A New Semantic Similarity Scheme for More Accurate Identification in Medical Data

Conference Paper · August 2023

DOI: 10.1109/ISC257844.2023.10293563

5 authors, including:

Colin Wilcox
Manchester Metropolitan University
3 PUBLICATIONS **4** CITATIONS
SEE PROFILE

Soufiene Djahel
Coventry University
99 PUBLICATIONS **2,072** CITATIONS
SEE PROFILE

Nicholas Costen
Manchester Metropolitan University
72 PUBLICATIONS **2,066** CITATIONS
SEE PROFILE

A New Semantic Similarity Scheme for more Accurate Identification in Medical Data

Colin Wilcox*, Soufiene Djahel†, Vasilios Giagos‡, Kristopher Welsh*, and Nicholas Costen*
*Manchester Metropolitan University, UK
† University of Huddersfield, UK
‡University of Essex, UK
{colin.r.wilcox@stu.mmu.ac.uk, s.djahel@hud.ac.uk, v.giagos@essex.ac.uk, (k.welsh, n.costen)@mmu.ac.uk}

***Abstract*—This paper aims to design a new measure of similarity between personal textual information retrieved from historic medical records to correct errors introduced due to poor encoding and data omission. The key motivation underlying our proposed layered algorithm, named Semantic Similarity scheme (SSIM), is to create a consistent, complete and accurate data set that may then be used as a basis for the identification and authentication of individuals in a medical context. Such consistent data may provide a basis for use as part of an access control system without compromising medical ethics or security. The obtained evaluation results, using four sample data sets from the UK, USA, Canada and Australia, highlight promising benefits compared to other similarity measures including Jaccard index, Sorensen-Dice and Cosine Similarity - especially when nicknames, abbreviations and synonyms are used to determine similarity.**

***Keywords* – Text Similarity, Semantic Similarity, Medical Data, Medical Records.**

I. Introduction

Usually, information retrieval is based on lexicographic matching of terms, where two terms are the same if they look the same. Little contextual awareness was used to determine similarity, with early work in the area of semantic textual similarity being focused on document-level models for determining textual similarity [1], often using an unsupervised approach, primarily for the purpose of indexing documents for search. These models were largely based upon the assumption that a greater overlap in terms indicate greater inter-document similarity. This area of research was built upon by Lee and others [2] who also modeled similarity at the document level but with support from human-based semantic judgments of similarity to refine the matching data sets. Modern needs and usage means that a purely lexicographic approach is insufficient to determine similarity [3], there is a need to also consider the context and meaning of the original data to determine the similarity of two pieces of text. Two terms can be lexicographically different but have the same meaning (synonyms or abbreviations by some measure) [4]. Such terms may have approximately the same meaning or they belong to the same class of information based on some categorization. The lack of common terms in two documents does not necessarily mean that the documents are unrelated to each other [5]. Adopting either a qualitative or quantitative approach to text comparison each brings challenges, and the appropriate balance between context and comparability may vary depending on intended use [6]. Related documents may contain semantically similar but not necessarily the same terms and still have the same meaning.

Textual metrics have been developed and applied in different scientific fields, including the detection/correction of spelling errors, statistics for probabilistic record linkage [7], databases for record matching [8], artificial intelligence for supervised learning and biometrics and biology [9]. There are many string matching algorithms to support these approaches, including Levenshten distance (and its extensions), Hamming Distance, Q-gram similarity [10], cosine similarity [11] and dice coefficient [12].

The remainder of this paper is organized as follows. In Section II, we present background information on textual similarity and the issues it presents, together with related work. Section III describes our proposed algorithm and its basic elements. Section IV presents and analyses the performance evaluation results. Section V concludes the paper.

II. Background and related work

There are many ways in which strings may be compared, however our focus is on those characteristics common in personal medical data. We apply measurements that relate to these characteristics when determining similarity. Regardless of the characteristics being compared, we use a standard scale of measurement to indicate how similar two pieces of textual information, or parts of them, are with each other. Those fragments which are considered to be *identical* pieces of information will be represented by *one* on this scale. Two strings which are totally different, with no common characteristics, are represented at the other scale extreme with a similarity measure of *zero*. Other degrees of similarity are expressed within this range. In this model, we focus on comparison metrics addressing characteristics common with names, addresses, and other textual information common within medical data records. The data model uses some of these in isolation and others as part of a multi-pass strategy.

