Initial Parsing Steps

Contents:

When a dataset is initially loaded into the Transformer page, one or more steps may be automatically added to the new recipe in order to assist in parsing the data. The added steps are based on the type of data that is being loaded and the ability of the application to recognize the structure of the data.

Automatic Structure Detection

> **NOTE:** By default, these steps do not appear in the recipe panel due to automatic structure detection. If you are having issues with the initial structuring of your dataset, you may choose to re-import the dataset with Detect structure disabled. Then, you can review this section to identify how to manually structure your data. For more information on changing the import settings for a dataset, see *Import Data Page*.

This section provides information on how to apply initial parsing steps to raw imported datasets. These steps should be applied through the recipe panel.

> **NOTE:** Imported datasets whose schema has not been detected are labeled, **raw datasets**. These datasets are marked in the application. When a recipe for this dataset is first loaded into the Transformer page, the structuring steps are added as the first steps to the associated recipe, where they can be modified as needed.

Overview

When data is first loaded, it is initially contained in a single column, so the initial steps apply to `column1`.

Step 1: Split the rows. In most cases, the first step added to your recipe is a `splitrows` transform, which breaks up the individual rows based on a consistently recognized pattern at the end of each line. Often, this value is a carriage return or a carriage return-new line. These values are written in Wrangle as `\r` and `\r\n`, respectively. See the example below.

Step 2: Split the columns. Next, the application attempts to break up individual rows into columns.

- If the dataset contains no schema, the `split` transform is used. This transform attempts to find a single consistent pattern or a sequence of patterns in row data to demarcate the end of individual values (fields).
- If the dataset contains a schema, that information is used to demarcate the columns in the dataset.

When the above steps have been successfully completed, the data can be displayed in tabular format in the data grid.

Step 3: Add column headers. If the first row of data contains a recognizable set of column names, a `header` transform might be applied, which turns the first row of values into the names of the columns.

Example recipe:

```
splitrows col: column1 on: '\r'
```

```
split col: column1 on: ',' limit: 3 quote: '\"'
```

```
header
```

See:

- *Splitrows Transform*
- *Split Transform*
- *Header Transform*

After these steps are completed, the data type of each column is inferred from the data in the sample. See *Supported Data Types*.

Excel, CSV

Microsoft Excel files are internally converted to CSV files and then loaded into the Transformer page. CSV files are treated using the general parsing steps. See previous section.

For more information, see *Import Excel Data*.

JSON

If 80% of the records in an imported dataset are valid JSON objects, then the data is parsed as JSON.

> **NOTE: Trifacta® Self-Managed Enterprise Edition** requires that JSON files be submitted with one valid JSON object per line. Consistently malformed JSON objects or objects that overlap linebreaks might cause import to fail.

Step 1: Split the rows. JSON data is initially split using `splitrows` typically. See *Splitrows Transform*.

Step 2: Unnest the rows. Then, the data must be broken out from the nested JSON structures into flat rows. Using the `unnest` transform, the application attempts to render the JSON into consistently formatted rows.

> **NOTE:** After initial parsing, you might need to apply the `unnest` transform multiple times on individual columns to completely unnest the data.

Step 3: Drop source. If the data is successfully unnested, the source column is removed with a `drop` transform.

Example recipe:

```
splitrows col: column1 on: '\n' quote: '\"'
```

In the following, the values `c1` - `c3` identify the keys used to demarcate top-level nodes in the JSON source. These become individual column headers in the data grid.

```
unnest col: column1 keys: 'c1','c2','c3'
```

If the above successfully executes, the source column is dropped:

```
drop col: column1
```

See:

- *Splitrows Transform*
- *Unnest Transform*
- *Drop Transform*

Avro

Avro-based sources of data do not require any initial restructuring.

Database Tables

Properly formatted database tables with a provided schema should not require any initial parsing steps.

Known Issues

- Some characters in imported datasets, such as `NUL` (ASCII character 0) characters, may cause problems with recognizing line breaks. If initial parsing is having trouble with line breaks, you may need to fix the issue in the source data prior to import, since the `splitrows` transform must be the first step in your recipe.

Troubleshooting

Fixing parsing issues from structured source after recipe has been created

If you discover that your dataset has issues related to initial parsing of a structured source after you have started creating your recipe, you can use the following steps to attempt to rectify the problem.

Steps:

1. Open the flow containing your recipe.
2. Select the imported dataset. From the context menu, select **Remove structure...**.
3. For the imported dataset, click **Add new recipe**.
4. Make any changes to the initial parsing steps in this recipe.
5. Select the recipe you were initially modifying. From its context menu, select the new recipe as its source.

The new initial parsing steps are now inserted into recipe flow before the recipe steps in development.