Name Similarity for Composite Element Name Matching

Conference Paper · October 2016

DOI: 10.1145/2975167.2975203

4 authors, including:

Naveen Ashish
Independent Researcher
2 PUBLICATIONS **1** CITATION
SEE PROFILE

Arihant Patawari
TNeGA
4 PUBLICATIONS **34** CITATIONS
SEE PROFILE

Name Similarity for Composite Element Name Matching

Naveen Ashish
Laboratory of Neuro Imaging
USC Stevens Neuroimaging and Informatics Institute
Keck School of Medicine of USC
University of Southern California
2001 N Soto Street,
Los Angeles, USA
+1-323-442-0142
nashish@loni.usc.edu

Arihant Patawari
City of Hope
National Medical Center
1500 East Duarte Road
Duarte CA 91010
USA
apatawari@coh.org

Simrat Singh Chhabra
Laboratory of Neuro Imaging
USC Stevens Neuroimaging and Informatics Institute
Keck School of Medicine of USC
University of Southern California
Los Angeles, USA
simratsc@usc.edu

Arthur W. Toga
Laboratory of Neuro Imaging
USC Stevens Neuroimaging and Informatics Institute
Keck School of Medicine of USC
University of Southern California
Los Angeles, USA
toga@loni.usc.edu

ABSTRACT

Background and Objective: Matching corresponding data elements is a critical problem in biomedical data harmonization for data sharing. The similarity of the element names is one of the many factors employed in determining data element matches. Determining name similarity is complicated by the fact that data element names in biomedical data are composite i.e., composed of multiple components. We provide a better approach to determining element name similarity for composite element names.

Methods: Our solution is based on decomposing composite element names into constituent components and then determining the name similarity by comparing corresponding components across element names. We use a machine-learning based classification approach to the problem, building upon a field-by-field matching model from record-linkage techniques.

Results: The element name similarity achieved by our approach is significantly superior to existing string matching techniques. The element name similarity metric consequently improves the matching accuracy of element matching systems overall

Conclusions: Our approach is effective and has been integrated as part of a more comprehensive "schema-mapping" system, which we have developed for harmonizing biomedical datasets.

CCS Concepts

• **Information systems→ Information integration • →Entity resolution.**

BCB '16, October 02-05, 2016, Seattle, WA, USA
© 2016 ACM. ISBN 978-1-4503-4225-4/16/10…$15.00
DOI: http://dx.doi.org/10.1145/2975167.2975203

Keywords

Element name matching; String similarity; Bio-medical data integration; Record-linkage; Machine-learning classification

1. BACKGROUND AND OBJECTIVES

This paper presents a solution for determining the likelihood that two biomedical data elements are the same, based on analyzing the element names. This likelihood is captured by an *element name similarity* measure which is a score 0-1 with 1 representing an 'exact' match of elements based on their names. Our work is part of a larger system which we have developed for automatically matching biomedical data elements. Elements are matched based on *features* of the elements provided in a data dictionary, such as descriptions of what each element essentially is and other metadata such as the element data type, range of permissible values etc. The similarity of element names is another feature that can help in determining corresponding data elements, for instance 'GENDER' and 'GNDR' could denote two data elements but both correspond to a patient's gender. However, given the complex nature of element names in biomedical datasets we found determining element name similarity using existing string matching techniques to be only partially effective. Our solution factors the complex nature of biomedical data element names in determining the element name similarity.

Our overall interest is in data sharing and harmonization of biomedical datasets from multiple, distributed, and independently created repositories of data. Our work is in the context of the "GAAIN" project [1] - a data sharing network for Alzheimer's disease researchers, providing investigators with access to harmonized Alzheimer's disease data from around the globe. Matching data elements i.e., identifying corresponding elements referring to the same entity across different datasets is a fundamental aspect of the data harmonization in GAAIN. It is also a very resource and time intensive task requiring considerable manual effort for each new dataset. This is a significant concern, as a typical GAAIN dataset contains 2000 to 10,000 distinct data elements. We currently have data from over 80 data partners around the globe, and this number increases by the week and month. We have developed a system called the GAAIN Entity Mapper (GEM) [2] which is an automated data-mapping software tool for element matching.

This paper describes the more sophisticated element name similarity approach which we have recently integrated into the overall GEM system. We have evaluated this approach for a) its effectiveness in matching data elements using only element name similarity as a feature, and b) the impact on the matching accuracy of the overall GEM system.

While name or string matching techniques are well developed in database technology as well as in bioinformatics, the problem is complicated by the fact that biomedical element names are often *composite*. As an example, the elements 'PTDEMENTIA' and 'PATIENTDEM' could be corresponding composite element names that refer to a data element regards whether a patient is demented or not. Formally, a composite name is a name (string) that is composed of multiple *components*. For instance, both the example element names in the example above are composite as the first name has two components (PAT, DEMENTIA) and the second name also has two components (PATIENT, DEM). Existing string matching techniques, such as based on the notion of *edit-distance* [36] which is a widely used string similarity measure, do not apply very well to composite strings as they consider the string as one monolithic entity. Composite strings should instead be matched intelligently in a component by component fashion. At the components level one can see that the string 'PAT' is a prefix of 'PATIENT' and 'DEM' is an abbreviation for 'DEMENTIA' in this domain. Our approach is based on matching individual components in composite element names. We decompose element names into components, identify relationships between corresponding components, and determine similarity using machine-learning classification.

