of the j-th attribute in tuple t_i as t_{ij}. Moreover, we consider a textual hypothesis (or *claim*), denoted by C. The problem of *table-based fact verification* is, given a pair (C, T) of claim C and table T, to determine whether table T can be used as evidence to *support* or *refute* claim C. Note that claim C may be either as simple as describing one tuple in table T, or as complex as aggregating or comparing multiple tuples in T. Figure 1 provides a running example.

2.2 Related Work

Fact Verification. Fact verification over textual evidence has been extensively studied in the last decade. Some early studies consider a premise sentence as evidence to support or refute a claim [2, 6], while some later works focus on collecting relevant passages from Wikipedia as evidence [13, 22, 27]. These studies rely on techniques including logic rules, knowledge bases and neural networks for verifying claims based on given evidence. Recently, large pre-trained language models (LM) [7, 18, 29] have been utilized for fact verification [28]. These models are reported to achieve superior performance due to their self-supervised learning from massive text corpora and effective adapting of the resulting models to target fact verification task.

However, most of the existing studies are limited to considering unstructured text as evidence. As verifying claims over structured tables is useful in many applications [5, 17], table-based fact verification has been introduced very recently [3, 9, 14, 31]. Chen et al. formalize the problem of table-based fact verification and provide a benchmarking dataset called TabFact [3]. Then, more approaches are proposed [9, 14, 31] and the basic idea of these approach is to serialize the table and feed it into a large-scale pre-trained language model. Unfortunately, these approaches may have a limitation that they do not thoroughly study how to effectively encode table structures. Specifically, compared with natural language, tabular data has its unique structural characteristics. For example, for each cell, the cells in the same column or the same row may provide more contextual information than others. In addition, each token in natural language has its inherent syntactic logic while each cell in a table may not. This limitation motivates us to develop our framework `TFV` and conduct an experimental study.

Question Answering over Tables. Another line of research closely related to our work is table-based question answering (QA) [16, 19, 20]. Compared with traditional methods based on logical forms [10, 25, 32], methods that utilizes an end-to-end approach to generate answers directly have been used successfully [14, 19]. TAPAS [14] is a representative end-to-end approach. By extending BERT's architecture and using new pre-training tasks [4], TAPAS [14] improves the understanding of tabular data for various QA tasks. In this paper, we borrow some representative techniques, e.g., TAPAS, to solve our table-based fact verification problem. We has an interesting observation that, although designed for QA over tables, these techniques can also improve the performance of fact verification due to its ability of modeling table structures.

3 Overview of Our `TFV` Framework

Figure 2 shows our general framework `TFV` for table-based fact verification. It takes as input a structured table T and a textual claim C, and predicts whether C can be supported (label 1) or refuted (label 0) by T. To this

[1]https://github.com/zihuig/TFV

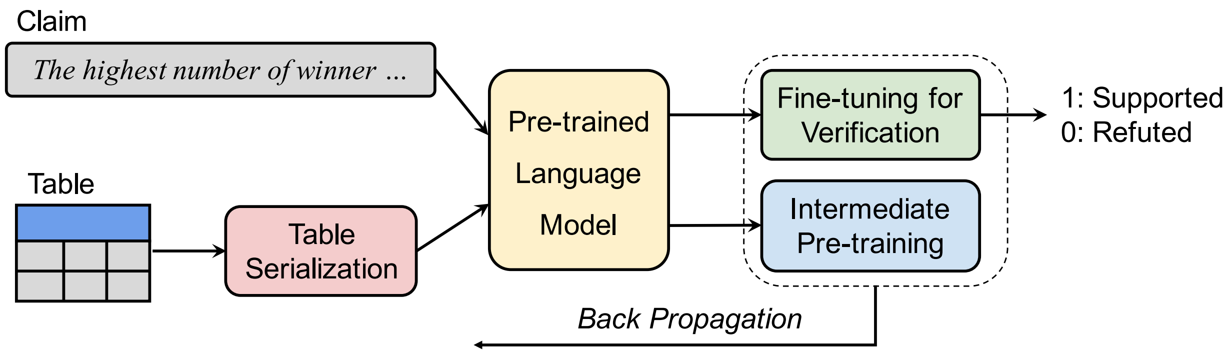

Claim
The highest number of winner ...
Table
Table Serialization
Pre-trained Language Model
Fine-tuning for Verification
Intermediate Pre-training
1: Supported
0: Refuted
Back Propagation

Figure 2: Overview of our TFV framework for table-based fact verification.

