UTOPIA: Automatic Pivot Table Assistant

Whanhee Cho
University of Utah
Salt Lake City, Utah, USA
whanhee@cs.utah.edu

Anna Fariha
University of Utah
Salt Lake City, Utah, USA
afariha@cs.utah.edu

ABSTRACT

Data summarization is required to comprehend large datasets, and aggregations are effective ways to summarize data. A pivot table is a mechanism to aggregate numerical attributes grouped by categorical attributes and spreadsheet pivot tables are particularly suitable for novice users, where they can rearrange, group, and aggregate data using intuitive interfaces. However, real-world spreadsheet data is often disorganized, with attributes having *multiple values* and *synonymous variants*. For instance, in the IMDb data, multiple genres can be stored as a comma-separated value ("Action, Comedy, Drama") or the genre "Science Fiction" can be represented in diverse ways: "Sci-Fi", "scifi", "Technological Fiction". Such data issues pose barriers for the novices while constructing pivot tables, and result in noisy and incomprehensible summarization. Parsing multi-valued attributes forces novices to resort to external tools (Power Query) or methods (Python), requiring additional expertise; and synonymous variants demand users to explore and modify variants manually.

We introduce UTOPIA an a**UTO**matic **PI**vot table **A**ssistant that extends the functionality of spreadsheet pivot tables, overcoming data issues such as multi-valued attributes and synonymous variants. To achieve that, UTOPIA (1) detects multi-valued attributes and automatically organizes the values, achieving *implicit data normalization*, (2) leverages Sentence-BERT to handle synonymous variants, enabling *semantic aggregation*, and (3) helps construct pivot tables without requiring additional expertise. We will demonstrate how UTOPIA enables easy and effective pivot table construction, relieving users from tedious data preprocessing while allowing them to remain in their familiar spreadsheet environment without requiring any external tools or additional expertise.

1 INTRODUCTION

The recent growth of data volume and data utilization in machine learning demand effective mechanisms for data analysis. Given the vast dimensions and complexities of data, analysts often turn to data summarization, such as using summary tables or charts where they can select attributes of interest to summarize over. Aggregation is widely acknowledged as one of the most effective summarization methods. This involves many operations that allow users to analyze data from diverse perspectives, such as "sum" over numerical attributes or "count" over categorical attributes.

Mechanisms for creating aggregations range from SQL, which requires expert proficiency, to spreadsheet pivot tables, accessible by novices. SQL is suitable for experts where (1) data must be clean and normalized, and stored in a relational format and (2) a query must be formed using aggregation operators (e.g., `SUM`, `AVG`, `CNT`). Alternatively, novices can utilize spreadsheet pivot table (Table 1(c)), which offers an intuitive and easy interface, to organize data by certain attributes and apply aggregates to other attributes of interest. In pivot tables, users can choose which attributes serve as "row" or "column".

Business Intelligence (BI) models such as Tableau[1] and Power BI[2], and spreadsheets such as Microsoft Excel[3] and Google Sheets[4] offer pivot table functionality. These tools allow users to use non-relational or "flat" data, and perform relatively straightforward data analysis. However, the burden is still on the user when it comes to deal with messy data that requires additional pre-processing. Despite being designed for novices, spreadsheet pivot tables bring adversities when the data is disorganized, preventing the users from obtaining the desired results directly.

EXAMPLE 1. *Patel, a novice data analyst, wants to analyze data for 1000 most popular movies from IMDb[5]. A sample of the dataset is shown in Table 1(a). Patel wants to know which film genres yield significant gross. She decides to use Microsoft Excel pivot table, using the attributes* `Genre` *and* `Gross`*, and expects to obtain the result shown in Table 1(c). However, to her disappointment, she gets the result shown in Table 1(b). The reason is that the attribute* `Genre` *contains multiple values for some movies, such as "action, crime, drama", and Excel incorrectly assumed that this entire comma-separated list is the value for genre.*

To get her desired result, Patel must parse the multi-valued attribute `Genre`*. This requires additional expertise such as using an advanced Excel functionality, or writing a Python script. However, Patel's challenge does not end just by being able to parse these comma-separated lists. She realizes that some movies belong to only one genre, whereas some have up to 10 different genres. So, she must figure out how to store such variable-length values against a single attribute* `Genre`*. One option is to derive 10 different attributes labeled by* `Genre1`*,* `Genre2`*, ...,* `Genre10`*, but this will result into additional issues. First, many cells will stay empty as not all movies have 10 genres, resulting in poor representation of the data. Second, she will still struggle to obtain her desired pivot table (Table 1(c)) as the target attribute is now split across 10 different attributes. Should she generate one pivot table for each of the newly generated attributes? That will result in 10 different pivot tables, which is not what Patel wants. While Microsoft Power Query[6] may offer a solution to this, Patel hesitates to leave her comfortable spreadsheet and move to a new tool.*

