The SearchEngine: a Holistic Approach to Matching

Thorsten Doherr[a]

November 2022

a) ZEW - Leibniz-Centre for European Economic Research

Abstract The SearchEngine is an open source project providing an integrated framework for diverse matching activities, especially the linkage of large scale firm data by fuzzy criteria like company names and addresses. At its core, it utilizes an efficient candidate retrieval mechanism implementing a word respectively token driven heuristic. Every record in one table becomes a search term to retrieve similar candidate records in the base table according to a search strategy replacing blocking strategies of conventional matching efforts. Because similarity is inherently established by the candidate selection, it is only required to filter false positives by using the meta data export file derived from the matching heuristic to implement a machine learning approach. This paper discusses the general foundation of the heuristic and the algorithm while two detailed walkthroughs of company linkages show practical examples.

Keywords data linkage, firm matching, entity resolution, machine learning

JEL C81, C88

Acknowledgements The access to the establishment data for the main example was granted by courtesy of the Institute of Employment Research (IAB) of the Federal Employment Agency of Germany. I especially thank Stefanie Wolter of their Research Data Centre for initiating and supporting this project.

1 Introduction

Matching two databases of unrelated origin is one of the most common tasks in data science expanding the research opportunities way beyond the potential of the isolated databases. Patent data alone provides ample insights of the spatial and temporal distribution of technologies and their markets but linking the patent applicants to firm level databases extends the portfolio from bibliometric assays to economic research of innovation. In this paper, we will take recourse to the matching of those databases because it illustrates how the different foci of the data collecting organizations directly influences the effort required to link their respective entities. Firm level data usually is provided by rating agencies, data sellers or government organizations who take the company into the main focus of their endeavors entailing a pristine definition of this entity because all information relates to it. Patent data is managed by patent offices that take great pride in maintaining the documentation of the technological progress of humankind distilled in legal documents. The focus is on the authentic replication of those documents in their database. This entails a fuzzy identification of the patent owner/applicant by name and address only because patent offices do not administer specialized address databases. This negligence causes a high amount of variation for an entity which is not even identified as such. In the context of this paper, we refer to those databases as unfocused, while databases that maintain the linkage criteria, in our case companies, as an entity are called focused.

To match the records of those databases, we need to find linkage criteria defined by intersecting data. If this intersection is a mutual unique key like a common company-id, i.e. VAT number, the matching task is resolved. We will discuss what is required, if this is not the case. In our prior example the intersection between patents and firm data comprises the firm, respectively applicant name, and the address. The obvious approach would be to harmonize the linkage fields in both data sources to increase the yield of linked records by normalizing character case, translating special characters, erasing stop words, applying phonetic transformations as well as hashing methods and so on to achieve a level of harmonization granting a reliable linkage without compromising the precision. The main disadvantage of this method is the absence of any inherent quality measurement of a match as it is always the result of a perfect join of harmonized data. Even sequential matching based on different levels of harmonization can only provide ordinal quality classifications. The identification of stop words is an onerous task, especially if international data is involved. Finally, the divergent positioning of words enforces standardized repositioning, which is highly sensitive to additional words. In general, harmonization methods are characterized by a high precision at the cost of recall. The latter can be improved by applying blocking strategies imposing exclusion restrictions on the data, therefore creating blocks of data with a reasonable probability of containing valid matches. A block can be defined by shared field attributes, like postal region, or by more complex spatial clustering methods,

like locality sensitive hashing as suggested by Steorts, Ventura, Sadinle and Fienberg (2014). Blocking reduces the computational complexity facilitating the application of supervised machine learning (ML) approaches, which were not feasible in the complete solution space of all pairwise combinations. Carefully stratified samples, manually scrutinized, provide the training data based on meta data, quality and distance measurements. Comber and Aries-Bel (2019) augmented this approach by comparing the ML methods of word2vec and conditional random fields for encoding distance measurements of text fields. The majority of the literature is about the efforts in the field of address matching rooting in spatial science. The address itself is the only purpose of the exercise. The inclusion of a specific purpose of an address, beyond referencing a location, shifts the focus to whatever can be found at this address, for instance a company. In firm related databases the context defining field is not the address but the firm diminishing the importance of the address. Blocking strategies become considerably more complex with the additional dimension aggravated by the fact that firm names are less structured than street names. Monath, Jones and Madhavan (2021) addressed this issue with so called canopies, a blocking strategy based on word prefixes of the firm name. This method does not require address fields while the overlap of the blocks mitigates the risk of unobserved false negatives. Still, they described occasions where the resulting blocks proved too large for pairwise processing. Our approach resolves this general shortcoming of blocking by implementing a token based heuristic capturing the idiosyncrasies of the data thus replacing blocking with dynamic candidate retrieval complemented with intrinsic meta data driving ML approaches to discriminate false from true positives within individual blocks of candidates.

The center stage is taken by a Registry containing all words of the relevant fields of one database. During the initialization of the Registry, only basic harmonization is performed on the words, which become entries in the dictionary along with their frequencies in the respective fields defining separate chapters. Additional field related chapters can be established to contain a specific tokenization of words like phonetic representations or n-grams to increase robustness versus misspellings and typos. Every entry can be traced back to all records in the database containing this token or word and vice versa via internal index tables. The frequencies are required to separate the identifying words from the filler respectively stop words. A high frequency designates a stop word as the potential to narrow the search space is low, while a low frequency word contributes to the relevancy of the associated candidate list. The underlying engine attaches a weight to every word of a search term based on its frequency and an arbitrary weighting scheme superimposed on the chapters of the Registry to implement a search strategy usually setting a higher weight on the context defining fields/chapters. If matching of companies is intended, the weight distribution should be skewed towards the firm name while the address fields fulfill auxiliary purposes. The words are processed in descending order of their weights by retrieving the respective list of candidates. A candidate accumulates the weights of the

associated lists. Only candidates passing a threshold are selected into the final list representing the result of the search term. This procedure can be repeated to implement incremental search strategies by merging the result sets, which are not commutative. The base table always provides the data feeding the Registry, while the records of other tables provide the search terms. The decision, which table should become the base table, is determined by structural properties of the tables. Size, prevalence of redundant noise and the general focus of the data acquisition are the main factors of this decision. As redundant noise causes more harm in the search term, noisy tables should have a prior to become base tables because additional noise on the candidates has no impact on the search quality unless it is explicitly intended to rank the results by the relevancy of the candidate noise. Focused data acquisition avoids redundancies by maintaining entity related tables, where an entity, like a firm, is assigned a unique key and variants to its representation in the data are only allowed to capture temporal changes in the context of a panel. As base tables, these data sources facilitate much stricter search strategies already eliminating the brunt of false positives by waiving unnecessary tolerance towards ambiguity in the representation of entities. In case of target conflicts between noise and focus, it is reassuring that a decision towards noise and therefore towards recall at the cost of precision can be mitigated with integral machine learning approaches filtering false positives.

The algorithm is capable to produce candidate lists for every search term replacing the conventional multi-layered blocking of data along arbitrary conditionalities, like same region, similar address, same harmonized respectively hashed name or canopies. Blocking is required to reduce the solution space and to provide quality measurements to drive statistical or ML models based on ground truths usually sampled from the current data. As the blocking of our algorithm completely relies on already registered frequencies the quality measurements are readily available. While the search process exploits the frequencies to create a relative order of the words within a search term, the general quality of a candidate is gauged by the absolute frequencies of the matching words, the candidate exclusive words and the omitted words of the search term. To retrieve the latter, it is required to create a separate Registry for the search table. Together with the accumulated weight sum, the number of co-candidates, its position within the list of co-candidates, the number of overlapping words between the different fields of the search term, string similarity and other derived information, every match engenders a multitude of data points to predict false positives among the candidates based on a ML model trained on a manually scrutinized sample.

The 2. Chapter describes the general workflow and the components of the standalone tool “SearchEngine” implementing the search respectively candidate retrieval heuristic. That Chapter contains an interlude delving into the particularities of self-referential searches and clustering followed by the consecutive implementation of ML methods based on the meta output of the SearchEngine to

discriminate between true and false positives. The paper concludes with detailed discussions of two large scale company database linkages.

2 The SearchEngine

2.1 Workflow

Given the diversity of matching tasks it is imperative to be able to rely on standardized routines easing the decision processes accompanying those. Deploying a tool with a dedicated focus on matching and a limited but specialized functional scope funnels the workflow. The first decision pertains the base table, which provides frequencies of the words constituting the meta information for the heuristic approach. The fields have to be in the same format as the corresponding fields of the external search table providing the search terms comprising those fields. This can lead to preliminary data adjustments, like concatenating fields to achieve structural congruency. In case of unfocused data sources or panel data, it is required to remove duplicate entries while maintaining a linkage key in the original data. This procedure should be applied to all tables to avoid redundancies and distorted frequency distributions. Basic harmonization may improve the level of compression. The next step after the assignment of the base table and the structural synchronization of the mutual fields is the declaration of the chapters in the Registry. Every chapter represents a so called Search Field in conjunction with a specific harmonization respectively tokenization of that field. Besides baseline harmonization regarding character case, language specific characters and removal of non-literal characters, the SearchEngine provides additional context specific routines that need to be manually attached to a field. These routines, called Preparers, usually split or curtail words in accordance with the context, e.g. separating house number appendages (12A into 12 A). We refer to the SearchEngine manual for a full list of Preparers. A word is delineated by space characters after harmonization. Multiple Preparers can be attached to a field defining a chapter in the Registry, from now on called Search Type. The charm of having multiple Search Types for a Search Field lies in the increased flexibility of potential search strategies. Some Preparers implement methods from the area of computational linguistics like n-grams or phonetic transformations (Metaphone, Soundex, …) to provide robustness towards misspellings. Their capacity to handle misspellings is ascribed to the aggregation respectively destruction of information bestowing them the moniker Destructive Preparers. A search strategy is constituted by the distribution of weights on the Search Types. A proven approach for a search on firm addresses is to assign a weight of 70% on the basic Search Type for the firm name, not implementing destructive Preparers, and distribute the remaining 30% on the address fields. With a threshold of 70%, there is a still a chance to capture a relocated firm while similarities in the address fields allow for some leeway in the firm name. These first candidates do not contain entries distorted by misspellings, which can be retrieved by a subsequent search run based on a shift of the

70% weight on a Search Type implementing n-grams for the firm name. The reason to not always use Destructive Preparers consists in their inherent tendency of retrieving false positives, but deploying a two thronged strategy of conventional and destructive searches enables tolerance without compromising integrity. In section 2.2.3 we will discuss this feature in detail.

