Journal of Biomedical Informatics 82 (2018) 63–69

Contents lists available at ScienceDirect

Journal of Biomedical Informatics

journal homepage: www.elsevier.com/locate/yjbin

Identifying and characterizing highly similar notes in big clinical note datasets

Rodney A. Gabriel[a,b,*], Tsung-Ting Kuo[a], Julian McAuley[c], Chun-Nan Hsu[a]

[a] UCSD Health Department of Biomedical Informatics, University of California, San Diego, 9500 Gilman Dr, La Jolla, CA 92093, USA
[b] Department of Anesthesiology, University of California, San Diego, 200 West Arbor Dr, San Diego, CA 92103, USA
[c] Department of Computer Science and Engineering, University of California, San Diego, 9500 Gilman Dr, La Jolla, CA 92093, USA

ARTICLE INFO

Keywords:
Electronic medical record
De-deduplication
Natural language processing

ABSTRACT

Background: Big clinical note datasets found in electronic health records (EHR) present substantial opportunities to train accurate statistical models that identify patterns in patient diagnosis and outcomes. However, near-to-exact duplication in note texts is a common issue in many clinical note datasets. We aimed to use a scalable algorithm to de-duplicate notes and further characterize the sources of duplication.
Methods: We use an approximation algorithm to minimize pairwise comparisons consisting of three phases: (1) Minhashing with Locality Sensitive Hashing; (2) a clustering method using tree-structured disjoint sets; and (3) classification of near-duplicates (exact copies, common machine output notes, or similar notes) via pairwise comparison of notes in each cluster. We use the Jaccard Similarity (JS) to measure similarity between two documents. We analyzed two big clinical note datasets: our institutional dataset and MIMIC-III.
Results: There were 1,528,940 notes analyzed from our institution. The de-duplication algorithm completed in 36.3 h. When the JS threshold was set at 0.7, the total number of clusters was 82,371 (total notes = 304,418). Among all JS thresholds, no clusters contained pairs of notes that were incorrectly clustered. When the JS threshold was set at 0.9 or 1.0, the de-duplication algorithm captured 100% of all random pairs with their JS at least as high as the set thresholds from the validation set. Similar performance was noted when analyzing the MIMIC-III dataset.
Conclusions: We showed that among the EHR from our institution and from the publicly-available MIMIC-III dataset, there were a significant number of near-to-exact duplicated notes.

1. Background and significance

Electronic health records (EHR) are becoming more essential in every medical center. Reusing EHR data for research, quality improvement, and clinical support is promising [1]. Big clinical note datasets found in EHRs present substantial opportunities to train accurate statistical models that identify patterns in patient diagnosis and outcomes [2–6]. However, near-to-exact duplication in note texts – defined as a cluster of notes with partial to exact similarity – is a common issue in many clinical note datasets. The existence of highly similar notes may arise due to various reasons, including the pervasive use of note templates, copy-and-pasting, or automatic generation of notes from machines or procedures. The widespread presence of highly similar notes within EHR systems is potentially problematic when training predictive algorithms that model the language or attributes in these notes [7–14]. For example, highly similar notes, especially exact duplicates, may skew statistics of terms frequencies and cause overfitting of the trained models to a certain site/corpus, leading to poor generalization of the clinical natural language processing (NLP) system. A predictive model may erroneously identify correlations between symptoms and comorbidities [8,11]. Furthermore, duplicate records may be associated with patient safety hazards, including missing important laboratory results [9]. Correcting duplication in clinical note data is challenging, as the sources of duplication are widely observed but poorly understood. De-duplication and data cleansing approaches may improve the quality of clinical note corpora as a vital information source for meaningful use of EHR.

To detect and correct such duplicates requires algorithms that are both accurate and highly scalable. In the setting of big clinical note datasets, consisting of millions of free text notes, such algorithms need to identify similar notes in a realistic time frame. A brute-force approach involves pairwise comparisons of every note in the dataset; however, this is intractable for big clinical note datasets.

We, therefore, aim to use a scalable algorithm to de-duplicate notes

* Corresponding author at: Department of Anesthesiology, University of California, San Diego, 200 West Arbor Dr, San Diego, CA 92103, USA.
E-mail address: ragabriel@ucsd.edu (R.A. Gabriel).

https://doi.org/10.1016/j.jbi.2018.04.009
Received 10 January 2018; Received in revised form 16 April 2018; Accepted 17 April 2018
Available online 19 April 2018
1532-0464/ Published by Elsevier Inc.

and further characterize the source of duplication. We leverage big data corpora of clinical notes in the EHR that have been accumulated for years from large medical centers to explore duplication and hidden structure that may impact their meaningful use for both clinical support and research. Such findings may be used in future studies that assess the effect of near-duplication on machine learning, NLP, and privacy-preserving algorithms.