2. MATERIALS AND METHODS

We provide a brief summary of our overall element matching approach and system, and then describe the details of our approach to determining element name similarity. First however we clarify the terminology:

- A *data element* is a fundamental unit of data described in a data dictionary, for instance a patient's gender, age or their 'MMSE score' would all be data elements. Since the context here is exclusively that of *data* elements, the term 'element' will imply data element henceforth.
- An *element match* is the determination that two data elements are the same or corresponding elements, for instance establishing a match between two elements such as MMSE and MMSCORE (that correspond to the same element).
- The *element name similarity* is a measure of the how similar the names of two data elements are, and is a value 0-1 with 1 denoting a perfect name match.
- An element for which we must find matches is called a *source element* and the dataset or schema it belongs to is called the *source*. The dataset or schema for which we must find matches for this given element is called the *target.*
- Finally, the terms 'matching' or 'similarity' can be assumed to imply element matching or similarity.

2.1 Data Element Matching

In the GEM system we take a classification approach to determining matches, where a classifier determines whether two given elements match or not given a number of features about the pair of elements [2]. These features are synthesized from element descriptions and other metadata details provided in associated data dictionaries. The features, as Figure 1 illustrates, include metadata about the individual elements such as the data type, and range or set of permissible values, etc. (this is the box labeled 'METADATA' in the figure). We also have a text similarity feature (box labeled 'TEXT SIMILARITY) which is a measure of similarity of the associated text descriptions for the two elements; this text similarity measure is computed based on a topic-modeling approach described in [2]. Finally we have a feature representing the element name similarity of the two elements (the box labeled 'NAME SIMILARITY'). Whereas 'TEXT SIMILARITY' looks at the descriptions of the two elements, 'NAME SIMILARITY' is based on the names of the elements themselves.

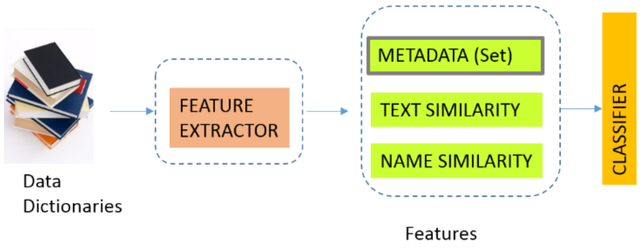


Figure 1. Determining Element Matches. *Overall element matches are determined using a classification approach, where name similarity is one of the features.*

Some degree of semantic resolution is also done over the text descriptions using ontology terms from SNOMED and UMLS [2] though more details on such are outside the scope of this paper. This work essentially is aimed at improving the feature provided by the NAME SIMILARITY box.

2.2 Name Similarity Approach

Our overall approach to name similarity determination is to train machine-learning classifiers to determine if any given pair of element names match (Y) or not (N), based on the element name, and the name similarity is then the classifier confidence in that match. The matching process and the matching system architecture are illustrated in Figure 2.

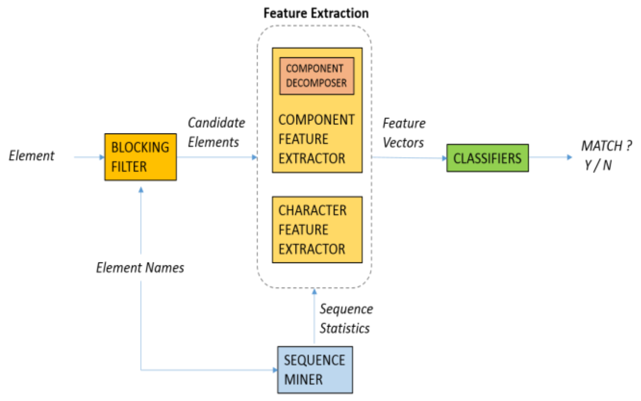


Figure 2 . GEM Name Similarity System Architecture. *A source element is provided as input to find a match. The sequence miner pre-processes all element names to find sequence frequencies. The blocking filter eliminates improbable target elements. The feature extraction module creates feature vectors, with multiple options, that are passed on to the classifier.*

Given a source element we first use a blocking filter [18] to select probable candidate matches for that element. This is the module labeled 'BLOCKING FILTER' (Figure 2) which exhaustively scans the elements in the target and selects as target matches, only those elements for which the edit-distance [36] match score with the source element is above a specified threshold. We then consider pairs of the source element with each of the candidate elements and generate features per pair. We have developed and evaluated features of different kinds over element pairs. The feature generation also leverages some *sequence statistics* that we distill from the set of all source and target element names. We describe the key steps in Figure 2 in more detail below.

2.3 Components Model for Element Names

Based on our observation of element names in Alzheimer's disease and other biomedical data, we have developed a model for such composite element names. We model an element name as composed of three components. These are:

(i) The PREFIX, which is a prefix qualifier typically describing the element or providing the group or table name (in the case of a database) which the element belongs to.

(ii) The CONCEPT, which is the data item that the element is essentially capturing.

(iii) The SUFFIX, which typically qualifies an aspect of the element such as the assessment year, or subject visit number etc.

The PREFIX and SUFFIX components may be empty in some cases but the CONCEPT is always a string of non-zero length. Table 1 provides some examples of decomposed element names. The concept component comprises data items such as dementia (DEM), or the patient gender (GENDER), PT is a prefix denoting that we are referring to the gender of a patient, and we have suffixes such as YR1 (for Year 1) and V1 (for visit one) qualifying when the data item was captured.

Table 1. Element Component Illustration. *An example of breaking an element name into three fundamental components – the prefix, concept and suffix*

ELEMENT	**PREFIX**	**CONCEPT**	**SUFFIX**
MOMDEMYR1	MOM	DEM	YR1
PTGENDER	PT	GENDER	
MMSEV1		MMSE	V1

2.4 Pre-processing for Sequence Statistics

The information about sequences of characters in element names, and their associated frequencies of occurrence over the entire collection of names is useful for synthesizing features. The 'SEQUENCE MINER' is a pre-processing module that identifies all sequences and occurrence frequencies of all substrings of length two or larger that occur in the entire set of element names. A high frequency of occurrence of a sequence is an indicator that the sequence may represent a "meaningful" string i.e., a component. For example a high occurrence of the sequence 'PT' is an indicator that it is a commonly used prefix abbreviation (for 'PATIENT' in this case). We employ a simple sliding-window based sequence mining algorithm [49] for extracting sequences with frequencies. The complexity of the algorithm is dependent on the number of elements and MAX (Length among element names). With a typical dataset having 2,000-10,000 data elements with a maximum element name length of 10 or 15 characters, this process executes in a few seconds on a typical contemporary laptop.

