TI 2023-055/VIII
Tinbergen Institute Discussion Paper

Fuzzy firm name matching: Merging Amadeus firm data to PATSTAT

Leon Bremer[1]

1 Vrije Universiteit Amsterdam

Fuzzy firm name matching: Merging Amadeus firm data to PATSTAT

Leon Bremer*

September 14, 2023

Abstract

When merging firms across large databases in the absence of common identifiers, text algorithms can help. I propose a high-performance fuzzy firm name matching algorithm that uses existing computational methods and works even under hardware restrictions. The algorithm consists of four steps, namely (1) cleaning, (2) similarity scoring, (3) a decision rule based on supervised machine learning, and (4) group identification using community detection. The algorithm is applied to merging firms in the Amadeus Financials and Subsidiaries databases, containing firm-level business and ownership information, to applicants in PATSTAT, a worldwide patent database. For the application the algorithm vastly outperforms an exact string match by increasing the number of matched firms in the Amadeus Financials (Subsidiaries) database with 116% (160%). 53% (74%) of this improvement is due to cleaning, and another 41% (50%) improvement is due to similarity matching. 18.1% of all patent applications since 1950 are matched to firms in the Amadeus databases, compared to 2.6% for an exact name match.

Keywords: Fuzzy name matching, supervised machine learning, name disambiguation, patents.
JEL codes: C81, C88, O34

1 Introduction

Merging databases is a common procedure for any empirical researcher. Often databases do not have common identifiers to merge on. This chapter presents an algorithm to merge databases using firm names as merging keys. For illustrative purposes I specifically taylor the algorithm to consider the problem of merging firms from the Amadeus databases to patent applicants in the PATSTAT database. Where the Amadeus databases have consistent within-database firm identifiers, PATSTAT does not have consistent applicant identifiers nor does it have consistent spelling of applicant names. Additionally the size of

*PhD candidate at the Spatial Economics department of the Vrije Universiteit Amsterdam, School of Business and Economics. Email: L.Bremer@vu.nl.

the databases creates computational challenges. This merging problem therefore requires a fuzzy string matching algorithm that can efficiently process millions of firm names.

The problem of merging databases in the absence of unique identifiers is not new. Others have encountered such problems, and several solutions are proposed in the literature. Whereas the application of merging Amadeus to PATSTAT data is particularly relevant for economic research, other fields especially provide methodological suggestions, like the academic literature in computer science, and even less formal literature like blog posts. I will touch upon some previous contributions to the topic.

Using text as data is an upcoming field of research, boosted by the improvements in hardware and software available to a broader group of people. An introduction to using text as data is provided by Gentzkow et al. (2019), in which several methods for text analysis are introduced in combination with several economic applications. Dugoua et al. (2022) follow up on this paper by specifically reviewing applications of text as data in environmental economics. For example, Kelly et al. (2021) build an innovation measure from patent texts.

To determine the similarity between two firm names, the character strings need to be compared and scored. Several common scoring measures are used in the literature to compare two strings. Edit distances count overlaps between the strings or the number of edits needed in order to transform the strings to a common value. The longest common subsequence (LCS) measures the longest common sequence between the two strings when only allowed to delete characters. The more common Levenshtein distance counts the number of edits to one string needed to produce the other string. It allows for deletion, insertion and substitution. The Jaro-Winkler distance is more complex and takes into account the shares of matching characters, regardless of their position, the share of matching characters that are in a different position, and the length of the common substring at the start of the strings. The literature further uses the cosine similarity. Text strings are then first converted into numerical vectors by scoring each string for its text features. The dot product between these normalized vectors then results in the geometric angle between the vectors, thereby producing a similarity score between the underlying character strings.

Peruzzi et al. (2014) introduce the REMERGE algorithm that performs firm matching between the PATSTAT database and other firm databases, like Orbis and Amadeus. The algorithm creates a set of candidate matches for each entry in PATSTAT across the other database. These candidates are selected to match the country and the first characters of the firm name. The country information in PATSTAT is derived from a geocoding exercise. All PATSTAT firm names are then compared to their candidates using the Levenshtein ratio and the Jaro-Winkler distance. A set of matching variables are then constructed from the available information. As the authors use a 2011 version of PATSTAT, their algorithm also had to guess whether PATSTAT entries are persons or firms. This has become irrelevant in later versions of PATSTAT, as the type of applicant is now mostly known. Subsequently the authors create a training set by manually determining which matches are correct from a set of 1,013 PATSTAT entries and their respective candidate matches. A penalized model is then fitted to the training data. The algorithm considers

the highest-scoring candidate to be a match, thereby implicitly enforcing a one-to-one match between PATSTAT entries and firms in the other database. The authors show that their REMERGE algorithm outperforms a simpler algorithm that only uses the Levenshtein distance.

Being concerned with merging USPTO patent data with public firms from the Compustat Global database, Bena et al. (2017) use fuzzy name matching algorithms. In a follow-up exercise, Bena et al. (2019) use search engines to further disambiguate firm names. Patent assignees are entered into a search engine and the resulting top suggestion for a company webpage is stored. Different company names and different companies within an ownership structure are often successfully pointed to the parent firm's webpage. Using a blacklist of webpages, the authors avoid firm names to be linked to generic webpages, like social platforms or online business registers. Lastly, the authors compare results to their fuzzy name matching results and others' results, and finally they perform manual disambiguations on conflicted cases.

As PATSTAT is notorious for its messy name formats and rather uninformative within-database applicant identifiers, researchers have specifically tried to improve the naming variables in the database itself. Two of these attempts have even been incorporated into later PATSTAT versions. The Harmonized Applicant Names (HAN) database is an attempt by the OECD to identify the same firms within the PATSTAT database. HAN names are about 40% less plentiful than the original naming formats (Dernis, n.d.). The PATSTAT Standardized Name (PSN) harmonization uses an automated cleaning algorithm and a manual cleaning for the top 2,700 applicants (Callaert, 2017).

A related, but arguably more complicated, problem is the disambiguation of inventor names. People's names can be differently spelled due to abbreviated first names, changing order of names, and multiple people having the same name. An early attempt at disambiguation is performed by Trajtenberg et al. (2006) who propose an algorithm that uses the way words are pronounced (Soundex) as well as further patent information, like address and technological field, in order to identify inventors within a patent database. Further efforts of researchers to identify either or both the inventors and applicants rely on cleaning, matching and filtering using supervised methods (Pezzoni et al., 2014), matching using the Levenshtein distance and consecutive manual checks (He et al., 2018), disambiguations of name and location data (Balsmeier et al., 2015), or on geocoding (De Rassenfosse et al., 2019; Morrison et al., 2017).

Further, non-scientific contributions are highly informative for these problems. Computer science problems and solutions are often shared in online communities, forums and blogs. I will cover two examples here that both use Python as a programming language. Berg (2017) provides a short and simple introduction to TF-IDF vectorization using n-grams and subsequent cosine similarity matching. A short application of firm name matching is provided. The advantage of vectorization and the use of the cosine similarity is its computational speed. Nijhuis (2022) provides a more elaborate and flexible cleaning and matching algorithm, accompanied by a Python package. Their algorithm contains several options for firmn name cleaning, like removing special characters and

some legal information (e.g. Corp, Inc), as well as several similarity options, like the cosine similarity or discounted Levenshtein distance. The algorithm ends with a manual check of produced matches.

This chapter proposes a high-performance fuzzy firm name matching algorithm to merge the Amadeus and PATSTAT databases. The algorithm takes four main steps, namely (1) firm name cleaning, (2) vectorization and similarity scoring to propose candidates, (3) evaluation of candidates using a supervised machine learning method, and (4) firm grouping using community detection.

The proposed algorithm increases performance significantly, as it matches 116% and 160% more firms from the Amadeus Financials and Subsidiaries databases, respectively. The algorithm also improves the number of applicants linked to a firm, as in total the number of linked patent applicants increases with 419% (454%) for the Amadeus Financials (Subsidiaries) database. This means that 18.1% of all patent applications have at least one applicant that is merged to a firm in the Amadeus data, compared to 2.6% for an exact name match.

In step 4 several solutions are provided to the problem of patent applicants matching to multiple firms. Matching on the individual firm level can be enforced, but I also provide a grouping solution. Groups of firms are constructed using a community detection algorithm that uses co-occurences in the matches to patent applicants. The benefit of this solution is that firm groups are identified and that no links are lost in this disambiguation exercise.

The methodologies used in this algorithm are readily available to anyone facing similar problems. Implementation is in Python and the matching can be performed efficiently on an everyday laptop.

This chapter proceeds with a description of the proposed algorithm in Section 2. The application of the proposed algorithm to the merging of the Amadeus and PATSTAT database is described and evaluated in Section 3. A discussion of the algorithm and (computational) considerations follows in Section 4. Section 5 concludes.

2 Methodology

The proposed matching algorithm takes four main steps, namely (1) cleaning of character strings, (2) similarity scoring, (3) a decision rule, and (4) disambiguation of the resulting matching links.

Step 1: Preprocessing

Text data is often messy. Misspellings and inconsistent spellings will obscure any comparison between texts. Cleaning the text according to some rules will therefore improve the comparability and the likelihood of identifying matching texts. There is no one-size-fits-all solution for text preprocessing, but here several suggestions and considerations are provided.

Consider a very simple example in which one needs to determine whether two firm names refer to the same firm. The firm names are s_1 = "Téchnology-company inc." and s_2 = "TECHNOLOGY COMPANY INCORPORATED". Asking a machine whether s_1 equals s_2 will result in a negative response. Asking a human whether s_1 and s_2 are similar will likely result in a positive response. What implicit transformations do we humans make in order for us to see two strings referring to the same firm?

There are three main transformations that we make. First, we ignore casing, as we notice that the casing in neither of the strings is informative. Second, we consider that accents and punctuation might have been removed from s_2. Third, we understand that "inc." likely refers to "incorporated". It is these human-like transformations that one should aim to replicate in the preprocessing stage.

The extent of optimal character name cleaning depends on the problem at hand. When one knows that names across databases are spelled in similar ways, one might want to abstain from thorough cleaning. Whereas cleaning increases the likelihood of more correct matches, it also increases the likelihood of more incorrect matches by introducing mistakes in the strings. Cleaning already-clean strings might do more harm than good. A thorough manual inspection of the text under consideration is strongly advised.

When dealing with special characters and accents, one should consider their role in the string. Some characters can be removed without losing identifying information. Think of the hyphen in s_1. Other characters should rather be replaced, as for the accented "é" in s_1. Replacing it with a regular "e" is the appropriate transformation. Simply removing all characters other than A-Z would be a mistake when there is information in accented characters. Instead a mapping of characters to their harmonized variants should be used in this case. I propose to replace all special characters in the text data by their aesthetic equivalents. For example, the special characters š and ç are replaced by their aesthetic equivalents s and c, respectively. The risk of such replacements is that they disregard linguistic context, as š might represent a different letter in Finnish as it does in Croatian. A mapping taking such linguistic origins into account would be preferred, but such information is not always available.

When text contains legal terms, as in the example of s_1 and s_2, one can use dictionaries to replace or remove such terms. When replacing legal terms I suggest to substitute full-length terms for their respective abbreviated variants. The benefit of replacing terms in this direction, and not from abbreviation to full-length term, is that the resulting string contains fewer characters that refer to rather uninformative legal information. A larger share of the characters will likely be more informative.

For some cleaning the order of steps is important, especially when harmonizing legal terms. When interested in removing (abbreviated) legal terms, it is necessary to consider word boundaries. If all instances of "inc" are removed, the risk of removing non-legal information is great, as the company name "Fixed income management Inc" will be transformed to "Fixed ome management ", which is undesirable. Regex expressions allow searching for "inc" with word boundaries on both sides, resulting in the desirable outcome of "Fixed income management ".

Taken all these considerations into account, I propose to use the following steps for cleaning in the given order. (A) Remove punctuation, like commas, dots, dashes, quotes, etc. (B) Ignore case by transforming all text to lowercase. (C) Handle any legal terms by replacing full-length terms with their abbreviated counterparts.[1] Alternatively, one can remove full-length and abbreviated terms altogether. (D) Replace any accented characters for their unaccented counterparts. (E) Remove any space characters, including tabs and other special space characters.

Returning to the earlier example, following the proposed cleaning steps will bring both firm names to the common "technologycoinc" term. A computer will now judge the strings to be exactly the same. Thereby the string preprocessing successfully mimicks the transformations humans make in the string comparison task.

Step 2: Vectorization and similarity scoring

Whereas in the above example the cleaned strings are identical, this will not always be the case. The resulting cleaned character strings should therefore be compared in a fuzzy way, allowing for some differences across the cleaned strings. To do so, the character strings will be inspected for features. Each feature in a string will be assigned a numerical score. The character string is thereby transformed into a numerical vector. This step is therefore also called vectorization. The similarity between these vectors can then be calculated using the cosine similarity.

To guide this part of the algorithm, consider two similar but different cleaned firm names, the previously considered "technologycoinc" and "tecnolologyco" (the cleaned version of "TECNOLOLOGY COMPANY"). Note the several misspellings in the second firm name (missing 'h' and additional 'ol') and the lack of one of the legal terms ('inc'). The task in this step is to quantify the similarity between these strings. A human would probably judge these two strings quite similar, assigning a fair likelihood that these strings refer to the same firm.