2. Material and methods

2.1. Data source

The clinical note dataset was collected from the medical centers of University of California, San Diego (UCSD), which is a large medical center that has deployed EHR systems for more than a decade. This project was exempt from the informed consent requirement by our Institutional Review Board. The clinical notes consisted of progress notes, history & physical examination notes, laboratory, and discharge summaries. With each note, we extracted metadata pertaining to date that the note was charted and patient medical record number. We chose to look specifically at a subset of notes related to patients with or at risk for morbid obesity, one of the three high priority use case conditions designated by our funding source and with the largest patient population among all of the use case conditions All notes pertaining to patients with the following criteria were included: (1) age greater than or equal to 18 years of age; and (2) body mass index greater than or equal to 25 kg/m^2 in the last five years. Patients excluded from the analysis were those that were pregnant or prisoners. Notes excluded from the analysis were those with fewer than 4 words. All other notes fitting the inclusion criteria were included in the analysis. As a separate analysis, we also explored clinical notes from the publicly available MIMIC-III dataset [15]. We obtained all note texts from the database and gathered the charting date and unique patient identification code for each note. No notes were excluded from the MIMIC-III dataset.

2.2. Similarity index

We used the Jaccard Similarity (JS) measure to compute the similarity between two documents [16]. JS operates on sets while a document may be considered to be a list, since words are ordered. Therefore, we converted each document into a set by a shingling process. For some fixed n, shingling is the process of taking n consecutive words from the document into a set, otherwise known as an n-gram method. Punctuation and whitespace are ignored. Thus, each shingle contains exactly n words. The JS of two documents is calculated using the n-grams from each set – defined as the number of shared n-grams divided by the total number of unique n-grams. Of note, how the order of words impacts the similarity between two documents can be controlled by setting n, while the order of the n-grams themselves will have low impact in determining the similarity.

JS has many desirable properties to be described below that allow for scalable algorithms to be developed. Other similarity measures can also be considered though in our experience they do not significantly impact de-duplication results, and have no impact at all when seeking to detect identical notes.

2.3. De-duplication algorithm

We developed a method to identify clusters of highly similar notes – we use an approximation algorithm to minimize pairwise comparisons and consists of three phases: (1) Minhashing (first used in AltaVista search engine to detect duplicate webpages from the entire World Wide Web) with Locality Sensitive Hashing [17,18]; (2) a clustering method using tree-structured disjoint sets; and (3) classification of near-duplicates via pairwise comparison of notes in each cluster. The algorithm can be used to analyze large clinical note corpora with limited available memory space and has been described previously [19]. Fig. 1 illustrates the major steps of this algorithm.

2.4. Minhashing

The first step in the de-duplication algorithm is Minhashing. The goal of this step is to replace large sets of documents by much smaller representations called "signatures". The similarity-preserving property must hold; that is when the signatures of two documents are compared, their similarity must be close to the true similarity of the two documents. One technique to produce such signatures is minhashing [17,18]. Each document is broken down into a set of n-grams, which consists of n consecutive words produced via a sliding window throughout the document. We set $n = 4$ here. In our previous study, we

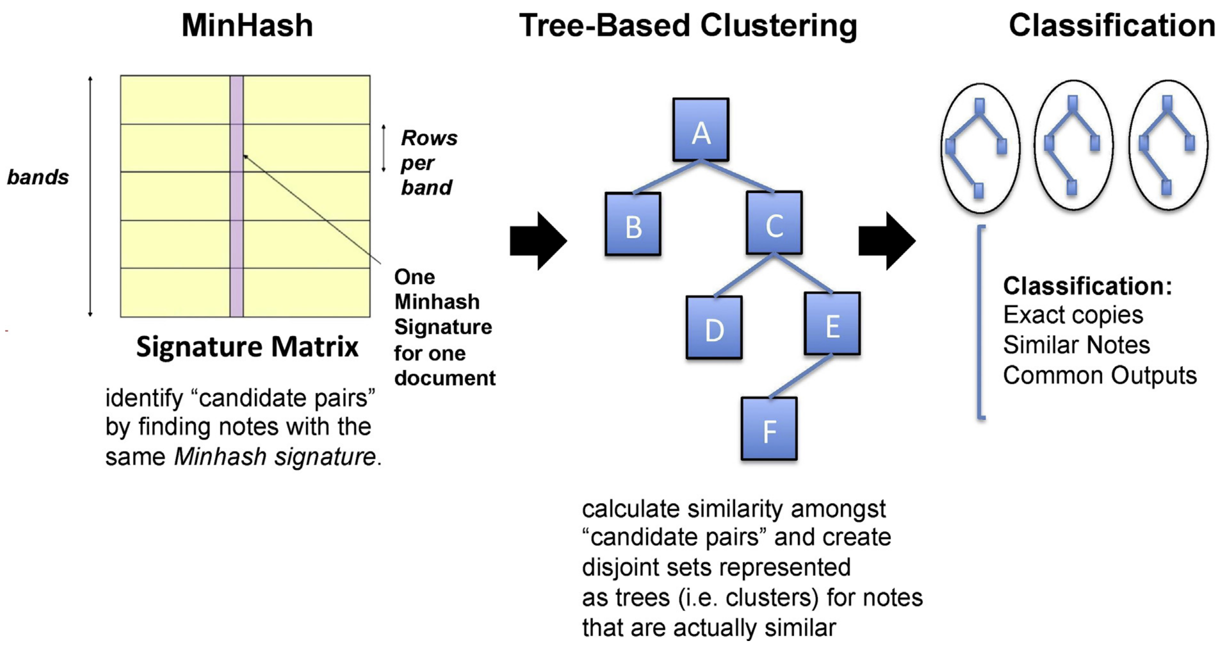
MinHash
Tree-Based Clustering
Classification
bands
Rows per band
One Minhash Signature for one document
Signature Matrix
identify "candidate pairs" by finding notes with the same *Minhash signature*.
A
B
C
D
E
F
calculate similarity amongst "candidate pairs" and create disjoint sets represented as trees (i.e. clusters) for notes that are actually similar
Classification:
Exact copies
Similar Notes
Common Outputs

