TABFACT: A LARGE-SCALE DATASET FOR TABLE-BASED FACT VERIFICATION

Anonymous authors
Paper under double-blind review

ABSTRACT

The problem of verifying whether a textual hypothesis holds based on the given evidence, also known as fact verification, plays an important role in the study of natural language understanding and semantic representation. However, existing studies are mainly restricted to dealing with unstructured evidence (e.g., natural language sentences and documents, news, etc), while verification under structured evidence, such as tables, graphs, and databases, remains unexplored. This paper specifically aims to study the fact verification given semi-structured data as evidence. To this end, we construct a large-scale dataset called TabFact with 16k Wikipedia tables as the evidence for 118k human-annotated natural language statements, which are labeled as either ENTAILED or REFUTED. TabFact is challenging since it involves both soft linguistic reasoning and hard symbolic reasoning. To address these reasoning challenges, we design two different models: Table-BERT and Latent Program Algorithm (LPA). Table-BERT leverages the state-of-the-art pre-trained language model to encode the linearized tables and statements into continuous vectors for verification. LPA parses statements into LISP-like programs and executes them against the tables to obtain the returned binary value for verification. Both methods achieve similar accuracy but still lag far behind human performance. We also perform a comprehensive analysis to demonstrate great future opportunities.

1 INTRODUCTION

Verifying whether a textual hypothesis is entailed or refuted by the given evidence is a fundamental problem in natural language understanding (Katz & Fodor, 1963; Van Benthem et al., 2008). It has been extensively studied under different natural language tasks such as recognizing textual entailment (RTE) (Dagan et al., 2005), natural language inference (NLI) (Bowman et al., 2015), claim verification (Popat et al., 2017; Hanselowski et al., 2018; Thorne et al., 2018), and commonsense reasoning (Chierchia & McConnell-Ginet, 2000; Zellers et al., 2018). RTE and NLI view a premise sentence as the evidence, whereas claim verification views passage as the evidence and commonsense reasoning views a context sentence as the evidence. These problems have been previously addressed using a variety of techniques including logic rules, knowledge bases, and neural networks. Recently large-scale pre-trained language models (Devlin et al., 2019; Peters et al., 2018; Radford et al., 2019; Yang et al., 2019; Liu et al., 2019) have surged to dominate the other algorithms to approach human performance on several textual entailment tasks (Wang et al., 2018).

However, existing related studies are restricted to dealing with unstructured text as the evidence, which would not generalize to the cases where the evidence has a highly structured format. Since such structured evidence (graphs, tables, or databases) are also ubiquitous in real-world applications like database systems, dialog systems, commercial management systems, social networks, etc, we argue that the fact verification under structured evidence forms is an equivalently important yet unexplored problem. Therefore, in this paper, we are specifically interested in studying fact verification with semi-structured Wikipedia tables (Bhagavatula et al., 2013)[1] as evidences owing to its structured and ubiquitous nature (Jauhar et al., 2016; Zhong et al., 2017; Pasupat & Liang, 2015). To this end, we introduce a large-scale dataset called TABFACT, which consists of 118K

[1]In contrast to the database tables, where each column has strong type constraint, the cell records in our semi-structured tables can be string/data/integer/floating/phrase/sentences.

United States House of Representatives Elections, 1972

District	Incumbent	Party	Result	Candidates
California 3	John E. Moss	democratic	re-elected	John E. Moss (d) 69.9% John Rakus (r) 30.1%
California 5	Phillip Burton	democratic	re-elected	Phillip Burton (d) 81.8% Edlo E. Powell (r) 18.2%
California 8	George Paul Miller	democratic	lost renomination democratic hold	Pete Stark (d) 52.9% Lew M. Warden , Jr. (r) 47.1%
California 14	Jerome R. Waldie	republican	re-elected	Jerome R. Waldie (d) 77.6% Floyd E. Sims (r) 22.4%
California 15	John J. Mcfall	republican	re-elected	John J. Mcfall (d) unopposed

Entailed Statement

1. John E. Moss and Phillip Burton are both re-elected in the house of representative election.
2. John J. Mcfall is unopposed during the re-election.
3. There are three different incumbents from democratic.

Refuted Statement

1. John E. Moss and George Paul Miller are both re-elected in the house of representative election.
2. John J. Mcfall failed to be re-elected though being unopposed.
3. There are five candidates in total, two of them are democrats and three of them are republicans.

Figure 1: Examples from the TABFACT dataset. The top table contains the semi-structured knowledge facts with caption "United...". The left and right boxes below provide several entailed and refuted statements. The error parts are highlighted with red font.

manually annotated statements with regard to 16K Wikipedia tables, their relations are classified as ENTAILED and REFUTED. The entailed and refuted statements are both annotated by human workers. With some examples in Figure 1, we can clearly observe that unlike the previous verification related problems, TABFACT combines two different forms of reasoning in the statements, (i) *Linguistic Reasoning*: the verification requires semantic-level understanding. For example, "John J. Mcfall failed to be re-elected though being unopposed." requires semantic-level understanding over the statement and the table entries to correctly classify the entailment relation. (ii) *Symbolic Reasoning*: the verification requires symbolic execution on the table structure. For example, the phrase "There are three Democrats incumbents" requires condition operation (where condition) and arithmetic operation (count) to be classified as an entailed statement. The two forms of reasoning are interleaved extensively across the statements, which makes it challenging for existing models.