After a pertinent amount of search runs, the Result Table contains candidates originating from different steps of the search strategy. The higher the ambition to fetch all potential valid matches, the higher the risk of catching unwanted false positives. The Extended Export files of the SearchEngine provide all the information required to manually scrutinize the matches accordingly. They are structured into distinctive blocks comprising of the original fields of the search terms, i.e. records of the search table, followed by the respective fields of the candidates, i.e. records of the base table. Record numbers or associated key entries accompany the entries along with a similarity index, called Identity, a Score expressing the overall uniqueness of the search term and the number of the search run responsible for obtaining the candidate. Table 8 in the Appendix shows an excerpt of a labeled Extended Export file. In case of a moderate number of search records, a completely manual validation step yields the best results. If checking larger datasets proves to be too time consuming, smaller samples can be drawn for manual inspection using a built in export feature. Those samples provide the ground truth for a machine learning approach based on another export format collecting a smorgasbord of meta information about the matches. A Neural Network or other ML models can be trained to predict true positives within the whole population of matches. The SearchEngine package contains specific STATA modules for this purpose but the output is suited to apply independent ML approaches.

2.2 The Heuristic

2.2.1 Relative Identification Potential

The heuristic is based on the assumption that the occurrence of a word is inverse proportional to the identification potential IP of this word. Using the internet as an analogy, a quite common word entered into a web search will result in a large list of pages making it difficult to find the intended entry. The resulting list of potential candidates for a rare word is smaller and therefore the identification potential higher. Because a search usually involves more than one word, the algorithm uses a relative identification potential rIP. The following section describes the development of this measurement starting with a very basic first version:

$$rIP(w) = \frac{occ(w)^{-1}}{\sum_{v \in S} occ(v)^{-1}} \quad (1)$$

with S being a set of words composed by the search term, $w \in S$ and $occ(w)$ returning the occurrence of the word w. In this version, the Registry, providing the frequencies retrieved by the occ function, is not divided into Search Fields and the search term is considered an unstructured bag of words.

By introducing Search Types as specific normalizations/tokenizations of Search Fields, the mutual fields of the base table and the search table, we divide the Registry into chapters. The same word can appear in different chapters with different frequencies to preserve the context of the word. By applying arbitrary weights on the Search Types, a search strategy can be implemented. The weights are defined as shares accumulating to 1. The occurrence function occ now requires two parameters: the word and the Search Type to retrieve the associated frequency. With $st(w)$ returning the Search Type of word w, $weight(st(w))$ returning the arbitrary weight of the Search Type of word w, the extended rIP can be defined as:

$$IP(w) = occ\big(w, st(w)\big)^{-1} \tag{2}$$

$$rIP(w) = weight(st(w)) \left(\frac{IP(w)}{\sum_{v \in S} \begin{cases} IP(v) & | \; st(w) = st(v) \\ 0 & | \; st(w) \neq st(v) \end{cases}} \right) \tag{3}$$

The function $occ(w, st(w))$ returns the average frequency within the Search Type $st(w)$

in case word w is not found in the Registry for the specified Search Type. The similarity of a candidate C with a search term S is defined by the intersection of matching words:

$$I(S, C) = \sum_{w \in S \cap C} rIP(w) \tag{4}$$

This sum is called Identity I of a candidate. If the overlap of mutual words between a candidate and the search term comprises all words of S, the accumulated rIP amounts to 1. If a candidate does not contain all words of the search request, its Identity is reduced by the rIP of the missing search words. Surplus words of a candidate are not considered for retrieval but can be used to rank them by the relevance of surplus noise called Feedback (see 2.2.5). A threshold for the accumulated Identity is applied in accordance with the search strategy as a barrier for the candidates. Internally, a candidate is represented by a record number of the Base Table. Given that computing power is a limited resource, it is beneficial to prioritize words in descending order of their rIP to exploit the fact that words with low frequencies have higher potentials. For every word in that sequence, the associated candidates are collected into a candidate buffer until the gradually accumulated rIP is above 1 minus the threshold. This reflects the fact that candidate records containing only the remaining search words will not be able to achieve an Identity above the threshold and become redundant. The attached rIP of

the buffered candidates will be aggregated into an intermediate Identity. If the rIP sum of the unused search words will not push a candidate over the threshold, it will be removed from the buffer. For every surviving candidate, the words of the corresponding base record not used for buffering are intersected with the unused words of the search term to accumulate the remaining rIP to obtain the final Identity to be checked against the threshold. This is the most time-consuming component of the procedure, which explains the preceding effort to avoid superfluous candidates. The overall performance can be optimized by introducing a maximum Search Depth limiting the buffer size to keep a healthy balance between performance and completeness. This feature prevents excessive resource usage caused by weak search terms consisting of stop words with extreme frequencies.

2.2.2 Absolute Identification Potential

The Identity is the main output of the SearchEngine algorithm, but it is only a relative measurement of similarity. It does not allow to compare the quality of matches, especially if they are close to the threshold. A match consisting of an assortment of exotic words can still be considered of higher quality than a match based on common words with a similar Identity. A typical indicator for a weak search term is a high number of candidates. Conversely, a strong search term has a tendency for small candidate lists. Besides the size of the candidate list the SearchEngine also reports the so called Score for every search term S, which is an arbitrary indicator for the absolute identification potential with a straight forward design:

$$score(S) = \sum_{w \in S} \frac{weight\big(st(w)\big)}{occ\big(w, st(w)\big)} \tag{14}$$

It relies on the current weight distribution and is therefore not consistent between consecutive search runs with alternating strategies. The Score increases with the weight of a word according to its Search Type and decreases with its occurrence. A search term with a high Score has a higher probability to yield proper results even with a low Identity. Together with the candidate count, it is part of the standard output format of the SearchEngine and a component of the meta output codifying matching results for the subsequent machine learning (see 2.4).

2.2.3 Misspellings

The SearchEngine uses a word based algorithm. If a word w cannot be found in the Registry for a specific Search Type, the $rIP(w)$ will be based on the average word frequency of this Search Type. As no actual records of the base table are connected to this word, it will become a dead weight reducing the overall achievable Identity by its rIP. This is acceptable for a real word that in fact does not exists in the Registry but should be avoided for existing words distorted by a misspelling. It is an even greater annoyance, if a misspelled common word actually appears in the Registry attracting the dominant rIP of the search term, due to the scarcity of its misspelling, the original word would have never obtained.

These words can misguide the search process in favor of the candidates containing the misspelled entries forfeiting the correctly spelled entries. Destructive Preparers, implementing phonetic methods like Soundex and Metaphone or the transformation into n-grams, reduce this problem by creating codes for similar sounding words or by diluting the impact of misspellings through overlapping tokenization (i.e.: n_gram(3,"DOHERR") = ["DOH", "OHE", "HER", "ERR”]) as they concern only a fraction of the tokens and not the whole word.

There is a price to pay for the gain in robustness. The algorithm will return much more false positives because the phonetic representations do not only include the misspellings but also similar "legal" words (see Table 1). In the case of n-grams, all candidates containing the same tokens have the same identity regardless of positioning causing unexpected results. These Preparers are called destructive for a reason. They deliberately destroy information for the sake of an increased recall. Phonetic and linguistic methods were originally designed for a supervised environment where an operator verifies the plausibility of retrieved candidates long before computers replaced physical file cabinets. The patent for Soundex was submitted 1918 by Robert C. Russell und Margaret King Odell while the origin of n-grams stays shrouded besides being already mentioned by Wolfgang Schönpflug in 1969 (“N-Gramm-Häufigkeiten in der deutschen Sprache”).

Method	**Soundex**	**Metaphone**	**Cologne**
Code	T652	BRTN	3467
Example 1	TARNOWSKI	BARATON	WAGNER
Example 2	THORENZ	BERTINI	WUCHENAUER
Example 3	TRUNK	BORDIN	WEGENER

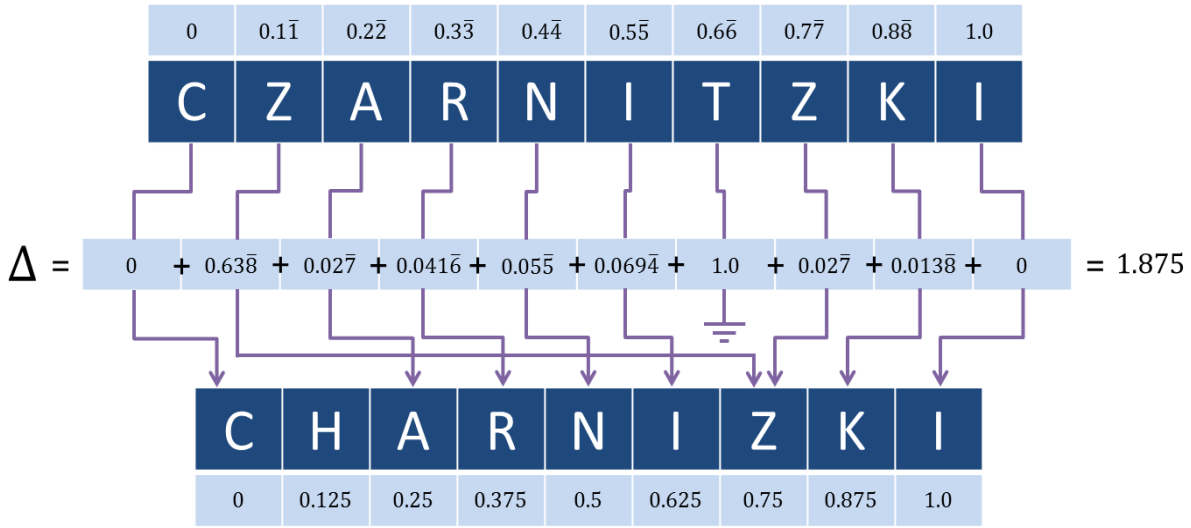
Table 1: Words represented by the same phonetic code

With the progress of computational capacities, the task of an operator can be replaced with an additional layer applying a string comparison function, simulating the ogling of the operator, for every Search Type implementing Destructive Preparers. This function returns a value between 0 and 1 for the similarity of strings. After the preliminary candidates of a search term have been collected using the basic algorithm, the refinement layer retrieves the linked records from the base and search table to compare the original content of fields affected by Destructive Preparers. The partial Identities of those Search Types are replaced by the weighted similarity indices to form a hybrid Identity.

