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 novices, 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 are stored as a comma-separated value (“Action, Comedy, Drama”) and “Science Fiction” can be represented in various ways: “Sci-Fi”, “scifi”, “Technological Fiction”, etc. Such data issues pose barriers for novices while constructing pivot tables, and result in noisy and incomprehensible summarization. Parsing multi-valued attributes forces users to resort to external tools (Power Query) or methods (Python), requiring additional expertise; and consolidating synonymous variants demand manual effort, which is a tedious task.

We introduce UTOPIA, an Automatic Pivot Table Assistant that extends the functionality of spreadsheet pivot tables, overcoming data issues such as multi-valued attributes and synonymous variants. UTOPIA helps construct pivot tables without requiring additional expertise by (1) automatically detecting multi-valued attributes and organizing the values, achieving implicit data normalization; and (2) leveraging SimCSE embeddings and K-Means clustering to consolidate synonymous variants, enabling semantic aggregation.

We will demonstrate how UTOPIA enables effective pivot table construction, relieving technical novices from tedious data preprocessing while allowing them to remain in their familiar spreadsheet environment, without requiring 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 and summarization. Aggregation is one of the most effective summarization methods, which involves many operations, 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, 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 group data by certain attributes (specified as “row”) and apply aggregates to other attributes of interest (specified as “column”).

Spreadsheets such as Microsoft Excel and Google Sheets offer pivot table functionality, where users can use non-relational or “flat” data, and perform relatively straightforward data analysis. Despite being designed for novices, spreadsheet pivot tables bring adversities when the data is disorganized and messy, preventing the users from obtaining the desired results directly. Disorganized data requires careful, often manual, pre-processing, which demands additional technical expertise—such as programming in Python or Power Query—and time-consuming manual effort such as identifying all synonyms and replacing them with a canonical value. Technical novices comprise approximately 11% of office workers [1], who lack programming expertise (Python, PowerQuery, Regex, etc.).

While business intelligence models such as Tableau and Microsoft Power BI target such novices, they still require the user to learn Power Query or RegEX to perform nontrivial data preprocessing, and offer no solution to automatically consolidate synonymous variants.

Example 1. Patel, a technical novice, is working with the IMDb data about 1000 most popular movies (Table 1(a)). Patel wants to know which film genres yield significant gross. She decides to use Microsoft Excel pivot table, grouping by Genre and summing over Gross, and expects to obtain Table 1(c). To her disappointment, she gets Table 1(b). The reason is that Genre contains multiple values for some movies, such as “action, crime, drama”, and Excel incorrectly assumes that this entire comma-separated list is the value for genre. To get her desired results, Patel must parse the multi-valued attribute Genre, which requires using an advanced Excel functionality or writing a Python script.

However, Patel’s struggle continues even after parsing these comma-separated lists. She realizes that some movies have only one genre, whereas some have up to 10 different genres. She must decide how to store such variable-length values against a single attribute Genre. She can derive 10 different attributes labeled by Genre1, Genre2, . . . , Genre10, but this will result into two issues. First, many cells will be empty as not all movies have 10 genres, resulting in poor data representation. 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. If she generates one pivot table for each of the newly generated attributes, it will result in 10 different pivot tables, which is not what Patel wants. While Power Query may offer a solution, Patel hesitates to leave the spreadsheet environment, which she is very comfortable with.

Example 2. Patel somehow parsed the multi-valued attribute Genre using an external tool (Microsoft Power Query) and is now working on a different movie dataset (Table 2(a)). She proceeds to generate pivot table over this new dataset and obtains Table 2(b). Patel struggles to interpret the results as she expects “action” to be the top-gross genre, but the result indicates “biography” to have a
Table 1: (a) An sample from the IMDb dataset. 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.

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

<table>
<thead>
<tr>
<th>Title</th>
<th>Sum of Gross</th>
</tr>
</thead>
<tbody>
<tr>
<td>action, crime, drama</td>
<td>535 M</td>
</tr>
<tr>
<td>action, sci-fi</td>
<td>464 M</td>
</tr>
<tr>
<td>action, adventure</td>
<td>323 M</td>
</tr>
<tr>
<td>biography, drama</td>
<td>97 M</td>
</tr>
<tr>
<td>drama</td>
<td>65 M</td>
</tr>
</tbody>
</table>

Related work. Microsoft Excel FlashFill and Google Sheets SmartFill can split multi-valued attributes but require manual input of examples with uniform delimiters and knowledge of the resulting fragments, demanding significant user effort. Unlike UTOPIA, these tools cannot handle synonymous variants. Auto-Tables [4] normalizes multi-valued attribute by keeping only the first value and disregarding the rest, addressing only the structural issue of the data, but neglecting its content. Automatic data analysis tools [3, 5, 9] assist in data summarization, but they focus only on summarization, while UTOPIA aims to reduce the barrier of data preprocessing for technical novices, specifically for the task of pivot table generation in a spreadsheet environment.
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. 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 consists of three key components: (1) Multi-Valued Attribute Handler, which automatically identifies and parses multiple values within an attribute, (2) Synonymous Variants Handler, which identifies and semantically aggregates variants, and (3) Data Organizer, which generates an interactive pivot table.

Multi-Valued Attribute Handler. This component is responsible for detecting and parsing multi-valued attributes, thereby achieving data normalization. UTOPIA can identify multiple values, even if explicit delimiters are absent. For instance, for the value “superhero actioncrimeepic drama”, UTOPIA recognizes the presence of multiple values and parses them by transforming them into a set: {“superhero action”, “crime”, “epic drama”}. To accomplish this, UTOPIA leverages a data extraction algorithm [7] that enables the extraction of multiple values even in the absence of explicit delimiters.

