On generating large-scale ground truth datasets for the deduplication of bibliographic records

James A. Hammerton* j_hammerton@yahoo.co.uk
Michael Granitzer† mgrani@know-center.at
Dan Harvey‡ danharvey42@gmail.com
Maya Hristakeva maya.hristakeva@mendeley.com
Kris Jack kris.jack@mendeley.com

ABSTRACT

Mendeley's crowd-sourced catalogue of research papers forms the basis of features such as the ability to search for papers, finding papers related to one currently being viewed and personalised recommendations. In order to generate this catalogue it is necessary to deduplicate the records uploaded from users' libraries and imported from external sources such as PubMed and arXiv. This task has been achieved at Mendeley via an automated system.

However the quality of the deduplication needs to be improved. "Ground truth" data sets are thus needed for evaluating the system's performance but existing datasets are very small. In this paper, the problem of generating large scale data sets from Mendeley's database is tackled. An approach based purely on random sampling produced very easy data sets so approaches that focus on more difficult examples were explored. We found that selecting duplicates and non duplicates from documents with similar titles produced more challenging datasets. Additionally we established that a Solr-based deduplication system can achieve a similar deduplication quality to the fingerprint-based system currently employed. Finally, we introduce a large scale deduplication ground truth dataset that we hope will be useful to others tackling deduplication.

1. INTRODUCTION

Mendeley[1] is a platform aimed at making it easier for researchers to keep on top of their fields. Amongst many other features, users can add references and papers to their libraries, tag and annotate their papers, upload their libraries to a central website, search a crowd-sourced catalogue currently containing over 40 million entries, obtain recommendations, view related research and obtain personalised recommendations.

In order to support key features such as the catalogue search and recommendations, the user's library entries plus various references imported from sources such as arXiv[2] and PubMed[3] need to be *deduplicated*[4]. In other words, Mendeley needs to spot when two users add the same paper to their libraries or when an import from an external source adds a paper already known to the system.

Whilst one can perform quick checks to detect duplicates based on filehashes when a PDF is attached to an entry or to check any unique identifiers associated with an incoming entry, it is not always possible since an entry may be unknown to the system and may not have any identifiers attached to it. Given that the metadata may also be noisy, this means Mendeley often has to compare metadata records to see how similar they are in order to determine whether entries are duplicates or not. This paper focusses on:

- Primarily, how to automatically generate large data sets to use for evaluating or training of systems for deduplicating metadata entries from Mendeley's database. This addresses one problem we had when it came to evaluating how well Mendeley's deduplication is working, namely that there was a lack of large scale deduplication data sets relevant to the task. The largest data sets we could find involved only hundreds documents which is a very small sample size compared to the over 40 million entries in Mendeley's catalogue.
- Secondarily, exploring an alternative means of metadata-based deduplication from the one currently used in Mendeley, including exploring a different means of selecting candidate duplicates and different similarity

*Dr Hammerton was a Senior Data Mining Engineer at Mendeley LTD, London, UK from September 2009 until January 2012. He now works as a Research Engineer at Visual DNA, London, UK

†Michael Granitzer is Professor for Media Informatics at the University of Passau. Until January 2012, he was an Assistant Professor at the Know Center, Technical University of Graz, Austria.

‡Dan Harvey was a Data Mining Engineer at Mendeley from September 2009 to December 2011. He now works as a Data Architect at State LTD, London, UK.

[1] http://www.mendeley.com/
[2] http://www.arxiv.com/
[3] http://www.ncbi.nlm.nih.gov/pubmed/
[4] Deduplication is also known by other names such as near duplicate detection and record linkage

metrics. This secondary focus both addresses the desire to improve the performance of Mendeley's deduplication system and also helps to ensure that any findings concerning different ways of generating data sets are applicable to different similarity metrics and different means of candidate duplicate selection.

The main contributions of this work are:

- A methodology for generating data sets for deduplicating bibliographic records from a set of noisy records carrying identifiers and the "ground truth" records for those identifiers or from records with files attached where the filehash can be used as an identifier.
- Evidence that straightforward random sampling of duplicates and non duplicates may be insufficient for generating good datasets for deduplication.
- Announcing a new data set comprising several hundred thousand documents for deduplicating bibliographic records.
- We establish that an alternative approach to candidate duplicate selection, based on Solr[5] title searches, can achieve as good a deduplication quality as the fingerprint based approach currently in use.

The remainder of this paper continues as follows: **Section 2** describes the problem of deduplicating bibliographic records and Mendeley's deduplication system. **Section 3** presents an analysis of the user library entries carrying arXiv or PubMed identifiers and their similarity to the entries imported from those two sources. **Section 4** presents the results of generating "ground truth" datasets via random selection of duplicate and (probable) non duplicate pairs. **Section 5** presents the results of attempts to produce more challenging datasets by focussing on more informative pairs. **Section 6** reports on experiments with the datasets generated during the work for section 5; **Section 7** discusses related work on deduplication. Finally, **Section 8** presents the conclusions of this paper plus some ideas for future work.

2. MENDELEY'S DEDUPLICATION OF BIBLIOGRAPHIC RECORDS

In order to generate a catalogue of publications from users' libraries, Mendeley needs to determine whether two records refer to the same publication, for a database of over 130 million user documents and 20 to 30 million references imported from external sources such as arXiv and PubMed. Since the PDF (or other document format file) associated with copies of a paper may not be byte-for-byte identical, identifiers such as arXiv or PubMed IDs are not always available and may have been erroneously extracted from the paper and the metadata may have been incorrectly extracted or entered by the user, this deduplication problem is not trivial. The challenge is similar to the problem tackled by systems for citation matching [7] and systems for performing record linkage across different databases. See Elmagarmid, Ipeirotis and Verykios [5] for a review of the literature on deduplication in databases.

Mendeley's near real-time[6] automated deduplication system tackles this problem as follows:

1. If a file is attached to an incoming entry or the entry has an identifier such as a PubMed ID we check if the file or identifier has a catalogue entry associated with it and, in the case of an identifier, whether the metadata is sufficiently similar to be deemed a duplicate. If so the entry is treated as a duplicate.
2. Failing step 1, the catalogue entry with the most similar core metadata to the incoming entry is looked up and if it is sufficiently similar, the incoming entry is deemed to be a copy. Otherwise it is deemed to be a new entry. The retrieval of the most similar entry is performed as follows:
   (a) A 64-bit fingerprint is generated from the entry's title via Charikar's simhash method [3] from 64-bit hashes of the character bi- and tri- grams occurring within each title keyword. All entries with fingerprints <= 3 bits different from the query fingerprint are then retrieved.
   (b) The closest entry is chosen according to the following similarity metric, S:
   $$S = \frac{21T + 7P + 2Y}{30} \quad (1)$$
   where T is the title similarity, P is the publication similarity and Y is the year similarity. T and P are both computed as the jaccard similarity between the two sets of tokens for the field values being compared. Y is computed as $1/(1 + d/4)$ where d = the absolute difference between the 2 years. [7]
   (c) If the closest entry has $S >= 0.4$ it is deemed to be a duplicate (set lower than when checking identifiers). This threshold and the earlier one were also arrived at after some tuning in the light of system performance.

