Supplementary Information for: Linking Datasets on Organizations Using Half a Billion Open-Collaborated Records *

Brian Libgober † Connor T. Jerzak‡

September 4, 2024

Abstract

Scholars studying organizations often work with multiple datasets lacking shared identifiers or covariates. In such situations, researchers usually use approximate string ("fuzzy") matching methods to combine datasets. String matching, although useful, faces fundamental challenges. Even where two strings appear similar to humans, fuzzy matching often struggles because it fails to adapt to the informativeness of the character combinations. In response, a number of machine learning methods have been developed to refine string matching. Yet, the effectiveness of these methods is limited by the size and diversity of training data. This paper introduces data from a prominent employment networking site (LinkedIn) as a massive training corpus to address these limitations. By leveraging information from the LinkedIn corpus regarding organizational name-to-name links, we incorporate trillions of name pair examples into various methods to enhance existing matching benchmarks and performance by explicitly maximizing match probabilities. We also show how relationships between organization names can be modeled using a network representation of the LinkedIn data. In illustrative merging tasks involving lobbying firms, we document improvements when using the LinkedIn corpus in matching calibration and make all data and methods open source.

Keywords: Record linkage; Interest groups; Text as data; Unstructured data

*We thank Beniamino Green, Kosuke Imai, Gary King, Xiang Zhou, members of the Imai Research Workshop, and two anonymous reviewers for valuable feedback. We would also like to thank Neil Arora, Danny Guo, Gil Tamir, and Xiaolong Yang for excellent research assistance. We also thank Daniel Carpenter for making this project possible.

†Assistant Professor of Political Science and Law, Department of Political Science, Northwestern University. Email: brian.libgober@northwestern.edu; URL: BrianLibgober.com; ORCID: 0000-0001-9638-0228.

‡Assistant Professor, Department of Government, The University of Texas at Austin. Email: connor.jerzak@austin.utexas.edu; URL: ConnorJerzak.com; ORCID: 0000-0003-1914-8905.

Contents

Appendix I: Machine Learning Model Details

We here provide details justifying the probabilistic outcome used in the machine learning approach.

A.I.1.1 Deriving Linkage Outcome Data via LinkedIn

We first assume that the alias-to-URL pairings are relevant for understanding alias-to-organization ties. That is, we assume validity:

$$\textit{Validity Assumption: } \Pr(O_i = O_j \mid A_i = a, A_j = a') = \Pr(U_i = U_j \mid A_i = a, A_j = a'), \quad (1)$$

where O_i, O_j refer to the organizations associated with index i and j, A_i, A_j refer to the aliases, and U_i, U_j refer to the URLs. Under this assumption, the overall behavior of users linking their aliases to organizational URLs is the same as if they were linking to actual organizations. Intuitively, the assumption means that the LinkedIn name-to-URL patterns give us information about the name-to-organization mapping.

Because, in linkage tasks, the organizational unit of analysis is defined by the researcher, this assumption may be satisfied in some circumstances but not in others. For example, General Motors has divisions called Buick and Cadillac, which were initially separate companies. A researcher might want these entities to match each other in some datasets. Other researchers may not want these to match, however, and for plenty of purposes, it may not make sense to make these entities connect. Therefore, whether alias-to-URL linkage is valid for a particular alias-to-organization match will depend on the way in which organizations are defined in the overall analysis.

Next, in order to use the LinkedIn corpus for learning about how different organizational names are related, we assume semantic mapping, which states that the organizational match probability is a function of the semantic content of the aliases. That is,

$$\textit{Semantic Mapping Assumption: } \Pr(O_i = O_j \mid A_i, A_j) = f_p(f_n(A_i), f_n(A_j)), \quad (2)$$

where f_n represents a function that numericizes each alias string, and f_p represents a function that transforms the alias numeric representations into overall match probabilities. Semantic mapping, which depends on the existence of a match function, is not insignificant given the many possibilities for the researcher-defined organizational unit of analysis. It also could be questionable for cross-language link tasks. Nevertheless, assuming that this function exists, we can attempt to approximate it using modern machine-learning algorithms in combination with the trillions of LinkedIn organizational alias pairs.

In clarifying the limitations of the LinkedIn data for improving linkage, we note that both the validity and semantic mapping assumptions would be vulnerable to errors in the LinkedIn data. Regardless of how researchers define their organizational unit of analysis, inaccuracies due to the selection of incorrect URLs by users would limit our ability to capture the link probability using the alias and URL information. It could introduce systematic bias as well. However, as a professional networking site, users have an incentive to maintain accurate information about themselves on the page, hopefully limiting the degree of systematic bias.