Synonymous Variants Handler. Next, UTOPIA searches for synonymous variants within each relevant attribute. To this end, UTOPIA computes the ratio \( r \) between the number of unique and total values. If \( r \) exceeds a certain threshold \( \tau_h \in [0, 1] \), UTOPIA determines that synonymous variants may exist. E.g., if there are semantically related values such as “Sci-Fi”, “Science Fiction”, and “Cyberpunk”, then a large number of unique values will exist, leading to a large \( r \). However, a very high value for \( r \) indicates that observing unique values is natural for that attribute. For instance, people’s first names may have many unique values, but they are not synonymous variants. Therefore, when \( r \) does not exceed the threshold \( \tau_h \in [0, 1] \) (while \( \tau_h < 0.4 \) and \( \tau_h = 0.8 \)), UTOPIA maintains minimum values within each attribute.

To find values that share similar or identical semantic meanings, UTOPIA employs word embeddings to compute the similarity between values in the vector space. For instance, in Table 2, “action” and “superhero action” share the same semantic meaning, resulting in a small distance between them in the embedding space. Since traditional word embeddings [6] struggle to comprehend long phrases such as “space opera sci-fi”, UTOPIA opts for state-of-the-art pre-trained sentence embedding model, SimCSE [2], which obtains superior performance over Sentence-BERT [8].

Next, UTOPIA consolidates semantically similar values using K-Means clustering. It suggests the value of \( k \) based on the best silhouette score, a specialized metric to measure cluster quality that is convex-shaped. However, \( k \) is a customizable parameter and the user can tune it according to their preferences. For instance, if fine-grained grouping is desired, \( k \) should be set to a large value, and for a more generalized view of the data, \( k \) should be set to a smaller value. Note that traditional relational databases do not offer such flexibility due to strict schema requirements.

Data Organizer. Finally, UTOPIA displays a pivot table with parsed data and semantically aggregated variants. UTOPIA represents parsed values as row or column labels in the pivot table. UTOPIA displays a representative value (e.g., “action” is chosen as representative for “action”, “superhero action”, “team action”, etc.) in the presence of synonymous variants. The representative value is the one with the closest embedding to the average embeddings over all variants.

To ensure that the original data integrity is preserved without any loss of information, UTOPIA stores the parsed data in JSON format, a natural choice over tabular format for storing multi-valued attributes. This also avoids repetitive parsing computation for subsequent operations. Moreover, UTOPIA is robust to data updates; instead of recomputing clusters for minor data changes, it assigns the new data to the most similar cluster. Note that our focus is on usability enhancement and our target platform is spreadsheets, which doesn’t support big data, indicating no scalability challenge is involved.

Preliminary results: Over the IMDb dataset and with 27 expected genres, we found UTOPIA to attain a purity score—which indicates how much each cluster contains semantically similar values—of 0.86. We also tried ChatGPT 3.5 using the prompt “Group the following words into semantically related groups. Don’t change or omit words. Create \( k \) groups.” for different values of \( k \). However, ChatGPT shows an interesting (but undesirable) behavior when \( k \) is smaller than ideal. With \( k = 10 \), ChatGPT forms the following groups: {“Action and Adventure”, “Animation and Comedy”, “Crime and Documentary”,...}. While ChatGPT succeeds in including all sub-genres of “Action” and “Adventure” into “Action and Adventure”, it incorrectly merges groups based on their lexicographic similarity: “Action” is alphabetically close to “Adventure” but not semantically. Ideally, one would prefer “Action” to be merged with “Thriller” or “Crime” over “Adventure”. In contrast UTOPIA provides semantically meaningful groups even when fewer clusters are requested by the user, usually for enhanced interpretability.

3 DEMONSTRATION SCENARIO

We will demonstrate UTOPIA over datasets from various domains, such as recipe data listing multiple ingredients\(^6\), IMDb data including various sub-genres, and university course survey over various departments\(^7\). Below we provide a demonstration scenario over part of the IMDb dataset, which contains metadata for the top 1000 successful movies, with eight attributes (movie title, year, genre, etc.). We randomly introduced some misspellings and augmented this data with sub-genres. We will guide users through eleven steps (annotated \( \odot \)) to generate a pivot table.

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

Step B (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 C (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 D (Choosing positions) The user can assign a selected attribute to one of the pivot table positions, “Column”, “Row”, or

\(^6\)Recipe dataset: www.github.com/majumderb/recipe-personalization
\(^7\)www.kaggle.com/datasets/sank3t/university-student-survey
“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 E (Enabling multi-valued attribute handler) UTOPIA displays an icon, next to each row/column attribute, representing a 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.

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 $\tau_l$, $\tau_u$, and $k$ (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 of the Genre “action” are aggregated together: showing the sum of gross as 558 M for 2010.

Step J (Viewing semantic aggregation) UTOPIA semantically aggregates the synonymous variants and shows the representative value on top. E.g., “action” has 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.

After the guided demonstration, participants may use UTOPIA to explore their own datasets. The key takeaway is the convenience UTOPIA provides for creating pivot tables over disorganized data, without requiring any additional skill from the user. In summary, UTOPIA targets technical novices who seek to avoid technically challenging and time-consuming data preprocessing while generating pivot tables in a spreadsheet environment.

REFERENCES