Prior to comparing the metadata in this workflow, the metadata entries are cleaned as follows: punctuation and other non alphanumeric characters are stripped; alphabetic characters are turned to lower case and finally the result is tokenized on whitespace. The metadata for each cluster of duplicates is aggregated via a regular batch job to generate/update the metadata for the corresponding catalogue entries. Details of this aggregation process are beyond the scope of this paper however.

[5]Solr is a distributed search engine. See: `http://lucene.apache.org/solr/`

[6]Documents are deduplicated as they arrive after users add them to their libraries, or as bulk importers process entries imported from external sources.

[7]We started with a set of weights, and a formula including authors, reflecting our intuition that the title was more important than the publication and authors which in turn are more important than the year. These weights were then tuned in the light of system performance. We dropped authors due to the field being noisy, but the present work suggests it would benefit to reintroduce them.

This set up was inspired by the use of fingerprints for indexing web pages described by Manku, Jain and Sarma [9], but some details were arrived at somewhat arbitrarily (e.g. the features for fingerprinting and the similarity metric) and then tuned in the light of the system's performance. This approach seemed to work well enough when the initial catalogue was launched with 3 to 5 million entries in the catalogue. The catalogue has since grown to over 40 million entries (generated from over 150 million source entries), and a more efficient and more effective means of deduplication is required to make the catalogue cleaner and support future growth.

An alternative approach to comparing metadata that we explored involves using Solr, for step 2 above, which we hope will be more efficient. This replacement performs the metadata based deduplication as follows:

1. The title bi-grams for an incoming entry are extracted and those consisting solely of stop words discarded.
2. The bi-grams are then used to find the N catalogue entries with the most similar titles via a Solr search with the bi-grams as the query. The Solr index stores the core metadata for each catalogue entry indexed by the title bi-grams.
3. The most similar of the N entries is chosen as the potential duplicate using the same similarity metric as before and the duplicate/new entry decision taken on the basis of the same threshold.

Note that for the experiments that follow, a dump of Mendeley's source documents taken on the 20th July, 2011 was used and that the filehash and identifier matching were disabled to concentrate on the performance of the metadata deduplication. Finally, for brevity subsequent sections will talk loosely of "documents" where strictly speaking we should be talking about metadata records/entries.

3. MENDELEY'S DATA: ANALYSIS OF ARXIV AND PUBMED RECORDS

Because the imported documents are themselves ground truth metadata, if we ensure that user documents carrying an arXiv identifier or PubMed identifier do not differ too much from the imported metadata then we can be certain that the user documents are noisy versions of the imported metadata. This insight enables us to generate the ground truth datasets for these cases. To determine how much noise there is, we first decided to compute the distribution of the similarities between the user documents carrying arXiv and PubMed identifiers and the corresponding imported documents. The distribution of similarities was computed for 3 different similarity metrics:

- **Default similarity**. The similarity metric currently used by Mendeley's deduplication system, as described in section 2, step 2b.
- **"Citation String" similarity**. This similarity metric involves concatenating the title, authors (ordered alphabetically), publication and year of an entry together then comparing the edit distance of the resulting token strings for each entry. This is converted to a similarity score between 0 and 1 by computing the actual edit distance as a proportion of the maximum possible distance.
- **The title similarity.** This metric involves computing the Jaccard similarity over the 2 sets of tokens from each title, i.e. the size of the intersection of the 2 sets divided by the size of the union of the 2 sets.

Table 1 shows the similarity distributions for the user documents with PubMed identifiers (PMIDs). Over 92% of documents fall in the range $[0.9, 1.0]$ for each similarity metric and the vast bulk of documents have a similarity score of $>= 0.7$. This shows that there is very little noise with the user documents that carry PMIDS. Table 2 shows the distributions for the user documents carrying arXiv identifiers. The main thing to note here is that there is more noise in this dataset than with the PubMed documents. However, even here the bulk of cases have similarity $>= 0.6$ for all 3 metrics. Note that for the default metric, the apparent cap at [0.7,0.8) is due to missing publication fields.

In both cases, the user documents are fairly similar (or very similar) to the imported documents, thus we'd expect that deduplication of either the arXiv or PubMed documents should yield high performance.

4. GENERATING RANDOM "GROUND TRUTH" DATA SETS

Due to the size of the Mendeley's database, we cannot indicate whether every pair of documents is a duplicate or not. To create a representative data set, we sampled documents from the arXiv and filehash data. We started off by employing a random sampling strategy as follows:

1. Obtain all user documents carrying a certain type of identifier (e.g. filehash or arXiv ID). Some filtering was done to ensure that the documents were all noisy versions of the same document. For the arXiv documents we ensured that we got a 80% levenshtein similarity on a "signature" consisting of the title keywords plus first author surname. For the filehash data set, see section 4.2 below for more details.
2. Group the documents by the identifier type, e.g. by their arXiv ID.
3. Randomly choose a document in a group and then:
   (a) Choose a second random item from the group to make up the duplicate pair.
   (b) Choose a random item from a different group to make up the non duplicate pair.
4. Repeat step 3 until you have got enough pairs, ensuring no pairs are repeated.

This was done for both the arXiv documents and documents carrying a filehash, with data sets of 500K duplicates and

Similarity range	Default similarity	Citation string similarity	Title similarity
[0.0, 0.1)	36,274 (0.26%)	47,684 (0.34%)	13,205 (0.10%)
[0.1, 0.2)	12,746 (0.09%)	17,173 (0.12%)	23,690 (0.17%)
[0.2, 0.3)	13,168 (0.09%)	14,415 (0.10%)	38,822 (0.28%)
[0.3, 0.4)	22,121 (0.16%)	21,417 (0.15%)	9,497 (0.07%)
[0.4, 0.5)	7,160 (0.05%)	39,177 (0.28%)	6,995 (0.05%)
[0.5, 0.6)	6,742 (0.05%)	89,483 (0.64%)	7,435 (0.05%)
[0.6, 0.7)	9,140 (0.07%)	156,722 (1.13%)	8,245 (0.06%)
[0.7, 0.8)	324,578 (2.34%)	244,746 (1.76%)	10,205 (0.07%)
[0.8, 0.9)	404,460 (2.91%)	436,719 (3.15%)	18,658 (0.13%)
[0.9, 1.0]	13,043,650 (93.96%)	12,812,503 (92.31%)	13,743,287 (99.01%)

Table 1: Similarity distributions computed for user documents carrying PubMed identifiers.

