Exploring Decomposition for Table-based Fact Verification

Xiaoyu Yang and Xiaodan Zhu
Ingenuity Labs Research Institute & ECE, Queen’s University, Canada
{xiaoyu.yang, xiaodan.zhu}@queensu.ca

Abstract

Fact verification based on structured data is challenging as it requires models to understand both natural language and symbolic operations performed over tables. Although pre-trained language models have demonstrated a strong capability in verifying simple statements, they struggle with complex statements that involve multiple operations. In this paper, we improve fact verification by decomposing complex statements into simpler subproblems. Leveraging the programs synthesized by a weakly supervised semantic parser, we propose a program-guided approach to constructing a pseudo dataset for decomposition model training. The subproblems, together with their predicted answers, serve as the intermediate evidence to enhance our fact verification model. Experiments show that our proposed approach achieves the new state-of-the-art performance, an 82.7% accuracy, on the TABFACT benchmark.

1 Introduction

Fact verification aims to validate if a statement is entailed or refuted by given evidence. It has become crucial to many applications such as detecting fake news and rumor (Rashkin et al., 2017; Thorne et al., 2018; Goodrich et al., 2019; Vaibhav et al., 2019; Kryscinski et al., 2020). While existing research mainly focuses on verification based on unstructured text (Hanselowski et al., 2018; Yoneda et al., 2018; Liu et al., 2020; Nie et al., 2019), a recent trend is to explore structured data as evidence, which is ubiquitous in our daily life.

Verification performed with structured data presents research challenges of fundamental interests, as it involves both informal inference based on language understanding and symbolic operations such as mathematical operations (e.g., count and max). While all statements share the same set of operations, complex statements, which involve multiple operations, are more challenging than simple statements. Pre-trained models such as BERT (Devlin et al., 2019) have presented superior performances on verifying simple statements while still struggling with complex ones: a performance gap exists between the simple and complex tracks (Chen et al., 2020).

In this paper, we propose to decompose complex statements into simpler subproblems to improve table-based fact verification, as shown in a simplified example in Figure 1. To avoid manually annotating gold decompositions, we design a program-guided pipeline to collect pseudo decompositions for training generation models by distinguishing four major decomposition types and designing templates accordingly. The programs we used are parsed from statements with a weakly supervised parser with the training signals from final verification labels. Figure 1 shows a statement-program example. We adapt table-based natural language understanding systems to solve the decomposed subproblems. After obtaining the answers to subproblems, we combine them in a pairwise manner as intermediate evidence to support the final prediction.

We perform experiments on the recently proposed benchmark TABFACT (Chen et al., 2020)
and achieve a new state-of-the-art performance, an 82.7% accuracy. Further studies have been conducted to provide details on how the proposed models work.

2 Method

2.1 Task Formulation and Notations

Given an evidence table $T$ and a statement $S$, we aim to predict whether $T$ entails or refutes $S$, denoted by $y \in \{0, 1\}$. For each statement $S$, the executable program derived from a semantic parser is denoted as $z$. An example of program is given in Figure 1. Each program $z = \{a_i\}_i=1^M$ consists of multiple symbolic operations $op_i$, and each operation contains an operator (e.g., max) and arguments (e.g., all_rows and attendance). A complex statement $S$ can be decomposed into subproblems $D = \{d_i\}_i=1^N$, with the answers being $\{a_i\}_i=1^N$. Using combined problem-answer pairs as intermediate evidence $E = \{e_i\}_i=1^N$ where $e_i = (d_i, a_i)$, our model maximizes the objective $\log p(y|T, S, E)$.

2.2 Statement Decomposition

Constructing a high-quality dataset is key to the decomposition model training. Since semantic parsers can map statements into executable programs that not only capture the semantics but also reveal the compositional structures of the statements, we propose a program-guided pipeline to construct a pseudo decomposition dataset.

2.2.1 Constructing Pseudo Decompositions

Program Acquisition. Following Chen et al. (2020), we use latent program algorithm (LPA) to parse each statement $S$ into a set of candidate programs $Z = \{z_i\}_i=1^K$. To select the most semantically consistent program $z^*$ among all candidates and mitigate the impact of spurious programs, we follow Yang et al. (2020) to optimize the program selection model with a margin loss, which is detailed in Appendix A.1.

By further removing programs that are label-inconsistent or cannot be split into two isolated sub-programs from the root operator, we obtain the remaining $(T, S, z)$ triples as the source of data construction.

Decomposition Templates. Programs are formal, unambiguous meaning representations for the corresponding statements. Designed to support automated inference, the program $z$ encodes the central feature of the statement $S$ and reveals its compositional structures. Our statement decomposition is based on the structure of the program. Specifically, we first extract program skeleton $z_s$ by omitting arguments in the selected program $z$, then we group the $(T, S, z)$ triples by $z_s$ to identify four major decomposition types: conjunction\(^2\), comparative, superlative, and uniqueness.

