# Structural Encoding and Pre-training Matter: Adapting BERT for Table-Based Fact Verification

**Rui Dong** **David A. Smith**
Khoury College of Computer Sciences
Northeastern University
{dongrui, dasmith}@ccs.neu.edu

## Abstract

Growing concern with online misinformation has encouraged NLP research on fact verification. Since writers often base their assertions on structured data, we focus here on verifying textual statements given evidence in tables. Starting from the Table Parsing (TAPAS) model developed for question answering (Herzig et al., 2020), we find that modeling table structure improves a language model pre-trained on unstructured text. Pre-training language models on English Wikipedia table data further improves performance. Pre-training on a question answering task with column-level cell rank information achieves the best performance. With improved pre-training and cell embeddings, this approach outperforms the state-of-the-art Numerically-aware Graph Neural Network table fact verification model (GNN-TabFact), increasing statement classification accuracy from 72.2% to 73.9% even without modeling numerical information. Incorporating numerical information with cell rankings and pre-training on a question-answering task increases accuracy to 76%. We further analyze accuracy on statements implicating single rows or multiple rows and columns of tables, on different numerical reasoning subtasks, and on generalizing to detecting errors in statements derived from the ToTTo table-to-text generation dataset.

## 1 Introduction

The rapid growth in the amount and sources of online textual content has raised concerns about misinformation and its potential harmful impacts on society when quickly spread to a massive audience. For example, a study on enhancing medical education with Wikipedia in 2015 found that 97% of medical students completing the survey disclosed that they found mistakes in Wikipedia medical entries (Herbert et al., 2015). Concerns about misinformation have stimulated extensive research on automatic fact verification, i.e., verifying whether a given textual statement is entailed or refuted by the given evidence.

**1983 - 84 Houston Rockets Season**

| Round | Pick | Player | College |
|---|---|---|---|
| 1 | 1 | Ralph Sampson | Virginia |
| 1 | 3 | Rodney Mccray | Louisville |
| 3 | 48 | Craig Ehlo | Washington State |
| 4 | 71 | Darrell Browder | Texas Christian |
| 5 | 94 | Chuck Barnett | Oklahoma |

**Entailed Statement**

Ralph Sampson was two picks ahead of Rodney Mccray in round one.

**Refuted Statement**

There were three players picked in the first round.

Figure 1: Example from the TabFact dataset: the top table contains the structured evidence; the bottom two boxes contain the statements entailed and refuted by the evidence. Errors are highlighted in red.

While most existing fact verification work focuses on unstructured textual evidence, fact verification with structured evidence is still underexplored. Recently, Chen et al. (2019) introduced a new large-scale dataset, TabFact, for verifying statements based on structured evidence in tables. Figure 1 presents an example table and the corresponding entailed and refuted statements from TabFact. The task of fact verification based on structured evidence is challenging in two aspects. First, traditional language models trained on unstructured text are not directly applicable to learn representations for structured text. It is difficult for such language models to understand a sentence like "*Round Pick Player College 1 1 Ralph Sampson Virginia ...*" by directly concatenating the table cells in Figure 1. Second, detecting misinformation with structured evidence involves not only linguistic inference but also numerical reasoning such as *addition*, *subtraction*, *sorting*, and *counting*



*Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics*, pages 2366–2375
April 19 - 23, 2021. 

over records. For example, to verify the statement "*Ralph Sampson was two picks ahead of Rodney Mccray*", we need first to find in which order each was picked and subtract them.

Table representation learning is important for utilizing table data as evidence for fact verification. Most existing methods apply BERT (Devlin et al., 2018) model to learn table representations. Table-BERT (Chen et al., 2019) uses simple templates to transform tables into "somewhat natural" sentences, and fine-tunes BERT on pairs of statement sentence and corresponding table "sentence". However, this model adds many extra tokens to the tables, sometimes doubling the length of the original table token sequence. Zhong et al. (2020) propose to first derive logical forms from the table and the statement. A heterogeneous graph is then constructed to capture connections between table cells, functions and arguments and statement tokens. A graph-enhanced contextual representation is learned for each token by only paying attention to the neighbor nodes in the graph when applying BERT model. However, as we show in §4.2, BERT is less effective when it is pre-trained on unstructured data but applied to structured data such as tables.