Fig. 1. Illustration of the methodology used to de-duplicate notes from a big clinical note dataset.

determined that *n* lower than 4 would create too many overlaps while *n* greater than 4 would create too many unique n-gram signatures that would be too expensive to efficiently process [19]. Four was sufficient to account for both word occurrences and ordering to capture duplications. Once a complete set of n-grams is created for all documents in the dataset, each n-gram is assigned a unique integer. For each document, we then randomly permute its integers from its n-grams. The "Minhash value" is then equal to the minimum permuted value among all its n-grams. The JS of two documents is equal to the probability that the Minhash value of the first document is equal to the Minhash value in the second document. To calculate the probability of two documents having the same Minhash value, we apply a fixed number (M) of random permutations to produce M Minhash values. If m is the number of trials for which the Minhash values are equal between the two documents, then we can approximate the probability of the two documents having the same Minhash value as m/M. Thus, for each document, we apply M random permutations and for each permutation, we take the minimum value among its n-grams. As a result, we have reduced the representation of our documents to a set of M numbers. We call this set a minhash signature.

However, producing the M random permutations on a large integer space can be computationally expensive. Hence, we need a much faster method to perform approximate random permutations. We use random hash functions for this purpose. If the hash space is very large compared to the cardinality of the set of numbers being permuted, the probability that a pair of numbers will hash to the same number will be very low. Thus, each number in the set will be permuted to a unique number with a high probability.

At this point, each document has an assigned Minhash signature, that is each document is reduced to a much smaller representation with M integers. However, it is still required to perform pairwise comparisons between documents' Minhash signatures, which are computationally expensive in big datasets. Therefore, we created what is known as a "signature matrix", in which the hash function to an n-gram's unique integer are the rows and each document's Minhash signature are the columns. The signature matrix is then split into equally sized bands of rows. If we have r rows in each band, then we will have $b = \text{M}/r$ bands. Now, we consider a pair of documents for similarity computation only if the two documents have the same Minhash signature in at least one band. The signature matrix, however, is too large to hold in memory. To circumvent this, we divide it into the bands so that each band can be processed independently in memory. To minimize the amount of memory taken by the bands while keeping accuracy high, we hashed the r rows in one band of a document to a single 64-bit number. Thus, one band can be represented by one integer per document. We call the matrix with these bands as rows a band matrix. The following is a step-by-step description of the creation of the signature matrix:

1. Compute the signature matrix in which the rows correspond to a specific n-gram and the columns correspond to a specific document. Therefore, for each column (document), only the rows (n-grams) that are present in that document are considered "positive".
2. Split the signature matrix into b bands, each with r rows
3. Let C be a (initially empty) list of candidate pairs
4. For each band:
   a. Find all pairs of documents which have the same signature in the band (i.e. the same pattern of "positive" n-grams)
   b. Add these pairs to C
   c. Compute the JS for all pairs in C.

2.5. Clustering using disjoint sets

Disjoint sets are a data structure used to process sets efficiently where the sets are represented by trees. Our goal is to identify documents with at least one other near duplicate document that have JS scores greater than or equal to the given threshold in a data set. Consider that all documents are connected as a graph with edges as their JS scores. This problem is equivalent to removing all edges with a score lower than the threshold. We therefore call this threshold the *edge threshold*. However, this would require exhaustive pairwise comparisons, which is intractable. Instead, we developed an approximate solution by maintaining trees representing each cluster such that all pairs of documents in a tree would have a lower bound on the JS of at least the threshold. This threshold is called the *tree threshold*. The approximation is closely related to the problem of community detection from a connected social graph [20,21].

Recall how candidate pairs are generated. For each band, all pairs of documents, which have the same signature in the band, are considered as candidate pairs. We can then check if the pair actually has a JS score higher than the edge threshold. If so, we replace each document in the pair by the root document in the cluster in which the document belongs. The next step is to combine clusters using the set Union operation. This is essentially the procedure of finding candidate pairs, which will be executed for each band. Clusters (disjoint sets) created in bands already processed will be inherited when the current band is processed and the Union procedure will merge two disjoint sets by applying the triangle inequality of JS [22] to ensure that the resulting set still maintains a mutual similarity at least as high as the tree threshold (document pairs in each cluster will be guaranteed to have JS scores higher than the given tree threshold). The guarantee serves our goal of identifying near duplicates. This algorithm also reduces the number of documents for which we need to actually evaluate their JS scores. The following is a brief step-by-step description of generation of these sets:

1. For each band (from signature matrix), sort the band based on its signature value.
2. For each set of documents with the same signature value:
   a. For all document pairs (A, B) in the set, calculate the JS
   b. If JS ≥ edge threshold, then Union(A, B). The function of Union is defined in more detail in a previous publication [19]. In brief, each pair (document A and B) is compared and determined whether their similarity satisfies the *tree threshold* and can be included into an existing set; otherwise, a new tree will be created and one of the documents will serve as the parent node for that tree.