Finally, to generate ground truth data for the prediction model training, we assume independence of i and j, which allows us to calculate $\Pr(O_i = O_j \mid A_i, A_j)$. This independence assumption would be violated if, for example, users could coordinate so that one use of an alias linking to a given URL gave additional information about another use. Given the wide geographic reach of the LinkedIn network, this sort of coordination is presumably rare, yet is difficult to evaluate for all pairs of

indices. In any case, the assumption is important because, using it, we can calculate the overall organizational link probability using the LinkedIn data:

$$\Pr(O_i = O_j \mid A_i = a, A_j = a') = \Pr(U_i = U_j \mid A_i = a, A_j = a') \text{ (by } \textit{Validity}) \tag{3}$$

$$= \sum_{u \in \mathcal{U}} \Pr(U_i = u, U_j = u \mid A_i = a, A_j = a') \tag{4}$$

$$= \sum_{u \in \mathcal{U}} \Pr(U_i = u \mid A_i = a) \Pr(U_j = u \mid A_j = a') \text{ (by } \textit{Independence}) \tag{5}$$

The two terms in the final expression can be calculated using the LinkedIn network with empirical frequencies (e.g., the fraction of times alias a and a' link to u).

A.I.1.1.1 Modeling Approaches

There are many estimation models consistent with the core assumptions we describe above. Indeed, as machine-learning methods continue to evolve, the future will no doubt yield improvements in any machine-learning approach to modeling these match probabilities, as the rapid progress in large language models has made apparent (Jiang et al., 2023; Wei et al., 2022).

We here illustrate use of the LinkedIn data using a modeling strategy that explicitly optimizes match probabilities. For computational reasons which will soon become clear, we assume the following functional form for the match probabilities:

$$\textit{Distance-to-Probability Mapping: } \log\left(\frac{\Pr(O_i = O_j \mid A_i = a, A_j = a')}{1 - \Pr(O_i = O_j \mid A_i = a, A_j = a')}\right) \propto -||f_n(a) - f_n(a')||_2. \tag{6}$$

Here, the match probability is a function of the Euclidean distance between the numerical representations for aliases a and a'. This captures the intuition that intuitively close names like "apple" and "apple inc." are more likely to be matched than distant names like "apple" and "jp morgan".

Besides this intuition, this distance-to-probability mapping also has important computational benefits: if we seek to calculate the match probability between two datasets of 100 observations each, we do not have to apply a computationally expensive non-linear f_p function to all 5,050 possible pairs. Instead, we only need to apply the numericization function 200 times—once for each alias in each dataset—after which we take the Euclidean distance between each pair of numerical representations in a step that can be readily parallelized and optimized for speed (Assis Zampirolli and Filipe, 2017; Hansen, 2007).

A.I.1.2 Model Implementation Details

The character vectors and neural weights are model parameters, which are jointly updated using stochastic gradient descent to minimize prediction error on our training data computed using the LinkedIn corpus. Specifically, we minimize the KL divergence between the true match probability, $\Pr(O_i = O_j | A_i = a, A_j = a')$, and the estimated probability, where the KL divergence between two distributions, P and Q, denoted by $D_{\text{KL}}(P||Q)$, is a measure of how much information Q contains that P does not and is 0 when the two distributions contain the same information. That is, we minimize:

$$\text{minimize} \sum_{a,a' \in \mathcal{A}} D_{\text{KL}}\left(\widehat{\Pr}(O_i = O_j | A_i = a, A_j = a') \;||\; \Pr(O_i = O_j | A_i = a, A_j = a')\right), \tag{7}$$

where $\mathcal{A}$ denotes the set of all aliases. In this way, we learn from data how the aliases should be numerically represented—learning along the way the representation of characters, words, and how these things combine together in a name—with the overall goal of predicting the organizational link probability between each alias pair. We apply the approach here jointly to the entire LinkedIn corpus so that name representations in many languages are learned together.

We now discuss in more detail the machine learning model used.

A.I.1.3 Step 1. Obtaining Organizational Name Representations

We first outline the machine learning modeling strategy for each organizational name.

Input: An organizational name (e.g., "j.p. morgan chase")

1. Break each name into words.
   - For each word of length l_w, query the vector representation for each character to form a $l_w \times D$-dimensional representation for that word.
   - Apply a Transformer model to each word to obtain a representation for each word.
2. For each alias of l_a words, query the word vectors obtained from the previous step to form a $l_a \times D$-dimensional representation for the alias.
   - Apply a Transformer model to each alias to obtain a representation for each sequence of words.
3. Select a [CLS]-token representing the alias-level representation. Normalize and project to the final output dimensionality.

Output: D-dimensional representation of the organizational name

A.I.1.4 Step 2. Transforming Name Representations into Match Probabilities

With the name representations in hand, we now discuss in greater detail how we form the final match probabilities.

Input: Numerical representations/embeddings for two organizational names

1. Take the normalized Euclidean distance between the two embeddings, e_i, e_j:

$$d_{ij} = \frac{||e_i - e_j||_2}{\sqrt{\dim(e_i)}}.$$

2. Form the match probability from the distance using unidimensional logistic regression:

$$\Pr(O_i = O_j | A_i = a, A_j = a') = \frac{1}{1 + \exp(-[\beta_0 + \beta_1 d_{ij}])}.$$

Output: A match probability indicating the probability that two names refer to the same organization

The normalization layers are included to avoid exploding gradients (e.g., large values are centered and scaled) and to make the learning more robust to initialization (Santurkar et al., 2018). The parameters used in steps 1 and 2 are updated jointly using a modified form of stochastic gradient descent. (Future progress can likely be made here, given the development of large language models involving billions of parameters.) We next give more details about the sampling procedure involved in the model training.