Some simple templates associated with each decomposition type are designed, which contain instructions on how to decompose the statement, and this manual process only takes a few hours. In this way, we can construct pseudo decompositions, including sub-statements and sub-questions, by filling the slots in templates according to the original statements or program arguments. Templates and decomposition examples can be found in Figure 2. Each sample in our constructed pseudo dataset is denoted as a $(S, c, D')$ triple, where $c$ indicates one of the four types and $D'$ is a sequence of pseudo decompositions.

Data Augmentation. With the $(T, S, z)$ triples, we perform data augmentation. Since some entity mentions in $S$ and $z$ can be linked to cells in $T$, we can randomly replace the linked entities in $S$ and $z$ with different values in the same column of $T$. For example, in Figure 1, we can replace the linked

\(^2\)The conjunction type has overlap with the other three types in the cases that the sub-statements connected by conjunctions can be further decomposed.
entity “firhill” with another randomly selected entity “cappielow”. Another augmentation strategy is inverting superlative and comparative. For the examples belong to superlative and comparative, we replace the original superlative or comparative in statements with its antonym, such as higher \( \rightarrow \) lower and longest \( \rightarrow \) shortest. In this way, we generate another \( 3k \) pseudo statement-decomposition pairs. In total, the final decomposition dataset used for generation model training includes \( 9,696 \) samples. More statistics are available in Appendix A.2.

### 2.2.2 Learning to Decompose

**Decomposition Type Detection.** Given a statement \( S \), we train a five-way classifier based on BERT to identify whether the statement is decomposable and if yes, which decomposition type it belongs to. In addition to the four types mentioned in the previous section, we add an atomic category by involving additional non-decomposable samples. Only the statements not assigned with atomic labels can be used for decomposition.

**Decomposition Model.** We finetune the GPT-2 (Radford et al., 2019) on the pseudo dataset for decomposition generation. Specifically, given the \( (S, c, D') \) triple, we train the model by maximizing the likelihood \( J = \log p_\theta(D'|S, c) \). We provide the model with gold decomposition type \( c \) during training and the predicted type \( \hat{c} \) during testing. Only informative and well-formed decompositions are involved in the subsequent process to enhance the downstream verification. In case some sub-statements need further decomposition, it can be implemented by resending them to our pipeline\(^3\).

### 2.3 Solving Subproblems

We adapt TAPAS (Eisenschlos et al., 2020), a SOTA model on table-based fact verification and QA task, to solve the decomposed subproblems. Verifying sub-statements is formulated as a binary classification with the TAPAS model fine-tuned on the TAB-FACT (Chen et al., 2020) dataset. To answer each sub-question, we use the TAPAS finetuned on WikiTableQuestions (Pasupat and Liang, 2015) dataset. We combine the subproblems and their answers in a pairwise manner to obtain the intermediate evidence \( E = \{e_i\}_i \) and \( \{(d_i, a_i)\}_i \), an example evidence is shown in Figure 1.

\(^3\)In most cases, there is no need to perform iterative decomposition, and we leave finer-grained decomposition for future research.

### 2.4 Recombining Intermediate Evidence

Downstream tasks can utilize the intermediate evidence in various ways. In this paper, we train a model to fuse the evidence \( E \) together with the statement \( S \) and table \( T \) for table-based fact verification\(^4\). Specifically, we jointly encode \( S \) and \( T \) with TAPAS to obtain the concentrated representation \( h_{ST} \). We encode multiple evidence sentences with another TAPAS following the document-level encoder proposed in Liu and Lapata (2019) by inserting [CLS] token at the beginning of every single sentence \( e_i \) and taking the corresponding [CLS] embedding \( h_{e_i} \) in the final layer to represent \( e_i \).

We employ a gated attention model to obtain aggregated evidence representation \( h_{evd} \) and predict the final label as follows:

\[
h_{evd} = \sum_{i=0}^N a_i h_{e_i}, \quad a_i = \sigma(h_{ST}^T h_{e_i})
\]

\[
y = \sigma(W([h_{evd} \oplus h_{ST}])))
\]

where \( W \) are trainable parameters, \( \sigma \) is the sigmoid function, and \( \oplus \) indicates concatenation.

### 3 Experiments

**Setup.** We conduct our experiments on a large-scale table-based fact verification benchmark TAB-FACT (Chen et al., 2020). The test set contains a simple and complex subset according to difficulty. A small test set is further annotated with human performance. Following the previous work, we use accuracy as the evaluation metric. Details of the data are listed in Appendix A.3.