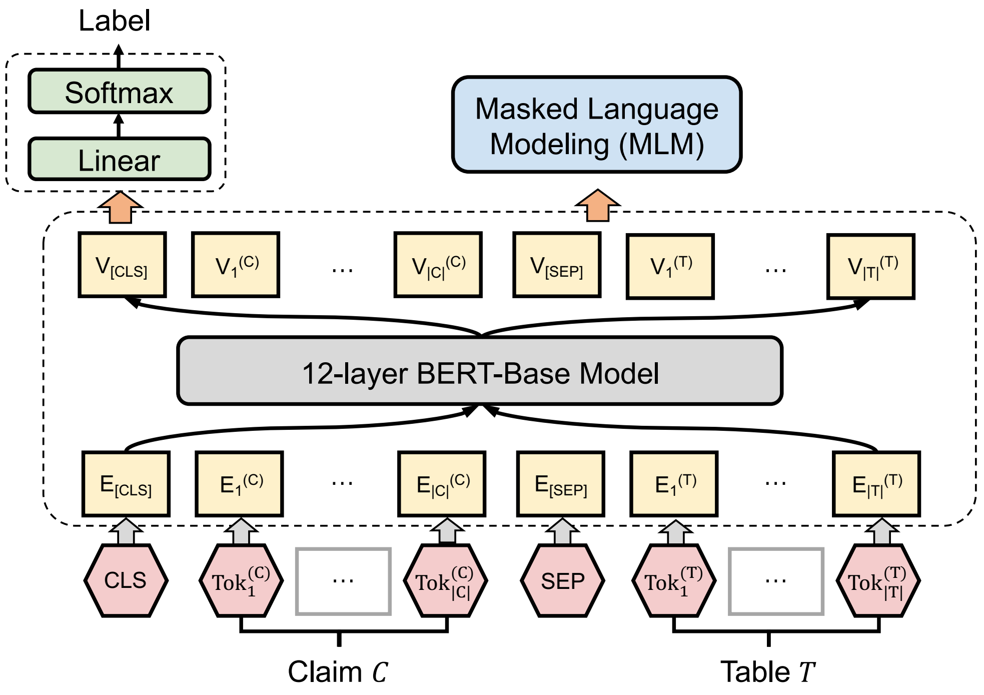


Figure 3: An example of using our TFV framework for table-based fact verification.

end, TFV utilizes the following four main modules. (1) *Table Serialization* converts our structured table T into a sequence of tokens, denoted by $\mathtt{seq}_T$. One straightforward serialization method is to horizontally scan table T and concatenate the rows into a sequence. (2) *Pre-trained Language Model (LM)* takes as input the serialized table $\mathtt{seq}_T$ and token sequence $\mathtt{seq}_C$ of claim C, and produces vector-based representations for the pair (T, C) of table T and claim C. (3) Fed with the vector-based representations, *Fine-Tuning for Verification* produces a predicted verification result on whether C is supported or refuted by T. By considering the classification loss between the predicted result and ground-truth, TFV updates the parameters in the pre-trained LM and the classification model. (4) To further boost the performance, *Intermediate Pre-training* [21] is introduced to utilize training dataset $\{(T, C)\}$ for fine-tuning the pre-trained LM.

Example 1: We use an example to illustrate how TFV works, as shown in Figure 3. We first serialize table T and claim C into a sequence with special tokens, e.g., [CLS] and [SEP], and then we use a pre-trained LM, e.g., BERT [7] to obtain embeddings for all the tokens. Next, we use a fully-connected layer followed by a softmax layer to produce a predicted label, i.e., supported/refuted. Moreover, we also use a masked language modeling

(MLM) task over our training dataset as intermediate pre-training to fine-tune the LM model for improving the token embeddings. Then, we use the LM model after intermediate pre-training to fine-tune for verification.

Next, we present design solutions for each module in our `TFV` framework. Specifically, we focus on how to encode structural information of tables into pre-trained LM (Section 3.1), fine-tuning for verification (Section 3.2), intermediate pre-training (Section 3.3) and table serialization (Section 3.4) respectively.

3.1 Pre-trained Language Models

Pre-trained LMs have been shown to achieve superior performance in many NLP tasks [7, 18, 29] . The basic idea is to first train an LM on a large unlabeled corpus to learn common representations with unsupervised methods, and then fine-tune the model with a small labeled datasets to fit the downstream tasks. Through pre-training, the LM model can gain better initialization parameters, improved generalization capabilities, faster convergence in downstream tasks, and robustness against over-fitting given small datasets. In our framework `TFV`, we consider two solutions for pre-trained LMs: (1) the BERT-based LM, and (2) TAPAS [14], an LM pre-trained over tabular datasets.

BERT-based LM. BERT [7] is currently the most popular pre-trained LM. It uses self-attention mechanisms to learn sequence semantic information. BERT is generally pre-trained on a large text dataset of 3.3 billion words, i.e., a concatenation of the BooksCorpus (800 million words) and the English Wikipedia (2.5 billion words) datasets. It leverages two pre-training tasks, namely masked language modeling (MLM) and next sentence prediction (NSP) [7]. Through fine-tuning on different downstream NLP tasks, BERT has achieved the best results so far on the tasks. In our framework, we implement BERT using PyTorch and use "*bert-base-uncased*" as the default BERT setting[2].

TAPAS-based LM. By extending BERT and using *structure-aware* pre-training tasks [4], TAPAS [14] is introduced by Google to improve the understanding of tabular data. Specifically, TAPAS adds additional table-aware positional embeddings for each token, such as column ID, row ID and rank ID, where column/row ID is the index of the column or row that the corresponding token appears in. If the data type of a column is float or date, TAPAS sorts the column values and assigns the values's rank IDs as their numeric ranks. In addition to the additional embeddings that capture tabular structure, TAPAS is pre-trained by using masked language modeling objective over millions of tables crawled from Wikipedia as well as carefully generated claims that associate with the tables. TAPAS can learn correlations between claims and table, and between cells in tables. In `TFV`, we use the original source code of TAPAS and use "*google/tapas-base*" as the default setting[3].