In this paper, we particularly probe two approaches to deal with such mixed-reasoning challenge: (i) *Table-BERT*, this model views the verification task completely as an NLI problem by linearizing a table as a premise sentence p, and applies state-of-the-art language understanding pre-trained model to encode both the table and statements h into distributed representation for classification. This model is expected to excel at the linguistic reasoning aspects but falls short in the symbolic reasoning aspects. (ii) *Latent Program Algorithm*, this model applies lexical matching to find linked entities and then use pre-defined APIs (e.g. argmax, argmin, count, etc) to construct the potential LISP-like program candidates, a discriminator is further utilized to select the most "consistent" latent programs. This model excels the symbolic reasoning aspects by executing database queries, which also renders better interpretability by laying out the decision rationale. We perform extensive experiments to investigate their performances: the best-achieved accuracy of both models are reasonable, but far below human performance. Thus, we believe that the proposed table-based fact verification task can serve as an important new benchmark towards the goal of building powerful AI that can reason over both soft linguistic form and hard symbolic forms. To facilitate future research, we released all the data, code with the intermediate results.

2 TABLE FACT VERIFICATION DATASET

First, we follow the previous Table-based Q&A datasets (Pasupat & Liang, 2015; Zhong et al., 2017) to extract web tables (Bhagavatula et al., 2013) with captions from WikiTables[2]. Here we filter out overtly complicated and huge tables(e.g. multirows, multicolumns, latex symbol) and obtain 18K relatively clean tables with less than 50 rows and 10 columns.

For crowd-sourcing jobs, we follow the human subject research protocols to hire Amazon Mechanical Turk[3] workers from the native English-speaking countries "US, GB, NZ, CA, AU" with approval rates higher than 95% and more than 500 accepted HITs. Following WikiTableQuestion (Pasupat & Liang, 2015), we provide the annotators with the corresponding table captions to help them better understand the background. To ensure the annotation quality, we develop a pipeline of "positive two-channel annotation" → "negative statement rewriting" → "verification", as described below.

[2] http://websail-fe.cs.northwestern.edu/wikiTables/about/
[3] https://www.mturk.com/

2.1 POSITIVE TWO-CHANNEL COLLECTION

To harvest statements of different difficulty levels, we design a two-channel collection process:

Low-Reward Simple Channel: the workers are paid 0.45 USD for annotating one Human Intelligent Task (HIT) that requires writing five statements. The workers are encouraged to produce plain statements meeting the requirements: (i) corresponding to a single row/record in the table without involving too much symbolic reasoning. (ii) mention the cell values without dramatic modification. The average annotation time of a HIT is 4.2 min.

High-Reward Complex Channel: the workers are paid 0.75 USD for annotating a HIT (five statements). They are guided to produce more sophisticated statements to meet the requirements: (i) involving multiple rows in the tables with higher-order semantics like argmax, argmin, count, difference, average, summarize, etc. (ii) rephrase the table records to involve more semantic understanding. The average annotation time of a HIT is 6.8 min. The data obtained from the complex channel are harder in terms of both linguistic and symbolic reasoning, the goal of the two-channel split is to help us understand the proposed models can reach under different levels of difficulty.

2.2 NEGATIVE REWRITING STRATEGY

As suggested in (Zellers et al., 2018), there might be annotation artifacts and conditional stylistic patterns such as length and word-preference biases, which can allow shallow models (e.g. bag-of-words) to obtain artificially high performance. Therefore, we design a negative rewriting strategy to minimize such linguistic cues or patterns. Instead of letting the annotators write negative statements from scratch, we let them rewrite the collected entailed statements. During the annotation, the workers are explicitly guided to modify the words, phrases or sentence structures but retain the sentence style/length to prevent from artificial cues. We disallow naive negations by adding "not, never, etc" to revert the statement polarity in case of obvious linguistic patterns.

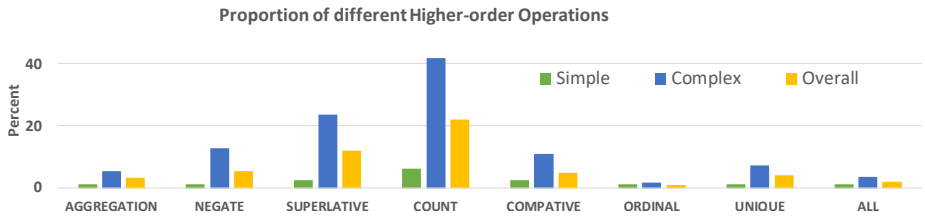

Figure 2: Proportion of different higher-order operations on the simple/complex channel statements including aggregation, unique, etc.

Channel	#Sentence	#Table	Len(Ent)	Len(Ref)	Split	#Sentence	Table	Row	Col
Simple	50,244	9,189	13.2	13.1	Train	92,283	13,182	14.1	5.5
Complex	68,031	7,392	14.2	14.2	Val	12,792	1,696	14.0	5.4
Total	118,275	16,573	13.8	13.8	Test	12,779	1,695	14.2	5.4

Table 1: Basic statistics of the data collected from the simple/complex channel and the division of Train/Val/Test Split in the dataset, where "Len" denotes the averaged sentence length.

2.3 DATASET STATISTICS

Inter-Annotator Agreement: After the data collection pipeline, we further perform quality control to filter 18% entailed of entailed statements and 27% refuted statements, we merged the instances from two different channels to obtain a diverse yet clean dataset for table-based fact verification. We sample 1000 annotated (table, statement) pairs and re-distribute each to 5 individual workers to re-label them as either `ENTAILED` or `REFUTED`. We follow the previous works (Thorne et al., 2018; Bowman et al., 2015) to adopt the Fleiss Kappa (Fleiss, 1971) as an indicator, where Fleiss $\kappa = \frac{\bar{p_c}-\bar{p_e}}{1-\bar{p_e}}$ is computed from from the observed agreement $\bar{p_c}$ and the agreement by chance $\bar{p_e}$. We obtain a Fleiss $\kappa = 0.75$, which indicates strong inter-annotator agreement and good-quality.