Similarity range	Default similarity	Citation string similarity	Title similarity
[0.0, 0.1)	20,420 (4.46%)	20,061 (4.39%)	9,119 (1.99%)
[0.1, 0.2)	5,639 (1.24%)	7,998 (1.75%)	10,574 (2.31%)
[0.2, 0.3)	6,678 (1.46%)	12,600 (2.75%)	6,579 (1.44%)
[0.3, 0.4)	11,560 (2.53%)	19,966 (4.36%)	4,405 (0.96%)
[0.4, 0.5)	16,248 (3.55%)	29,251 (6.39%)	5,572 (1.22%)
[0.5, 0.6)	20,430 (4.47%)	51,013 (11.15%)	7,979 (1.74%)
[0.6, 0.7)	15,399 (3.36%)	61,043 (13.34%)	8,546 (1.87%)
[0.7, 0.8)	361,069 (78.93%)	62,036 (13.56%)	6,672 (1.46%)
[0.8, 0.9)	0 (0%)	63,915 (13.97%)	7,925 (1.73%)
[0.9, 1.0]	0 (0%)	129,560 (28.32%)	390,071 (85.28%)

Table 2: Similarity distributions computed for user documents carrying arXiv identifiers.

500K non duplicates (probable non duplicates in the case of the filehashes). However we found that the system gave very high performance on the resulting data sets (Fscores of over 0.99!) which is in contrast to our knowledge of deduplication errors in the catalogue. The remaining subsections detail this unexpectedly high performance.

4.1 The "random" arXiv dataset

A set of 410,535 user documents with arXiv identifiers was first created, validated by checking that their signatures have a levenshtein similarity computed over their characters of at least 80%, where the signature includes the title keywords and final token from the first author surname. This validation step was performed to remove cases where the incorrect identifier had been attached to a user document and cases where the metadata was just noise. By looking at the most similar cases rejected and the least similar cases accepted, the threshold of 80% was chosen.

500K duplicate and 500K non duplicate pairings were drawn from the validated documents and these pairings collectively covered 400,336 documents.

4.1.1 Deduplication via fingerprint lookups

These experiments involved deduplicating the 410,535 validated documents and then evaluating with the dataset of duplicate/non duplicate pairs. The similarity distributions of the duplicates and non duplicates are illustrated in the table 3 below. The distributions were computed for the default similarity and an altered similarity where the title and publication similarities were computed using the character levenshtein similarity. There is only a small overlap between the duplicate and non duplicate distributions, particularly for the altered similarity. From this we can deduce that we should get high performance with this dataset and that the altered similarity is a better metric for this dataset.

Indeed, when deduplicating the user documents with arXiv Ids, and then evaluating with the arXiv data set, we do get good results. Table 4 illustrates the precision, recall, Fscore and the number of identifiers per cluster and the number of clusters per identifier, and the upper part shows the results for this experiment. These metrics would all ideally have a value of 1. The number of pairs skipped needs explanation. It is possible when querying Mendeley's lookup service that it doesn't find a document that matches the query — the evaluation performed uses the full deduplication infrastructure albeit with a subset of the data. The pairs that are skipped involve cases where one or other document fails to be found. The problem here is that the catalogue entries resulting from deduplication are insufficiently similar to find a match. Thus we count these errors separately.

However when we cluster the imported documents themselves, although we get precision, recall, Fscore of 1.0, there are 17,573 duplicate pairs skipped and 25,188 non duplicate pairs skipped due to not being able to find one of the documents. This is because the 689,944 arXiv documents have been mapped to 663,545 "canonical" documents[8] when they

[8]When we create a cluster of user documents that we think refer to the same underlying document we also select metadata to represent the underlying document. This metadata constitutes the canonical document for that cluster. In the experiments here, but unlike the live system, the metadata

Similarity	Default similarity		Altered Similarity	
	Duplicate	Non-Duplicate	Duplicate	Non-Duplicate
[0.0, 0.1)	45 (0.01%)	438,583 (87.72%)	0 (0%)	5,522 (1.10%)
[0.1, 0.2)	130 (0.03%)	58,004 (11.60%)	0 (0%)	286,425 (57.28%)
[0.2, 0.3)	321 (0.06%)	2,660 (0.53%)	0 (0%)	198,698 (39.74%)
[0.3, 0.4)	1,482 (0.30%)	729 (0.15%)	1 (0.00%)	7,539 (1.51%)
[0.4, 0.5)	6,537 (1.31%)	23 (0.00%)	22 (0.00%)	1,781 (0.36%)
[0.5, 0.6)	14,743 (2.95%)	1 (0.00%)	1,256 (0.25%)	33 (0.01%)
[0.6, 0.7)	13,931 (2.79%)	0 (0%)	14,493 (2.90%)	1 (0.00%)
[0.7, 0.8)	319,897 (63.98%)	0 (0%)	320,159 (64.03%)	1 (0.00%)
[0.8, 0.9)	7,661 (1.53%)	0 (0%)	24,027 (4.80%)	0 (0%)
[0.9, 1.0]	135,253 (27.05%)	0 (0%)	140,042 (28.01%)	0 (0%)

Table 3: Similarity distributions of the "random" arXiv dataset

							Pairs skipped	
Docs	Lookup	P	R	F1	IDs/cluster	Clusters/ID	Duplicate	Non duplicate
Validated	fingerprint	1.00	0.98	0.99	1.025	1.019	2 (0.00%)	15 (0.00%)
Raw	fingerprint	1.00	0.98	0.99	1.071	1.134	2 (0.00%)	15 (0.00%)
Validated	Solr	1.00	0.98	0.99	1.212	1.05	552 (0.11%)	3,705 (0.74%)
Imported	Solr	1.00	1.00	1.00	n/a	n/a	16,523 (3,30%)	23,371 (4.67%)

Table 4: Results of deduplicating "random" arXiv data via fingerprint and Solr lookups. P=Precision, R=Recall, F1=Fscore.

should have been assigned a cluster each. So 26K (3.67%) documents have been incorrectly clustered together. Looking through the clusters of size >= 3 that were generated, there were 4 main causes of this problem:

1. Some document groups had some similar keywords in their titles, yielding high similarity scores but were obviously different documents to a human reader.
2. Some document groups had near identical titles but were different documents to a human reader, e.g. "Proceedings of the 16th Workshop..." vs "Proceedings of the 17th Workshop...".
3. Some documents had identical titles but were otherwise different.
4. Some documents had identical metadata entries in the core fields used for the similarity metric but were different.

These problems illustrate the challenges that need to be tackled in constructing a deduplicated catalogue of research publications from a mix of user library entries and imported data as Mendeley is doing. Whilst improving the similarity metric will help with the first case, it is clear that judging by similarity alone is problematic. You need to allow for noise but also to be able to distinguish cases where small differences are actually crucial (case 2 above). To help quantify the problem we counted how often identical titles appear in source documents. For the arXiv imported documents here there were 686,621 titles of which 2,550 were shared (accounting for 5449 documents, 424 other documents actually had no titles!). Thus a title had a roughly 1 in 269 chance of being shared amongst multiple arXiv documents.

chosen is the metadata of the user document that started the cluster.

4.1.2 Deduplication via Solr lookups

For this experiment, the Solr based prototype replacement for the lookup service was used for clustering instead. Clustering user documents with validated arXiv ids and evaluating with arXiv dataset give the results in the lower part of table 4.