With only the firm name as information, all features should come from the firm name. A common approach to extract features from text is to split the text according to some rules. For long texts, one can split on word boundaries to produce a set of words. As firm names consist of only a few words this would create the character vectors ["technology", "co", "inc"] and ["tecnolology", "co"]. Note that only one element of the vectors overlaps ("co"). A more granular split is more appropriate in settings with short texts, like firm names. Using n-grams one can split text in a set of n-length characters in a rolling manner. A 3-gram partitioning creates a vector of text data for each of the example strings, such that

$$\vec{w}_1 = [\text{tec}, \text{ech}, \text{chn}, \text{hno}, \text{nol}, \text{olo}, \text{log}, \text{ogy}, \text{gyc}, \text{yco}, \text{coi}, \text{oin}, \text{inc}] \tag{1}$$

[1] A list of legal terms and their abbreviations by country is provided by GLEIF (2021). This list is slightly adjusted before use. When removing or replacing such terms, always start with the longest terms to prevent removing parts of words only. For example, replace or remove "limited corporation" before "limited", otherwise the "corporation" part will remain.

is the word vector of "technologycoinc" and

$$\vec{w}_2 = [\text{tec}, \text{ecn}, \text{cno}, \text{nol}, \text{olo}, \text{lol}, \text{olo}, \text{log}, \text{ogy}, \text{gyc}, \text{yco}] \tag{2}$$

is the word vector of "tecnolologyco". In general such n-gram representations yield $L - (n-1)$ elements for a string of length L. This partitioning is likely more forgiving to misspellings as more elements overlap between these two vectors compared to the word-partitioning from earlier. Additionally, word order matters as several n-grams span multiple words. Each element of the vector represents a feature of the underlying text.

Additional information can be embedded into the vectors by giving each element a numerical score. When scoring each element on importance, the vector becomes a numerical list of which features to stress most in a later comparison. I use TF-IDF vectorization to count and weigh the elements in $\vec{w}_i$. The term frequency (TF) and inverse document frequency (IDF) are calculated separately and combined afterwards.

The TF part counts the number of occurrences of each element in its own vector. The TF function can be presented as

$$TF(t, d) = |\{t : t \in d\}| \tag{3}$$

with t each term in a vector and d the vector itself. It effectively counts for each unique term how often it occurs in the respective vector. Performing a TF transformation on $\vec{w}_1$ and $\vec{w}_2$ creates the following matrix

$$\begin{array}{l} \\ \text{technologycoinc} \\ \text{tecnolologyco} \end{array} \begin{array}{c} \begin{array}{cccccccccccccccc} \text{tec} & \text{ech} & \text{chn} & \text{hno} & \text{nol} & \text{olo} & \text{log} & \text{ogy} & \text{gyc} & \text{yco} & \text{coi} & \text{oin} & \text{inc} & \text{ecn} & \text{cno} & \text{lol} \end{array} \\ \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 2 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \end{array} \tag{4}$$

where row and column axis labels indicate the original string and the 3-grams, respectively. One already notices that the numerical row vectors are similar, due to much textual overlap between the underlying string names. Also note how each numerical vector keeps its link with the underlying n-grams, producing zeros for n-grams that do not occur in the character string.

The IDF adds information from the broader data source. The IDF calculates weights for each element, whereby often-occurring elements, *across* character strings, are down-weighted as they contain less identifying information. The chosen IDF function can be described as

$$IDF(t, D) = \ln\left(\frac{N+1}{n_t+1}\right) + 1 \tag{5}$$

where $N = |D|$ is the total number of firm names (or documents), and where n_t is the number of documents containing term t, i.e. $n_t = |\{d \in D : t \in d\}|$. The additions of a 1 in both the numerator and the denominator are optional and help smooth the fraction. The addition of one to the IDF is to set the minimum score to one instead of zero (Pedregosa et al., 2011). One occurs when t is in each and every document, i.e. when $n_t = N$. The natural logarithm diminishes the magnitude of the IDF correction.[2]

[2]Other definitions for the TF and the IDF can be found on Wikipedia. The `sklearn` package in Python also allows for different IDF definitions.

Performing the IDF vectorization on the text strings is simple in the two-string example. The IDF can only take two values, as n_t either equals 1 or 2. When $n_t = N = 2$, the IDF equals 1 and when $n_t = N = 1$ the IDF equals $\ln\left(\frac{2+1}{1+1}\right) + 1 \approx 1.41$. The resulting vectors are again forced to be of the same length, inserting zeros for features that do not occur in a string. To save space I will not present the matrix resulting from the IDF transformation of the two strings. Instead I directly present the combined TF-IDF matrix, for which each element is produced by multiplying the TF value with the IDF value, i.e. the element-wise product is taken such that $TF\text{-}IDF = TF \circ IDF$. The following TF-IDF matrix results

$$\begin{array}{c} \\ \text{te...nc} \\ \text{te...co} \end{array} \begin{array}{c} \begin{array}{cccccccccccc} \text{tec} & \text{ech} & \text{chn} & \text{hno} & \text{nol} & \text{olo} & \text{log} & & \text{inc} & \text{ecn} & \text{cno} & \text{lol} \end{array} \\ \begin{bmatrix} 1 & 1.41 & 1.41 & 1.41 & 1 & 1 & 1 & \cdots & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 2.81 & 1 & \cdots & 1 & 1.41 & 1.41 & 1.41 \end{bmatrix} \end{array} \tag{6}$$

where the 2.81 occurs due to the TF transformation resulting in 2. To preserve space several columns with only 1s are omitted, and the row labels are abbreviated.

As firm names are often short, the TF term will mostly only take the values 0 and 1. Most data sources contain many firm names, resulting in a large variety of IDF terms. The TF-IDF matrix will have dimensions corresponding to the number of considered strings and the number of unique features (i.e. n-grams). As the number of unique features is large and the number of features per firm name is small, the matrix will be very sparse (i.e. containing many zeros).

A consideration at this step is what data is used to produce the unique n-grams and their IDF terms. Or put differently, what data is used to *train* the vectorizer? As the IDF term punishes the frequency of n-grams, it is important not to submit duplicates to the vectorizer. It is also best to train the IDF vectorizer on just one data source. When entering the combination of the to-be-matched data sources the features of the firm names that occur in both sources, the actual matches, will be punished due to their increased frequency. This is undesirable. Simply removing duplicates will not solve this issue, as slightly differently spelled names referring to the same firm will still feed many duplicate features to the IDF vectorizer.

To determine the similarity between the produced numerical vectors, I employ the cosine similarity. The cosine similarity states that the dot product of two normalized vectors results in the cosine of the geometric angle between the two vectors. Specifically

$$\cos(\theta) = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\|_2 \|\vec{b}\|_2} \tag{7}$$

where $\|\vec{a}\|_2$ refers to the Euclidean norm of vector $\vec{a}$, or the length of $\vec{a}$ in Euclidean space, and θ is the angle between the vectors. This similarity also demonstrates how vectors consist of a directional and a length component. When ridding vectors of their length component, the only remaining difference between the vectors is their direction. The direction difference is the angle between the two vectors.

In order to reveal the angles between each row in Equation 6, each vector must first be normalized by dividing all elements by the Euclidean distance of the respective vector. Defining the resulting matrix of normalized vectors as M, performing a matrix multiplication with itself reveals the similarity between each vector in M. Mathematically

$$C = MM' = \begin{bmatrix} \vec{\hat{w}}_1 \cdot \vec{\hat{w}}_1 & \vec{\hat{w}}_1 \cdot \vec{\hat{w}}_2 \\ \vec{\hat{w}}_2 \cdot \vec{\hat{w}}_1 & \vec{\hat{w}}_2 \cdot \vec{\hat{w}}_2 \end{bmatrix} = \begin{bmatrix} \cos(\theta_{11}) & \cos(\theta_{12}) \\ \cos(\theta_{21}) & \cos(\theta_{22}) \end{bmatrix} = \begin{bmatrix} 1 & \cos(\theta_{12}) \\ \cos(\theta_{12}) & 1 \end{bmatrix} \tag{8}$$

where $\vec{\hat{w}}_i$ refers to the cleaned, vectorized and normalized vector $\vec{w}_i$ from earlier, θ_{ij} is the angle between $\vec{w}_i$ and $\vec{w}_j$, and M' is the transpose of M. The last step highlights the symmetry of this simple 2×2 matrix. The ones on the diagonal indicate that the angle of a vector with itself is zero ($\cos(0) = 1$). Considering the two-string example from earlier, one finds tat $\cos(\theta_{12}) = \cos(\theta_{21}) \approx 0.62$, meaning $\theta_{12} \approx 0.90$ or $52°$. To put that number in perspective, the dot product ranges between 0 and 1, and the angle ranges between 0 and $90°$.

Equation 8 shows the redundancy of many of the calculations, when only interested in the across-database comparisons. Consider a larger example, where the vectorizer is trained on m_1 firm names to create a TF-IDF matrix A with dimensions $m_1 \times n$. The vectorizer is then used to produce a TF-IDF matrix B for the m_2 firm names in another database, with the dimensions of B being $m_2 \times n$. Here m_1 and m_2 do not need to be equal. Normalizing the rows of both matrices gives the hatted versions of the matrices, $\hat{A}$ and $\hat{B}$. The angles between character strings across databases are then found through

$$D = \hat{A}\hat{B}' = \begin{bmatrix} \cos(\theta_{11}) & \cdots & \cos(\theta_{1m_2}) \\ \vdots & \ddots & \vdots \\ \cos(\theta_{m_1 1}) & \cdots & \cos(\theta_{m_1 m_2}) \end{bmatrix} \tag{9}$$

which yields an $m_1 \times m_2$ matrix of $\cos(\theta)$s where the element on row i and column j refers to the cosine of the angle between the ith firm name in A and the jth firm name in B.

The last option to Step 2 of the algorithm is to redo steps 1 and 2 with different parameter choices in the preprocessing step. For example, one could perform different cleaning on the firm names and redo the similarity scoring. All scores should be stored together with the methods used. Step 3 will then provide the decision rule.

Step 3: Decision rules

After obtaining similarity scores between character strings, one needs to determine which combinations are a match. Although smaller θs mean that firm names are more similar, there will always be exceptions. That means there is no one cutoff on θ that guarantees that values below are true matches and values above are no true matches. Whatever decision rule is therefore taken, it will always result in some discrepancy.

Therefore I propose to take a two-step decision-rule. First, a simple cutoff value for θ is taken based on inspection of the results. Let us define this angle as $\bar{\theta}$. All combinations i, j for which $\cos(\theta_{ij}) > \cos(\bar{\theta})$ are considered *candidates*. Second, depending on the number

of candidates, one can manually check all candidates or employ a supervised machine learning method to identify the true matches.

The supervised machine learning method involves predicting the true matching status of a candidate combination with available information. Anything containing information on the match likelihood could be used. Think of whether the names start with the same letter, whether the first letters of one string are contained in the other string, but also the similarity score θ. To use the available characteristics to predict true matching status, a binary outcome model can be fitted such that

$$p_{ij} = P[q_{ij} = 1|x_{ij}] = F(x_{ij}\beta) \tag{10}$$

where $q_{ij} = 1$ if i and j are a truthful match, x is a vector of explanatory characteristics, and F is a cumulative distribution function (CDF) making sure the conditional likelihood on the left-hand side is bounded between 0 and 1. Here I will use the CDF of the normal distribution, making Equation 10 a probit model.[3]

As the vector β is unknown, one cannot readily determine p_{ij}. Instead, β will be fitted using a training sample. This training sample has to be constructed manually from the candidates. Manually determining whether candidate matches are true matches or not gives q_{ij} for a sample of candidates. The sample can then be used to fit β, which in turn can be used to predict p_{ij} for the out-of-sample candidates, with the prediction denoted as $\hat{p}_{ij}$.

Lastly, one needs to determine the threshold $\bar{p}$ for which $\hat{p}_{ij}$ is high enough to consider the candidate a match. Machine learning algorithms are often evaluated using precision and recall, which are the share of justified matches in the set of identified matches and the share of justified matches in the set of all true matches, respectively. One can also use a combination of precision and recall, as e.g. proposed in the F1 score.[4] These scores can be calculated for the training sample, as for that sample both the prediction and the truth value are known. One can then choose a cutoff $\bar{p}$ that satisfies the restrictions on the evaluation criteria. Any out-of-sample $\hat{p}_{ij} > \bar{p}$ will then be considered a match.

Considerations for the decision rule are the chosen $\bar{\theta}$, the size of the training sample, and the acceptable values for the evaluation criteria, e.g. precision, recall or the F1 score. A higher $\bar{\theta}$ will lead to more candidates and a larger share of incorrect candidates, while increasing the nominal number of true matches in the set of candidates. The larger the training sample, the more accurate the model in Equation 10 can be fitted, but the more manual labelling has to be done. Setting looser evaluation criteria will result in more matches, a larger share of identified correct matches (recall), but also a larger share of incorrect matches (precision).

[3] For a discussion of binary outcome models one can consult Cameron and Trivedi (2005, pp. 463-474).

[4] The F1 score ($F1$) is composed of the precision (p) and recall (r) scores, specifically such that $F1 = \frac{2pr}{p+r}$.

Step 4: Disambiguation

The last step is to disentangle the resulting matches. So far there have been no rules on multiplicity of matching links. This means that one firm can match with multiple patent applicants and that one applicant can match with multiple firms. While the first should be allowed, the second is problematic. When using these matches to do follow-up research the multiplicity of links might cause double counting. Step 4 therefore provides several solutions for disambiguation. Each solution will enforce that each patent applicant can only link to one firm (group).

I propose three disambiguation methods that all have their own strengths and weaknesses. First, one can use the $\hat{p}$s to pick the most likely link and remove all other links the patent applicant is involved with. Any ties can be broken by using firm characteristics. One could for instance pick the largest firm. Second, a community detection algorithm can be used to identify firm groups. Firm groups are determined by co-occurence between firms in their links with patent applicants. The motivation for this grouping is that if firms often link to the same patent applicant, their names must be very similar and therefore they are likely related. If patent applicants then link to multiple firm groups (communities) ties can be broken by linking to firm group consisting of most firms. Third, the largest firm in each community that has consolidated financial statements can be identified. The consolidated statements make sure that the data encapsulates the financial data of the entire firm group. This results in links between patent applicants and firm groups.

Figure 1 gives an overview of the proposed algorithm. The steps indicate the steps discussed as above, while each solid node represents a data table in a specific form, and each dashed node represents an operation along an arrow.

3 Application

This section takes the proposed methodology to the data. The task is to merge firms from the Amadeus databases to applicants from the PATSTAT database by name. As for this application I am interested in patenting activities of *firm groups*, the merge should be sufficiently fuzzy to allow for generic differences in firm names. For example, it should be forgiving enough to consider "ABC Car Technology Holdings Inc" and "ABC Car Technology SA" a match. Throughout this application outcomes and performance of the algorithm will be presented and discussed.