For every affected field in the search term, the similarity function compares every word of the search term with every word of the candidate term to identify the pairings with the highest similarities after basic normalization. The final result is the sum of these values divided by the number of words in the search term field. An additional index will be calculated that compares the terms as whole strings after removing blanks. The weighted maximum of both scores replaces the Identity share of the Search Type

for a specific candidate. Both types of comparisons are necessary to guarantee a high flexibility of the measurement towards diverging positioning of words and uncleanly separated words (i.e. missing blanks). Because this flexibility requires a large number of comparisons, the underlying algorithm has to be very efficient.

The string comparison method we invented for this purpose is called Least Relative Character Position Deltas (LRCPD) (Doherr, 2017). The function assigns a relative position between 0 for the first and 1 for the last character of the two strings consigned as parameters. For every character in the first string, the algorithm searches for the matching character in the second string with the smallest difference between the relative positions called delta. The starting character of every search in the second string is the character at roughly the same relative position than the respective character of the first string. From this position the search is conducted outward. If a character cannot be found, a maximum delta of 1 is assigned. The sum of the deltas divided by the length of the first string returns a disparity measure between 0 and 1.



$$lrcpd(w1, w2) = 1 - \frac{\Delta(w1, w2)}{len(w1)} = 1 - \frac{1.875}{10} = 0.8125$$

Figure 1: Least Character Position Delta (LRCPD) of a misspelled name

The LRCPD depends on the direction of the comparison. For a commutative behavior the comparison has to be done in both directions using the lower result. The SearchEngine allows for the specification of the direction of the comparison but, to be in accord with the general heuristic, the default setting is to compare the search term with the candidate term.

An issue is the atrophy of the deltas with increased string lengths. The limes of the average delta approaches zero for the comparison of infinite strings. For this reason, the LRCPD implements a search scope around the relative position of the respective starting character. Starting from this position the search will be carried out in both directions until the character is found or the absolute distance to the

start position exceeds the scope. The delta of a found character will be adjusted as if the string length equals this limit. The SearchEngine uses an arbitrary default scope of 12 characters in both directions (excluding the start position).

Search Types implementing phonetic Preparers somehow distort the idea behind the original heuristic. The codes or fragments returned by linguistic methods have a different distribution of frequencies than the unadulterated words. Through fragmentation or aggregation the number of words stored in the Registry is reduced, thereby the average occurrence increased, which leads to an inflation of candidates. This effect is subdued by the LRCPD layer but the main advantage of the original heuristic, finding candidates based on the Identification Potential of words, is watered down. Because of that, the SearchEngine supports incremental search steps. Multiple runs with different settings can be merged into one result set. Pairings of previous runs will not be overwritten by following search steps. It is advised to use linguistic Preparers for later runs to fetch only the candidates that actually have misspellings and to keep the main bulk of the results according to the heuristic especially in respect of subsequent ML approaches based on meta data derived from the Registry.

2.2.4 Smoothing

The general assumption that the Identification Potential of a word is represented by the number of candidates it will procure is not always sensible, because it does not account for the context of the Search Type. Some Search Fields have an intrinsic structure, requiring more than one word to properly identify a match. A street address, for instance, is not complete without the house number but the number alone has no identification potential at all. The rIP of a common street name can be reduced to insignificance by an extraordinarily high house number leading to matches based exclusively on the number. Although, there is no reason to favor high house number to lower ones, which almost every street is able to obtain. For street addresses, the direct translation of the inverse frequencies results in a too accentuated distribution of the relative Identification Potentials. The same is true for person names. The fact that a common first name has to compete with a rare last name does not mean that the first name can be omitted as a stop word. Finally, the misspelling of a word may lead to an infrequent n-gram dominating the rIP distribution counteracting the original purpose of diluting misspellings with this kind of linguistic method (see 2.2.3). Smoothing can be applied to mitigate this issue by enforcing a more conservative search behavior requiring more words to match. It can be assigned per Search Type to fine-tune the search strategy. The SearchEngine implements two approaches that can be combined. Both change the IP of a word before insertion into equation (3).

The offset based smoothing method adds an offset to the occurrence of a word. The offset can be positive or negative and is defined per Search Type via the $off(st(w))$ function. The $max()$ function returns the maximum of its parameters and prevents negative Identification Potentials.

$$IP(w) = max\big(occ\big(w, st(w)\big) + off\big(st(w)\big), 1\big)^{-1} \quad (4)$$

Negative offsets can be used to transform the frequency-based heuristic into a simple word-based metric in case the absolute value of the offset is larger than the highest frequency of the respective Search Type. The uncomfortable arbitrariness of offsets has led to the implementation of a widely accepted smoothing method: the logarithmic inverse word frequency ratio, which is a close relative to the inverse document frequency also based on a logarithmic distribution. As opposed to the latter, the ratio uses the maximum frequency of a Search Type instead of a document count.

$$IP(w) = max\left(ln\left(\frac{maxocc\big(st(w)\big)}{occ\big(w, st(w)\big)}\right), 1\right) \quad (5)$$

The function $maxocc(st(w))$ retrieves the maximum frequency within a Search Type. This smoothing method is applicable by a simple binary choice per Search Type and has mostly replaced the cumbersome offset method. For the sake of completeness, the SearchEngine allows for the combination of both methods:

$$IP(w) = max\left(ln\left(\frac{max\big(maxocc\big(st(w)\big) + off\big(st(w)\big), 1\big)}{max\big(occ\big(w, st(w)\big) + off\big(st(w)\big), 1\big)}\right), 1\right) \quad (6)$$

Smoothing is an ex-post decision as it can be switched on and off independently for every Search Type without reinitiating the creation of the Registry. Its main application is the adjustment of n-gram based frequency distributions to avoid rIP peaks just at the misspelling (see 2.2.3). Still, it can be beneficial to combine normal and smoothed search runs in incremental search steps.

2.2.5 Feedback

Hitherto, the heuristic completely disregards the surplus words of the candidate not matching the search term. This characteristic of the SearchEngine causes the general non-commutativity of the search function if the base table is used as search table in a self-referential search. This behavior is desirable, because it elevates the heuristic beyond the capabilities of simple harmonization, which, by definition, is commutative. Still, the disadvantage of the relativity of the rIP can lead to inflated candidate lists in cases of weak search terms with a low absolute Identification Potential (see 2.2.2). This surge can be mitigated by curtailing candidate lists exceeding a reasonable size. The candidates are sorted by their Identity in descending order to use the Identity of the candidate at the respective clipping position as the new temporary threshold. This approach prevents the inconsistent splitting of lists. Unfortunately, weak search terms have a notoriously low variance in terms of Identities rendering this mechanism almost useless. Additional variance has to be introduced by ranking the candidates by the relevance of contained words not represented by the search term. The surplus words of the candidates generate a discount on the Identity, called Feedback. The extent of the discount can be

adjusted with the Feedback parameter f which has a valid range from 0 to 1. With F being the set of all the words of the found candidate record, the final definition of the rIP is available:

$$Jaccard(w) = \frac{\sum_{v \in S} \begin{cases} IP(v) | \ st(w) = st(v) \\ 0 \quad\quad | \ st(w) \neq st(v) \end{cases}}{\sum_{v \in S \cup F} \begin{cases} IP(v) | \ st(w) = st(v) \\ 0 \quad\quad | \ st(w) \neq st(v) \end{cases}} \tag{7}$$

$$rIP(w, f) = rIP(w)\big((1 - f) + Jaccard(w)f\big) \tag{8}$$

The function name Jaccard refers to the Jaccard similarity coefficient (Paul Jaccard, 1902). It measures the similarity of two sets of properties by dividing the number of shared properties by the size of the union of both sets. To transform the Identity into a classical Jaccard index, we have to push the slide control f to 1, equalize the weights and smooth out any differences in the words by applying negative offsets exceeding the maximum frequencies of the respective Search Types. Of course, a dedicated word-based similarity index would be a more efficient implementation compared to subduing the frequency based heuristic of the SearchEngine, but applying just a hint of Jaccard with a low Feedback, i.e. $f = 0.1$, will engender the necessary variance to effectively clip inflated candidate lists. Unless specifically stated otherwise, the adjusted rIP will not be reported nor will it be reevaluated against the threshold. As there is little reason to always integrate Feedback, it is usually only activated when a candidate list exceeds the clipping limit based on user assessment.

2.3 Entity Resolution

We already have stated that the results of the SearchEngine are not commutative. The result of a search operation depends on the search direction, which is determined by the assignment of the base table. Of course, switching search and base table will yield completely different results. What if the base table and the search table are the same? This scenario arises, if unfocused data needs to be transformed into focused data meaning that all context defining entities in the data have to be resolved into separate clusters. A typical case for such unfocused data is applicants or inventors of patents, which are maintained by organizations focused on the digital preservation of documents regardless of the consistency of applicant or inventor names and their addresses. Name or address changes of companies or inventors, but also just different ways to express names and addresses, not to mention misspellings, cause variation in the representation of an entity. We have learned that the SearchEngine is capable to identify similar entries in two different data sources, which holds true for a self-referential search in one data source. The decisive difference consists in the interpretation of the matching results: the match tuples refer to the same data space. The simple one-way assignment from one set into another transforms into a definition of vertexes and edges to constitute a directed graph. Every candidate list for a search term can be interpreted as a graph of edges linking the search record with

the candidate records, which eventually become search terms by themselves to accumulate their respective candidates and so on. Because of the non-commutativity of the heuristic in conjunction with the threshold, the resulting graphs are not transitive and not complete. The threshold prevents that a reversed case exists for every tuple. There may be connections from A to B to C, but none from C to A or from B to A. The SearchEngine supports a function to enforce the mirrored version for every existing tuple to attach an Identity for these unobserved edges even if they are below the threshold. The Identity of the mirrored cases is based on a logarithmically smoothed IP distribution, which is better suited to represent similarity as retrieval is not of concern. Figure 2 shows an exemplary excerpt from such a graph with mirrored edges. The size of the letters represents the rIP of words within the candidate vertices. The larger the letter the more dominant the word. The black arrows represent the original matches of a search with a 90% threshold, while the mirrored connections are depicted as light arrows fitted with Identities below the threshold.

