Similarity Predicate Committee and its Application to Entity Resolution

September 30, 2017

Abstract

The goal of entity resolution is to determine whether two entities are the same (e.g. whether George Smith and Smith George refers to the same person). A fundamental issue is to derive an appropriate similarity predicate (a.k.a similarity function). In the last few decades, many similarity predicates are exploited, such as Jaccard coefficient, cosine similarity, BM25, language model, etc. While we agreed that all existing similarity predicates are well defined and effective, we observed that none of them can consistently outperform the others in all situations. Choosing which similarity predicate to use is usually being treated as an empirical question by evaluating a particular entity resolution task using a number of different similarity predicates. This is not efficient and the result obtained is not portable. As a result, we are curious about whether we can combine all similarity predicates together to form a committee so that we do not need to worry about choosing which of them to use, meanwhile we can obtain an even better result. In this paper, we focus on this issue and propose a generic framework to formulate a similarity predicate committee. To the best of our knowledge, this is the first piece of work about similarity predicate committee. However, formulating an effective similarity predicate committee is not trivial. We need to model at least two elements, namely, confidences of similarity predicates and reliabilities of attributes. In this paper, we explain how they affect the effectiveness of similarity predicate committee and how we model them on the fly dynamically. We apply our proposed work to solve some entity resolution problems. We report all our findings in this paper. It is worth to note that the proposed framework is unsupervised.

1 Introduction

Entity resolution is also known as record linkage [2], object identification or name disambiguation [21], data cleaning [8], approximate joins [12] and fuzzy matching [1]. A typical entity resolution problem is to identify records that belong to the same entity (e.g. whether George Smith and George Smath refers to the same person) [23]. Entity resolution is especially important in this Internet age because numerous online applications require entity resolution, such as

name disambiguation in digital libraries, searching a person in a social network (such as facebook), etc. Many different techniques can be found in the existing work to tackle this problem (e.g. [23, 13, 16, 22, 20, 11, 3]). Regardless of which techniques we use, a fundamental issue is to define an appropriate similarity predicate (a.k.a. similarity function) to measure the similarities among records.

Since [14] introduced the entity resolution problem into our community, many similarity predicates were exploited (e.g. [20, 11, 3, 5]). While we agreed that most, if not all, of the existing similarity predicates (e.g. Jaccard coefficient, cosine similarity, BM25, language model, Hidden Markov Model, etc) are well defined and effective, we observed that none of them can consistently outperform the others in all situations. Each of them has their own strengths and weaknesses. Yet, we do not have any systematic approach to help us to choose which similarity predicate to use in practice. Choosing which of them to use is usually being treated as an empirical question by evaluating a particular entity resolution task using a number of different similarity predicates or simply by the so-call "rule of trump method". This is not very efficient and the result obtained is usually not portable.

In this paper, we are not trying to propose any new similarity predicate as we do not lack of it; instead, we are trying to propose a generic framework that can combine a set of similarity predicates to form a committee. By doing so, we do not need to worry about choosing which of them to use. In addition, we believe that we should be able to achieve an even better entity resolution result by means of similarity predicate committee. The reason is that, if a decision requires expert's judgement, then the decision made by N different experts is usually better than any one of them if their decisions are combined properly. In a similarity predicate committee, each similarity predicate can be regarded as an expert. Our task is to determine whether a set of records refers to the same entity. To the best of our knowledge, this is the first piece of work about similarity predicate committee. We try to apply it to solve the entity resolution problem. Choosing entity resolution problem as our application domain is that it is one of the most important problems in our community (especially in this information overwhelming century). It is certainly possible to implement our framework in some other domains, such as information retrieval, classification, nearest neighbor search, etc.

In this paper, we will show that in order to formulate an effective similarity predicate committee, we need to consider at least two elements, namely, confidences of similarity predicates and reliabilities of attributes; otherwise, the result obtained by a similarity predicate committee may not be as good as any of the single similarity predicates. However, modelling these two elements is not trivial which poses some new challenges:

Challenge 1 (Confidences of Similarity Predicates). One of the simplest ways to formulate a similarity predicate committee is to aggregate the results of all similarity predicates in the committee. For example, assume we have three similarity predicates: Jaccard coefficient [20], cosine similarity [5] and edit distance [11]. Given a query record q and a set of records R, assume that we have calcu-

ID	Predicate	Similarity	Overall
r_1	Jaccard coeff.	0.85	0.55
	Cosine sim.	0.45	
	Edit distance	0.35	
r_2	Jaccard coeff.	0.70	0.57
	Cosine sim.	0.50	
	Edit distance	0.50	
...	...	...	...

Table 1: Similarities.

lated the similarities between q and each record in R by using the just mentioned three similarity predicates, which are shown in Table 1. In Table 1, the values in the column "Overall" are computed by averaging their corresponding three similarity values.[1] According to Table 1, we say that q is more similar to r_2 than r_1 because the overall similarity between q and r_2 is higher.

Despite of this approach is intuitive, it may not always be able to solve the entity resolution problem effectively. For example, if we have some prior knowledge that the two similarity values computed by Jaccard coefficient is far more reliable than the others, then we may want to alter our aforementioned decision and conclude that q is more similar to r_1 than r_2. A different conclusion is obtained. In fact, associating a confidence value to every member's decision in a committee is a very important topic. For instance, a research in classifier committee [10] shows that associating a confidence value to each classifier in the committee can significantly improve the effectiveness of a classifier committee. Moreover, [3] also shows that different similarity predicate has different effectiveness in handling different type of error. Hence, assigning a confidence value to each similarity predicate is necessary.