Table-based fact verification also requires numerical reasoning over table records. Chen et al. (2019) propose a Latent Program Algorithm (LPA), where the statements are parsed into potential programs and a weakly supervised discriminator is trained to assign confidence score to each program. The output from the latent programs are aggregated or ranked according to their confidence score as the final prediction. Zhong et al. (2020) propose to learn a representation for the program with a program-driven neural module network, where semantic compositionality is dynamically modeled along the program parsing structure in a bottom-up style. The program representation and the token representation for the table, statement and linearized program are then combined to make the final prediction. However, these two models depends on weakly-supervised labels, which could be noisy, to derive potential programs. Recently, the same authors of Table-BERT released a new model, GNN-TabFact[1], applying Numerical-aware Graph Neural Networks (NumGNN) on top of Table-BERT to learn to compare numerical cells in the same column. Nonetheless, it requires constructing a graph neural network and conducting iterative message passing among the nodes in the graph.

We propose to adapt the Table Parsing (TAPAS) model (Herzig et al., 2020), which has proven effective in question answering over tables, to model tables for fact verification. TAPAS concatenates all table cells without adding extra tokens and then extends BERT's architecture with additional embedding layers to capture the table structure and numerical comparison information for each token in the table. We replace its original two top classification layers for answer generation with a single classification layer on the `[CLS]` token to classify whether a given statement is entailed or refuted by the table. Our experimental results show that with proper pre-training, TAPAS outpeforms the state-of-the-art GNN-TabFact model, increasing the accuracy of statement classification from 72.2% to 73.9% on the TabFact test set even without modeling numerical information. By further adding ranking information for numerical rows and pre-training on the question answering task, TAPAS achieves 76% accuracy, with 89% on the simple statements and 69.8% on the complex statements. We also perform further analysis to examine: (1) how the numerical comparison information improves TAPAS's performance when ranking information is needed; (2) how the complexity of the training set affects model performance on simple and complex statements; and (3) how well systems trained on TabFact generalize to other fact verification tasks.

This paper's primary contributions are: (1) exploring the effect of table structure modeling on fact verification; (2) measuring the importance of language model pre-training on tabular data; and (3) analyzing the performance of fact-verification models on different numerical reasoning subtasks and errors.

## 2 Related Work

**Fact Verification** Thorne et al. (2018) introduce a new dataset for fact extraction and verification (FEVER) with claims generated by altering sentences extracted from Wikipedia. Nie et al. (2019) propose a neural semantic matching network based on a bidirectional LSTM to retrieve related evidence and predict whether the claim is entailed, neutral, or contradicted by the evidence. (Soleimani et al., 2019) propose to adopt a pre-trained BERT model for evidence retrieval and

[1] https://github.com/wenhuchen/GNN-TabFact

fine tuning it for evidence-claim relation prediction. Jobanputra (2019) propose an unsupervised question-answering based approach for fact checking by generating questions for a claim first and predicting the masked span, which is compared to the ground truth answer for label classification. Zhong et al. (2020) construct two graphs for evidence and claim via semantic role labeling, and use graph-based reasoning for fact checking.

**Structured Data Modeling** Modeling structured data is essential for multiple tasks, e.g., table classification, table population, table retrieval, question answering , data-to-text generation, and table-based fact verification.

Ghasemi-Gol and Szekely (2018), Trabelsi et al. (2019) and Deng et al. (2019) propose to embed tabular data into a vector space, using table structure to classify tables into different categories, adding more rows or columns to tables and retrieving tables given query keywords. Nishida et al. (2017) employ RNNs to encode cell content and CNNs to encode table structure to better capture semantic features for table classification.

Haug et al. (2018) propose to generate candidate logical forms from a question, convert logical forms to paraphrases, and rank them according to their similarity to the original question to generate the answer for the questions. Herzig et al. (2020) add additional embedding layers to a BERT model to capture table structure and numerical information, and add two classification layers to predict aggregation functions and corresponding table cells to generate an answer for a given question.

Much current table-to-text generation work focus on generating biographies from Wikipedia infoboxes (Lebret et al., 2016; Liu et al., 2018; Sha et al., 2018; Bao et al., 2018) or summarizing basketball games (Wiseman et al., 2017; Puduppully et al., 2019) according to the box- and line-score tables. Parikh et al. (2020) release a dataset for controlled table-to-text generation, where table cells are highlighted for the target sentences. Chen et al. (2020) propose a new natural language generation task, where the model tasked with generating statements that entailed by a given table.

