CENTRE FOR TRANSFORMATIVE INNOVATION

Working Paper Series

Representing Text as Abstract Images Enables an Image Classifier To Perform Text Classification for Name Disambiguation

Stephen Petrie
T'Mir Julius

Working Paper No 5/18

November 2018

Representing text as abstract images enables an image classifier to perform text classification for name disambiguation

Stephen M. Petrie
Swinburne University of Technology
Hawthorn, Victoria, Australia
spetrie@swin.edu.au

T'Mir D. Julius
The University of Melbourne
Parkville, Victoria, Australia

Abstract

Patent data are often used to study the process of innovation and research, but patent databases lack unique identifiers for individual inventors, making it difficult to study innovation processes at the individual level. Here we introduce an algorithm that performs highly accurate disambiguation of inventor names in US patent data, producing a set of unique identifiers with an associated precision of 99.41% and recall of 98.76%. The algorithm includes a novel method for converting text-based record data into abstract image representations — i.e. text from any given pairwise comparison between two inventor records is converted into a 2D colour image. We then train an image classification neural network to discriminate between pairwise comparison images that correspond to either of two classes: matched records (same inventor) or non-matched records (different inventors). String similarities or differences between each pair of records produce spatial and colour features in the associated abstract image representation, and the neural network learns which image features are useful for discriminating between the two classes. The resulting disambiguation algorithm produces highly accurate results, out-performing other inventor name disambiguation studies on US patent data. Our record linking algorithm could easily be adapted to other record linking problems, such as name disambiguation within academic publications. More broadly, a key contribution of this work is that our novel method of converting a string of text into an abstract image representation could potentially be applied to many other problems, as it allows image-based processing techniques, such as image classification neural networks, to be applied to text-based comparison problems, such as name disambiguation.

1. Introduction

Databases of patent applications and academic publications can be used to investigate the process of research and innovation. For example, patent data can be used to identify prolific inventors [3] or to investigate whether mobility increases inventor productivity [6], and academic publication data can be used to compare the scientific impact of open access journals relative to subscription journals [1]. However, the names of individuals in large databases are rarely distinct, and consequently individuals in such databases are not uniquely identifiable. For example, an individual named "Chris Jean Smith" may have patents under slightly different names such as "Chris Jean Smith", "Chris J. Smith", "C J Smith", etc... There may also be other distinct inventors with patents under the same or similar names, such as "Chris Jean Smith", "Chris J. Smith", "Chris Smith", etc... Thus it is ambiguous which names (and hence patents) should be assigned to which individuals, and this ambiguity makes it difficult to investigate research questions that involve individual authors/inventors. Resolving this identifiability issue and assigning unique identifiers to individuals — a process often referred to as name disambiguation — is important for research that relies on such databases.

However, manually disambiguating large databases that contain, say, millions of records, is infeasible. Machine learning algorithms have been used increasingly in recent years to perform automated disambiguation of inventor names in large databases (e.g. [14, 22, 10]). See Ventura *et al.* [22] for a review of supervised and semi-supervised machine learning approaches to disambiguation, as well as unsupervised approaches. These more recent machine learning approaches have generally out-performed more traditional rule- and threshold-based methods, but they have generally use feature vectors containing several pre-selected measures of string similarity as input for their machine learning algorithms. That is, the researcher generally preselects a number of string similarity measures which they believe may be useful as input for the machine learning algorithm to make discrimination decisions.

Here we intoduce a novel approach of representing text-based data which enables a supervised machine learning algorithm to learn its own features from the data, rather than selecting from a number of pre-defined string similarity measures chosen by the researcher. To do this, we treat the name disambiguation problem primarily as a classification problem, where pairwise comparisons between records are assessed as being either matched (same inventor) or non-matched (different inventors), an approach used in previous inventor name disambiguation studies [21, 15, 14, 22, 10]. Then, for a given pairwise comparison between two inventor records, we apply our novel text-to-image representation method to convert the associated text strings from those records into a 2D colour image that represents the underlying text data. We describe the text-to-image conversion method in detail in Section 4.1 (see Figure 1 in that section for an example of text-to-image conversion).

Our method of representing text-based record comparisons as abstract images also enables image processing algorithms, such as image classification neural networks, to be applied to text-based comparison problems, such as inventor name disambiguation. To demonstrate this, we modify a common Convolutional Neural Network (CNN) originally designed to classify images amongst 1,000 different classes [12] to make pairwise comparison decisions, in order to match inventor names in patent records, thus taking advantage of that CNN's powerful feature learning and pattern recognition capabilities. String similarities or differences between each pair of records produce spatial and colour features in the associated abstract image representation. The neural network is then able to learn which image features are useful for discriminating between matched and non-matched record pairs.

Our novel text-to-image conversion method also represents the data in such a way that the neural network is capable of accounting for spelling variations and word-ordering differences when discriminating between matches and non-matches, as well as potentially ignoring common (and hence non-specific) words that are not useful for discrimination decisions.

When combined with deduplication and blocking procedures that effectively reduce the complexity of the disambiguation problem, as well as a clustering algorithm that converts pairwise match/non-match probabilities into groups of matched inventor records, the resulting disambiguation algorithm generates highly accurate results, such as a precision of 99.41% and a recall of 98.76%.

2. Related Work

Inventor name disambiguation studies have often used measures of string similarity in order to make automated discrimination decisions. For example, counts of n-grams (sequences of n words or characters) can be used to vectorise text, with the cosine distance between vectors providing a measure of string similarity [19, 18]. Measures of edit distance consider the number of changes required to transform one string to another, such as the number of additions, subtractions, or substitutions used in the calculation of the Levenshtein distance [13, 17], or of other operations such as transpositions (the switching of 2 letters) used in the calculation of the Jaro-Winkler distance [8, 23]. Phonetic algorithms, such as Soundex, recode strings according to pronunciation, providing a phonetic measure of string similarity [19].

Measures of string similarity such as these have been used to guide rule- and threshold-based name disambiguation algorithms (e.g. [15, 16]). They can also be used within feature vectors inputted into machine learning algorithms. For example, Kim *et al.* [10] use such string similarity feature vectors to train a random forest to perform pairwise classification. Ventura *et al.* [22] reviewed several supervised, semi-supervised, and unsupervised machine learning approaches to inventor name disambiguation, as well as implementing their own supervised approach which utilised selected string similarity features as input to a random forest model.

CNNs have been used extensively in recent years in image processing applications, often exhibiting state-of-the-art level performance (e.g. [12, 20]). Also, neural networks (usually CNNs) have been used previously to assess pairwise comparison decisions — e.g. in the case of pairs of: images [11], image patches [27, 26], sentences [25], images of signatures [2], and images of faces [7].