Unfortunately, even we theoretically know which similarity predicate is more effective in handling which type of error, we can never anticipate the types of errors may occur and how many errors would exist in a record. Assigning confidences to the similarity predicates becomes very difficult. For example, in Table 1, it is difficult to tell which similarity predicate is more reliable by simply looking at those numbers. In this paper, we try to solve this problem by using a consensus world model [18].

Challenge 2 (Reliabilities of Attributes). From our experience, it is quite easy to make mistakes on a person's email address, name or affiliation but is not very likely to make mistakes on a person's sex. Suppose there is a relation having three attributes: Name, Email Address and Sex. Given a query record q, assume there are two records, r_M and r_F, where the attribute Sex is male in r_M and is female in r_F. Suppose r_M and r_F are equally similar to q (i.e.

[1] In this example, we use average to aggregate the decisions, which is the most common technique. We can use other functions, such as maximum. Using which function does not affect the discussion in this section.

both have the same similarity value) and the confidences of computing them are the same (assume we know how to compute the confidences). Now, if the attribute Sex in q is female, then we properly believe r_F is in fact more similar to q because we seldom mistaken about one's sex. In this example, we say that sex is an attribute that is more reliable than the others. It is worth noting that whether or not an attribute is reliable is not related to the distribution of values. Reliability depends on whether an attribute can effectively resolve which of the records a query may concerning with.

One may point out that identifying the reliabilities of attributes can easily be achieved by applying expert knowledge – we manually examine which of the attributes are more reliable. However, expert knowledge is not domain portable and is expensive to obtain. Another possible approach is to follow a supervise learning framework – obtain a set of training data and a set of testing query records to evaluate the effectiveness of each attribute against a given similarity predicate. However, training data and testing query records are very expensive to obtain usually. Hence, we are curious about whether we can identify the reliabilities of attributes automatically without any expert knowledge or training data. We try to model this issue by using a probabilistic model – identify the probability of expected rank of a given record.

Challenge 3 (Generating Similarity Predicate Committee). Assume we have obtained the aforementioned two elements, the final question is how we can incorporate these two pieces of information in a similarity predicate committee to help us tackle the entity resolution problem more effectively. In this paper, we show that this problem can be modeled as a classical 0-1 integer programming optimization problem which tries to maximize the confidences of the ranking of records. Unfortunately, the search space of such integer programming problem will be too large, which is impossible to be solved by using any normal modern computer. Furthermore, this formulation will rank all records R in a database based on the similarities between R a query q, but in most situations we are always interested in extracting the top K records which are the most similar to q where K is much less than the total number of records. Accordingly, we propose a greedy-like algorithm to solve this problem in a more efficient way.

In this paper, our contributions are as follows: (1) We focus ourselves on solving the entity resolution problem by formulating a similarity predicate committee that leverages the above three issues. (2) We propose a generic similarity predicate committee framework which can incorporate any kinds of similarity predicates. (3) We implement and evaluate our proposed ideas by using datasets with different kinds and degrees of errors. The detailed experiment results could be served as an important resources for the future research in this topic or related areas. The motivations and contributions of this paper are both solid. The rest of the paper is organized as follows – Section 2 formally defines our problem; Section 3 briefly discusses some related work; Section 4 presents our proposed work; Section 5 conducts evaluation; Section 6 concludes this paper.

Notation	Meaning
R	a relation having $\|R\|$ records
A	a set of attributes in R
$r.a$	the value of attribute a in record r
$T_{r.a}$	a set of tokens in $r.a$
F	a set of similarity predicates.
f	a similarity predicate. $f \in F$
$sim_f(q.a, r.a)$	the similarity between $q.a$ and $r.a$ by using f
$sim_f(q, r)$	the similarity between q and r by using f
$\beta(a)$	the reliability of attribute a
$\alpha_f(r, k)$	the confidence of r is at rank k according to f

Table 2: Major Notation.

2 Problem Definition

Table 2 shows the notations used in this paper. Let R be a relation which contains $|R|$ records. Let A be a set of attributes in R. Let r be a record in R and a be an attribute in A. We use $r.a$ to denote the value of attribute a in record r. Let q be a query record and $q.a$ be the value of attribute a in q. In this paper, we assume all attributes contain string values.

Let $T_{r.a}$ and $T_{q.a}$ be two sets of tokens in $r.a$ and $q.a$, respectively. Unless otherwise specified, we refer to substrings of a string as tokens in a generic sense such that they can be words, q-grams [11], v-grams [17], etc. For example, suppose $r.a = \texttt{George}$, then $T_{r.a} = \{\texttt{Geo}, \texttt{eor}, \texttt{org}, \texttt{rge}\}$ for $q = 3$ in q-grams.

Given a query record q, our goal is to resolve the records in R that are relevant to q. There are lots of methods in the existing literatures to solve this problem. In general, all methods require us to compute the similarities between q and each $r \in R$ by using some similarity predicates. After that, we can determine which of the records should be returned based on their similarities with q. Let $sim_f(q, r)$ be the similarity value between q and r by using a similarity predicate f. For example, f can be Jaccard coefficient predicate [20], edit distance predicate [11], language model predicate [3], etc. Let F be a set of all similarity predicates. We will have a brief review of the existing similarity predicates in Section 3. Let $sim_f(q.a, r.a)$ be the similarity value between $q.a$ and $r.a$ that is computed by f. Since each record may have m attributes, in the existing work, one of the most common ways to obtain the overall similarity between q and r $sim_f(q, r)$ is to average $sim_f(q.a, r.a), \forall a \in A$, i.e. $sim_f(q, r) = \frac{1}{|A|} \sum_{\forall a \in A} sim_f(q.a, r.a)$. Yet, in this paper, we will take a different approach in modeling this element. In this paper, we are trying to solve the following problem:

Problem Statement 1. Given a relation R, a query record q and a set of similarity predicates F, our goal is to rank the records in R according to q by properly

combining the decisions of the similarity predicates in F.