3.2 Fine-Tuning for Fact Verification

In the fine-tuning step, the LM model is fine-tuned using a labeled dataset denoted as $\{(T, C, l)\}$, where l is the label (supported or refuted). The fine-tuning would make the pre-trained LM more fit the task of fact verification. In our framework `TFV`, we consider two solutions for fine-tuning: (1) A traditional binary classification model, and (2) a structure-aware binary classification model.

Traditional Classification Model. As illustrated in Figure 3, this solution takes the embedding of token [`CLS`] as input, and then utilizes a fully connected layer followed by a softmax layer to produce a predicted label. Given the training dataset, it measures the loss between predicted labels and ground-truth to update parameters in both pre-trained LM and the classification model.

Structure-Aware Classification Model. The limitation of the previous fine-tuning method is that it ignores the structure of tables. To address the limitation, Zhang et al. introduce a structure-aware method called SAT for fine-tuning in fact verification [31]. The basic idea of SAT is illustrated in Figure 4(a). SAT aims to capture the

[2]https://pypi.org/project/pytorch-pretrained-bert/
[3]https://github.com/google-research/tapas

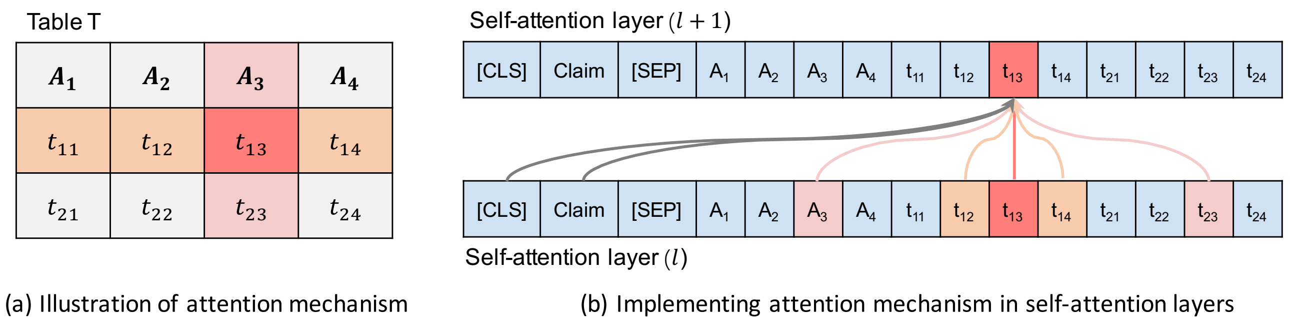


Table T
A_1
A_2
A_3
A_4
t_{11}
t_{12}
t_{13}
t_{14}
t_{21}
t_{22}
t_{23}
t_{24}
Self-attention layer $(l+1)$
[CLS]
Claim
[SEP]
Self-attention layer (l)
(a) Illustration of attention mechanism
(b) Implementing attention mechanism in self-attention layers

Figure 4: An illustration of structure-aware model for fact verification fine-tuning.

structure of table T by using an attention mechanism. For example, given a value t_{13}, SAT aims to pay more attention to attribute name A_3 and values t_{11}, t_{12}, t_{14} and t_{23}, which would have more correlation with t_{13}. To realize this attention mechanism, SAT modifies the attention mask in self-attention layers in our pre-trained LM model when fine-tuning the model, as shown in Figure 4(b). For example, for computing representation of t_{13} in the $(l+1)$-th layer, instead of considering all the tokens in the (l)-the layer, SAT only attends to the correlated tokens and masks out the other ones.

3.3 Intermediate Pre-training

In the pre-training step, our model is pre-trained using large-scale general datasets, which learns common language representations. Then, an intermediate pre-training module [8, 15, 23, 24] can be utilized to learn *task-specific* representation using a fact verification dataset. We extend BERT's masked language modeling (MLM) [7] objective to structured data. As in BERT, the training data generator chooses 15% of the tokens of table sequence and claim sequence at random for prediction. For each chosen token, we replace it with "[MASK]", replace it with a random token or keep it intact, with a 80%, 10% and 10% chance, respectively. Note that, when using SAT for fine-tuning, we also use the modified attention mask mechanism for MLM.