Chen et al. (2019) introduce a dataset for fact verification given tabular data as evidence and propose a BERT-based Table-BERT model and a latent program algorithm (LPA) model for this task. Zhong et al. (2020) propose to first drive the program from the table and statement, and then learn graph-enhanced contextual representations for both the table tokens and the program to classify the statements. After this paper was submitted, four related papers were published, which explore additional questions in table-based fact verification. Zhang et al. (2020) propose to utilize masking in the self attention layer to model table structure. Shi et al. (2020) and Yang et al. (2020) explore how to effectively combine both linguistic information and symbolic information for table-based fact verification. Eisenschlos et al. (2020) generate synthetic datasets to pre-train a TAPAS model to better understand tables for downstream tasks such as table-based fact verification and question answering.

## 3 Methods

We describe adapting the TAPAS model to fact verification over tables and then introduce different pre-trained models on which we fine-tune the TAPAS for the table verification task.

**TAPAS for table-based fact verification** Similar to the Table-BERT model, the TAPAS model also flattens the input table into a sequence of tokens. It concatenates all the tokenized table cells in a row and then concatenates all the row sequences into a table sequence. In addition to the position and segment embeddings used in Transformer language models, four additional position embedding layers are introduced for encoding table structure and numerical comparison relations among cells in each column. For each token in the table, there is a column index and a row index indicating the position of the token in the table. For numerical and date columns (as determined by pandas), table cells are sorted to generate a rank index and a inverse rank index for each token according to their position in the sorted column. The sequence of tokens in the statement to be verified is concatenated with the corresponding table token sequence with the `[SEP]` token to indicate where the table sequence starts. For the statement sequence, all the row, column, rank and inverse rank indexes are set as 0. A special token `[CLS]` is added before this entire input sequence for classification purpose. Figure 2 shows how the table and statement from Figure 1 are indexed for each embedding layer. We remove the top two classification layers used in the original TAPAS model for the question answering task, and use the pooled output for the `[CLS]` token for statement classification. To explore the impact

of different table encodings, we experiment with two variants of TAPAS: TAPAS-Row-Col, which only utilizes the column and row index embedding, and TAPAS-Row-Col-Rank, which leverages the additional ranking and inverse ranking information for numerical columns.

**Pre-training** We fine-tune the adapted TAPAS model for the table verification task starting from three different pre-trained models. The first pre-trained model is the widely applied BERT model (Devlin et al., 2018) trained on the BooksCorpus and Wikipedia text. Lists and tables are removed from the training text, so this model is merely trained on unstructured data. The second pre-trained model is the TAPAS-Row-Col-Rank model pre-trained by Herzig et al. (2020) on a large number of Wikipedia text-table pairs with a masked language modeling task. This pre-trained model should be able to better understand the tables by capturing the structure information and numerical information. The third pre-trained model is the TAPAS-Row-Col-Rank model trained on the SQA (Iyyer et al., 2017) dataset with a question answering task. We only utilize their parameters for the bottom layers encoding the text and table sequence. Since this model is trained to generate answers by selecting table cells and predicting the aggregation function to be used on the cells, it should be able to capture more complicated numerical information.

## 4 Experiments

In this section, we first introduce the experimental settings in detail (§4.1). Then we present the results of comparing two variants of TAPAS model and other state-of-the-art models on the table-based fact verification task in §4.2. More discussions about the models are provided in §4.3.

### 4.1 Experimental Setup

**Dataset** In the main experiment, we evaluate different models on TabFact, a large-scale dataset for table-based fact verification proposed by Chen et al. (2019). TabFact is generated by asking annotators to write statements given a table and its caption. The statements are either entailed or refuted by the table content. This dataset contains both *simple* and *complex* statements. Simple statements refer to a single row of the table without complicated logical inference and mention the table cells without much modification. Complex statements are more sophisticated and involve aggregation functions such max, min, count, average, difference, etc., over multiple table records. The mentioned table records are rephrased to involve more semantic understanding. A total of 118,275 statement sentences are generated for 16,573 tables, of which 50,244 are simple statements and 68,031 are complex statements. We use the training, validation and test set splits from Chen et al. (2019) to conduct a fair comparison with their models. Table 1 shows the statistics of these splits.

| Split | #Statements | #Tables |
|---|---|---|
| Train | 92,283 | 13,182 |
| Val | 12,792 | 1,696 |
| Test | 12,779 | 1,695 |

Table 1: Statistics of the training, validation, and test sets for the TabFact in Chen et al. (2019).