These networks are generally set up such that multiple images are provided simultaneously as input, such as in the case of siamese neural networks where two identical sub-networks are connected at their output [2, 11]. In this work we generate a single image for a given pairwise record comparison, meaning that any image classification network could potentially be adapted with minimal modification to classify text-based record pairs as matched/non-matched. We demonstrate this using the seminal "AlexNet" image classification network [12].

3. Data

We use a combination of two labelled datasets in this work to train the neural network and assess its performance. Each dataset was derived from the United States National Bureau of Economics Research (NBER) Patent Citation Data File [5] by separate authors; a labelled dataset of Israeli inventors generated by Trajtenberg, Shiff, and Melamed [21] (the "IS" dataset), and another labelled dataset that focuses on patents filed by engineers and scientists compiled by Ge, Huang & Png [4] (the "E&S" dataset). These two datasets were combined with United

States Patent and Trademark Office (USPTO) patent data as part of the PatentsView Inventor Disambiguation Workshop[1] hosted by the American Institutes for Research (AIR) in September of 2015. Labelled data were matched by an AIR research team to USPTO bulk patent data, in order to recover additional patent data not provided by the labelled dataset creators. 80% of the labelled data were provided to participants of the workshop, with the remaining 20% being retained for algorithm evaluation by the workshop organisers (not used in this work).

Each labelled dataset contains unique IDs (UIDs) that identify all inventor-name records from different patents belonging to each individual inventor. We also extacted several other variables from each inventor-name record in the bulk USPTO patent data to use in our disambiguation algorithm: first name, middle name, last name, city listed in address, international patent classification (IPC) codes which specify subjects/fields covered by the patent, assignees (i.e. associated companies or institutes), and co-inventor names on the same patent.

4. Disambiguation Algorithm

Our novel inventor disambiguation algorithm involves the following main steps:

1. **Duplicate removal:** remove duplicate records from the patent data.

2. **Blocking:** block (or ”bin”) all names by last name, and also by first name in some cases. This improves the accuracy and efficiency of training, reduces the number of false positive errors, and improves computational efficiency.

3. **Genarate pairwise comparison-map images:** convert text from each pairwise record comparison into a 2D RGB bitmap image representation.

4. **Train neural network:** use such 2D images generated from manually labelled data to train a neural network to classify whether a given pairwise record comparison is a match (same inventor) or non-match (different inventors).

5. **Classify pairwise comparison-map images:** deploy the trained neural network to classify pairwise comparison images generated from the bulk patent data, producing a match probability for each record pair.

6. **Convert pairwise match probabilities into clusters:** convert the pairwise match/non-match probabilities generated by the neural network into clusters of inventors — i.e. linked groups of inventor-name records that each belong to a distinct individual inventor. Assigning a unique ID (UID) to each of these groups then leads to a single set of disambiguated inventor names.

[1]http://www.patentsview.org/community/workshop-2015

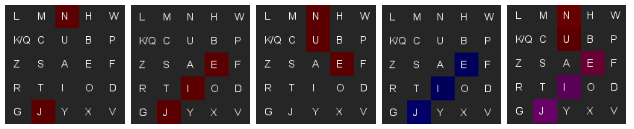

Figure 1. **Constructing a string-map image.** The first four images show the sub-maps that are summed to construct the final string-map image (rigth-most image) for the example word “JEN”.

Note that the main purpose of the first two steps is to reduce the complexity of the disambiguation problem. That is, if we were to investigate all possible pairwise comparisons, then the ≈ 12.4 million inventor name records in the bulk data would require that ≈ 77 trillion such 2-way comparisons be assessed. To improve the computational tractability of the problem, we (1) remove duplicate records and (2) separate the bulk patent data into related clusters (often referred to as “blocking” or “binning”). This reduces the number of pairwise comparisons that need to be assessed by the neural network, decreasing run-time and also helping to reduce the number of false positive matches obtained by the algorithm. We describe steps 1 and 2 in greater detail in the Supplementary Material, Appendices A and B respectively. Steps 3–6 are described in detail below.

4.1. Comparison-map images (representing text as 2D colour images)

Our intent is to assess all possible within-block pairwise comparisons between patent-inventor records, classifying each comparison as either a match or non-match. We use a CNN that was originally designed for image classification — i.e. the seminal “AlexNet” network [12] — and adapt it to perform such binary classification. We could have used any image classification neural network, but chose to use AlexNet since it is a seminal image classification network which has been re-used as a benchmark in many later research papers. We do not expect that choosing a more advanced image classification network would affect our results substantially, as there are only two classes in our classification problem and we obtain very high classification accuracy after training.

In order to perform text-based classification using an image classification neural network, we introduce a novel method of converting any string of text into an abstract image representation of that text, which we apply to pairwise comparisons between inventor-name records. This novel text-to-image conversion method firstly involves defining a specific 2-dimensional character layout that we will refer to as a “string-map” — i.e. a grid of pixels, where each pixel corresponds to a particular letter in the English alphabet (see

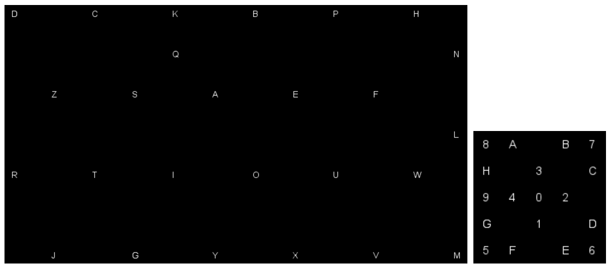

Figure 2. **Larger string-map for assignees and co-inventors, and IPC-map.** The larger string-map used to convert a given list of assignees or co-inventors into an abstract image representation (left), and the IPC-map used to convert a given list of IPC classes into an abstract image representation (right).

the layout within each of the five images in Figure 1). For a given word (e.g. "JEN"), we then add a particular colour (e.g. red) to the pixels on the map corresponding to each letter within the word, as well as to any pixels that fall within straight lines connecting the letters (see sequence of images in Figure 1). In particular, we add colour to the pixels of the first and last letters (Figure 1, left-most image), and to all pixels in a line connecting each two-letter bi-gram (e.g. there are two bi-grams in "JEN": "JE" and "EN"). To highlight the beginning of each string-map, we also repeat the process for the first bi-gram only (e.g. "JE") in blue, rather than red (Figure 1, second image from the right).

The string-map shown in Figure 1 is used for the inventor's first name, middle name, last name, and the city listed in their address. Since a given patent-inventor record can have multiple assignees and/or co-inventors, we use a larger string-map for those fields (see Figure 2, left image), which reduces the possibility that pixels will become saturated in cases where many assignees (or co-inventors) are overlayed onto the same string-map. We also add less colour to each pixel in these larger string-maps compared with the smaller string-maps, again to reduce the possibility of saturation. For international patent classification (IPC) codes, which contain numbers as well as letters, we use a different string-map shown in Figure 2 (right image; an "IPC-map").