2.6. Classification

Classification of notes generated in the final clusters were categorized to either:

- exact copies (defined as two notes with JS = 1.0 and charted at the same exact date for the same exact patient),
- common output notes (defined as two notes with JS = 1.0 but which do not have to be for the same patient or same chart date), and
- similar notes (defined as all other notes greater than or equal to the set JS threshold).

Examples of common output notes are notes automatically generated from machine output (i.e. electrocardiogram, laboratory tests, etc.). "Similar notes" is a broad classification category, which includes any notes with a JS greater than the given threshold and that were not considered exact notes or common output notes. Basically, if a "similar" pair (based on the selected JS) is not an exact copy or a common output note (i.e. JS = 1.0 but not same patient and/or time), it is defaulted to be considered a "similar note". We used various JS thresholds (from 0.4 to 1.0) in our experiments to be considered "similar notes". Similar notes could potentially include notes created from a similar template or notes with a high degree of copy-and-pasting. To classify these notes, pairwise comparisons are performed for each pair of notes in each cluster. For each pair, the following conditions are tested: (1) whether JS = 1.0, (2) whether the chart date was the same; and (3) whether the patient was the same.

2.7. Validation methodology

From the institutional and MIMIC-III clinical note datasets, we generated two million random pairs and calculated the JS for each pair. All pairs that had a JS greater than or equal to 0.3 and less than or equal to 1.0 were included in a final validation list. The lists consisted of 1568 and 400 random pairs of notes for the institutional and MIMIC-III clinical note datasets, respectively. The lowest JS threshold we analyzed was 0.4, therefore, we chose 0.3 as the lower range of notes to include in the final validation list in order to: (1) describe notes that should not be included in clusters; and (2) limit the number of comparisons needed for the validation step to preserve run-time. The presence of each random pair from the final validation list was compared to all notes from each cluster generated from the de-duplication algorithm. If the JS of that random pair is greater than or equal to the JS threshold of the de-duplication algorithm, then the two notes of this pair are expected to belong to the same cluster. If the JS of that random pair is less than 5% below the threshold of the JS of the de-duplication algorithm, the two notes of this pair are expected to not be in the same cluster. The proportion of correctly grouped pairs and incorrectly grouped pairs were calculated and reported as a percentage. The same validation step was similarly performed for both the institutional and MIMIC-III clinical note datasets.

3. Results

There was a total of 10,672,699 clinical notes in the institutional dataset from June 4, 2012 – August 6, 2015. Of these, 1,528,940 notes were associated with morbid obesity and thus were included in our initial analysis. The de-duplication algorithm was processed on a virtual machine with 256 gigabytes of random access memory – the total time to process all notes into clusters was 36.3 h. Fig. 2A illustrates the number of clusters and total notes included based on the JS threshold. When the JS threshold was set at 0.7, the total number of clusters was 82,371, which included 304,418 total notes. Table 1 lists the frequency of cluster sizes in each JS threshold setting: clusters with 2 notes, 3–10 notes, 11–100 notes, 101–1000 notes, and greater than 1000 notes.

Though clusters containing exactly two notes made up the vast majority of clusters, there are six clusters with more than 1000 notes that are exactly identical and fifteen clusters with more than 1000 notes that had a JS of 0.8 or higher. Notes in each cluster were then classified into whether they were exact copies, common outputs, or similar notes. The proportion of notes fitting into each of these three categories are illustrated in Fig. 2B. Table 2 lists the validation results of these clusters. We manually investigated 100 random sets that had cluster sizes of 2 when the JS threshold was set at ≥0.7. The majority (86%) of sets were for the same patient. The remaining 14% were for different patients but otherwise highly similar notes. We then manually investigated all 17 sets with cluster size > 1000 notes. All sets consisted of common output notes – automatic machine-generated notes (i.e. electrocardiogram) that were less than 20 words with identical vocabulary for a variety of patients and time.

Among all JS thresholds, no clusters contained pairs of notes that were incorrectly clustered (false positive) with the JS below the 5% allowable level. When the JS threshold was set at 0.9 or 1.0, the de-duplication algorithm captured 100% of all random pairs with the JS higher than or equal to the set similarity thresholds from the validation set. As the threshold decreased from 1.0 to 0.4, the proportion of random pairs correctly clustered decreased.

Next, our algorithm was tested on the publicly available dataset, MIMIC-III. There was a total of 2,065,096 clinical notes in this dataset that were included in the analysis. Fig. 3A shows the number of clusters and total notes based on the JS threshold. When the JS threshold was set at 0.7, the total number of clusters was 110,030, which included 310,230 total notes. Table 3 lists the frequency of cluster sizes for each JS threshold setting.