3.1 Data

PATSTAT is a worldwide patent database and contains information on the patent level, such as applicants, inventors, and technology types. The PATSTAT database contains information on patenting by firms and individuals at patent offices around the world for many years. This application uses the PATSTAT 2018 Autumn edition. It contains millions of patents and patent applicants. The main difficulties with the database is its

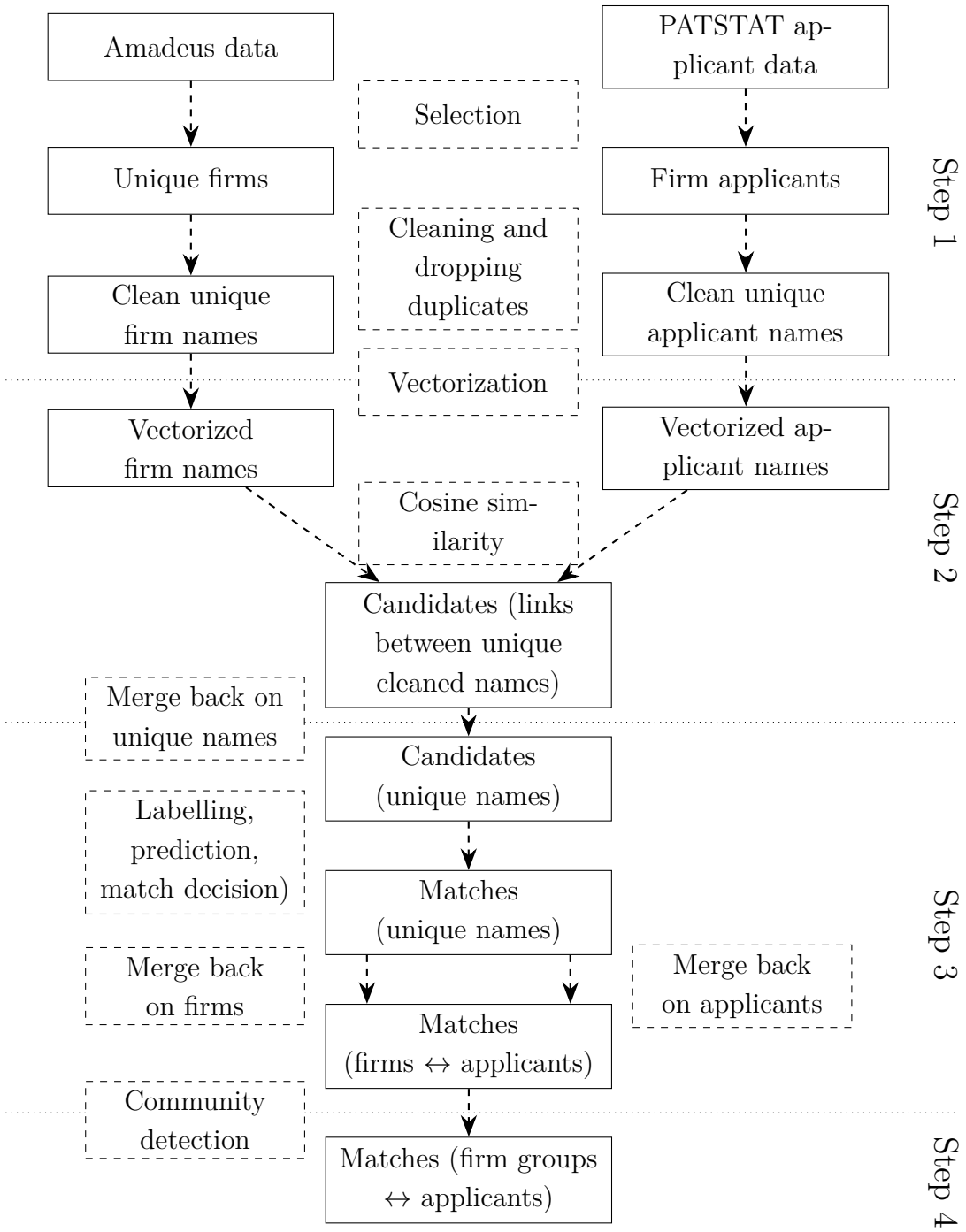

Figure 1: The fuzzy matching program.

Note: This schematic illustrates the main steps involved in the fuzzy name matching program. Boxes with solid outlines indicate data tables, while the arrows and accompanying boxes with dashed outlines describe the operational steps of the program.

messiness. Internal identifiers for applicants are unreliable, applicant names are inconsistently spelled, and misspellings are common.[5] Anyone interested in studying the patenting behavior of firms or individuals has to disentangle the numerous applicant entries and figure out which refer to the same real life applicant. PATSTAT's applicant names also contain lower- and uppercase characters, punctuation and accented characters, as well as a multitude of special characters.

Two Amadeus databases are considered, the Amadeus Financials database and the Amadeus Subsidiaries database. The Amadeus Financials database contains firm-level business information, like revenue, number of employees, profits, asset book values, etc. The Amadeus Subsidiaries database contains information on ownership links and some business information on firms down the ownership hierarchy. The internal identifiers of the Amadeus databases are consistent and the firm names are to some extent harmonized and rather clean.

Examples of firm names in these three databases are presented in Table 1. The table presents the names without any processing. The firm names are not chosen randomly, but they represent the largest firms and most active innovators, based on the number of employees and the number of patent applications, respectively. These names are relatively clean, likely because they are the most occuring firms. Alternative spellings and misspellings will for instance not rank as high in the patent data. Most noticeable in terms of harmonization are the legal terms that are not consistently represented. Some terms are written in full, some are abbreviated, and some contain interpunction.

Due to the lack of externally valid firm identifiers, these databases are best merged on their firm names. And as naming conventions are inconsistent a fuzzy string matching approach is suitable.

When working with small databases, one could opt for manual matching. For larger databases the problem needs a more automated solution. The scale of the problem at hand becomes clear from Table 2. The table presents the number of firms and applicants in the databases. Looking at the unique names only, the problem requires comparing $8,311,667 \times (521,877 + 1,143,181) \approx 13.8T$ pairs of firm names. A simple check of the average character length of the names shows the names are similar in length, signaling some degree of comparability.

Usually more information than only the firm name is known. In the Amadeus databases the country of the firm is known and from the PATSTAT database address and patent office information is known. Furthermore, both the Amadeus Financials as well as the PATSTAT database offer multiple name variables. Depending on the application such additional information can be used in the proposed algorithm. As the application at hand seeks to merge firm groups to patent applicants, the location data of individual firms and applicants is purposely ignored. Also, firms can seek patent protection in multiple countries, also outside their headquarter's country.

[5] Applicant-related information is taken from the `TLS206_PERSON` table.

Table 1: Examples of firm names.

Amadeus Financials (NAME)
JOINT STOCK COMPANY RUSSIAN RAILWAYS
VOLKSWAGEN AKTIENGESELLSCHAFT
G4S PLC
COMPASS GROUP PLC
ISS A/S
ISS GLOBAL A/S
DEUTSCHE POST AG
TESCO PLC
BELLON
SODEXO
Amadeus Subsidiaries (SUBS_NAME)
WALMART INC.
VOLKSWAGEN AKTIENGESELLSCHAFT
AMAZON.COM, INC. via its funds
COMPASS GROUP PLC via its funds
G4S PLC via its funds
DEUTSCHE POST AG
ISS A/S
ISS GLOBAL A/S
UNITED PARCEL SERVICE INC via its funds
PETROCHINA COMPANY LIMITED
PATSTAT (person_name)
MATSUSHITA ELECTRIC IND CO LTD
HITACHI LTD
TOSHIBA CORP
CANON INC
MITSUBISHI ELECTRIC CORP
Samsung Electronics Co., Ltd.
NEC CORP
FUJITSU LTD
SONY CORP
RICOH CO LTD

The Amadeus firm names are from the firms with most employees in 2016. The PATSTAT applicant names are from those involved in most patent applications over all years. The original variable names are provided in parentheses.

Table 2: DATA SOURCE DESCRIPTIVES.

Source	Variable	Obs	Unique obs	Avg length
PATSTAT	person_name	10,160,817	8,311,667	29.0
Amadeus Fin	NAME	528,257	521,877	26.8
Amadeus Sub	SUBS_NAME	1,165,557	1,143,181	27.5

For the PATSTAT data individuals are filtered out. For the Amadeus datasets only observations that are unique across the identifier, name and country variables are kept for this exercise. The number of observations (obs) refers to the resulting number of names per data source. The unique number of observations removes the duplicate names per data source. The average length (avg length) of the raw firm names gives an indication of the type of data.

3.2 Applying the algorithm

Step 1: preprocessing. It is advisable to inspect the text data before deciding on the exact preprocessing steps to take. In any case, inspecting the unique characters in the data is highly informative. In the case of PATSTAT one will find many special characters that are likely uninformative. Inspecting the character strings containing these special characters helps judge whether these characters should be removed or be replaced. Such exercises can be prioritized by counting the number of occurrences of these special characters.

Table 3 presents the most common special characters and their counts across the three databases used in this application. To compare these numbers, the % column indicates the number of occurrences over the number of firm names in that database. Noticeable is that these special characters are not so special. They are regular punctuation and numbers. It is quite striking how many periods there are in the data. For each 100 firm names in the PATSTAT data, there are 79 periods. Note that some firm names might have more than one period, so more than 21% of firm names does not have a period at all. This abundance has mostly to do with legal abbreviations, like "inc.". Whereas the PATSTAT names are full of special characters, the top ten of characters is rather normal. And it is good to know that less than 1% of applicant names contains characters beyond punctuation and numbers.

For the application, the cleaning steps from Section 2 are followed. One way of demonstrating the success of the proposed string cleaning algorithm is to compare how the cleaned firm names perform in an exact merge compared to merging on the raw firm names. Table 4 shows the results of exact merges between the Amadeus databases and the PATSTAT data for different available name variables, including some already-harmonized applicant names discussed earlier in Section 1. Unique firms and entries are used, but duplicate firm names are not removed (see Table 2). There are several interesting findings worth highlighting. First, cleaning significantly increases the number of resulting links, at one occurrence even with a factor 3.5. Second, improvements are greatest for the unpro-

Table 3: Most common special characters.

	PATSTAT			Amadeus Fin			Amadeus Sub		
#	Char	Count	%	Char	Count	%	Char	Count	%
1	.	7,989,685	78.6	.	355,768	67.3	.	616,502	52.9
2	,	4,239,464	41.7	-	57,856	11.0	-	116,344	10.0
3	-	945,561	9.3	,	34,016	6.4	,	67,990	5.8
4	&	568,415	5.6	&	26,079	4.9	(	56,875	4.9
5	’	510,188	5.0	’	19,101	3.6	)	56,680	4.9
6	(	325,430	3.2	(	17,325	3.3	&	52,403	4.5
7	)	322,533	3.2	)	17,296	3.3	1	44,338	3.8
8	0	154,543	1.5	1	9,951	1.9	2	37,853	3.2
9	1	98,131	1.0	2	9,736	1.8	’	29,509	2.5
10	2	94,787	0.9	0	7,684	1.5	0	24,882	2.1

For the PATSTAT data individuals are not considered. Special characters are any character that is not in the English alphabet (a-z) and is not a regular space. The rank is indicated in the column #, and the % column indicates the average number of the special characters in a firm name.

cessed applicant name (person_name). Third, cleaned merging is most successful with the unprocessed applicant name, whether measuring by the number of links, number of firms in the Amadeus data, or number of patent applicants. Fourth, there are some issues with patent applicants merging to multiple firms in the Amadeus database. This occurs when (after cleaning) firms with different IDs in the Amadeus data have the same name. This issue is most prominent for the already-cleaned names (e.g. han_name). And, importantly, this issue does not seem to become more prominent with the proposed cleaning. Fifth, the cleaning especially improves the intensive margin of the matching. While more firms are matched to patent applicants (41% more for the NAME-person_name match), the number of patent applicants matching to firms increases from 2.8 to 5.7 (106% more). Without cleaning many innovative activities will not be assigned to firms, and will go unmeasured in any follow-up analysis.

Step 2: vectorization and similarity scoring. When training the TF-IDF vectorizer on the Amadeus Financials database using the NAME variable, I find a sparse matrix of size $520,352 \times 22,460$ corresponding with the 520,352 unique cleaned firm names and the 22,460 unique 3-grams. The most common 3-grams are presented in Table 5. One might recognize these terms, with “ltd” referring to the abbreviated legal term “limited”, the two 3-grams of the German legal term “GmbH” being represented, and all 3-grams of “holding” being represented.

When vectorizing the PATSTAT names, the vectorizer will not be retrained. The PATSTAT data is too messy. PATSTAT contains several differently-spelled names referring to the same firm, either through misspellings or inconsistencies. Entering all these alternative spellings into the vectorizer will result in undesirable punishments through the IDF score.

Table 4: Matching improvements from firm name cleaning.

Amadeus variable	PATSTAT variable	Cleaned	Links	Amadeus firms	PATSTAT entries
NAME	person_name	No	83,024	28,414	78,511
NAME	person_name	Yes	240,698	40,034	227,695
NAME	doc_std_name	No	84,092	15,858	77,794
NAME	doc_std_name	Yes	213,577	34,902	197,245
NAME	psn_name	No	136,164	11,782	104,362
NAME	psn_name	Yes	165,338	18,724	129,237
NAME	han_name	No	90,355	25,754	85,293
NAME	han_name	Yes	229,836	37,864	216,418
SUBS_NAME	person_name	No	113,361	38,072	103,927
SUBS_NAME	person_name	Yes	399,549	56,931	347,394
SUBS_NAME	doc_std_name	No	179,254	26,180	149,811
SUBS_NAME	doc_std_name	Yes	369,874	50,650	312,557
SUBS_NAME	psn_name	No	232,547	16,701	164,028
SUBS_NAME	psn_name	Yes	322,352	27,016	222,479
SUBS_NAME	han_name	No	167,152	37,833	144,623
SUBS_NAME	han_name	Yes	377,715	54,254	324,907

The Amadeus Financials variable NAME and the Amadeus Subsidiaries variable SUBS_NAME are merged to different applicant naming conventions in the PATSTAT database. The links present the number of firm names merged across databases, the number of Amadeus firms are counted by unique firm IDs. PATSTAT data has no consistent IDs, so the number of entries are counted. It is very possible that one firm in the Amadeus database is merged to multiple applicants in the PATSTAT database. Similarly, some patent applicants (entries) might be merged to multiple firms in the Amadeus databases if they have the same (cleaned) name. Note that the unique firms and entries are used from the Amadeus databases and PATSTAT database, respectively, but that duplicate names are not removed (see Table 2).