**Implementation Details.** During fine-tuning the GPT-2 model to generate decomposition, we run the model with a batch size of 5 for 30 epochs using Adam optimizer (Kingma and Ba, 2015) with a learning rate of 2e-6. We optimize the model for final verification prediction using Adam optimizer with a learning rate of 2e-5 and a batch size of 16. It usually takes 11 to 14 epochs to converge. Our code is available at https://github.com/arielsho/Decomposition-Table-Reasoning.

**Main Results.** We compare our model with different baselines on TAB-FACT, including LPA (Chen et al., 2020), Table-BERT (Chen et al., 2020), LogicalFactChecker (Zhong et al., 2020), HeterTFV (Shi et al., 2020), SAT (Zhang et al., 2020), HeterTFV (Shi et al., 2020), SAT (Zhang et al., 2020)
Table 1: The accuracy (%) of models on TABFACT.

<table>
<thead>
<tr>
<th>Model</th>
<th>Val</th>
<th>Test</th>
<th>Simple</th>
<th>Complex</th>
<th>Small</th>
</tr>
</thead>
<tbody>
<tr>
<td>Human</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>92.1</td>
</tr>
<tr>
<td>LPA</td>
<td>57.7</td>
<td>58.2</td>
<td>68.5</td>
<td>53.2</td>
<td>61.5</td>
</tr>
<tr>
<td>Table-BERT</td>
<td>66.1</td>
<td>65.1</td>
<td>79.1</td>
<td>58.2</td>
<td>68.1</td>
</tr>
<tr>
<td>LogicalFactChecker</td>
<td>71.8</td>
<td>71.7</td>
<td>85.4</td>
<td>65.1</td>
<td>74.3</td>
</tr>
<tr>
<td>HeterTFV</td>
<td>72.5</td>
<td>72.3</td>
<td>85.9</td>
<td>65.7</td>
<td>74.2</td>
</tr>
<tr>
<td>SAT</td>
<td>73.3</td>
<td>73.2</td>
<td>85.5</td>
<td>67.2</td>
<td></td>
</tr>
<tr>
<td>ProgVGAT</td>
<td>74.9</td>
<td>74.4</td>
<td>88.3</td>
<td>67.6</td>
<td>76.2</td>
</tr>
<tr>
<td>TAPAS-BASE</td>
<td>79.1</td>
<td>79.1</td>
<td>91.4</td>
<td>73.1</td>
<td>81.2</td>
</tr>
<tr>
<td>TAPAS-LARGE</td>
<td>81.5</td>
<td>81.2</td>
<td>93.0</td>
<td>75.5</td>
<td>84.1</td>
</tr>
<tr>
<td>OURS-BASE</td>
<td>80.8</td>
<td>80.7</td>
<td>91.9</td>
<td>75.1</td>
<td>82.5</td>
</tr>
<tr>
<td>OURS-LARGE</td>
<td>82.7</td>
<td>82.7</td>
<td>93.6</td>
<td>77.4</td>
<td>84.7</td>
</tr>
</tbody>
</table>

Table 2: Decompositions improve the performance on test set over 4 decomposition types.

<table>
<thead>
<tr>
<th>Type</th>
<th>TAPAS-BASE</th>
<th>OURS-BASE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conj.</td>
<td>79.9</td>
<td>82.6</td>
</tr>
<tr>
<td>Sup.</td>
<td>81.3</td>
<td>82.4</td>
</tr>
<tr>
<td>Comp.</td>
<td>69.1</td>
<td>72.1</td>
</tr>
<tr>
<td>Uniq.</td>
<td>70.4</td>
<td>74.4</td>
</tr>
<tr>
<td>Atomic</td>
<td>81.7</td>
<td>82.5</td>
</tr>
</tbody>
</table>

Table 3: Evaluation of decomposition quality.

<table>
<thead>
<tr>
<th>Our Decomp.</th>
<th>BLEU-4 on Dev</th>
<th>Human Val</th>
</tr>
</thead>
<tbody>
<tr>
<td>w/o data aug</td>
<td>48.42</td>
<td>56%</td>
</tr>
<tr>
<td>w/o type info</td>
<td>54.74</td>
<td>63%</td>
</tr>
</tbody>
</table>

Table 4: Percentage of valid decomposition on all splits in TABFACT.

<table>
<thead>
<tr>
<th></th>
<th>train</th>
<th>val</th>
<th>test</th>
<th>simple</th>
<th>complex</th>
</tr>
</thead>
<tbody>
<tr>
<td>Our Decomp. w/o data aug</td>
<td>41.6</td>
<td>46.3</td>
<td>46.7</td>
<td>20.2</td>
<td>59.5</td>
</tr>
<tr>
<td></td>
<td>35.2</td>
<td>39.1</td>
<td>39.4</td>
<td>16.3</td>
<td>50.7</td>
</tr>
</tbody>
</table>

4 Related Work

Existing work on fact verification is mainly based on evidences from unstructured text (Thorne et al., 2018; Hanselowski et al., 2018; Yoneda et al., 2018; Thorne et al., 2019; Nie et al., 2019; Liu et al., 2020). Our work focuses on fact verification based on structured tables (Chen et al., 2020). Unlike the previous work (Chen et al., 2020; Zhong et al., 2020; Shi et al., 2020; Zhang et al., 2020; Yang et al., 2020; Eisenschlos et al., 2020), we propose a framework to verify statements via decomposition.

