Approximate String Matching for Merging Market Data

**Introduction**

In the summer of 2016, I worked as an intern at Logitech in the Market Analytics team.

**Background**

Market Analytics is a function in business that evaluate business performance by studying and analyzing the market to obtain insights that help inform decisions and optimize the profitability of the business.

Due to the increasing competitiveness in business and speed in production and manufacturing, there is a growth in need for Market Analytics functions, especially in the consumer products industry.

For any business it has become necessary to understand the market size and demand of their products/services, trends in the market and consumer behaviors, and the proportion of the market (market share) controlled by the business and by competitors in the same market.

With the increase in availability and quality of data that can be obtained due to the rise of the digital world and the web, businesses are able to even further leverage the value of Market Analytics to the performance of their business.

From a wide variety of sources, businesses are now able to obtain a large amount of market data. With that, the potential to dig deeper insights about market trends and consumer behaviors have grown drastically.

Logitech is a company that has a wide variety of product categories in their portfolio. Currently Logitech (and its subsidiaries) has products in Computer Accessories, Tablet Accessories, Webcams, Mobile Speakers, PC Speakers, Headphones and Earphones, and more.

**Definitions**

Product Line (PL): a group of products comprising of different sizes, colors, or types, produced or sold under one unique product or model name (i.e. Apple iPhone 5, UE Boom 2)

Stocking Keeping Unit (SKU): a particular product identified by its product line as well as by its size, color, or type, where its identification is typically used for inventory purposes (i.e. Apple iPhone 5 64GB Black, UE Boom 2 Blue)

Data Mapping

Edit Distance

**Problem**

With the increasing size and availability of data, a major issue that has surged is the need of maintaining data quality.

Due to the variety of product categories and the global presence of Logitech, we obtain data from a number of different sources. As a result, we face the issue of data reported slightly differently. A major issue that we encountered were the levels of granularity that the data is reported in (SKU vs PL), and inconsistencies with names of the Stocking Keeping Units and Product Lines.

Because of this, we are not able to join data at neither the Product Line nor the Stock Keeping Unit, and therefore we are not able to get an accurate measure of sales for each SKU or PL. This hinders the ability to perform market analysis and uncover meaningful insights that could be used to inform business decisions, such as increasing or decreasing production of certain SKUs or PLs.

In this paper, I will discuss two scenarios that I encountered, and the solutions I came up with to resolve this issue.

1: Identifying SKU that belong to the same Product Line

In one of our major data source, the sales were reported at the SKU level, but no data was available for the Product Line to which these SKUs belong to. A similar example of how the data appeared looks like this:

Example table

PL SKU

? Product A Bluetooth Speaker Color Black

? Product A Bluetooth Speaker Color Green

? Product A Bluetooth Speaker Color Gray

? Product B Tablet Case for iPad Black

? Product B Tablet Case for iPad Gray

(what we need: )

Problem here: we are not able to product Product Line Rankings and only compare sales at the SKU level

2: Multiple instances of the same Product Line appearing with different names

Between two data sources, the names of the Product Line was inconsistent. This was a problem for us because we could not map the data by Product Line, and therefore we could not aggregate the sales data of the Product Line.

PL Sales

Product A Bluetooth Speaker 25

A Product Speaker 30

Product B Tablet Case for iPad 40

(what we need: )

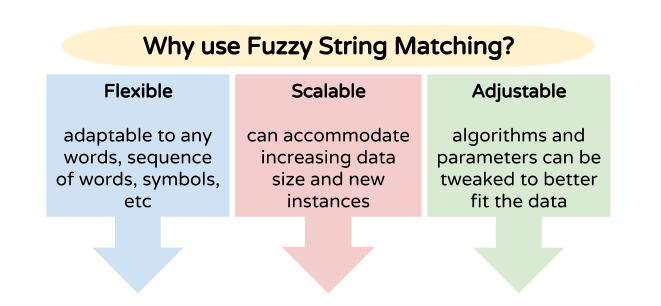
Problem here: not accurate ranking of product line (by this Product B is most selling, where actually Product A has most sales)

**Method**

Since we are dealing with a large data set (~250,000 rows), we sought for a fast and scalable approach.

String patterns seem to be consistent

(discuss all methods?)



(r bloggers)

Many algorithms

Many parameters

Able to alter and adjust algorithms to fit scenarios

Now we will discuss two scenarios I encountered, and the algorithms I used to match

**Scenario 1: Grouping Stocking Keeping Units (SKU) to Product Lines**

(from table above)

Assumptions about data

**Optimal String Alignment**

Also known as Damerau Levenshtein Algorithm (wikipedia)

Edit Distance (4 types)