The 410,535 validated user docs were reduced to 156,933 canonical documents, 23,050 fewer than with the fingerprinting. When clustering imported documents, the more permissive clustering becomes more apparent, with 16,523 duplicate pairs and 23,371 non duplicate pairs skipped. Overall the accuracy was maintained but more pairs were skipped due to documents not being found. The 689,944 imported documents were reduced to 592,489 canonical documents, with 97,455 documents (19.49%) clustered together. This is a consequence ironically of Solr being better able to ensure all the documents that have a similarity greater the threshold to the current document appear in the list of candidates. The solution is be stricter with the deduplication, e.g. via a higher threshold or stricter similarity function.

4.2 The "random" filehash dataset

First a selection of user documents with filehashes was created. Some account needed to be taken of some mislabelling of files that occurs due to both user and system errors. The approach was to group the titles for a filehash based on their similarity as follows:

1. For each group of documents sharing a filehash:
   (a) Select the first remaining title and group it with all titles with levenshtein similarity >= 0.8.
   (b) Repeat step 1a until there are no more titles.
2. Select the filehash if the largest title group forms a majority of the documents with that filehash.

After selecting 1,000,000 user documents with filehashes, a data set of 389,799 docs with 500K duplicates and (probable) non duplicates was created via the same random method as before, except care was taken to ensure each pair of non duplicates was not identical. The similarities are distributed as shown in table 5. Note that the overlap in the two distributions is rather small and thus high performance should also be possible with this dataset. Also, the spikes at $[0.7, 0.8)$ are due to the high weight ($21/30 = 0.7$) given to the title in the similarity metric.

Similarity	Duplicate	Non-Duplicate
$[0.0, 0.1)$	1,466 (0.29%)	446,656 (89.33%)
$[0.1, 0.2)$	4,162 (0.83%)	52,040 (10.41%)
$[0.2, 0.3)$	4,408 (0.88%)	988 (0.20%)
$[0.3, 0.4)$	5,711 (1.14%)	119 (0.02%)
$[0.4, 0.5)$	7,284 (1.46%)	2 (0.00%)
$[0.5, 0.6)$	9,210 (1.84%)	1 (0.00%)
$[0.6, 0.7)$	10,100 (2.02%)	0 (0%)
$[0.7, 0.8)$	97,705 (19.54%)	194 (0.04%)
$[0.8, 0.9)$	25,720 (5.14%)	0 (0%)
$[0.9, 1.0]$	334,234 (66.85%)	0 (0%)

Table 5: Similarity distributions of "random" filehash dataset with the default metric

4.3 Optimal threshold ("sweet spot") experiments

Here the results of experiments to find the optimal thresholds for a set of similarity metrics and each of the datasets are presented. This analysis enables evaluation of a similarity metric and also an assessment of how easy the datasets are. The approach is as follows:

- for each threshold 0.0, 0.1, ... up to 1.0:
  - compute precision, recall and Fscore with the dataset and similarity metric assuming similarity scores greater than or equal to the thresholds are duplicates.

The similarity metrics used below are:

- The default metric as used in the deduplication system.
- The title similarity metric used in the deduplication system.
- Similarity metric where the title and publication similarities are the character levenshtein similarities i.e. where the edit distance is computed and divided by the length of the longer string (the max edit distance possible), otherwise the metric is the same as the default.
- Default similarity modify to add Jaccard similarity over the author strings with weight 10 (score out of 40 now instead of 30).

4.3.1 "Random" arXiv dataset sweet spots

Table 6 shows the sweet spot analysis for the default similarity metric used in the current deduplication system. The peak Fscore is achieved with a threshold of 0.3 and is very high — 0.9988! Yet we know that the system is too permissive at clustering the imported arXiv documents together which would imply raising the threshold (or otherwise making the deduplication more strict). However note that the differences in Fscore for the thresholds of 0.2 upto 0.5 are very small, which suggests that there is scope for favouring the higher precision achieved at 0.4 or 0.5.

Threshold	Precision	Recall	Fscore
0.0	0.5000	1.0000	0.6667
0.1	0.8905	0.9999	0.9421
0.2	0.9932	0.9996	0.9964
0.3	0.9985	0.9990	**0.9988**
0.4	1.0000	0.9961	0.9980
0.5	1.0000	0.9832	0.9915
0.6	1.0000	0.9541	0.9765
0.7	1.0000	0.9266	0.9633
0.8	1.0000	0.2858	0.4445
0.9	1.0000	0.2705	0.4258
1.0	1.0000	0.1772	0.3011

Table 6: Sweet spot for "random" arXiv data set, default similarity

Table 7 shows the similarities purely of the titles, using the default title similarity metric. Note that performance has dropped with a peak Fscore of "only" 0.9793. This is still high and reflects the fact that the title is the major discriminating feature here, but also shows that some gain comes from using the other metadata in the default metric. Table 8 shows the similarities obtained using the altered similarity metric from earlier. This time we manage an Fscore of 1.0000! Finally table 9 shows the sweet spot when we add the authors to the default similarity, weight 10. The authors seem to provide a benefit raising the Fscore to 0.9996 from 0.9988, but the difference is very small.

Threshold	Precision	Recall	Fscore
0.0	0.5000	1.0000	0.6667
0.1	0.8247	0.9886	0.8992
0.2	0.9747	0.9812	0.9780
0.3	0.9984	0.9609	**0.9793**
0.4	0.9999	0.9327	0.9651
0.5	1.0000	0.9077	0.9516
0.6	1.0000	0.8815	0.9370
0.7	1.0000	0.8593	0.9243
0.8	1.0000	0.8273	0.9055
0.9	1.0000	0.7707	0.8705
1.0	1.0000	0.7508	0.8577

Table 7: Sweet spot for "random" arXiv data set, default title similarity

4.3.2 "Random" filehash dataset sweet spots

Table 10 shows the "sweet spot" for the "random" filehash dataset. Again we get a very high Fscore of 0.9930! This is despite the filehash data set being expected to be noisy. Table 11 shows the "sweet spot" for the altered similarity from earlier. Unlike the arXiv dataset the performance drops

Threshold	Precision	Recall	Fscore
0.0	0.5000	1.0000	0.6667
0.1	0.5028	1.0000	0.6691
0.2	0.7063	1.0000	0.8278
0.3	0.9816	1.0000	0.9906
0.4	0.9964	1.0000	0.9982
0.5	0.9999	1.0000	**1.0000**
0.6	1.0000	0.9976	0.9988
0.7	1.0000	0.9843	0.9845
0.8	1.0000	0.3280	0.4939
0.9	1.0000	0.2800	0.4375
1.0	1.0000	0.1770	0.3007

Table 8: Sweet spot for "random" arXiv data set, char levenshtein similarity for title, publication, otherwise same as default

Threshold	Precision	Recall	Fscore
0.0	0.5000	1.0000	0.6667
0.1	0.9637	1.0000	0.9815
0.2	0.9970	0.9999	0.9984
0.3	0.9999	0.9993	**0.9996**
0.4	1.0000	0.9957	0.9978
0.5	1.0000	0.9855	0.9927
0.6	1.0000	0.9231	0.9600
0.7	1.0000	0.8373	0.9114
0.8	1.0000	0.6282	0.7717
0.9	1.0000	0.2426	0.3905
1.0	1.0000	0.1595	0.2751

Table 9: Sweet spot for "random" arXiv data set, default similarity with authors added, weight 10

here, though it's still a high 0.9886. Table 12 shows the sweet spot analysis for the default similarity with authors added, weight 10. With a peak Fscore of 0.9929 this is essentially the same performance as we got with out the authors. The 0.0001 drop in performance is so small as to make it unclear whether it reflects an underlying real drop in performance.