For a given inventor name record, the image representations of first name, middle name, last name, city, IPC codes, co-inventors, and assignees are arranged as shown in Figure 3, which we refer to as a "record-map".

We compare any two inventor name records by overlaying (summing) each corresponding record-map in a different colour (one in red, the other in green) onto the same image, which we refer to as a "comparison-map" (Figure 4). Since red and green combined (summed) produce yellow in the RGB colour model, then if the two words are similar, one distinctive feature of the resulting comparison image will be that it will contain a lot of yellow (e.g. Figure 4, left image). If the two records are dissimilar, there will be more red and green in the image due to less overlap between the two record-maps (e.g. Figure 4, right image). When training on labelled comparison-map images, we expect that the neural network will learn to identify which image features are important for discriminating between matched/non-matched patent-inventor name records. That is, the neural network's learned pattern recognition on comparison-map images will essentially recognise underlying text patterns which are present in the patent-inventor name records, such as similarities or differences between the last names of each of the two records, etc…

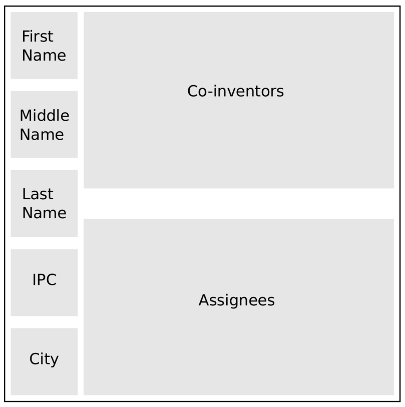

Figure 3. **Record-map layout.** This shows the positioning of each wordmap and IPC-map within a given record-map.

Note that we chose the particular layout of the letters in the string-map shown in Figure 1 heuristically, such that vowels are positioned towards the centre of the grid, in an attempt to make it more straightforward for the neural network to learn which features are associated with matches/non-matches. Our rationale for this is that we expect (1) more saturation to occur in the pixels towards the centre of each string-map, and (2) vowels to be generally less important than consonants for identifying particular words and distinguishing them from other words. We also sought to group letters with similar phonetic interpretations, such as "S" and "Z", close to each other. We test how the heuristic layout performs compared with several alternative random layouts later in Section 5.4.

4.1.1 Benefits of representing text-based records as comparison-map images

Note that the above method of converting text into a 2D RGB bitmap for neural network-based image classification is likely to be more computationally expensive than most name disambiguation algorithms. However, the method also has several benefits:

- the powerful classification capabilities of previous image classification neural networks can be utilised for text-based record matching, with minimal modification,

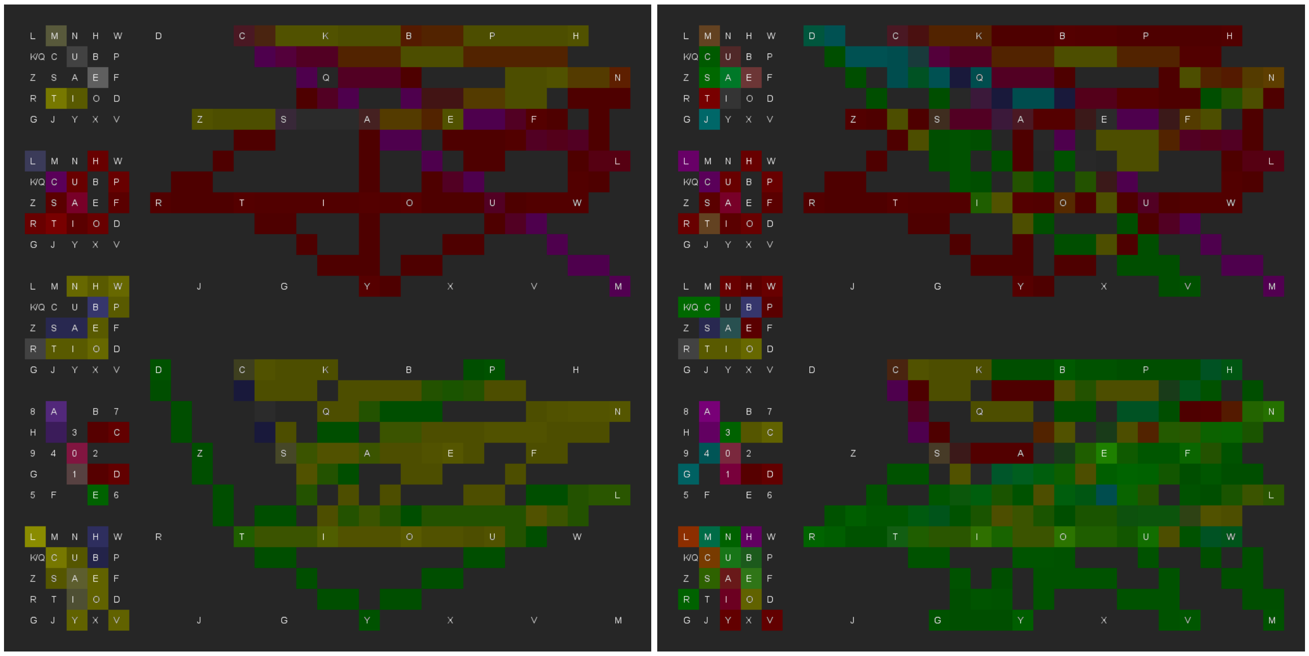

Figure 4. **Comparison-map examples.** Two examples of comparison-map images. The left comparison-map image was generated using two mock matched records (Table 1, rows 1 and 2), and the right image from two mock non-matched records (Table 1, rows 1 and 3).

Table 1. Mock records of three patent inventor name instances. Rows 1 and 2 are patent-inventor name instances for the same mock inventor, while row 3 is a different inventor.

Name	IPC codes	City	Co-inventors (last names)	Assignees
Emmett Lathrop Brown	A10C, A10D	Hill Valley	McFly, Clayton-Brown, Sanchez	Science Solutions
Emmett L. Brown	A11E	Hill Valley	Sanchez	Science Solutions Pty. Ltd.
James T. Brock	G03C	Melbourne	Edison, Da Vinci	Swinburne University of Technology, The University of Melbourne

- the neural network learns its own features from the data, rather than learning from a selection of pre-defined string similarity measures chosen by the researcher,
- minor spelling variations and errors do not alter the resulting string-map very much, and the neural network has the potential to learn to recognise such minor features in resulting comparison-map images as relatively unimportant features (that are not necessarily associated with a non-match),
- if two matched records have different word ordering (i.e. words are translated to different positions in the string, such as when a shared co-inventor name appears in different positions on the two different patents), the resulting comparison-map image is still likely to be recognisable as a match by the neural network, because overlapping (summing) green from one record-map and red from the other generates yellow regardless of word order,
- the neural network has the potential to learn to recognise and ignore certain shapes corresponding to common words that are not very useful for discriminating between matches and non-matches (such as “Ltd”, “LLC”, “Incorporated”, etc. . .),
- we show later in Section 5.4 that our novel disambiguation algorithm performs well under multiple different choices of alternative string-maps other than those shown in Figures 1 & 2, suggesting that the neural network has quite robust pattern recognition of features within our comparison-map representations.