**Comparisons** We compare TAPAS-Row-Col and TAPAS-Row-Col-Rank model with the following state-of-the-art models:
*a. Latent Program Algorithm (LPA).* This model (Chen et al., 2019) first uses trigger words to prune pre-defined APIs including around 50 functions such as min, max. Then candidate latent programs are constructed by using breadth-first-search with memoization, and a discriminator is trained with weakly-supervised labels to assign confidence score to the programs. Their best performing model ranks all the latent programs by the discriminator confidence score and make the prediction by executing the top-rated program.
*b. Table-BERT.* This model (Chen et al., 2019) applies a pre-trained BERT model to classify the concatenated table and statement. Their best performing model first concatenates the header of each column to each table cell in the same column by using the template `[HEADER] is [CELL]`. Then table cells in the $i^{th}$ row are concatenated with the ";" symbol and a short phrase `row` $i$ `is ...` is added to the beginning of the concatenated cells to identify the cells from each row of the table. For example, the first row of the table from Figure 1 would be flatted to "*Row 1 is: Round is 1; Pick is 1; Player is Ralph Sampson; College is Virginia.*". All the row sequences are then concatenated as the table sequence, which is further concatenated to the statement sequence with the `[SEP]` symbol

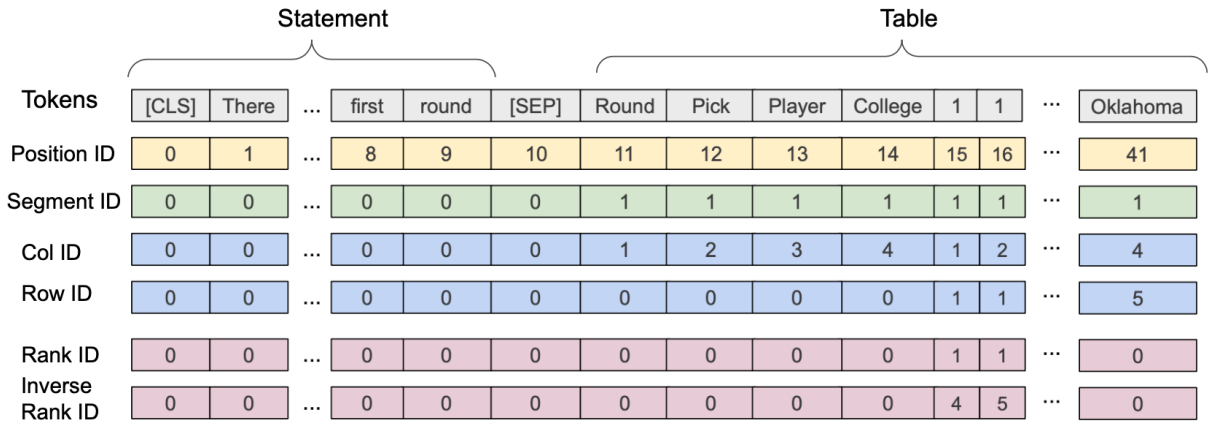


Figure 2: Example of how TAPAS encoding the refuted statement and the table from Figure 1 for classification.

and fed into BERT model for classification.

*c. GNN-TabFact.* This is a newly released state-of-the-art model by the creators of TabFact. It first extracts the representations for table tokens and statement tokens based on Table-BERT, then the representations for the tokens in the same cell are averaged as the representation for each table cell. All the table cells with numerical values in the same column are treated as nodes, and a graph is constructed by connecting the nodes with two types of edges—greater and less than—according the values of the cells. A NumGNN layer is then utilized to propagate information among the nodes in the graph to integrate the numerical comparison information into each table cell's representation. Cross attention is then computed over the new table representation and the textual statement for final classification.

*d. LogicalFactChecker.* This model (Zhong et al., 2020) first derives a program from the table and statement, then learns a contextual representation for the tokens by constructing a heterogeneous graph to capture the connections among the statement, the table and the program. A program-guided neural module network is introduced to learn a representation for the program by capturing its structural and compositional semantics. The token representations and the program representation are then combined to make the final prediction.

**Model Parameters** Following Table-BERT, all of our models are based on the open-source implementation of BERT with 12-layers, 768-hidden, 12-heads[2]. Both statements and tables are tokenized into sequences of subwords with the Standard BERT tokenizer and joined by the `[SEP]` special token. `[CLS]` is added to the beginning of the joined sequence. We modify the BERT code by adding embedding layers for special token types such as row index, column index and (inverse) rank index information following the TAPAS variants described above modeling for table structure.

**Evaluation Metric** All models are evaluated for accuracy on classifying test statements as entailed or refuted by the corresponding evidence table.

### 4.2 Main Results

Table 2 presents our main results, comparing TAPAS-derived models with baseline models on table-based fact verification. As we can see, the TAPAS-Row-Col-Rank model pre-trained on the question answering task over tables achieves the best performance. The TAPAS-Row-Col model pre-trained on WikiTables and fine tuned on TabFact data outperforms the state-of-the-art GNN-TabFact model no matter whether the full table is used or only the subset columns related to the statements are used. (This subsetting gives the best results for GNN-TabFact.)