Sentence decomposition takes the form of Split-and-Rephease proposed by Narayan et al. (2017) to split a complex sentence into a sequence of shorter sentences while preserving original meanings (Ahaaroni and Goldberg, 2018; Botha et al., 2018; Guo et al., 2020). In QA task, question decomposition has been applied to help answer multi-hop ques-
We thank the anonymous reviewers for their insightful comments. We also thank Yufei Feng for his helpful comments and suggestions on the paper writing.

Acknowledgements

We thank the anonymous reviewers for their insightful comments. We also thank Yufei Feng for his helpful comments and suggestions on the paper writing.

References


A Appendix

A.1 Program Selection

We fine-tune the BERT (Devlin et al., 2019) to model $p_\theta(z|S)$, the probability of program $z$ being semantically consistent with $S$. Since the gold programs are not available, we use the final verification labels as weak supervision. To mitigate the impact of spurious programs, i.e., programs execute to correct answers with incorrect operation combinations, we follow Yang et al. (2020) to optimize the model with a margin loss:

$$J = \max \left( p(z^-|S) - p(z^+|S) + \gamma, 0 \right)$$

where $z^-$ and $z^+$ denote the label-inconsistent and label-consistent programs with the highest probability, respectively. $\gamma$ is the parameter to control the margin. The margin loss can encourage selecting one program that is most semantically relevant to the statement while maintaining a margin between the positive (label-consistent) and the negative (label-inconsistent) programs.

A.2 Statistics of Pesudo Dataset

We have 9,696 pseudo statement-decomposition pairs in total, and the number of samples belong to four decomposition types is given in Table 5. To train the decomposition type detection model, we add an additional atomic category with 1,739 statements.

<table>
<thead>
<tr>
<th>Decomp. Type</th>
<th># of samples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conjunctive</td>
<td>1,798</td>
</tr>
<tr>
<td>Superlative</td>
<td>2,452</td>
</tr>
<tr>
<td>Comparative</td>
<td>4,528</td>
</tr>
<tr>
<td>Uniqueness</td>
<td>918</td>
</tr>
</tbody>
</table>

Table 5: Statistics of pseudo decomposition dataset.

A.3 Statistics of TABFACT Dataset

The statistics of TABFACT (Chen et al., 2020) can be found in Table 6, a large-scale table-based fact verification benchmark dataset on which we evaluate our method. The test set is further split into a simple set and a complex set, which include 4,171 and 8,608 sentences, respectively. A small test set with 1,998 samples are provided for human performance evaluation.

<table>
<thead>
<tr>
<th>Split</th>
<th>Sentence</th>
<th>Table</th>
<th>Row</th>
<th>Col</th>
</tr>
</thead>
<tbody>
<tr>
<td>Train</td>
<td>92,283</td>
<td>13,182</td>
<td>14.1</td>
<td>5.5</td>
</tr>
<tr>
<td>Val</td>
<td>12,792</td>
<td>1,696</td>
<td>14.0</td>
<td>5.4</td>
</tr>
<tr>
<td>Test</td>
<td>12,779</td>
<td>1,695</td>
<td>14.2</td>
<td>5.4</td>
</tr>
</tbody>
</table>

Table 6: Statistics of TABFACT.

A.4 Compared Systems

- **LPA** (Chen et al., 2020) derives a program for each statement by ranking the synthesized program candidates and takes the program execution results as predictions.
- **Table-BERT** (Chen et al., 2020) takes a linearized table and a statement as the input of BERT for fact verification.
- **LogicalFactChecker** (Zhong et al., 2020) utilizes the structures of programs to prune irrelevant information in tables and modularize symbolic operations with module networks.
- **HeterTFV** (Shi et al., 2020) is a graph-based reasoning approach to combining linguistic information and symbolic information.
- **SAT** (Zhang et al., 2020) is a structure-aware Transformer that encodes structured tables by injecting the structural information into the mask of the self-attention layer.
- **ProgVGAT** (Yang et al., 2020) leverages the symbolic operation information to enhance verification with a verbalization technique and a graph-based network.
- **TAPAS** (Herzig et al., 2020; Eisenschlos et al., 2020) is the previous SOTA model on TABFACT which extends BERT’s architecture to encode tables and is jointly pre-trained with text and tables.