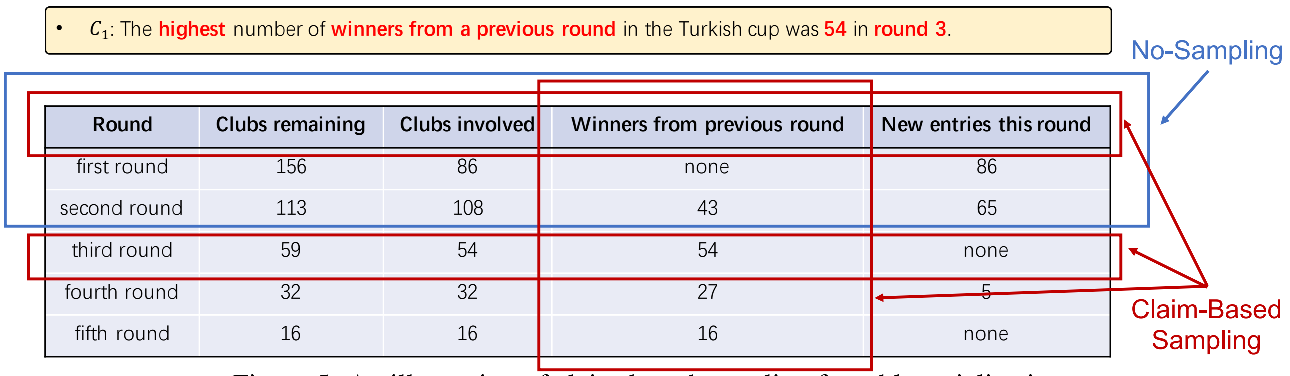
3.4 Table Serialization

Table serialization converts our structured table T into a sequence of tokens denoted by $\texttt{seq}_T$. The key challenge here is that there may be a constraint on the maximum length of the sequence. For example, when using BERT as pre-trained LM, one needs to set a hyper-parameter "`max_sequence_length`", which is often 256. However, our table T contains much more tokens. Therefore, it is non-trivial to select the most *valuable* tokens into sequence $\texttt{seq}_T$. In `TFV`, we consider the following heuristic solutions and will study more sophisticated methods in the future work.

No Sampling first serializes table T via horizontal scan, and, when sequence length constrain is met, neglects all remaining tokens. For example, consider table T in Figure 5: given a sequence length constraint, say 15, this strategy only converts attribute names and the first two tuples into the sequence. Obviously, such information is insufficient for verifying our claim C_1, which may degrade the overall performance.

Claim-based Sampling. To address the limitation of the previous strategy, we introduce a claim-based sampling strategy. The basic idea is to selectively sample tokens from table T by considering their correlations with claim C. Figure 5 shows an example: we sample the tokens in attribute names, the third tuple and the fourth columns and convert these tokens into sequence $\texttt{seq}_T$, as these tokens are more related to claim C_1, and thus would benefit the downstream pre-trained LM and fact verification fine-tuning.

To realize claim-based sampling, `TFV` first identifies the cells in T that have overlapping tokens with claim C. For example, cell t_{34} with value 54 can be selected as 54 also occurs in the claim. Then, `TFV` considers two



• C_1: The highest number of winners from a previous round in the Turkish cup was 54 in round 3.

Round	Clubs remaining	Clubs involved	Winners from previous round	New entries this round
first round	156	86	none	86
second round	113	108	43	65
third round	59	54	54	none
fourth round	32	32	27	5
fifth round	16	16	16	none

Figure 5: An illustration of claim-based sampling for table serialization.

Table 10: Basic statistics of the TabFact dataset.

Type	# Claims	# Tables	Label Ratio
Complex	50,244	9,189	1:1
Simple	68,031	7,392	1:1
All	118,275	16,573	1:1

Table 11: Train/Val/Test splits in TabFact.

Split	# Claims	# Tables	# Rows	# Cols
Train	92,283	13,182	14.1	5.5
Val	12,792	1,696	14.0	5.4
Test	12,779	1,695	14.2	5.4

methods for sampling tokens. (1) Row-based sampling that samples all the rows, in which the identified cells reside, e.g., the attribute row and the 3rd tuple in Figure 5. (2) Cross-based sampling that also considers the columns that contain the identified cells, e.g., the fourth column in Figure 5.

4 Experiments

In this section, we conduct an experimental study to explore the design space in TFV. We present the experiment settings in Section 4.1 and report the main results in Section 4.2. Finally, we summarize the experiment findings and provide insightful takeaways in Section 4.3.

4.1 Experiment Setup

Dataset: We use TabFact [3], the benchmarking dataset for table-based fact verification, for evaluating the design solutions in TFV. This dataset collects tables from Wikipedia and utilizes crowdsourcing to generate claims, where the claims are categorized into *simple claims* and *complex claims*. Specifically, simple claim only describes one tuple in a table, and verifying a simple claim does not involve complex symbolic reasoning [1, 12]. In contrast, complex claims describe multiple tuples in the table, and thus verifying a complex claim needs to consider more complex operations, such as argmax, argmin, count, difference, average, etc. The basic statistics of the TabFact dataset are summarized in Table 10. There are 16573 tables and 118275 claims in the dataset, where each table has on average 14 rows, 5 to 6 columns and 2.1 words in each cell and each table corresponds to 2 to 20 claims. Moreover, the label ratio between supported and refuted claims is 1:1, and the average lengths of positive and negative claims are nearly the same. To evaluate our framework TFV, we divide the dataset into training set, validation set and test set according to the ratio of 8:1:1, where simple and complex claims are stratified in all the three sets. The statistics of these three sets are reported in Table 11.

Evaluation Metric. We use *accuracy*, which is the ratio of correctly predicted supported/refuted labels to all the labels, as the evaluation metric. Specifically, we implement TFV and apply certain design solutions as described above. We utilize the training and validate datasets to train our model for fact verification, and then measure the accuracy of the model on the test dataset.

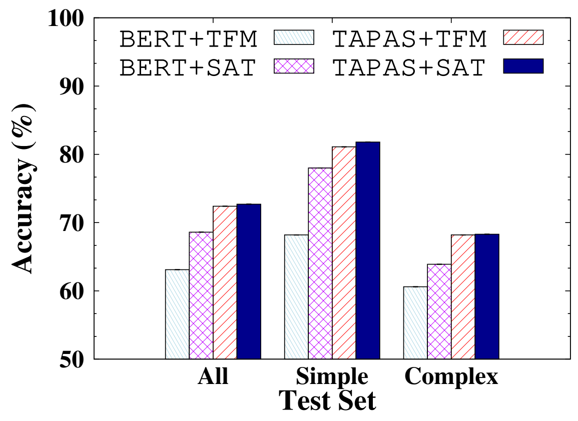


Figure 6: Evaluation of TFV modules.