2.5 Feature Generation

We developed two approaches for feature generation. The first approach attempts to explicitly identify the prefix, concept and suffix components in each element name in a pair, and then generates features by determining the relationship between corresponding components across the two element names. We refer to this as *'component based feature generation'* or *'COMP'* and the features generated as *'component features'*. The second approach does not consider breaking element names into components but generates features based on the comparison of characters in the two element names, and we refer to it as *'character comparison based feature generation'* and the features generated as *'character features'*.

2.5.1 Component Based Feature Generation

The first step in component based feature generation is to actually decompose each of the element names in a pair to their constituent components. The decomposition process is heuristic based and we provide *two* different decompositions (of element names in a pair) using different heuristics.

The first heuristic is summarized as follows:

Step 1: Find the longest common substring across the two element names. Assign that common substring as the concept component for both element names.

Step 2: For each element name, the substring (if any) to the left of the concept substring becomes its prefix and the substring (if any) to its right becomes the suffix. These left and right substrings are referred to left and right residues.

For example, given the pair (PATMMSEYR1, PTMMSE), we would identify 'MMSE' as the longest common string to both names resulting in the decomposition <PAT,MMSE,YR1> for the first name and <PT,MMSE,NULL> for the second.

The second heuristic is more sophisticated and attempts to identify complete names of actual data items. Consider another example pair (PTGENDER, PTGEN). With the first heuristic we would identify 'GEN' as the longest common substring resulting in the decompositions <PT,GEN,DER> and <PT,GEN,NULL>. The decomposition of the first name is clearly erroneous as we would want it to be <PT,GENDER,NULL> instead. The process using the second heuristic can be described as follows:

Step 1: Determine the longest string across the two element names that is a valid *dictionary word*. Assign this as the concept of the element name it is part of.

Step 2: In the other name, determine an exact match or an abbreviation or shorthand notation for the above identified concept. Assign that as the concept for the other element name.

Step 3: Determine prefixes and suffixes for each element name using left and right residue strings as in the first heuristic.

In this example, we would now identify 'GENDER' as the longest valid dictionary word across the two element names and assign it as the concept for PTGENDER. From PTGEN, the string 'GEN' would be identified as an abbreviation of 'GENDER' as if is assigned as the concept for that name. The resulting decomposition is <PT,GENDER,NULL> and <PT,GEN,NULL>. Table 2 illustrates the pseudo-code for the element name decomposition step where the function 'decompose_1' is the

Table 2. Element Name Decomposition. *Algorithm for decomposing the element names into components.*

```
Input: name1, name2 which are a pair of element names
Return: Individual components of both element names

function decompose_2(name1, name2)
{
<concept1, concept2> ← determineConcepts(name1,name2)
<prefix1,suffix1>        ← residue(name1,concept1)
<prefix2,suffix2>        ← residue(name2,concept2)
return <[prefix1,concept1,suffix1],[prefix2,concept2,suffix2]>
}

function determineConcepts(name1, name2)
{
word1 ← longestDictionaryWord(name1)
word2 ← longestDictionaryWord(name2)
if (length(word2)>length(word1))
  concept1 ← word1
  concept2 ← abbreviation(name2,word1)
else
  concept2← word2
  concept1 ← abbreviation(name1,word2)
return{concept1,concept2}
}

function longestDictionaryWord(name)
{
S ← length(name)
for (S=L, S>1, --L)
  for(K=0;K<S-L,++K)
    possibleWord ← substring(name,K,S)
    if (isDictionaryWord(possibleWord)  return possibleWord
return null
}

function abbreviation(name,word)
{
L    ←  length(name)
W    ← length(word)
Cn   ← setOfCharacters(name)
Cw   ← setOfCharacters(word)
C    ←  Cw-Cn
word  ← eliminate(word,C)
W     ← length(word)
for (S=W, S>0,--S)
  possibleAbbreviation ← substring(word,0,S)
  if (name contains possibleAbbreviation) return possibleAbbreviation
return null
}


function decompose_1(name1, name2)
{
lcs        ← largestCommonSubstring(name1,name2)
concept1 ← lcs
concept2 ← lcs
<prefix1,suffix1> ← residue(name1,concept1)
<prefix2,suffix2> ← residue(name2,concept2)
return <[prefix1,concept1,suffix1],[prefix2,concept2,suffix2]>
}
```

simpler heuristic based on largest common substring across the element names and 'decompose_2' is the more complex approach based on recognizing dictionary words. A Dictionary API [26] is used for recognizing strings that are valid dictionary words.

The next step generates features. The 'COMPONENT FEATURE EXTRACTOR' module takes a pair of element names that have been decomposed into components and creates a feature vector for the element pair. A key set of features is based on how corresponding components relate to each other. Table 3 illustrates the 6 distinct types of qualitative relationships that we have defined for how components (strings) can relate to one another, accompanied with an example for each type. A pre-assembled domain specific synonym table is used for determining any SYNONYM relationships.