**How do different ways of modeling table structure affect the model performance?** We first examine how the method of modeling table structure affects performance. Table-BERT uses a simple template to transform a table into a "somewhat natural" sentence to exploit a language model pre-trained on natural sentences. However, Table 2 shows that TAPAS-Row-Col, which directly concatenates all table cells and uses row index and column index position embedding to capture the table structure, significantly outperforms Table-BERT on the complex test set with the same pre-training,

[2] https://github.com/huggingface/pytorch-pretrained-BERT

| Model | Pre-train | Columns | Test | | |
|---|---|---|---|---|---|
| | | | All | Simple | Complex |
| BERT classifier w/o Table | BooksCorpus + Wikipedia Text | N/A | 50.5 | 51.0 | 50.1 |
| Table-BERT | BooksCorpus + Wikipedia Text | subset | 65.1 | 79.1 | 58.2 |
| LPA | N/A | N/A | 65.3 | 78.7 | 58.5 |
| LogicalFactChecker | BooksCorpus + Wikipedia Text | all | 71.7 | 85.4 | 65.1 |
| GNN-TabFact | BooksCorpus + Wikipedia Text | subset | 72.2 | 86.4 | 65.4 |
| TAPAS-Row-Col | BooksCorpus + Wikipedia Text | all | 60.5 | 63.8 | 57.9 |
| TAPAS-Row-Col | BooksCorpus + Wikipedia Text | subset | 68.3 | 79.5 | 62.9 |
| TAPAS-Row-Col | Wikipedia Tables | all | 73.4* | 86.6 | 67.0* |
| TAPAS-Row-Col | Wikipedia Tables | subset | 73.9* | 87.7* | 67.2* |
| TAPAS-Row-Col-Rank | Wikipedia Tables | all | 74.5* | 87.0 | 68.5* |
| TAPAS-Row-Col-Rank | Wikipedia Tables | subset | 74.8* | 88.1* | 68.5* |
| TAPAS-Row-Col-Rank | Wikipedia Tables + SQA | all | **76.0*** | **89.0*** | **69.8** * |
| TAPAS-Row-Col-Rank | Wikipedia Tables + SQA | subset | 74.6* | 88.9* | 67.7* |

Table 2: Percent accuracy of different models for table-based fact verification on the TabFact test set. "Subset" means only use the table columns with entities linked to the statements during training, while "all" means use all the columns. Models with accuracies significantly surpassing the previous state of the art, GNN-TabFact, are highlighted with * ($p < 0.05$, paired-permutation test). **Bold** indicates the best results in a column.

increasing the accuracy from 58.2% to 62.9%. It reveals that TAPAS-Row-Col's structural encoding is more effective for verifying statements involving complicated aggregation over multiple table rows.

**Does filtering the table columns help?** Table-BERT works on the subset of the table columns relevant to the statements, since the a table serialized in "natural" language might be too long for sequence models. Chen et al. (2019) filter tables to the columns containing entities linked to the statement. We run TAPAS-Row-Col model on both full table and subset of the columns. Table 2 shows that, when TAPAS-Row-Col is fine-tuned from the original BERT model, shrinking the table only to the related columns significantly improves its accuracy from 60.5% to 68.3%. However, when we fine tune TAPAS-Row-Col model pre-trained on the Wikipedia tables, the difference is not that significant. It shows that filtering the table columns mainly helps when applying language models pre-trained on unstructured data to structured data.

**What is the effect of pre-training?** We fine-tuned TAPAS-Row-Col on two different types of pre-trained model: the original BERT model pre-trained on unstructured text and the TAPAS model pre-trained on Wikipedia Tables together with sentences from the corresponding text paragraph by Herzig et al. (2020). Both of the pre-trained models are trained with the masked language modeling task. As we can see, pre-training on tables significantly improves performance of TAPAS-Row-Col from 68.3% to 73.9%. Fine-tuning TAPAS-Row-Col-Rank on the model pre-trained on the question answering task, where numerical reasoning skills are required to generate an answer for a given question, further improves accuracy from 74.5% to 76%. However, if we only fine-tune on the subset of table columns matching the statement, TAPAS-Row-Col-Rank does not benefit from the pre-trained question answering model. We conjecture that it is because the original model is pre-trained on full tables.