These benefits of our text-to-image conversion method could potentially apply to many other text-based comparison problems, in addition to the name disambiguation problem that we address in this work.

4.2. Modifications to neural network architecture

The neural network we use in this study is based on the CNN developed by Alex Krizhevsky *et al.* [12]

Hyperparameter	AlexNet	This work	Rationale for modification
Number of neurons in input layer	$224 \times 224 \times 3 = 150,528$	$31 \times 31 \times 3 = 2,883$	Smaller size of input image bitmaps (note that we did not crop our images)
Kernel size in first convolutional layer	$11 \times 11 \times 3$	$3 \times 3 \times 3$	Smaller-scale features to learn
Stride length for kernels in first convolutional layer	4	1	Smaller kernel size
Number of neurons in output layer	1,000	2	Fewer classes

Table 2. Hyperparameters that differ between the two network architectures. See Krizhevsky *et al.* [12] for more details on the network architecture and corresponding hyperparameters.

(often referred to as "AlexNet"). AlexNet was originally designed to classify colour images (224×224×3-pixel bitmaps) amongst 1,000 different classes (container ship, motor scooter, leopard, mite, etc...). Here we modify the network architecture to enable classification of pairwise comparison-map images (31×31×3-pixel bitmaps) into two classes (match/non-match), by altering several hyperparameters as shown in Table 2.

We use the NVIDIA Deep Learning GPU Training System[2] (DIGITS) v2.0.0 implementation of AlexNet, using the Caffe backend [9]. We use the default solver (stochastic gradient descent), batch size (100), and number of training epochs (30). Rather than use the default learning rate (0.01) we use a sigmoid decay function to progressively decrease the learning rate from 0.01 to 0.001 over the course of the 30 training epochs, as initial testing indicated that this produced slightly higher accuracies. Note that the default settings of the DIGITS v2.0.0 implementation of AlexNet also transform the input data by (1) altering input images to show the deviation from the mean of all input images (by subtracting the mean image from each input image), (2) randomly mirroring input images, and (3) taking a random square crop from the input image. The main purpose of performing such transformations is to introduce variability into the training images that are expected to be present in the unlabelled data, however we do not use any of those transformations in this work because our images are much more self-consistent than those in the ImageNet database.

The output layer of AlexNet is a 1,000-neuron softmax layer, which generates a probability distribution across the 1,000 possible classes. Our modified network has a 2-neuron softmax output layer which, for a given inputted pairwise comparison image, produces a probability distribution across the two possible classes (match/non-match).

4.3. Converting pairwise probabilities into inventor groups, and assigning UIDs

After running the trained neural network on bulk patent data, each within-block pairwise comparison has an associated match/non-match probability. To assign unique IDs (UIDs) to the bulk data, we convert these pairwise probabilities into linked (matched) "inventor groups". Each inventor group is a linked cluster of inventor name records which all refer to the same individual. The clustering algorithm is described in Appendix C. Once the clustering algorithm has been applied to each block, every patent-inventor name instance has an associated UID, and the disambiguation process is complete.

[2]https://developer.nvidia.com/digits

5. Results

Here we firstly describe our procedure for dividing our labelled datasets into training and test data. We then evaluate our inventor disambiguation algorithm, compare those results to previous studies, and test alternative string-map layouts.

5.1. Datasets

We use the IS and E&S labelled datasets to train the neural network to discriminate between matched and non-matched pairwise comparisons. Each of the labelled datasets are separated into 80% training data (used to train the neural network) and 20% test data (used to assess algorithm performance). The split is made by randomly selecting 80% of labelled inventor UIDs and using the associated records as the training dataset, while the remainder of all records are used as the test dataset. We use 75% of the training data to train the network, and the remaining 25% to perform validation assessments during training in order to monitor potential overfitting.

Duplicate removal and blocking is then performed on the labelled data, and comparison-map images are generated for all possible pairwise record comparisons within each block (723,178 comparison-maps for training and 144,552 comparison-maps for testing).

We generate comparison-maps for all possible pairwise within-block comparisons in the labelled data and the bulk data (stored as 3D numerical arrays for computational efficiency). The trained neural network is then deployed on the bulk patent data, generating match/non-match probabilities for all pairwise within-block comparisons in the bulk data (112,068,838 comparison-maps). We experimented with multiple different values for the pairwise comparison probability threshold, $\bar{p}$, and linking proportion threshold, $\bar{l}$ (see

Appendix C), based on evaluating the trained neural network on the labelled test data (prior to processing the bulk data). Different $\bar{p}$ and $\bar{l}$ values produce different trade-offs between precision and recall, and we use values that produce an optimal trade-off (we state those values whenever quoting results from a given run of our disambiguation algorithm).

5.2. Evaluation

To evaluate the performance of the disambiguation algorithm, we use the labelled IS and E&S test data to estimate precision, recall, splitting, and lumping based on numbers of true positive (*tp*), false positive (*fp*), true negative (*tn*), and false negative (*fn*) pairwise links within the labelled test data, as follows [22, 10]:

$$Precision = \frac{true\ positive\ matches}{all\ positive\ matches} = \frac{tp}{tp + fp} \qquad (1)$$

$$Recall = \frac{true\ positive\ matches}{total\ true\ matches} = \frac{tp}{tp + fn} \qquad (2)$$

$$Splitting = \frac{false\ negative\ non\text{-}matches}{total\ true\ matches} = \frac{fn}{tp + fn} \qquad (3)$$

$$Lumping = \frac{false\ positive\ matches}{total\ true\ matches} = \frac{fp}{tp + fn} \qquad (4)$$

Higher values are better for precision and recall, while lower values are better for lumping and splitting.

Note that obtaining a very high score on just one of these metrics is not necessarily desirable, as there is a trade-off between precision and recall that must be considered (or, equivalently, a trade-off between lumping and splitting). For example, we can easily obtain 100% recall by simply matching every single pair of records, but the precision of such an algorithm would be extremely low. Or we could obtain 100% precision by requiring that all record fields must be exactly identical for a matched pair, but then recall would be very low. Thus we also estimate the pairwise F1 score, a measure that attempts to take into account the trade-off between precision and recall:

$$F1 = 2 \times \frac{Precision \cdot Recall}{Precision + Recall} \qquad (5)$$

Given that F1 takes into account the trade-off between precision and recall, it is the primary measure we use when comparing different disambiguation algorithms.