3 Related Work

To the best of our knowledge, this is the first piece of study about similarity predicate committee and its application to entity resolution. Entity resolution has been studied extensively in the last few decades. [7] and [9] are two classical studies that use statistical techniques to solve the entity resolution problem. [14] was the first one to introduce entity resolution problem to our computer science discipline. A variety of similarity predicates were exploited thereafter. In general, we can classify the existing similarity predicates into one of the following five areas [3]: overlap predicates (e.g. intersection [20], Jaccard Coefficient [20]), aggregate weighted predicates (e.g. cosine similarity [5, 20], BM25 [3]), language modeling predicates (e.g. [3]), edit distance based predicates (e.g. [11, 20]) and combination predicates (e.g. generalized edit similarity [4], softTfIdf [6]). Table 3 lists the most common and popular similarity predicates. Due to the space limit, we will not discuss them one by one.

A necessary preprocessing step in entity resolution is to define the meaning of word in a string. Q-grams [11] is therefore proposed. [17] proposed a concept called v-gram to improve the efficiency of q-gram. In this paper, our proposed framework is a generic one, which is independent of the implementation details of similarity predicates or preprocessing processes.

Although there are many well-documented similarity predicates in the existing literatures, none of them can outperform the others in all situations. How to select an appropriate similarity predicate to use in practice is still an open question. In this paper, we propose a framework to combine a given set of similarity predicates to form a committee. In terms of forming a committee to make decisions, there are lots of studies in the data mining community. Most come from the area of classification. There are two types of classifier committees, homogeneous classifier committee and heterogeneous classifier committee. A homogeneous classifier committee contains N classifiers come from the same learning algorithm [24]. From a high level point of view, the idea of formulating a similarity predicate committee is parallel to formulate a heterogeneous classifier committee – both are trying to make decisions by pooling individual decisions and each individual comes from a different predicate/algorithm. Yet, the technical details of classifier committee and similarity predicate committee are totally different as we will see this in the next section. For instance, in all classification problems, we are given a set of training data for training the classifiers; for our problem, we neither have any training data nor request any user contribution. Our framework is an unsupervised learning framework. We cannot apply the existing techniques directly to solve our problem.

Predicate	Equation
Intersect	$sim_{\text{intersect}}(q.a, r.a) = \lvert T_{q.a} \cap T_{r.a} \rvert$ (1)
Jaccard	$sim_{\text{Jaccard}}(q.a, r.a) = \frac{T_{q.a} \cap T_{r.a}}{T_{q.a} \cup T_{r.a}}$ (2)
Cosine[a,b]	$sim_{\text{cosine}}(q.a, r.a) = \sum_{t \in T_{q.a} \cap T_{r.a}} w(t, q.a) \cdot w(t, r.a)$ (3) $w(t,s) = \frac{w'(t,s)}{\sqrt{\sum_{t' \in s} w'(t',s)^2}} \qquad w'(t, r.a) = tf(t, r.a) \cdot idf(t, a)$ (4)
BM25[c]	$sim_{\text{BM25}}(q.a, r.a) = \sum_{t \in T_{q.a} \cap T_{r.a}} w_q(t, q.a) \cdot w_r(t, r.a)$ (5) $w_q(t,s) = \frac{(k_3+1) \times tf(t,s)}{k_3 + tf(t,s)} \qquad w_r(t,s) = w'(t,R) \frac{(k_1+1) \times tf(t,s)}{k(s,R) + tf(t,s)} \qquad w'(t,R) = \log\left(\frac{\lvert R \rvert - n_t + 0.5}{n_t + 0.5}\right)$ (6) $k(r.a, R) = k_1 \left[(1-b) + b \frac{\lvert T_{r.a} \rvert}{\sum_{\forall r' \in R} \lvert r'.a \rvert / \lvert R \rvert} \right]$ (7)
Language Model[a,d]	$sim_{\text{LM}}(q.a, r.a) = \prod_{t \in T_{q.a}} \hat{p}(t \lvert M_{ik}) \times \prod_{t \notin T_{q.a}} (1 - \hat{p}(t \lvert M_{ik}))$ (8) $\hat{p}(t \lvert M_{ik}) = \begin{cases} \hat{p}(t, r.a)^{1-\hat{R}(t,r.a)} \hat{p}_a(t)^{\hat{R}(t,r.a)} & \text{if } tf(t, r.a) > 0 \\ \sum_{\forall r' \in R} tf(t, r'.a) / \sum_{\forall r' \in R} \lvert r'.a \rvert & \text{otherwise} \end{cases}$ (9) $\hat{p}(t, r.a) = \frac{tf(t, r.a)}{\lvert t \rvert} \qquad \hat{p}_a(t) = \frac{\sum \hat{p}(t \lvert M_{ik})}{df(t,k)} \qquad \hat{R}(t, r.a) = \left(\frac{1}{1+\bar{f}_{ik}(t)}\right) \left(\frac{\bar{f}_{ik}(t)}{1+\bar{f}_{ik}(t)}\right)^{tf(t,r.a)}$ (10) $\bar{f}_{ik}(t) = p_a(t) \times \lvert T_{r.a} \rvert$ (11)
HMM[a,e]	$sim_{\text{HMM}}(q.a, r.a) = \prod_{t \in T_{q.a}} \left(a_0 P(t \lvert G) + a_1 P(t \lvert r.a)\right)$ (12) $P(t \lvert r.a) = \frac{tf(t, r.a)}{\lvert T_{r.a} \rvert} \qquad P(t \lvert G) = \frac{\sum_{\forall r' \in R} tf(t, r'.a)}{\sum_{\forall r' \in R} \lvert r'.a \rvert}$ (13)
Edit Distance[f]	$sim_{\text{edit}}(q.a, r.a) = 1 - \frac{tc(q.a, r.a)}{\max\{\lvert T_{q.a} \rvert, \lvert T_{r.a} \rvert\}}$ (14)
GES[f,g]	$sim_{\text{GES}}(q.a, r.a) = 1 - \min\left\{ \frac{tc(q.a, r.a)}{wt(q.a)}, 1.0 \right\}$ (15)
SoftTfIdf[h]	$sim_{\text{soft}}(q.a, r.a) = \sum_{t \in T} w(t, q.a) \times \max_{t' \in T_{r.a}} (s(t,t')) \times w(\max_{t' \in T_{r.a}} (s(t,t')), r.a)$ (16)

