October 2021

Automatically Deriving Spreadsheet Cell Values Using Natural Language

Andy Lavery

Earl J. Wagner

Matthew Albright

Automatically Deriving Spreadsheet Cell Values Using Natural Language

ABSTRACT

This disclosure describes techniques to populate spreadsheet cells that depend on, but aren’t numerically calculable from, other cells. Based on a natural language query, the empty cells of a partially populated sheet that are contextually dependent on the thus-far-filled cells are automatically filled. The techniques provide greater speed, accuracy, and scalability for datasets small and large, enabling users to efficiently explore data. The techniques obviate the need to manually populate each cell in the sheet, a time-consuming and error prone procedure that does not scale well. Automatically filling in values in a spreadsheet, as described herein, opens up possibilities for the user that don’t currently exist, e.g., answering hundreds or thousands of questions based on the context in a sheet using simple, templated, natural-language queries.

KEYWORDS

- Spreadsheet
- Spreadsheet formula
- Natural language query
- Smart spreadsheet
- Smart autocomplete
- Spreadsheet auto-filling
- Natural language processing
- Question-answer system
- NLP-based spreadsheet
- Unstructured data

BACKGROUND

	A	B	C	D	E	F	G	H
1	**Book Title**	**Author**	**Published 1st Edition**	**Newest Edition Date**	**Sales Rank**	**Sales 2019**	**Sales 2020**	**Sales 2019 to 2020**
2	How to Win	John X. Author						=sum(F2:G2)
3	Folk Tales	Jane Y. Author						=sum(F3:G3)

Fig. 1: An example spreadsheet, with some columns dependent, but not numerically calculable from, other columns

Spreadsheets enable users to organize numeric and text data. Spreadsheets also provide functions and include programming capability, such that a cell may be derived from other cells. A challenge in constructing spreadsheets is populating data into the spreadsheet that depend on but cannot be calculated from other cells in the sheet.

For instance, consider the sheet of Fig. 1, which tabulates a set of books and their attributes. Although columns C-G depend on columns A (book title) and B (author), they aren't numerically calculable from columns A and B. Columns A and B (the static or independent columns, in blue) are thus entered by the user, but the user typically manually finds and enters values for columns C to G. In contrast, column H (in green) is a numerical column derived from columns F-G, and is automatically populated once values are entered in columns F-G.

Some current spreadsheet products have limited ability to create formulas or charts based on natural language processing (NLP). However, these generally cannot automatically search and populate the cells of a spreadsheet with relevant information. For example, a spreadsheet user can currently say "Calculate the average sales per year for columns F and G," but such a command is merely restating a task or mathematical formula in natural language. As another example, spreadsheets can fetch, in response to natural language queries, structured data such as stock quotes and geographic population data from external sources.

DESCRIPTION

This disclosure describes techniques to automatically populate cells in a sheet that depend on but aren't numerically calculable from other cells. Based on a natural language query formulated by the user, empty cells of a partially populated sheet are automatically filled with information dependent on the cells that have been filled. In the example of Fig. 2, natural language queries formulated by the user cause an automatic filling-in of columns C-G based on the entries of columns A and B.

	A	B	C	D	E	F	G	H
1	**Book Title**	**Author**	**When was the 1st edition published?**	**What is the newest edition?**	**What is the sales rank?**	**What are the 2019 sales?**	**What are the 2020 sales?**	**Sales 2019 to 2020**
2	How to Win	John X. Author						=sum(F2:G2)
3	Folk Tales	Jane Y. Author						=sum(F3:G3)

Fig. 2: Automated filling-in of dependent cells of a spreadsheet based on natural language queries

Per the techniques, illustrated in Fig. 2, the user annotates NLP-derived columns (purple) in a manner similar to a mathematical formula. For example, a natural language query for column C, "Published 1st Edition" can be formulated as "When was the 1st edition for book {A} written by author {B} published?" While somewhat similar to a mathematical query, e.g., "`=sum(F2:G2)`" of column H, the natural language query is more unstructured, and its answer depends on contextual information within the sheet (e.g., columns A and B in this case) as well as externally sourced information.

Formally, a natural language query for column C can be written as

```
=NLP(“when was the first edition published”, A2, B2),
```

where the first parameter is the natural language question, and the second and third parameters establish context (e.g., static, or independent columns) for the question. For example, the above natural language query formula can be parsed as

```
“When was the first edition of,” A2, “authored by,” B2, “published?”
```

which, when applied to Row 2, reads as

```
“When was the first edition of ‘How to Win’ authored by John X. Author
published?”
```

and, when applied to Row 3, reads as

```
“When was the first edition of ‘Folk Tales’ authored by Jane Y. Author
published?”
```

Similarly, a natural language query for column D can be

```
=NLP(“what is the newest edition”, A2, B2);
```

a natural language query for column E can be

```
=NLP(“what is the sales rank”, A2, B2);
```

a natural language query for column F can be

```
=NLP(“what are the 2019 sales”, A2, B2);
```

a natural language query for column G can be

```
=NLP(“what are the 2020 sales”, A2, B2);
```

etc.