Table 3. Comparison of the performance of two example runs of our disambiguation algorithm (the runs with highest F1 and highest precision), relative to that of other inventor name disambiguation studies of bulk US patent data. All values in percentages (%).

Method [$\bar{p}$; $\bar{l}$]	Split	Lump	Recall	Precision	**F1**
Li2014[3]	3.26	2.34			
Ventura2015	2.31	1.64			
Morrison2017			92	98	95
Yang2017			96.15	99.61	97.85
Ours [0.02; 0.1]	**1.33**	0.51	**98.67**	99.48	**99.07**
Ours [0.01; 0.4]	2.87	**0.28**	97.13	**99.71**	98.40

Table 4. Comparison of the performance of our disambiguation algorithm relative to that of Yang *et al.* [24] results evaluated on the IS and E&S labelled datasets. All values in percentages (%).

Method	Recall	Precision	**F1**
Yang2017 (IS)	83.79	99.57	91.00
Yang2017 (E&S)	90.31	**99.87**	94.85
Ours [0.02; 0.1]	**98.67**	99.48	**99.07**
Ours[4] [0.01; 0.4]	96.84	99.84	98.32

5.3. Disambiguation algorithm performance

The precision, recall, splitting, lumping, and F1 estimates for the two runs of our disambiguation algorithm which generated the highest F1 and highest precision are shown in Table 3 (bottom two rows). We also show the best results from other state-of-the-art studies involving name disambiguation of bulk USPTO patent data in Table 3 (in studies that quote multiple F1 scores we show the results with the highest F1 score). Note that we do not include results from the Kim *et al.* [10] US patent disambiguation study in Table 3, because we found that their blocking procedure adversely affects their estimates of recall (see Appendix B.1). Our inventor disambiguation algorithm outperforms other state-of-the-art disambiguation studies in terms of obtaining the highest F1 score.

It is important to note that the results from other studies shown in Table 3 use different labelled datasets to the IS and E&S datasets used in this work, making comparison difficult. This is a common problem with comparing the performance of different inventor name disambiguation studies. For Yang *et al.* [24], the results we quote in Table 3 are their highest F1 results, which they evaluated using the

[3]Note that Ventura *et al.* [22] also use their "optoelectronics" labelled dataset to evaluate the Li *et al.* [14] disambiguation algorithm, obtaining 2.49% splitting error and 0.39% lumping error on the full optoelectronics dataset, but much poorer performance on a random sample of the optoelectronics labelled data, with 10.54% splitting error and 1.21% lumping error.

[4]Note that these results were obtained using a randomly-generated string-map character order (see Section 5.4).

Table 5. Comparison of results from alternate string-map layouts. Each row shows the highest F1 result obtained for that string-map layout, with the corresponding pairwise comparison probability threshold ($\bar{p}$) and linking proportion threshold ($\bar{l}$) shown in square brackets.

String-map layout [$\bar{p}$-threshold; $\bar{l}$-threshold]	Splitting	Lumping	Recall	Precision	**F1**
Random character order [0.03; 0.05]	1.24	0.58	98.76	99.41	99.09
Random character order and layout [0.05; 0.05]	1.23	0.70	98.77	99.29	99.03
Random character order and layout, small string-maps [0.05; 0.2]	1.54	0.47	98.46	99.52	98.99
Heuristic character order and layout [0.02; 0.1]	1.33	0.51	98.67	99.48	99.07

"academic life sciences" labelled dataset, a relatively clean labelled dataset that is relatively easy to disambiguate [22]. However, Yang *et al*. [24] also used the IS and E&S labelled datasets to evaluate their algorithm, and we compare those results to our results in Table 4. This more equitable comparison shows that our disambiguation algorithm outperforms theirs even moreso when evaluated using the same labelled datasets, as our F1 score is much higher. Note that the Yang *et al*. [24] result evaluated on the E&S dataset displays slightly higher precision than ours, but this comes at the cost of very low recall and hence a much lower F1 score. Our precision-recall trade-off is much better, as our highest precision result of 99.84% (Table 4, bottom row; obtained using a randomly-generated string-map character order — see Section 5.4 below) has much higher recall than that of Yang *et al*. [24], and hence, much higher F1.

5.4. Testing different string-maps

Here we compare the performance of our heuristically-determined string-map layout (Section 4.1) to multiple pseudo-random string-map layouts. For a given random string-map (or IPC-map), we keep the pixel co-ordinates of each string-map identical to that of the associated heuristic layout, but randomise the position of each character. This randomised string-map is shown in the Supplementary Material, Appendix D, Figure S1. We also try another alternative string-map in which we randomise both the pixel co-ordinate layout and character positions (Appendix D, Figure S2), as well as another alternative with random layout and character positions (Appendix D, Figures S3 & S4) in which we use the smaller 5×5 pixel string-map (Figure 1) for co-inventors and assignees, rather than the larger string-map (Figure 2, left image).

Estimates of precision, recall, splitting, lumping, and F1 are shown in Table 5. For each alternative string-map layout, we ran the algorithm multiple times using different settings of the comparison probability threshold ($\bar{p}$) and linking proportion threshold ($\bar{l}$), and only show results from the run which produced the highest F1 score. Results obtained from each of the alternative random string-maps are very similar to those obtained using the heuristically-determined layout (F1 scores range from 98.99% to 99.09%). Thus, using a random string-map layout appears to have very little effect on the ability of the neural network to learn to recognise patterns in comparison-maps which are useful for discriminating between matched and non-matched records. This suggests that our method of converting text into abstract image representations, and using a neural network to classify those images, is quite robust to such changes in string-map layout.

6. Conclusion

Our novel inventor name disambiguation algorithm produces highly accurate results, out-performing other state-of-the-art inventor disambiguation algorithms with an F1 score of 99.09%, a precision of 99.41%, and a recall of 98.76%. Our text-to-image conversion method also represents the data in such a way that the neural network is capable of accounting for spelling variations and word-ordering differences when discriminating between matches and non-matches, as well as potentially ignoring common, non-specific words that are not useful for discrimination decisions. Our method also works with previous image classification neural networks, allowing the network to learn its own features from the data, rather than learning to select from a number of pre-defined string similarity measures chosen by the researcher. We also analysed several variants of alternative string-map layouts, finding that the accuracy of the disambiguation algorithm was highly robust to such variation.

Our disambiguation algorithm would also be easily adapted to other text-based record linkage problems such author name disambiguation in scientific publications. The algorithm could also potentially be adapted to process records that contain both text and image data, by combining each record's associated image with the abstract image representation of its associated text in a single comparison-map.