Dataset Statistics: As shown in Table 1, the amount of data harvested via the complex channel slightly outnumbers the simple channel, the averaged length of both the positive and negative samples are indistinguishable. More specifically, to analyze to which extent the higher-order operations are included in two channels, we group the common higher-order operations into 8 different categories. As shown in Figure 2, we sample 200 sentences from two different channels to visualize their distribution. We can see that the complex channel overwhelms the simple channel in terms of the higher-order logic, among which, count and superlatives are the most frequent. We split the whole data roughly with 8:1:1 into train, validation, and test splits and show their statistics in Table 1. Each table with an average of 14 rows and 5-6 columns corresponds to 2-20 different statements, while each cell has an average of 2.1 words. In the training split, the positive instances slightly outnumber the negative instances, while the validation and test split both have rather balanced distributions over positive and negative instances.

3 MODELS

With the collected dataset, we now formally define the table-based fact verification task: the dataset is comprised of triple instances $(\mathbf{T}, S, L)$ consisting of a table $\mathbf{T}$, a natural language statement $S = s_1, \cdots, s_n$ and a verification label $L \in \{0, 1\}$. The table $\mathbf{T} = \{T_{i,j} | i \leq R_T, j \leq C_T\}$ has R_T rows and C_T columns with the T_{ij} being the content in the (i, j)-th cell. T_{ij} could be a word, a number, a phrase or even a natural language sentence. The statement S describes a fact to be verified against the content in the table $\mathbf{T}$. If it is entailed by $\mathbf{T}$, then $L = 1$, otherwise the label $L = 0$. Figure 1 shows some entailed and refuted examples. During training, the model and the learning algorithm are presented with K instances like $(\mathbf{T}, S, L)_{k=1}^{K}$ from the training split. In the testing stage, the model is presented with $(\mathbf{T}, S)_{k=1}^{K'}$ and supposed to predict the label as $\hat{L}$. We measure the performance by the prediction accuracy $Acc = \frac{1}{K'} \sum_{1}^{K'} \mathbb{I}(\hat{L}_k = L_k)$ on the test set. Before building the model, we first perform entity linking to detect all the entities in the statements. Briefly, we first lemmatize the words and search for the longest sub-string matching pairs between statements and table cells/captions, where the matched phrases are denoted as the linked entities. To focus on statement verification against the table, we do not feed the caption to the model and simply mask the phrases in the statements which links to the caption with placeholders. The details of the entity linker are listed in the Appendix. We describe our two proposed models as follows.

3.1 LATENT PROGRAM ALGORITHM (LPA)

In this approach, we formulate the table fact verification as a program synthesis problem, where the latent program algorithm is not given in TABFACT. Thus, it can be seen as a weakly supervised learning problem as discussed in Liang et al. (2017); Lao et al. (2011). Under such a setting, we propose to break down the verification into two stages: (i) latent program search, (ii) discriminator ranking. In the first program synthesis step, we aim to parse the statement into LISP-like program format to represent its semantics. We define the plausible API set to include roughly 50 different functions like *min, max, count, average, filter, and* and realize their interpreter with Python-Pandas. Each API is defined to take arguments of specific types (*number, string, bool and view (e.g sub-table)*) to output specific-type variables. During the program execution, we store the generated intermediate variables to different-typed caches $\mathcal{N}, \mathcal{R}, \mathcal{B}, \mathcal{V}$ (Num, Str, Bool, View). At each execution step, the program can fetch the intermediate variable from the caches to achieve semantic compositionality. In order to shrink the search space, we follow NSM (Liang et al., 2017) to use trigger words to prune the API set and accelerate the search speed. The definitions of all API, trigger words can be found in the Appendix. The comprehensive the latent program search procedure is summarized in Algorithm 1, and the searching procedure is illustrated in Figure 3.

After we collected all the potential program candidates $\mathcal{P} = \{(P_1, A_1), \cdots, (P_n, A_n)\}$ for a given statement S (where (P_i, A_i) refers to i-th candidate) . we need to learn a discriminator to identify the "appropriate" traces from the set from many erroneous and spurious traces. Since we do not have the ground truth label about such discriminator, we use a weakly supervised training algorithm by viewing all the label-consistent programs as positive instances $\{P_i | (P_i, A_i); A_i = L\}$ and the label-inconsistent program as negative instances $\{P_i | (P_i, A_i); A_i \neq L\}$ to minimize the cross-entropy of discriminator $p_\theta(S, P)$ with the weakly supervised label. Specifically, we build our discriminator with a Transformer-based two-way encoder (Vaswani et al., 2017), where the

Algorithm 1 Latent Program Search with Comments

Initialize Number Cache $\mathcal{N}$, String Cache $\mathcal{R}$, Bool Cache $\mathcal{B}$, View Cache $\mathcal{V} \rightarrow \emptyset$
Push linked numbers, strings from the given statement S into $\mathcal{N}, \mathcal{R}$, and push $\mathbf{T}$ into $\mathcal{V}$
Initialize the result collector $\mathcal{P} \rightarrow \emptyset$ and an empty program trace $P = \emptyset$
Initialize the Queue $\mathcal{Q} = [(P, \mathcal{N}, \mathcal{R}, \mathcal{B}, \mathcal{V})]$, we use $\mathcal{Q}$ to store the intermediate states
Use trigger words to find plausible function set $\mathcal{F}$, for example, $more$ will trigger $Greater$ function.
while loop over time $t = 1 \rightarrow$ MAXSTEP **do**:
    while $(P, \mathcal{N}, \mathcal{R}, \mathcal{B}, \mathcal{V}) = \mathcal{Q}.pop()$ **do**:
        while loop over function set $f \in \mathcal{F}$ **do**:
            if arguments of f are in the caches **then**
                Pop out the required arguments $arg_1, arg_2, \cdots, arg_n$ for different cachess.
                Execute $A = f(arg_1, \cdots, arg_n)$ and concatenate the program trace P.
                if Type(A)=Bool **then**
                if $\mathcal{N} = \mathcal{S} = \mathcal{B} = \emptyset$ **then**
                $\mathcal{P}.push((P, A))$ # The program P is valid since it consumes all the variables.
                $P = \emptyset$ # Collect the valid program P into set $\mathcal{P}$ and reset P
                else
                $\mathcal{B}.push(A)$ # The intermediate boolean value is added to the bool cache
                $\mathcal{Q}.push((P, \mathcal{N}, \mathcal{R}, \mathcal{B}, \mathcal{V}))$ # Add the refreshed state to the queue again
                if Type(A) $\in$ {Num, Str, View} **then**
                if $\mathcal{N} = \mathcal{S} = \mathcal{B} = \emptyset$ **then**
                $P = \emptyset$;break # The program ends without consuming the cache, throw it.
                else
                push A into $\mathcal{N}$ or $\mathcal{S}$ or $\mathcal{V}$ # Add the refreshed state to the queue for further search
                $\mathcal{Q}.push((P, \mathcal{N}, \mathcal{R}, \mathcal{B}, \mathcal{V}))$
Return the triple $(\mathbf{T}, S, \mathcal{P})$ # Return (Table, Statement, Program Set)