A.I.1.5 Sampling Design for Model Training

Recall that we seek to optimize the following:

$$\text{minimize} \sum_{a,a' \in \mathcal{A}} D_{\text{KL}} \left(\widehat{\Pr}(O_i = O_j | A_i = a, A_j = a') \;||\; \Pr(O_i = O_j | A_i = a, A_j = a') \right).$$

One challenge in performing this optimization in practice is that most a' are very distinct semantically from a, so the learning process is slowed by the vast number of "easy" predictions (e.g., where $\Pr(O_i = O_j | A_i, A_j)$ is clearly 0). To make matters worse, most match probabilities are 0, a kind of imbalance that can impair final model performance (Kuhn and Johnson, 2013)

We adopt two approaches to address these challenges. First, we implement a balanced sampling scheme: in the optimization, we ensure half of all training points have $\Pr(O_i = O_j | A_i = a, A_j = a') = 0$ and half have $\Pr(O_i = O_j | A_i = a, A_j = a') > 0$.

Second, we select a, a' pairs that will be maximally informative so that the model is able to more quickly learn from semantically similar non-matches (e.g., "The University of Chicago" from "The University of Colorado") or semantically distinct matches (e.g., "Massachusetts Institute of Technology" and "MIT"). In our adversarial sampling approach (which is somewhat similar to (Kim et al., 2020)), we select, for non-matches in our training batch, alias pairs that have $\Pr(O_i = O_j | A_i = a, A_j = a') = 0$ but have the largest $\widehat{\Pr}(O_i = O_j | A_i = a, A_j = a')$. Likewise, for matches, we select pairs that have the lowest predicted probability. More formally, we find the negative a and a' pairs by solving:

$$\text{argmax}_{a,a' \text{ s.t. } \Pr(O_i = O_j | A_i = a, A_j = a') = 0} \; D_{\text{KL}} \left(\Pr(O_i = O_j | A_i = a, A_j = a') \;||\; \widehat{\Pr}(O_i = O_j | A_i = a, A_j = a') \right),$$

with a similar approach applied to positive pairs.

We achieve this in practice by, at the current state of the model, finding the closest alias to a' in the alias vector space for which $\Pr(O_i = O_j | A_i = a, A_j = a')$ is 0. A similar approach is used to obtain high-information positive matches. Intuitively, this process magnifies the importance of similar aliases that refer to different organizations in the learning process so as to learn how to model the semantic mapping of aliases into match probabilities.

We found this adaptive sampling scheme to be important in learning to quickly distinguish similar aliases that refer to different organizations, although only use this adversarial approach to obtain half of each training batch to mitigate against potential problems with this approach (such as occurs if the model is fit on only high surprise mistake alias pairs that occur when users incorrectly link to the same URL).

Appendix II: Additional Community Detection Details

The first approach towards organizational name community detection uses Markov clustering (Van Dongen, 2008). The Markov clustering algorithm operates on weighted adjacency matrices that have a probabilistic interpretation.

We denote the organization names in the source data as aliases a. These aliases constitute nodes on our graph. Many aliases appear simultaneously with profile URLs in our source data. One possible approach to defining links between aliases is to assume that two aliases are connected whenever they both link to the same profile URL. However, this approach would place as strong a connection on aliases that rarely link to the same profile URL as those that are frequently connected. For example, the URL `linkedin.com/company/university-of-michigan` has been associated with an alias 75,462 distinct times in our data. The vast majority of these connections occur via the alias "University of Michigan", but a link with the alias "Michigan State University" does in fact occur a handful of times. In defining links on the network, it will help to upweight the former but downweight the latter in a data-driven way.

While there are many possible ways to ensure more frequently made connections are weighted higher than connections that are rarer, we adopt an approach inspired by naive Bayesian classification methods. We define the edge weight, $w_{aa'}$, between node a and a' using Equation 8:

$$w_{aa'} = \sum_{u \in \mathcal{U}} \frac{\#\text{ of times } a \text{ co-occurs with } u}{\#\text{ of times } u \text{ co-occurs with any alias}} \times \frac{\#\text{ of times } u \text{ co-occurs with alias } a'}{\#\text{ of times } a' \text{ occurs with any profile URL}} \tag{8}$$

These edge weights then make up the full adjacency matrix, $\mathbf{W}$, which captures the interrelationship between all pairs of aliases.

The connection of this expression to naive Bayesian classifiers requires some elaboration. As a pure formalism, one can write the law of total probability as

$$\Pr(A_i = a \mid A_j = a') = \sum_{u \in \mathcal{U}} \Pr\left(A_i = a \mid U = u, A_j = a'\right) \Pr\left(U = u | A_j = a'\right) \tag{9}$$

Taking this equation as given, suppose further one were to "naively" assume conditional independence of A_i and A_j given U (Rish et al., 2001), or more formally $\Pr\left(A_i = a \mid U = u, A_j = a'\right) = \Pr\left(A_i = a \mid U = u\right)$, which is similar to the independence assumption of Section A.I.1.1.1.[1] Equation 8 is then what would result if one also replaced the true probabilities with sample proportions.