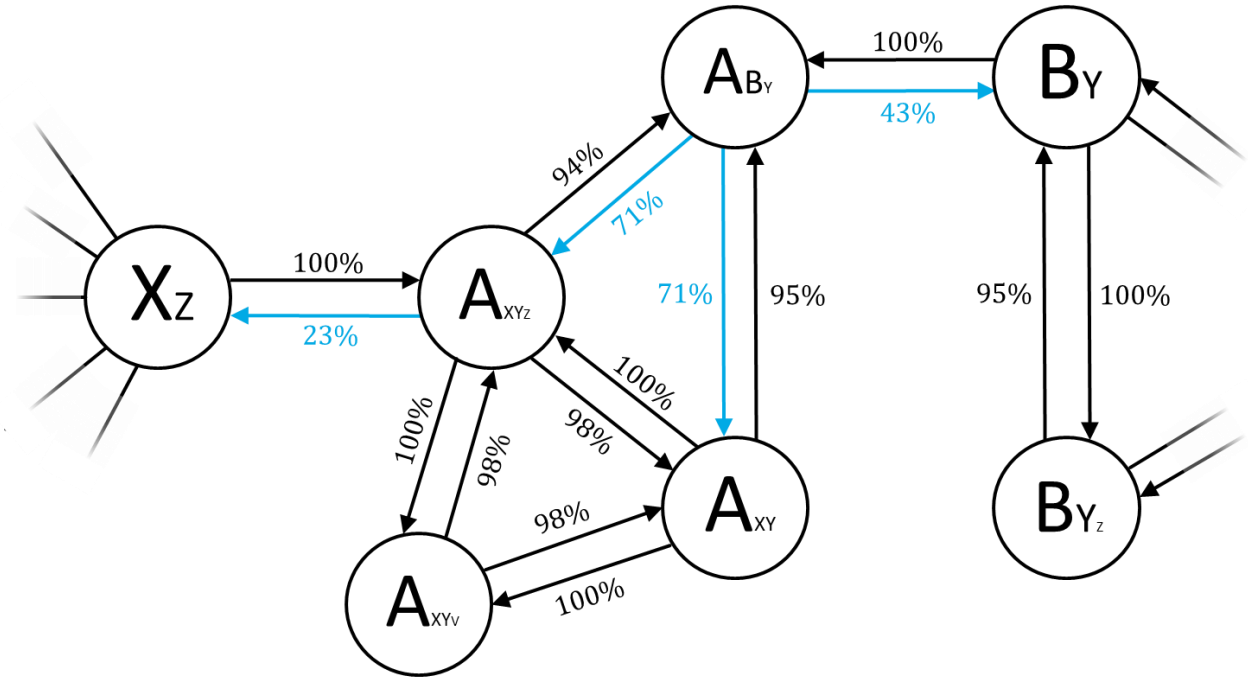


Figure 2: Directed candidate graph with enforced bi-directionality

By transforming the directed graph into an undirected graph, the bi-directional connections provide additional information about the similarity relationship of two candidates beyond the Identity in both directions. Every undirected edge has a maximum max and a minimum min identity. The minimum of the absolute Identification Potential, the Score s, of both candidates as search terms is attached to an edge. This value represents the overall "exoticness" of the match. And finally, to mitigate the arbitrariness of the Score, the percentile p of the respective Score completes the value attributes distinguishing every undirected edge. Figure 3 shows the candidate network of Figure 2 after the transformation into an undirected graph.

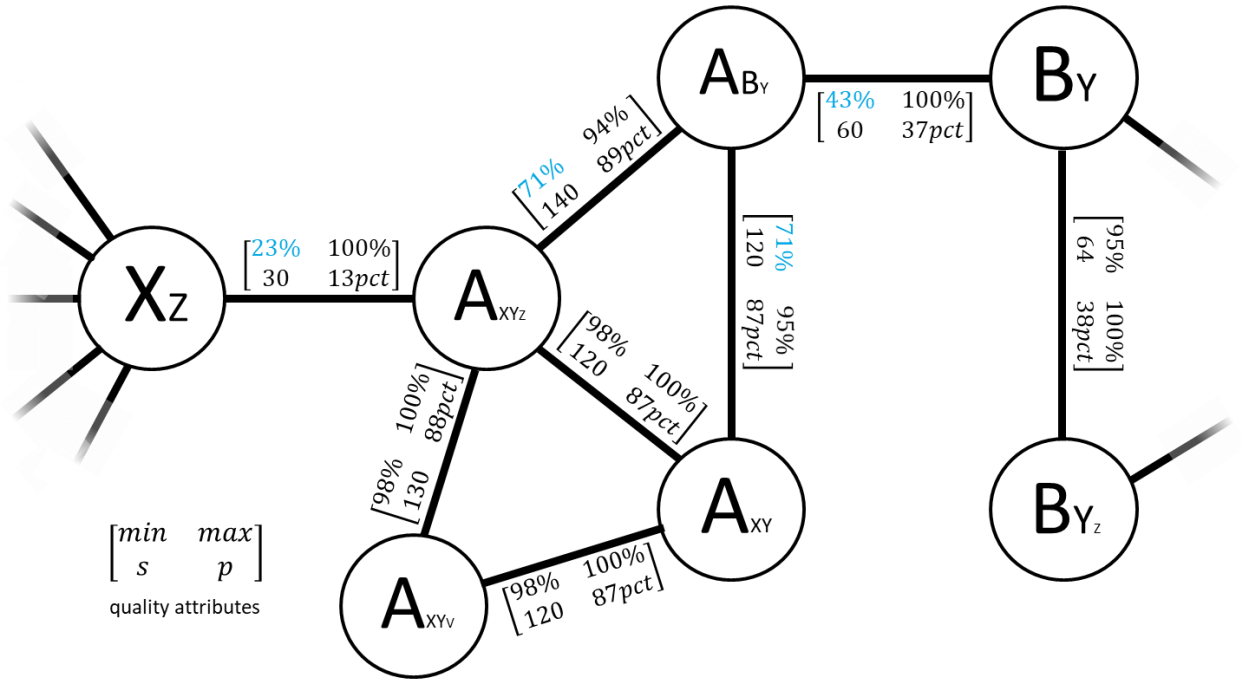


Figure 3: Transformation into an undirected candidate graph

To identify the entities within such a network, we have to recursively traverse the edges, collecting encountered nodes into enclosed entity clusters. Unfortunately, nodes with a very low absolute IP (Score) may form black holes with countless protruding edges connecting otherwise unrelated sub-graphs. Such a black hole could be a data artefact consisting of a legal form only instead of a proper company name connecting all firms with the same legal form in a city. The maximum Identity would be high while the minimum Identity would be close to zero. Another type of artefacts are network thickets consisting of inconspicuous nodes and connections extending over vast uninterrupted formations. If those aberrations stay unattended, the clustering will produce super-clusters of unrelated entities. Black holes can be fended off by preemptively applying Feedback activation (see 2.2.5) for suspiciously inflated candidate lists. We call the remedy for thickets Nested Cascaded Traversal (Doherr, 2017, 2021), a method imposing escalating rules on the connection attributes in accordance with the intermediate cluster size. Every time during traversal, if the already collected number of nodes for a given start node exceeds a threshold, the traversal is reset to that start node and a logical expression based on the quality attributes is evaluated to determine the eligibility of an edge. If the expression evaluates to false, the edge is excluded from traversal. A cascade is defined by a sequence of rules taking effect if the threshold is exceeded under the regime of the previous rule. Consequently, the rules should become more restrictive over the course of a cascade. Nesting takes place when a first cascade, usually initiated by a rule activated from the beginning of the traversal with a threshold of zero, is pre-clustering the network into hyper-nodes with a high confidence in their unity. A second cascade set is based on those hyper-nodes. The quality attributes of the edges in this

hyper-network are derived from the aggregated edges of the respectively encased original nodes choosing the maximum within each attribute. Nesting allows to partition the decision process of the otherwise uncomfortably arbitrary rule definition. The first cascade defines what the user considers semi-transitive connections. In general, these are connections with a high minimum identity min and/or a high Score percentile p. The SearchEngine allows to export the resulting clusters in an easy to browse way, to verify and fine-tune the rule set for the nesting. If satisfied, a second rule set can be added to define the number of intransitive transitions between the internally transitive hyper-nodes. In reality, an intransitive transition can be a joint venture or a merger between companies incorporating parts of the original names. Of course, these occasions are scarce and the threshold should reflect this. The high variation in the length of firm names is another reason to apply nesting. Long firm names have a much higher potential to create variations than short firm names due to more opportunities for divergent word positioning or misspellings. This variation cannot be captured by a uniform threshold for plausible cluster sizes. Nesting harmonizes long and short firm names into hyper-nodes and relegates the decision regarding cluster sizes to the number of plausible major transitions.

The following example of a Nested Cascade is based on a self-referential search with a threshold of 90% and subsequent mirroring of matched tuples:

$$min \geq 90 \; or \; min \geq 70 \; and \; p \geq 75 \; @ \; 0, min \geq 90 \; @ \; 21; min \geq 90 \; @ \; 4$$

The first cascade up to the semicolon defines the conditions for a valid edge without a threshold. The number following the "@" symbol defines this threshold. A zero designates a rule that takes immediate effect. The second rule, separated by a comma, is activated when a resulting hyper-node exceeds the threshold of 20 nodes. This fallback rule drops the relaxed requirement for the minimum Identity for the top quantile in the IP distribution in case of implausible cluster sizes. The second cascade, after the semicolon, is activated when more than three hyper-nodes are connected. The first cascade defines the maximum quality of all remaining edges connecting hyper-nodes for the second cascade as all stronger connections are now encapsulated within the hyper-nodes. Further adjustments can be made to this definition as the user sees fit. It is important to mention, that the cascade framework is highly reliant on the context of the match. The thresholds are purely based on the assessment for plausible cluster sizes of the user and iterative probing of different settings. Nesting is less applicable in the case of person names, where the variance in length and word positions is much lower. Also, the further application of the clustered data has an impact on the justifiable leeway of the cascade definitions. If the data is used for a more general descriptive analysis or if external restrictions minimize the impact of inflated clusters, like the identification of self-citations, the cascades can be more audacious. If the intent is the provision of focused data, one should muster a stricter mindset in regard of the ruleset.