Table 3. Qualitative Relationships. *Relationships across two strings*

STRING 1	STRING 2	RELATIONSHIP
GENDER	GENDER	EXACT MATCH
GEN	GENDER	ABBREVIATION PREFIX
GNDR	GENDER	SHORTHAND
SEX	GENDER	SYNONYM
NULL	GENDER	MISSING
AGE	GENDER	OTHER

Table 4 illustrates the set of features that comprise the feature vector, with a description and the actual value of the feature for the element pair (PTGENDER, PTGEN). We consider features obtained from both decompositions into our feature vector, as our experimental results demonstrate that we get the best classification of element name matching when considering features of *both* decompositions as opposed to either (just) one of them. The frequency information, of the prefix, concept and suffix components of the two names is obtained from the sequence statistics collected during pre-processing.

Table 4. Features. *Lists the features generated with description and value for a particular component pair (PTGENDER, PTGEN).*

FEATURE	DESCRIPTION	VALUE
Prefix1	The actual string of prefix 1	PT
Prefix1Length	The character length	2
Prefix1Frequency	Frequency of occurrence cross all element names	45
Concept1	Actual concept string	GENDER

Concept1Length	The character length	6
Concept1Frequency	Frequency of occurrence cross all element names	1
Suffix1	Actual suffix string	NULL
Suffix1Length	The character length	0
Suffix1Frequency	Frequency of occurrence cross all element names	NA
Prefix2	The actual string of prefix 2	PT
Prefix2Length	The character length	2
…..		
Suffix2Frequency	Frequency of occurrence cross all element names	62
PrefixRelation	Qualitative relationship between prefixes	EXACT MATCH
PrefixMatch	Edit distance based match score	1.0
ConceptRelation	Qualitative relationship between concepts	SHORTHAND
ConceptMatch	Edit distance based match score	0.66
SuffixRelation	Qualitative relationship between suffixes	MISSING
SuffixMatch	Edit distance based match score	0
IsDictionaryWordConcept1	Is concept 1 a valid dictionary word?	Y
IsDictionaryWordConcept2	Is concept 2 a valid dictionary word?	N

2.5.2 *Character and Sequence Frequency Based Feature Generation*

In this approach, we generate features for a pair of element names based on character and sequence information in *corresponding* character positions. We describe two kinds of feature vectors that we designed and evaluated.

Character Comparison Based (CHAR-CC) For the first kind of feature vector, we look at specific character positions in the element names. To ensure meaningful features and also to ensure that the feature vectors from all instances have the same dimension, we conduct two pre-processing operations on a given pair of strings. The first operation is *alignment* where we identify the longest common substring across the pair and ensure that the longest common substring starts at the same absolute character position in both strings. This can be done by adding *padding characters* to the left of one of the strings. We have chosen '-' as the padding character and as an example given the pair of strings (HOPELESS, GDHOPE) we would transform it to (--HOPE,GDHOPE) i.e., we pad two '-' characters to the left of the first string so that the longest common substring i.e., 'HOPE' starts at the same (3rd) character position now in both strings. The second operation is *normalization* where we ensure that both strings have the same length. The feature vector length is determined and fixed in advance, let us call it V. If either string is shorter than this feature vector length we add the required number of padding characters to its right. In this example if this vector size length is 10 then normalization will transform the pair to (--HOPELESS, GDHOPE----).

Now we describe the feature vector details. The feature vector has size V. The vector element value at position i is defined as:

value = 1 if the characters at position i are the same

= 0 if the characters at position i are not the same

= 2 if either of the characters at position i is a padding character

-	-	H		O	P	E	L	E	S	S
G	D	H		O	P	E	-	-	-	-

For instance for the above pair of (aligned and normalized) strings the feature vector would be

[2, 2, 1, 1, 1, 1, 2, 2, 2, 2]

Common Substring Based (CHAR-CS) We first determine the longest common substring across the two given strings. The left residues in each of the strings (modulo the identified common substring) form the prefixes of the strings. The feature vector is composed of (i) The ratio of the length of the common sub-string to the length of the first string, (ii) The ratio of the length of the common sub-string to the length of the second string, (iii) The sequence frequency of the first string prefix and (iv) The sequence frequency of the second prefix.

With our existing example, the longest common substring to both strings is 'HOPE' which has a length of 4. The ratio of its length to that of 'HOPELESS' is 4/8 = 0.5 and to that of GDHOPE is 4/6 = 0.66. The prefix of the first string is an empty string and that of the second is identified as 'GD'. Assuming that the sequence frequency of 'GD' is 26, the feature vector would be

[0.5, 0.66, 0, 26].

2.6 Classification

The final step is that of classification. For each pair of datasets or schemas, we train the classifiers with manually identified positive and negative matches across the datasets. Given that a) the number of features in any of the feature alternatives is relatively small (less than 50) and b) the features comprise both nominal as well as numerical values, we evaluated classifiers such as Support Vector Machines (SVM) [9] with the Gaussian and RBF kernels, tree based classifiers with boosting such as Random Forest and Rotation Forest [5,39], Logistic Regression [5] and hybrid approaches such as Sequential Minimal Optimization (SMO) [34]. We also evaluated the Multi-Layer Perceptron [19] amongst the neural-network classifier family. We used the Weka toolkit [17] for the classifier application and evaluation.

3. RESULTS

We have experimentally evaluated the following:

Table 5. Mapping Accuracy. *This table provides name matching accuracy, as F-Measure (percentage) over various dataset pairs and for multiple feature options.*

Approach ➡ **Datasets Pairs** ⬇	**COMP** (Component based features)	**CHAR-CC** (Character comparison based features)	**CHAR-CS** (Common substring based features)	**String Matching**			
				FRIL	*Edit Distance*	*Jaro Winkler*	*Monge Elkan*
ADNI-NACC	81.6	21.0	78.7	39.0	24.4	24.2	26.7
ADNI-INDD	73.2	27.6	67.6	35.2	30.6	34.1	38.4
ADNI-LAADC	88.6	28.0	65.7	29.8	42.9	39.0	39.4
ADNI-DIAN	89.1	31.0	73.4	31.0	35.5	22.1	34.4
ADNI-CLSA	82.3	26.3	69.9	32.1	29.6	23.7	29.0
Averaged	83.8	25.4	70.7	32.6	33.3	30.1	30.5

(i) The name matching accuracy i.e., the accuracy of matching data elements based on *only* the element name similarity. A better element name matching accuracy provides us a basis for a better element name similarity measure.