There are several benefits to using this formula, especially over the more obvious "shared link" approach. Besides using much more of the data, the naive Bayesian calculation reweights edges in ways that are proportionate to the actual prevalence of relationships found in the data. One corollary is that the edge weights are asymmetric, and the specificity with which an alias and link are shared matters a great deal. For example, ICPSR is a rare alias, but when it does occur, it is very often entered with `linkedin.com/company/university-of-michigan`, which in turn usually points to the "University of Michigan" alias. Therefore, the weight $w_{\text{"University of Michigan","ICPSR"}}$ will be close to 1 even if $w_{\text{"ICPSR","University of Michigan"}}$ is very small. By contrast, since "Michigan State University" and "University of Michigan" are both relatively common aliases, one spurious link

[1] By asserting the equation as a formalism, we do not dwell on questions about what exactly the probability of A_i given A_j means in this context, which might distract us from the core task of showing how community detection methods on networks can be useful for record linkage problems.

between them will not result in large weight in either direction.

Having built an adjacency matrix, $\mathbf{W}$, the probabilistic network of alias-to-alias links, we next apply the Markov clustering algorithm to it to find communities of aliases that tend to link to the same URLs. Because our focus is on the LinkedIn data contribution, we leave details to Section A.II.1.1.

Figure A.II.1 illustrates the clustering process on a subset of our data. Darker shades reflect heavier weights at initialization. Some links are much stronger than others. In the initial weighting, two cliques reflect a set of names associated with "JP Morgan Chase." Another reflects names associated with "Bank of America." However, these initial links are dense, making it difficult to distinguish one cluster of aliases from another. As the algorithm iterates, some links weaken and disappear while others strengthen. Eventually, each node links to exactly one other node. Notably, the final cliques contain lexicographically dissimilar nodes that do indeed belong in the same cluster. For example, the "Chase" clique contains "Wamu", "Paymenttech", and "摩根大通" which are all Chase affiliates. The "Bank of America" clique includes "Countrywide Financial" and "MBNA," both under the Bank of America umbrella. In this way, the semantic mapping assumption from Section A.I.1.1.1 has been weakened; graph-theoretic information has been exploited to assist organizational matching.

A.II.1.1 Markov Clustering Algorithmic Details

Briefly, this clustering algorithm proceeds in two steps—expansion and inflation—that are repeated until convergence. In the expansion step, the network's adjacency matrix is multiplied by itself k times. This matrix multiplication simulates the diffusion of a Markov process on the nodes (i.e. "traveling" k steps on the network, with probabilities of where to go defined by $\mathbf{W}$). In the inflation step, entries of the resulting matrix are raised to some power p, and the matrix is renormalized into a valid Markov transition matrix. Since small probabilities shrink faster under exponentiation than large probabilities, the inflation step causes higher probability states to stand out (i.e. likely places are made even more likely). After alternating expansion and inflation, the output converges: row i will have only one non-zero valued entry in column j, which defines the representative node in the community. All rows that have a one in column j are a part of the same community of alias linking to (hopefully) the same organizational entity.

This clustering process involves hyperparameters, in particular, the number of matrix multiplications to do in the expansion step (k) and the power of exponentiation (p) in the inflation step. For both steps, k and p are frequently set to 2. The total number of clusters is not explicitly specified but instead controlled indirectly through p and k. We adopt the common choice of $p = k = 2$.[2]

A.II.1.2 Greedy Bipartite Clustering Details

Following Clauset et al. (2004), our modularity score for bipartite community detection via greedy clustering is

$$\text{Modularity Score} = \frac{1}{2m} \sum_{a,u} \left[B_{au} - \frac{k_a k_u}{2m} \right] \mathbb{I}\{c_a = c_u\}, \tag{10}$$

where m denotes the total number of edges in the graph, B_{au} denotes the au-th entry in the weighted bipartite network (i.e., the number of ties between alias a and URL u), and k_a denotes the degree of node a (i.e. $k_a = \sum_{a \in \mathcal{A}} B_{au}$). The indicator variable, $\mathbb{I}\{\cdot\}$, is 1 when the community for a equals the community for u (these communities are denoted as c_a and c_u).

[2]Alternative choices yielded substantially similar results in the applications that follow.