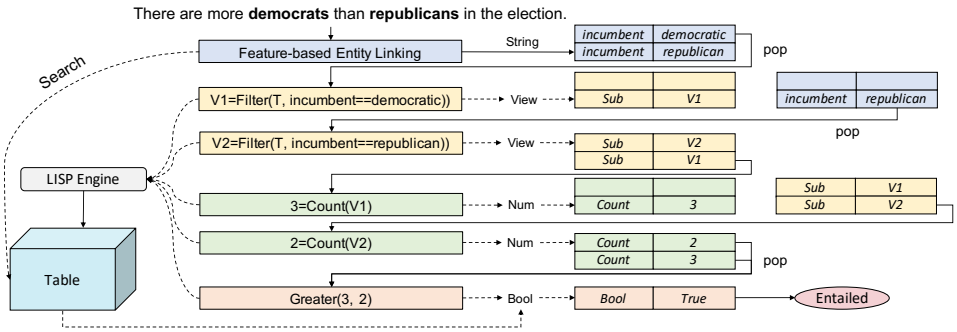

Figure 3: The program synthesis procedure for the table in Figure 1. We link the entity (e.g. *democratic*, *republican*), and then composite functions on the fly to return the values from the table.

statement encoder encodes the input statement S as a vector $Enc^S(S) \in \mathbb{R}^{n \times D}$ with dimension D, while the program encoder encodes the program $P = p_1, \cdots, p_m$ as another vector $Enc^P(P) \in \mathbb{R}^{m \times D}$, we concatenate these two vectors and feed it into a linear projection layer to compute $p_\theta(S, P) = \sigma(v_p^T[Enc^S(S); Enc^P(P)])$ as the relevance between S and P with weight $v_p \in \mathbb{R}^D$. At test time, we use the discriminator p_θ to assign confidence $p_\theta(S, P)$ to each candidate $P \in \mathcal{P}$, and then either aggregate the prediction from all hypothesis with the confidence weights or rank the highest-confident hypothesis and use their outputs as the prediction.

3.2 TABLE-BERT

In this approach, we view the table verification problem as a two-sequence binary classification problem like NLI or MPRC (Wang et al., 2018) by linearizing a table $\mathbf{T}$ into a sequence and treating the statement as another sequence. Since the linearized table can be extremely long surpassing the limit of sequence models like LSTM, Transformers, etc. We propose to shrink the sequence by only retaining the columns containing entities linked to the statement to alleviate such memory issue. In order to encode such sub-table as a sequence, we propose two different linearization methods, as is depicted in Figure 4. (i) Concatenation: we simply concatenate the table cells with [SEP] tokens

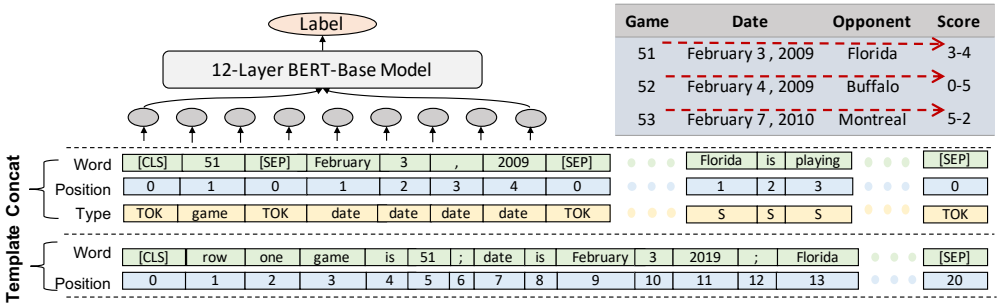

Figure 4: The diagram of Table-BERT with horizontal scan, two different linearizations are depicted.

in between and restart position counter at the cell boundaries; the column name is fed as another type embedding to the input layer. Such design retains the table information in its machine format. (ii) Template: we adopt simple natural language templates to transform a table into a "somewhat natural" sentences. Taking the horizontal scan as an example, we linearize a table as "row one's game is 51; the date is February; ..., the score is 3.4 (ot). row 2 is ...". The isolated cells are connected with punctuations and copula verbs in a language-like format.