[a] $tf(t,s)$ is the number of times t appears in s.
[b] $idf(t,a)$ is the inverse number of $r.a$ contains t, $\forall r \in R$.
[c] $k_1 \in [1,2], k_3 = 8, b \in [0.60, 0.75]$ as suggested by [3]
[d] $df(t,a)$ is the number of $r.a$ contains t, $\forall r \in R$.
[e] a_0 and a_1 are transition probabilities of the HMM. $a_1 = 1 - a_0$.
[f] $tc(q.a, r.a)$ is the min cost of edit operations to converts $q.a$ to $r.a$.
[g] $wt(q.a)$ is the sum of weights of all word tokens in $q.a$.
[h] $s(t,t')$ is the similarity between t and t', $w(\cdot,\cdot)$ is the normalized tf-idf weight, $T \subseteq T_{q.a}$ s.t. $\forall t \in T, \exists t' \in s_{r.a}, s(t,t') > \theta$.

Table 3: Similarity Predicates.

4 Proposed Work

In order to formulate an effective similarity predicate committee, we need to model at least two elements properly: reliabilities of attributes and confidences of similarity predicates. In the following sections, we will discuss how we model them without using any training data.

4.1 An Overview

Given a relation R, a query record q and a set of similarity predicates F, for each record $r \in R$, we can use a similarity predicate $f \in F$ to measure the similarity between $r.a$ and $q.a$ for each $a \in A$. As a result, for each r, we can obtain a $|F| \times |A|$ similarity matrix. Each cell in this matrix denotes $sim_f(q.a, r.a), \exists f \in F, \exists a \in A$. Since we have $|R|$ records, we will have $|R|$ number of $|F| \times |A|$ similarity matrices. We try to identify the reliabilities of attributes based on these similarity matrices and the consensus world model [18].

The general idea here is that: given an attribute a and a record r, if $sim_f(q.a, r.a)$ for all $f \in F$ are similar, then it implies that the consensus among similarity predicates on attribute a of record r is high. In other words, we may say that attribute a of record r is reliable because its performance is consistent among different similarity predicates. If we can identify an attribute a such that it performs reliable among most records in R, then we can properly say that a is a reliable attribute. Details of this process will be discussed in Section 4.2.

After we have obtained the reliabilities of all attributes, we try to identify the confidences of different similarity predicates. We observed that our ultimate objective is not trying to compute the similarity between q and r, but is trying to rank r according to $sim_f(q, r), \forall f \in F$. In addition, for each $f \in F$, we can obtain the ranking of all $r \in R$ based on $sim_f(q, r)$. Accordingly, for each $r \in R$ and each $f \in F$, we try to model the confidence of f in ranking r by the probability of r appears in a given position in the final rank based on $sim_f(q, r)$. Details of this process will be discussed in Section 4.3.

Finally, we try to rank the records by combining the decisions of all member similarity predicates in the committee based on their confidences in ranking the records. We will show that we can formulate this problem as a 0-1 integer programming problem, but the search space is too large to solve in reality. So we provide another greedy like algorithm to tackle it. Details of this process will be discussed in Section 4.4.

4.2 Reliabilities of Attributes

Given an attribute $a \in A$ and a record $r \in R$, if the similarity $sim_f(q.a, r.a)$ for all $f \in F$ are very similar, then it should imply the consensus among similarity predicates on $a.r$ is very high. In other words, we can say that attribute a of record r is reliable because its performance is consistent among most similarity predicates so that most similarity predicates do not have any serious disagreement with each other. As a result, if we can identify an attribute a such that it

performs reliable among most records in R, then we can say that attribute a is a reliable attribute. Accordingly, we model the reliabilities of attributes based on the consensus of $sim_f(q.a, r.a)$ for all $f \in F$. We try to model the consensus among similarity predicates based on the variance of $sim_f(q.a, r.a)$ for all $f \in F$. Specifically, the variance of attribute a's similarity values, $var_r(a)$, is:

$$var_r(a) = \sum_{\forall f \in F} \left(sim_f(q.a, r.a) - \overline{sim}_r(a) \right)^2, \tag{17}$$

where $\overline{sim}_r(a)$ is the mean similarity between $q.a$ and $r.a$. It is computed by averaging $sim_f(q.a, r.a)$ for all $f \in F$. Let $\beta_r(a)$ be the normalized value of $var_r(a)$ such that it is within unit value:

$$\beta_r(a) = \frac{v_r(a)}{\sqrt{\sum_{\forall a' \in A} v(r.a')^2}}. \tag{18}$$