EXAMPLE 2. *Now let's assume that Patel successfully parsed the multi-valued attribute* `Genre` *using some external tool (Microsoft Power Query) and is now working on a different dataset about movies, as shown in Table 2(a), but over the same schema as before. She proceeds to generate a pivot table over this new dataset and obtains Table 2(b). Patel struggles to interpret the results as she*

[1]Tableau: `www.community.tableau.com/`
[2]PowerBI: `www.microsoft.com/en-us/power-platform/products/power-bi`
[3]Microsoft `www.microsoft.com/en-us/microsoft-365/excel`
[4]Google Sheets: `www.google.com/sheets/about/`
[5]IMDB: `www.kaggle.com/datasets/PromptCloudHQ/imdb-data`
[6]Microsoft Power Query `powerquery.microsoft.com/en-us/`

Title	Genre	Gross
Joker	drama	28 M
2001: A Space Odyssey	action, crime, drama	535 M
Queen	action, sci-fi	171 M
The Prestige	biography, drama	97 M
The Departed	action, sci-fi	293 M
The Usual Suspects	drama	37 M
Back to the Future	action, adventure	323 M

(a)

Genre	Sum of Gross
action, crime, drama	535 M
action, sci-fi	464 M
action, adventure	323 M
biography, drama	97 M
drama	65 M

(b)

Genre	Sum of Gross
action	1322 M
drama	697 M
crime	535 M
sci-fi	464 M
adventure	323 M
biography	97 M

(c)

Table 1: (a) An example dataset from IMDb. `Title` indicates title of a movie, `Genre` indicates the movie's genres, separated by commas, and `Gross` indicates the corresponding profit. (b) Since `Genre` contains multiple values, if the values are not parsed before pivot table construction, such ill-formed pivot table is produced. (c) The desired pivot table.

Title	Genre	Gross
The Shawshank Redemption	prison drama	28 M
The Dark Knight	superhero action, crime, epic drama	535 M
The Matrix	action, epic sci-fi	171 M
Schindler's List	biography	630 M
Inception	team action, space opera sci-fi	93 M
Fight Club	drama	37 M
Star Wars	epic action, space adventure	323 M

(a)

Genre	Sum of Gross
biography	630 M
epic drama	535 M
crime	535 M
superhero action	535 M
epic action	323 M
space adventure	323 M
action	171 M
...	...

(b)

Genre	Sum of Gross
action	1029 M
biography	630 M
drama	600 M
crime	535 M
adventure	323 M
sci-fi	264 M

(c)

Table 2: (a) An example dataset from IMDb. (b) Interpreting this pivot table is challenging due to the presence of synonymous variants in `Genre`, such as misspellings and sub-genres. (c) The expected pivot table.

expected "action" to be the top-gross genre, but the result indicates otherwise ("biography" seems to have a larger gross than "action").

Upon taking a closer look, she realizes that the data has different representations for the same genres, such as "action" as "superhero action", "team action", or "epic action". These synonymous variants also include typos, or sub-genres with identical semantics. The pivot table Patel desires is shown in Table 2(c). Patel realizes that to obtain the desired result, she has to replace all variants of similar semantics with a canonical value, which demands domain expertise and is an extremely tedious process.

Patel's examples show that generating a pivot table with disorganized values compels users to spend an enormous amount of time on data preprocessing. While some current tools offer mechanisms for parsing multi-valued attributes, they often force users to depart from the spreadsheet environment and resort to external tools demanding specialized knowledge. For instance, Microsoft Excel requires users to utilize Microsoft Power Query for parsing multiple values, Tableau requires users to use regex grammar for parsing data, and user forums frequently recommend addressing the issue through code implementation, such as spreadsheet functions. Using Python is an option, but it requires expertise on programming, which novices often lack. Even for this simple parsing problem, the implementation involves complex coding steps, as illustrated in Figure 1.