More broadly, our method of representing text strings as abstract images provides a novel way of integrating image processing with text processing. This has the potential to allow image pattern recognition algorithms, such as image classification neural networks, to be applied more broadly to problems requiring text pattern recognition.

References

[1] B.-C. Björk and D. Solomon. Open access versus subscription journals: a comparison of scientific impact. *BMC Medicine*, 10(1):73, 2012. 1

[2] J. Bromley, J. W. Bentz, L. Bottou, I. Guyon, Y. Lecun, C. Moore, E. Säckinger, and R. Shah. Signature Verification Using a Siamese Time Delay Neural Network. *International Journal of Pattern Recognition and Artificial Intelligence*, 07(04):669–688, 1993. 2

[3] C. Gay, W. Latham, and C. Le Bas. Collective knowledge, prolific inventors and the value of inventions: An empirical study of French, German and British patents in the US, 1975-1999. *Economics of Innovation and New Technology*, 17(1-2):5–22, 2008. 1

[4] C. Ge, K. Huang, and I. P. L. Png. Engineer/scientist careers: Patents, online profiles, and misclassification bias. *Strategic Management Journal*, 37:232–253, 2016. 2

[5] B. H. Hall, A. B. Jaffe, and M. Trajtenberg. The NBER Patent Citation Data File: Lessons, Insights and Methodological Tools. *National Bureau of Economic Research Working Paper*, 8498, 2001. 2

[6] K. Hoisl. Does mobility increase the productivity of inventors? *Journal of Technology Transfer*, 34:212–225, 2007. 1

[7] J. Hu, J. Lu, and Y. P. Tan. Discriminative deep metric learning for face verification in the wild. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1875–1882, 2014. 2

[8] M. a. Jaro. Advances in record-linkage methodology as applied to matching the 1985 census of Tampa, Florida. *Journal of the American Statistical Association*, 84(406):414–420, 1989. 2

[9] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional Architecture for Fast Feature Embedding. *Proceedings of the 22nd ACM international conference on Multimedia*, pages 675–678, 2014. 6

[10] K. Kim, M. Khabsa, and C. L. Giles. Random Forest DBSCAN for USPTO Inventor Name Disambiguation. *arXiv:1602.01792*, [cs.IR], 2016. 1, 2, 7, 10, 11

[11] G. Koch, R. Zemel, and R. Salakhutdinov. Siamese Neural Networks for One-Shot Image Recognition. *Proceedings of the 32nd International Conference on Machine Learning*, 37, 2015. 2

[12] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. *Advances in Neural Information Processing Systems*, 25:1097–1105, 2012. 2, 3, 5, 6

[13] V. I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10(8):707–710, 1966. 2

[14] G. C. Li, R. Lai, A. D'Amour, D. M. Doolin, Y. Sun, V. I. Torvik, A. Z. Yu, and F. Lee. Disambiguation and co-authorship networks of the U.S. patent inventor database (1975-2010). *Research Policy*, 43(6):941–955, 2014. 1, 2, 7

[15] E. Miguélez and I. Gómez-Miguélez. Singling Out Individual Inventors from Patent Data. *Research Institute of Applied Economics Working Paper*, 2011. 2

[16] G. Morrison, M. Riccaboni, and F. Pammolli. Disambiguation of patent inventors and assignees using high-resolution geolocation data. *Scientific Data*, 4:1–21, 2017. 2

[17] G. Navarro. A guided tour to approximate string matching. *ACM Computing Surveys*, 33(1):31–88, 2001. 2

[18] M. Pezzoni, F. Lissoni, and G. Tarasconi. How to kill inventors: testing the Massacrator algorithm for inventor disambiguation. *Scientometrics*, 101(1):477–504, 2014. 2

[19] J. Raffo and S. Lhuillery. How to play the "Names Game": Patent retrieval comparing different heuristics. *Research Policy*, 38(10):1617–1627, 2009. 2

[20] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015. 2

[21] M. Trajtenberg, G. Shiff, and R. Melamed. The "Names Game": Harnessing Inventors' Patent Data for Economic Research. *National Bureau of Economic Research Working Paper*, 12479, 2006. 2

[22] S. L. Ventura, R. Nugent, and E. R. H. Fuchs. Seeing the non-stars: (Some) sources of bias in past disambiguation approaches and a new public tool leveraging labeled records. *Research Policy*, 44(9):1672–1701, 2015. 1, 2, 7, 8, 10

[23] W. E. Winkler. String Comparator Metrics and Enhanced Decision Rules in the Fellegi-Sunter Model of Record Linkage. *Proceedings of the American Statistical Association Section on Survey Research Methods*, 1990. 2

[24] G.-C. Yang, C. Liang, Z. Jing, D.-R. Wang, and H.-C. Zhang. A Mixture Record Linkage Approach for US Patent Inventor Disambiguation. *Advanced Multimedia and Ubiquitous Engineering. FutureTech 2017, MUE 2017. Lecture Notes in Electrical Engineering*, 448:331–338, 2017. 7, 8

[25] W. Yin, H. Schütze, B. Xiang, and B. Zhou. ABCNN: Attention-Based Convolutional Neural Network for Modeling Sentence Pairs. *Transactions of the Association for Computational Linguistics*, 4:259–272, 2016. 2

[26] S. Zagoruyko and N. Komodakis. Learning to Compare Image Patches via Convolutional Neural Networks. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 2

[27] J. Zbontar and Y. LeCun. Stereo Matching by Training a Convolutional Neural Network to Compare Image Patches. *Journal of Machine Learning Research*, 17:1–32, 2016. 2

A. Removal of duplicate records

It is sometimes obvious that two inventor name records likely belong to the same individual, because the two records contain several fields that are identical. For example, if the last name, first name, city, and IPCs of two different records are all exactly identical, it is highly likely that the two records belong to the same individual. Removing these "duplicates" from the bulk data will make disambiguating the entire dataset more computationally tractable.

We remove duplicate records based on the following duplication keys:

```
duplicnkey_ipc = last_name + first_name
+ city + '_'.join(ipcs)

duplicnkey_assignee = last_name + first_name
+ city + '|'.join(assignees)
```

Duplicate records are identified firstly using the IPC duplication key (`duplicnkey_ipc`), then using the assignee duplication key (`duplicnkey_assignee`). For a given group of duplicate records sharing the same duplication key, all records except for the first record to be processed are removed from the bulk data. The first record then remains within the bulk data to be processed by the disambiguation algorithm, receiving a unique ID once the algorithm has completed its run. That same ID is then assigned to each removed record in the corresponding group of duplicate records.

B. Blocking

The blocking procedure broadly involves grouping together inventor name records into "blocks" (or "bins") using each inventor's last name, and sometimes also their first name. Latter parts of the algorithm will only assess pairwise comparisons within these blocks, never across different blocks.