After obtaining the linearized sub-table $\tilde{\mathbf{T}}$, we concatenate it with the natural language statement S and prefix a [CLS] token to the sentence to obtain the sequence-level representation $H = f_{BERT}([\tilde{\mathbf{T}}, S])$, with $H \in \mathbb{R}^{768}$ from pre-trained BERT (Devlin et al., 2019). The representation is further fed into multi-layer perceptron f_{MLP} to obtain the entailment probability $p_\theta(\tilde{\mathbf{T}}, S) = \sigma(f_{MLP}(H))$, where σ is the sigmoid function. We finetune the model θ (including the parameters of BERT and MLP) to minimize the binary cross entropy $\mathcal{L}(p_\theta(\tilde{\mathbf{T}}, S), L)$ on the training set. At test time, we use the trained BERT model to compute the matching probability between the (table, statement) pair, and classify it as `ENTAILED` statement when $p_\theta(\tilde{\mathbf{T}}, S)$ is greater than 0.5.

4 EXPERIMENTS

In this section, we aim to evaluate the proposed methods on TABFACT. Besides the standard validation and test sets, we also split the test set into a simple and a complex partition based on the channel from which they were collected. This facilitates analyzing how well the model performs under different levels of difficulty. Additionally, we also hold out a small test set with 2K samples for human evaluation, where we distribute each (table, statement) pair to 5 different workers to approximate human judgments based on their majority voting, the results are reported in Table 2.

Model	Val	Test	Test (simple)	Test (complex)	Small Test
BERT classifier w/o Table	50.9	50.5	51.0	50.1	50.4
Table-BERT-Horizontal-F+T-Concatenate	50.7	50.4	50.8	50.0	50.3
Table-BERT-Vertical-F+T-Template	56.7	56.2	59.8	55.0	56.2
Table-BERT-Vertical-T+F-Template	56.7	57.0	60.6	54.3	55.5
Table-BERT-Horizontal-F+T-Template	66.0	65.1	79.0	58.1	67.9
Table-BERT-Horizontal-T+F-Template	**66.1**	**65.1**	**79.1**	**58.2**	**68.1**
LPA-Voting w/o Discriminator	57.7	58.2	68.5	53.2	61.5
LPA-Weighted-Voting	62.5	63.1	74.6	57.3	66.8
LPA-Ranking w/ Transformer	**65.2**	**65.0**	**78.4**	**58.5**	**68.6**
Human Performance	-	-	-	-	**92.1**

Table 2: The results of all proposed models, the numbers are reported in percentage. T+F means table followed by fact, while F+T means fact followed by table.

Table-BERT We build Table-BERT based on the open-source implementation of BERT[4] using the pre-trained model with 12-layer, 768-hidden, 12-heads, and 110M parameters trained in 104 lan-

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

guages. We use the standard BERT tokenizer to break the words in both statements and tables into subwords and join the two sequences with a [SEP] token in between. The representation corresponding to [CLS] is fed into an MLP layer to predict the verification label. We finetune the model on a single TITAN X GPU with a mini-batch size of 6. The best performance is reached after about 3 hours of training (around 10K steps). We implement and compare the following variants of the Table-BERT model including (i) Concatenation vs. Template: whether to use natural language templates during linearization. (ii) Horizontal vs. Vertical: scan direction in linearization.

LPA We run the latent program search in a distributed fashion on three 64-core machines to generate the latent programs. The search terminates once the buffer has more than 50 traces or the path length is larger than 7. The average search time for each statement is about 2.5s. For the discriminator model, we design two transformer-based encoders (3 layers, 128-dimension hidden embedding, and 4 heads at each layer) to encode the programs and statements, respectively. The variants of LPA models considered include (i) Voting: assign each program with equal weight and vote without the learned discriminator. (ii) Weighted-Voting: compute a weighted-sum to aggregate the predictions of all latent programs with the discriminator confidence as the weights. (iii) Ranking: rank all the hypotheses by the discriminator confidence and use the top-rated hypothesis as the output.

Preliminary Evaluation In order to test whether our negative rewriting strategy eliminates the artifacts or shallow cues, we also fine-tune a pre-trained BERT (Devlin et al., 2019) to classify the statement S without feeding in table information. The result is reported as "BERT classifier w/o Table" in Table 2, which is approximately the majority guess and reflects the effectiveness of the rewriting strategy. Before presenting the experiment results, we first perform a preliminary study to evaluate how well the entity linking system, program search, and the statement-program discriminator perform. Since we do not have the ground truth labels for these models, we randomly sample 100 samples from the dev set to perform the human study. For the entity linking, we evaluate the precision of correctly linked entities and the recall of entities which should be linked. For latent program search, we evaluate whether the "true" programs are included in the candidate set $\mathcal{P}$ and report the recall score. For discriminator, under the cases where the "true" program lies in the candidate set, we use the trained model to select the top K hypothesis and calculate the HITS@K accuracy (the chance of correct program being included in the top K candidates). Please note that the discriminator can also select a spurious program which happens to obtain the same label as ground truth, but this does not count as a hit. These preliminary case study results are reported in Table 3.

Steps	Prec%	Rec%	F1%	Discriminator	HITS@1	HITS@3	HITS@5
Entity Linking	83	81	82	LSTM	17	24	29
Systematic Search	-	77	-	Transformer	19	28	32

Table 3: Case Study results on different components, including the entity linking accuracy, systematic search recall, and discriminator accuracy.

Results We report the performance of different methods as well as human performance in Table 2. First of all, we observe that the naive serialized model fails to learn anything effective (same as the Majority Guess). It reveals the importance of template when using the pre-trained BERT (Devlin et al., 2019) model: the "natural" connection words between individual cells is able to unleash the power of the large pre-trained language model and enable it to perform reasoning on the structured table form. Such behavior is understandable given the fact that BERT is pre-trained on purely natural language corpora. In addition, we also observe that the horizontal scan excels the vertical scan because it better captures the convention of human expression. Among different LPA methods, we found that LPA-Ranking performs the best since it can better suppress the spurious programs than the voting-based algorithm. As suggested in Table 3, the current LPA method is upper bounded by 77% (recall of "true" program hypothesis), but the real accuracy (65%) is still far from that. Diving into specific cases to examine the performance of discriminator, we found that only 17% "true" programs are ranked at the top Table 3. We hypothesize that the weakly supervised learning of the discriminator is the main bottleneck for LPA. By comparing the performance of simple-channel with complex-channel split, we observe a significant accuracy drop ($\approx$ 20%), which reveals the weakness of existing models in dealing with higher-ordered semantics.