There are many existing algorithms and techniques that attempt to measure the degree of similarity between two strings or string fragments. These are usually restricted to simplistic difference measures or metrics of the transformations required to convert one string into the other. Either approach can offer a measure of the difference between the two and, indirectly, a measure of their similarity. Such techniques, however, tend

to measure *physical* similarities rather than *semantic* similarity and although rather simplistic and mechanically in their nature are widely used and so form a good basis for comparison against the performance of our proposed layered algorithm.

Token based approaches are the set of algorithms which base similarity on the number of common string tokens. Algorithms falling under this category are based on whether or not the token under consideration appears in both set of tokens, modified to work for the case of string tokens. Some of the more commonly used methods are described below.

A. Jaccard index

The Jaccard index measure J is a popular algorithm based on domain similarity and is used to find the number of tokens common to two token sets, t_1 and t_2 and can be expressed as

$$J(t_1, t_2) = \frac{|t_1 \cap t_2|}{|t_1 \cup t_2|} = \frac{|t_1 \cup t_2|}{|t_1| + |t_2| - |t_1 \cup t_2|} \quad (1)$$

where the numerator is the intersection (*common tokens*) and denominator is the union (*unique tokens*). The second case is for when there is some overlap, for which common terms are removed as they would add up twice by combining all tokens of both strings. As the required input is tokens instead of complete strings, users are responsible for appropriately tokenizing the strings, depending on the use case.

B. Sorensen-Dice

The Sorensen-Dice approach measures similarity SD by finding common tokens between two sets of string tokens t_1 and t_2. The number of pairs of matching tokens is then divided by the total number of tokens in both sets to give a measure of *proportionate* similarity between the two. Sorensen-Dice approach can be expressed as follows:

$$SD(t_1, t_2) = \frac{2\,|t_1 \cap t_2|}{|t_1| + |t_2|} \quad (2)$$

The basic principal with this measure is that if a token is present in both strings, its *total count* is added twice. The use of the set intersection ensures double counting is avoided. The denominator is simple combination of all tokens in both strings. SD differs from J when considering the denominators in each formula. The Jaccard formula creates the union of two string sets and uses this to overcome the double counting problem but as a result Sorensen-Dice has the tendency to will always overestimate the similarity between two strings.

III. Proposed Semantic Similarity Algorithm

Our proposed semantic similarity (SSIM) algorithm consists of a set of comparison techniques, each of which calculates a weighted value indicating the degree of similarity between two textual objects. Each operation yields a measure within a normalized range between 0 (*no similarity*) to 1.0 (*identical match*), from which an overall average measure of similarity is calculated. The nature of the text being compared in some cases may mean that some algorithms are inappropriate and may not be performed. SSIM is built upon the fact that medical