Handling synonymous variants requires domain knowledge and entails tedious work as users must explore and modify variants manually. In the case of large data, the amount of time users have to spend on this task becomes enormous. For instance, in Example 2, if there are on average 25 variants for each of the 20 genres, one will have to manually issue 500 find-replace operations. Moreover, this will fail if there are typos, case mismatches, or formatting issues. Additionally, synonymous variants mandate that users thoroughly

```python
import pandas as pd

# Load CSV file into a DataFrame
file_path = 'imdb_top_1000.csv'
df = pd.read_csv(file_path)

# Split Multi-Value Attribute
split_genres = df['Genre'].str.split(', ', expand=True)
num_attributes = split_genres.shape[1]

# Create new attribute names for parsed genre attributes
new_columns = [f'Genre{i+1}' for i in range(num_attributes)]
split_genres.columns = new_columns

# Concatenate the original dataframe with the parsed attributes
df = pd.concat([df, split_genres], axis=1)

# Drop the original Genre attribute
df.drop('Genre', axis=1)
```

Figure 1: Python script for parsing multiple value attributes

explore the entire data to determine which values have equivalent meanings. Another critical issue here is that such an operation is irreversible, i.e., the user will lose the information about variants once they perform they merge synonymous variants explicitly.

Therefore, an ideal pivot table functionality should be able to handle disorganized data gracefully and have three key features: (1) it should be able to extract values from multi-valued attributes, (2) it should be able to perform *semantic aggregation*, by grouping synonymous variants *implicitly*, without altering the data, and (3) it should let the user stay in their preferred spreadsheet environment.

We introduce UTOPIA (a**UTO**matic **PI**vot table **A**ssistant), which extends the functionality of spreadsheet pivot tables, overcoming data issues such as multi-valued attributes and synonymous variants. UTOPIA automatically parses and organizes values for multi-valued attributes, providing *implicit* data normalization. Additionally, UTOPIA enables *semantic aggregation* using Sentence-BERT embeddings [7] to group synonymous variants. Finally, UTOPIA produces a *dynamic* and *interactive* pivot table with options for expanding and collapsing data values to display synonymous variants.

In our demonstration, participants will observe how UTOPIA effectively generates the desired pivot table, bypassing data issues such as multi-valued attributes and synonymous variants, and without requiring any additional effort from the user.

Related work. Data normalization [2] is related to UTOPIA as it also aims to make data suitable for easy querying and aggregation. However, existing works in data normalization tend to focus more on syntax than semantics. For instance, they normalize multi-valued data by keeping only the first value and disregarding the rest [2]. While this addresses the structural issue of the data, it neglects its content. In the realm of automatic data analysis [1, 3, 8], there are also related works to assist users in summarizing data. However, the primary focus of these works is on automatic summarization, while UTOPIA aims to ease data preprocessing tasks for novice users for the task of pivot table generation. To the best of our knowledge, semantically aggregating synonymous variants for pivot table construction has not been studied before.

We proceed to provide an overview of our system in Section 2 and then a detailed walkthrough of our demonstration scenario based on Example 2 in Section 3.

2 SYSTEM OVERVIEW

UTOPIA addresses issues in disorganized data containing multi-valued attributes and synonymous variants. UTOPIA consists of the following key components: Multi-Valued Attribute Handler, Synonymous Variants Handler, and Data Organizer. The *Multi-Valued Attribute Handler* automatically identifies and parses multiple values within an attribute. Next, a *Synonymous Variants Handler* identifies variants and semantically aggregates such variants. Finally, a *Data Organizer* generates an organized pivot table.

Multi-Valued Attribute Handler. The Multi-Valued Attribute Handler in UTOPIA is responsible for detecting and parsing multi-valued attributes, thereby achieving data normalization. UTOPIA has the capability to identify multiple values, even if explicit delimiters are absent. For instance, if the "Genre" attribute contains a value like "superhero actioncrimeepic drama," UTOPIA recognizes the presence of multiple values. Therefore, UTOPIA parses these values, transforming them into a set: {"superhero action", "crime", "epic drama"}. To accomplish this, UTOPIA leverages a data extraction algorithm [6] that enables the extraction of multiple values even in the absence of explicit delimiters.

Synonymous Variants Handler. In this phase, UTOPIA searches for synonymous variants within the parsed data. The initial step involves checking the cardinality of unique words in the selected attributes and identifying any overlapped words. If the cardinality exceeds a certain threshold τ, and the number of overlapped words surpasses a certain fraction α of the cardinality, it is determined that variants are present. Here, τ and α are system parameters with default values 50 and 60%, respectively. However, an administrator can tune these parameters to better suit the target data domain.

Afterwards, UTOPIA aggregates variants that have similar or identical semantic meanings. To achieve such semantic aggregation, UTOPIA employs word embeddings to calculate the similarity between values in the vector space. Since traditional word embeddings like Word2Vec [4] or Glove [5] struggle with comprehending long phrases such as "space opera sci-fi", UTOPIA opts for pre-trained sentence-BERT embeddings [7] to group variants effectively. To compute similarity between values, we use cosine similarity, due to its prevalence in measuring distances between words.