In the above formulation, we can make sure $\beta_r(a) \in [0..1]$ and $\sum_{\forall a \in A} \beta_r(a)^2 = 1$. Let $\beta(a)$ be the reliability of attribute a. It is worth mentioning that the value computed by the above two equations is a reference value only. The value itself does not have much meaning. It is useful only if we compare it with all other values in other records. According to our previous discussion, we try to model the reliability of attribute a by reviewing whether a is reliable in most records. So we compute $\beta(a)$ based on the idea of root mean square error [19]. We choose root mean square error because of its popularity and it is also one of the most commonly used techniques to measure the discrepancy between an object's value and the mean value. Mathematically:

$$\beta(a) = 1 - \sqrt{\left(\beta_r(a) - \bar{\beta}_a \right)^2}, \tag{19}$$

$$\bar{\beta}_a = \frac{1}{|R|} \sum_{\forall r \in R} \beta_r(a). \tag{20}$$

In Eq.(19), $\bar{\beta}_a$ is the mean of $\bar{\beta}_a(r), \forall a \in A$. From this equation, the higher the value of $\beta(a)$, the more reliable attribute a will be.

4.3 Confidences of Similarity Predicates

In this section, we present how we obtain the confidences of similarity predicates. Based on the reliability of attribute a, $\beta(a)$, we can compute the similarity value of each record r to the query record q by using weighted average [24]:

$$sim_f(r, q) = \frac{1}{|A|} \sum_{\forall a \in A} sim_f(r.a, q.a) \times \beta'(a), \tag{21}$$

where $\beta'(a)$ is the normalized value of $\beta(a)$. We need to normalize $\beta'(a)$ to unit length so as to make sure $sim_f(r, q)$ must be within 0 and 1. $\beta'(a)$ is computed as follows:

$$\beta'(a) = \frac{\beta(a)}{\sqrt{\sum_{\forall a' \in A} \beta(a')^2}}. \tag{22}$$

Rank	ID	$sim_f(q,r)$
1	r_2	0.95
2	r_1	0.93
3	r_4	0.91
4	r_3	0.05
5	r_6	0.03
6	r_5	0.01

A. The ranking of records in L_f.

Rank	ID	$\alpha_f(r,k)$					
		$k=1$	$k=2$	$k=3$	$k=4$	$k=5$	$k=6$
1	r_2	0.314	0.308	0.302	0.031	0.025	0.019
2	r_1	0.301	0.306	0.301	0.036	0.031	0.025
3	r_4	0.290	0.298	0.303	0.042	0.036	0.030
4	r_5	0.030	0.036	0.042	0.303	0.298	0.290
5	r_6	0.025	0.031	0.036	0.301	0.306	0.301
6	r_3	0.019	0.025	0.031	0.302	0.308	0.314

B. The confidence in ranking.

Figure 1: An example dataset.

Although the idea of weighted average in Eq. (21) is straightforward, it is robust because the similarity values outputted by different similarity predicates by using weighted average tend to be more similar with each other than the values outputted by using simple average[2]. The reason is that weighted average will assign higher weights to the attributes that have more consensus among different similarity predicates. As a result, we can reduce the discrepancy among different similarity predicates to a certain degree.

Given a particular similarity predicate $f \in F$, we can compute $sim_f(r,q)$ for all $r \in R$. Accordingly, we can rank the records in R based on the values of $sim_f(r,q), \forall r$. Let L_f be a list of records sorted by the descending order of $sim_f(r,q)$. In this paper, we use $L_f(i)$ to denote the record at rank i in L_f.

Now, our problem is to determine how confident a similarity predicate f would have when it ranks the records in L_f. Intuitively, if the similarity values in L_f are very similar, then we are more easy to make mistakes in ranking. Let us take Figure 1A as an example. It shows the ranking of six records ranked by a similarity predicate f. The similarity values among the first three records are very similar, and the similarity values among the last three records are also very similar. Yet, the similarity values between the first three records and last three records are very different. From the table, we can conclude that:

- We are quite sure that both r_1, r_2 and r_4 should be ranked higher than all other records as their similarity values are significantly higher than the others, but we are not very confident that their ranking are correct as their similarity values are very similar.
- Similarly, we are quite sure that both r_3, r_5 and r_6 should be ranked lower than all other records, but we are not very confident that which of them should be ranked lowest as their similarity values are very similar.

As a result, we claim that the confidences of similarity predicates in rankings should be highly related to the distribution of similarity values. Let $\alpha_f(r,k)$ be the confidence of similarity predicate f in the ranking of record r is k. We try to model $\alpha_f(r,k)$ by the probability of record r is ranked at position k in the final ranking given that its similarity value is $sim_f(r,q)$. Specifically, $\alpha_f(r,k)$ is

[2]Simple average: $\frac{1}{|A|}\sum_{\forall a \in A} sim_f(r.a, q.a)$

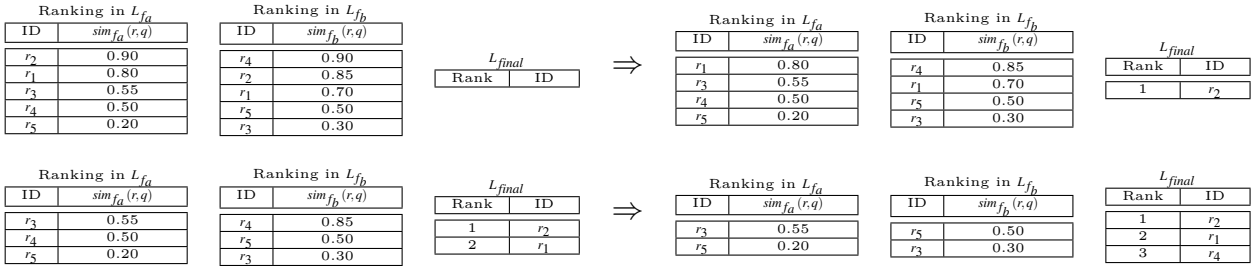

Figure 2: Results Returned by a Similarity Predicate Committee.