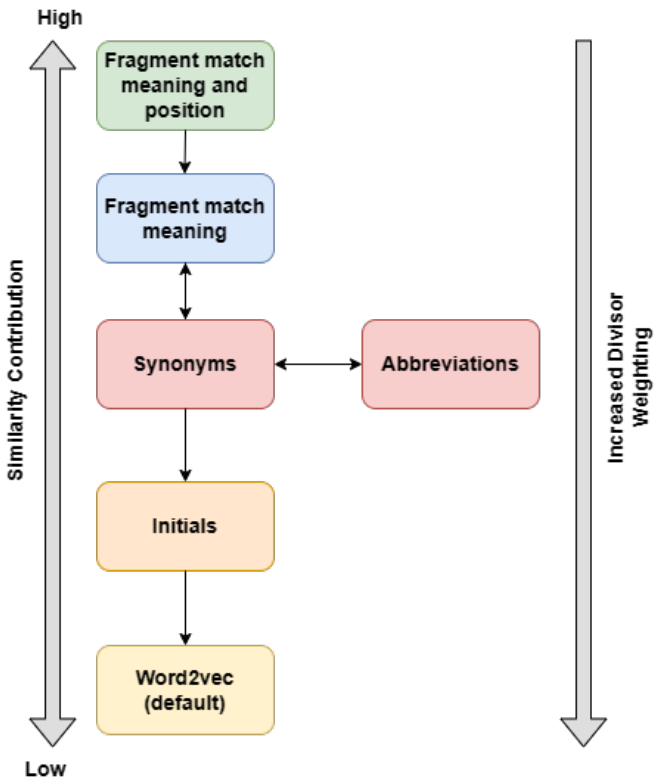

Figure 1: Algorithm hierarchy of measurement

professionals describe a patient's symptoms in a similar, but not necessarily identical, manner. Some terms and phrases persist regardless of the transcriber, while differences are influenced by environment, societal and historic factors and not just medical differences [13]. In practical terms, these factors would be the transposition of words or phrases (Figure 1), abbreviation of terms due to limited space or time, and the use of synonyms for some words or phrases. SSIM, a layered approach, considers many of these factors when looking for points of similarity between two bodies of medical text using a weighted scale to influence the importance of these points and their contribution to the overall similarity measure.

As shown in Figure 1, SSIM works by moving from *specific* to more *general* comparisons between two strings, building a measure of similarity in the process. The simplest measure of similarity is when the corresponding characters in both strings being compared are the same at all relative positions from the start of the string. This implies that if the two strings are of different lengths or there is a mismatch of capitalisation within the strings then they are not the same. SSIM considers that a string's *meaning* is a more accurate measure of similarity, making factors such as capitalisation irrelevant.

At the most specific level each string is broken into string fragments, usually words, based on the position of white space characters. It is these string fragments that are initially compared in order to determine similarity. The length of each string is now considered to be the number of fragments rather

than the number of literal characters in the string. Individual character level comparisons are used as the default level of comparison if all other techniques have failed to match all of the string tokens. We consider each fragment in the string being compared in turn applying a number of weighted rules to determine similarity based on relative position.

- The pair of fragments being compared at the same relative position have the *exact* same content.
- The fragment in the source string exists within the other string but at a different relative position.
- The fragment in the source string has a partial match with the fragment at the same relative position in the other string.
- The fragment in the source string has a partial match with a fragment in the other string at a different relative position.
- The fragment in the source string doesn't not exist in the other string.
- Some of the characters in the source string match those in the other string.

For those string fragments that have no match of any form in the second string, we then apply other techniques to see if there are any other meanings that can be applied to extract a degree of similarity. SSIM uses a table of common public domain nicknames as a secondary means of comparing fragments (i.e. names) between strings. We identify a weighted partial match if a name in one string can be matched with a suitable nickname in the other string. For those fragments that still cannot be matched we compare initials using the following top down rules. In SSIM, initials are considered to be stylised types of name abbreviations. As we move further down the hierarchy of comparative rules the influence in terms of the contribution to the overall similarity weighting is reduced.

- Can we match a string fragment to a matching initial fragment in the other string
- Can we match a string fragment to an incorrect initial fragment in the other string

The logic here is that it is a worse match to match a name to the wrong initial than to not being able to match it to any initial at all, that is if no matching initial is found there is still more likelihood that there could be a match than to a an initial that is definitely wrong. Consider the following example: ***"John Smith"*** is a stronger match (by some definition [14]) to ***"J. Smith"*** than ***"B. Smith"*** and a weaker match to just ***"Smith"***.

If, after all these techniques has been applied, we still have fragments that are not matched against fragments in the other string we default to considering letter grouping characteristics to try and find a degree of similarity that is less obvious. At the end of the process, we will get a measure of how similar the two strings under comparison are. This is analogous to the cosine similarity used for numeric data previously and can be taken forward when considering authentication and authorisation concerns in access control.

Below we discuss the various similarity techniques used starting with the most significant and working towards the minor matching scenarios.

A. Segmentation of textual data represented in a different order

As a secondary step to the other techniques, we propose to break strings down into components, i.e., phrases or segments, to pre-process the data. Once segmented, similarity is then measured by considering a number of factors, including the relative position of identical tokens in each string and the existence of similar names or words. These factors are then used to determine the set of criteria that are used in measuring similarity within our algorithm. When comparing two processed strings, it is a matter of comparing corresponding pairs of components from one string in turn against all elements of the other string. There are three possible outcomes for any given component comparison (in decreasing order of relevance). For two strings s_1 and s_2;

1) Same component exists in same relative position.
   $\exists n \in \mathbf{N}, s_1[n] = s_2[n]$.
2) Matching component exists in the other string but at a different relative position.
   $\exists m, n \in \mathbf{N}, s_1[m] = s_2[n]., m \neq n$
3) Matching component does not exist in second string.
   $\nexists\, m, n \in N, s_1[m] = s_2[n]$

This approach raises the idea of creating a weighted value for each fragment. This value will need to be normalized against the length of the shortest string since the actual value size will be a factor of string length meaning that longer strings will yield larger values which is counter productive.