With these datasets, it seems there is little room for improving the performance given the high performance being achieved in the first place. However this contrasts with our knowledge of clustering errors (both missed duplicates and incorrectly clustered documents) in Mendeley's catalogue and other knowledge we have about deduplication. In the

Threshold	Precision	Recall	Fscore
0.0	0.5000	1.0000	0.6667
0.1	0.9033	0.9979	0.9479
0.2	0.9974	0.9887	**0.9930**
0.3	0.9996	0.9799	0.9895
0.4	0.9996	0.9685	0.9838
0.5	0.9996	0.9540	0.9762
0.6	0.9996	0.9356	0.9665
0.7	0.9996	0.9155	0.9557
0.8	1.0000	0.7201	0.8373
0.9	1.0000	0.6691	0.8018
1.0	1.0000	0.6093	0.7572

Table 10: Sweet spot for "random" filehash data set, default similarity

Threshold	Precision	Recall	Fscore
0.0	0.5000	1.0000	0.6667
0.1	0.5287	0.9999	0.6915
0.2	0.6537	0.9938	0.7887
0.3	0.9763	0.9855	0.9809
0.4	0.9990	0.9745	**0.9866**
0.5	0.9996	0.9609	0.9799
0.6	0.9996	0.9478	0.9730
0.7	0.9996	0.9336	0.9655
0.8	1.0000	0.7676	0.8685
0.9	1.0000	0.6924	0.8838
1.0	1.0000	0.6073	0.7557

Table 11: Sweet spot for "random" filehash data set, character levenshtein similarity for strings, otherwise same as default

Threshold	Precision	Recall	Fscore
0.0	0.5000	1.0000	0.6667
0.1	0.9694	0.9952	0.9819
0.2	0.9991	0.9869	**0.9929**
0.3	0.9996	0.9781	0.9887
0.4	0.9996	0.9678	0.9834
0.5	0.9996	0.9554	0.9770
0.6	1.0000	0.8986	0.9466
0.7	1.0000	0.8642	0.9271
0.8	1.0000	0.7756	0.8736
0.9	1.0000	0.6232	0.7678
1.0	1.0000	0.5557	0.7144

Table 12: Sweet spot for "random" filehash data set, character levenshtein similarity for strings, otherwise same as default

next section, we try to generate more challenging data sets.

5. GENERATING INFORMATIVE "GROUND TRUTH" DATA SETS

As was seen in Section 4, the data sets produced via simply choosing duplicates and non duplicates at random turned out to be very easy. Is it just that the sets of data chosen are easy to learn or is there a problem with the method we are using to generate the data sets?

There is a case for the latter interpretation. Consider a set of 100 documents of which 10 are duplicates. The maximum number of duplicate pairs in this example is 45, whilst the total number of pairs is 4,950. For the case of 1000 documents with 100 duplicates, the figures are 4,950 and 499,500. These numbers grow quadratically but unless duplicates are very high percentage of the total, the number of duplicate pairs will be a very small proportion of the total number of pairs for data sets with large numbers of documents. Thus random sampling involves sampling very unequal and huge sample spaces. One can sample much more of the duplicate sample space than the non duplicate space and the examples you get back will tend to be very well separated due to sampling only a small proportion of each space in the first place. In short, as pointed out by Saragawai and Bhamidipaty [11], finding informative examples for deduplication is not straightforward. In this section, 3 different approaches to selecting more difficult/informative cases are explored:

Threshold	Precision	Recall	Fscore
0.0	0.5000	1.0000	0.6667
0.1	0.5982	1.0000	0.7486
0.2	0.8980	0.9996	0.9461
0.3	0.9611	0.9985	0.9795
0.4	0.9910	0.9960	**0.9935**
0.5	0.9977	0.9839	0.9907
0.6	0.9989	0.9655	0.9819
0.7	0.9995	0.9466	0.9723
0.8	0.9994	0.2976	0.4587
0.9	0.9997	0.2840	0.4424
1.0	0.9998	0.1852	0.3125

Table 13: 200K + 200K arxiv dataset sweet spot for title similarity sampling

- Section 5.1 reports on sampling non duplicates with a probability based on title similarity. The intuition here is that non duplicates with high similarities are "surprising" non duplicates. One can similarly argue this for low similarity duplicates, however the duplicates are a small proportion of the space so we still randomly select from all duplicates.
- Section 5.2 reports on sampling based on the uncertainty of the title similarity. The idea here is to use several measures of title similarity and to choose pairs with a probability that increases with the divergence between the minimum and maximum similarity.
- Section 5.3 reports on sampling via a Solr title similarity search on the indexed documents. The idea here is to sample both duplicates and non duplicates from the nearest neighbours (by title similarity) of a query document. This should be biased towards the duplicates and any non duplicates in the nearest neighbours are likely to be cases that can confuse a deduplication algorithm.

5.1 Title similarity sampling

For this approach, we group documents by identifier and then emit duplicate pairs and non duplicate pairs, randomly sampling the latter with a probability computed as the square of the levenshtein similarity of the titles. Pairs are additionally only considered if the titles have one keyword in common. The "sweet spot" results for a dataset of 200K duplicate and non duplicate pairs are below. The duplicate pairs were chosen at random with a 20% probability, the window size for selecting non duplicate pairs was 250 — each group of duplicates was considered in turn and a window is randomly selected from the previous entries used for generating non duplicates. Table 13 shows the results from this approach with the arXiv data.

We still get a high Fscore of 0.9935 suggesting the data set isn't much more difficult than the earlier one. A variation on this approach also excluded titles whose fingerprints were > 3bits apart. We were able to generate 200K duplicates, but only 932 non duplicate pairs even after doubling the window size to 500. The Fscore for this dataset peaked at 0.9989 but with so few non duplicates it's doubtful what the value of it is. Running the code again so that all non duplicates whose title fingerprints are <= 3 bits apart are used (i.e. we no longer sample with probability = similarity squared) increased the number of pairs to 6,727. The best Fscore was 0.9978 at threshold 0.4. At this point, we decided to try the approach below.

5.2 Uncertain similarity sampling

For this approach, again we group the documents by identifier and filehash, however here we compute several similarity scores for the title:

- Character levenshtein. The levenshtein distance is computed for the characters, and divided by the maximum possible distance for the two strings concerned.
- Token set similarity. The Jaccard similarity between the two sets of tokens is used.
- Token levenshtein. The levenshtein distance is computed for the tokens in the string, and divided by the maximum possible distance for the two strings concerned.
- Tokens in common. This computes the proportion of non-distinct tokens the strings have in common.