**Does adding numerical comparisons help?** Both GNN-TabFact and TAPAS-Row-Col-Rank model numerical comparison relations among cells in the same column. GNN-TabFact learns a representation for each table cell by iteratively passing their numerical relations such as greater or less in a graph neural network. TAPAS-Row-Col-Rank model adds the rank and inverse rank index for cells in the same column as special position embedding for each token in the table cells. GNN-TabFact significantly outperforms Table-BERT by adding a numerically aware graph neural network on top of it. It also outperforms the LogicalFactChecker model, which first derives a program from the table and the statement and then learns the representation for the programs via a program-guided neural module network. Also, TAPAS-Row-Col-Rank model significantly outperforms TAPAS-Row-Col by introducing rank position embedding for each numerical column. We also find that TAPAS-Row-Col-Rank outperforms GNN-TabFact without complicated graph inference. It is worth noting that even TAPAS-Row-Col, which does not utilize any numerical information, outperforms GNN-TabFact, the previous state of the art on TabFact. This again demonstrates the advantage of directly encoding ta-

| Model | Pre-train | Columns | Test | | | |
|---|---|---|---|---|---|---|
| | | | Superlative | Comparative | Sum | Count |
| Table-BERT | BooksCorpus + Wikipedia Text | subset | 59.2 | 56.3 | 60.9 | 54.9 |
| GNN-TabFact | BooksCorpus + Wikipedia Text | subset | 63.7 | 62.1 | 61.2 | 64.6 |
| TAPAS-Row-Col | BooksCorpus + Wikipedia Text | all | 58.7 | 61.2 | 57.8 | 59.8 |
| TAPAS-Row-Col | Wikipedia Tables | all | 66.3 | 63.4 | 65.6 | 65.3 |
| TAPAS-Row-Col-Rank | Wikipedia Tables | all | 72.0 | 64.7 | 63.3 | 67.2 |
| TAPAS-Row-Col-Rank | Wikipedia Tables + SQA | all | **74.6** | **66.8** | **66.3** | **70.6** |

Table 3: Results on test statements with superlative, comparative, sum, and count operations.

ble structure in TAPAS and the value of pre-training on structured data.

### 4.3 Further Experiments and Discussion

To analyze model performance, we perform further experiments on different training and test sets.

**How does numerical information help?** To examine how numerical comparison information helps to improve performance, we use heuristic rules to find the test statements involving *comparing* table records, *summing* table columns, and *counting* table records. We first find all words ending in *-est* in complex statements and remove those that do not belong to superlative words. Then we extract all test samples from the complex set containing these words or the word *most* as the **superlative test set**. We construct a **comparative test set** in a similar way by finding words ending in *-er*. Non-comparative words are removed and the words *more* and *less* are added to the set. We constrain the comparative samples to have the word *than* together with one of the comparative words. The **sum test set** is constructed by finding all the samples containing *total* or *sum*, the **count** set is constructed by finding all the samples containing *all*, *every*, *none*, *only* and *each*. The final superlative and comparative test sets contain 1,701 and 1,366 instances, respectively. The sum and counting sets contain 344 and 1,710, respectively. Table 3 shows the results of different models on these four sets. All results are reported for encoding full tables for variants of TAPAS-Row-Col models. As we can see, by modeling numerical comparison information among table cells in the same column, GNN-TabFact model significantly outperforms Table-BERT model on the superlative, comparative and count sets. Fine tuning TAPAS-Row-Col model pre-trained on the Wikipedia tables dataset improves performance on all the sets compared to fine tuning it from pre-trained BERT model. By adding ranking information, TAPAS-Row-Col-Rank significantly improves over TAPAS-Row-Col model on the superlative set. The increase of performance on other sets are not significant. Fine-tuning TAPAS-Row-Col-Rank on the model pre-trained for question answering further improves performance on all sets. It is also worth noting that TAPAS-Row-Col, with pre-training on Wikipedia tables, even outperforms GNN-TabFact model without utilizing any numerical comparison information. It also reveals that adding ranking information model benefits superlative statements most, and classifying comparative statements is more difficult than superlative statements.

| Training Set | All | Simple | Complex |
|---|---|---|---|
| Complex | 70.2 | 76.4 | **67.2** |
| Simple | 68.4 | **88.2** | 58.9 |
| Mixed | **73.0** | 86.1 | 66.6 |

Table 4: Results of fine-tuning Tapas-Row-Col-Rank model on statements with different complexity.

