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.

File Encoding

When a text file is used as an imported dataset, Trifacta® Wrangler assumes that the imported files are encoded in UTF-8, by default.

> **NOTE:** Assessing the file encoding type based on parsing an input file is not an accurate method. Instead, Trifacta Wrangler assumes that the file is encoded in the default encoding. If it is not, the Trifacta application should be prompted with the appropriate encoding type.

> **NOTE:** In some cases, imported files are not properly parsed due to issues with encryption types or encryption keys in the source datastore. For more information, please contact your datastore administrator.

As needed, you can change the encoding to use when parsing individual files. In the Import Data page, click **Edit Settings** in the right-hand panel.

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.

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

> **NOTE:** Imported datasets whose schema has not been detected are labeled, **unstructured 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 transformation, 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.

> **NOTE:** The maximum permitted length of any individual record on input is 20 MB.

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

- If the dataset contains no schema, the Split Column transformation used. This transformation attempts to find a single consistent pattern or a sequence of patterns in row data to demarcate the end of individual values (fields).

  > **NOTE:** Avoid creating datasets that are wider than 1000 columns. Performance can degrade significantly on very wide datasets.

- 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 Rename Columns with Rows transformation might be applied, which turns the first row of values into the names of the columns.

Example recipe:

1.

Transformation Name	`Split into rows`
Parameter: Column	`column1`
Parameter: Split on	`\r`
Parameter: Ignore matches between	`\"`
Parameter: Quote escape character	`\"`

2.

Transformation Name	`Split column`
Parameter: Column	`column1`
Parameter: Option	`on pattern`

Parameter: Match pattern	`','`
Parameter: Number of matches	9
Parameter: Ignore matches between	`\"`

3.

Transformation Name	`Add header`
Parameter: Row number	`1`

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

Splitting Columns

When you import a dataset, the application can automatically split your column into separate columns based on one or more delimiters.

> **NOTE:** Avoid importing datasets that are wider than 1000 columns. Particularly with previewing transformations in the data grid, very wide datasets can consume a significant amount of memory, which can cause browser crashes. Depending on your local environment, you may be able to work with these wide datasets. However, if the dataset is joined with other datasets or shared with other users, crashes can occur.

> **Tip:** If you select the delimiter in a column with a very large number of delimiters, any suggestion card limits the split to a maximum of 250 columns. You can edit the suggested transformation to increase the number of split columns as needed. Increasing the limit can impact browser performance.

Header Row

When a dataset is imported, the application may infer the names of your columns from the first row of the dataset.

> **Tip:** Avoid importing data that contains missing or empty values in the first row. These gaps can cause problems in your headers.

- In some cases, the application may be unable to create this header row. Instead, the columns are titled `column1`, `column2`, `column3` and so on.
- If the column names are split across multiple rows in your dataset, you may need to modify the column naming transformation step.

Converted data

Some formats, such as binary data or JSON, are converted to a format that is natively understood by the product before the data is available for sampling and transformation.

Excel

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.

JSON

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

Notes:

- For JSON files, it is important to import them in unstructured format.
- Trifacta® Wrangler requires that JSON files be submitted with one valid JSON object per line.
    - Multi-line JSON import is not supported.
    - Consistently malformed JSON objects or objects that overlap linebreaks might cause import to fail.

For more information, see *Working with JSON v2*.

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 transformation 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.