Pairs are sampled based on the difference between the maximum and minimum of these similarities, the idea being that such pairs should in theory be difficult or informative pairs to decide on due to the uncertainty over their similarities. However in order to counter the imbalance between non duplicate and duplicate pairs, the duplicates and non duplicates were selected with the following probabilities:

- Duplicates were selected with the probability $d = 0.5 + 0.5m$ where m is the maximum difference between similarity metrics.
- Non duplicates were selected with probability d^2a where a is the average similarity of the metrics above.

This set up was chosen after trying several possibilities including sampling both duplicates and non duplicates with probability d and sampling non duplicates with probability d^2. A window size of 400 was used for generating the non duplicate pairings. The results of the "sweet spot test" are shown in table 14 for a dataset of 200K of duplicate and 200K non duplicate pairings. Again we get a very high Fscore of 0.9969.

5.3 Sampling via Solr search

Here we store user documents in a Solr index in a similar manner to the index of catalogue entries used for the prototype Solr lookup service. We generate the dataset by presenting each document's title bi-grams in turn as a query to Solr and looking through the returned results for duplicates and non duplicates of the query document, outputting the pairs until we have enough of each. The rationale is that we are sampling directly from the nearest neighbours of a document, and if the neighbours include non duplicates, then the problem of deciding whether a document is duplicate or not is likely to be harder than a random choice. Indeed we are

Threshold	Precision	Recall	Fscore
0.0	0.5000	1.0000	0.6667
0.1	0.7555	0.9999	0.8607
0.2	0.9773	0.9998	0.9884
0.3	0.9923	0.9990	0.9956
0.4	0.9993	0.9946	**0.9969**
0.5	0.9999	0.9819	0.9908
0.6	1.0000	0.9561	0.9819
0.7	1.0000	0.9347	0.9663
0.8	1.0000	0.3558	0.5248
0.9	1.0000	0.3417	0.5093
1.0	1.0000	0.2472	0.3965

Table 14: Sweet spot for 200k + 200k arXiv data set generated using uncertainty sampling

MR	Best F1	Duplicates	Nonduplicates	Threshold
1	0.4548	2,601	9,735	0.7
2	0.8383	4,441	40,000	0.7
3	0.9449	29,015	40,000	0.5
4	0.9648	40,000	40,000	0.5
5	0.9681	40,000	40,000	0.5
6	0.9686	40,000	40,000	0.5
7	0.9692	40,000	40,000	0.5
8	0.9700	40,000	40,000	0.5
9	0.9701	40,000	40,000	0.5
10	0.9704	40,000	40,000	0.5

Table 15: Using arXiv data and solr search to generate duplicate/non duplicate data. "MR" is the minimum required rank of a non duplicate.

mimicking the choices being made during the deduplication itself.

One variable here is the number of results returned, the more returned, the larger the dataset that can be generated, but also the less focussed on the difficult cases the dataset will be. The fewer returned, the more difficult the focus is and the smaller the dataset that can be generated. Another variable is to consider the minimum rank of the highest ranked non duplicate in the results. If we require that there must be a non duplicate with a specified rank than the higher the rank, the more difficult the dataset will be. Here we explore the number of results being fixed at 10, and varying from generating pairs regardless of whether non duplicates appear in the results to generating pairs only when the highest ranked non duplicate is of rank 1. Note that one of the results returned may be the query document itself, below these self comparisons are included when deciding what the rank of each document is, but excluded from the dataset.

Table 15 reports the minimum non duplicate rank required, best Fscore found, number of duplicate and non duplicate pairs and the threshold where the best result was found, based on finding the "sweet spot" for the default similarity metric. Also the code aimed to generate 40K pairs of each type. Altering the code to ignore the copy of the query document in the results, we ran a similar experiment reported in table 16, but this time with minimum non dup ranks of 1,3,5,7 and 9. The performance improved as expected.

MR	Best F1	Duplicates	Nonduplicates	Threshold
1	0.8066	3,651	40,000	0.7
3	0.9648	40,000	40,000	0.5
5	0.9686	40,000	40,000	0.5
7	0.9700	40,000	40,000	0.5
9	0.9706	40,000	40,000	0.5
none	0.9741	40,000	40,000	0.5

Table 16: Using arXiv data and solr search to generate duplicate/non duplicate data, ignoring copy of query document in results.

Data set	Best F1	Threshold
"informative" arXiv	0.9774	0.5
"informative'; filehash	0.9196	0.4

Table 17: "Sweet spot" analysis for the "informative" data sets and default similarity metric.

Finally we tried generating an arXiv dataset of 500K duplicates and non duplicates this way, but found we couldn't get enough non duplicates with 10 results being returned from Solr unless we dropped all restrictions on the presence of non duplicates in the results. In doing so we got a dataset with 330,668 documents. The "sweet spot" code gave the results reported in the first row of table 17. This is the most difficult set so far, suggesting this method successfully focuses on some more difficult cases. However an Fscore of 0.9774 is still a good Fscore, which is more evidence the arXiv data is actually quite easy to deduplicate. It may be possible to generate more difficult datasets than this by sampling from the search results in the ways tried in the earlier approaches but this option remains to be explored.

We have released this data set publicly in the hope it may be of use to other researchers tackling deduplication. You can download the data set from the first author's Mendeley profile[9] or from the TEAM project web site[10].

We also generated a new filehash dataset with the above approach (after sampling from filehashes with at least one duplicate). A dataset of 642,896 documents was produced with 500K duplicate and 500K probable non duplicate pairs. No attempt was made to take into account possible duplicates with different filehashes and row 2 of table 17 displays the results. This is the most difficult dataset so far — the best Fscore is only 0.9196, but still good. The similarities for this data set are distributed as shown in table 18.

We had hoped to release this data set too. However Mendeley users can request that a library entry does not appear in the public catalogue to enable them to store unpublished work or confidential reports privately and we realised we hadn't taken this into account during generation and thus we couldn't be certain the data set did not contain such entries.

Note that because no attempt was made to eliminate duplicates in the probable non duplicate part of this dataset,

[9] `http://www.mendeley.com/profiles/james-hammerton/`

[10] `http://team-project.tugraz.at/2012/01/13/new-de-duplication-data-set-published/`

Similarity	Duplicate	Non-Duplicate
[0.0, 0.1)	401 (0.08%)	14,798 (2.96%)
[0.1, 0.2)	3,971 (0.79%)	237,429 (47.49%)
[0.2, 0.3)	8,277 (1.66%)	150,115 (30.02%)
[0.3, 0.4)	11,836 (2.37%)	38,918 (7.78%)
[0.4, 0.5)	15,927 (3.18%)	15,595 (3.12%)
[0.5, 0.6)	21,345 (4.27%)	5,212 (1.04%)
[0.6, 0.7)	21,783 (4.36%)	1,527 (0.30%)
[0.7, 0.8)	157,914 (31.58%)	25,828 (5.17%)
[0.8, 0.9)	46,481 (9.30%)	951 (0.19%)
[0.9, 1.0]	212,065 (42.41%)	9,627 (1.92%)

Table 18: Similarity distributions of "informative" filehash dataset with the default metric

it's likely that a high proportion of the 7% of probable non duplicates with similarity >= 0.7 are actually duplicates. It would be interesting to see what impact e.g. explicitly excluding pairs with identical titles would have. However for the bulk of the data set this shouldn't make much impact and the "sweet spot" threshold of 0.4 found above is unlikely to be altered.