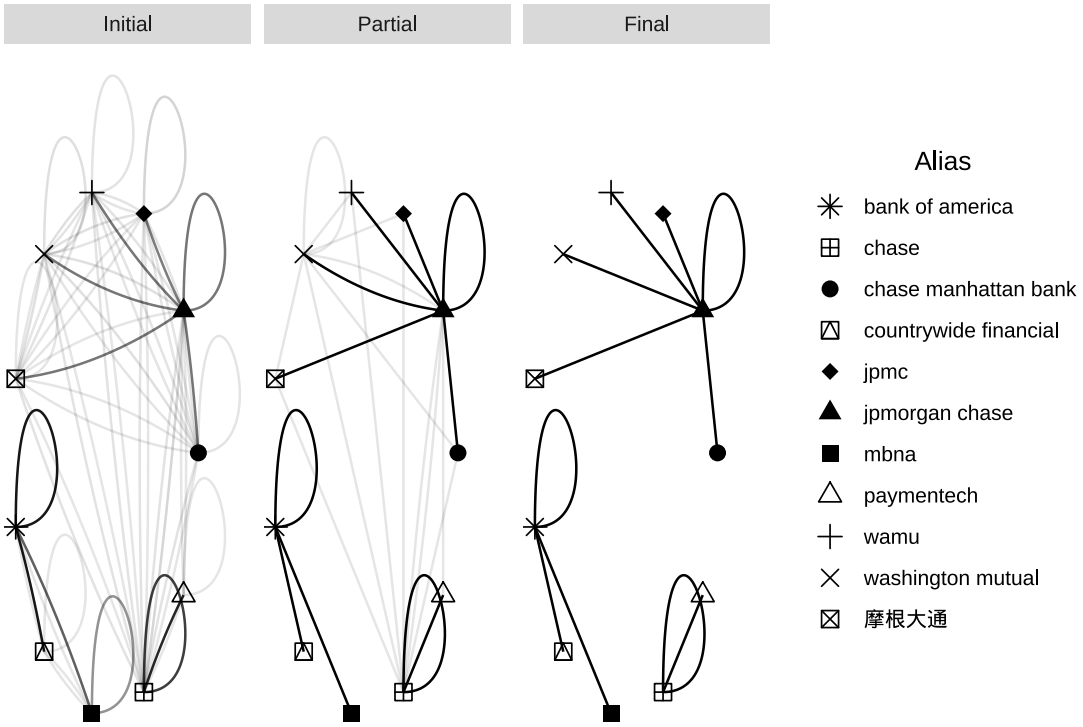

Figure A.II.1: *Illustration of Markov clustering.*

The intuition in Equation 10 is that the modularity score is maximized when the number of ties between a and u is large (i.e., B_{au} is big) and a, u are placed in the same cluster (so that $c_a = c_u$). We obtain community estimates based on the greedy optimization of Equation 10 (see Clauset et al. (2004) for details).

Appendix III: Additional Fuzzy String Distance Details

A.III.1.1 Fuzzy String Matching Details

Fuzzy string distances in our baseline linkage method are calculated as follows. Let $\tilde{a}$ and $\tilde{a}'$ denote the decomposition of organizational aliases a and a' into all their q-character combinations (known as q-grams). For example, if $q = 2$, "bank" would be decomposed into the set {"ba","an","nk"}. The Jaccard measure is then defined to be

$$d(a, a') = 1 - \frac{|\tilde{a} \cap \tilde{a}'|}{|\tilde{a} \cup \tilde{a}'|}.$$

If all q-grams co-occur within a and a', this measure returns 0. If none co-occur, the measure returns 1. If exactly half co-occur, the measure returns 0.5. Following (Navarro and Salmela, 2009), we set $q = 2$.

Appendix IV: A Task Matching English and Non-English Company Names

In this supplementary evaluation task, we examine the performance of our approach in a particularly challenging case where we seek to match organizational entities using their names written in English and in Mandarin Chinese. This matching task is especially challenging because English and Mandarin Chinese are based on two entirely different kinds of writing systems (i.e., alphabetic and logographic).

The data for this task is from FluentU, an organization focusing on language learning (*FluentU* 2022). The organization has provided a directory of 76 companies with their English names paired with their Chinese names. Table A.IV.1 displays some of the companies found in this dataset, which is composed primarily of large multinational corporations.

English Name	Chinese Name
amazon	亚马逊
coca cola	可口可乐
jp morgan	摩根
marlboro	万宝路

Table A.IV.1: *A sample of the organizational entities in the cross-language dataset. We attempt to match the pool of English names to their associated names in Mandarin Chinese.*

We compare linkage performance using our community detection algorithms based on the LinkedIn network and using our character-based machine learning and the baseline fuzzy matching approach. The fuzzy matching approach gives *no* matches (except when the fuzzy distance threshold is set to 1 so that all English names are matched with all Chinese names). This occurs because none of the English names share any Latin character combinations with any of the Chinese names.

The machine learning linkage approach based on the LinkedIn network fares somewhat better—yielding a true positive rate of 0.2 when the matched dataset size approaches 1,000. In contrast to fuzzy matching, it also achieves a non-zero F_2 score. However, the community detection approaches using the LinkedIn network fare better, successfully picking out 33 of the 76 matches with only five false positives. The left panel of Figure A.IV.1 shows how the Markov clustering approach somewhat outperforms the bipartite clustering, although both approaches achieve similar maximum true positive rates and F_2 scores just above 0.4. In this way, the network information present in the LinkedIn corpus allows us to find almost half of the true matches, even though the names to be matched are written in entirely different language systems.

