Our project, for our project, we specifically focused on data pertaining to category attributes which consisted of 208,944 rows and 12 columns. We focused on the “Taxonomy Category” and “Attribute Name” columns with our target column being the “Values” column for text mining and transforming.

Data Preprocessing
First, we began with some initial text cleaning for all values. This included filtering missing values, trimming whitespace, and separating by commas and “and” into separate row values. For example, with a value such as “Accessory, Block”, this had to be separated into two separate values of “Accessory” and “Block”.

Qualitative Values
Using the edit distance (Levenshtein distance) with a threshold of distance 1, we were able to identify and fix many of the common typographical errors in the data, such as plural vs. singular words, lowercase and uppercase letters, extra punctuations and other misspellings. After that, we developed string similarity algorithms that combined Levenshtein and Jaro-Winkler distances in order to identify possible groupings among the values, which we propose as a way to find possible filtering categories. In order to be able to propose suggestions beyond the simple Levenshtein distance, we note that we will have to make more generalizations among the strings. Our proposed system thus has the flexibility of managing the trade-off between making very few changes so as to preserve all the information and making more assumptions and generalizations to identify possible groupings, based on a parameter which defines how strict the string similarity must be to output a match.

Quantitative Values
With numerical attributes, because we were dealing with different numerical values and units, we had to use a different approach. We did some additional cleaning by removing any miscellaneous characters such as semicolons and ampersands. We then separated the numerical values from the units and any additional text and dealt with these two entities separately. We standardized numerical value to be the appropriate integer or float value and to be consistent with a positive sign in front of the value. For units, we used a string similarity algorithm which was based on comparing the set of tokens and assigning a new value based on majority rule. This method was used to standardize the units whether this was with casing, spacing, or abbreviations. Lastly, given the new units and additional text, we combined the two parts together to suggest a new final value that would be outputted.