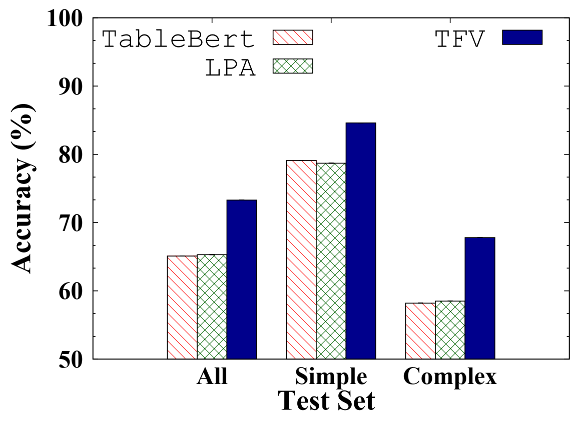


Figure 7: Comparison with existing methods.

Model Parameters: All experiments in this paper use "*bert-base-uncased*" or "*google/tapas-base*" , as described previously, as the pre-trained LM model, where the model has 12 self-attention layers in total. The maximum sequence length of the LM's input, i.e., `max_sequence_length`, is set to 256, and characters exceeding the length will be automatically truncated. Batch size is 32 and the learning rate is 2e-5. We set the maximum number of epochs as 20 and the model usually converges in epochs 15 to 18. The epoch number of every evaluated model is selected as the epoch with the best validation result.

4.2 Experimental Results

We first report our evaluation results on various design solutions in the main modules of `TFV`. Then, we compare `TFV` with existing methods for table-based fact verification.

Evaluation of pre-training and fine-tuning. We first evaluate the design solutions of the two main modules in `TFV`, namely pre-trained LM and fact verification fine-tuning. Figure 6 shows the experiment results, where `TFM` and `SAT` represent the traditional and structure-aware classification models respectively. We have two key observations. First, TAPAS-based LM can significantly improve the accuracy on various types of test sets. For example, given traditional fine-tuning method `TFM`, TAPAS-based LM improves the accuracy by 5.5%, 10% and 3.3% on `All`, `Simple` and `Complex` test sets respectively. This is attributed to the structure-aware pre-training task in TAPAS over millions of tables crawled from Wikipedia and related text segments. The results show that TAPAS can effectively learn correlations between claim C and table T and among cells in table T. Second, the structure-aware fine-tuning method `SAT` is useful. The main reason is that `SAT` utilizes the attention mechanism shown in Figure 4 to capture the structure information in tables. We also have an interesting observation that using `SAT` over the basic LM BERT achieves comparable accuracy with TAPAS, e.g., 68.6% vs. 72.4% on the `All` test set. Because TAPAS needs to pre-train its LM model over millions of tables and text, which is non-trivial and will incur high cost, the alternative of using `SAT` is more lightweight. Note that, when using TAPAS as the pre-trained LM, the improvement of using `SAT` is not significant. This is because TAPAS has already captured table structures. In summary, the experiment results show that encoding the structure of tables is beneficial to table-based fact verification.

We also evaluate the effect of intermediate pre-training, and report the results in Table 12. We can see that intermediate pre-training is helpful under different pre-trained LMs and fine-tuning methods. The results show that it is beneficial to utilize the downstream datasets to learn *task-specific* representation before fine-tuning.

Evaluation on table serialization. Next, we evaluate the strategies for table serialization on various types of test sets. For a more comprehensive comparison, we consider two settings of "`max_sequence_length`", namely 128 and 256, in the pre-trained LM model. Table 13 shows the results. First, the performance of

Table 12: Effect of intermediate pre-training for different models (Accuracy %).

Model	**All Test**		**Simple Test**		**Complex Test**	
	w/o training	**w/. training**	**w/o training**	**w/. training**	**w/o training**	**w/. training**
BERT+TFM	63.1	66.3 (+3.2)	68.2	74.0 (+5.8)	60.6	62.6 (+2.0)
BERT+SAT	68.6	69.8 (+1.2)	78.0	80.9 (+2.9)	63.9	64.4 (+0.5)
TAPAS+TFM	72.4	72.5 (+0.1)	81.1	81.8 (+0.7)	68.2	67.9 (-0.3)
TAPAS+SAT	72.7	73.3 (+0.6)	81.8	84.6 (+2.8)	68.3	67.8 (-0.5)

Table 13: Comparison of sampling strategies over tables (Accuracy %, Model=`BERT+SAT`).

Model	**All Test**		**Simple Test**		**Complex Test**	
	max-seq=128	**max-seq=256**	**max-seq=128**	**max-seq=256**	**max-seq=128**	**max-seq=256**
NoSample	64.2	70.6	70.7	82.0	61.0	65.0
RwSample	66.5	71.8	74.5	83.4	62.5	65.9
CeSample	**70.1** (+5.9)	**72.5** (+1.9)	**81.7** (+11.0)	**85.9** (+3.9)	**64.4** (+3.4)	**66.1** (+1.1)

`NoSample` degrades with a large margin, when reducing "`max_sequence_length`" from 256 to 128, e.g., from 70.6% to 64.2% on the `All` test set. This result reveals the limitation of existing table serialization techniques for fact verification over large structured tables. Second, claim-based sampling methods, i.e., row-based sampling (`RwSample`) and cross-based sampling (`CeSample`) can effectively address the limitation. For instance, when setting "`max_sequence_length`" as 128, `CeSample` can outperform `NoSample` by 5.9% and achieves comparable accuracy with `max_sequence_length` $= 256$ (i.e., 72.5%).