For instance, in Table 2, "action" and "superhero action" share the same semantic meaning, resulting in a small distance between them. UTOPIA computes word embeddings and distances between all potential values. If the distance between two values is below a threshold δ, the handler aggregates those variants. While UTOPIA suggests a default value for δ, it is a customizable parameter and the user is free to tune it as desired. In fact, the ability to tune this parameter enables the user to obtain the best pivot table according to their preferences. Note that in traditional relational database settings, when data must be normalized *before* processing, such flexibility is unavailable.

Data Organizer. Finally, UTOPIA organizes data to display a pivot table with parsed data and semantically aggregated variants. UTOPIA represents parsed values as row or column labels in the pivot table. If the parsed values include synonymous variants, UTOPIA displays a representative value (e.g., "action" is chosen as representative for "action", "superhero action", "team action", and "epic action"). UTOPIA derives the representative value by choosing the one whose embedding is closest to the average embeddings over all variants.

3 DEMONSTRATION

We will demonstrate UTOPIA over part of the IMDb dataset. The data contains metadata for the top 1000 successful movies, with eight attributes, such as movie title, year, genre, director name, etc. We randomly introduced some misspellings and augmented this data with sub-genres. UTOPIA's user interface is depicted in Figure 2. We will guide users through eleven steps (annotated in Figure 2). For our guided scenario, we will use Patel's case study. The user will impersonate Patel: they are interested in generating a pivot table to find top-grossing movie genres for each year.

Step Ⓐ (Uploading data) The user first uploads a (disorganized) data containing multi-valued attributes and synonymous variants. In this case, the IMDb dataset for top 1000 successful movies.

Step Ⓑ (Viewing data overview) Next, the user previews the data. UTOPIA displays the first three rows. The user can horizontally or vertically scroll to see more data.

Step Ⓒ (Selecting attributes) Then the user selects the attributes they want to focus on for the pivot table generation. In our scenario, the user chooses `Year`, `Genre`, and `Gross`.

Step Ⓓ (Choosing positions) The user can assign a selected attribute to one of the pivot table positions, "Column", "Row", or "Value", by dragging it to the desired position. When an attribute moves to "Row" or "Column", then the values of that attribute will be assigned as row or column labels in the pivot table, respectively. If the user selects an attribute to "Value", it will be aggregated. In our guided scenario, the user chooses `Year` for column, `Genre` for row, and `Gross` for value.

Step Ⓔ (Enabling multi-valued attribute handler) UTOPIA displays an icon, next to each row/column attribute, representing multi-valued attribute handler. This icon is gray (disabled) if the attribute does not contain multiple values. In our guided scenario, `Year` has a disabled icon, while `Genre` has an enabled icon. The user chooses to keep it enabled and proceeds.

