Fuzzy merging of company names

About the presenter: *Richard Vogg studied mathematics at the Technical University of Kaiserslautern in Germany. He later obtained a master's degree with a focus in Statistics and Machine Learning and learned to use R as a tool in academic applications. In 2018, he started his first job as a Business Analyst at Evalueserve in Viña del Mar. He currently works in the company's Customer Analytics area for one of the top 5 US banks by managed assets.*

Datasets are increasingly large, and this brings along many challenges. Manually input data is especially problematic in that humans are prone to errors and do not always write or type in the same way. Take the answer to the question "Where do you work?" as an example. In our data, bankers wrote down the intended name "McDonald's" in formats as varied as: "McDonalds", "MCDONALDS CORPORATION", "McDonald's Corp", "mcdonlads" etc.

How can we match misspelled fields with their intended counterpart? How can we make sure that the right company name is being considered? This paper will shed some light onto how this problem was approached in a real business case delivered to a top 5 US-Bank by managed assets, one of Evalueserve's clients. Using the stringdist package in R, which implements several functions that measure the similarity of strings, we were able to accurately estimate what the intended input of the bankers had been.

We faced several hundreds of thousands of manually input company names and were interested in knowing whether we could identify clients in management positions at one of the S&P500 Companies in the US. We downloaded the S&P500 companies list as a reference of the correct company names. After cleaning the data, we decided to take the Jaro-Winkler distance between the strings. Unlike other distance measures, the Jaro-Winkler measure yields a higher score if one string is included in the other string (like "McKinsey" is included in "McKinsey & Company). Moreover, it can place more weight on the comparison of the first four letters of the strings.

The Jaro-Winkler distance is defined in two steps. First, the Jaro similarity sim_j between two strings s_1 and s_2 is defined as

$$sim_j(s_1,s_2) = \begin{cases} 0, & if\ m = 0 \\ \frac{1}{3}\left(\frac{m}{|s_1|} + \frac{m}{|s_2|} + \frac{m-t}{m}\right), & else, \end{cases}$$

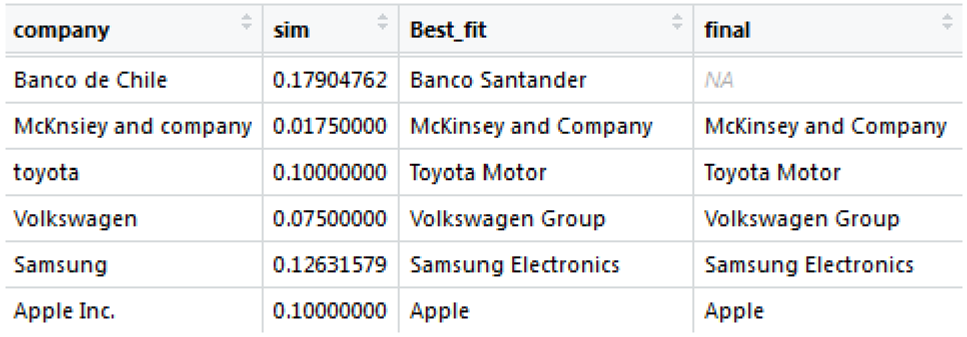
where m is the number of matching characters, $|s_i|$ is the length of string s_i and t is the number of transpositions of letters. The Jaro-Winkler similarity then adds weight to the first four letters of both strings:

$$sim_w(s_1,s_2) = sim_j + lp(1 - sim_j),$$

where l is the number of matching letters within the first four letters of both strings and $p \in [0,0.25]$ is a weight factor towards the first four letters. We define the Jaro-Winkler distance as $d_j = 1 - sim_j$.

We used the stringdist package to calculate the distance matrix of Jaro-Winkler distances between the hand-typed company names and the list of large US companies. We further determined column-wise minima, and – with these minima – the best fit from the large companies list. By simple thresholding, we selected a decision criterion for selecting the fits we wanted to keep.

The following screenshot is a toy example output with simulated data. “Company” shows the hand-typed company name, “Sim” gives us the minimum Jaro-Winkler distance found within the list of “unifying names”, Best_fit is the related company name, and final is the output. If the sim value was lower than the chosen threshold (in this example, 0.15), then we would continue the analysis with the new “unified” company name.



company	sim	Best_fit	final
Banco de Chile	0.17904762	Banco Santander	NA
McKnsiey and company	0.01750000	McKinsey and Company	McKinsey and Company
toyota	0.10000000	Toyota Motor	Toyota Motor
Volkswagen	0.07500000	Volkswagen Group	Volkswagen Group
Samsung	0.12631579	Samsung Electronics	Samsung Electronics
Apple Inc.	0.10000000	Apple	Apple

The stringdist package is implemented in a way that it automatically parallelizes if possible. Given that the calculation of the distance matrices is by far the computationally most expensive step, having the option to parallelize makes a big difference. We also tried to calculate the distances and best fits with the parallel package row by row which led to a minor improvement in runtime. As a reference: The final method took around two minutes to merge 500,000 company names to their potential counterparts in the S&P500 list – on a machine with the following hardware: Intel i5 (4 Cores) ~2.5 GHz - 8GB RAM.

We faced several challenges throughout the development of this project. Firstly, due to both the quantity of company names and the time limit, we could not implement a supervised algorithm. Secondly, we had to determine the business impact of false positives and false negatives and select the threshold accordingly. Lastly, we had to find ways to run the algorithm repeatedly on smaller chunks of data, as the stringdist package has size limits for the matrix.

In conclusion, the fuzzy merging algorithm helped us to identify 25k potential customers that have a management role in one of the S&P 500 companies. As a result, the bank’s relationship with these customers can be optimized. R helped not only to solve this business problem in an effective manner, but also to conduct posterior analyses related to clustering customers.

As a side product, this analysis inspired new applications and ideas across teams within the bank. Fuzzy merging ultimately provides us with the opportunity to glean more valuable insights from applications for which hand-typed or not standardized data is the input.