computed as follows:

$$\alpha_f(r,k) = \frac{1-d(r,k)}{\sum_{i=1}^{|L_f|} d(r,i)} \tag{23}$$

$$d(r,k) = |sim_f(r,q) - sim_f(L_f(k),q)| \tag{24}$$

where $L_f(k)$ is the record at position k in L_f and $d(r,k)$ is the different between the "the similarity between r and q" and "the similarity between the record at position k in L_f and q". $d(r,k)$ tries to compute the difference between $sim_f(r,q)$ and $sim_f(L_f(k),q)$ and uses it to model the required confidence value. The summation at the denominator is used to normalize the value such that $\sum_{\forall k} \alpha_f(r,k) = 1$. Having this constraint is necessary and reasonable because r can only be at one and only one position in the final ranking.

Figure 1B shows the result of $\alpha_f(r,k)$ for the records in Figure 1A. Each row contains six confidence values, each of which denotes the confidence in the ranking of a particular record is at position k. For example, for r_2, we have similar confidence values when r_2 is ranked at any of the top three positions because its similarity value is similar to those at the second and third positions. However, the confidence values of ranking r_2 at the bottom three positions are all very low because r_2 is very unlikely to be ranked at any of the bottom three positions. Note that the confidence values in boldface are the highest values in their own rows, which also belong to the records at their own ranks. This follows our intuition: given a record r, if it is ranked at position k, then $\alpha_f(r,k)$ should be larger than all $\alpha_f(r,k')$ where $k' \in \{1,2,\ldots|R|\}, k' \neq k$.

4.4 Committee Generation

In this section, we present how we combine the decisions of different similarity predicates so as to rank the records according to their similarities to the query record q. Intuitively, since for every record r, we can compute its confidence at rank k by similarity predicate f, so we can logically obtain the averaged confidence of r at rank k as follows:

$$\bar{\alpha}(r,k) = \frac{1}{|F|} \sum_{\forall f \in F} \alpha_f(r,k). \tag{25}$$

where $\bar{\alpha}(r,k)$ is the required averaged confidence. Accordingly, we can model this problem as a classical optimization problem which tries to maximize the total averaged confidence of ranking all records subject to each rank can have one and only one record. Specifically, our problem can be modeled as the following 0-1 integer programming problem:

$$\max_{A} \quad \sum_{m=1}^{|R|} \left(\frac{1}{|F|} \sum_{\forall f \in F} \alpha_f(r_m, k_m) \right) \tag{26}$$

$$\text{subject to:} \quad \mathrm{A} = [a_{mn}]_{|R|\times|R|}, a_{mn} \in \{0,1\} \tag{27}$$

$$\mathrm{B} = [b_{mn}]_{|R|\times 1}, b_{m1} = m \tag{28}$$

$$\sum_{n=1}^{|R|} a_{mn} \times b_{n1} = k_m \tag{29}$$

$$\sum_{n=1}^{|R|} a_{mn} = 1, \forall m \tag{30}$$

$$\sum_{m=1}^{|R|} a_{mn} = 1, \forall n \tag{31}$$

where A is a $|R| \times |R|$ matrix which denotes the ranking of records. For example, the m^{th} row in A denotes the ranking of r_m such that if r_m is ranked at position k, then $a_{mk} = 1$ and $a_{mk'} = 0, \forall k' \in \{1,2,\ldots|R|\}, k' \neq k$. Since a record can only exist in one position, Eq. (30) is necessary. In addition, each ranking can only have one record, so Eq. (31) is necessary.[3] B is a $|R| \times 1$ matrix which contains all possible rankings. Specifically, B= $[\ 1\ 2 \ldots |R|\ ]^T$. Hence, the ranking of r_m (i.e. k_m) can be obtained from Eq.(29).

Unfortunately, the search space of the above integer programming problem is too large for any normal modern computer to solve it. Specifically, the number of records is usually several hundred thousands to millions so that the matrix A will be too large to be fitted in the main memory. Furthermore, to solve this optimization problem directly, we may need to expand all of the possible ranking combination, which is in the order of $O(n!)$ (Due to the limited space, we obmit the proof). This is clearly computational infeasible. In addition, the above formulation will rank all records in the database, but in reality we are always interested in extracting the top K records which are the most similar to the query record q where $K \ll |R|$. We usually do not care about the ranking of other records. Accordingly, we propose a greedy-like algorithm to solve the problem.

Let L_{final} be a list containing records that are ranked according to the descending order of their similarities to q. Figure 2 shows a simple example with 5 records to illustrate the idea of how we use a greedy-like algorithm to rank the records. In this example, we have two similarity predicates: f_a and f_b. We rank all records based on their similarity values (Eq. (21)). The ranking of records in

[3] Eq. (31) can be removed if we allow a rank can have multiple records.

L_{f_a} is: $r_2 > r_1 > r_3 > r_4 > r_5$, and in L_{f_b} is: $r_4 > r_2 > r_1 > r_3 > r_5$. Initially, the list L_{final} is empty.

In the first iteration, we obtain the first record from every list for comparison, i.e. to determine which of them should be ranked at the first position. In this example, r_2 (which is ranked at the first position in L_{f_a}) and r_4 (which is ranked at the first position in L_{f_b}) are taken out for comparison.