6. EXPERIMENTS WITH "INFORMATIVE" DATASETS

6.1 Deduplication with "informative" arXiv dataset

Table 19 shows the results of deduplicating 410,535 user documents with validated arXiv ids to produce a set of "canonical" documents and evaluating with the "informative" arXiv dataset. The Settings column indicates whether fingerprint (Fprint) or Solr lookups were used for clustering and if Solr was used, how many results were returned in the query (e.g. Solr-40 means 40 results were returned). Otherwise the table shows precision, recall, F1 and the number of pairs skipped. The fingerprinting did best, but the Solr performance is not far behind. Raising the threshold resulted in fewer skips, but lowered performance whilst reducing the number of results returned resulted in increased performance but mixed results for the number of skips.

The difference in performance between the fingerprints and the Solr clustering is small enough that the latter's greater efficiency and scalability win out. It may be that with a modified similarity metric, we can get better performance out of Solr as well, e.g. by altering the title similarity to the levenshtein similarity or by incorporating authors into the metric.

6.2 Deduplication with the "informative" filehash dataset

Table 20 shows the results of deduplicating 642,968 user documents with filehashes to produce a set of "canonical" documents and evaluating with the "informative" filehash dataset. The fingerprint lookup service does worse here than Solr, except on the number of pairs skipped. However, the numbers of pairs skipped is too small a proportion of the dataset in either case to overturn the verdict of the Fscore.

Data set	Metric	Best F1	Threshold
"informative" arxiv	char levenshtein	0.9866	0.6
	default + authors	0.9815	0.5
"informative" filehash	char levenshtein	0.9096	0.5
	default + authors	0.9294	0.4

Table 21: "Sweet spot" analysis for "informative" data sets and alternate similarity metrics.

6.3 Other metrics and the "informative" datasets

We performed the "sweet spot" analysis on both "informative" data sets using the character levenshtein similarity metric and the default similarity metric with authors added, weight 10. Table 21 summarises the results. For the "informative" arXiv dataset and the similarity metric using character levenshtein similarity for strings, this clearly improves over the default similarity metric, yielding a peak Fscore of 0.9866. For the "informative" arXiv dataset and the default similarity metric, again this gives a boost over the default metric, giving a peak Fscore of 0.9815. For the "informative" filehash dataset and the default similarity with authors added, this also shows improvement compared to the default metric, the Fscore rising to 0.9294. However for the character levenshtein metric, unlike the arXiv case, the Fscore drops to 0.9096. Whilst we get contradictory performance with the character levenshtein similarity for string fields, clearly adding the authors to the metric improves performance in both data sets and is likely to boost performance generally.

7. RELATED WORK

Bilenko and Mooney [2] tackle the problems of evaluation and of training set construction for duplicate detection and propose two approaches to collecting training data:

- Static-active learning. Here, by using "off the shelf" similarity metrics for each field in a record and selecting records with high similarities across multiple fields a set of candidates can be quickly extracted from a database and then labeled as duplicates or non duplicates by users. The non duplicates found this way are likely to be informative examples whilst the method overall should find a high proportion of duplicates. The technique we apply of using a Solr title search achieves a similar goal of finding potential duplicates and non duplicates in an informative way, but we use the filehash or arXiv identifiers to substitute for the manual labelling. In the absence of such identifiers though, this will provide a useful means of getting started, less expensive than full blown active learning such as that covered in Saragawai and Bhamidipaty [11].
- Weakly labeled non-duplicates. This is where you randomly select documents as probable non duplicates, based on the idea that randomly selected pairs of documents are unlikely to be duplicates. We employed a similar approach to obtain the non duplicates to go with the filehash duplicates where the duplicates were found via the filehash, however without a means of focussing in on informative cases we found this leads

Settings (Fprint/Solr-N, thresh)	Prec	Rec	F1	Canonical docs	Pairs Skipped	
					Duplicate	Non-Dup
Fprint, 0.5	0.9877	0.9751	0.9813	180,951	0 (0.00%)	0 (0.00%)
Solr-40, 0.5	0.9802	0.9760	0.9781	176,118	50 (0.01%)	1,505 (0.30%)
Solr-40, 0.6	0.9907	0.9457	0.9677	185,286	1 (0.00%)	2,202 (0.44%)
Solr-20, 0.5	0.9799	0.9771	0.9785	170,035	61 (0.01%)	1,388 (0.28%)
Solr-10, 0.5	0.9799	0.9775	0.9786	177,040	141 (0.03%)	7,282 (1.46%)

Table 19: Results of deduplication experiments with the "informative" arXiv dataset. NB: Solr-N means Sol search returning N results, e.g. Solr-40 involved returning <= 40 results for each query.

Settings (Fprint/Solr-N, thresh)	Prec	Rec	F1	Canonical docs	Pairs Skipped	
					Duplicate	Non-Dup
Fprint, 0.5	0.9176	0.9124	0.9150	278,329	73 (0.01%)	839 (0.17%)
Solr-10, 0.5	0.9207	0.8779	0.8988	289,127	15 (0.00%)	8 (0.00%)

Table 20: Results of deduplication experiments with the "informative" filehash dataset. NB: Solr-N means Sol search returning N results, e.g. Solr-10 involved returning <= 10 results for each query.

to very easy datasets when combined with randomly selecting duplicate pairs via the filehash.

Bilenko and Mooney also proposed plotting precision/recall curves for evaluation of deduplication. Whilst we report the more traditional Fscore, we also compute the precision and recall and report them alongside the Fscore which should enable similar judgements to be made when performing the "sweet spot" analyses above. A more expensive "sweet spot" evaluation would also involve evaluating the performance of the deduplication system as a whole using different thresholds, however our results suggest that evaluating the metric (or a machine learner) directly at the different thresholds will give a good indication of overall performance of the system when employing the metric as well as indicating a good threshold to use.

We noted earlier that pre-existing relevant public deduplication data sets had hundreds of documents in them at best. For example, the Cora[11] and Citeseer [7] datasets for citation matching (a closely related task) respectively have 1295 citations to 122 papers and 1564 citations to 784 papers and these are the largest publicly available deduplication datasets for research paper metadata that we are currently aware of, outside the datasets generated here. By providing a deduplication data set with hundreds of thousands of metadata records, we hope better performance can be achieved than was hitherto likely with the existing data sets.