Still, it is once again instructive to examine names for which the community detection approaches succeed and fail in this cross-language matching task. The approach succeeds in cases where an organization has employees who use both English and non-English organizational names while linking to the same company URL. For example, we successfully match the "Coca-cola" to "可口可乐" match, as Coca-cola has at least six Chinese offices, and employees in these offices often use Coca-Cola's Chinese name in their LinkedIn URL. However, we do not find the "lamborghini" to "兰博基尼" match. Lamborghini does operate offices in China, but its employees use the name "Lamborghini" in their LinkedIn profiles. In this way, the community-detection algorithms based

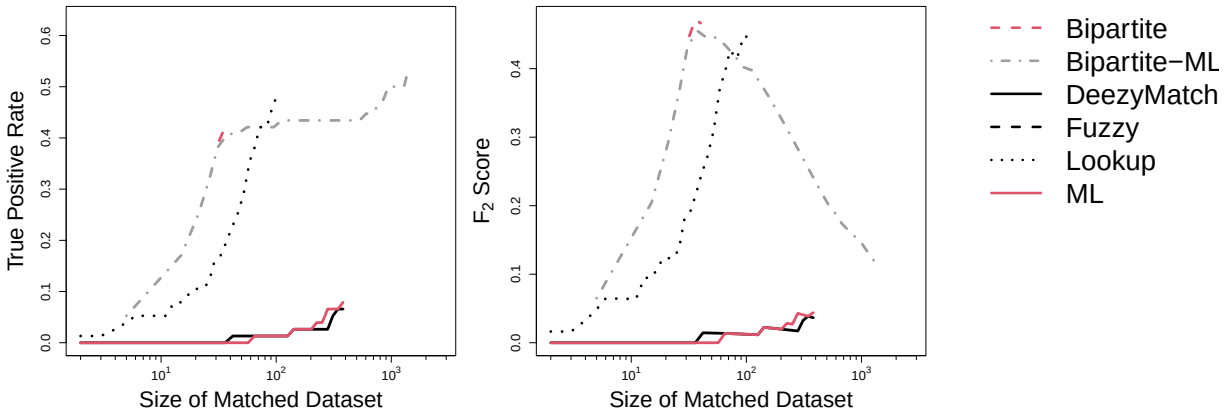

Figure A.IV.1: *In the left panel, we see that the LinkedIn-based community detection algorithms find a significant fraction of the true positive matches between the English and non-English aliases. In the right panel, we see that these true positive matches are found without introducing numerous false positives (so that the F_2 scores for the community detection algorithms approach 0.5). The machine learning approach yields some true positive matches; fuzzy matching yields none.*

on the LinkedIn network are, for cross-language merge tasks, best suited for the matching of organizational entities that have many employees across linguistic contexts and where non-English names are commonly used on LinkedIn.

Appendix V: A LinkOrgs Package Tutorial

The package can be downloaded via `GitHub`:

```
# Download package via github
devtools::install_github("cjerzak/LinkOrgs-software/LinkOrgs")

# load it into your R session
library(LinkOrgs)
```

There are several major functions in the package. The first is `FastFuzzyMatch`, which performs traditional string distance matching using all available CPU cores. This offers a substantial time speedup compared to the sequential application of a fuzzy distance calculator. The second function, `LinkOrgs`, is the main estimation function used in this paper to calculate the match probabilities. We also have a function, `AssessMatchPerformance`, that computes performance metrics of interest. To get help with any of these functions, one can run the following:

```
# To see package documentation, enter
# ?LinkOrgs::LinkOrgs
# ?LinkOrgs::AssessMatchPerformance
# ?LinkOrgs::FastFuzzyMatch
```

Here is the syntax for how to use the package using synthetic data:

```
# Create synthetic data to try everything out
x_orgnames <- c("apple","oracle","enron inc.","mcdonalds corporation")
y_orgnames <- c("apple corp","oracle inc","enron","mcdonalds co")
x <- data.frame("orgnames_x"=x_orgnames)
y <- data.frame("orgnames_y"=y_orgnames)

# Perform a simple merge with the package
linkedOrgs <- LinkOrgs(x = x,
                       y = y,
                       by.x = "orgnames_x",
                       by.y = "orgnames_y",
                       algorithm = "bipartite")

# Print results
print( linkedOrgs )
```

An up-to-date tutorial for the `LinkOrgs` package is available at

`GitHub.com/cjerzak/LinkOrgs-software/blob/master/README.md`

Appendix VI: Algorithm Compute Times by Task

A.VI.1.1 Matching Execution Times by Algorithm by Task in Minutes

Table A.VI.1: *Run time on the meetings data analysis.*

Algorithm	Run Time (mins)
Bipartite	13.12
Bipartite-ML	251.38
DeezyMatch	0.24
Fuzzy	0.27
Lookup	1.35
Markov	8.61
Markov-ML	113.00
ML	1.63

Table A.VI.2: *Run time on the company lobbying data analysis.*

Algorithm	Run Time (mins)
Bipartite	13.40
Bipartite-ML	318.44
DeezyMatch	0.32
Fuzzy	0.33
Lookup	1.36
Markov	9.47
Markov-ML	142.50
ML	2.09