Pros

Most of the string is the same for same product lines

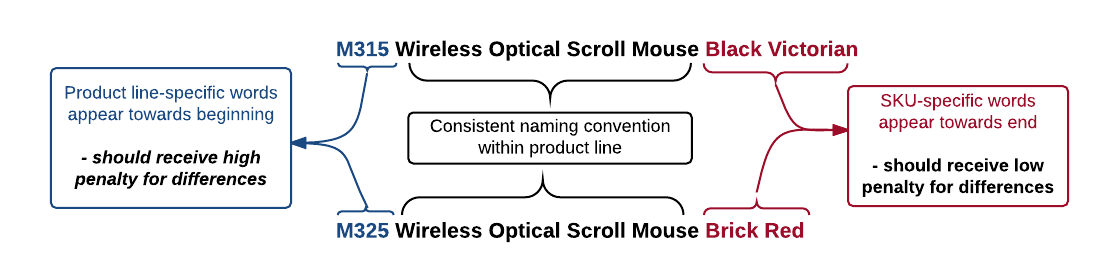
BUT Why it wouldn't work in our scenario

BOSE SoundLink Mini Bluetooth speaker

https://www.amazon.com/BOSE-SoundLink-Mini-Bluetooth-speaker/dp/B00D89H1NO/ref=sr\_1\_2?s=aht&ie=UTF8&qid=1471317044&sr=1-2&keywords=Bose+SoundLink+Mini+Bluetooth+Speaker

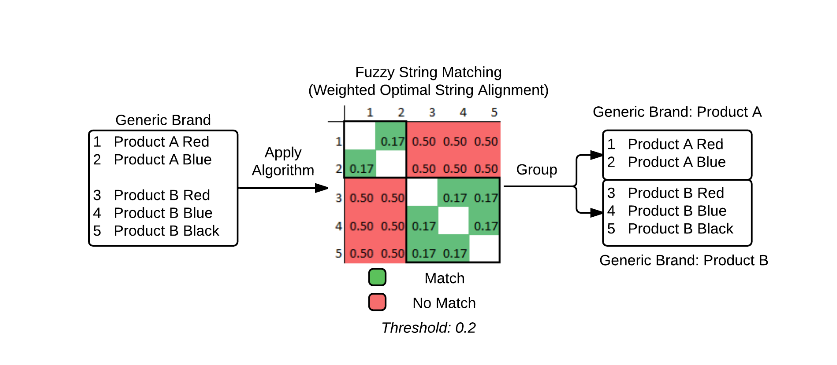
**Weighted Optimal String Alignment**

Similar example from actual data source (not exactly, to maintain anonymity of data source)



Algorithm at a glance

Score, against threshold



Works well.

This algorithm can have many applications

Parameters

| Parameter | Description | Default |

|-----------|---------------------------------------------|-----------|

| Type | Option to apply algorithm by whole “item” or by single “character”. Since we want to apply algorithm to WHOLE WORDS rather than single characters, by “item” is ideal. | character |

| Weight | Weightfunction (linear, quadratic, root). In the previous example, linear was used | linear |

| Sum.Right | Flagfor whether or not to consider everything after the first mismatch (from the left-hand side) as mismatches (in other words, sum scores of all mismatches to the right of first mismatch) | F |

Performance of parameters

**Example: Amazon Products**

https://www.amazon.com/dp/B00N32I2Q6/ref=twister\_B01AS57B0I?\_encoding=UTF8&psc=1

Now let us look at an example from amazon

```{r}

wosa("Bose SoundLink Color Bluetooth Speaker (Blue)", "Bose SoundLink Color Bluetooth Speaker (Black)", type = "item", weight = "linear", sum.right = T)

``

**Scenario 2: Matching Product Lines**

assumptions

**n-gram/q-gram**

Definition: All subsequences of N/Q consecutive items from a given sequence or string (N/Q is an integer)

Item can be identified as whole words, syllables, single characters, etc (example to the right is a 5-gram by single characters) (chart?)

N-gram is a frequency table of all subsequent occurrences

Q-gram is a distance measure based on common n-gram between two strings

Ideal for catching typos

**Jaccard Distance**

distance measure using q-gram

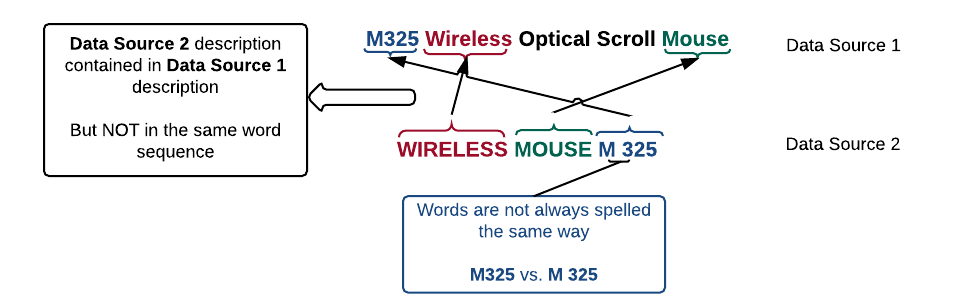
tables from slides

C:\Users\rjsai\Dropbox\UMN Courses\Plan B\jd_formula.png

Why it doesn’t work

JD score depends on number of all possible n gram

**Adjusted Jaccard Distance**



matching diagrams

parameter

**C:\Users\rjsai\Dropbox\UMN Courses\Plan B\ajd_formula.png**

**Example: Matching Product Line data from different sources**

**Discussion**

good if assumptions are met

need to be careful if assumptions are met

**Conclusion**

location weighting can be applied to other algorithms

**References**