Table 5: Most common n-grams in the Amadeus Financials data.

#	3-gram	Counts	#	3-gram	Counts	#	3-gram	Counts
1	ltd	87,791	11	ldi	32,560	21	pro	21,454
2	ing	58,291	12	ter	31,549	22	nte	21,354
3	mbh	54,869	13	srl	27,802	23	spa	20,823
4	gmb	49,805	14	ngs	26,626	24	ran	20,738
5	ion	47,830	15	slt	26,511	25	sch	20,457
6	ent	39,097	16	and	26,383	26	ons	20,457
7	din	37,952	17	tio	26,369	27	ner	20,365
8	hol	34,770	18	ati	25,496	28	ers	20,129
9	est	34,599	19	str	24,414	29	tra	19,533
10	old	34,533	20	men	22,522	30	res	19,532

These are the most common 3-grams in the 520,352 cleaned firm names in the Amadeus Financials database, out of 22,460 3-grams. The firm name variable NAME is used here. Counts refer to the number of firm names containing the respective 3-gram and # refers to the rank. The firm names are thoroughly cleaned before vectorization.

When vectorizing the PATSTAT names, the vectorizer will not be retrained. The PATSTAT data is too messy, and creating additional unique n-grams is useless for the matching exercise as n-grams unique to one database will never match across databases. Ignoring these database-specific n-grams from the start results in the same similarity scoring later on. Another reason not to re-train the vectorizer on the PATSTAT data is that PATSTAT contains several differently-spelled names referring to the same firm, either through misspellings or inconsistencies. Entering all these alternative spellings into the vectorizer will result in undesirable punishments through the IDF score.

Instead, the trained vectorizer from Amadeus's NAME variable is used, meaning that those n-grams and IDF scores are used to vectorize PATSTAT's applicant names. Using that vectorizer one can create a similar table for the PATSTAT database, as presented in Table 6. Also here "ltd" is most prominent, but there are noticeable differences. With the top 15 3-grams one can spell "technology", something which is not as prominent in the Amadeus Financials database. This partially can be explained by the nature of the data source. Patenting firms are more likely to have "technology" in their names. Other differences are that the terms "holding" and "GmbH" cannot be spelled with these 30 3-grams. This might indicate that firms do not apply for patents through their holding firm, but rather through their subsidiaries. This supports the approach of considering firm groups to analyze patenting behavior.

Table 6: Most common n-grams in the PATSTAT data.

#	3-gram	Counts	#	3-gram	Counts	#	3-gram	Counts
1	ltd	1,619,193	11	log	523,790	21	ter	394,753
2	col	1,328,115	12	hno	515,248	22	che	374,078
3	olt	1,296,039	13	nol	500,910	23	ong	371,880
4	ing	785,604	14	ion	472,445	24	str	371,680
5	ech	678,661	15	ogy	449,918	25	inc	371,561
6	tec	666,945	16	han	441,486	26	men	370,161
7	ang	630,298	17	yco	438,267	27	eng	350,705
8	chn	577,391	18	ian	427,568	28	tio	349,589
9	olo	544,629	19	and	414,672	29	ica	345,845
10	ent	530,610	20	ine	406,556	30	ele	345,747

These are the number of occurrences of 3-grams in the PATSTAT database that also occur in the 520,352 firm names in the cleaned unique Amadeus Financials database. The PATSTAT variable person_name is used here. Counts refer to the number of applicant names containing the respective 3-gram and # refers to the rank. The applicant names are thoroughly cleaned before vectorization.

Separately, the Amadeus Subsidiaries data will be merged to the PATSTAT data. To do so, the vectorizer will be trained on the Amadeus Subsidiaries data and consecutively be used to also vectorize the PATSTAT data. When training the TF-IDF vectorizer on the Amadeus Subsidiaries database using the SUBS_NAME variable, I find a sparse matrix of size $1,136,376 \times 29,311$, where the first dimension refers to the number of unique cleaned

subsidiary firm names and the second dimension to the number of unique 3-grams. It usually makes sense that both dimensions scale in the same direction, i.e. more unique character strings produce more unique n-grams. The most common n-grams are reported in Table 7. Similar 3-grams rank high compared to that from the Amadeus Financials database. Noticeable is that the word "holding" cannot be fully spelled with this top 30, which makes sense as subsidiaries are less likely to be holding firms. For completeness Table 15 in the appendix presents the most common 3-grams in the PATSTAT data using the vectorizer that is trained on the Amadeus Subsidiaries data.

Table 7: Most common n-grams in the Amadeus Subsidiaries data.

#	3-gram	Counts	#	3-gram	Counts	#	3-gram	Counts
1	ltd	230,301	11	and	55,910	21	pro	48,257
2	mbh	122,506	12	men	55,610	22	tra	45,400
3	ing	111,137	13	din	55,550	23	ner	45,035
4	gmb	110,147	14	ngs	54,625	24	ica	44,868
5	ion	105,468	15	ons	54,372	25	erv	44,824
6	ent	99,016	16	str	52,807	26	res	44,716
7	est	71,643	17	nte	52,272	27	ers	44,610
8	ter	69,175	18	ati	51,892	28	hol	44,521
9	tio	62,910	19	srl	50,098	29	ste	44,418
10	slt	57,613	20	ser	48,924	30	old	44,244

These are the most common 3-grams in the 1,136,376 cleaned unique firm names in the Amadeus Subsidiaries database, out of 29,311 3-grams. The firm name variable SUBS_NAME is used here. Counts refer to the number of firm names containing the respective 3-gram and # refers to the rank. The firm names are thoroughly cleaned before vectorization.

Having vectorized the to-be-matched name variables, the angles between all combinations across databases can be calculated according to Equation 9. From eyeballing some combinations and their scores, the cutoff angle for a combination to be considered a candidate match is set at $\bar{\theta} = 0.4 \times \frac{1}{2}\pi \approx 0.628$. The 0.4 refers to 40% of the possible range of angles $[0, \frac{1}{2}\pi]$.[6] This cutoff results in 1,145,132 candidate matches for both Amadeus databases combined, and the split between databases is presented in Table 8. The Amadeus Subsidiaries database results in more candidate matches. But this database is also more than twice as large as the Amadeus Financials database (see Table 2). Percentage-wise the Amadeus Financials database results in more candidate matches.

Whereas 40% might seem lenient, comparing the number of candidates to the number of tested links, $(520,352 + 1,136,376) \times 6,558,030 = 10,864,871,925,840$, reveals that

[6] The possible range only spans a quarter circle, because the elements in the TF-IDF vectorizer, names are all non-negative. Non-negativity eliminates half the circle. Further, the order is not relevant when calculating the angle, i.e. the angle between $\vec{a}$ and $\vec{b}$ is the same as the angle between $\vec{b}$ and $\vec{a}$ (evident from $\cos(-\theta) = \cos(\theta)$), eliminating half a circle of possibilities. The exclusion of these two overlapping half circles leaves a quarter circle of possibilities.

Table 8: Number of candidate matches.

Train variable	Links	Amadeus unique cleaned names	PATSTAT unique cleaned names
NAME	420,478	75,449	133,547
SUBS_NAME	724,826	123,000	196,067
Total	1,145,304		

These are the number of candidate matches. Note that the numbers refer to unique cleaned names and might therefore underestimate the number of firms, as some firms share the same name and as some firms' names are processed to the same name.

only one in 10 million combinations is proposed as a candidate match. Moreover, some 15% of combinations have non-zero dot products, indicating that a 40% cutoff also filters out the fast majority of links with non-zero cosine similarity scores. In step 3 the accuracy of these candidates is evaluated.

The distribution of θs of the candidates is plotted in Figure 2. For these candidates, the majority of dot products results in non-zero angles, although there is a clear spike at zero representing exact matches (after cleaning). For efficiency reasons, only angles below the cutoff $\bar{\theta}$ (40%) are stored. The figure therefore only shows the distribution of candidate matches. The figure shows how the number of candidates increases exponentially with the angle θ, illustrating further that the 40% cutoff only selects a small part of non-zero dot products.

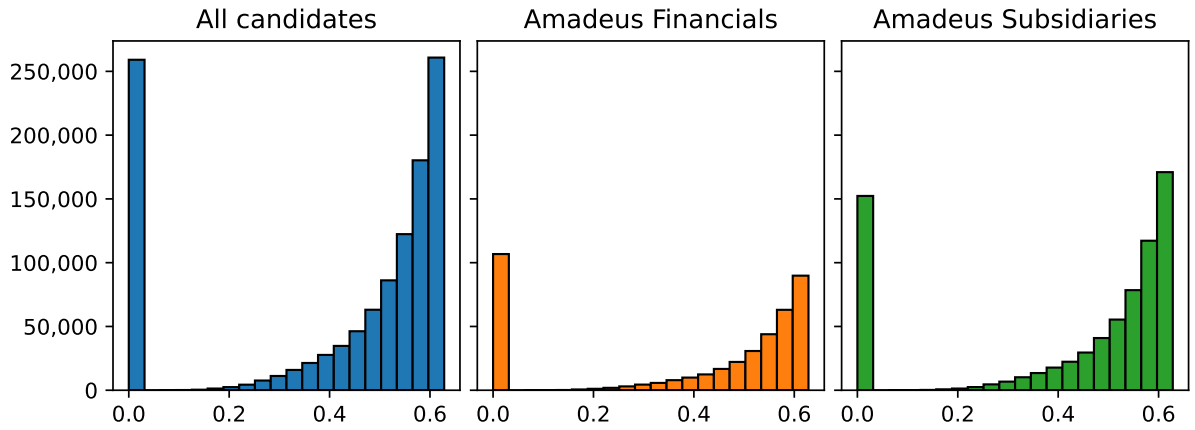

Figure 2: Distribution of angles of candidate matches.

Note: The distribution of the angles between the TF-IDF vectorized Amadeus firm names and PATSTAT applicant names within the 40% threshold (i.e. $\bar{\theta} = 0.4 \times \frac{1}{2}\pi \approx 0.628$ or 36°). The Amadeus Financials NAME variable, Amadeus Subsidiaries SUBS_NAME variable and PATSTAT's person_name are used. Figures are split by data source.

Step 3: decision rules. Of the 1,145,132 candidate matches some will be incorrect. Filtering out these incorrect matches is the task of this third step. While manually checking all candidates is an option, this is very laborious. It takes one person about 30-40 minutes per 1,000 candidate matches, resulting in 668 hours of work, or close to 17

40-hour work weeks. For most researchers this is not a reasonable consideration. Instead, a random sample of 1% of candidates will be taken and manually labelled. This task can be completed in one work day. This information will then be used to find relationships between candidate's characteristics and the true matching status, which in turn will be used to predict the true matching status of the 99% of unlabelled candidates.

While I refer to the labelled data as the truth, there is no guarantee manual checks of candidate matches lead to truth discovery. There is a human element to judging whether names refer to the same firm or not. While some matches clearly involve misspellings, other matches are more ambiguous. For example, where "ABC imternational Inc"' is likely the same firm as "ABC international Inc", it is not obvious whether "ABD international Inc" also refers to the same firm. Also, some biases might occur due to unfamiliarity with certain languages. Personally I have some knowledge of the larger European languages, but little knowledge of e.g. Eastern European language. Understanding what parts of a firm name refer to legal terms and what parts refer to name information is not always straightforward. Performing internet searches and using translation services somewhat reduce this issue, but some biases likely persist. Also, in order to label nearly 11.5 thousand candidate matches, in-depth web searches are not feasible for all candidates. Lastly, manual labelling is inherently prone to some human error.

The labelled match status and its relation to the angle θ is presented in Figure 3 for the labelled sample. First, the histogram's pattern is similar to the underlying data presented in Figure 2, indicating the random sampling led to a representative sample. Second, the angle between the firm names is negatively correlated with the share of candidates that are true matches. Third, while the cutoff $\bar{\theta}$ likely excludes many incorrect matches, it surely also excludes some correct matches. For the last bin below the cutoff, incorrect candidate matches outweigh correct candidate matches, but nominally there are still plenty of true matches.

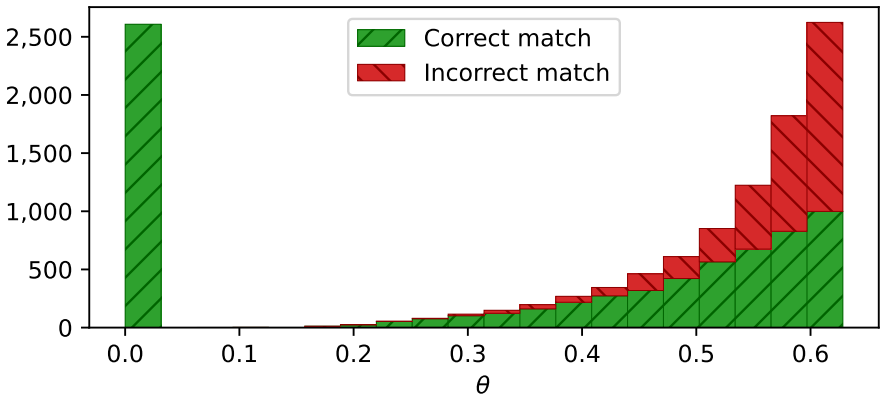

Figure 3: Angles of candidates and actual status.

Note: The distribution of the angles between the TF-IDF vectorized Amadeus firm names and PATSTAT applicant names within the 40% threshold for the manually labelled sample of candidate matches. Their labelled status is indicated by color.