The basic premise here relies on the idea that a longer set of matching phrases between two strings are a better indicator than shorter matching phrases and relative phrase order is significant. Consider the string *"The dog barks"*. Applying the concept above then the string *"dog barks"* should be considered a better match than *"dog the barks"*, on a purely textual basis and ignoring grammar implications. Similarly for a search *"The big house on the hill"*, the string *"house on the hill"* should be considered a stronger match than the shorter string *"big house"*. A simple weighted mapping table can be setup to provide a value for the comparison of two strings that gives preference to longer matching sub-strings as well as exact matches in both length and relative position.

- w_{exact} = 1.0, w_{none} = 0.0
- $w_{transpose} \geq w_{synonym} \geq w_{nickname} \geq w_{abbrev} \geq w_{initial} \geq w_{none}$
- w_{exact} = max(1.0, $w_{transpose}$, $w_{synonym}$, $w_{nickname}$, w_{abbrev}, $w_{initial}$)
- w_{none} = min(0.0, $w_{transpose}$, $w_{synonym}$, $w_{nickname}$, w_{abbrev}, $w_{initial}$)

Fragments which have the most similar meaning are weighted in priority order with the highest weightings to emphasises the degree of similarity.

Strings which are an identical match in all cases will yield a weighted value of 1.0, while with no matching characteristics score zero. A similarity measure based solely on fragment ordering for two strings, $sim(s_1, s_2)$ is given by

$$sim(s_1, s_2) = (\frac{1}{n * w_{exact}}) \sum_{i=1}^{i=n} weight(i)$$

further refinement can be achieved by considering abbreviations, punctuation and initials as a secondary measure. Our model treats nicknames and abbreviated words to be the same thing, namely a shortened version of a longer string still having the same meaning. As such their identification will offer the same level of contribution to a similarity measure.

B. Data set sensitivity

SSIM uses a layered/hierarchical approach to suit the relative importance of characteristics found in the source data sets. As such, the results it produces can depend on the nature of these characteristics. For example, the results for *Transposition - Single Pair* and *Transposition - All Tokens* in Tables I - IV where the three comparison algorithms, JI, SD and CS each measures similarity based on the presence or absence of the same token in the two strings being compared. The set of tokens are thus treated mathematically merely as an *unsorted bag*. This shows a potential for these algorithms to overestimate similarity where the order of tokens is relevant and not just their presence. The layered algorithm is slightly less successful in such cases because we consider tokens that match in position and content exactly to be more significant than tokens matching in different positions. It would be possible to equalise values in our weighting tables to desensitise the layered algorithm to give similar results if required. The algorithm has been designed to be extendable so that new characteristics may be modelled and added. The layered algorithm currently implements the characteristics described in Figure 1 in a purely *sequential* manner starting with the top level (*matching token content and position*) and ending with comparing individual characters in individual words. Each characteristic is weighted to reflect its significance, controlling its influence on the overall similarity metric. The weightings used to achieve the published results were tuned by trial and error against the specific data sets we used for our testing. Other data sets may needs weightings to be adjusted.

C. Nicknames

Nicknames do not necessarily correspond clearly to a person's given name. As such, it is impossible to provide a full list of all possible nicknames, with regional variations, for all common given names. We use a defined list of known nicknames for a given name. The source of these nicknames was taken from a public database. Nicknames are derived from a unidirectional many to many mapping between a subset of names in the domain name list to a subset of names in the co-domain nickname list. There are several scenarios that are resolved by this mapping approach. Give a domain of complete names, N, and a co-domain of nicknames, M and a mapping function $f(n \in N) \mapsto M$

- A given name, $n \in N$, may have no nicknames. $f(n) = \emptyset$.
- A given name, $n \in N$, has a single nickname $m \in M$. $f(n) = m$.
- A given name, $n \in N$, has multiple nicknames. $f(n) \mapsto [m_1, m_2, ..., m_k]$, where $m_1, m_2, ...m_k \in M$
- A given nickname, $m \in M$, is the nickname of multiple complete names. $f(n_1) = f(n_2) = ...f(n_k) = m$, where $n_1, n_2, ...n_k \in N$

During the similarity checking process each element in the source string can be replaced recursively by each mapped nickname in turn to find the strongest match.