Table A.VI.3: *Run time on the cross-language merge task.*

Algorithm	Run Time (mins)
Bipartite	0.70
Bipartite-ML	6.17
DeezyMatch	0.15
Fuzzy	0.02
Lookup	1.11
Markov	0.29
Markov-ML	3.52
ML	1.39

Table A.VI.4: *Run time on the Y Combinator task.*

Algorithm	Run Time (mins)
Bipartite	7.63
Bipartite-ML	142.84
DeezyMatch	0.33
Fuzzy	0.15
Lookup	1.26
Markov	4.41
Markov-ML	66.16
ML	1.22

Appendix VII: Additional Empirical Results

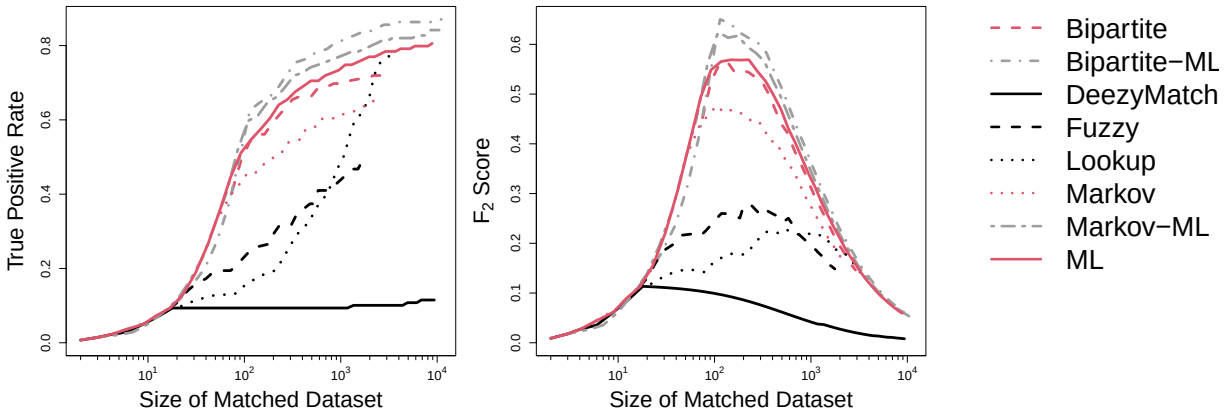

Figure A.VII.1: *We find that dataset linkage using any one of the approaches using the LinkedIn network obtains favorable performance relative to fuzzy string matching both when examining only the raw percentage of correct matches obtained (left panel) and when adjusting for the rate of false positives and false negatives in the F_2 score (right panel). In both figures, higher values along the Y-axis are better. The "Bipartite" and "Markov" refer to two network-based approaches to linkage. "ML" refers to the machine learning approach introduced above. "Fuzzy", "DeezyMatch", and "Lookup" refer to the string distance, machine learning, and network method baseline methods. "Markov-ML" and "Bipartite-ML" refer to the ensemble approaches.*

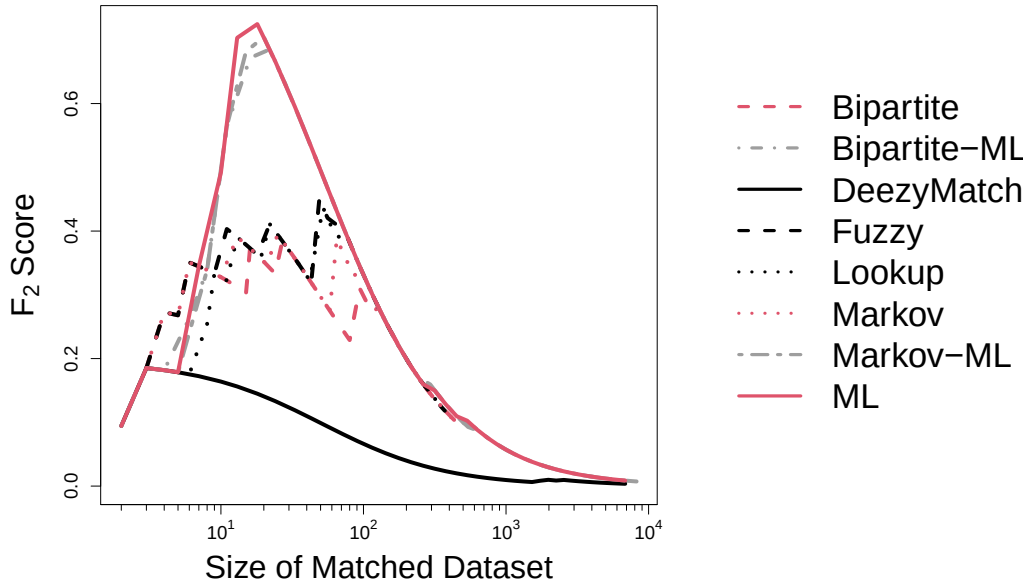

Figure A.VII.2: *In this YCombinator example, we see that the network-based approaches offer no relative benefit in terms of true positives when adjusting for false positives, yet the machine-learning-assisted approaches using the LinkedIn corpus perform well over fuzzy matching. Higher values along the Y-axis are better.*