**How does training set complexity affect model performance?** We sampled three subsets from the original training set: one only contains simple statements (as defined by TabFact), one only contains complex statements, and one mixed set with equal size of simple and complex statements. All three sets have the same size: 41,366 training samples and 12,793 validation samples. We fine-tune the TAPAS-Row-Col-Rank model pre-trained on the question-answering task on three different sets and present the results in Table 4. The model trained on the mixed data achieves the highest overall accuracy on the test set. A significant drop of performance could be observed when the model is trained and test on statements with mismatched complexity.

**How well do models trained on TabFact generalize to other data?** Since statements in TabFact were generated by annotators specifically for fact verification, we explored generalization performance with a synthetic table-based fact verification dataset generated from the ToTTo dataset, a

| Train | TabFact | | | Synthetic | | |
|---|---|---|---|---|---|---|
| | All | Simple | Complex | All | Postive | Negative |
| TabFact | 76.0 | 89.0 | 69.8 | 79.2 | 86.9 | 71.6 |
| Synthetic | 63.3 | 76.4 | 57.0 | 87.0 | 92.1 | 81.8 |

Table 5: TAPAS-Row-Col-Rank model training and testing on TabFact and ToTTo-derived synthetic data.

| | Original Entailed Statement | Synthetic Refuted Statements |
|---|---|---|
| 1 | Baron Fabian Von Fersen (1626 – 1677) was a Swedish field marshal. | Baron Fabian Von Fersen (626 – 1677) was a Swedish field marshal. |
| 2 | As a sophomore, Peters averaged 16.8 points and 6.7 rebounds per game. | As a sophomore, Peters averaged 16.8 points and 66.7 rebounds per game. |
| 3 | Ralph Herseth (1909 – 1969) was the 21st governor of South Dakota from January 6, 1959 to January 3, 1961. | Ralph Herseth (1909 – 1969) was the 11st governor of South Dakota from January 6, 1959 to January 3, 1961. |

Table 6: Examples of synthetic refuted statements that the Tapas-Row-Col-Rank model trained on TabFact failed to recognize. The errors are in red and the corresponding correct numbers are underlined.

table-to-text generation dataset proposed by Parikh et al. (2020). ToTTo was constructed by asking annotators to modifying the original sentence in Wikipedia that referenced a table. Annotators deleted irrelevant parts of the sentence and replaced pronominal references with a named entity from the table or the contextual metadata. Annotators were also asked to highlight the table cells that support the sentence. In all, 128,461 tables were annotated, with one sentence per table.

From ToTTo, using negative sampling, we derive a synthetic dataset for fact verification to compare with the TabFact data. We first removed all the samples in Totto data that also appears in TabFact data. Since the cells related to the summarizing sentence are highlighted in ToTTo, we extract all the sentences containing entities that exactly match the highlighted table cells and treat them as as facts. Then for each fact statement, we randomly choose an entity in it and replace it with a randomly chosen cell from the same column to generate a false statement. To ensure the false statement is different from the fact statement, we only choose cells whose values differ from the original. We end up with 75,292 training, 8,366 validation, and 5,242 test samples. We fine-tune the TAPAS-Row-Col-Rank model on this synthetic dataset. The results on this synthetic dataset and TabFact are listed in Table 5. Training on synthetic data achieves 76.4% accuracy on the simple TabFact statements, while training on TabFact achieves 79.2% accuracy on synthetic data. This confirms that the model trained on TabFact model is able to recognize cell copying errors. Both models are better at classifying positive statements than negative statements, which is more obvious for the model trained on TabFact.

We also generated another synthetic dataset with uniformly distributed character-editing errors on digits to see whether models trained on TabFact could capture this kind of error. We first extract all the test sentences containing numbers that exactly matches the highlighted table cells. Then we randomly choose to insert, delete or substitute one random digit in the first position (to make the task easier) of a randomly chosen matching number in the sentence (to ensure there is clue in the table). A total of 1,126 synthetic refuted statements are generated for testing. We check whether the TAPAS-Row-Col-Rank model trained on TabFact can recognize this type of error. Only 43.0% of these synthetic statements are classified as refuted by the model, which is much lower than the 71.6% accuracy on the refuted statements generated above by cell copying. It reveals that models trained on TabFact data are less sensitive to purely numerical errors. Table 6 shows examples of synthetic false statements the model trained on TabFact failed to recognize.