D. Abbreviations and synonyms

Initials are stylized (minimal) versions of nicknames or abbreviations and can easily use the same reduction techniques as nicknames and punctuation simplification. It should be possible to used stylized initials to compare against full names or nicknames. However, we need to be careful that any matches are not treated as anything more than potential approximations when resolving the data records they match against. Each word may have zero or more synonyms. If a word has no alternatives then it must be spelt in full each time. However, if one or more synonym is known each may be a valid candidate for substitution when determining similarity. This implies, counterintuitively, that a word could be a potential replacement for itself thereby creating the following situation.

Consider a function $alias(s_1, s_2)$ which maps a string s_1 to a candidate synonym s_2. The function will yield true if s_2 is a known synonym or abbreviation of s_1 and false otherwise. For a string s and a set of synonyms, S, given by $s_1, s_2, ..., s_n$ under the function $alias()$ we get the cyclic Abelian group defined by [15]:

- $\forall s \in S$,alias(s, s)$= true$
- $\forall a, b, c \in S$,alias(a, b)$\wedge$alias(b, c) $\implies$ alias(a, c)
- $\forall a, b \in S$,alias(a, b) $\implies$ alias(b, a)

In practical terms, the first statement implies a name or abbreviation is its own abbreviation; the second makes the connection between chains of abbreviations all being mutually connected thereby defining a closed set of words or phrases that become interchangeable and finally the third statement specifies the two way nature of the mapping between two words reinforcing the the first two rules.

IV. Performance evaluation

In this section we evaluate the performance of our semantic similarity based approach (SSIM), using 4 sample address data sets and compare the results with three other common token based techniques, Jaccard Index (JI), Sorensen Dice (SD) and Cosine Similarity (CS).

A. Implementation overview

In our implementation, each string is broken down into its constituent (*fragments*) using spaces and other non critical white space characters as delimiters. We then attempt to map each fragment in T to a **unique** fragment in R by applying a series of techniques in turn to find the fragment in T which has the most similar *meaning* to the current fragment in R

under consideration. Please note that T and R are the two sets of tokens obtained by breaking down the two strings being compared. Our algorithm SSIM uses a series of mapping tables to map known words or phrases that are expected to exist in an individual's personal details to words which have the same or similar meaning, such as synonyms, abbreviations and nicknames. These mappings have been taken from public domain sources and may be supplemented over time to give a higher degree of mapping confidence. This process repeats for all fragments in T until all the possible algorithm choices have been applied. There are two possible outcomes to the algorithm; one is that each fragment in T has been paired with a corresponding fragment in R or there is at least one fragment in T where no fragment in R has been found to have a similar meaning. For all such unpaired fragments in T we consider letter grouping to determine any measure of non-obvious similarity for the remaining fragments in R. At each stage, or level, within the comparison hierarchy (Figure 1) the measure of similarity between any two fragments, $sim(R_i, T_i)$ with n mapped fragments, is diluted by a corresponding weighting factor, $weight_k$ to influence the pairs contribution to the overall similarity measure. Those fragments found to have a match at the highest level, where matches are most literal, such as exact word matches, will have their influence devalued the least in comparison to matching a single initial in a fragment. The overall similarity measure between T and R, $S(R,T)$ is given by

$$S(R,T) = \frac{1}{weight_k} \sum_{i=0}^{i=n} sim(R_i, T_i) \tag{3}$$

The data sources used in the evaluation were a set of 4 free open source, comma separated flat text file containing 500 UK/USA/Canadian and Australian names and addresses[1]. Although the focus was on the UK data source but the data sets from the other three countries were used as a means of comparison based on their slightly different formats and localised abbreviations.

