Accelerating Tabular Inference: Training Data Generation with TENET

Enzo Veltri
enzo.veltri@unibas.it
University of Basilicata
Potenza, Italy

Donatello Santoro
donatello.santoro@unibas.it
University of Basilicata
Potenza, Italy

Jean-Flavien Bussotti
jflavien.bussotti@gmail.com
EURECOM
Biot, France

Paolo Papotti
paolo.papotti@eurecom.fr
EURECOM
Biot, France

ABSTRACT

Tabular Natural Language Inference (TNLI) involves machine learning models that assess whether structured tabular data supports or contradicts a hypothesis formulated in natural language. TNLI models typically require large sets of training examples, which are costly to produce manually. In this demonstration, we present TENET, a system for the automatic generation of training examples for TNLI applications. Existing TNLI training approaches either depend on costly human annotation or generate simplistic examples that lack data diversity and complex reasoning. In contrast, TENET can start from a small set of manually annotated examples to automatically generate a large and diverse training dataset. TENET is based on the idea that SQL queries are the right tool for obtaining rich and complex generated examples. To ensure data variety, evidence-queries extract cell values from tables based on diverse data patterns. Once the relevant data are identified, semantic queries define different ways to interpret it using SQL clauses. These interpretations are then verbalized as text to create annotated examples for TNLI. This demonstration offers an interactive experience where users will be able to select evidence from tabular data, inspect and refine generated queries, and observe how TENET transforms structured data into natural language hypotheses. By engaging with different scenarios, users will see how TENET enables the rapid creation of high-quality TNLI datasets, leading to inference models with performance comparable to those trained on manually crafted examples.

PVLDB Reference Format:
Enzo Veltri, Donatello Santoro, Jean-Flavien Bussotti, and Paolo Papotti. Accelerating Tabular Inference:
Training Data Generation with TENET. PVLDB, 18(12): 5303 - 5306, 2025.
doi:10.14778/3750601.3750657

PVLDB Artifact Availability:
Source code, data, and/or other artifacts: https://github.com/dbunibas/tenet.

Proceedings of the VLDB Endowment, Vol. 18, No. 12 ISSN 2150-8097.
doi:10.14778/3750601.3750657

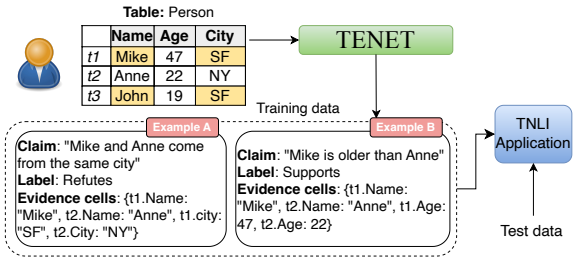

Figure 1: Given a table and selected cells, TENET creates training examples for TNLI: Example A's hypothesis is contradicted by the data; Example B's is supported.

1 INTRODUCTION

Recently, a new class of applications has emerged in Natural Language Processing (NLP) that focus on inference with structured data as evidence, i.e., *tabular natural language inference* (TNLI), which includes tasks such as computational fact-checking [3–5].

The most effective TNLI solutions are supervised and rely on manually curated datasets [1, 3]. However, these datasets present three major limitations. (i) They primarily cover generic topics using tables from Wikipedia, making them unsuitable for domains that fall outside its coverage. (ii) Their scale and variety are significantly smaller than those available for textual Natural Language Inference. For instance, approximately 80% of the examples in ToTTo (a widely used TNLI dataset) describe tabular data without incorporating mathematical expressions such as max, min, count, or value comparisons. (iii) They contain biases and inaccuracies that can negatively affect TNLI model learning.

The problem of the lack of labeled examples has been treated in the literature for NLI. If some examples are given (*warm start* setting), existing NLI augmentation methods can be used in the TNLI setting: the text part of the example can be rewritten with augmentation w.r.t. the (fixed) data. While these methods increase the number of training examples, they do not enhance the variety and complexity of the examples. Ultimately, this results in a minimal impact on the accuracy of a TNLI task.

In this demonstration, we present a novel GUI built on top of TENET [2], enabling the rapid creation of ad-hoc training datasets for TNLI model training. Figure 1 provides an overview of our method.