The definition of those cascading rules seems arbitrary at first, but they represent specific restrictions imposed by the search strategy in conjunction with the threshold. It seems tempting to apply clustering methods from the field of network and graph theory because they do not require such profound user intervention and are well suited to cluster social networks based on intrinsic motivations. Unfortunately, intransitive similarity networks do not have those intrinsic properties in the nodes, the edges nor within aggregated vicinities that would explain the plausibility of an intransitive transition besides the arbitrary assessment of the user regarding the overall data quality and the tolerance toward false classification according to the mission target. The burden of the decisions can be eased by overlapping the network with networks derived from other sources. In case of company data, including mutual ownership data can improve the clustering immensely up to a point where the self-referential search becomes unnecessary. To obtain focused company information for unfocused company data, it is required to match unique company keys already acting as cluster IDs. A conventional search between an entity-complete focused dataset and an unfocused dataset is preferable to a self-referential search because it avoids the necessity of cascaded traversal and combines two previously separated information spaces.

Nested Cascaded Traversal exploits the tendency of weak matches to form hedges caused by a higher propensity of false positives. If there would be a way to eliminate false positive links from the network, we could renounce rule-based traversal. In the following chapter, we will discuss a method to identify false positives within the candidates.

2.4 Identification of False Positives with ML

The SearchEngine heuristic is specialized in retrieving candidates according to the relative composition of search terms. The absolute identification potentials of words comprising the search term is transformed into relative potentials. This warrants unbiased retrieval even for weak search terms. The Score, representing the absolute IP of a search term (see 2.2.2), is already an important indicator for the probability of false positives and is therefore reported in the Extended Export format intended to manually filter false positives. Of course, this manual control is not applicable for large scale matches. Extensive self-experimentation in this field has shown that the assessment of the validity of a match during manual checking depends to a large extent on the perceived absolute Identification Potential. To reflect this intuition in a codified form, the Registry provides all necessary information in the form of frequencies. A candidate can be represented by the frequency of every word according to its Search Type. A similar vector can be obtained for its search term. Words that are exclusive to the data realm of the search records require the creation of a separate Registry for that data source implementing the Search Types. This auxiliary Registry and the original Registry are used to define the absolute IP of a

word. To obtain a healthy and consistent number range, we use logarithmic distributions of the word frequencies to normalize the absolute Identification Potentials aIP:

$$aIP(w) = 1 - \min\begin{pmatrix} ln\left(occ_R(w, st(w))\right)/ln\left(maxocc_R(st(w))\right) \mid w \in R \\ ln\left(occ_A(w, st(w))\right)/ln\left(maxocc_A(st(w))\right) \mid w \in A \end{pmatrix} \quad (9)$$

If a word can be found in the original Registry R, an intermediate, normalized logarithmic IP will be calculated based on the frequency $occ_R(w, st(w))$ and the maximum frequency $maxocc_R(st(w))$ encountered for the Search Type of this word. If the word can be found in the auxiliary candidate Registry A, a similar intermediate IP will be derived using the corresponding functions. The minimum of both values constitutes the absolute Identification Potential of the word. An aIP close to zero has a low absolute Identification Potential, while an aIP of 1 represents a unique word. By accounting for both data realms, the aIP is independent of the search direction. The information representing a match tuple consists of a given number of aIP for every Search Type: one set for the mutual words that caused the match (positive), one set for the words exclusive to the found candidate (neutral) and a set for the words exclusive to the search term (negative). The potentials are reported in descending order until a specified number of entries for every set of a Search Type is reached. For example: 5 entries for every set of the Search Type firm name, 3 entries for the street, 2 for the city, 1 for the postal code and 15 for the 3-gram representation of the firm name resulting in 78 elements. The count per Search Type should reflect the number of the most relevant words required to identify an entity by the respective Search Type on average. Because the potentials are reported in order of relevancy, expanding the vector size beyond that provides only diminishing returns. The intention behind the juxtaposition of the 3 sets for every Search Type, one supporting the claim of a true positive, one neutral and one indicating a potential false positive, is to mimic the intuitive assessment of the examiner (usually a student), who has labeled the training data. These elements are the first components of the Meta Vector, a collection of derived data points providing variation for every match. It will be used to train Neural Networks or other ML applications based on labeled training data to transfer the intuitive assessment on the complete match.

Another decisive factor influencing the decisions of the examiner is the repetition of words among different Search Fields. If a company name encompasses the name of the city the company is situated, the absolute Identification Potential of this word for Search Types related to the firm name is demeaned because of the redundancy of this information. If the city is a small town, the rIP is potentially high, which increases the risk of retrieving false positive candidates that share the same city name as ostensibly distinctive feature of the firm name. To reflect this detrimental potential of repetition, the Meta Vector is enriched with the number of distinct words of the search term $scnt$, the

count of words that repeat at least once in more than one Search Field $rcnt$ and the maximum of the aIP of words fulfilling the latter condition for every Search Field $r_1 \dots r_n$ (with n being the number of Search Fields).

Furthermore, the Meta Vector implements a string comparison based on the LRCPD routine described chapter 2.2.3 for every Search Field. These values capture visual similarities, which are not accounted for by the heuristic, for instance, if a candidate was already picked due to a word with a dominating rIP, while the other words of the search term were distorted by misspellings and therefore not represented in the original Registry. Furthermore, the comparisons should support the significance of destructive Search Types in the Meta Vector. As the LRCPD comparison function is not commutative, the two comparison directions are reported per Search Field, i.e. csf_1 for comparing the search term field 1 with the respective candidate field or cfs_3 for comparing the 3rd candidate field with the corresponding search term field.

The Meta Vector is complemented by the record IDs of the search record and the candidate record, the Identity and the Score of the match, one-hot dummy vectors marking the number of the incremental search run responsible for retrieving the candidate, the overall number of candidates retrieved for this search term, the number of distinct Identities of those candidates and the relative position of the candidate Identity among those distinct Identities. As the search run dummies represent the overall fixed-effects of the different steps of the search strategy pertaining weight distribution, smoothing, feedback and other search specific settings otherwise not represented in the meta vector, it seems prudent to aggregate similar steps only differentiated by the threshold to avoid unnecessary over-specification (see Chapter 3).

External information can be used to complement the Meta Vector, for instance, the geographical distance to the candidate derived from geolocation efforts on the data to represent tacit knowledge of the examiner about the spatial closeness of addresses. If two focused company data sources are merged, further augmentations of the Meta Vector become available in the shape of similarity measures of firm parameters additionally disclosed in the Extended Export sample. The sample is drawn by a fixed number or a percentage of search records. All candidates associated with a drawn search record are extracted for the training data to preserve intrinsic relations among the candidates. To capture the degree of homogeneity within such a local block of data, the standard deviation of the LRCPD comparisons (see above) can be reported. The following table lists all components of the Meta Vector:

Variable	Description
searched	record number of the search term in the search table
found	record number of the candidate in the base table
equal	1 = match (true positive), 0 = no match (false positive)
identity	Identity (either sum of rIP or LRCPD score or a mix)
score	absolute IP
cnt	number of candidates, block size
icnt	number of distinct Identities within a block
ipos	relative position of the candidate Identity as a percentile rank
run1	dummy vector designating candidates retrieved during the initial search run
run2…	dummy vector for candidates retrieved during the second run, if applicable
…run*n*	dummy vector for candidates retrieved during the n^{th} run, if applicable
csf1…	LRCPD comparison (searched → found) of the first search field
…csf*n*	LRCPD comparison (searched → found) of the n^{th} search field
cfs1…	LRCPD comparison (found → searched) of the first search field
…cfs*n*	LRCPD comparison (found → searched) of the n^{th} search field
scnt	number of distinct words in the search term
rcnt	number of distinct words found in multiple fields of the search term
r1…	maximum *aIP* of the words in the first search field repeating in other search fields
…r*n*	maximum *aIP* of the words in the n^{th} search field repeating in other search fields
m1_1…	highest *aIP* of matching words for search type 1
…m1_*n*…	n^{th} highest *aIP* of matching words for search type 1, repeat over search types
f*1_1*	highest *aIP* of surplus words of the candidate for search type 1
…f*1_n*…	n^{th} highest *aIP* of surplus words for search type 1, repeat over search types
s*1_1*	highest *aIP* of words not found in the candidate for search type 1
…s*1_n*…	n^{th} highest *aIP* of not found words for search type 1, repeat over search types
csf1sd…	standard deviation of *csf1* within *searched*, optional, external, repeat over csf fields
cfs1sd…	standard deviation of *cfs1* within *searched*, optional, external, repeat over cfs fields

Table 2: Meta vector

The major advantage of this kind of individual blocking versus exclusion blocking methods is the implicit similarity between the search term and the candidates forming such a block. The information needed to separate true from false positives has already a strong foundation in this similarity while exclusion blocking relies on rules to fragment the solution space to facilitate methods based on pairwise comparisons, which would not be feasible for the full data. As each entry in such a group has the same legitimation to be a member, similarity among members can only be ascertained at latter stages usually based on scoring mechanisms to identify the true positive pairings among a vast majority of false positive linkages. The quadratic nature[1] of the complexity puts a high load on computational

[1] If the number of records in a block is doubled on both sides, the number of pairwise comparisons is quadrupled.

resources while amplifying the disproportion of true negatives. Individual blocking directly provides similarity by the association of search term and candidates with a strongly subdued increment of the computational effort due to the optimized search heuristic. With the similarity of pairings already given, subsequent steps only have to filter false positives instead of additionally establishing similarity.

The Meta Vector incorporates enough variation correlated with the decision process of the examiner labeling the training data to create a digital copycat by means of ML that will scrutinize the remaining matches. Even though ML approaches are known for relatively strong pattern recognition capacities, we suggest to repeat the process of supervised training for every major matching endeavor to capture the idiosyncrasies of the task at hand in regard of the involved data, the overarching context and the associated labeling regime.