B. Testing strategy

The testing process aims to compare the accuracy and distribution of results when comparing a set of test records against a simulated data set of incomplete records. A number of test records, m, are chosen at random from a larger data pool of n records. For our testing purpose we used 4 data files, each containing localised (i.e., geographically specific formatting of names and addresses) content of exactly n = 500 records. From this pool a random selection of 10% (m=50) records were chosen for comparative purposes. The process is repeated so that each of the n data records are used as test records during an iteration and as such n/m iterations are performed and an average result for each metric is taken.

Each of the three baseline algorithms (i.e., JD, SI and CS) were tested alongside our algorithm (SSIM) to compare both the accuracy of prediction and the distribution of the closest matching records. We maintained a list of the closest ten ($= N$) records for each of the algorithms. The similarity of each test record in turn was measured against each record in the data pool and the degree of similarity was stored. Let T_i denote the number of test records lying within the closest i records of the actual data record. In particular, let T_1 denote the number of precise matches against the actual data record, T_3 denotes the number of test records lying within the closest three most similar records up to T_{10}) denoting the number of test records lying within the closest ten most similar records. In the best case scenario, all of the test records would be matched against the correct record from the data pool, such that

$$\forall i \in \mathbb{Z}^+, 1 \leq i \leq n, T_i = \begin{cases} m & \text{if } i = 1 \\ 0 & \text{Otherwise} \end{cases} \tag{4}$$

and the worst case scenario would be that none of the test records are found to lie within the closest ten records

$$\forall i \in \mathbb{Z}^+, 1 \leq i \leq n, T_i = 0 \tag{5}$$

These two values provide a limited range of values showing the distribution of how accurately each algorithm is able to predict the correct test record.

Each of the test categories (i.e., baseline tests, capitalisation, transposition and associations) is designed to focus on a known problem area when comparing medical data records. Each test is ran using all records in the data set for each local file and a count is maintained showing the spread of each predicted test record against the most actual most similarly matched record. It is important to note that these tests are not only used as a measure of where SSIM provides more accurate results when compared to the baseline algorithms, but also to show that SSIM is at least comparable with existing token algorithms in the space when no improvement is possible.

The contents of the 4 test data files varied by including localised abbreviations and word abbreviations suitable for each locale thereby allowing us to compare the effectiveness of SSIM algorithm against these specific localised differences. The reason for choosing these data files is that they provided sufficiently varied data which could be used to exercise the different characteristics used within SSIM, namely capitalisation, transposition and substitution (both nicknames and synonyms) and so offer an insight into how the SSIM algorithm would perform on personal medical data in each case.

The testing process involved considering each data file in turn, from which a number of test records (m) was selected from the entire data set (n). A suite of different tests were then performed on these test records in turn in order to measure the effectiveness of out algorithm. Each test record, in turn, was modified by removing a *random* number of fields to simulate missing data. These were then presented to the complete data set, one record at a time, and a measure of similarity was determined using each of the 4 token based algorithms, JD, CS, SD and SSIM. The accuracy and distribution of similarity matches were recorded across all of the test records. This

[1]https://www.briandunning.com/sample-data/

process was repeated for all of the test categories and then restarted for each of the four data files.

It is worth considering that the *JI* and *SD* metrics each produce higher degrees of similarity when the number of tokens in each of the two strings under comparison are the same. When sizes of the two sets of text tokens differs the decrease in similarity is proportional to the difference between the sizes of the two token sets due to the reduced value of each ratios numerator. In this case, given two token sets t_1 and t_2 we can rewrite the previous equations 1 and 2 respectively as

$$J(t_1, t_2) = \frac{min(|t_1|, |t_2|)}{|t_1 \cup t_2|} \quad (6)$$

$$SD(t_1, t_2) = \frac{min(|t_1|, |t_2|)}{|t_1| + |t_2|} \quad (7)$$

C. Results overestimation and comparison

A general analysis of the results produced by each algorithm, (Figure 2), highlights an issue which leads to an over estimation of similarity by each of the three baseline methods. When the free text under comparison is relatively small in terms of the number of words/tokens, we see a sparsely populated list of most similar matches compared to SSIM. In the case of *JI*, often only a single record is being matched against a test record with no alternative close matches. This is due to data records having very dissimilar contents which in turn yield a similarity score of zero for almost all of the other data records. As such by default the only record which comes close to matching the test record being the original data record. This gives an artificially high degree of match which in turn leads to a higher number of exact matches than we would expect with a larger free text pool with more similar words. Although *SD* and *CS* can yield more than one possible match in general the similarity of the most similar record tends to be lower than that indicated by SSIM.