Starting with an existing table, the user selects a set of cells, named evidence[1]. TENET then uses this evidence and its relationships within the table to generate a training set suitable for training an inference model. Our GUI helps the user navigate the three key steps of generating a TNLI training example.

Data Evidence. One intuition behind TENET is that tabular data inherently contains rich information suitable for generating new training examples. The user begins by selecting an initial set of seed cells, such as the yellow-highlighted cells in the Person table in Figure 1. Based on these seed cells, TENET generalizes the underlying pattern by generating evidence queries (*e-queries*). These queries identify additional sets of cells that follow the same pattern within the table. Each extracted set of cells serves as evidence to generate a TNLI training example. For instance, the system automatically extracts cells *Mike*, *Anne*, *SF*, and *NY*, which are then used as evidence to generate Example A. This automated process ensures that generated examples maintain logical consistency with the original pattern. If more tables are available, the same e-queries are applied to them to generate examples at scale.

Textual Hypothesis. Once the data is identified, the next step is to generate a textual statement (or *hypothesis*) for the annotated example. Given a set of cells, we create semantic queries (*s-queries*) that select the data evidence identified by the e-queries. Indeed, a set of cells can be identified by multiple s-queries, e.g., an s-query with a condition p1.age>p2.age or p1.city<>p2.city. Our intuition is that each s-query *describes* the evidence using specific conditions (e.g., selections with constants) or constructs (e.g., aggregates). Given an s-query, we use it to generate a textual hypothesis through a prompting method that leverages the human-like generation capabilities of large language models (LLMs). This approach ensures a diverse set of factual hypotheses — such as *'Mike is older than Anne'* in Example B in Figure 1 — while maximizing evidence coverage and minimizing hallucinations. The GUI facilitates the exploration of s-queries applicable to the generated evidence. By default, all s-queries are executed for a given set of cells to get several examples.

Inference Label. Each generated example requires an associated label. While *Supports* examples naturally arise when the hypothesis aligns with the evidence in the table, generating *Refutes* examples requires a controlled injection of errors into the data evidence. For instance, in Figure 1, we change the CITY value of t_2 from "NY" to "SF". Applying text generation to this "noisy" table produces hypotheses that contradict the original clean data, such as *'Mike and Anne come from the same city'*.

This demonstration extends the original system by introducing *i*) an intuitive user interface that enables the fast generation of large training datasets with minimal user effort; *ii*) a method for users to explore and validate the generated training dataset to improve its quality; *iii*) extended support for both closed and open LLMs. In the following, we first describe our solution (Section 2). Then, we illustrate how users engage with TENET, using pre-loaded tables or providing new ones (Section 3). We also demonstrate how the generated training data improves the quality of target applications.

[1]User selection is not needed in the system, as automatic selection leads to high quality examples [2]. Human interaction is enabled for a better demo experience.

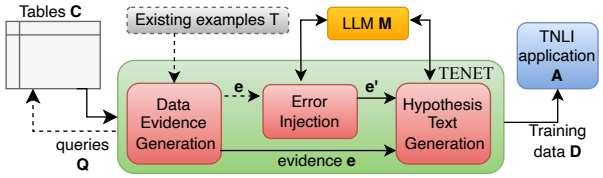

Figure 2: TENET overview: Existing examples are optional. Supports any LLM and TNLI task (e.g., tabular fact-checking).

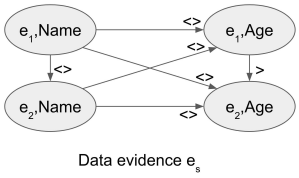

Figure 3: Evidence graphs derived from Example A in Fig. 1.

2 SYSTEM OVERVIEW

Problem Formulation. A *textual hypothesis* is a sentence in natural language. A Tabular Natural Language Inference (TNLI) application takes as input a pair (table c; textual hypothesis h) and outputs if h is supported or refuted by c. *Data evidence* is a non-empty subset of cell values from c, ranging from a small fraction [1] to the entire table [3]. TNLI solutions rely on supervised models trained on annotated examples. Our goal is to automate the generation of this training data, reducing the need for costly manual annotation.