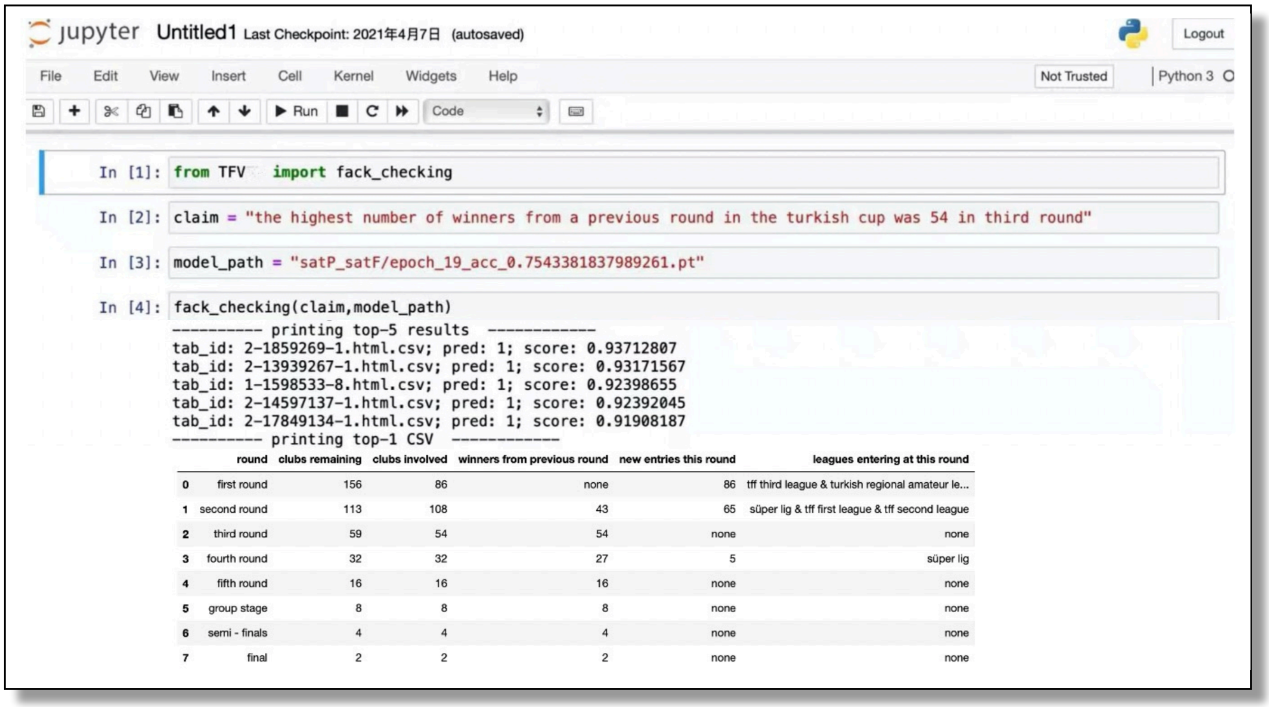
Comparison with existing methods. We compare `TFV` with the existing methods proposed in TabFact [3]. Specifically, TabFact introduces two solutions: (1) `TableBert` is the same as our basic setting `BERT+TFM` with BERT as the pre-trained LM model and basic binary classification for fine-tuning. (2) `LPA` parses claims into symbolic-reasoning programs and executes the programs over the structured tables to obtain binary verification result. Figure 7 shows the results. We can see that `TFV` achieves better accuracy than `TableBert` and `IPA`. The results show that the current symbolic-reasoning methods, such as `IPA`, are still under-explored, which may raise new research directions.

4.3 Summary and Takeaways

Based on our experiment findings, we summarize the following key insights that provide guidance to practitioners and researchers on the study of table-based fact verification.

- **Finding 1: Structure information is indispensable for table-based fact verification.** There are two useful solutions to capturing structure information: (i) The first solution utilizes structure-aware pre-trained LM, such as TAPAS, which achieves the best performance but would incur high cost during pre-training. (ii) The second solution relies on more lightweight structure-aware fine-tuning that also achieves comparable accuracy. It calls for more thorough explorations on how to combine the two solutions.

- **Finding 2: Table serialization is important but yet under-explored.** The state-of-the-art benchmarking dataset TabFact [3] only considers small tables from Wikipedia. However, when adopting table-based fact verification in practice, we need to study how to cope with large tables. The preliminary results show that the current table serialization approaches are not effective and simple heuristic sampling solutions are helpful. Therefore, it calls for more theoretical and empirical studies on table serialization.