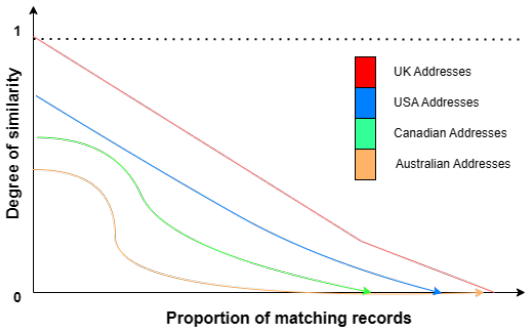

Figure 2: Results distribution for each localised file using SSIM

D. Results analysis and discussion

The compared token-based algorithms fell into two groups: the JI, SD and CS algorithms produced higher measures of similarity for those categories which used token *representation* for comparison, such as *literal comparison* and *transposition* where the actual token contents were the objects of comparison. This highlights the drawback of such techniques when comparing the meaning of text: superficial attributes (e.g capitalisation) may negatively influence the similarity measure. In contrast, SSIM produced higher measures of similarity for those categories that relied on determining the *meaning* contained within text, such as the *associations* category which included the use of nicknames, abbreviations and synonyms. Such categories ignore superficial encoding attributes since they have little influence on the meaning of the strings being compared. The results achieved by SSIM were comparable to those obtained from the three baseline algorithms in the more literal test categories. (Tables I - IV)

Table I: Similarity measurement results using UK personal details

Test Category		Average Token Based Similarity Measure			
		JI	**SD**	**CS**	**SSIM**
Baseline Tests					
	No changes	100%	100%	100%	100%
Capitalisation					
	Same class	17.3%	29.5%	46.7%	100%
	Random case	17.9%	30.4%	47.5%	100%
	Random chars	18.8%	31.3%	48.3%	100%
Transposition					
	Single token pair	100%	100%	100%	96.7%
	All tokens	100%	100%	100%	96.7%
Associations					
	Nicknames	47.7%	51.7%	74.6%	57.5%
	Abbreviations /Synonyms	1.5%	2.9%	3.3%	93.3%

The results obtained using the four localised data files show that, with slight regional variations, they are consistent irrespective of national or geographical formatting and notation techniques used. A closer alignment could be achieved if the repositories of nicknames, abbreviations and synonyms (and any other semantic measurement that may need to be considered) were broadened to include a more geographically diverse set of words and phrases aligned with a target domain or location. Larger data repositories could, however, impact performance negatively since the number of potential matches could increase dramatically, especially if the size of the sets of tokens extracted were particularly large.

V. Conclusion

This paper introduced a new similarity measurement algorithm for textual data recorded in medical records. Our algorithm can be used as a basis for determining the identity of a person in an access control scenario within a medical facility. The performance evaluation of our algorithm shows that it compares well to the three baseline algorithms and outperforms them when subtle differences such as abbreviations, nicknames and synonyms were introduced. This work provides an alternative to blind content matching approaches by using a degree of domain-sensitive meaning to the data. The results of the tests performed highlight that our algorithm performed as well as the three baseline tests and slightly better with those tests specific to meaning rather than content.

Table II: Similarity measurement results using American personal details.

Test Category		Average Token Based Similarity Measure			
		JI	SD	CS	SSIM
Baseline Tests					
	No changes	98.3%	98.3%	100%	100%
Capitalisation					
	Same class	19.0%	31.7%	49.2%	100%
	Random case	19.6%	32.6%	49.8%	100%
	Random chars	18.5%	30.8%	48.5%	100%
Transposition					
	Single token pair	98.3%	98.3%	100%	97.2%
	All tokens	98.3%	98.3%	100%	96.6%
Associations					
	Nicknames	46.8%	47.4%	83.7%	63.8%
	Abbreviations /Synonyms	0.0%	0.0%	0.0%	100%

Table III: Similarity measurement results using Australian personal details.