We address the *example generation problem* for a TNLI application A, given the label space L, a corpus of tables C, and optionally, a set of training examples T for A. Each example consists of a quadruple (h, l, e, c) with textual hypothesis h, label $l \in L$, set of data evidence cells e contained in one relational table c in C. We assume access to a large language model (LLM) M. We do not assume access to the TNLI application A at hand.

In this demonstration, we present first the *warm start* version of the problem, where some training examples for A are available and used by TENET. These examples may originate from an existing annotated dataset or be provided by the user. In the *cold start* setting, in which we do not assume available examples T, we allow the users to select initial seed cells via the GUI.

Process. TENET is designed around three main steps, as depicted in Figure 2. Given a relational table $c \in C$, it first gathers the evidence e to produce a Supports example. Second, to generate a Refutes example, it injects errors in table c to create its noisy version and derive data evidence e'. Third, a textual claim (hypothesis) h is generated for every data evidence e. The quadruple (data evidence e, textual claim h, label Supports/Refutes, table c) is a complete example for training data D for the target TNLI application A.

Data Evidence Generation. Our intuition is that every example contains data evidence e_i reflecting a human-defined pattern. Users can define data evidence directly via the GUI if no examples are available. Capturing these patterns allows us to efficiently derive additional sets of evidence. Given an existing example from T, we refer to it as the *seed* s. A complete example comes with its label l_s,

the evidence set e_s, a textual hypothesis h_s, and the table used to verify it c_s. In the case of the demo, only e_s is provided. Given the set of cell values e_s and table c_s as input, we identify the *evidence query* (or *e-query*) q that outputs such e_s among its results. The e-query q is generated through an evidence graph [2]. Each node in the graph corresponds to a cell from e_s, and a (directed) edge across two nodes represents the relationship between their values, e.g., equality, difference, greater than/less than. An example graph derived from data evidence e_s with tuples (Mike, 47), (Anne, 22) is in Figure 3. The e-query q can be derived from the evidence graph by associating every tuple id across the nodes in the graph to a tuple variable in q, and exploiting the edges relationship in the WHERE clause. From the evidence graph in Figure 3, we generate e-query q:

```
q: SELECT c1.Name, c2.Name as Name2, c1.Age, c2.Age as Age2
    FROM people c1, people c2
    WHERE c1.Age > c2.Age AND c1.Name <> c2.Name
```

Executing e-query q over the original table c_s, we obtain more data evidence $e_1, \ldots, e_n$ following the data pattern in e_s. Assume another table d_{new} has schema: *player(name, weight, height)*. As Name and Age from Figure 1 match the data types of player.name and player.weight, respectively, the system generates e-query q':

```
q':SELECT c1.name, c2.name as name2, c1.weight, c2.weight as weight2
   FROM player c1, player c2
   WHERE c1.weight > c2.weight AND c1.name <> c2.name
```

Query q' selects cells from the new table *player* with the same pattern of example t to generate new training examples. This method generates multiple new e-queries, depending on the number of available tables and compatible attributes.

In the *cold start* scenario, where no initial examples are provided, Tenet enables users to directly select initial seed cells via the GUI. These seeds are used to automatically derive meaningful evidence queries, ensuring effective bootstrapping and the rapid generation of training examples even without any pre-existing labeled data

Textual Hypothesis Generation. Given a set of cells e, we want to generate a textual hypothesis that describes them (*data-to-text* generation). Data evidence e can be described by several SQL queries that identify it in the table. These queries are alternative ways to describe the data. By computing such queries, we immediately obtain a *semantic* characterization that can be used to generate different textual hypotheses. Given a table c and data evidence e, a *semantic query* (or s-query) over c returns exactly e. The goal is to achieve diversity in queries for the same set of cells, in terms of variety at the intensional level (over the data description). We identify several s-queries that could be discovered by Tenet [2]. For example, an s-query q_s^1 extends q by adding constraints in the WHERE clause, such as c1.Name='Mike' AND c2.Name='Anne'. The purpose of q_s^1 is to compare two individuals based on their age, with the condition `c1.Age > c2.Age` encoding the semantic meaning that Mike is older than Anne. By further adding the constraints `c1.Age > 19 AND c2.Age > 19` to q_s^1, we obtain the s-query q_s^2, which refines the comparison by filtering the individuals considered to only those older than 19 (where 19 is derived from the data). Moreover, Tenet supports s-queries that take into account the selected cells in relation to the entire dataset, allowing the formulation of *global* hypotheses over the data. A single evidence could generate more s-queries. Once the s-queries for each evidence e have been identified, we use the discovered semantic associated with each query to prompt the LLM M to generate the textual hypothesis. The prompt contains the evidence e and the semantics to generate the textual hypothesis thanks to the conditions in the WHERE clause. Figure 4 contains more examples obtained with s-queries.