## 5 Conclusions

We adapted the Table Parsing (TAPAS) model, with efficient encoding of table content, structure, and numerical comparison information for the task of table-based fact verification. We compared two variant models: TAPAS-Row-Col, which models table contents and structure, and TAPAS-Row-Col-Rank, which adds numerical comparison information. Experiments showed that both TAPAS-Row-Col and TAPAS-Row-Col-Rank outperform the state-of-the-art numerically-aware graph neural net-

work model by pre-training on tabular data. We also examined how ranking information helps improve TAPAS's performance on superlative, comparative, sum, and count statements. Models trained on different datasets were compared to study how question complexity affects model performance. We also constructed two synthetic datasets to examine the generalization of these models. We find models trained on TabFact perform well on errors arising from simple cell replacement but not on digit-editing errors. In future, we aim to extend TAPAS to explicitly model more numerical operations for fact verification and to correct false statements automatically given tabular evidence.

## Acknowledgements

Rui Dong was supported by a U.S. Department of Energy Award (DE-SC0019958). We thank Brian Simmons for helpful discussions and the anonymous reviewers for their suggestions.

## References

Junwei Bao, Duyu Tang, Nan Duan, Zhao Yan, Yuanhua Lv, Ming Zhou, and Tiejun Zhao. 2018. Table-to-text: Describing table region with natural language. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Wenhu Chen, Jianshu Chen, Yu Su, Zhiyu Chen, and William Yang Wang. 2020. Logical natural language generation from open-domain tables. *arXiv preprint arXiv:2004.10404*.

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*.

Li Deng, Shuo Zhang, and Krisztian Balog. 2019. Table2vec: Neural word and entity embeddings for table population and retrieval. *arXiv preprint arXiv:1906.00041*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Julian Eisenschlos, Syrine Krichene, and Thomas Mueller. 2020. Understanding tables with intermediate pre-training. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 281–296.

Majid Ghasemi-Gol and Pedro Szekely. 2018. Tabvec: Table vectors for classification of web tables. *arXiv preprint arXiv:1802.06290*.

Till Haug, Octavian-Eugen Ganea, and Paulina Grnarova. 2018. Neural multi-step reasoning for question answering on semi-structured tables. In *European conference on information retrieval*, pages 611–617. Springer.

Verena G Herbert, Andreas Frings, Herwig Rehatschek, Gisbert Richard, and Andreas Leithner. 2015. Wikipedia–challenges and new horizons in enhancing medical education. *BMC medical education*, 15(1):32.

Jonathan Herzig, Paweł Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Martin Eisenschlos. 2020. Tapas: Weakly supervised table parsing via pre-training. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Seattle, Washington, United States. To appear.

Mohit Iyyer, Wen-tau Yih, and Ming-Wei Chang. 2017. Search-based neural structured learning for sequential question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1821–1831.

Mayank Jobanputra. 2019. Unsupervised question answering for fact-checking. *arXiv preprint arXiv:1910.07154*.

Rémi Lebret, David Grangier, and Michael Auli. 2016. Neural text generation from structured data with application to the biography domain. *arXiv preprint arXiv:1603.07771*.

Tianyu Liu, Kexiang Wang, Lei Sha, Baobao Chang, and Zhifang Sui. 2018. Table-to-text generation by structure-aware seq2seq learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Yixin Nie, Haonan Chen, and Mohit Bansal. 2019. Combining fact extraction and verification with neural semantic matching networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6859–6866.

Kyosuke Nishida, Kugatsu Sadamitsu, Ryuichiro Higashinaka, and Yoshihiro Matsuo. 2017. Understanding the semantic structures of tables with a hybrid deep neural network architecture. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Ankur P Parikh, Xuezhi Wang, Sebastian Gehrmann, Manaal Faruqui, Bhuwan Dhingra, Diyi Yang, and Dipanjan Das. 2020. Totto: A controlled table-to-text generation dataset. *arXiv preprint arXiv:2004.14373*.

Ratish Puduppully, Li Dong, and Mirella Lapata. 2019. Data-to-text generation with content selection and planning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6908–6915.

Lei Sha, Lili Mou, Tianyu Liu, Pascal Poupart, Sujian Li, Baobao Chang, and Zhifang Sui. 2018. Order-planning neural text generation from structured data. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

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.

Amir Soleimani, Christof Monz, and Marcel Worring. 2019. Bert for evidence retrieval and claim verification. *arXiv preprint arXiv:1910.02655*.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. Fever: a large-scale dataset for fact extraction and verification. *arXiv preprint arXiv:1803.05355*.

Mohamed Trabelsi, Brian D Davison, and Jeff Heflin. 2019. Improved table retrieval using multiple context embeddings for attributes. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 1238–1244. IEEE.

Sam Wiseman, Stuart M Shieber, and Alexander M Rush. 2017. Challenges in data-to-document generation. *arXiv preprint arXiv:1707.08052*.

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.

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.

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.