Test Category		Average Token Based Similarity Measure			
		JI	SD	CS	SSIM
Baseline Tests					
	No changes	100%	100%	100%	100%
Capitalisation					
	Same class	17.1%	29.2%	50.8%	100%
	Random case	17.8%	30.1%	51.7%	100%
	Random chars	18.4%	30.5%	52.1%	100%
Transposition					
	Single token pair	100%	100%	100%	96.7%
	All tokens	100%	100%	100%	96.7%
Associations					
	Nicknames	61.3%	64.2%	91.7%	72.5%
	Abbreviations /Synonyms	0.0%	0.0%	0.0%	91.3%

Table IV: Similarity measurement results using Canadian personal details.

Test Category		Average Token Based Similarity Measure			
		JI	SD	CS	SSIM
Baseline Tests					
	No changes	100%	100%	100%	100%
Capitalisation					
	Same class	21.8	35.8%	40.0%	100%
	Random case	26.1%	41.2%	45.0%	100%
	Random chars	24.1%	38.4%	42.5%	100%
Transposition					
	Single token pair	100%	100%	100%	96.7%
	All tokens	100%	100%	100%	96.7%
Associations					
	Nicknames	46.9%	48.3%	62.8%	54.2%
	Abbreviations /Synonyms	0.0%	0.0%	0.0%	94.2%

REFERENCES

[1] M Lee, B. Pincombe, and M. Welsh. Predicting semantic similarity between clinical sentence pairs using transformer models: Evaluation and representational analysis. *JMIR Med Inform 2021;9(5):e23099*, 2021.

[2] M Lee, B. Pincombe, and M. Welsh. An empirical evaluation of models of text document similarity. *Proceedings of the Annual Meeting of the Cognitive Science Society. 2005 Jul Presented at: Annual Conference of the Cognitive Science Society; 21-23 July 2005; Stresa, Italy*, pages 1254–1259, 2005.

[3] A. Frankenberg-Garcia, R. Lew, and G.P. Lees. Slipping through the cracks in e-lexicography. *International Journal of Lexicography, Volume 34, Issue 2, June 2021*, pages 206–234, 2021.

[4] G. Gimaletdinova, L. Khalitova, V. Solovyev, and V. Bochkarev. Lexicographic study of synonymy: Clarifying semantic similarity between words. *Computación y Sistemas. 25. 10.13053/cys-25-3-4028.*, 2021.

[5] A. Kosmutzky, T. Nokkala, and S. Diogo. Between context and comparability: Exploring new solutions for a familiar methodological challenge in qualitative comparative research. *Special Issue: Towards a Methodology Discourse in Comparative Higher Education, Volume 74, Issue 2*, 2020.

[6] L. Hantrais. Methodological pluralism in international comparative research. *International Journal of Social Research Methodology, 17(2), 133–145.*, 2014.

[7] D. Vatsalan, Z. Sehili, P. Christen, and E. Rahm. Privacy-preserving record linkage for big data: current approaches and research challenges. *Handbook of Big Data Technologies, pp. 851–895. Springer (2017)*, 2017.

[8] S. Vaiwsri, T. Ranbaduge, and P. Christen. Accurate and efficient privacy-preserving string matching. *Int J Data Sci Anal (2022)*, 2022.

[9] P. Christen, T. Ranbaduge, and R. Schnell. Linking sensitive data: Methods and techniques for practical privacy-preserving information sharing. *Springer International Publishing AG (2020)*, 2020.

[10] Z. Liptak. Bioinformatics algorithms - q-gram distance, 2019. (Last accessed : 1st August 2023).

[11] S. Prabhakaran. Cosine similarity – understanding the math and how it works., 2018. (Last accessed : 2nd August 2023).

[12] K. H. Zou and al. Statistical validation of image segmentation quality based on a spatial overlap index. *Academic radiology, 11(2)*, pages 178–189, 2004.

[13] I. Surano. Challenges of integrating hetergeneous data sources, 2020. (Last accessed : 2nd August 2023).

[14] Fotis Aisopos, George Papadakis, Konstantinos Tserpes, and Theodora Varvarigou. Content vs. context for sentiment analysis: a comparative analysis over microblogs. *HT'12 - Proceedings of 23rd ACM Conference on Hypertext and Social Media*, 2012.

[15] Neil Strickland. Algebraic theory of abelian groups. 2020. (Last accessed : 1st August 2023).