The labelled sample is then used to fit the probit model as proposed by Equation 10, in order to find relationships between information that is available and the true matching status of candidate matches. For the fitting of the model two-thirds of the labelled sample will be used, such that the remaining one third can be used for evaluation of the probit model's performance and to determine the best threshold $\bar{p}$. All unlabelled candidate matches i, j for which $\hat{p}_{ij} > \bar{p}$ are then considered a true match.

The researcher can decide which identifying characteristics go in the vector x of Equation 10. For this application in which firm groups are matched to their patenting behavior, I will not consider location data, like registered country or patent office of application. Instead I will only stick to the information in the names and the data sources. The variables used to predict true matching status are the training source of the vectorizer, i.e. either Amadeus Financials or Amadeus Subsidiaries, whether the cleaned names start with the same letter, whether the first two letters of one name are encapsulated in the candidate match's name, for both names, the angle θ between the names, the number of occurrences of the patent applicant's name in the candidate matches, and interactions between these variables. The interactions allow for more complicated combinations of predictive characteristics.

Additionally two small lists of words are used to flag certain candidates. It turned out that some firm names often are involved in incorrect matches. Taking the words from these firm names and flagging these candidates helps the identification. One list of 13 words contains legal information in Russian, Bulgarian, Ukrainian, Maltese, Polish and Uzbek. This list is used to flag any combination for which either firm name in the candidate match contains any of these words. A second list of two words contains "societe" and "societa", common legal terms in Italian and French. Since these words also occur often in correct matches, candidates will only be flagged if both firm names contain any of these two words. These lists of words improve predictive power significantly. They also highlight the difficulty of capturing all uninformative terms in the preprocessing and vectorization steps. For a complete definition of all variables in the x vector, Appendix A provides an overview.

For the probit model, candidate combinations with $\theta < 0.01$ will not be considered, as these are exact matches. All such candidate matches are considered true matches. The probit model is therefore only used to identify the true status of candidate matches that do not match exactly.

The regression results of the fitted probit model using two-thirds of the labelled sample can be found in Table 9. The model is also fitted separately for the Amadeus Financials and Amadeus Subsidiaries data. While the only purpose of this regression is to accurately predict true matching status out of sample, the coefficients can be interesting. For example, θ strongly negatively correlates with the probability of a combination being a true match, confirming the hypothesis embedded in Figure 3. The first characters in a firm name also hold additional information on top of the similarity established by θ. The flagged words significantly impact the probabilities, in the same magnitude as the first characters of the firm names, but having a negative effect on the probability instead. The

data source variables are not significant, which is furthermore confirmed by the similarity of the coefficients in the last two columns. The two main differences between these regression results are for the number of occurrences of a patent applicants (Count matched) and the interaction of θ and the first letter overlapping.

Table 9: Regression explaining candidate match status.

	Full	Ama Fin	Ama Sub
θ	-2.535***	-1.041	-2.845***
	(0.494)	(0.724)	(0.539)
Same first letter (indicator)	2.499***	2.598***	2.268***
	(0.375)	(0.581)	(0.486)
$\theta \times$ Same first letter	-1.872***	-2.488**	-1.380
	(0.667)	(1.044)	(0.872)
First 2 letters in matched (indicator)	0.849***	0.942***	0.803***
	(0.065)	(0.116)	(0.079)
First 2 letters in trained (indicator)	1.045***	1.034***	1.050***
	(0.064)	(0.109)	(0.079)
$\theta \times$ log(Count matched)	-0.021	-0.114**	0.030
	(0.027)	(0.045)	(0.033)
Amadeus Financials (indicator)	-0.435		
	(0.371)		
$\theta \times$ Amadeus Financials	0.946		
	(0.670)		
Amadeus Financials $\times$ Same first letter (indicator)	-0.157		
	(0.105)		
Flagged words in any (indicator)	-3.238***	-2.923***	-3.646***
	(0.113)	(0.146)	(0.196)
Flagged words (alt) in both (indicator)	-0.913***	-0.905***	-0.967***
	(0.161)	(0.279)	(0.198)
Constant	-0.318	-0.970**	-0.192
	(0.275)	(0.405)	(0.301)
Observations	5,896	2,028	3,868
Pseudo R^2	0.619	0.593	0.635

$^{***}p < 0.01$; $^{**}p < 0.05$; $^{*}p < 0.1$

A probit model is fitted using two-thirds of the labelled sample and only considering candidates for which $\theta > 0.01$, thereby excluding exact matches. The dependent variable is the true matching status, as labelled by the author. The different models are based on all candidate matches or only the ones stemming from either Amadeus source. Standard errors in brackets.

The relationship between the thresholds on $\hat{p}$ ($\bar{p}$) and several evaluation measures are presented in Figure 4. Precision is the share of justified matches in the set of identified matches. It evaluates the precision of the identified matches. Recall is the share of justified matches in the set of all true matches, and therefore evaluates how successful the algorithm is in identifying true matches. Whereas precision in general benefits from a stricter threshold, recall benefits from a more lenient threshold. The F1 score ($F1$) is a metric combining both precision (s) and recall (r), such that $F1 = \frac{2sr}{p+r}$. I determine the

optimal threshold by maximizing the F1 score (see Figure 4). This optimum is found at the threshold $\bar{p} = 0.53$ for the labelled sample that was not used to fit the probit model. Note that the F1 score is nearly flat for a prolonged range of thresholds, roughly from 0.3 to 0.9. For that range precision and recall are close to one another.

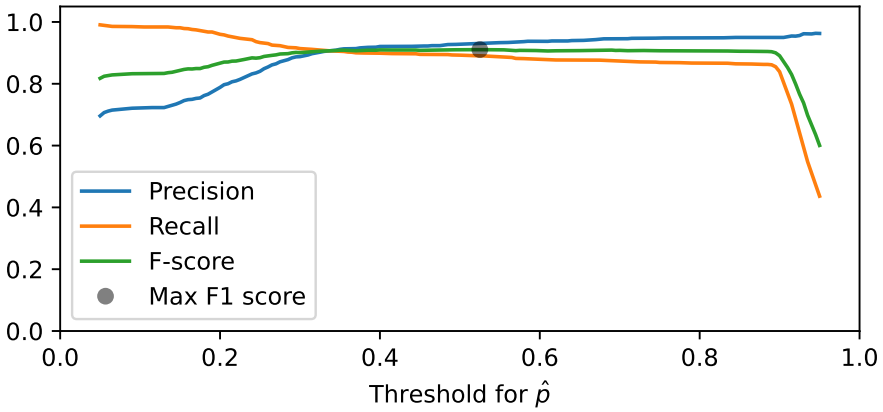

Figure 4: Performance by threshold.

Note: Precision, recall and the F1 score are plotted against their required threshold on $\hat{p}$. The maximum F1 score is also plotted. These scores are calculated for the labelled sample that was not used to train the probit model. It is an out-of-sample performance evaluation.

Alternatively one could determine $\bar{p}$ through other createria. One could also set separate criteria on precision and recall and take the maximum of the corresponding $\bar{p}$s. Lastly, one could take a weighted average of precision and recall in order to give one measure more weight. Depending on the application, the researcher can determine the appropriate criteria.

Additionally these performance indicators can be calculated for the fitted models that use only one of the Amadeus databases. The optimal threshold $\bar{p}$ and the respective errors are presented in Table 10. Note that all indicators are very close to each other across different samples. As these values are so close to one another, it is difficult to judge whether one is outperforming the other or whether it is purely due to chance.

The classification exercise is more intuitively visualized by Figure 5. The fitted probit model is used to produce out-of-sample $\hat{p}$s for the one-third labelled data. These $\hat{p}$s are plotted for both the set of true candidate matches and the set of false candidate matches, together with the threshold $\bar{p}$. One sees that most of the mass of the actual correct candidate matches is above the threshold, while most of the mass of the actual incorrect candidate matches is below the threshold.

The fitted probit model and the threshold $\bar{p}$ can now be used to predict the true matching status of the unlabelled candidate matches. The fitted probit model is used to produce out-of-sample $\hat{p}$s for the unlabelled data. Any candidate match for which $\hat{p} > \bar{p}$ will be considered a true match. Figure 6 shows the distribution of $\hat{p}$s, together with

Table 10: Performance evaluation.

Model	Full	Ama Fin	Ama Sub
Optimal threshold ($\bar{p}$)	0.525	0.695	0.430
Precision	0.931	0.933	0.927
Recall	0.891	0.880	0.903
F1 score	0.911	0.906	0.915
Observations training set	5,896	2,028	3,868
Observations prediction set	2,949	1,085	1,864

The threshold refers to $\bar{p}$ which is determined by maximizig the F1 score. Precision, recall and the F1 score are calculated at that $\bar{p}$. The observations refer to the split in the labelled sample, where the training sample was fed into the probit function and the prediction sample was fed into the fitted probit function to determine $\hat{p}$s and the performance indicators presented here. The columns correspond to the columns in Table 9, fitting the full model or separate models per data source.

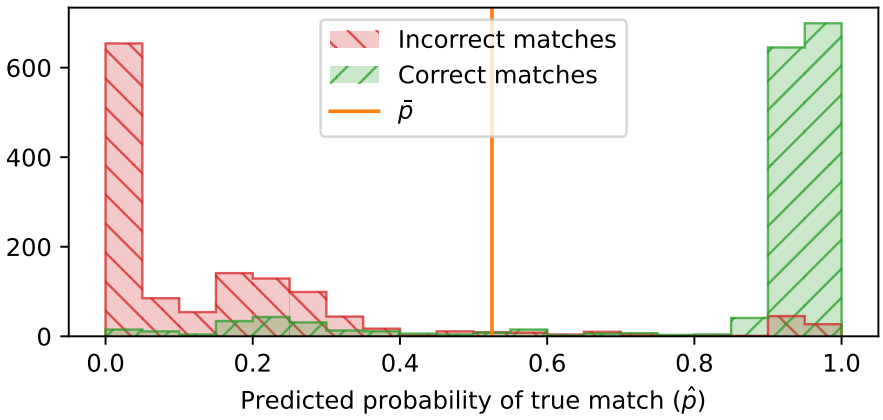

Figure 5: Distribution of candidate matches' $\hat{p}$s by true status.

Note: Distribution of the predicted probability ($\hat{p}$) of candidate matches being a true match by true matching status. Only one third of the labelled sample is presented here, extrapolating the fitted probit model that used the other two-thirds of the labelled sample. The determined threshold is presented by $\bar{p}$. The distributions are overlaying, and not stacked.

the threshold. Out of the 810,118 candidate matches that have a $\theta > 0.01$, 416,603 are determined to be matches, which is 51.4%.

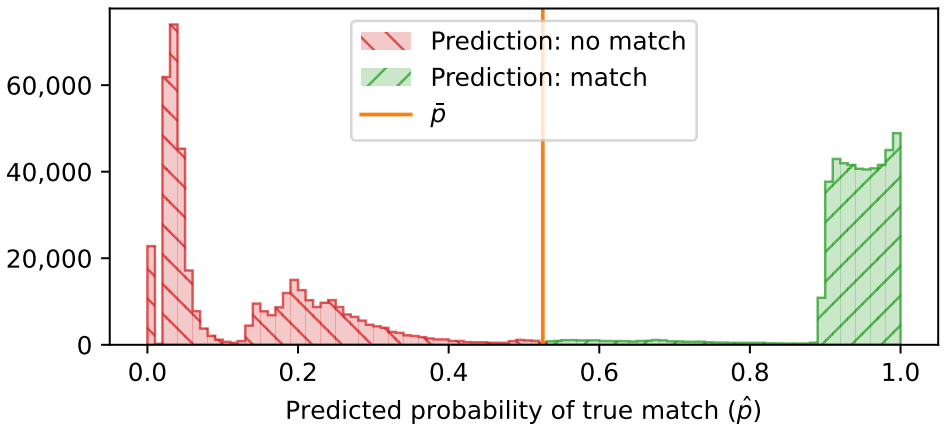

Figure 6: Out-of-sample (unlabelled) prediction and decision.

Note: Distribution of the predicted probabilities ($\hat{p}$) of candidate matches being a true match, and the matching threshold ($\bar{p}$). Only the out-of-sample unlabelled data is considered for which $\theta > 0.01$. Candidates with a smaller θ are exact matches and therefore do not need a prediction of their true status.

Lastly, we can pull all matching results together. Some candidate matches were considered correct due to their angle being small enough ($\theta < 0.01$), some candidates were labelled, and some candidates were predicted to be correct using the supervised machine learning method. When only considering unique cleaned names, 259,273 candidate links have $\theta < 0.01$ and are therefore considered a match.[7] Another 4,854 links (with $\theta > 0.01$) were determined a match in the labelling exercise. And another 457,890 links were predicted to be a match. This brings the number of matches between unique cleaned firm names to 722,017. Fuzzy matching, i.e. matching while $\theta \geq 0.01$, therefore increases the number of matching links with 178%. These figures are summarized in Table 11.

Note that these numbers cannot easily be compared to the cleaning-only merge outcomes from Table 4, as those outcomes are based on matching before dropping duplicated cleaned firm names. In the fuzzy matching algorithm each cleaned firm name only occurs once. To make a comparison, the cleaned names in the links need to be merged back onto the original data in order to count the number of unique firm identifiers and PATSTAT entries involved in these matches. The results are presented in Table 12. The number of matched Amadeus firms increases with 54% and 72% for the Amadeus Financials and Subsidiaries database respectively compared to the cleaning-only merge. And the number of matched PATSTAT entries increases with 81% and 25% respectively. As with the cleaning-only exercise, the fuzzy matching increases the number of links more (178%)

[7] This number is smaller than the number reported in Table 4, as only *unique* cleaned firm names are considered here. Also, a handful of exact matches that were missed by the fuzzy algorithm are added at this point. These are 190 unique name pairs and 844 total name pairs. subsection 4.3 discusses where this discrepancy comes from.

Table 11: Matches by decision rule.