(ii) The impact of an improved name similarity measure, on overall matching.

(iii) The evaluation of various applicable classifiers in our approach to name similarity, and their optimal configuration.

3.1 Data for Evaluation

We used data elements from multiple data sources of Alzheimer's disease data available in GAAIN [41]. We selected 6 datasets namely 1) the Alzheimer's Disease Neuroimaging Initiative (ADNI) [29,41] with 4800 elements, 2) the National Alzheimer's Coordinating Center database (NACC) [3] with 720 elements, 3) the Dominantly Inherited Alzheimer Network database (DIAN) [28] with 3100 elements, 4) Integrated Neurogenerative Disease Database (INDD) [48] with 2200 elements, 5) Layton Aging and Alzheimer's Disease Center database (LAADC) [47] with 5100 elements and 6) Canadian Longitudinal Study of Aging (CLSA) (CLSA 2015) with 6250 elements [35]. We manually created truth sets of element data mappings across different pairs of these data sources. On an average there are about 100 instances (pairs) of mappings across each pair of data sources. We also included negative instances across dataset pairs in the truth sets. Negative instances are significant as they comprise about 55% of elements across any two datasets.

3.2 Specific Evaluations

We evaluated the mapping accuracy, in terms of precision and recall, for this data. We also provide a comparison with the baseline approaches of three string matching algorithms relevant to this problem, namely Edit-Distance, Jaro-Winkler and Monge-Elkan [8,30]. We also compared mapping performance with that provided by the "FRIL" system for record-linkage. FRIL [22] is a fine-grained record integration and linkage tool which extends traditional record-linkage tools with a richer set of parameters. Our use of FRIL is as a special case where a 'record' contains exactly one attribute, which is the element name.

3.2.1 Mapping Accuracy

Table 5 provides the accuracy as F-Measure (percentage) for our component and character feature sets, as well as for the various string matching algorithms and the FRIL linkage tool. The model generated for evaluating mapping accuracy is 10-fold cross validated and we provide results for 5 out the 15 pairs (for brevity) of the datasets we evaluated and also the accuracies averaged over all 15 pairs.

From the results, first, we achieve an absolute mapping accuracy of over 80% for most dataset pairs with our approach. Second, the component features are significantly more effective than the character features. This validates our component based model of element names where breaking element names into their constituent components results in significantly more accurate name matching. Third, the mapping accuracy is significantly higher than that achieved by existing string matching algorithms which provide a mapping accuracy of at best 42.9% (as shown in Table 5) on the various datasets. This implies that a name similarity measure based on the component based name matching technique would be superior to existing measures.

3.2.2 Impact of Better Name Similarity

We further evaluated the impact of better name matching i.e., an element name similarity score/feature based on this new approach, on the schema matching accuracy of the overall GEM system. Table 6 shows the mapping accuracy achieved when employing GEM with the edit-distance based similarity score for the name similarity feature, and also employing GEM with the new component based name matching where the classifier confidence (a value in the range [0,1]) of the component based name matching is used as name similarity. As Table 6 illustrates, the impact of the new approach to name matching is significant for the GEM system, resulting in a 10-15% increase in overall mapping accuracy.

Table 6. Impact on GEM. *The impact of better name matching on overall schema-matching in GEM.*

Approach ➡ **Datasets Pairs** ⬇	**GEM** *(Edit-distance name similarity)*	**GEM** *(Composite name similarity)*
ADNI-NACC	83	91.6
ADNI-INDD	81	89.3
ADNI-LAADC	82.5	93.0
ADNI-INDD	81.9	91.2
ADNI-CLSA	80.8	93.3
Averaged (over 15 dataset pairs)	81.4	92.1

3.2.3 Evaluating and Configuring Classifiers

As mentioned earlier we applied multiple different classifiers including Support Vector Machines (SVM), Random Tree, Rotation Forest, Logistic Regression, Sequential Minimal Optimization (SMO) and the Multi-Layer Perceptron. Table 7 provides classifier details where Table 7(a) illustrates the performance of the six classifiers we evaluated with results shows for three specific dataset pairs and also averaged over all pairs. From these results we determine the SMO and Multi-Layer Perceptron classifiers as best applicable based on the matching accuracy.

We also configured the classifiers applied by tuning the key parameters for each. In Table 7 (b) we provide the optimal parameter settings for the top 3 classifiers. These optimal values were determined using informed heuristics about setting some of the parameter values [11,39] and further using grid-search [11] to search and determine optimal settings for this classification task.