In order to determine whether r_2 or r_4 should be ranked at the first position in L_{final}, we compute their averaged confidence at position 1 by using Eq. (25) and setting $k = 1$. In Figure 2, $\alpha_{f_a}(r_2, 1) = 0.294$ and $\alpha_{f_b}(r_2, 1) = 0.243$. So $\bar{\alpha}(r_2, 1) = (0.294 + 0.243)/2 = 0.269$. On the other hand, $\alpha_{f_a}(r_4, 1) = 0.152$ and $\alpha_{f_b}(r_4, 1) = 0.267$. So $\bar{\alpha}(r_3, 1) = (0.152 + 0.267)/2 = 0.210$. Since $\bar{\alpha}(r_2, 1) > \bar{\alpha}(r_4, 1)$, we rank r_2 at the first position in L_{final}. Note that we cannot put r_4 at the second position in L_{final} immediately because it is possible that r_1 (which is the second record in L_{f_a}) should be in fact at the second rank in L_{final} but not r_4. We do not know whether r_1 or r_4 should be ranked higher at this moment. Once r_2 is returned, r_2 will be removed from L_{f_a} and L_{f_b} because its ranking is already confirmed and the probability for any other record to be ranked at the first position in L_{final} is 0.

In the second iteration, note that since the sizes of L_{f_a} and L_{f_b} are changed, their confidences, which is computed by Eq. (23), will also be changed (the denominator of Eq. (23) will be changed once the size of L_{f_a} is changed). We have to stress that we need to alter the confidence values after each iteration. The reason is that, as discussed in the previous section, we model $\alpha_f(r,k)$ by the probability of the record r is ranked at position k in the final ranking given that its similarity value is $sim_f(r,q)$. If the positions of some records are confirmed (e.g. r_2 in the example), then the probability of the record r ranked at any of these positions should be 0 (e.g. $\alpha_{f_b}(r_4, 1) = 0$ after the first iteration). However, the total probability obviously has to be 1 (e.g. $\sum_{\forall k} \alpha_{f_b}(r_4, k) = 1$). Hence, $\alpha_f(r,k)$ should be changed after each iteration. After computing the new confidence values, $\alpha_{f_a}(r_1, 1) = 0.351$ and $\alpha_{f_b}(r_1, 1) = 0.262$. So $\bar{\alpha}(r_2, 1) = 0.307$. On the other hand, $\alpha_{f_a}(r_4, 1) = 0.209$ and $\alpha_{f_b}(r_4, 1) = 0.339$. So $\bar{\alpha}(r_3, 1) = 0.274$. Since $\bar{\alpha}(r_1, 1) > \bar{\alpha}(r_4, 1)$, we rank r_1 at the second position in L_{final}. The whole process continues until all records are added into L_{final} or we have ranked K number of records. In Figure 2, we only show the first three iterations due to limited space (or one may consider $K = 3$ in this example).

Algorithm 1 outlines the major steps for implementation. Due to the limited space, we cannot have any further elaboration on it, but this algorithm should be self-explained.

5 Experiments

All experiments are conducted using an Intel 2.5GHz CPU and 8GB RAM in Microsoft Windows Server 2003. All programs are written in Java. We compare our work with the predicates in Table 3. We set all parameters as suggested in [3]: for BM25, we set $k_1 = 1.5$, $k_3 = 8$, $b = 0.675$; for HMM, we set a_0 to 0.2;

Algorithm 1:

input : The ranking of records by predicate f, $L_f, \forall f \in F$
output: The ranking of records in R, L_{final}

$L_{final} \leftarrow \emptyset$; // a list;
for $i = 1$ to $|R|$ do
  $\bar{\alpha}(r,1) \leftarrow 0, \forall r \in R$;
  foreach $f \in F$ do
    if $\bar{\alpha}(r,1) = 0$ then
      $r = L_f(1)$; // the top most record in L_f
      foreach $f' \in F$ do
        $\bar{\alpha}(r,1) \leftarrow \bar{\alpha}(r,1) + \bar{\alpha}_{f'}(r,1)$;
      end
    end
  end
  $r^* \leftarrow$ the record with maximum $\bar{\alpha}(r,1), \forall r \in R$;
  append r^* at the end of L_{final};
  remove r^* from $L_f, \forall f \in F$;
  re-compute confidences for all $f \in F$ according Eq. (23);
end
return L_{final};

Erroneous	Name	% Duplicate	CE (%)	TE (%)	AE (%)
Low	D1	25	15	15	15
Medium	D2	50	30	30	30
High	D3	75	60	60	60
–	D4	50	60	0	0
–	D5	50	0	60	0
–	D6	50	0	0	60

Table 4: Settings of datasets.

for SoftTFIDF, two words are similar in if their similarity score exceeds 0.8; for q-gram, we set $q = 2$.

5.1 Dataset

Since we do not have any benchmark dataset, we define our own datasets with controlled error. We modify the UIS database generator [3, 15] to generate datasets for our experiments. UIS database generator accepts a set of clean tuples as the source and generates erroneous duplicates based on a set of user defined parameters. We generate our datasets based on the following steps:

Step 1. We archive data from DBLP[4]. Table 5 show its schema. It is served as clean dataset. There are 10,000 records. We conducted experiments with

[4] www.informatik.uni-trier.de/~ley/db/

ID	Attributes	words/tuple	char/tuple
1	Conference Name	6	52
2	Conference Place	2	19
3	Paper Title	5	48
4	Author Names	8	40
5	Intuitions	8	72

Table 5: Statistics Information of DBLP

datasets of increasing size and observed that the overall accuracy trend presented remains the same.