3 Walkthroughs

3.1 EPO Applicants vs. German Company Panel

We match the German patenting firms in the EPO with the German company data of the Mannheim Company Panel, a superset of the German Orbis data. The EPO patent applicants are exported from the publicly available Patstat database, Spring Edition 2021. They are already aggregated on firm name and address. We create a tab-delimited text file containing 62,036 German companies. We excluded 36,533 individual inventors associated with less than 10 patents from the applicants because those are better suited to be searched in the ownership database. The separation was relatively simple because person names always contain a comma separating first from last name and do not brandish a legal form. The company database consists of 23,539,928 aggregated historic firm address variants representing 15,186,675 firms accrued from 1993 till 2021. Since patent applicant data is not well maintained, particularly in the case of lapsed patents, we have to consider that many applicants appear with different names and addresses in the data reflecting the company history.

Given the pronounced size difference between the datasets and the clear focus of the Mannheim Company Panel on maintaining consistency among the firms compared to the unorganized applicant data, we decided that the Company Panel should become the base table. An applicant as a search term only needs to be matched to the best candidate as the second best is redundant within a focused dataset. If we would have chosen the unfocused applicant table, every firm has to match all potential applicant variants implying a low threshold and a high susceptibility for false positives. With the focused database providing the candidates, we can apply a gradually escalating search strategy, starting with high constrains filling the gaps by loosening the threshold and shifting the weights among the Search Types in further search runs.

We issue the following command to create the Registry:

```
create("Firma FIRMA, Firma FIRMA GRAM3, Strasse STRASSE, Plz, Ort")
```

Every parameter of this command consists of a field name of the base table optionally followed by a list of capitalized Preparers. FIRMA and STRASSE are specialized Preparers handling idiosyncrasies of German legal forms and the haphazard concatenation of street names with street types, i.e. “Berliner Straße” and “Berlinerstraße” are both valid representations of the same street. The German language poses a challenge for word-based algorithms because it features almost limitless concatenation of words. Specialized Preparers or equivalent data preparation is not required for languages without that feature, e.g. English. We introduce a second Search Type for the company name “Firma” based on 3-grams to enable misspelling-proof search steps. As we intent to set our focus on the firm name, it is not necessary to apply the same for the other fields. After the Registry was established we utilized the following script[2] to implement our search strategy:

```
types("Firma 70, Firma 0, Strasse 10, Plz 10, Ort 10")
depth(999999)
threshold(79)
cutoff(10)
feedback(10)
activation(10)
relative(.f.)
darwinian(.t.)
ignorant(.f.)
zealous(.f.)
search()
types("Firma 0, Firma 70, Strasse 10, Plz 10, Ort 10")
search(1, 1)
types("Firma 0, Firma 70 log, Strasse 10, Plz 10, Ort 10")
search(1, 1)
types("Firma 70 -9000000, Firma 0, Strasse 10, Plz 10, Ort 10")
search(1)
types("Firma 70, Firma 0, Strasse 10, Plz 10, Ort 10")
threshold(65)
search(1)
types("Firma 70 log, Firma 0, Strasse 10, Plz 10, Ort 10")
search(1)
types("Firma 0, Firma 40, Strasse 20, Plz 20, Ort 20")
threshold(80)
search(1, 1)
types("Firma 0, Firma 40 log, Strasse 20, Plz 20, Ort 20")
search(1, 1)
types("Firma 0, Firma 40, Strasse 20, Plz 20, Ort 20")
zealous(.t.)
search(1, 1)
```

The search strategy consists of 9 consecutive search runs. Each search run employs a Feedback of 10% which will be activated if 10 or more candidates are found. This number corresponds with the Cutoff limit to ensure that large candidate lists are curtailed to reduce redundancy (see 2.2.5). Feedback enabled in conjunction with a Cutoff and an Activation threshold only creates the necessary variation

[2] The SearchEngine also supports an interactive GUI mode. All interactive activities are logged as scripts for later reference. An extensive manual is available in the GitHub package (see Software section).

for the Cutoff. It will not be reported nor has it any impact on the Threshold, which is 79% for the first 4 runs. All runs implement the Darwinian approach of only keeping the best candidates. The Darwinian culling of the candidates occurs before the Cutoff. The first run implements the basic heuristic of the SearchEngine by assigning the main weight of 70% on the company name (Firma). The remaining 30% are evenly distributed over the address fields (Strasse, Plz, Ort = Street, Postcode, City). The Threshold of 79% implicitly requires that at least one of those fields matches to 90% while maintaining a perfect match for the firm Search Type. Of course, more matching address fields automatically create more leeway for the similarity of the firm names. As our search strategy slowly reduces the requirements for candidates from run to run, there is no need to merge results assuming higher quality in concluded runs. We select the option to exclude applicants with already associated candidates from future matches (defined by the parameters of the search function call). The second run implements a search tolerant to misspellings by shifting the 70% weight to the Search Type implementing a 3-gram Preparer on the firm name. This run is complemented with a log smoothed distribution for this Search Type. The fourth run repeats the setting of the first run but with a simple word based heuristic by applying negative offset smoothing beyond the highest word count for this Search Type. This run is intended to handle longer firm names with rare deviating words not contributing to the identification of a firm, like exotic activity descriptions ("Tresorhandel" vs. "Eisenwaren" = "vault retailer" vs. "hardware"), which is exacerbated by the tendency to concatenate words in the German language. The 5th run re-implements the first run with a lower Threshold of 65% disregarding similarity in the address. The 6th run uses a log smoothed distribution for the firm Search Type. For all applicants still without candidates the following two runs shift the focus to the address by distributing 60% over the address fields and only 40% on the 3-gram Search Type of the firm field. With a threshold of 80% we try to find cases with major misspellings. The 9th and final run repeats this attempt with a dynamic Threshold initiated with the option "zealous", which adapts the Threshold to the respective best candidate Identity scouring the data for the last stragglers. The following table shows the yield of the different search steps:

run	description	threshold	applicants	candidates
1	firm 70, street 10, zip 10, city 10	79	56,603	108,775
2	firm 3-gram 70, street 10, zip 10, city 10	79	682	977
3	firm 3-gram 70 log, street 10, zip 10, city 10	79	275	385
4	firm 70 -9000000, street 10, zip 10, city 10	79	167	358
5	firm 70, street 10, zip 10, city 10	65	3,078	8,251
6	firm 70 log , street 10, zip 10, city 10	65	69	134
7	firm 3-gram 40, street 20, zip 20, city 20	80	17	20
8	firm 3-gram 40 log, street 20, zip 20, city 20	80	34	53
9	firm 3-gram 40, street 20, zip 20, city 20	zealous	35	61
			60,960	119,014

Table 3: Search strategy for EPO applicants vs. company panel

We managed to match 60,960 of 62,036 applicants with this search strategy. We draw a stratified sample of 3,264 matched applicants paired with 6,535 candidates over-representing the runs 2 & 3 (281) and 5 (620). This sample is supplemented with the completely validated matches of run 4 and 6 to 9 due to the marginal representation in the matching population. We aggregated the run dummies to encompass runs 2 to 4 and 6 to 9 respectively while keeping the dummies for run 1 and 5. This is an arbitrary decision to mitigate the underrepresentation of special search runs in the training data. There is no evidence on the marginal impact of the run dummies on the final ML prediction, but it seems prudent to support the training by aggregating those fixed-effects of the search strategy to avoid over-specification. The training data is manually scrutinized using the extended export format revealing that our search strategy and data preparation successfully avoided false positives. To improve the result further, we apply the STATA script seml_train.do, which is part of the SearchEngine package on GitHub, on the training data enriched with the before mentioned meta information (see 2.4). This script uses the STATA module brain implementing a Neural Network (NN) toolkit, which can be downloaded from GitHub or directly from the Statistical Software Components (SSC) archive (see Software section). The program automatically retains 10% of the sample, skipping the completely validated matches, to test out-of-sample performance. Training is a completely automatized trial process of different NN setups to find the one with the highest accuracy within the retained sample to mitigate overfitting. The script picked a Neural Network with one hidden layer of 25 neurons out of the following hidden layer combinations: [25], [50], [100], [25x25], [50x50], [100x100]. Each combination includes 2 setups: a basic one and a weighted one balancing the distribution between matches and non-matches. Additionally, we juxtapose the NN results with the output from a Probit regression to have a baseline for its performance:

Probit	True	False
Positive	575	13
Negative	70	19
Recall	96.80%	84.34%
Precision	97.79%	78.65%
Accuracy	95.27%	

NN[25]	True	False
Positive	590	5
Negative	78	4
Recall	99.33%	93.98%
Precision	99.16%	95.12%
Accuracy	98.67%	

Table 4: Confusion matrix comparison of Probit vs. NN with 25 hidden neurons (non-weighted)

Finally, the script seml_think.do applies the best NN on the meta data to convey its specific expertise to identify false positives on the whole data set. The prediction is augmented with the validated sample. Of 119,014 links between applicants and company address variants 4,836 were identified as false positive. After joining the entity keys person_id for applicants and crefo for the companies we are able to aggregate the data to 59,630 applicants matched to 88,176 companies. Further steps are

required to resolve multiple assignments mostly stemming from duplicates in the company data and ambiguities caused by subsidiaries. We defined a priority rank based on data quality, size and industry codes to disambiguate the assignment. We do not drop the lower ranked assignments to increase the probability of matches with external company data products using the same identifier, namely Orbis. The whole exercise allows us to enrich 621,781 of 629,246 German firm owned EPO patents, excluding 42,206 person owned patents, with company data. Although, we have used specific software packages to implement a machine learning approach, it does not rely on them. All necessary data is available as simple tab-delimited text files to be processed with any software tool of choice by either replicating our approach or implementing other machine learning methods.

3.2 Establishment Data vs. German Company Panel

This linkage exercise is quite similar to the previous matching effort in respect of the base table, which is again the German company panel. The establishment dataset is a part of the databases maintained by the German Federal Employment Agency supervising the data of all employees partaking the mandatory social insurance system. The term establishment in this context refers to a working location as part of an employee report. These establishments have a large overlap with legal firms, but can also describe dependent subsidiaries, branches or departments. Even though the database can be considered focused on the context "working place", this fuzziness and the rather subordinate classification compared to firms favors the company panel as the source of candidates. Search strategies based on Darwinian exclusion and gradual relaxation of candidate restrictions are much less cumbersome to implement than strategies requiring low thresholds and non-exclusive overlapping of search runs forced by unfocused or, like in this case, misaligned contexts.