Table 7 (a) Classifiers. *Accuracies in terms of F-score estimate evaluated over three dataset pairs for different classifiers for name matcher.*

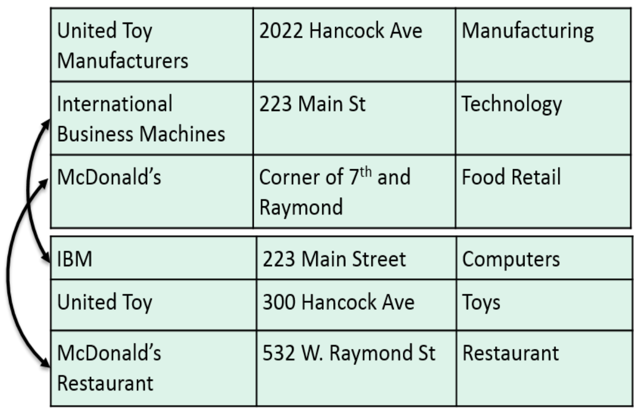
Dataset Pairs ➡ **Classifiers** ⬇	**ADNI-NACC**	**ADNI-INDD**	**ADNI-LAADC**	**Averaged**
Sequential Minimal Optimization (SMO)	81.6	73.2	88.6	82.9
Multi-Layer Perceptron	72.9	80	81.9	80.4
Simple Logistic Regression	77.9	75.4	78.9	76.6
Random Tree	72.9	66.7	78.9	72.3
Random Forest	72.3	66	69.7	68.1
Support Vector Machines (SVM)	75.3	64.2	41.9	58.3

Table 7 (b). Classifier Parameters. *The optimal set of parameters used for classification for the three best classifiers.*

Classifier	**Optimal Parameters**
SMO	RBF Kernel; C=1.0; έ=1.0E-12; numFold=-1
Multi-Layer Perceptron	hiddenLayers=a ; learning rate=0.3 ; momentum=0.2
Simple Logistic Regression	heuristic stop=50; maximum boosting iterations=500

4. RELATED WORK

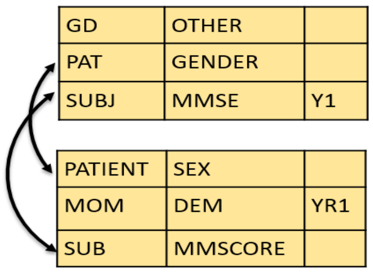
Many techniques have been developed for the problems of approximate string matching [8,30], finding patterns or specified strings in larger strings/sequences and especially in bioinformatics [31]. However, the problem of effectively matching composite name strings has not been addressed to the best of our knowledge. Certain aspects of the database *record-linkage* problem are similar to the work presented here. Record-linkage is the process of identifying different database records that correspond to the same entity in reality [46], and is motivated by applications in data integration [16] and data de-duplication [14]. Figure 3 illustrates the analog between record-linkage and composite name matching. Record-linkage algorithms match records in a *field by field* fashion, for instance in the example in Figure 3 we would match the name, address and industry sector fields individually (pairwise) and then combine the match across the different fields to determine the matching of records as a whole. The relationship to composite name matching stems from the fact that while the element names themselves are (typically) single words, they consist of multiple components that must be compared pairwise. This is illustrated in Figure 3(b) where we should consider an element name as composed of multiple components much in the same way a database record is composed of fields.



United Toy Manufacturers	2022 Hancock Ave	Manufacturing
International Business Machines	223 Main St	Technology
McDonald's	Corner of 7th and Raymond	Food Retail

IBM	223 Main Street	Computers
United Toy	300 Hancock Ave	Toys
McDonald's Restaurant	532 W. Raymond St	Restaurant

(a) Record-linkage in Databases



(b) Matching Names

Figure 3. Entity Linkage. *The analogy of record-linkage with name matching.*

The record-linkage techniques relevant to our name-matching problem are the ones based on using probabilistic linkage [22,27]. The overall probabilistic linkage approach employs machine-learning based classification where, for a given pair of database records, we identify features based on how corresponding fields from the two records relate to each other and train a classifier on that. Analogously, we have proposed matching composite element names by examining how their individual corresponding components relate to each other. The additional challenge however is that unlike database records which are already segmented into distinct fields, the composite names are not explicitly demarcated into separate components. In our approach we address determining such components using the dictionary and largest common substring methods.

The probabilistic inference of name matching based on multiple components is based on the probabilistic record-linkage theory developed in [15]. This paper formulates entity matching as a classification problem, where the basic goal is to classify entity pairs as matching or non-matching. There are also many record-linkage software tools available including LinkageWiz [38], LinkKing [7], LinkPlus [40], TAILOR [13] and FRIL [22] and also country specific record-linkage tools [10]. However name matching is, at best, a special case of record-linkage i.e., where a record has exactly one field and we have focused on the matching complexities within the one field.

The SimConcept system [45] addresses a different problem of resolving composite name references, for instance names such as "SMADSs1, 5 and 8" in biomedical text, which must be resolved to the explicit names SMAD1, SMAD5 and SMAD8. Their approach is based on Conditional-Random-Field (CRF) classifiers [23], where a name is divided into tokens and the CRF classifier employs features based on such tokens. The identification of tokens however needs to be powered by domain and data specific lexicons for various kinds of tokens. The techniques described in [32] use term familiarity and match synonymous terms based on lexical resources. The approach in [6] is another CRF based method with three kinds of tokens or "states" - conjunction, conjuncts, and ellipsis antecedent. The identification of the states is however domain specific, requiring lexicons for their identification. A common aspect is that we also require the resolution of element names into components. However for our purpose and data we are able to achieve the decomposition required for feature generation with minimal domain knowledge i.e., a synonym dictionary for the domain and a dictionary API.

There has been work on using a dictionary-based approach for protein name identification [42] and chemical name identification [20] in the bioinformatics field. [42] is focused on the problem of protein name recognition tackling false recognition caused by short names and low recall due to spelling variations. The approximate name match aspect is relevant to our work, however their problem is different from ours in that the name i.e., string does not involve multiple components. Work in [43] has explored the effectiveness of a fuzzy logic approach for matching patient records in databases. They have tried to solve issues arising with data entry errors, different text formats, and missing data which exact-match algorithms fail to detect. [43] also uses edit-distance as their basic metric over which they apply fuzzy logic. Researchers have also looked at modifying the Fellegi-Sunter scoring implementations to improve record-linkage in case of missing data [32]. [12] extends the Fellegi-Sunter method to improve linkage results when field values contain misspellings, alternate spellings, and typographical errors. It introduces an approach using approximate field comparators in the calculation field weights. As mentioned earlier, parallels can be drawn from missing data in record-linkage to missing prefixes and suffixes in our problem of composite name matching. The work in [36] is aimed at reducing the training effort in record-linkage classification, by using active-learning methods.