Evidence: Cells for {Name, Age}	**Evidence:** Cells for {Name, Age, City}
Sentences:	**Sentences:**
• *John is the youngest*	• *There are two persons from SF and one from NY*
• *Mike is the oldest*	• *Persons from SF on average are older than persons from NY*
• *The average age is 29.3*	
Label: Support	**Label:** Support

Figure 4: Supports examples generated from data in Figure 1.

Label Generation. By construction, the evidence from the generated data reflects the semantics expressed in the input table. Such evidence leads to an example with a Supports label w.r.t. the data in the table. However, applications also need examples with a Refutes label, i.e., textual claims not supported by the input. We tackle this problem through an error injection approach, perturbing the input table to disrupt the original relationships among the cell values. This new version is then used to identify a set of evidence again e' through the same e-query q, which leads to a textual hypothesis that does not reflect the semantics of the original (clean) table.

3 DEMONSTRATION

The demonstration is structured in two parts. First, we generate examples for TNLI applications using the GUI. We then show how such examples improve result quality in a target application.

3.1 Example Generation

At the beginning of the demo, the visitors will load an existing scenario or create a new one with their own dataset.

Step 1. Figure 5 shows the evidence selection and new evidence generation steps. The visitor, after loading the dataset, selects a set of cells as evidence e. In Figure 5(1.1), the selection contains (Mike, 47) and (John, 59) from two tuples. Tenet generates the e-query q and allows the user to set the number of Supports (Refutes) new evidence to generate as shown in Figure 5(1.2).

Step 2. The visitor inspects the new data evidence generated using q through tabs. Figure 6 (2.1) shows a new evidence generated with different cells for a Support example: (Anne, 22) and (John, 19). Figure 6 (2.2) shows a evidence generated for a Refutes example. Indeed, John's age has been changed from 59 to 47.

Step 3. Given the selected evidence, the visitor can apply one of the possible s-queries available for such evidence. By default, all s-queries that apply are executed to get more examples. Figure 6 (3) reports a generated example: the user selected, as a s-query, a comparison over the attribute age. Then the user selected the number of sentences to generate and obtained the textual hypotheses. Both generated sentences (*'John is younger than Mike as Mike's age is 48 and John's age is 47'*, *'John is younger than Mike'*) are Refutes examples w.r.t. the original data in Figure 5.

Step 4. Finally, the visitor can export the generated examples as a training dataset for the target application. We also provide a Python API to automatically generate the training set without using the user interface.