The establishment data is available as panel from 2000 to 2018 and aggregated to 21,156,478 name and address variants of 14,283,323 establishments. Systematic harmonization of street addresses was required to prevent an inflation of variants[3]. We used the same SearchEngine setup as for the linkage with the EPO applicants (see 3.1).

We implemented a slightly more conservative approach to the search strategy compared to the EPO linkage due to the considerable larger amount of records involved. First excerpts have shown that the overlap between establishments and companies is much lower than for the patent applicants and therefore the ratio of false positives much higher, further deterring audacious strategies. The implementation is as follows:

[3] The house numbers have been systematically embellished separating different components with blanks, i.e. letter attachments or hyphens instead of slashes defining ranges. We have changed older street addresses to comply with the new standard to avoid this systematic disruption.

run	description	threshold	establishments	candidates
1	firm 70, street 10, zip 10, city 10	85	9,393,950	14,531,161
2	firm 3-gram 70 log, street 10, zip 10, city 10	85	294,805	356,327
3	firm 3-gram 70, street 10, zip 10, city 10	85	32,075	39,234
4	firm 70 -9000000, street 10, zip 10, city 10	85	841,062	1,296,231
5	firm 70, street 10, zip 10, city 10	65	6,964,399	43,400,697
6	firm 3-gram 70 log, street 10, zip 10, city 10	65	571,931	801,739
7	firm 3-gram 70, street 10, zip 10, city 10	65	141,064	202,351
			18,239,286	60,627,740

Table 5: Search strategy for establishments vs. company panel

We start with a high threshold imposing a significant overlap of the address followed with two dedicated search runs to capture misspellings by shifting the weight from the basic firm search type to the one implementing a 3-gram Preparer. Log smoothing is toggled on and off to ensure that misspellings do no dominate the rIP distribution of the 3-grams preventing matches but also allowing for idiosyncratic features within the search term. For an interlude, we switch to a word-based heuristic by applying a negative offset smothering any variance in the frequency distribution of the firm search type. The first four steps retrieve high quality linkages due to the high threshold. The nature of the relation between establishments and firms require to accept matches at different locations. Hence, we repeat runs 1 to 3 but with a lower threshold not requiring similarity in the address. By staging the threshold shift in that manner, we can prioritize misspellings at a similar location to matches at different locations because establishments that already have candidates are omitted from further search runs. The number of 18 million establishments with 61 million candidates is beyond any feasibility of manually labeling false positives.

The training data consists of a random sample of 2,000 establishments paired with 5,523 candidates. To reduce redundancy in the specification of the search runs in the meta data, we aggregate the run dummies into three overarching search steps: the classical frequency-based heuristic [1, 5], the word-based heuristic [4] and misspellings [2, 3, 6, 7]. The training script automatically retains 10% (569) pairings for out-of-sample prediction. After iterating through a set of predefined NN settings, it picked a Neural Network with two hidden layers of 25 neurons each. The training used weights to balance the matching distribution. The Probit regression, already indicating a high predictive power of the meta data, is outperformed by this NN:

Probit	True	False
Positive	180	10
Negative	367	12
Recall	93.75%	97.35%
Precision	94.74%	96.83%
Accuracy	96.13%	

NN_w[25x25]	True	False
Positive	184	3
Negative	374	8
Recall	95.83%	99.20%
Precision	98.40%	97.91%
Accuracy	98.07%	

Table 6: Confusion matrix of Probit vs. NN with 25x25 hidden neurons (weighted)

After aggregating variants of the matching data on the actual entities, 8,273,283 of 14,283,322 establishments (57.92%) have been matched to 7,281,447 company keys. On average, each establishment is linked to 4.35 companies (2.96 excluding the top percentile). This ambiguity is caused by redundancy (doublets) and subsidiaries in the company data. Table 7 shows the industry distribution of the two databases for the full and matched populations. We excluded self-employment based businesses from the statistics due to the lopsided coverage in the establishment data while the company data is almost devoid of those firms. The company panel is constructed from semiannual copies of a cross-sectional database of a German credit rating agency[4] since 1992. Due to technical limitations, the unbalanced panel is only complete since the year 2000. Before, the data incorporates only startups and companies of the new federal states (former territory of the German Democratic Republic). Even though the provided establishment data only reaches back to 2000, we used the complete company panel (1992-2021) for the linkage to take into account the ambiguity of the timing of exits in the company data. To keep the populations comparable in Table 7, we have removed companies that do not exert any economic activities after 1998 or are flagged by the data provider as duplicates as long as they are not required to establish a link. We will not delve into the differences between the two populations before and after the match considering the prevalence of missing industry classifications and the potentially incongruous units, but the overall picture does not indicate any unsettling distortions caused by the matching approach.

To further improve the linkage quality, we defined a sorting mechanism ranking the companies linked to an establishment. As establishments show a high tendency for spatial closeness, we bolster the rank with distance calculations between establishments and companies based on center coordinates of the respective postal code regions. The highest priority lies on companies within a distance of 25km operating in the same industry as the establishment without any negative internal quality indicators from the data provider (creditreform). Among those the largest company in terms of employees, discounted by 25% per year to favor more recent specifications, is ranked higher. Further sort items are the separate quality components of the top priority category like quality indicators, industry

[4] The agency is called “creditreform”. It also provides the majority of the German data for the Bureau van Dyke products like ORBIS or MARKUS.

congruence, distance and last update year. We have to disclose that the data quality of the company data is less than ideal enforcing such a multilayered approach to accommodate missing values. Of course, this is an arbitrary approach to handle the ambiguity caused by the data quality but also by the fuzziness of the relation between establishments and legal companies. Specific research question may require other solutions to this entity resolution issue. Discussing those is beyond the scope of this paper.

Industry (nace2)	Establishments					Companies				
	N	%	Matched	%	Match %	N	%	Matched	%	Match %
food (10-12)	105,369	1.08	78,925	1.17	74.90	77,890	0.99	69,824	1.20	89.64
textile (13-15)	37,481	0.38	24,808	0.37	66.19	31,877	0.40	24,322	0.42	76.30
wood/paper (16,17)	40,327	0.41	32,356	0.48	80.23	40,368	0.51	33,685	0.58	83.44
chemistry (20)	11,091	0.11	8,821	0.13	79.53	10,260	0.13	8,089	0.14	78.84
pharma (21)	2,637	0.03	2,026	0.03	76.83	2,703	0.03	2,189	0.04	80.98
rubber (22)	22,808	0.23	18,253	0.27	80.03	16,770	0.21	14,700	0.25	87.66
glas (23)	29,533	0.30	22,821	0.34	77.27	23,860	0.30	20,273	0.35	84.97
metall (24,25)	143,306	1.46	114,936	1.70	80.20	106,939	1.36	92,604	1.60	86.60
electro (26,27)	46,907	0.48	36,625	0.54	78.08	37,980	0.48	30,947	0.53	81.48
machine (28)	50,812	0.52	42,428	0.63	83.50	42,672	0.54	36,976	0.64	86.65
mobile (29,30)	14,571	0.15	11,631	0.17	79.82	12,653	0.16	10,292	0.18	81.34
furniture (31,32)	79,098	0.81	64,821	0.96	81.95	54,607	0.69	44,932	0.78	82.28
repair (33)	28,678	0.29	24,141	0.36	84.18	22,150	0.28	16,876	0.29	76.19
energy/mining (5-9,19,35)	30,517	0.31	22,671	0.33	74.29	54,018	0.69	26,172	0.45	48.45
water/recycling (36-39)	30,023	0.31	22,588	0.33	75.24	23,381	0.30	19,854	0.34	84.92
wholesale (46)	492,709	5.04	360,197	5.32	73.11	503,738	6.39	374,321	6.46	74.31
logistics (49-53,79)	489,453	5.00	357,529	5.28	73.05	362,733	4.60	301,635	5.20	83.16
media (58-60)	58,383	0.60	43,552	0.64	74.60	58,295	0.74	40,082	0.69	68.76
ict (61-63)	240,296	2.46	165,696	2.45	68.95	203,266	2.58	130,695	2.25	64.30
finance (64-66)	260,231	2.66	175,461	2.59	67.43	338,901	4.30	184,846	3.19	54.54
rnd/engineering (71-72)	274,763	2.81	198,308	2.93	72.17	233,626	2.96	177,741	3.07	76.08
consulting (69,70,73)	547,846	5.60	397,250	5.87	72.51	623,331	7.91	390,281	6.73	62.61
service (74,78,80-82)	533,573	5.45	357,639	5.28	67.03	545,982	6.93	359,279	6.20	65.80
construction (41-43)	1,033,643	10.56	828,949	12.25	80.20	916,079	11.62	747,458	12.89	81.59
retail (47)	1,280,288	13.09	875,220	12.93	68.36	1,001,701	12.71	745,849	12.87	74.46
gastronomy/hotel (55,56)	1,076,911	11.01	689,417	10.19	64.02	574,021	7.28	497,360	8.58	86.64
real estate (68)	649,797	6.64	371,028	5.48	57.10	359,954	4.57	225,869	3.90	62.75
leasing (77)	75,903	0.78	54,546	0.81	71.86	58,023	0.74	40,749	0.70	70.23

continued Industry (nace2)	Establishments					Companies				
	N	%	Matched	%	Match %	N	%	Match	%	Match %
public service (84,85)	250,386	2.56	140,149	2.07	55.97	110,419	1.40	82,393	1.42	74.62
medical/social (75,86-88)	591,382	6.04	384,851	5.69	65.08	330,565	4.19	279,419	4.82	84.53
culture/gambling (90-94)	347,044	3.55	209,343	3.09	60.32	298,137	3.78	169,257	2.92	56.77
agriculture/fishing (1-3)	267,910	2.74	181,566	2.68	67.77	170,802	2.17	136,631	2.36	79.99
car sale (45)	250,636	2.56	201,083	2.97	80.23	241,932	3.07	181,624	3.13	75.07
print (18)	39,388	0.40	32,230	0.48	81.83	37,410	0.47	30,511	0.53	81.56
self-employed (97-99)	1,333,829		391,626		29.36	4,599		3,234		70.32
unknown (0)	3,165,804		1,113,129		35.16	966,807		504,455		52.18
Population	14,283,322		8,273,283		57.92	8,853,109		6,305,046		71.22
Pop. without (0,97-99)	9,783,689	100.00	6,768,528	100.00	69.18	7,881,703	100.00	5,797,357	100.00	73.55

Table 7: Full populations vs. matched populations
Missing industry classifications and self-employment based businesses are excluded from the distribution shares due to systematic deviations in the coverage between the establishment and company databases.