Besides, we observe that Table-BERT exhibits instability during training, after the model achieves the reported ceiling performance, it will gradually degrade to random guess for the following epochs.

Additionally, it also exhibits poor consistency during evaluation as it can miss some very simple cases but hit super hard test cases. These two major weaknesses are yet to be solved in the future study. In contrast, LPA behaves much more consistently and provides a clear latent rationale for its decision. But, such a pipeline system requires laborious handcrafting of API operations and is also very sensitive to the entity linking accuracy. Both methods have merits and weaknesses; how to combine the strengths of these two models still remains an open question.

5 RELATED WORK

Natural Language Inference: Modeling reasoning and inference in human language is a fundamental and challenging problem towards true natural language understanding. There has been extensive research on RTE in the early years (Dagan et al., 2005) and more recently shifted to NLI (Bowman et al., 2015; Williams et al., 2017). NLI seeks to determine whether a natural language hypothesis h can be inferred from a natural language premise p. With the surge of deep learning, there have been many powerful algorithms like the Decomposed Model (Parikh et al., 2016), Enhanced-LSTM (Chen et al., 2017) and BERT (Devlin et al., 2019). Our proposed fact verification task is also closely related to these inference tasks, where our semi-structured table can be seen as a collection of "premises" exhibited in a structured format. Our proposed problem hence could be viewed as the generalization of NLI under the structured domain.

Table Question Answering: Another line of research closely related to our task is the table-based question answering, such as WikiTableQuestion (Pasupat & Liang, 2015), Sequential Q&A (Iyyer et al., 2017), and WikiSQL (Zhong et al., 2017), for which approaches have been extended to handle large-scale resources like Wikipedia (Bhagavatula et al., 2013). However, in these Q&A tasks, the question types typically provide strong signals needed for identifying the type of answers, while TABFACT does not provide such specificity. Moreover, TABFACT involves stronger rephrase from the semi-structured table contents: the statements usually reformulate the original cell values to fit them smoothly into a human-readable natural language sentence. For example, if the cell contains more than one entries like "john, peter", people usually translate them into "peter and john", or "john ... with peter". Such common rephrasing makes the semantic matching (entity linking) much more challenging than Q&A. Furthermore, TABFACT has One-to-Many mappings; statements might not be explicitly transformed into one Q&A pair. For example, a sentence like "Legace played for the St. Louis Blues against Anaheim at home on February 22nd, in the next year, and the record between the two teams becomes 24197." is hard to be parsed into one Q&A pair, rather it can correspond to different possibilities like "Who plays for St. ..... 24-19-7?" or "Which team does Legace play for ...?", etc. These factors greatly increase fact verification's difficulty in terms of understanding.

Program Synthesis for Q&A: There have also been great interests in using program synthesis or logic forms to solve the table question answering problem. They aim to retrieve answers by synthesizing programs and executing them on the tables, such as in Neural Programmer (Neelakantan et al., 2016; 2017) and Neural Symbolic Machines (Liang et al., 2017; 2018; Agarwal et al., 2019). Compared to the table-based Q&A, our proposed TABFACT exhibits even more challenging spurious programs (Berant et al., 2013; Pasupat & Liang, 2015) (i.e., wrong programs with the true returned answers) issue due to the fact that the program can only return binary values, which can easily misguide the policy search using standard reinforcement learning. How to resolve such extremely under-specified binary rewards in NLP domain becomes an interesting direction to pursue. Besides, the previously defined API sets are not enough because the verification task requires additional binary-typed operations, which also greatly enlarges the search space.

6 CONCLUSION

This paper investigates a very important yet previously unexplored research problem: semi-structured fact verification. We construct a large-scale dataset and proposed two methods, Table-BERT and LPA, based on the state-of-the-art pre-trained natural language inference model and program synthesis. In the future, we plan to push forward this research direction by inspiring more sophisticated architectures which can perform both linguistic and symbolic reasoning.

REFERENCES

Rishabh Agarwal, Chen Liang, Dale Schuurmans, and Mohammad Norouzi. Learning to generalize from sparse and underspecified rewards. *International Conference of Machine Learning*, 2019.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pp. 1533–1544, 2013.

Chandra Sekhar Bhagavatula, Thanapon Noraset, and Doug Downey. Methods for exploring and mining tables on wikipedia. In *Proceedings of the ACM SIGKDD Workshop on Interactive Data Exploration and Analytics*, pp. 18–26. ACM, 2013.

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 632–642, 2015.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. Enhanced lstm for natural language inference. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1657–1668, 2017.

Gennaro Chierchia and Sally McConnell-Ginet. *Meaning and grammar: An introduction to semantics*. MIT press, 2000.

Ido Dagan, Oren Glickman, and Bernardo Magnini. The pascal recognising textual entailment challenge. In *Machine Learning Challenges Workshop*, pp. 177–190. Springer, 2005.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *Proceedings of NAACL-HLT*, 2019.

Joseph L Fleiss. Measuring nominal scale agreement among many raters. *Psychological bulletin*, 76(5):378, 1971.

Andreas Hanselowski, Hao Zhang, Zile Li, Daniil Sorokin, Benjamin Schiller, Claudia Schulz, and Iryna Gurevych. Ukp-athene: Multi-sentence textual entailment for claim verification. *arXiv preprint arXiv:1809.01479*, 2018.