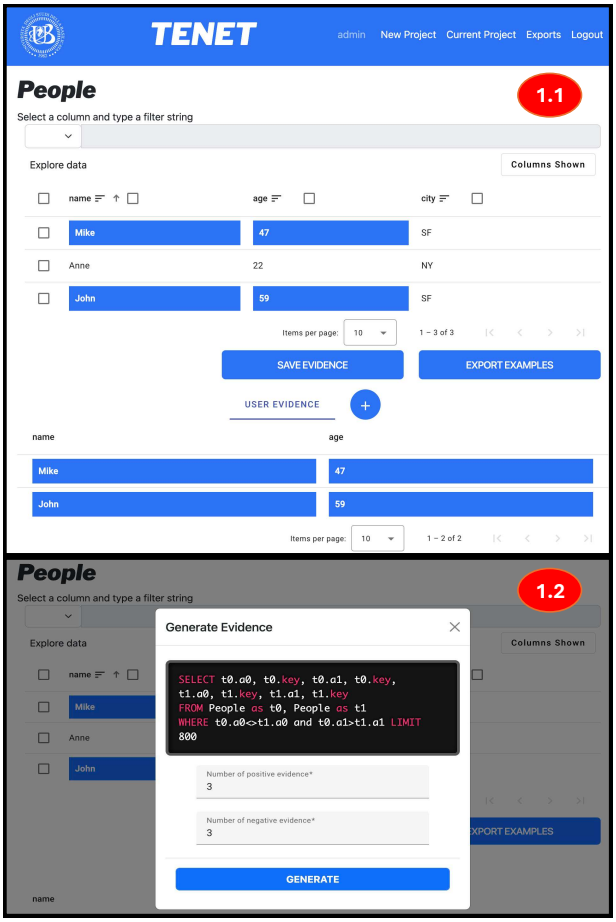

Figure 5: The user selects a set of cells (1.1) and TENET generates the evidence query (1.2). Positive (negative) evidence is used for Supports (Refutes) training examples.

In the current implementation, TENET supports both GPT and any open model as LLM for the text generation step. During the demonstration, the visitor could select the LLM and experiment with the differences in the generated sentences.

3.2 Target Application: Fact-Checking

We present an experimental evaluation of the generated training examples on Feverous [1], a fact-checking dataset. The original dataset contains training and test sets of manually written examples. Every example consists of a *table*, a *textual hypothesis*, the *data evidence* (a subset of the table), and a Supports/Refutes *label*. For each example, we generate the evidence query e_q from the data evidence. Then we select all the tables in Feverous with the same attribute names used in e_q, we name them *compatible*. We apply e_q to every *compatible* table and we use the subsequent steps in the TENET pipeline. We select 100 training examples from Feverous and train the end-to-end Feverous baseline [1] achieving an accuracy of 0.597. Next, we apply our approach by deriving e-queries from the original evidence and applying them to *compatible* tables, automatically generating approximately 4,000 additional training

Figure 6: The user could inspect the generated new evidences. 2.1 (2.2) reports a new positive (negative) evidence. The user applies one of the proposed s-queries to generate the text (3).

examples. Training the same end-to-end Feverous baseline on the expanded dataset yields an accuracy of 0.644, representing an 8% improvement with no additional user effort. The same experiment can be reproduced with the examples generated starting from the data annotations from the audience during the demo.

This use case also shows that TENET efficiently scales with the number of tables by using SQL query generation and execution. SQL enables the system to handles large volumes of data and varied schemas, limiting the interaction with LLMs to their inference for sentence generation.

The demonstration further illustrates TENET 's scalability to complex table schemas and its adaptability across diverse domains. Users can explore additional scenarios, such as generating TNLI examples from financial statements (comparing revenues across periods) or medical databases (verifying patient clinical records). This flexibility underscores Tenet's broad applicability and value for different types of structured datasets.

REFERENCES

[1] Rami Aly, Zhijiang Guo, Michael Sejr Schlichtkrull, James Thorne, Andreas Vlachos, Christos Christodoulopoulos, Oana Cocarascu, and Arpit Mittal. 2021. FEVEROUS: Fact Extraction and VERification Over Unstructured and Structured information. In *NeurIPS (Datasets and Benchmarks)*. 1–13.

[2] Jean-Flavien Bussotti, Enzo Veltri, Donatello Santoro, and Paolo Papotti. 2023. Generation of Training Examples for Tabular Natural Language Inference. *Proc. ACM Manag. Data* 1, 4 (2023), 27.

[3] Vivek Gupta, Maitrey Mehta, Pegah Nokhiz, and Vivek Srikumar. 2020. INFOTABS: Inference on Tables as Semi-structured Data. In *ACL*. Association for Computational Linguistics, 2309–2324.

[4] Georgios Karagiannis, Mohammed Saeed, Paolo Papotti, and Immanuel Trummer. 2020. Scrutinizer: A Mixed-Initiative Approach to Large-Scale, Data-Driven Claim Verification. *Proc. VLDB Endow.* 13, 11 (2020), 14.

[5] Preslav Nakov, David Corney, Maram Hasanain, Firoj Alam, Tamer Elsayed, Alberto Barrón-Cedeño, Paolo Papotti, Shaden Shaar, and Giovanni Da San Martino. 2021. Automated Fact-Checking for Assisting Human Fact-Checkers. In *IJCAI*.