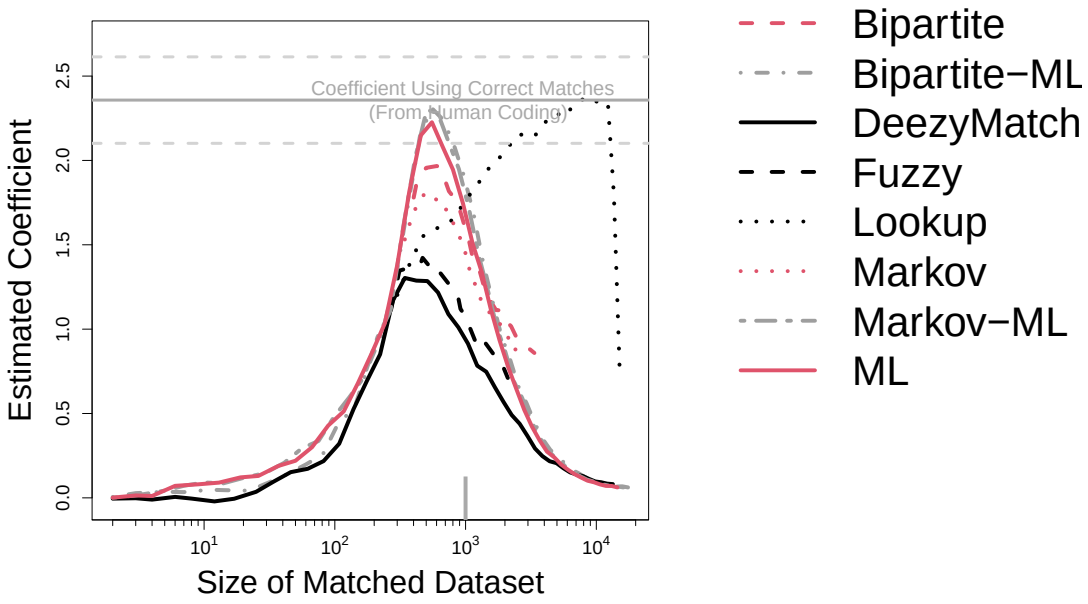

Figure A.VII.3: *The coefficient on log(Assets) for predicting log(1+Expenditures) using the ground truth data is about 2.5 (bold gray line, 95% confidence interval displayed using dotted gray lines). At its best point, fuzzy matching underestimates this quantity by about half. The LinkedIn-based matching algorithms better recover the coefficient.*

Appendix VIII: Data Availability Statement

The LinkedIn data corpus is available in a Harvard Dataverse:

doi.org/10.7910/DVN/EHRQQL

as well as a Hugging Face repository:

HuggingFace.co/datasets/cjerzak/LinkOrgs.

All methods are accessible in an open-source codebase available at

GitHub.com/cjerzak/LinkOrgs-software.

References

Assis Zampirolli, Francisco de and Leonardo Filipe (2017): "A Fast CUDA-based Implementation for the Euclidean Distance Transform". In: *2017 International Conference on High Performance Computing & Simulation (HPCS)*. IEEE, pp. 815–818.

Clauset, Aaron et al. (2004): "Finding Community Structure in Very Large Networks". In: *Physical Review E*, no. 6, vol. 70, p. 066111.

FluentU (2022). `https://www.fluentu.com/blog/chinese/business-chinese-vocabulary-list-foreign-companies-in-chinese/`. Accessed: 2022-01-01.

Hansen, Ben B (2007): "Optmatch: Flexible, Optimal Matching for Observational Studies". In: *New Functions for Multivariate Analysis*, no. 2, vol. 7, pp. 18–24.

Jiang, Albert Q et al. (2023): "Mistral 7B". In: *arXiv preprint arXiv:2310.06825*.

Kim, Jaekyeom et al. (2020): "Model-agnostic Boundary-adversarial Sampling for Test-time Generalization in Few-shot Learning". In: *European Conference on Computer Vision*. Springer, pp. 599–617.

Kuhn, Max and Kjell Johnson (2013): "Remedies for Severe Class Imbalance". In: *Applied Predictive Modeling*. Springer, pp. 419–443.

Navarro, Gonzalo and Leena Salmela (2009): "Indexing Variable Length Substrings for Exact and Approximate Matching". In: *International Symposium on String Processing and Information Retrieval*, pp. 214–221.

Rish, Irina et al. (2001): "An Empirical Study of the Naive Bayes classifier". In: *IJCAI 2001 Workshop on Empirical Methods in Artificial Intelligence*. Vol. 3. 22, pp. 41–46.

Santurkar, Shibani et al. (2018): "How Does Batch Normalization Help Optimization?" In: *Advances in Neural Information Processing Systems*, vol. 31.

Van Dongen, Stijn (2008): "Graph Clustering Via a Discrete Uncoupling Process". In: *SIAM Journal on Matrix Analysis and Applications*, no. 1, vol. 30, pp. 121–141.

Wei, Jason et al. (2022): "Emergent Abilities of Large Language Models". In: *arXiv preprint arXiv:2206.07682*.