Mohit Iyyer, Wen-tau Yih, and Ming-Wei Chang. 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)*, volume 1, pp. 1821–1831, 2017.

Sujay Kumar Jauhar, Peter Turney, and Eduard Hovy. Tables as semi-structured knowledge for question answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pp. 474–483, 2016.

Jerrold J Katz and Jerry A Fodor. The structure of a semantic theory. *language*, 39(2):170–210, 1963.

Ni Lao, Tom Mitchell, and William W Cohen. Random walk inference and learning in a large scale knowledge base. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pp. 529–539. Association for Computational Linguistics, 2011.

Chen Liang, Jonathan Berant, Quoc Le, Kenneth D Forbus, and Ni Lao. Neural symbolic machines: Learning semantic parsers on freebase with weak supervision. *International Conference of Machine Learning*, 2017.

Chen Liang, Mohammad Norouzi, Jonathan Berant, Quoc V Le, and Ni Lao. Memory augmented policy optimization for program synthesis and semantic parsing. In *Advances in Neural Information Processing Systems*, pp. 9994–10006, 2018.

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

Arvind Neelakantan, Quoc V Le, and Ilya Sutskever. Neural programmer: Inducing latent programs with gradient descent. *International Conference on Learning Representation*, 2016.

Arvind Neelakantan, Quoc V Le, Martin Abadi, Andrew McCallum, and Dario Amodei. Learning a natural language interface with neural programmer. *International Conference on Learning Representation*, 2017.

Ankur Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. A decomposable attention model for natural language inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 2249–2255, 2016.

Panupong Pasupat and Percy Liang. Compositional semantic parsing on semi-structured tables. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pp. 1470–1480, 2015.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proceedings of NAACL-HLT*, pp. 2227–2237, 2018.

Kashyap Popat, Subhabrata Mukherjee, Jannik Strötgen, and Gerhard Weikum. Where the truth lies: Explaining the credibility of emerging claims on the web and social media. In *Proceedings of the 26th International Conference on World Wide Web Companion*, pp. 1003–1012. International World Wide Web Conferences Steering Committee, 2017.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 1:8, 2019.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. Fever: a large-scale dataset for fact extraction and verification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pp. 809–819, 2018.

Johan Van Benthem et al. *A brief history of natural logic*. LondonCollege Publications9781904987444, 2008.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pp. 5998–6008, 2017.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. *EMNLP 2018*, pp. 353, 2018.

Adina Williams, Nikita Nangia, and Samuel R Bowman. A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv:1704.05426*, 2017.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 2019.

Rowan Zellers, Yonatan Bisk, Roy Schwartz, and Yejin Choi. Swag: A large-scale adversarial dataset for grounded commonsense inference. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 93–104, 2018.

Victor Zhong, Caiming Xiong, and Richard Socher. Seq2sql: Generating structured queries from natural language using reinforcement learning. *arXiv preprint arXiv:1709.00103*, 2017.

A APPENDIX

A.1 FUNCTION DESCRIPTION

We list the detailed function description in Figure 5.

Type	Overrides	arg0 (? Means Dependency)	arg1	arg2	arg3	Output
Relationship	Relate	?	Relation	-	-	Region
	Relate_with_name	?	Relation	Object	-	
	Relate_invese	?	Relation	-	-	
	Relate_inverse_with_name	?	Relation	Object	-	
	Relate_with_same_attribute	?	Relation	Attribute	-	
Selection	Select	-	Object	-	-	Region
Filter	Filter_horizontal_position	?	H-Position	-	-	Region
	Filter_Vertical_position	?	V-Position	-	-	
	Filter_with_color	?	Color	-	-	
	Filter_with_shape	?	Shape	-	-	
	Filter_with_activity	?	Activity	-	-	
	Filter_with_material	?	Material	-	-	
	Filter_with_color_noteq	?	Color	-	-	
	Filter_with_shape_noteq	?	Shape	-	-	
Choose	Choose_name	?	Name1	Name2	-	Answer
	Choose_scene	-	Scene1	Scene2	-	
	Choose_color	?	Color1	Color2	-	
	Choose_shape	?	Shape1	Shape2	-	
	Choose_horizontal_position	?	H-Position1	H-Position2	-	
	Choose_vertical_position	?	V-Position1	V-Position2	-	
	Choose_relation_name	?	Relation1	Relation2	Name	
	Choose_relation_inverse_name	?	Relation1	Relation2	Name	
	Choose_younger	?	?	-	-	
	Choose_older	?	?	-	-	
	Choose_healthier	?	?	-	-	
	Choose_less_healthier	?	?	-	-	
Verify	Verify_color	?	Color	-	-	Answer
	Verify_shape	?	Shape	-	-	
	Verify_scene	-	Scene	-	-	
	Verify_relation_name	?	Relation	Name	-	
	Verify_relation_inv_name	?	Relation	Name	-	
Query	Query_name	?	-	-	-	Answer
	Query_color	?	-	-	-	
	Query_shape	?	-	-	-	
	Query_scene	-	-	-	-	
	Query_horizontal_position	?	-	-	-	
	Query_vertical_position	?	-	-	-	
Common	Common_color	[? ? .. ?]	-	-	-	Answer
	Common_material	[? ? .. ?]	-	-	-	
Different	Different_name	[? ? .. ?]	-	-	-	Answer
	Different_name	?	?	-	-	
	Different_color	?	?	-	-	
Same	Same_name	[? ? .. ?]	-	-	-	Answer
	Same_name	?	?	-	-	
	Same_color	?	?	-	-	
And	And	?	?	-	-	Answer
Or	Or	?	?	-	-	Answer
Exist	Exist	?	?	-	-	Answer