	Total	Amadeus Financials	Amadeus Subsidiaries
Exact match ($\theta < 0.01$)	259,273	106,841	152,432
Labelled	4,854	1,687	3,167
Predicted (out-of-sample) ($\hat{p} > \bar{p}$)	457,890	162,767	295,123
Total	722,017	271,295	450,722

Matching outcomes by decision rule. Firms are either matched because their angle is small enough ($\theta < 0.01$), they were labelled as a true match, or they were determined to be a true match using out-of-sample prediction by the fitted probit model.

than the number of involved firms and applicants. Note that both firms and applicants can occur in multiple links. Step 4 will suggest disambiguation methods to disentangle this multiplicity.

Table 12: Final matches.

		Total	Amadeus Financials	Amadeus Subsidiaries
Links	Firm-Applicant (unique name pairs)	722,017	271,295	450,722
	Firm-Applicant (all name pairs)	1,727,400	599,223	1,128,177
Amadeus	Unique firm names	123,132	58,713	92,831
	Firms	132,811	61,325	99,085
PATSTAT	Unique applicant names	371,432	211,868	287,896
	Entries	713,037	408,008	575,743

Final matching outcomes using the proposed algorithm. These are matches between the Amadeus database firm names and PATSTAT's applicant names. The columns split up the matches by source. The first two rows provide information on the number of links found. Both the links between unique firm names as well as links between raw entries are considered. The remaining rows present the number of firms or applicants that are matched, either through the number of unique (cleaned) names or the number of firm IDs or PATSTAT entries. The columns do not always add up to the value in the Totals column, as a single PATSTAT entry can match to firm names in both Amadeus data sources.

The matching outcomes can further be studied by comparing matched and unmatched firms in either database. For the firm databases, Amadeus Financials and Amadeus Subsidiaries, the number and shares of matched firms by country are presented in Figure 7. The unmatched category contains all firms not matched by the algorithm. The size of the unmatched group is not only determined by the success rate of the matching algorithm, but also by the patenting decision of firms. If firms do not patent, a group of unmatched firms exists inherently. As patenting rates might differ between countries, comparing the shares of matched firms between countries to determine the performance of the matching algorithm is also not watertight. Nevertheless, the country comparison shows both the dominance of the countries in the Amadeus databases, as well as the dominance in the

match rate of the algorithm. One noticeable observation is that the matching rates are higher in the Amadeus Financials database, likely because these firms are larger or higher up in the ownership hierarchy. Further, the algorithm seems to work well for different languages. For example, French and German firms are more likely to be matched with patent applicants than British firms. The only country that seems to be underrepresented in the matches is Ukraine. It is plausible that part of the underrepresentation can be attributed to the algorithm's performance.

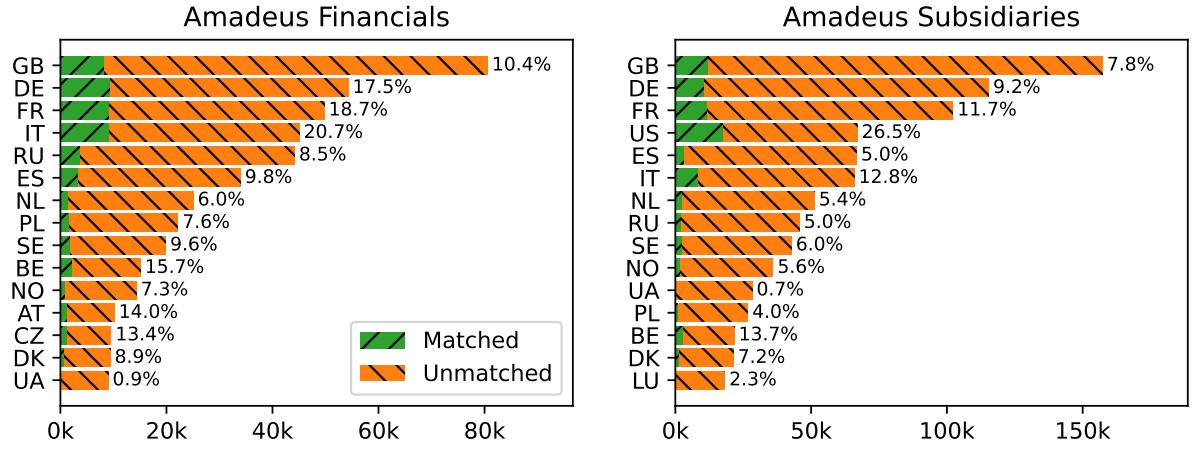

Figure 7: Match coverage in the Amadeus databases by country.

Note: The number of firms in the Amadeus databases (horizontal axis) and the share of them being matched to a patent applicant (colors and percentage) per country. The figure presents the top 15 countries per Amadeus database. Each firm is represented once with its most recent data.

Similarly firms can be grouped by their industries, as shown in Figure 8. Also here differences in matching rates cannot be attributed fully to the algorithm's performace, as industries differ in their innovativeness and patenting behavior. Again the matching rate is higher for the Amadeus Financials database. Additionally, the difference in matching rates is noticeable as it can differ with a factor ten. For example finance and insurance (code 52) and real estate, rental and leasing (53) have a low matching rate, in accordance with a lower expected patenting rate. On the other hand, manufacturing firms (31-33) have a higher matching rate, likely due to a higher patenting rate. Of these food and textile manufacturing (31) matches about 13% of the time, the broad industry 32, which includes chemicals and pharmaceuticals, matches roughly 30% of the time, and metals, machinery, electronics and transport (33) find a match for 39% of firms in the Amadeus Financials database.

The matching coverage can also be studied from the patenting side. Figure 9 presents the number of patent applications that have at least one applicant that was matched to a firm in either of the Amadeus databases. While the match rate is significant over the 70 year period, with a 5-year rolling average mostly above 10%, the peak in the 2000s is likely attributed to the Amadeus data being available mostly as of 2008. Of all patent applications since 1950, 18.1% can be linked to a firm in the Amadeus databases.

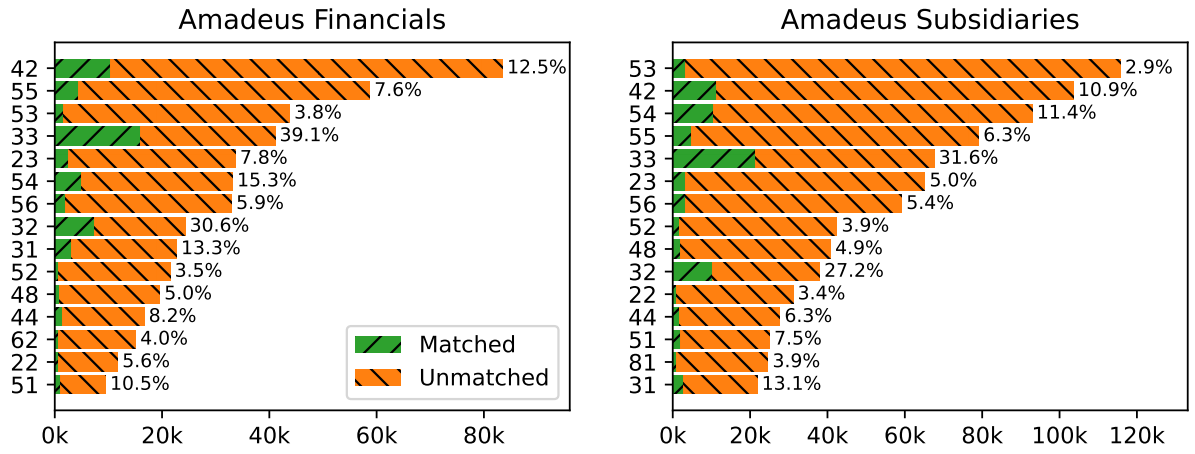

Figure 8: Match coverage in the Amadeus databases by industry.

Note: The number of firms in the Amadeus databases (horizontal axis) and the share of them being matched to a patent applicant (colors and percentage) per industry (2-digit NAICS code). The figure presents the top 15 industries per Amadeus database. Each firm is represented once with its most recent data.

This is a rather high share, since the Amadeus databases do not cover all existing firms. Furthermore, some patents are applied for by individuals or governments, making them impossible to be matched to private firms.

The right panel of Figure 9 shows the coverage per patent office as of 2000. Noticeable is that many applications occur at the Chinese patent office and that these applications are difficult to link to firms in the Amadeus databases. This likely has two main reasons, namely (1) the applicants are not included in a firm database like Amadeus that mostly focuses on firms in Europe and the US, and (2) firm names are more difficult to link across different alphabets. At other patent offices matching rates can be significant, like in Germany (45%), Spain (46%), Great Brittain (32%), or at the European Patent Office (EP) (46%).

Step 4: Disambiguation. While the decision rules above have set the acceptable amount of error in these links, even the correct links can still be ambiguous. This ambiguity is inherent to name matching exercises, as one name can refer to multiple firms. The firm name cleaning in Step 1 has exacerbated this issue by potentially mapping different names into similar cleaned names. But also uncleaned firm names can be ambiguous, especially when the data is imprecise. Such ambiguity in the matching outcome makes it difficult to do follow-up analysis, as it causes issues with double counting. This step illustrates ways to rid the matchings of ambiguous linkages.

As discussed earlier, the names in the PATSTAT data are especially problematic. Not only do they contain misspellings, it is often unclear to which exact firm the name refers. Names are often abbreviated or they leave out parts of the name that are necessary for identification. For example, the name "ABC Technology Inc" might very well refer to the firm "ABC Technology Germany Incorporated". To provide a real example, there are

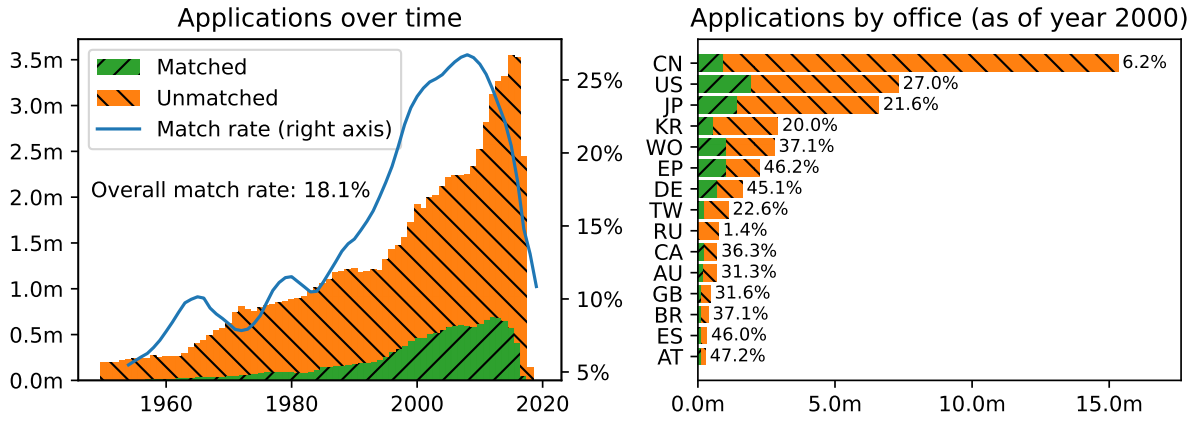

Figure 9: Match coverage in the PATSTAT database.

Note: The number of patent applications in the PATSTAT database that have at least one applicant that was matched to a firm in the Amadeus databases. The left figure presents all data by earliest filing year as of 1950. The right figure shows the top 15 patent offices by patent applications as of 2010. Percentages refer to the share of patents that is matched by the algorithm.

75 unique firms in the Amadeus Financial database that contain the word 'Bayer', all likely referring to some firm that is involved with the Bayer firm group. In the Amadeus Subsidiaries data there are 304 unique firm names and there are 2,594 unique patent applicant names with the word 'Bayer'.

The firms in the Amadeus databases can be uniquely identified by a firm identifier. The PATSTAT applicants have no consistent identifiers and can therefore only be told apart by differences in their names. It is also possible that one applicant shows up in the data with different applicant identifiers and even with different name spellings. Multiple applicants should therefore be allowed to be matched to any one firm in the Amadeus database, but one applicant should never match to multiple Amadeus firms. I propose three methods to achieve such an outcome.

First, one can for each patent applicant choose the link with the highest $\hat{p}$ from Step 3 (see Equation 10). Any ties can be broken by firm information available in the Amadeus databases. I opt for breaking ties by selecting the firm with the highest number of employees, largest total assets, or largest revenue, in that order. If all are equal the tie is broken randomly. When identifying ties one should be aware of computational imprecision. Calculations with float number formats often lead to some imprecision, especially when opting to reduce memory usage by reducing float precision. Allowing for a small degree of error in testing for ties is a solution. I suggest to consider all links of a patent applicant with a $\hat{p}$ within 0.01 of the max $\hat{p}$ a tie.

As this disambiguation seeks to match each patent applicant to one individual firm, it is appropriate to only consider unconsolidated firm data. This does reduce the number of links. The results of this disambiguation can be found in the Firm column of Table 13. Compared to the no-disambiguation case, the number of links is nearly halved. A small share of applicants no longer finds any match as their previously matched firm has no

unconsolidated data. And as expected, the number of involved firms in the remaining matches is reduced more. The resulting links involve about a quarter fewer firms.

Table 13: Disambiguation results.

		No disamb.	Firm	Community	Firm in comm.
Total	Links (firm-applicant)	1,727,400	894,049	1,727,337	523,686
	Links (community-applicant)	-	-	713,037	125,630
	Firms	132,811	102,047	132,811	19,578
	Applicants	713,037	702,657	713,037	125,630
	Communities	-	-	90,999	9,257
Ama Fin	Links (firm-applicant)	599,223	332,056	599,208	206,475
	Links (community-applicant)	-	-	408,008	112,086
	Firms	61,325	47,712	61,325	13,309
	Applicants	408,008	332,056	408,008	112,086
	Communities	-	-	49,497	9,207
Ama Sub	Links (firm-applicant)	1,128,177	561,993	1,128,129	317,211
	Links (community-applicant)	-	-	575,743	93,627
	Firms	99,085	75,306	99,085	12,383
	Applicants	575,743	561,993	575,743	93,627
	Communities	-	-	68,311	5,113

The first two columns indicate the sample and the statistic, respectively. The numerical columns can be used to compare different disambiguation techniques. The first of these columns is the baseline, without any disambiguation. The second links each patent applicant to only one unconsolidated firm, based on the accuracy of the link ($\hat{p}$) and the size of the firm. The third column presents the results from community detection. The last column selects the largest firm with consolidated statements within each community. The blocks of rows refer to the underlying sample, which either are all firms (Total), Amadeus Financials firms only (Ama Fin), or Amadeus Subsidiaries firms only (Ama Sub). Mind that firms and applicants can occur in matches of both subsamples.