5. CONCLUSIONS AND FUTURE WORK

We presented an approach and implemented system for mapping composite element names in biomedical data. We demonstrated that our approach results in significantly superior element mappings than can be achieved by existing approaches based on string matching algorithms that are sophisticated for string matching in itself but do not factor the multiple component aspect of composite names. The experimental results also validate the proposed component model for composite element names as features based on individual element name components and their relationship prove to be most effective for mapping. Our system is largely unsupervised and requires minimal knowledge specific to an application or particular dataset and thus can scale across different types of datasets. The features employed however are hand-crafted and as future work we will investigate the applicability of unsupervised feature learning or what is referred to as "deep-learning" [4] for this problem. We used an English dictionary for the second heuristic of Component Based Feature Generation. In the future we could also try either generating a domain-specific dictionary or using one if it is available for that domain. This could reduce computation overhead of a large dictionary and lead to accurate identification of domain specific terms.

6. ACKNOWLEDGMENTS

This research was supported by the GAAIN project which is funded by the Alzheimer's Association under grant number 003278-0001 and by the Laboratory of Neuro Imaging Resource (LONIR) NIH grant number P41EB015922. Additional research supported was provided by the National Institute of Biomedical Imaging and Bioengineering of the National Institutes of Health under award number U54EB020406.

7. REFERENCES

1 Ashish, Naveen, Bhatt, Priya, and Toga, Arthur W. 2015. Global Data Sharing in Alzheimer Disease Research. *Alzheimer disease and associated disorders*.

2 Ashish, Naveen, Dewan, Peehoo, Ambite, Jose-Luis, and Toga, Arthur W. 2015. GEM: The GAAIN Entity Mapper. In *Data Integration in the Life Sciences* , 13-27.

3 Beekly, Duane L., Ramos, Erin M., Lee, William W. et al. 2007. The National Alzheimer's Coordinating Center (NACC) database: the uniform data set. *Alzheimer Disease \& Associated Disorders*, 21, 3, 249-258.

4 Bengio, Yoshua, Courville, Aaron C., and Vincent, Pascal. 2012. Unsupervised feature learning and deep learning: A review and new perspectives. *CoRR, abs/1206.5538*, 1.

5 Breiman, Leo, Friedman, Jerome, Stone, Charles J., and Olshen, Richard A. 1984. *Classification and regression trees*. CRC press.

6 Buyko, Ekaterina, Tomanek, Katrin, and Hahn, Udo. 2007. Resolution of coordination ellipses in biological named entities using conditional random fields. In *PACLING 2007-Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics* , 163-171.

7 Campbell, Kevin M. 2005. Rule Your Data with The Link King\copyright(a SAS/AF\textregistered application for record linkage and unduplication). *SUGI 30*, 1-9.

8 Cohen, William, Ravikumar, Pradeep, and Fienberg, Stephen. 2003. A comparison of string metrics for matching names and records. In *Kdd workshop on data cleaning and object consolidation*, 73-78.

9 Cristianini, Nello and Shawe-Taylor, John. 2000. *An introduction to support vector machines and other kernel-based learning methods*. Cambridge university press.

10 Dal Maso, Luigino, Braga, Claudia, and Franceschi, Silvia. 2001. Methodology used for "software for automated linkage in Italy"(SALI). *Journal of biomedical informatics*, 34, 6, 387-395.

11 Duan, Kaibo, Keerthi, S. Sathiya, and Poo, Aun Neow. 2003. Evaluation of simple performance measures for tuning SVM hyperparameters. *Neurocomputing*, 51, 41-59.

12 DuVall, Scott L., Kerber, Richard A., and Thomas, Alun. 2010. Extending the Fellegi--Sunter probabilistic record linkage method for approximate field comparators. *Journal of biomedical informatics*, 43, 1, 24-30.

13 Elfeky, Mohamed G., Verykios, Vassilios S., and Elmagarmid, Ahmed K. 2002. TAILOR: A record linkage toolbox. In *Data Engineering, 2002. Proceedings. 18th International Conference on* , 17-28.

14 Elmagarmid, Ahmed K., Ipeirotis, Panagiotis G., and Verykios, Vassilios S. 2007. Duplicate record detection: A survey. *Knowledge and Data Engineering, IEEE Transactions on*, 19, 1, 1-16.

15 Fellegi, Ivan P. and Sunter, Alan B. 1969. A theory for record linkage. *Journal of the American Statistical Association*, 64, 328, 1183-1210.

16 Halevy, Alon, Rajaraman, Anand, and Ordille, Joann. 2006. Data integration: the teenage years. In *Proceedings of the 32nd international conference on Very large data bases* , 9-16.

17 Hall, Mark, Frank, Eibe, Holmes, Geoffrey, Pfahringer, Bernhard, Reutemann, Peter, and Witten, Ian H. 2009. The WEKA data mining software: an update. *ACM SIGKDD explorations newsletter*, 11, 1, 10-18.

18 Han, Jiawei, Cheng, Hong, Xin, Dong, and Yan, Xifeng. 2007. Frequent pattern mining: current status and future directions. *Data Mining and Knowledge Discovery*, 15, 1, 55-86.

19 Haykin, Simon and Network, Neural. 2004. A comprehensive foundation. *Neural Networks*, 2, 2004.

20 Hettne, Kristina M., Stierum, Rob H., Schuemie, Martijn J. et al. 2009. A dictionary to identify small molecules and drugs in free text. *Bioinformatics*, 25, 22, 2983-2991.