For each row of an NLP-derived column, the natural language query is evaluated using a question-answer system or search engine to arrive at an answer. As explained earlier, the answer depends not only on the static columns (A and B in the above example) but also on externally sourced information. The sheet is automatically updated with the answers. If any of the dependent cells change, e.g., if the NLP query changes to "when was the 2nd edition published", or if any of the context values change (e.g., cell A2 or B2), the NLP-based answers are automatically updated.

The question-answer system can be a collection of structured data about entities (e.g., people, places, things, mathematical theorems, organizations, etc.) that is automatically derived from unstructured data. The structured data can include relationships between the entities and factual attributes and semantic representations of them. The question-answer system can also be leveraged to understand the intent of the questions or text as formulated by the user.

Some features of the described techniques include:

- ***Links to source documents*****:** The automatically filled-in answers can include links to source documents from which the automatically populated answers were derived, enabling the user to explore the source and additional details of the natural language answers. Linking to source documents also provides a way for the user to validate the data and to test its accuracy.
- ***Correcting data input by the user*****:** The techniques can discover greater detail about static (user-inputted) columns. For example, in Fig. 1, it may be the case that the book 'How to Win' actually has two co-authors, John X. Author and Bob Z. Author. If the natural language query discovers this fact, it can offer to correct or update the data input by the user.

- ***Confidence level*****:** The techniques can display a confidence level associated with an answer and can enable the user to set a confidence threshold that is to be reached to automatically populate a cell. The confidence metric can be the same one used by the underlying question-answer system. A visual indicator can be displayed, e.g., green circle for greater than 90% confidence, red X for less than 50% confidence, etc. Users can set different confidence thresholds for different data. For example, for cells with high accuracy requirements, only answers with high confidence can be automatically populated. Cells with greater flexibility or tolerance for ambiguity can be set to lower confidence levels.
- ***Use of structured and unstructured data sources*****:** The described techniques analyze unstructured text to understand the intent of the user's question or spreadsheet data request. Having determined intent, the techniques combine the column, e.g., (date of) published 1st edition, along with one or more context values (e.g., book title, author) and flexibly leverage unstructured or structured data sources (or both) to arrive at an answer.
- ***Multi-dimensional data*****:** In response to a query formulated in natural language, the techniques can handle and import multi-dimensional as well as single-dimensional data. For example, consider a spreadsheet with a list of cities and their demographics. One of the columns is “population,” and the spreadsheet-writer is looking to build graphs and perform calculations on the populations of cities. The “population” cell can either be a single value, e.g., the latest population, or it could be an array of populations over time, such as populations from ten-year censuses. Per the techniques, which populate cells programmatically in response to natural language queries, a time series of population data can be retrieved, which the user can use to graph and to calculate.

- ***Auto-discovery of the data types for the answers to questions*:** Different questions have answers with different data types, e.g., GPS coordinates, dates, single integer values, arrays of integers, etc. Per the techniques, metadata pertaining to the answer type is maintained and used for display, calculation, charting, graphing, helping the user manipulate the data more easily, etc. For example, if the data type is GPS coordinates, a chart based on that data can automatically select a geospatial visualization. As another example, a chart with access to a single integer value is visualized differently from a chart with access to a time series in each cell.

In this manner, natural language queries can be used to populate a portion of a spreadsheet. A row of a spreadsheet includes one or more cells that provide context information for the data in that row. The context is used in conjunction with natural language queries applied to other columns to generate the remainder of that row. The information that populates the NLP-derived cells is collected and analyzed dynamically, without the user needing to enter the information manually.

The described techniques provide greater speed, accuracy, and scalability for datasets small and large, enabling users to efficiently explore data. The techniques obviate the need to manually populate each cell in the sheet, a time-consuming and error-prone procedure that does not scale well. The techniques open up possibilities for the user that don’t currently exist, e.g., answering hundreds or thousands of questions based on the context in a sheet using simple, templated, natural-language queries.

CONCLUSION

This disclosure describes techniques to populate spreadsheet cells that depend on, but aren’t numerically calculable from, other cells. Based on a natural language query, the empty cells of a partially populated sheet that are contextually dependent on the thus-far-filled cells are automatically filled. The techniques provide greater speed, accuracy, and scalability for datasets small and large, enabling users to efficiently explore data. The techniques obviate the need to manually populate each cell in the sheet, a time-consuming and error prone procedure that does not scale well. Automatically filling in values in a spreadsheet, as described herein, opens up possibilities for the user that don't currently exist, e.g., answering hundreds or thousands of questions based on the context in a sheet using simple, templated, natural-language queries.

REFERENCES

[1] https://vyasa.com/solutions/synapse/ accessed Jul. 26, 2021.

[2] https://vyasa.com/videos/features-of-synapse/ accessed Jul. 26, 2021.

[3] https://vyasa.com/videos/rare-disease-use-case/ accessed Jul. 26, 2021.

[4] Vagell, Vance Julius, and Colleen O'banion. "Populating values in a spreadsheet using semantic cues." U.S. Patent Application 16/847,522 filed Apr. 13, 2020.

[5] "Google's Smart Fill autocompletes Sheets columns using AI" available online at https://venturebeat.com/2020/10/15/googles-smart-fill-autocompletes-sheets-columns-using-ai/ accessed Jul. 26, 2021.