jupyter Untitled1 Last Checkpoint: 2021年4月7日 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Code

```
In [1]: from TFV   import fack_checking

In [2]: claim = "the highest number of winners from a previous round in the turkish cup was 54 in third round"

In [3]: model_path = "satP_satF/epoch_19_acc_0.7543381837989261.pt"

In [4]: fack_checking(claim,model_path)
        ---------- printing top-5 results  ------------
        tab_id: 2-1859269-1.html.csv; pred: 1; score: 0.93712807
        tab_id: 2-13939267-1.html.csv; pred: 1; score: 0.93171567
        tab_id: 1-1598533-8.html.csv; pred: 1; score: 0.92398655
        tab_id: 2-14597137-1.html.csv; pred: 1; score: 0.92392045
        tab_id: 2-17849134-1.html.csv; pred: 1; score: 0.91908187
        ---------- printing top-1 CSV  ------------
```

	round	clubs remaining	clubs involved	winners from previous round	new entries this round	leagues entering at this round
0	first round	156	86	none	86	tff third league & turkish regional amateur le...
1	second round	113	108	43	65	süper lig & tff first league & tff second league
2	third round	59	54	54	none	none
3	fourth round	32	32	27	5	süper lig
4	fifth round	16	16	16	none	none
5	group stage	8	8	8	none	none
6	semi - finals	4	4	4	none	none
7	final	2	2	2	none	none

Figure 8: An example of using our python package of TFV in Jupyter Notebook.

- **Finding 3: Symbolic reasoning approaches are not well studied.** Our experimental results show that the current symbolic reasoning approach IPA [3] achieves inferior accuracy compared with TFV. IPA applies lexical matching to find linked entities in the table and then uses pre-defined templates (e.g., count, argmax, etc.) to generate programs. However, as observed from our example, it may not be easy for linking entities (e.g., "third round" vs. "round 3") and determining correct templates.

5 A Python Package of TFV

We develop a python package[4] that implements TFV. Figure 8 shows an example of using the python package for table-based fact verification in Jupyter Notebook. Fed with a collection of structured tables, we offer fact verification services for users. Note that we do not need the users to provide a specific table T for verification. Instead, we index all the tables and, given a claim provided by a user, we first select candidate tables by applying a keyword matching method based on the TaBERT model [30]. Then, our pre-trained and fine-tuned LM model are then leveraged in the second step to produce the probabilities that support the claim. Finally, we rank the candidate tables according to the supporting probabilities and return the top-k tables.

Moreover, the python package implements representative design solutions for all the modules in Figure 2. Specifically, in each module, such as pre-trained LM, fine-tuning and table serialization, users can choose an appropriate solution to use and combine solutions in multiple modules, so as to easily evaluate various approaches. Note that our framework is extensible, i.e., it is possible to incorporate new modules, new categories, or new methods or variants of existing methods. Note that it is also possible to define the search space from a different angle – we contend that our proposal is rational but may not be the only sensible one.

[4]https://github.com/zihuig/TFV

6 Conclusions and Future Direction

In this paper, we have systemically investigated the problem of table-based fact verification. We have introduced a general framework called TFV and defined a space of solutions for the modules in TFV. We have conducted experiments to test different combinations of methods in the design space with several empirical findings: (1) Structure information is indispensable for table-based fact verification; (2) Table serialization is important but yet under-explored; (3) Symbolic reasoning approaches are not well studied. We have developed a python package that implements TFV and presented its user-friendly features for fact verification.

We also identify several future directions in table-based fact verification that may be worthy of exploration.

- **Benchmarking Datasets.** The state-of-the-art benchmarking dataset TabFact [3] has a limitation that the claims are *not real*. Instead, TabFact solicits crowdsourcing workers to narrate the tables, e.g., describing a single tuple or comparing multiple tuples, to generate claims. Therefore, this dataset may not be effective to tackle claims that people make in natural scenarios, which are more difficult to be aligned to the tables. Thus, it calls for more benchmarking datasets containing real claims.

- **Tabular Data Representation.** Another fundamental problem is whether the current pre-trained LM models [7, 18, 29], which are original designed for natural language, is adequate for tabular data representation. For example, one inherent property of tabular data is *permutation invariant*, i.e., changing the order of rows/columns will not affect the result of fact verification. The current LM models have not considered this property. Therefore, it is desirable to develop new models, such as relational pre-trained transformers [26] and generative adversarial networks [11], for tabular data representation.

- **Symbolic Reasoning.** One experiment finding revealed by this paper is the current symbolic approach achieves inferior performance. However, symbolic reasoning should be indispensable for table-based fact verification, especially for complex claims. Thus, it is desirable to study more effective symbolic reasoning techniques, which may be combined with structure-aware LM models to improve the performance.

References

[1] A. Asai and H. Hajishirzi. Logic-guided data augmentation and regularization for consistent question answering. In *ACL*, pages 5642–5650, 2020.

[2] S. R. Bowman, G. Angeli, C. Potts, and C. D. Manning. A large annotated corpus for learning natural language inference. In *EMNLP*, pages 632–642, 2015.

[3] W. Chen, H. Wang, J. Chen, Y. Zhang, H. Wang, S. Li, X. Zhou, and W. Y. Wang. Tabfact: A large-scale dataset for table-based fact verification. In *ICLR*, 2020.

[4] K. Clark, M. Luong, Q. V. Le, and C. D. Manning. ELECTRA: pre-training text encoders as discriminators rather than generators. In *ICLR*, 2020.

[5] P. Clark. Project aristo: Towards machines that capture and reason with science knowledge. In *K-CAP*, pages 1–2, 2019.

[6] I. Dagan, O. Glickman, and B. Magnini. The PASCAL recognising textual entailment challenge. In *MLCW 2005*, volume 3944, pages 177–190, 2005.