21 Jaro, Matthew A. 1989. Advances in record-linkage methodology as applied to matching the 1985 census of Tampa, Florida. *Journal of the American Statistical Association*, 84, 406, 414-420.

22 Jurczyk, Pawel, Lu, James J., Xiong, Li, Cragan, Janet D., and Correa, Adolfo. 2008. FRIL: A tool for comparative record linkage. In *AMIA* .

23 Lafferty, John, McCallum, Andrew, and Pereira, Fernando C. N. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data.

24 Larsen, M. D. 1999. Predicting the Residency Status for Administrative Records that Do Not Match Census Records. *Administrative Records Research Memorandum Series*, 20.

25 Mamun, Abdullah-Al, Mi, Tian, Aseltine, Robert, and Rajasekaran, Sanguthevar. 2014. Efficient sequential and parallel algorithms for record linkage. *Journal of the American Medical Informatics Association*, 21, 2, 252-262.

26 *Merriam-Webster Dictionary API.*

27 Minton, Steven N., Nanjo, Claude, Knoblock, Craig A., Michalowski, Martin, and Michelson, Matthew. 2005. A heterogeneous field matching method for record linkage. In *Data Mining, Fifth IEEE International Conference on* , 8--pp.

28 Morris, John C., Aisen, Paul S., Bateman, Randall J. et al. 2012. Developing an international network for Alzheimer's research: the Dominantly Inherited Alzheimer Network. *Clinical investigation*, 2, 10, 975-984.

29 Mueller, Susanne G., et al. 2008. Alzheimer's Disease Neuroimaging Initiative. *Advances in Alzheimer's and Parkinson's Disease, Springer US.*

30 Navarro, Gonzalo. 2001. A guided tour to approximate string matching. *ACM computing surveys (CSUR)*, 33, 1, 31-88.

31 Navarro, Gonzalo and Raffinot, Mathieu. 2002. *Flexible pattern matching in strings: practical on-line search algorithms for texts and biological sequences*. Cambridge University Press.

32 Ong, Toan C., Mannino, Michael V., Schilling, Lisa M., and Kahn, Michael G. 2014. Improving record linkage performance in the presence of missing linkage data. *Journal of biomedical informatics*, 52, 43-54.

33 Peng, Yifan, Tudor, Catalina O., Torii, Manabu, Wu, Cathy H., and Vijay-Shanker, K. 2012. iSimp: A sentence simplification system for biomedicail text. In *Bioinformatics and Biomedicine (BIBM), 2012 IEEE International Conference on* , 1-6.

34 Platt, John and others. 1998. Sequential minimal optimization: A fast algorithm for training support vector machines.

35 Raina, Parminder S., Wolfson, Christina, Kirkland, Susan A. et al. 2009. The Canadian longitudinal study on aging (CLSA). *Canadian Journal on Aging/La Revue canadienne du vieillissement*, 28, 03, 221-229.

36 Ristad, Eric Sven and Yianilos, Peter N. 1998. Learning string-edit distance. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 20, 5, 522-532.

37 Sariyar, Murat, Borg, Andreas, and Pommerening, Klaus. 2012. Active learning strategies for the deduplication of electronic patient data using classification trees. *Journal of biomedical informatics*, 45, 5, 893-900.

38 Software, LinkageWiz. *LinkageWiz Software*.

39 Statnikov, Alexander, Wang, Lily, and Aliferis, Constantin F. 2008. A comprehensive comparison of random forests and support vector machines for microarray-based cancer classification. *BMC bioinformatics*, 9, 1, 1.

40 Thoburn, K. K., Gu, D., and Rawson, T. 2007. Link Plus: Probabilistic record linkage software. In *2nd Probabilistic Record Linkage Conference Call* .

41 Toga, Arthur W., Neu, Scott, Crawford, Karen, Bhatt, Priya, and Ashish, Naveen. 2015. The global Alzheimer's association interactive network (GAAIN). *Alzheimer's \& Dementia: The Journal of the Alzheimer's Association*, 11, 7, P121.

42 Tsuruoka, Yoshimasa and Tsujii, Jun'ichi. 2004. Improving the performance of dictionary-based approaches in protein name recognition. *Journal of biomedical informatics*, 37, 6, 461-470.

43 Wang, Xiaoyi and Ling, Jiying. 2012. Multiple valued logic approach for matching patient records in multiple databases. *Journal of biomedical informatics*, 45, 2, 224-230.

44 Weber, Susan C., Lowe, Henry, Das, Amar, and Ferris, Todd. 2012. A simple heuristic for blindfolded record linkage. *Journal of the American Medical Informatics Association*, 19, e1, e157--e161.

45 Wei, Chih-Hsuan, Leaman, Robert, and Lu, Zhiyong. 2014. SimConcept: a hybrid approach for simplifying composite named entities in biomedicine. In *Proceedings of the 5th ACM Conference on Bioinformatics, Computational Biology, and Health Informatics* , 138-146.

46 Winkler, William E. 2006. Overview of record linkage and current research directions. In *Bureau of the Census*

47 Wu, Xia, Li, Juan, Ayutyanont, Napatkamon et al. 2013. The receiver operational characteristic for binary classification with multiple indices and its application to the neuroimaging study of Alzheimer's disease. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, 10, 1, 173-180.

48 Xie, Sharon X., Baek, Young, Grossman, Murray et al. 2011. Building an integrated neurodegenerative disease database at an academic health center. *Alzheimer's \& Dementia*, 7, 4, e84--e93.

49 Zaki, Mohammed J. 2001. SPADE: An efficient algorithm for mining frequent sequences. *Machine learning*, 42, 1-2, 31-60.