The proportion of notes fitting into each of these three categories is illustrated in Fig. 3B. Table 4 lists the validation results of these clusters. Among all JS thresholds, again, no clusters contained pairs of notes that had a JS less than 5% of the set threshold.

When the JS threshold was set to as low as 0.6, the de-duplication algorithm still captured 100% of all random pairs with the JS higher than the thresholds from the validation set. As the JS threshold was decreased to 0.5 and 0.4, the percent captured decreased to 97% and 64%, respectively.

Table 1
Frequency of clusters with different number of notes from clinical notes from our institutional dataset.

Jaccard similarity	Cluster size				
	2	3–10	11–100	101–1000	> 1000
1.0	38,589	5389	634	101	6
0.9	51,677	7109	669	109	10
0.8	59,476	8369	842	143	15
0.7	68,975	12,265	964	150	17
0.6	95,565	14,826	915	140	17
0.5	130,073	15,601	943	137	17
0.4	180,997	18,059	950	137	17

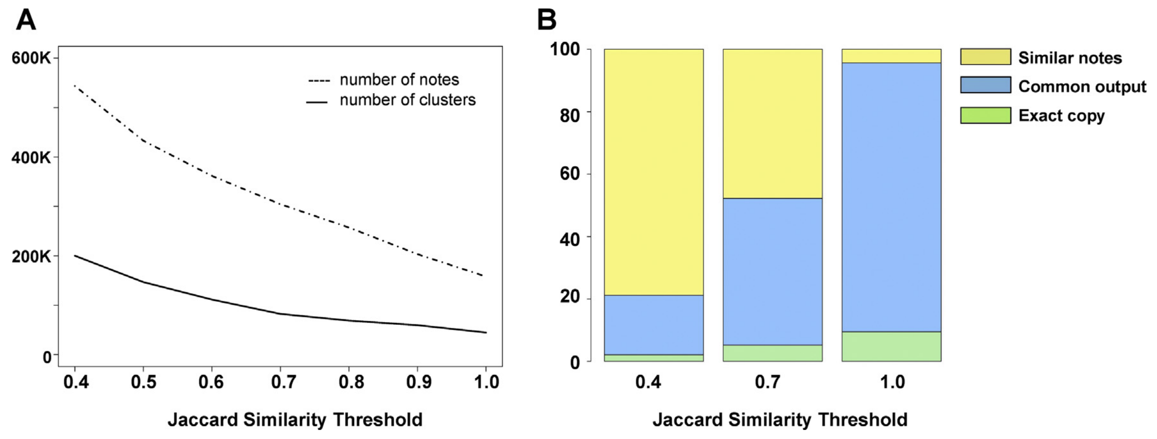

Fig. 2. (A) Line graph illustrating the number of clusters and total number of notes generated from the algorithm when the Jaccard Similarity threshold was set at 0.4–1.0; (B) stacked bar plots illustrating the proportion of notes categorized as exact copies, common output, or similar notes. These graphs are based on our institutional clinical note dataset.

Table 2
Validation results for our institutional notes. FPR: False positive rate. TPR: True positive rate, or sensitivity.

Jaccard similarity threshold	# of pairs in clusters below JS threshold	Total number of random pairs tested below JS threshold	% FPR	# of pairs in clusters below JS threshold (allowable)	% FPR (allowable)	# of pairs in clusters ≥ JS threshold	Total number of random pairs ≥ JS threshold	% TPR
1.0	0	1545	0.00	0	0.00	23	23	100.00
0.9	0	1517	0.00	0	0.00	49	49	100.00
0.8	19	1448	1.31	0	0.00	62	120	51.67
0.7	8	1290	0.62	0	0.00	105	278	37.77
0.6	1	1205	0.08	0	0.00	99	363	27.27
0.5	0	1102	0.00	0	0.00	99	466	21.24
0.4	0	715	0.00	0	0.00	99	853	11.61

4. Discussion

To our knowledge, no large-scale attempts have been made to comprehensively identify highly similar notes in big clinical note datasets. One reason is that comprehensive pairwise comparisons would be prohibitively expensive in terms of computational resources. We showed that among the EHR from our institution and from the publicly-available MIMIC-III dataset, there were a significant number of near-to-exact duplicated notes. This work attempts to address note-level duplication versus event-level duplication. Though the presence of note-level duplication is well-known, few studies provide quantitative and large-scale measures of the extent of this issue. One study analyzed approximately 20,000 notes and found that only 18% of the text was manually entered [23]. This manuscript represents a step toward understanding its scale, the potential consequences, and scalable solutions to the problem by analyzing millions of notes from two different databases.