[7] J. Devlin, M. Chang, K. Lee, and K. Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, pages 4171–4186, 2019.

[8] J. Dodge, S. Gururangan, D. Card, R. Schwartz, and N. A. Smith. Show your work: Improved reporting of experimental results. In *EMNLP-IJCNLP*, pages 2185–2194, 2019.

[9] J. M. Eisenschlos, S. Krichene, and T. Müller. Understanding tables with intermediate pre-training. In *EMNLP*, volume EMNLP 2020, pages 281–296. Association for Computational Linguistics, 2020.

[10] J. Fan, G. Li, and L. Zhou. Interactive SQL query suggestion: Making databases user-friendly. In *ICDE*, pages 351–362, 2011.

[11] J. Fan, T. Liu, G. Li, J. Chen, Y. Shen, and X. Du. Relational data synthesis using generative adversarial networks: A design space exploration. *Proc. VLDB Endow.*, 13(11):1962–1975, 2020.

[12] M. Geva, A. Gupta, and J. Berant. Injecting numerical reasoning skills into language models. In *ACL*, pages 946–958, 2020.

[13] A. Hanselowski, H. Zhang, Z. Li, D. Sorokin, B. Schiller, C. Schulz, and I. Gurevych. Ukp-athene: Multi-sentence textual entailment for claim verification. *CoRR*, abs/1809.01479, 2018.

[14] J. Herzig, P. K. Nowak, T. Müller, F. Piccinno, and J. M. Eisenschlos. Tapas: Weakly supervised table parsing via pre-training. In *ACL*, pages 4320–4333, 2020.

[15] J. Howard and S. Ruder. Universal language model fine-tuning for text classification. In *ACL*, pages 328–339, 2018.

[16] M. Iyyer, W. Yih, and M. Chang. Search-based neural structured learning for sequential question answering. In *ACL*, pages 1821–1831, 2017.

[17] D. Khashabi, T. Khot, A. Sabharwal, P. Clark, O. Etzioni, and D. Roth. Question answering via integer programming over semi-structured knowledge. In *IJCAI*, pages 1145–1152, 2016.

[18] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692, 2019.

[19] T. Müller, F. Piccinno, P. Shaw, M. Nicosia, and Y. Altun. Answering conversational questions on structured data without logical forms. In *EMNLP-IJCNLP*, pages 5901–5909, 2019.

[20] P. Pasupat and P. Liang. Compositional semantic parsing on semi-structured tables. In *ACL*, pages 1470–1480, 2015.

[21] R. Peeters, C. Bizer, and G. Glavas. Intermediate training of BERT for product matching. In *DI2KG@VLDB*, volume 2726 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2020.

[22] K. Popat, S. Mukherjee, J. Strötgen, and G. Weikum. Where the truth lies: Explaining the credibility of emerging claims on the web and social media. In *WWW*, pages 1003–1012. ACM, 2017.

[23] Y. Pruksachatkun, J. Phang, H. Liu, P. M. Htut, X. Zhang, R. Y. Pang, C. Vania, K. Kann, and S. R. Bowman. Intermediate-task transfer learning with pretrained language models: When and why does it work? In *ACL*, pages 5231–5247, 2020.

[24] Y. Pruksachatkun, J. Phang, H. Liu, P. M. Htut, X. Zhang, R. Y. Pang, C. Vania, K. Kann, and S. R. Bowman. Intermediate-task transfer learning with pretrained models for natural language understanding: When and why does it work? *CoRR*, abs/2005.00628, 2020.

[25] F. Salvatore, M. Finger, and R. H. Jr. A logical-based corpus for cross-lingual evaluation. In *DeepLo@EMNLP-IJCNLP*, pages 22–30, 2019.

[26] N. Tang, J. Fan, F. Li, J. Tu, X. Du, G. Li, S. Madden, and M. Ouzzani. RPT: relational pre-trained transformer is almost all you need towards democratizing data preparation. *Proc. VLDB Endow.*, 14(8):1254–1261, 2021.

[27] J. Thorne, A. Vlachos, C. Christodoulopoulos, and A. Mittal. FEVER: a large-scale dataset for fact extraction and verification. In *NAACL-HLT*, pages 809–819, 2018.

[28] A. Wang, Y. Pruksachatkun, N. Nangia, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman. Superglue: A stickier benchmark for general-purpose language understanding systems. In *NeurIPS*, pages 3261–3275, 2019.

[29] Z. Yang, Z. Dai, Y. Yang, J. G. Carbonell, R. Salakhutdinov, and Q. V. Le. Xlnet: Generalized autoregressive pretraining for language understanding. In *NeurIPS*, pages 5754–5764, 2019.

[30] P. Yin, G. Neubig, W. Yih, and S. Riedel. Tabert: Pretraining for joint understanding of textual and tabular data. In *ACL*, pages 8413–8426. Association for Computational Linguistics, 2020.

[31] H. Zhang, Y. Wang, S. Wang, X. Cao, F. Zhang, and Z. Wang. Table fact verification with structure-aware transformer. In *EMNLP*, pages 1624–1629, 2020.

[32] W. Zhong, D. Tang, Z. Feng, N. Duan, M. Zhou, M. Gong, L. Shou, D. Jiang, J. Wang, and J. Yin. Logicalfactchecker: Leveraging logical operations for fact checking with graph module network. In *ACL*, pages 6053–6065, 2020.