Second, the multiplicity of links can be used to identify firm groups. As links are established based on name similarity, the multiplicity of links also groups firms with similar names. In the example above, the hundreds of firms containing 'Bayer' likely are part of the Bayer firm group. If different Amadeus firms link to the same patent applicant, they are therefore likely part of the same group, especially if they frequently link to the same patent applicants together. The co-occurences in links to patent applicants can be used to identify groups using community detection algorithms. Figure 10 illustrates the idea of community detection through the matching links. The key concept is co-occurence. Firms A and B co-occur in their links with patent applicants 1 and 2. They are therefore considered one community (indicated by the dashed rectangles). Firm C only co-occurs with firm B once and is therefore not included in the same community. The link C-2 is therefore broken (dotted line). Firms D and E do not co-occur with other firms and therefore form their own community.

For the community detection the Louvain Community Detection algorithm is used (see Blondel et al., 2008). As community detection works with co-occurence, singletons have to be considered separately. Singletons are defined as Amadeus firms that are only

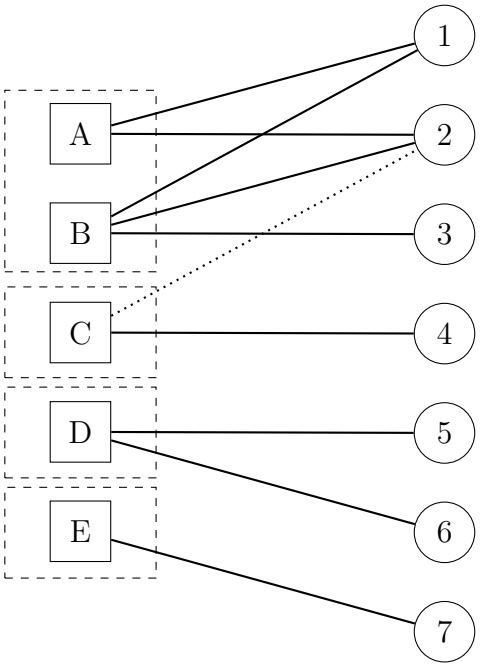

Figure 10: Community detection.

Note: An example of disambiguation using community detection. The squares and letters represent unique firms, circles and numbers represent patent applicants and lines indicate the established links from the fuzzy matching algorithm. Disambiguation is achieved when each applicant is linked to only one unit. Community detection uses co-occurence to establish disjoint sets of firms, the communities (indicated with dashed rectangles). E-7 is the simplest link as it is uncontested. Also D-5,6 is uncontested. Firms A and B co-occur in their links with applicants 1 and 2 and are therefore grouped together. C also co-occurs in the link with applicant 2, but it lacks sufficient co-occurence with A and B to belong to the same community. The link C-2 is therefore broken (dotted line).

linked to one patent applicant. If firms are also involved in co-occurences, they are not considered singletons. All co-occurences are then represented in an edge list and the algorithm starts moving nodes into communities until it has maximized modularity. The resulting communities then serve as firm groups that should be considered as the unit that is linked to the patent applicant. Follow-up analysis should therefore consider patenting on this firm group level.

The results of this disambiguation can be found in the Community column in Table 13. As communities are a set of firms, it is still possible to track the firm-applicant links. Noticeable is that several of these links are broken as some patent applicants are linked to multiple communities. In the example in Figure 10 this would be the case for applicant 2. This multiplicity is resolved by linking the applicant to the community with most firms in it. In the example the link C-2 is therefore broken.

All firms and all applicants remain represented in the community-level links. The number of community-level links is 41% of the number of firm-level links. And the nearly 133 thousand firms are grouped into nearly 91 thousand communities. Compared to the firm-level disambiguation the researcher can analyze 91 thousand communities instead of 102 thousand firms. Communities are larger than single firms. This also results in communities linking to more patent applicants.

Third, the patent applicants can be linked to the dominant firm within each community. The Amadeus data contains both consolidated as well as unconsolidated data. By identifying the largest firm with consolidated data within a community, each patent applicant could be linked to the financial data of the consolidated firm group. A benefit over using the separate firms in the community is that the consolidated reporting also contains the data from subsidiaries with different names. It therefore might be more accurate data on the firm group than the self-constructed communities. An accompanying drawback is that firms within the firm group that have different names are not matched to their patent applicant in the patent data. The number of patent applications of the firm group is therefore underestimated by the data. Another issue is that multiple firms in the community might have consolidated financial statements. Determining which firm to consider as the community head requires adding another step to the data preparation exercise. I propose to take the firm with the largest consolidated figures, measured by total assets, employees or revenue. That firm is most likely to encapsulate the other firms in the ownership structure.

This last disambiguation exercise leads to the links presented in the last column of Table 13. As not all communities contain a firm with consolidated financial statements, about 90% of communities are removed from the data. It also leads to fewer firms (-85%) and fewer patent applicants (-82%). The remaining communities do contain more firms and are linked to more applicants per community.

Depending on the research question the researcher should decide the unit of analysis and the accompanying algorithmic choices, keeping in mind the respective strengths and shortcomings. Linking patent applicants to one single firm risks incorrect assignment. And when correctly assigned, the data might not allow to answer some questions. When

for example testing whether firms apply for more patents in the space of electrification in response to more stringent environmental policies, firm-level links might miss crucial strategic behavior within a firm group. If a manufacturing subsidiary in the group is exposed to the new policies, the firm group might respond with additional patenting at the innovation subsidiary. Reversely, linking patent applicants to firm groups has a potentially diluting effect. Grouping many firms might obscure the link between the new policies and additional patenting when only a few firms in the group are affected by the new policies. Any estimates of the effect will be biased towards zero. It also makes it more difficult to isolate the effect of the policy, as more events will play a role in the patenting decision of a large firm group.

4 Discussion

This section provides some context to the methodological choices made, it will share computational considerations that might be valuable to researchers facing similar matching problems, and it suggests extensions.

4.1 Alternative matching methods

One could compare text strings using several methods. I will just highlight five common methods. This list is not exhaustive and some of these methods are also used in the discussed literature in Section 1. First, the Levenshtein distance counts the number of characters that need to be changed or added to one string in order to produce the other string. It is therefore also known as the edit distance. Second, one could simply measure the longest common substring across two strings. Third, the Jaro-Winkler distance, or similarity, combines the first two measures. Fourth, other vectorizers can be used to represent the text's features, although some form of the TF-IDF vectorizer is often used. Fifth, pre-determined mappings can be used to link text to features. Think of pre-trained text models or Soundex, which links the English pronunciation of words to features.

In the proposed algorithm I perform vectorization and similarity scoring before employing a supervised machine learning method. Why not run the machine learning algorithm on the raw data directly? This has to do with the share of true matches in a random sample. Consider merging the 521,877 unique raw firm names from the Amadeus Financials database on the 8,311,667 unique applicant names in the PATSTAT database. Imagine an optimistic scenario in which all patent applicants can be linked to one firm name. Having no pre-selection in place, drawing random combinations and manually labelling them as true or false will result in few true matches being observed. Out of the 4,337,667,838,959 combinations only 8,311,667 are true matches. The odds of a random combination being a true match is 1 in 521,877. In order to predict the true matching status using Equation 10, a training set should be provided that contains a fair amount of both true and false matches. If only a few true values are provided, the probit model cannot be fitted properly as there will be no clear relationship between the identifying characteristics x

and outcome q. The predictive power of such a trained model will be poor. Say one wants the training data to contain 100 true matches, one needs to manually check more than 52 million combinations. This is a near impossible task, as labelling 11,5 thousand candidates already takes a full day of work. And remember this is in the optimistic situation in which each patent applicant can be matched to a firm, a situation that is highly unlikely. If 10% of applicants can be matched to a firm, one needs to manually check more than half a billion combinations. Instead using the similarity scoring to provide candidates, greatly reduces manual labor.

4.2 Computational considerations

Matching many firm names can be computationally intensive and memory heavy. There are several clever implementations that reduce memory usage and speed up calculations, such that merging 1.6 million firms to 8.3 million patent applicants becomes feasible with a basic laptop.[8] For the implementation Python is used, but similar solutions are likely available in other programming languages.

Several measures can be taken to reduce memory usage. First, the data type can be altered to a type with a smaller memory footprint. Whereas most float variables by default are stored in a 64-bit format, opting for 32 bits halves the memory footprint while upholding enough precision for this task. Reduced precision may affect outcomes slightly, translating into more rounding errors. For example, multiplying $\frac{1}{3}$ with itself results in 0.1111111111111111 with `numpy`'s float64 format and in 0.11111112 with `numpy`'s float32 format. While the difference is small, such rounding errors can accumulate. It will for instance cause some dot products in Equation 7 to evaluate to $\cos(\theta) > 1$, for which there exists no such θ. Such rounding errors should be corrected for, e.g. by calculating the angle as $\theta = \arccos(\min(\vec{\hat{a}} \cdot \vec{\hat{b}}, 1))$. It is also possible that rounding causes an angle to cross the $\bar{\theta}$ threshold. There is no reason to believe such cases result in a problematic bias in the resulting candidate matches.

Second, working with sparse matrices reduces memory usage immensely. Vectorization creates a 22,460 length vector for each firm name when training the vectorizer on the Amadeus Financials' cleaned NAME variable. These vectors contain on average 19.72 non-zero values. All other positions are filled with zeros. Instead of storing all these zeros in memory, one can also only store the non-zeros and their locations, implicitly assuming all other locations contain zeros. The more sparse matrices are, the more memory will be saved storing matrices in a sparse format. Python's `scipy` package has sparse matrix formats. Besides memory efficiency, calculations with sparse matrices are also faster. The dot product between vectors can ignore all zeros, drastically reducing the number of calculations.

Third, calculating more than 3 trillion dot products at once is likely causing memory errors. Whereas the vectorized firm names hold many zeros, their dot products are

[8] The matching is performed by the author on a laptop with 8GB memory and a 1.6GHz CPU with 4 cores.

surprisingly often non-zero. In roughly 15% of the cases the dot products are non-zero. This means that calculating all angles at once requires storing more than 600 million values and their locations. On a normal computer this is not feasible. Instead, one could partition one sparse matrix from Equation 9 and perform the angle search for each part in succession. For example, one could take a block of 1,000 rows out of $\hat{A}$, calculate D using all of $\hat{B}$, determine the cosine values that pass the threshold, store these, and move on to the next block of 1,000 rows. A useful tool to achieve the partitioning in Python is the built-in generator function. Such a function only performs its action when called upon. The above partitioning can then be performed with a small memory footprint. Instead of splitting $\hat{A}$ up in parts and looping over these parts, one can pass the generator function to the loop. The blocks of 1,000 rows will then only be created when requested by the loop. At any point in time there will only be one block of 1,000 rows in memory.

Fourth, one could perform multi-processing using the `multiprocessing` package in Python. The benefit is that multiple CPU cores can be used to perform calculations in parallel, speeding up the process. The drawbacks are that more memory is used as all these parallel processes require memory. The number of rows of $\hat{A}$, and likely $\hat{B}$, fed to each iteration should be downwards adjusted accordingly. Also, using all CPU cores will make it difficult to perform other tasks on the computer while the code is running. It will be difficult to run the matching algorithm as a background process. One could therefore actively choose not to parallelize the matching algorithm and have the process run in the background.

4.3 Extensions

There are several extensions and improvements possible from here onwards. There exist algorithms and dictionaries that recognize firm names even when they are misspelled. Such recognitions likely only work for the more common, older and larger firms, but they would improve the cleaning process. Another method would be to use online search engines to point to the parent firm's website, as suggested by Bena et al. (2019).

Further, two more cleaning steps can be taken that likely improve the matching outcomes. First, many incorrect matching candidates contained short identifying information and long generic information. Consider "ABC international technologies Corp" for which "ABC" is the most important identifying information and the rest is rather generic. Abbreviating terms like "international" and "technologies", just as I did for legal terms, might improve the matching algorithm. Too often such firm names were incorrectly matched to other firm names with equal generic terms. Alternatively, n-grams at the beginning of the character string could be upweighted as most identifying information is at the start of the firm name, with some exceptions. Besides n-grams, one could also partition the string by word and pass both the n-grams and the words to the vectorizer. This will additionally downweight generic words, but only in case of correct and consistent spelling.

Second, n-grams from (abbreviated) legal terms or other generic information should be kept separately from the same n-grams stemming from more identifying information.

From Table 5, 6 and 7 we learned that the most common 3-gram is "ltd", which most likely stems from the abbreviation of the uninformative legal term "Limited". If indeed the 3-gram stems from "limited" it is desirable that the IDF term in the vectorizer downweights it. But when the 3-gram actually stems from identifying information, it should not be downweighted. For example, the firm name "Bolt delivery corp" has its "ltd" 3-gram disproportionately downweighted in the current algorithm.

Working with n-grams also results in a few incorrect vectorized representations. First, short firm names might not have an n-gram. For example firm name "AB" has no 3-gram, resulting in an empty vector and no calculated angles with firm names. It will therefore not be matched to any firm name, including firm names that are spelled the same. This issues is solved in the presented application by adding matches between short firm names from an exact merge. Second, n-grams that occur only in the matching data and not in the training data will be omitted, leading to optimistic matching results for firm names in the matching data with such n-grams. For example consider the firm name "ABC Inc" in the matching data. If the training data does not contain the n-gram "ABC", vectorization will lead to the following hypothetical non-zero vector elements (after cleaning)

$$\begin{array}{c} \\ \text{ABC Inc} \end{array} \begin{array}{c} \begin{array}{ccc} \text{bci} & \text{cin} & \text{inc} \end{array} \\ \begin{bmatrix} 1.3 & 1.5 & 0.2 \end{bmatrix} \end{array} \tag{11}$$

where the n-gram "ABC" does not exist and is hence implicitly assumed to be zero. Angles will therefore be incorrectly calculated. The angle between "ABC Inc" and "BC Inc" will be zero, which is overly optimistic. The exclusion of a positive element also avoids downweighting of each element in the vector normalization (see Equation 7). It thereby further inflates the estimated $\cos(\theta)$ and leads to a lower estimated θ.