Regarding the deduplication problem Mendeley is tackling, the work most directly related to this work is citation matching. Probably the most well known work here is the work behind Citeseer [7, 8, 4], although Google Scholar[12] has presumably had to tackle a similar problem. The earlier Citeseer work demonstrated that a remarkably simple word and phrase matching approach performed sufficiently well for unassisted use in automated citation indexing. Similarly our current approach is also quite simple but has enabled us to build a catalogue of over 40 million papers from user records and imported sources, albeit one that needs some cleaning to be done. The later Citeseer work from Councill *et al* [4] employs Lucene's indexing and fuzzy searching to find candidate duplicates, mirroring our consideration of Solr for this purpose, plus a cluster repair algorithm to correct clustering errors when the canonical metadata changes. At the moment, clustering errors in Mendeley are dealt with by *ad hoc* jobs for identifying missed duplicates and documents that need to be reclustered. It may be worth considering whether something like Citeseer's cluster repair algorithm can be incorporated to improve Mendeley's deduplication system.

A growing body of work [1, 10, 6] demonstrates that machine learning and probabilistic modelling can be used to obtain better performance for adaptive deduplication and citation matching than that achieved via hand crafted similarity metrics and Mendeley's deduplication would benefit from exploring such approaches, whilst the large data sets here should aid in maximising the achievable performance.

Finally with regards deduplication more generally, interesting work has been done on deduplicating a set of records efficiently in batch with MapReduce jobs [13, 12], which contrasts with Mendeley's incremental approach. However, such batch approaches may provide a way of quickly trying out new similarity metrics on large data sets or reclustering the existing data in one fell swoop.

8. CONCLUSIONS

This work has demonstrated that one can generate large data sets for deduplication of bibliographic records via either filehashes (for a set with definite duplicates and probable non-duplicates) or identifiers such as arXiv and PubMed identifiers (for definite duplicates and non duplicates) if you have noisy metadata entries with files or identifiers attached and, in the case of the identifiers, you have access to the ground truth metadata.

It also shows that these datasets are most effectively generated via a title similarity search that yields the nearest neighbours of a given query document from which their du-

[11] `http://www.cs.umass.edu/~mccallum/data/cora-refs.tar.gz`
[12] `http://scholar.google.com/`

plicate or non duplicate status can be easily deduced via the filehash or identifier. We believe these findings are likely to be applicable to other deduplication problems, especially in cases where one or two fields can be used to narrow the search space. We hope other researchers can apply the methodology here to generate ground truth data sets automatically themselves.

Additionally, we have shown that with sources such as arXiv and PubMed, Mendeley users' library entries tend to be similar to the entries imported from these sources, where the user's entry carries the relevant identifier. We further show that a Solr-based lookup system can perform deduplication as effectively as the existing fingerprint-based system Mendeley uses.

With regards to the findings about how the imported documents are currently being clustered (see section 4), given our analysis, the most immediate lessons are:

- to treat arXiv identifiers, and probably most others, as unique identifiers, because even where similar metadata suggest they might be duplicates, this is not actually likely to be the case.
- to improve the similarity metric as far as possible and where possible to do full text fingerprinting so we can more reliably tell duplicates from non duplicates.
- it may be worth doing some parsing of titles to see if small differences when measured via levenshtein or token set similarity are crucial or not.

Some future lines of work are suggested here:

- Applying machine learning to learn a similarity metric. With the ability to generate large scale ground truth datasets in place, a natural progression from this work is to look into applying a machine learning algorithm by training it on such data. This would build on work already performed on smaller scale data sets such as the Cora dataset used by Bilenko and Mooney [1].
- It may be possible to produce more challenging data sets yet by combining the methods explored in Section 5.
- Mendeley's deduplication system is currently fully automated. Whilst we believe the system can be improved significantly whilst remaining fully automated, it may be that for the cleanest possible catalogue crowdsourcing will need to be introduced. Automation would still be required to cover the bulk of cases in order not to overwhelm users with requests to deduplicate, and thus users would only be asked to deduplicate uncertain cases. However, crowd sourcing would enable a high degree of certainty for deduplication decisions, so long as care is taken to deal with the possibility of user error or of people trying to game the system.

Acknowledgements

The authors would like to thank the members of staff at both Mendeley and the Know Center at the Technical University of Graz for their support during this work. This work was partially funded by the European Commission as part of the FP7 Marie Curie IAPP project TEAM (grant no. 251514).

9. REFERENCES

[1] M. Bilenko and R. Mooney. Adaptive duplicate detection using learnable string similarity measures. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 39–48. ACM New York, NY, USA, 2003.

[2] M. Bilenko and R. Mooney. On evaluation and training-set construction for duplicate detection. In *Proceedings of the KDD-2003 Workshop on Data Cleaning, Record Linkage, and Object Consolidation*, pages 7–12, 2003.

[3] M. Charikar. Similarity estimation techniques from rounding algorithms. *Proceedings of the Thirty-Fourth Annual ACM Symposium on Theory of Computing*, pages 380–388, 2002.

[4] I. Councill, H. Li, Z. Zhuang, S. Debnath, L. Bolelli, W. Lee, A. Sivasubramaniam, and C. Giles. Learning metadata from the evidence in an on-line citation matching scheme. In *Proceedings of the 6th ACM/IEEE-CS Joint Conference on Digital libraries*, pages 276–285. ACM, 2006.

[5] A. Elmagarmid, P. Ipeirotis, and V. Verykios. Duplicate record detection: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 19(1):1–16, 2007.

[6] H. Hajishirzi, W. Yih, and A. Kolcz. Adaptive near-duplicate detection via similarity learning. In *Proceeding of the 33rd International ACM SIGIR Conference on Research and Development in information retrieval*, pages 419–426. ACM, 2010.

[7] S. Lawrence, C. L. Giles, and K. D. Bollacker. Autonomous citation matching. In *Proceedings of the Third International Conference on Autonomous Agents*. ACMPress, 1999.

[8] S. Lawrence, L. C. Giles, and K. Bollacker. Digital libraries and autonomous citation indexing. *IEEE Computer*, 32(6):67–71, 1999.

[9] G. Manku, A. Jain, and A. Sarma. Detecting near-duplicates for web crawling. In *The 16th International Conference on World Wide Web*, 2007.

[10] H. Pasula, B. Marthi, B. Milch, S. Russell, and I. Shpitser. Identity uncertainty and citation matching. In *Advances in Neural Information Processing Systems*, pages 1425–1432, 2003.

[11] S. Sarawagi and A. Bhamidipaty. Interactive deduplication using active learning. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Datamining - KDD '02*, page 269, New York, New York, USA, 2002. ACM Press.

[12] R. Vernica, M. J. Carey, and C. Li. Efficient parallel set-similarity joins using mapreduce. In *Proceedings of SIGMOD '10*, pages 495–506. ACM Press, 2010.

[13] C. Wang, J. Wang, X. Lin, W. Wang, H. Wang, H. Li, W. Tian, J. Xu, and R. Li. Mapdupreducer: detecting near duplicates over massive datasets. In *Proceedings of the 2010 International Conference on Management of Data*, pages 1119–1122. ACM, 2010.