We firstly group patent-inventor name records together by the first three letters of the last name (this first step is identical to the initial stage of the blocking procedure used by Ventura *et al.* [22]). However, some of the resulting blocks contain very large numbers of records, and hence large numbers of pairwise comparisons. To improve efficiency, we further divide such large blocks into smaller blocks by progressively increasing the number of letters used for blocking. That is, if the number of records within a given block (n_b) is above some threshold number ($\bar{n}_b$), then the records within that block are separated into smaller blocks according to the first *four* letters of the last name. We then continue sub-dividing any blocks that still have $n_b > \bar{n}_b$, according to the first five letters of the last name, then six letters, and so on. If all letters of the last name have been used and any blocks still have $n_b > \bar{n}_b$, then we append a comma to the string and begin progressively appending letters from the first name as well.

We use $\bar{n}_b = 100$ throughout this work, as initial testing indicated that it produced a good balance between:

- computational efficiency: i.e. smaller $\bar{n}_b$ leads to more numerous, smaller bins, and hence fewer comparisons (which are $\mathcal{O}(n_b^2)$ for each bin) and less computation time,
- accuracy: i.e. smaller $\bar{n}_b$ reduces the number of unnecessary comparisons between records (often non-matched records) which should reduce the number of false positives,
- recall: i.e. larger $\bar{n}_b$ leads to fewer, larger bins, which decreases the number of splitting errors, decreasing false negatives,

Together with the deduplication procedure, this reduces the number of pairwise comparisons from ≈ 77 trillion before the blocking procedure to ≈ 112 million.

Note that since latter parts of the algorithm only assess within-block pairwise comparisons and some inventors' sets of records may have been separated across two or more different blocks, there is a maximum limit to the possible recall attainable by the disambiguation algorithm. After running the blocking procedure on the labelled dataset, we use known pairwise matches in the labelled data to estimate this maximum limit to recall, obtaining the following values: 99.47% (E&S training data), 99.98% (E&S test data), 99.83 (IS training data), and 99.86 (IS test data).

B.1. More stringent binning procedures produce poorer estimates of recall

Kim *et al.* [10] perform a simpler, more stringent blocking procedure than that described above, in which records are assigned to different blocks based on the full last name and first initial of first name. This more stringent *"full last name, first initial"* blocking procedure will separate more unique inventors' matched records into different blocks. Also, the larger the dataset, the more likely it is to contain cases of unique inventors with large numbers of associated records that may become separated into different blocks during the blocking procedure. Note that since the number of pairwise links for a given unique inventor scales with the number of associated records (n) as: $n(n-1)/2$, the number of *broken* pairwise links will have a disproportionately large contribution from any very prolific inventors (large n) whose associated records become separated into different blocks during blocking.

As an example of how this issue could potentially be problematic for recall estimates, let us consider a labelled dataset in which there are, say, only one or two cases where very prolific inventors (with very large n) have their associated records separated across blocks. We can expect such cases to be more likely to appear in the training data rather than the test data, simply because the training dataset comprises a larger random sample of the whole labelled dataset (80% training versus 20% test). Thus estimates of recall made using the test data may underestimate recall on the larger training dataset. The underestimate may be even more profound for the bulk data, as it is by far the largest of the three datasets.

To investigate the extent of this issue under the Kim *et al.* [10] blocking procedure, we applied the same *"full last name, first initial"* blocking procedure to the IS and E&S data (which were also used in the [10] study), comparing the maximum possible recall obtainable on the test and training datasets in each case. For the smaller IS dataset ($6,711$ records), we find a 0.74% difference in maximum recall (test: 99.28%, training: 98.54%) and for the E&S dataset ($41,795$ records) we find an even larger difference in maximum recall of 2.36% (test: 99.09%, training: 96.73%), and we expect that this issue will also apply to the even larger bulk dataset ($12,391,977$ records). While the result that Kim *et al.* [10] obtained for recall on the E&S data (98.05%) is consistent with the maximum possible recall that we computed for the E&S test dataset (99.09%), it is not consistent with our calculated value of the maximum possible recall obtainable on the E&S training dataset (96.73%), as their estimate lies above that value.

This suggests that when using a more stringent binning procedure, recall estimates based on small test datasets will likely provide a relatively poor representation of recall on larger datasets — such as the training dataset — and will likely provide an even poorer representation of recall on the much larger bulk data. Given this issue, as well as the above inconsistency between the Kim *et al.* [10] E&S recall estimate and our calculated value of the maximum possible recall obtainable on the E&S dataset using their binning procedure, we exclude the Kim *et al.* [10] results when comparing to recall and precision estimates from previous studies in Section 5.3, Table 3.

Note that the above issue will inevitably apply to some extent to all possible blocking procedures, but will be less prevalent for less divisive blocking procedures. For example, the issue appears to be less prevalent under our adaptive blocking procedure, as it has more consistency between the maximum possible recall of the testing and training datasets, for both the IS data (0.03% difference, i.e. test: 99.86%, training: 99.83%) and E&S data (0.51% difference, i.e. test: 99.98%, training: 99.47%). Note also that there are no inconsistencies between these maximum possible recall values and any of the recall estimates we show in our results (i.e. recall estimates from Tables 3–5 range from 96.84% to 98.77%).

C. Clustering algorithm to assign inventor groups

Here we describe the clustering algorithm we use to convert pairwise match/non-match probabilities into groups of records each belonging to a single unique inventor. We firstly convert each pairwise probability between the ith and jth record (p_{ij}) into one of the binary classes (c_{ij}; either "match" or "non-match") based on a threshold probability value ($\bar{p}$) as follows:

$$c_{ij} = \begin{cases} \text{match}\,, & \text{if } p_{ij} \geqslant \bar{p} \\ \text{non-match}\,, & \text{otherwise.} \end{cases} \tag{6}$$

Note that the value we choose for the pairwise comparison probability threshold, $\bar{p}$, will have an effect on the trade-off between precision and recall attained by the algorithm.

The inventor group linking algorithm then primarily involves combining different sub-groups together into the one group if they share enough links (pairwise matches). Within a given block, the algorithm involves the following steps:

1. Order all patent-inventor name records by the number of links they have to other records (i.e. the number of asserted matches to other records), highest first.

2. Assign a UID to each isolated (non-matched) patent-inventor name.

3. Assign records to inventor groups. That is, for a given record, the corresponding inventor group initially comprises just the record itself and all records it is linked (matched) to. Each of these linked records (nodes) are kept in the current inventor group only if the number of links (l) it has to the current group is $\geqslant$ the number of nodes in the group (n) times some threshold proportion ($\bar{l}$); i.e. if $l \geqslant n\bar{l}$. This removes the most weakly-linked records from each group (i.e. the nodes with fewest links to their group), which are more likely to be false positive matches. Note that the value we choose for the linking threshold proportion, $\bar{l}$, will have an effect on the trade-off between precision and recall attained by the algorithm. Also, note that due to Step 1, the most strongly-linked nodes are processed first, which ensures that insufficiently-linked nodes do not erroneously get assigned to a group simply because they are processed at an earlier stage (when the number of nodes in the group is still building up). Any outside-group links — i.e. links to nodes that are not within the current group — are also recorded during this step.