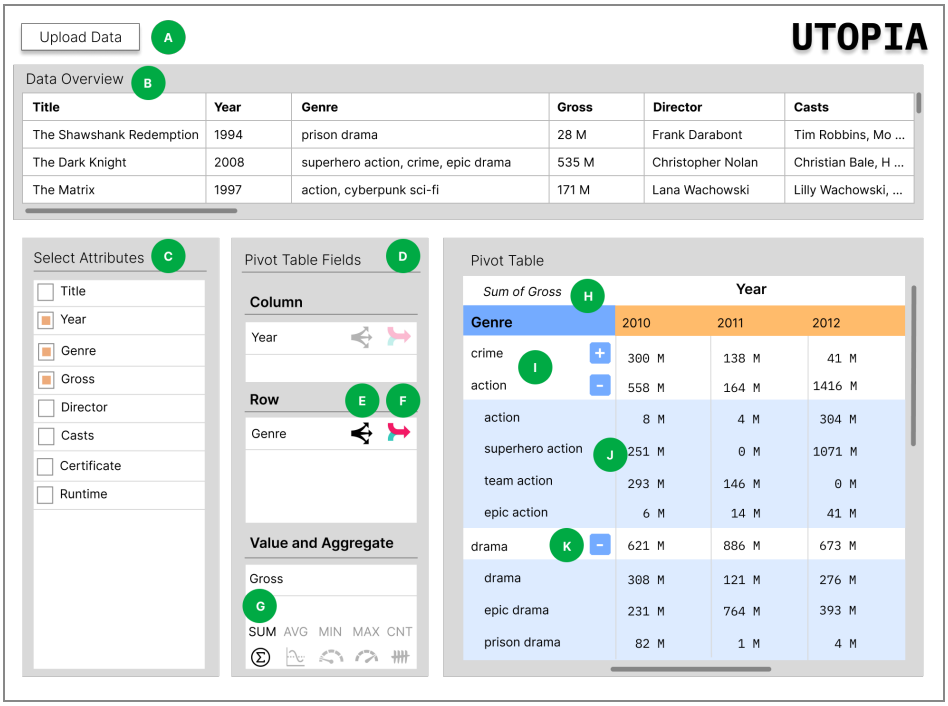

Figure 2: The UTOPIA demo: (A) upload disorganized data, (B) view data overview, (C) select attributes to analyze, (D) choose a position for the selected attribute for the pivot table, (E) select multi-valued attribute handler, (F) select synonymous variants handler, (G) choose aggregation method, (H) view aggregated attribute, (I) view the parsed data of multi-valued attributes, (J) view semantically aggregated values from synonymous variants, (K) extend or collapse semantically aggregated variants.

Step (F) (Enabling synonymous variants handler) Next to the multi-valued attribute handler icon, UTOPIA displays another icon, representing synonymous variants handler. This icon is gray (disabled) if the attribute does not contain synonymous variants. Otherwise, it is enabled. The user can choose to disable it if they wish. In our guided scenario, `Year` has a disabled icon, while `Genre` has an enabled icon. The user chooses to keep it enabled and proceeds. Moreover, by right-clicking this icon, the user can specify additional system parameters τ, α, and δ (not shown in the figure).

Step (G) (Choosing an aggregation method) The user selects `SUM` as the aggregation method over `Gross`.

Step (H) (Viewing the pivot table) UTOPIA now produces a pivot table in the bottom right panel. All values for `Gross` for the `Genre` "action" are aggregated together: showing the sum of gross as `558 M` for `2010`.

Step (I) (Viewing parsed data from multi-valued attribute) UTOPIA automatically parses the values within `Genre` and organizes the resulting values as row labels.

Step (J) (Viewing semantically aggregated variants) UTOPIA semantically aggregates the synonymous variants and shows the representative value on top. For example, under "action", there are four synonymous variants.

Step (K) (Expanding or collapsing variants) The user can expand (collapse) the row labels to show (hide) the synonymous variants. E.g., expanding "action" reveals four synonymous variants.

Demonstration engagement. After the guided demonstration, participants may use UTOPIA to explore their own datasets. Through the demonstration, we will showcase how UTOPIA can effectively display a well-summarized pivot table from disorganized data. The key takeaway is the convenience UTOPIA provides for creating pivot tables without any additional requirements from the user.

Target audience and implementation status. UTOPIA targets a broad user spectrum, ranging from novices to expert data analysts seeking to avoid time-consuming data preprocessing when generating pivot tables in a spreadsheet environment. We have implemented the synonymous variants handler, and the data organizer; the multi-valued attribute handler and UI development are under progress.

REFERENCES

[1] R J L John, D Bacon, J Chen, U Ramesh, J Li, D Das, R V Claus, A Kendall, and J M Patel. 2023. DataChat: An Intuitive and Collaborative Data Analytics Platform. In *SIGMOD*. 203–215.

[2] P Li, Y He, C Yan, Y Wang, and S Chaudhuri. 2023. Auto-Tables: Synthesizing Multi-Step Transformations to Relationalize Tables without Using Examples. *PVLDB* 16 (2023), 3391–3403.

[3] P Ma, R Ding, S Wang, SHan, and D Zhang. 2023. InsightPilot: An LLM-Empowered Automated Data Exploration System. In *EMNLP*. 346–352.

[4] T Mikolov, K Chen, G Corrado, and J Dean. 2013. Efficient Estimation of Word Representations in Vector Space. In *ICLR*.

[5] J Pennington, R Socher, and C D Manning. 2014. Glove: Global Vectors for Word Representation. In *EMNLP*. 1532–1543.

[6] M Raza and S Gulwani. 2017. Automated Data Extraction Using Predictive Program Synthesis. In *AAAI*. 882–890.

[7] N Reimers and I Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *EMNLP*. 3980–3990.

[8] A Wu, Y Wang, M Zhou, X He, H Zhang, H Qu, and D Zhang. 2022. MultiVision: Designing Analytical Dashboards with Deep Learning Based Recommendation. *IEEE Transactions on Visualization and Computer Graphics* 28, 1 (2022), 162–172.