4 Conclusion

The SearchEngine avoids the pitfalls of the classical approach of blocking the data along arbitrary exclusion restrictions to reduce the solution space to facilitate pairwise comparisons. Even though those approaches prove viable, the blocking is by definition rather driven by computational feasibility than efficiency. The SearchEngine retrieves candidates for search terms akin individually directed blocking. Because the computational load increases not in a quadratic fashion, multiple search runs can be formed into a search strategy that concentrates on efficient candidate retrieval instead on computational limitations. The holistic approach is rooted in the inherent similarity within those individual blocks. Classic blocking methods have to incorporate the identification of potential matches by excluding true negatives and the separation of true from false positives into the comparison steps and the post-processing. Machine learning may improve the handling of multiple quality measurements and rankings but is hampered by the ambiguity of the task of drawing a training sample when the overwhelming majority of pairings are true negatives. The SearchEngine only requires the identification of false positives as similarity is already established. For this purpose, it provides lean meta data with high predictive power, which is not prone to over-specification by having too many features for too little data. Drawing a training sample is trivial as the solution space is clearly structured into search terms and associated candidates. Even though the determination of the general search direction is far from being an innocent decision, this is a low price to pay for having a streamlined framework for matching.

Appendix

Searched	Found	Identity	Equal	Name	Street	Zip	City	Score	Cnt	Run
5935			1	Isotopen Technologien München AG	Rathausplatz 5	83435	Bad Reichenhall	2.65	5	
5935	684067	70.00		ITM Isotopen Technologien München AG	Theatinerstr. 23	80333	München	2.65	5	5
5935	19425145	70.00		ITM Isotopen Technologien München AG	Walter-Meissner-Str. 2	85748	Garching	2.65	5	5
5935	19425146	70.00		ITM Isotopen Technologien München AG	Lichtenbergstr. 1	85748	Garching	2.65	5	5
5935	21419919	70.00		ITM Isotopen Technologien München AG	Schleißheimer Str. 91a	85748	Garching	2.65	5	5
5935	21419920	70.00		ITM Isotopen Technologien München AG	Walther-von-Dyck-Str. 4	85748	Garching	2.65	5	5
3885			9	Universität Stuttgart	Keplerstrasse 7	70174	Stuttgart	0.04	4	
3885	16183176	100.00	1	Universität Stuttgart	Keplerstr. 7	70174	Stuttgart	0.04	4	1
3885	17625066	100.00		Akademische Motorsportgruppe an der Universität Stuttgart	Keplerstr. 7	70174	Stuttgart	0.04	4	1
3885	17706003	100.00		Vereinigung von Freunden der Universität Stuttgart	Keplerstr. 7	70174	Stuttgart	0.04	4	1
3885	17706007	100.00		Förderkreis Betriebswirtschaft an der Universität Stuttgart e.V.	Keplerstr. 7	70174	Stuttgart	0.04	4	1
40529			9	Klaus Sindel Rusi-Kosmetik-Pinsel-Brushes GmbH	Ansbacher Strasse 53	91572	Bechhofen	0.05	4	
40529	19723005	85.69		Hauck Pinsel GmbH	Ansbacher Str. 47a	91572	Bechhofen	0.05	4	9
40529	19700672	83.76		Johann Führ & Söhne Pinselfabrik GmbH	Ansbacher Str. 27-29	91572	Bechhofen	0.05	4	9
40529	19707191	82.98		Elco-Pinsel GmbH	Ansbacher Str. 86	91572	Bechhofen	0.05	4	9
40529	19773153	81.77		Ernst Bock & Sohn Pinselfabrik GmbH	Ansbacher Str. 68	91572	Bechhofen	0.05	4	9
25276			1	RHODIA AG	Engesserstrasse 8 Postfach 1320	7800	Freiburg	2.41	4	
25276	4371760	80.32		RHONE-POULENC RHODIA AG	Engesserstr. 8	79108	Freiburg	2.41	4	1
25276	5408910	80.32		RP Rhodia AG	Engesserstrasse 8		Freiburg	2.41	4	1
25276	5408911	80.32		RP Rhodia AG	Engesserstr. 8	79108	Freiburg	2.41	4	1
25276	15531543	80.32		Rhodia Acetow AG	Engesserstr. 8	79108	Freiburg	2.41	4	1
42597			9	Airbus Operations GmbH	Kreetslag 10	2129	Hamburg	5.64	4	
42597	2625985	90.00	1	Airbus Operations GmbH	Kreetslag 10	21129	Hamburg	5.64	4	1
42597	3238943	90.00		SATYS SEALING & PAINTING Germany GmbH c/o Airbus Operations	Kreetslag 10	21129	Hamburg	5.64	4	1
42597	3273434	90.00		MAAS Aviation GmbH c/o Airbus Operations	Kreetslag 10Geb.225	21129	Hamburg	5.64	4	1
42597	3394060	90.00		Airbus Operations GmbH Dr. Helena Auber	Kreetslag 10	21129	Hamburg	5.64	4	1
52085			1	Karlsruher Institut für Technologie	Körpersch. des öffentl. Rechts,Kaiserstrasse 12	76131	Karlsruhe	0.15	3	
52085	16156922	90.02	9	Akademische Fliegergruppe am Karlsruher Institut für Technologie	Kaiserstr. 12	76131	Karlsruhe	0.15	3	1
52085	16156923	90.02	9	Akademische Fliegergruppe am Karlsruher Institut für Technologie e.V.	Kaiserstr. 12	76131	Karlsruhe	0.15	3	1
52085	16240255	90.02		KIT Karlsruher Institut für Technologie	Kaiserstr. 12	76131	Karlsruhe	0.15	3	1

20010			1	Dynamic Microsystems Semiconductor Equipment GmbH	Im Wiesengrund 17	78315	Radolfzell	0.09	2	
20010	16318610	82.32		DMS DYNAMIC MICRO SYSTEMS SEMICONDUCTOR EQUIPMENT GMBH	Im Wiesengrund 17	78315	Radolfzell	0.09	2	2
20010	16318611	82.32		DMS Dynamic Micro Systems Semiconductor Equipment GmbH	Im Wiesengrund 17	78315	Radolfzell	0.09	2	2
19904			1	KARL BROTZMANN COMSULTING GmbH	von Scheffel-Strasse 34	92224	Amberg	0.10	2	
19904	19376777	98.80		Karl Brotzmann Consulting GmbH	Von-Scheffel-Str. 34	92224	Amberg	0.10	2	3
19904	19776576	98.80		K a r l B r o t z m a n n C o n s u l t i n g GmbH	Von-Scheffel-Str. 34	92224	Amberg	0.10	2	3
22278			1	Eerec Technology GmbH Development & Design	Borntalstrasse 9	36460	Merkers/Thür.	0.11	2	
22278	7273363	81.58		EuRec Technology GmbH Development & Design	Borntalstr. 9	36460	Merkers-Kieselbach	0.11	2	2
22278	7273364	81.58		EuRec Technology GmbH Development & Design	Borntalstr. 9	36460	Merkers	0.11	2	2
33792			1	A L M Ü PRAZISIONSWERKZEUG GmbH	Ohmder Strasse 12	73119	Zell	1.74	1	
33792	15622884	98.70		ALMÜ Präzisions-Werkzeug GmbH	Ohmder Str. 12	73119	Zell	1.74	1	8

Table 8: Extended Export Format

Every block comprises of the search term in the header followed by the associated candidates. The Searched and Found columns refer to the record number respectively ID of the search term and the candidates. A block header has no Identity as it only has the purpose to provide the search term fields for comparison with the candidate fields. The column Score indicates the absolute Identification Potential of the search term (see 2.2.2). Cnt contains the number of candidates, while Run designates the number of the search run responsible for the retrieval of the candidate. Extended Export files are always sorted in descending order by Cnt and ascending order by Score. Manual labeling can be accelerated by entering the default label per block into the header row of the Equal column and marking only the exceptions with the inverse label (1 = valid match = true positive, 9 = invalid match = false positive, 0 or empty = replaced by default value in block header, ignored otherwise).

Software

SearchEngine: https://github.com/ThorstenDoherr/searchengine

The SearchEngine is a stand-alone software tool for Windows. Follow the instructions on the GitHub for the installation. The package includes a manual and the source code. The program is written in Microsoft Visual Foxpro and C. It implements multiprocessing curtesy of ParallelFox by Joel Leach. This package also includes the ML scripts for the statistical software STATA discussed in this paper within the SEML directory.

Brain: https://github.com/ThorstenDoherr/brain

The brain package is a module for the statistical software STATA. Follow the instructions on the GitHub for the installation. It is also available from SSC (Statistical Software Components) and can be directly installed within STATA by entering the following command sequence: ssc install brain

References

Comber S. and D. Arribas-Bel (2019), `Machine learning innovations in address matching: A practical comparison of word2vec and CRFs´, *Transactions in GIS*, 23(2), 334-348.

Doherr T. (2021), `Disambiguation by Namesake Risk Assessment´, *ZEW Discussion Paper No. 21-021*

Doherr T. (2017), `Inventor Mobility Index: A Method to Disambiguate Inventor Careers´, *ZEW Discussion Paper No. 17-018*

Jaccard P. (1902), `Lois de distribution florale dans la zone alpine´, *Bulletin de la Societe Vaudoise des Sciences Naturelles*, 38(144), 72.

Monath N., C. Jones and S. Madhavan (2021), `Disambiguating Inventors, Assignees and Locations´, *USPTO PatentsView*, https://patentsview.org/disambiguation.

Russell R.C. and M. King Odell (1918), `Index', *United States Patent and Trademark Office*, US1261167A.

Schönpflug W. (1969), `N-Gramm-Häufigkeiten in der deutschen Sprache', *Zeitschrift für experimentelle und angewandte Psychologie*, vol. 16, 157-183.

Steorts R. C., S. L. Ventura, M. Sadinle and S. E. Fienberg (2014), `A Comparison of Blocking Methods for Record Linkage´, *Lecture Notes in Computer Science*, vol. 8744.