Though big clinical note datasets present substantial opportunities to train accurate statistical language models for advanced clinical natural language processing tools and meaningful use applications, duplicates may prevail for various reasons – a phenomenon widely observed and reported, but never comprehensively studied within a large medical center [7–13,24]. In a previous study analyzing ~1500 random clinical notes, the authors reported that 22% of an average signout note was considered unique, while 44% of progress notes were unique. Duplicated sections of signout notes most frequently included medication lists and history of present illness text while duplicated sections in progress notes were frequently from assessment and plan sections [25]. This high density of similarity among clinical notes will directly impact how we approach clinical NLP problems. Sampling from such a corpus for training and testing may require special treatment to avoid unwanted biases that degrade statistical language models. Many standard machine learning and NLP algorithms make assumptions about the data from which they are built (*e.g.,* data samples are independent and identically distributed). Unfortunately, these assumptions can be violated once duplicates are introduced into the data. For example, for supervised learning algorithms, data that is intended to be 'held-out' for validation may be replicated in the training set, causing a predictive model's accuracy to be overstated, which would limit the model's ability to work externally. Another example is with outlier detection, in which a highly unusual combination of symptoms might go unnoticed if several redundant copies of the symptoms appear across multiple notes. However, some duplicated notes should not be considered a problem if the high similarity shared between notes is clinically important. The challenges that follow successful de-duplication include ranking the importance of these similar notes. Identifying duplication is only part of the solution. We show that there are several complex sources of redundant information, where "exact copies" of notes only accounted for a fraction. These sources of near-to-exact duplication also included notes that were generated automatically, generated from templates, or have had significant portions copy-and-pasted. Simply aggressively removing even partial duplicates would

Table 3
Frequency of clusters with different number of notes from clinical notes from the MIMIC-III dataset.

Jaccard similarity	Cluster size				
	2	3–10	11–100	101–1000	> 1000
1.0	29,715	5448	221	8	1
0.9	58,803	18,307	221	8	1
0.8	66,509	31,456	339	12	1
0.7	67,800	41,700	514	15	1
0.6	76,619	48,406	477	15	1
0.5	95,742	51,333	431	14	1
0.4	120,955	53,108	462	14	1

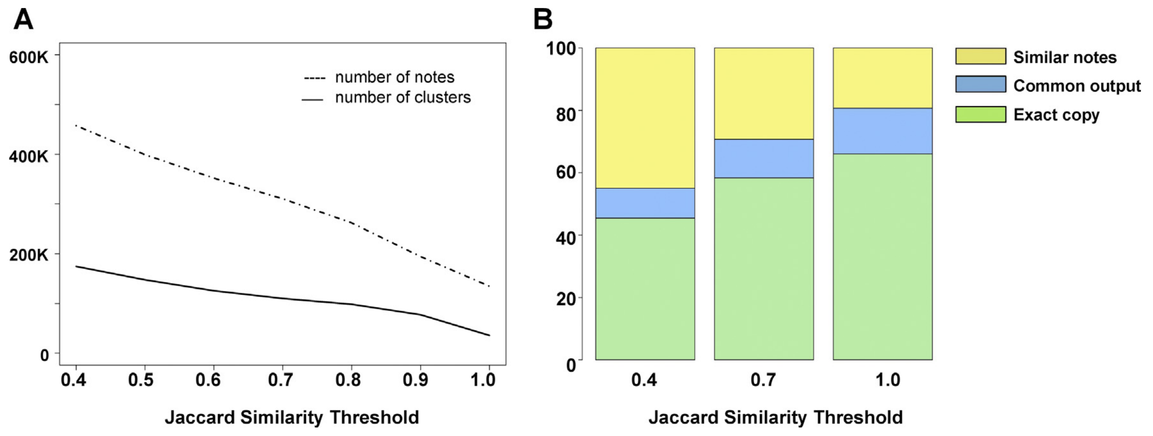

Fig. 3. (A) Line graph illustrating the number of clusters and total number of notes generated from the algorithm when the Jaccard Similarity threshold was set at 0.4–1.0; (B) stacked bar plots illustrating the proportion of notes categorized as exact copies, common output, or similar notes. These graphs are based on the publicly available MIMIC-III dataset.

Table 4
Validation results for MIMIC-III notes. FPR: False positive rate. TPR: True positive rate, or sensitivity.

Jaccard similarity threshold	# of pairs in clusters below JS threshold	Total number of random pairs tested below JS threshold	% FPR	# of pairs in clusters below JS threshold (allowable)	% FPR (allowable)	# of pairs in clusters ≥JS Threshold	Total number of random pairs ≥JS Threshold	% TPR
1.0	0	367	0.00	0.00	0.00	33	33	100.00
0.9	0	367	0.00	0.00	0.00	33	33	100.00
0.8	1	367	0.27	0.00	0.00	33	33	100.00
0.7	0	366	0.00	0.00	0.00	34	34	100.00
0.6	0	366	0.00	0.00	0.00	34	34	100.00
0.5	0	365	0.00	0.00	0.00	34	35	97.14
0.4	0	347	0.00	0.00	0.00	34	53	64.15

also bias algorithms in unknown ways and impede system re-use.

Once similar clusters are found, the next step would be to classify the source of near-to-exact duplication. We chose three types of categorization schemes – exact copies, common output notes, and a third, broad category termed similar notes. The latter category (i.e., similar notes) is essentially all other notes not considered exact copies or common output notes and could be further classified as notes constructed from a common template or notes with high prevalence of copy-and-pasting. Future studies will aim to further categorize this third group. Redd et al. demonstrated the feasibility of identifying documents containing templates in a small cohort of notes [26]. Furthermore, it is important to elucidate reasons why exact copies are being generated in the note dataset as there are several issues with this phenomenon. This is essential for determining which duplicates are considered clinically important versus problematic.