This can be illustrated mathematically. Consider a vectorized firm name $\vec{a_1}$ (with only non-negative elements) and a vector $\vec{a_2}$ which is identical except that one positive element is set to zero. It must be that $\|\vec{a_1}\|_2 > \|\vec{a_2}\|_2$. Now consider the angle with another vector that only has non-negative elements $\vec{b}$. It must be that $\cos(\theta_1) = \frac{\vec{a_1} \cdot \vec{b}}{\|\vec{a_1}\|_2 \|\vec{b}\|_2} \leq \frac{\vec{a_2} \cdot \vec{b}}{\|\vec{a_2}\|_2 \|\vec{b}\|_2} = \cos(\theta_2)$ where the equality occurs only if $\vec{a_1}$ and $\vec{b}$ are perpendicular. Therefore the omission of an n-gram from one firm name leads to lower angles between that firm name and any other firm name that is not perpendicular to it.

An inherent risk to working with firm names is their dynamics. Firms can change their names or merge, making the firm name an unstable identifier over time. Such changes could be taken into account when one uses information on mergers, acquisitions and name changes. Firm name matches would then only be sought within the time periods between name changes. Finding such information for all considered firms might be difficult and tedious.

5 Conclusion

This chapter examines the problem of merging databases by firm name, a common problem when doing analysis with firm-level data. The problem arises from the lack of common firm identifiers across databases and is worsened by the absence of a common firm name spelling. In this chapter I propose a fuzzy name matching algorithm and I apply it to the problem of merging the Amadeus databases to the PATSTAT database. For this application, the algorithm increases the number of matched firms with 116% and 160% for the Amadeus Financials and Subsidiaries databases, respectively, compared to an exact match on raw firm names. It increases the number of matched applicants by 419% and 454% for the Amadeus Financials and Subsidiaries databases, respectively. It therefore identifies significantly more innovative firms, and it vastly increases the measured innovation intensity of these firms. Instead of 2.6% for an exact match, 18.1% of all patent applications since 1950 are linked to Amadeus firms using the proposed algorithm.

The proposed algorithm consists of four steps. Step 1 thoroughly cleans the firm names of each database, taking into account casing, accents, special characters and legal terms (like Inc and Corp). Step 2 tests for similarity between all firm names using high-performance fuzzy string matching based on vectorization and the Cosine Similarity. Step 3 selects candidate matches and subsequently filters out incorrect matches using a supervised machine learning method. Step 4 disambiguates the links in the matching outcomes and guarantees that each patent applicant is linked to only one firm or firm group.

The benefits of the proposed solution, besides greatly improving merging performance, is that it is relatively easy to run even with limited memory and CPU. In the application of merging the Amadeus databases to the PATSTAT database, roughly 1.67 million firm names were compared to 8.31 million patent applicant names (or more than 13.8 trillion combinations) in about five days on a normal laptop with 8GB memory, just as a background process.

References

Balsmeier, B., Chavosh, A., Li, G.-C., Fierro, G., Johnson, K., Kaulagi, A., O'Reagan, D., Yeh, B., & Fleming, L. (2015). *Automated disambiguation of US patent grants and applications* (tech. rep.).

Bena, J., Ferreira, M. A., Matos, P., & Pires, P. (2017). Are foreign investors locusts? The long-term effects of foreign institutional ownership. *Journal of Financial Economics*, *126*(1), 122–146.

Bena, J., Ferreira, M. A., Matos, P., & Pires, P. (2019, May). *UVA Darden Global Corporate Patent Dataset: Construction and features* (tech. rep.).

Berg, C. v. d. (2017). *Super fast string matching in python*. Retrieved December 13, 2019, from https://bergvca.github.io/2017/10/14/super-fast-string-matching.html

Blondel, V. D., Guillaume, J.-L., Lambiotte, R., & Lefebvre, E. (2008). Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, *2008*(10), P10008.

Callaert, J. (2017, May). Harmonizing patentees [WIPO Name Harmonization Workshop].

Cameron, A. C., & Trivedi, P. K. (2005). *Microeconometrics: Methods and applications*. Cambridge university press.

De Rassenfosse, G., Kozak, J., & Seliger, F. (2019). Geocoding of worldwide patent data. *Scientific data*, *6*(1), 260.

Dernis, H. (n.d.). OECD HAN database: A solution on the harmonisation of applicant names for patent statistics.

Dugoua, E., Dumas, M., & Noailly, J. (2022). Text as data in environmental economics and policy. *Review of Environmental Economics and Policy*, *16*(2), 346–356.

Gentzkow, M., Kelly, B., & Taddy, M. (2019). Text as data. *Journal of Economic Literature*, *57*(3), 535–74.

GLEIF. (2021). *Iso 20275: Entity legal forms code list*. Retrieved December 27, 2021, from https://www.gleif.org/en/about-lei/code-lists/iso-20275-entity-legal-forms-code-list

He, Z.-L., Tong, T. W., Zhang, Y., & He, W. (2018). Constructing a chinese patent database of listed firms in china: Descriptions, lessons, and insights. *Journal of Economics & Management Strategy*, *27*(3), 579–606.

Kelly, B., Papanikolaou, D., Seru, A., & Taddy, M. (2021). Measuring technological innovation over the long run. *American Economic Review: Insights*, *3*(3), 303–20.

Morrison, G., Riccaboni, M., & Pammolli, F. (2017). Disambiguation of patent inventors and assignees using high-resolution geolocation data. *Scientific data*, *4*(1), 1–21.

Nijhuis, M. (2022). *Company name matching*. Retrieved May 1, 2023, from https://medium.com/dnb-data-science-hub/company-name-matching-6a6330710334

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Peruzzi, M., Zachmann, G., & Veugelers, R. (2014). *Remerge: Regression-based record linkage with an application to patstat* (tech. rep. No. 2014/10iii). Bruegel Working Paper.

Pezzoni, M., Lissoni, F., & Tarasconi, G. (2014). How to kill inventors: Testing the massacrator© algorithm for inventor disambiguation. *Scientometrics*, *101*, 477–504.

Trajtenberg, M., Shiff, G., & Melamed, R. (2006, September). *The "names game": Harnessing inventors' patent data for economic research* (Working Paper No. 12479). National Bureau of Economic Research.

Appendices

Appendix A Probit model variables

The variables used in the probit regression of the application in this chapter are defined in Table 14.

Table 14: Descriptions of the variables in the probit model.

Variable	Description
θ	The angle calculated by the cosine similarity on the TF-IDF vectorized cleaned firm names.
Same first letter (indicator)	Indicator whether the cleaned firm names in a candidate match share the same first letter.
First 2 letters in matched (indicator)	Indicator for whether the first two letters of the cleaned firm name of the IDF training source (Amadeus Financials or Amadeus Subsidiaries) occur anywhere in the cleaned firm name of the matching source (PATSTAT). The two letters have to occur in sequence.
First 2 letters in trained (indicator)	Indicator for whether the first two letters of the cleaned firm name of the matching source (PATSTAT) occur anywhere in the cleaned firm name of the IDF training source (Amadeus Financials or Amadeus Subsidiaries). The two letters have to occur in sequence.
Count matched	Number of occurences of the cleaned firm name from the matching source (PATSTAT) in all candidate matches. For example it takes the value 5 if the PATSTAT firm name in the respective candidate match occurs in 4 other candidate matches' PATSTAT name. It does not count appearances in the other (training) data source.
Amadeus Financials (indicator)	Indicator for whether the training source is Amadeus Financials.
Flagged words in any (indicator)	Indicator whether a set of flagged words occurs in any of the two firm names in the respective candidate match. The set of 13 words contains 'zakrytoe', 'aktsionernoe', 'obshchestvo', 'otkrytoe', 'zaklady', 'wlokien', 'chemicznych', 'przedsiebiorstwo', 'handlowe', 'nauchno', 'proizvodstvennaya', 'proizvodstvennoe', and 'predpriyatie'.
Flagged words (alt) in both (indicator)	Indicator whether an alternative set of flagged words occurs in both firm names in the respective candidate match. The set of 2 words contains 'societe' and 'societa'.

Appendix B Further results

Table 15 describes the most common 3-grams in the PATSTAT database when considering the 3-grams produced when training the vectorizer on the Amadeus Subsidiaries data. It complements the vectorization results in the main text (Table 5-7).

Figure 11 presents the matching status of the firms in the Amadeus databases by firm size. Here size is directly derived from the number of employees. Due to many firms with

Table 15: Most common n-grams in the PATSTAT database when training on Amadeus Subsidiaries data.

#	3-gram	Counts	#	3-gram	Counts	#	3-gram	Counts
1	ltd	1,619,193	11	log	523,790	21	ter	394,753
2	col	1,328,115	12	hno	515,248	22	che	374,078
3	olt	1,296,039	13	nol	500,910	23	ong	371,880
4	ing	785,604	14	ion	472,445	24	str	371,680
5	ech	678,661	15	ogy	449,918	25	inc	371,561
6	tec	666,945	16	han	441,486	26	men	370,161
7	ang	630,298	17	yco	438,267	27	eng	350,705
8	chn	577,391	18	ian	427,568	28	tio	349,589
9	olo	544,629	19	and	414,672	29	ica	345,845
10	ent	530,610	20	ine	406,556	30	ele	345,747

These are the number of occurrences of 3-grams in the PATSTAT database that also occur in the 1,136,376 firm names in the cleaned unique Amadeus Subsidiaries database. The PATSTAT variable person_name is used here. Counts refer to the number of applicant names containing the respective 3-gram and # refers to the rank. The applicant names are thoroughly cleaned before vectorization.

few employees the figure is somewhat difficult to read. This shows that financial data is inherently tricky to work with, as such statistics raise the question whether firms indeed have few employees or whether employees are employed by another entity within the firm group. Although reassuringly firms with no employees see a significantly lower matching rate. Maybe surprisingly the match rate does not increase with firm size, meaning that also smaller firms are actively patenting and are being matched by the algorithm.

Appendix C Code

Code is written in Python using open-source libraries. The code is available on the author's Github page (https://github.com/Leonbremer/namematch).

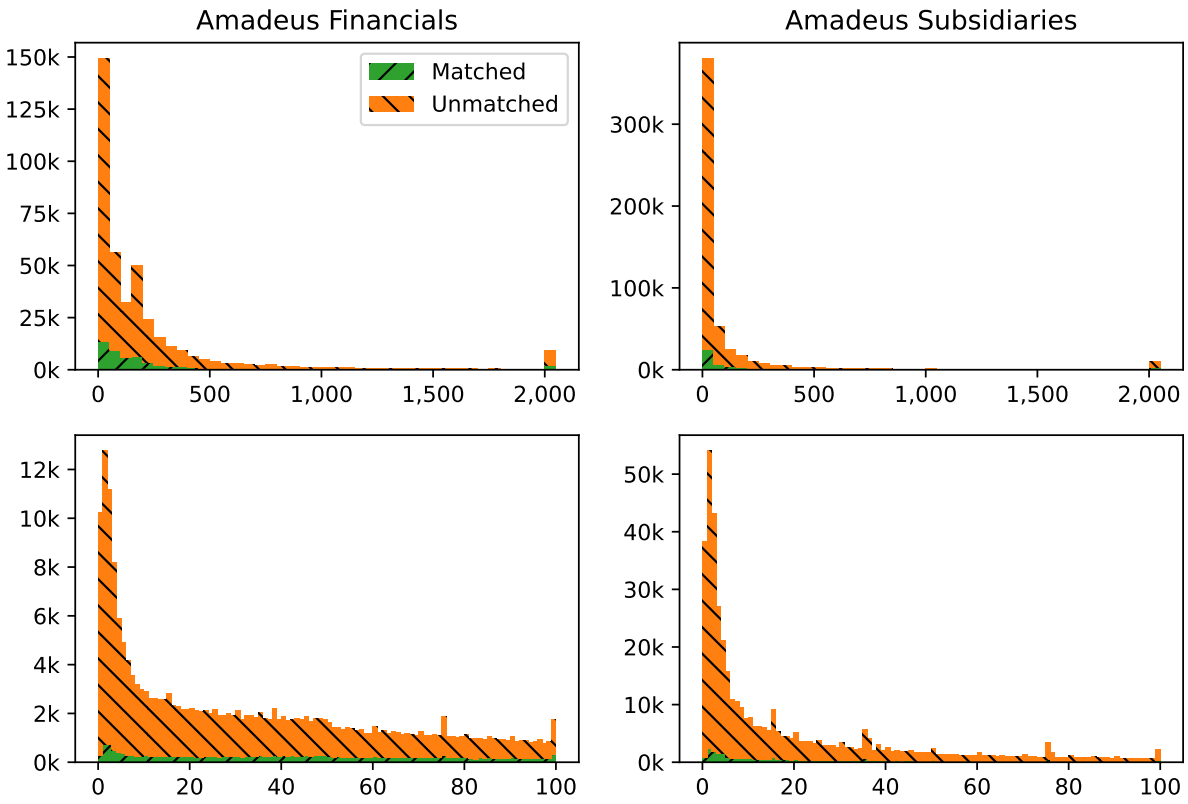

Figure 11: Match coverage in the Amadeus databases by number of employees.

Note: The number of firms in the Amadeus databases (horizontal axis) and the share of them being matched to a patent applicant (colors) per size category. Size is measured by the number of employees. The figures on the top row show all size categories. The last bin contains any value larger than 2,000. The figures on the bottom row zoom in on firms with at most 100 employees. Each firm is represented once with its most recent data.