4. Repeat Step 2, because some records may have become isolated (non-matched) following Step 3.

5. For each inventor group containing nodes with outside-group links, combine it with any of those other groups

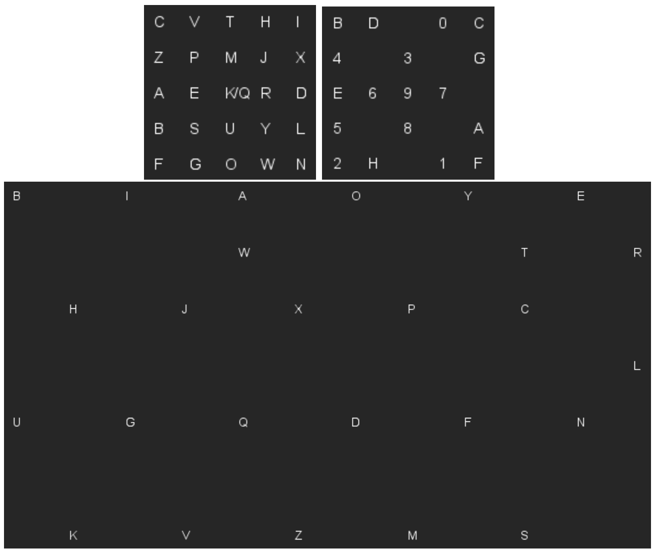

Figure S1. **Random character order (string-maps).** Here we show the smaller string-map (top-left), IPC-map (top-right), and larger string-map (bottom) we use for runs in which the character order has been randomised.

if the number of links they share is greater than a specified threshold. In particular, for an inventor group with n_{self} records (nodes), we combine it with any other group with n_{other} nodes if the number of links to that other group (l) satisfies both: $l \geqslant \bar{l}\, n_{\text{self}}$, and: $l \geqslant \bar{l}\, n_{\text{other}}$.

6. For each resulting inventor group, assign an identical UID to all patent-inventor name records within the group.

Once this clustering algorithm has been applied to each block, every patent-inventor name has an associated UID, and the disambiguation process is complete.

D. Random string-map layouts

Here we show the random string layouts analysed in Section 5.4. Figure S1 shows the string-maps and IPC-map we use for runs where characters are positioned using an identical pixel co-ordinate layout to the heuristic layouts shown earlier in Section 4.1, but where the order of each character has been randomised.

Figure S2 shows the string-map and IPC-map we use for runs where both the pixel co-ordinate layout and character positions have been randomised.

Figure S3 shows the comparison-map with random layout and character order in which we use the smaller 5×5 pixel string-map (Figure 1 in Section 4.1) for co-inventors and assignees, rather than the larger string-map (Figure 2 in Section 4.1, left image). Figure S4 shows the record-map layout associated with the comparison-map in Figure S3.

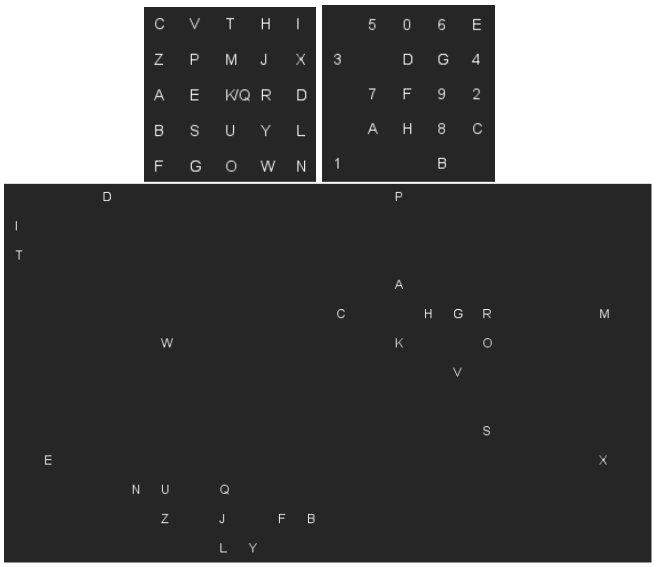

Figure S2. **Random character order and layout (string-maps).** Here we show the smaller string-map (top-left; identical to the top-left string-map in Figure S1), IPC-map (top-right), and larger string-map (bottom) with both random character order and random pixel co-ordinate layout.

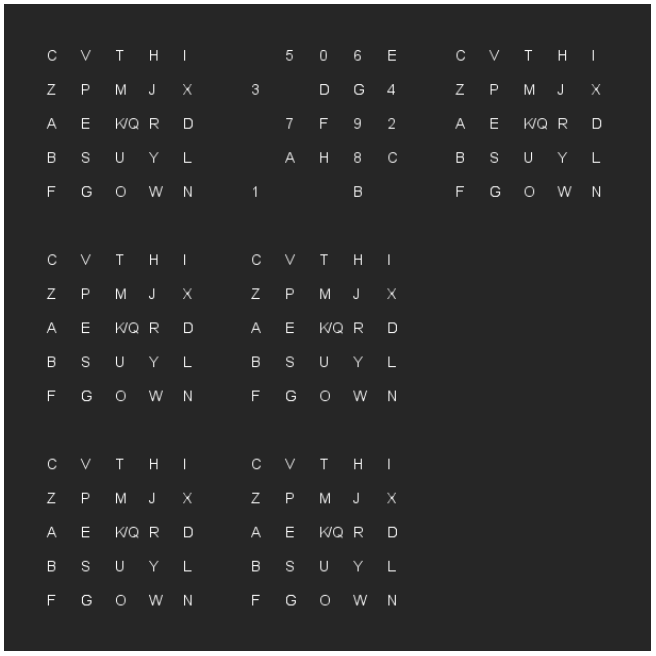

Figure S3. **Random character order and layout, small string-maps (comparison-map).** This shows the comparison-map used for runs with smaller string-maps for co-inventors and assignees, as well as random character order and random pixel co-ordinate layout.

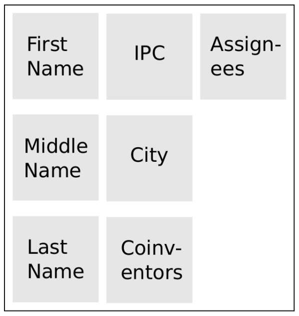

Figure S4. **Record-map layout with smaller string-map for co-inventors and assignees.** The record-map layout associated with the comparison-map in Figure S3.