Comparing the results for the two datasets, we found that the proportion of exact copy notes in the exactly identical clusters (JS = 1.0) was higher in MIMIC-III than at our institution. Also, we had more than a hundred large clusters with a size greater than 100 from our institutional dataset compared to less than 20 for MIMIC-III, though their total numbers of near-to-exact duplicate notes and clusters are similar. Whether any curation process may have reduced the number of large clusters for MIMIC-III and whether the presence of large clusters from our institutional dataset are typical among real clinical datasets and contributed to lower true positive rates for our clustering algorithm are among our topics for further investigation.

We utilized an efficient de-duplication algorithm that required computation time linearly proportional to the size of the input dataset. Our results show that the algorithm is scalable to cluster near-to-exact duplicates among millions of notes from both institutional and publicly available datasets. The required processing time (~40 h) for approximately 2 million notes was much more realistic compared to the runtime required for exhaustive pairwise comparisons.

Our de-duplication algorithm is a greedy algorithm that merges notes that the Minhashing estimates as highly likely to be similar to a root note of a tree representing a cluster. The algorithm is efficient and ideal for clustering exactly identical notes. For clustering near-duplicate notes, we use the triangle inequality property of the JS metric to ensure that no note that is too far below a given threshold (compared with any note in a cluster) is incorrectly merged. This approach guarantees no false positives, but may exclude notes that should be merged into the same cluster, as confirmed by our validation tests of the resulting clusters.

Within our institutional dataset, when the JS threshold was set to high thresholds (higher than 90% similarity), our algorithm achieved perfect true positive rates, with all pairs of notes with at least that similarity level were correctly clustered. However, when the threshold setting decreases to 0.8 or lower, the true positive rates of captured pairs decreased from 52% to 12%, respectively. When the algorithm was applied to MIMIC-III, the true positive rates stay perfect even when the threshold is set to as low as 0.6. On the other hand, validation results demonstrated that no notes were *incorrectly* clustered (false positives), meaning that all notes within a cluster had a JS within at least 5% of the set similarity threshold. This suggests that the actual number of near-duplicate notes would be greater than the number reported here, especially for the institutional dataset. Based on our algorithm, by definition, no two documents in a set could have a JS less than the defined threshold. Per our methods, during the clustering step, candidate pairs are compared and the JS calculated. If they do not meet criteria, then they are not placed in the same set. Therefore, we would expect a very low false positive rate as we did for all similarity indices.

There are several limitations to our study. First, our greedy clustering algorithm missed pairs of near-duplicate notes to merge them into the same cluster, underestimated the number of near-duplicate notes, and ran out of memory when we attempted to process the entire set of 11 million notes from our institutional dataset. Though our preliminary study shows that this over-partitioning issue can be improved significantly simply by applying a second round of clustering after an initial clustering, as future work, we would like to develop a solution with guarantee of performance in terms of both computational efficiency and quality of clustering.

Furthermore, we chose to classify the clustered notes into three categories: exact copies, common output notes, and similar notes. Classification of highly similar notes is not limited to the three categories we chose. In any case, we chose these three categories as a proof-of-concept. Future studies will be aimed at defining and characterizing sources of duplication with potential clinical implications.

In conclusion, we described a scalable algorithm that clusters highly similar notes (based on a set similarity index) from large clinical note datasets that contain millions of documents. Such an algorithm may be executed in a more realistic timeframe compared to an approach that depends upon performing pairwise comparisons. In the setting of expanded utilization of EHR in healthcare systems, the number of clinical documents will continue to expand rapidly.

Author contributions

All the author helped design the study, conduct the study, collect the data, analyze the data, and prepare the manuscript.

Financial support

This work was supported by the National Library of Medicine (NLM) training Grant No. T15LM011271 and partially funded through a Patient-Centered Outcomes Research Institute (PCORI) Award (Contract CDRN-1306-04819). Authors utilized resources provided by Integrating Data for Analysis, Anonymization and SHaring (iDASH). The statements in this article are solely the responsibility of the authors and do not necessarily represent the views of the Patient-Centered Outcomes Research Institute (PCORI), its Board of Governors or Methodology Committee.

Conflicts of interest

None.