Figure 5: The overall architecture of our proposed dynamic module network

B HIGHER-ORDER OPERATIONS

1. Aggregation: the aggregation operation refers to sentences like "the averaged age of all ....", "the total amount of scores obtained in ...", etc.
2. Negation: the negation operation refers to sentences like "xxx did not get the best score", "xxx has never obtained a score higher than 5".
3. Superlative: the superlative operation refers to sentences like "xxx achieves the highest score in", "xxx is the lowest player in the team".
4. Comparative: the comparative operation refers to sentences like "xxx has a higher score than yyy".
5. Ordinal: the ordinal operation refers to sentences like "the first country to achieve xxx is xxx", "xxx is the second oldest person in the country".
6. Unique: the unique operation refers to sentences like "there are 5 different nations in the tournament, ", "there are no two different players from U.S"
7. All: the for all operation refers to sentences like "all of the trains are departing in the morning", "none of the people are older than 25."
8. None: the sentences which do not involve higher-order operations like "xxx achieves 2 points in xxx game", "xxx player is from xxx country".

C WHETHER TO KEEP WIKIPEDIA CONTEXT

Before crowd-sourcing the annotation for the tables, we observed that the previous WikiTableQuestion Pasupat & Liang (2015) provides context (Wikipedia title) during annotation while the WikiSQL Zhong et al. (2017) does not. Therefore, we particularly design ablation annotation tasks to compare the annotation quality between w/ and w/o Wikipedia title as context. We demonstrate a typical example in Figure 6, where a Wiki table[5] aims to describe the achievements of a tennis player named Dennis, but itself does not provide any explicit hint about "Tennis Player Dennis". Unsurprisingly, the sentence fluency and coherence significantly drops without such information. Actually, a great portion of these Wikipedia tables requires background knowledge (like sports, celebrity, music, etc) to understand. Therefore, we argue that such a context is necessary for annotators to understand the background knowledge to write more fluent sentences. On the other end, we also hope to minimize the influence of the textual context in the table-based verification task, therefore, we design an annotation criterion: the Wikipedia title is provided to the workers during the annotation, but they are explicitly banned bring any unrelated background information other than the title into the annotation. As illustrated in Figure 6, the title only acts as a placeholder in the statements to make it sound more natural.

outcome	year	championship	surface	partner
winner	1960	Wimbledon championships	grass	Rafael Osuna
winner	1961	US Championships	grass	Chuck Mckinley
runner - up	1962	US Championships	grass	Chuck Mckinley
winner	1963	US Championships (2)	grass	Chuck Mckinley

Context (Title)	*Richard Dennis Ralston (born July 27, 1942, an American former tennis player*	*No Information is provided*
Annotate	From 1960 to 1969, Ralston won five major double championships.	Winner is on the grass surface. Rafael Osuna is partner in the Wimbeldon

Figure 6: Comparison of worker annotation w/ and w/o Wikipedia title as context

[5] https://en.wikipedia.org/wiki/Dennis_Ralston

D ENTITY LINKING

Here we propose to use the longest string match to find all the candidate entities in the table, when multiple candidates coexist, we select the one with the minimum edit distances. The visualization is demonstrated in Figure 7.

District	Incumbent	Party	Result	Candidates
California 3	**John E. Moss**	democratic	re-elected	John E. Moss (d) 69.9% John Rakus (r) 30.1%
California 5	Phillip Burton	democratic	re-elected	Phillip Burton (d) 81.8% Edlo E. Powell (r) 18.2%
california 8	George Paul Miller	democratic	lost renomination democratic hold	Pete Stark (d) 52.9% Lew M. Warden , Jr. (r) 47.1%
California 14	Jerome R. Waldie	republican	re-elected	Jerome R. Waldie (d) 77.6% Floyd E. Sims (r) 22.4%
California 15	John J. Mcfall	republican	re-elected	John J. Mcfall (d) unopposed

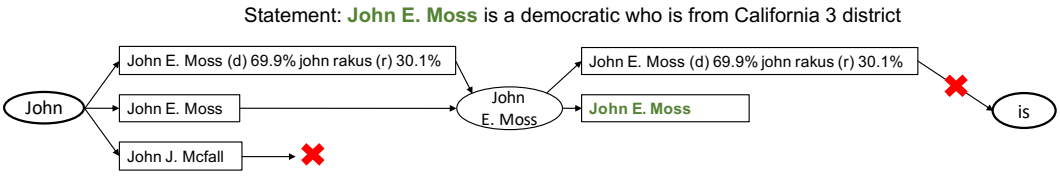

Figure 7: Entity Linking System.

E THE PROGRAM CANDIDATES

Here we demonstrate some program candidates in Figure 8, and show how our proposed discriminator is designed to compute the matching probability between the statement and program. Specifically, we employ two transformer-based encoder Vaswani et al. (2017), the left one is aimed to encode the program sequence and the right one is aimed to encode the statement sequence. Their output from [CLS] position are concatenated and fed into an MLP to classify the verification label.

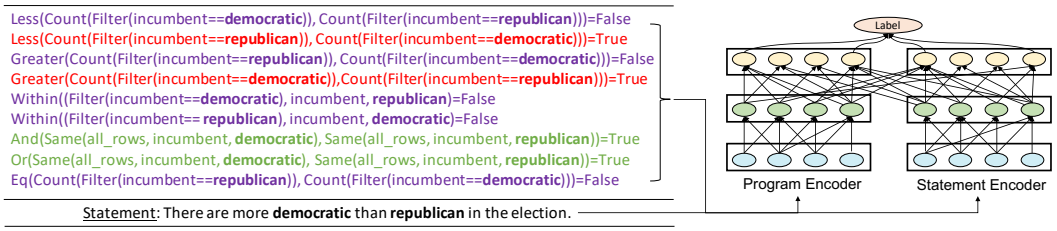

Figure 8: We demonstrate the top program candidates from the example and use discriminator to rank them.