References

[1] D. Blumenthal, M. Tavenner, The, "meaningful use" regulation for electronic health records, N. Engl. J. Med. 363 (6) (2010) 501.
[2] S. Meystre, R. Gouripeddi, J. Tieder, J. Simmons, R. Srivastava, S. Shah, Enhancing comparative effectiveness research with automated pediatric pneumonia detection in a multi-institutional clinical repository: a PHIS+ pilot study, J. Med. Internet Res. 19 (5) (2017) e162.
[3] J.D. Osborne, M. Wyatt, A.O. Westfall, J. Willig, S. Bethard, G. Gordon, Efficient identification of nationally mandated reportable cancer cases using natural language processing and machine learning, J. Am. Med. Inform. Assoc. 23 (6) (2016) 1077.
[4] R. Patel, R. Wilson, R. Jackson, M. Ball, H. Shetty, M. Broadbent, R. Stewart, P. McGuire, S. Bhattacharyya, Cannabis use and treatment resistance in first episode psychosis: a natural language processing study, Lancet 385 (Suppl 1) (2015) S79.
[5] C. Shivade, P. Raghavan, E. Fosler-Lussier, P.J. Embi, N. Elhadad, S.B. Johnson, A.M. Lai, A review of approaches to identifying patient phenotype cohorts using electronic health records, J. Am. Med. Inform. Assoc. 21 (2) (2014) 221.
[6] M.F. Toerper, E. Flanagan, S. Siddiqui, J. Appelbaum, E.K. Kasper, S. Levin, Cardiac catheterization laboratory inpatient forecast tool: a prospective evaluation, J. Am. Med. Inform. Assoc. 23 (e1) (2016) e49.
[7] C. Delcher, N. Puttkammer, R. Arnoux, K. Francois, M. Griswold, I. Zaidi, Y.A. Patrice Joseph, B.J. Marston, Validating Procedures used to identify duplicate reports in Haiti's national HIV/AIDS case surveillance system, J. Registry Manag. 43 (1) (2016) 10.
[8] K.W. Hammond, S.T. Helbig, C.C. Benson, B.M. Brathwaite-Sketoe, Are electronic medical records trustworthy? Observations on copying, pasting and duplication, AMIA Annu. Symp. Proc. 269 (2003).
[9] E. Joffe, C.F. Bearden, M.J. Byrne, E.V. Bernstam, Duplicate patient records–implication for missed laboratory results, AMIA Annu. Symp. Proc. 2012 (2012) 1269.
[10] A.B. McCoy, A. Wright, M.G. Kahn, J.S. Shapiro, E.V. Bernstam, D.F. Sittig, Matching identifiers in electronic health records: implications for duplicate records and patient safety, BMJ Qual. Saf. 22 (3) (2013) 219.
[11] S. Thielke, K. Hammond, S. Helbig, Copying and pasting of examinations within the electronic medical record, Int. J. Med. Inform. 76 (Suppl 1) (2007) S122.
[12] J.D. Thornton, J.D. Schold, L. Venkateshaiah, B. Lander, Prevalence of copied information by attendings and residents in critical care progress notes, Crit. Care Med. 41 (2) (2013) 382.
[13] J.M. Weis, P.C. Levy, Copy, paste, and cloned notes in electronic health records, Chest 145 (3) (2014) 632.
[14] M. Zhang, M. Shubina, F. Morrison, A. Turchin, Following the money: copy-paste of lifestyle counseling documentation and provider billing, BMC Health Serv. Res. 13 (2013) 377.
[15] A.E. Johnson, T.J. Pollard, L. Shen, L.W. Lehman, M. Feng, M. Ghassemi, B. Moody, P. Szolovits, L.A. Celi, R.G. Mark, MIMIC-III, a freely accessible critical care database, Sci. Data 3 (2016) 160035.
[16] A. Strehl, J. Ghosh, M. Mooney, Impact of similarity measures on web-page clustering, vol. 58, Workshop on Artificial Intelligence for Web Search (AAAI 2000), 2000.
[17] Broder A. On the resemblance and containment of documents, vol. 21, in: Proceedings of the Compression and Complexity of Sequences, 1997.
[18] A. Broder, M. Charikar, A. Frieze, Min-wise independent permutations, J. Comput. Syst. Sci. 60 (3) (2000) 630.
[19] S. Shenoy, T.T. Kuo, R.A. Gabriel, J. McAuley, C.N. Hsu, Deduplication in a massive clinical note dataset, arXivorg, 2017.
[20] M.E. Newman, Modularity and community structure in networks, Proc. Natl. Acad. Sci. U. S. A. 103 (23) (2006) 8577.
[21] V.D. Blondel, J.L. Guillaume, R. Lambiotte, E. Lefebvre, Fast unfolding of communities in large networks, J. Stat. Mech.: Theory Exp. 2008 (2008).
[22] A.H. Lipkus, A proof of the triangle inequality for the tanimoto distance, J. Math. Chem. 26 (1–3) (1999) 263.
[23] M.D. Wang, R. Khanna, N. Najafi, Characterizing the source of text in electronic health record progress notes, JAMA Intern. Med. 177 (8) (2017) 1212.
[24] R. Zhang, S. Pakhomov, B.T. McInnes, G.B. Melton, Evaluating measures of redundancy in clinical texts, AMIA Annu. Symp. Proc. 2011 (2011) 1612.
[25] J.O. Wrenn, D.M. Stein, S. Bakken, P.D. Stetson, Quantifying clinical narrative redundancy in an electronic health record, J. Am. Med. Inform. Assoc. 17 (1) (2010) 49.
[26] A.M. Redd, A.V. Gundlapalli, G. Divita, M.E. Carter, L.T. Tran, M.H. Samore, A pilot study of a heuristic algorithm for novel template identification from VA electronic medical record text, J. Biomed. Inform. 71S (2017) S68.