Step 2. We first select the fraction of records in which errors are injected, then inject errors to them. In each record selected for error injection, we will introduce these errors: (1) Character edit error (CEE), which is the percentage of characters that will be selected for injecting edit error (character insertion, deletion, replacement or swap); (2) Token swap error (TSE), which is the percentage of word pairs that will be swapped; (3) Abbreviation error (AE), which is the percentage of word that would be replaced by abbreviation (e.g. replacing "George Smith" to "G Smith"). Table 4 shows the details of the erroneous datasets. We classify the first three datasets into three level of erroneous: low, medium and high. Each of the last three datasets have only one specific type of error. Each dataset contains 100,000 tuples generated from 10,000 clean records with uniform distribution. We conducted experiments with different distributions and the similar results are obtained.

5.2 Evaluation Matrix

Following the most up-to-date evaluation methodology [3], we compute the Mean Average Precision (MAP) of the rankings of each dataset.[5] Average Precision(AP), is the average of the precision after each similar record is retrieved:

$$AP(q,R) = \frac{1}{N_q} \sum_{i=1}^{|D|} (pr(i|q) \times rel(i|q)) \tag{32}$$

where N_q is the number of records relevant to q, $pr(i|q)$ is the precision at rank i given q, which is the ratio of the number of relevant records having rank $\leq i$ to the number of records having rank $\leq i$, $rel(i|q)$ is 1 if the record at rank i is relevant to q and 0 otherwise. MAP is the mean AP value over all queries. For each dataset, we compute MAP over 2,000 randomly selected query record. Since a query record can be a clean record or an erroneous record, our results should represent the expected behavior of predicates.

[5]Another popular measurement is maximum F_1. Due to the limited space, we only report MAP but it is consistent with mean maximum F_1 in all experiments.

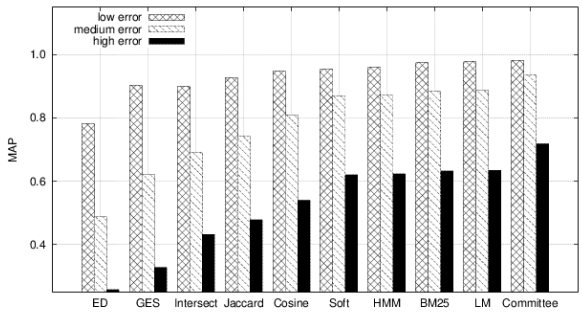

Figure 3: MAP on low (D1), medium (D2) and high (D3) erroneous datasets

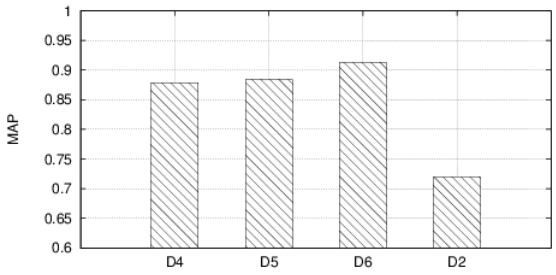

Figure 4: MAP on D2, D4, D5 and D6

5.3 Effectiveness Analysis

In this section we present a detailed comparison of the effectiveness of similarity predicates in capturing different types of errors. Figure 3 shows MAP for different predicates on the low, medium and high erroneous datasets. The first 9 groups of bars denote the evaluation results of 9 different similarity predicates (which are shown in Table 3). The last group of bars denotes the evaluation result of our proposed work – similarity predicate committee. At a first glance, our proposed work, outperforms all similarity predicates in all datasets. This finding is especially clear for the high erroneous dataset. For the low erroneous datasets, all predicates perform very well except edit distance and GES. In all situations, the performances of SoftTfIdf (Soft), HMM, BM25 and Language Model (LM) are very similar. They consistently outperform all other similarity predicates, except our proposed work.

Figure 4 shows the evaluation results of the proposed work on the datasets D2, D4, D5 and D6. Each of D4, D5 and D6 contains one and only one type of error, whereas D2 contains all three types of errors. D2 is presented here for a comparison only. In general, MAP for D4, D5 and D6 are all very similar. D6 is a little bit more accurate. The reason is that the error in D6 is abbreviation error, which can be detected by quite many different types of similarity predicates

very well. For the other two types of errors, their impacts are similar. In this experiment, we can see that having a similarity predicate committee can handle different types of errors quite effective. We do not need to worry about choosing which predicate to use.

5.4 Further discussion

One important question we have to answer is: how important is the factor of reliabilities of attributes and confidences of similarity predicates when we formulate a committee? In this experiment, we try to address this issue by conducting the following three experiments:

Experiment 1 (Simple). We do not consider reliabilities of attributes and confidences of similarity predicates. We simply average the similarity values of all members in a committee. This approach is as same as the approach demonstrated in Challenge 1 in Section 1.

Experiment 2 (No Confidence). We only consider reliabilities of attributes but ignore confidences of similarity predicates when we formulate a committee. We average the similarity values of all members in a committee and rank them.

Experiment 3 (No Reliability). We only consider confidences of similarity predicates but ignore reliabilities of attributes when we formulate a committee.

Figure 5 shows the evaluation results. In the figure, there are four groups of bars, each of which denotes the result against different types of dataset of a specific setting. From the figure, the MPAs of Experiment 1 (Simple) are all lower than the corresponding MAPs of Soft, BM25, HMM and LM (Figure 3). This demonstrate that simply combining all decisions may not necessary yield a better result. The reason is that some similarity predicates, such as edit distance and GES, perform poorly in a very noisy environment. They will bias the averaged value greatly. The MAPs of Experiment 3 (no reliability) are consistently higher than the corresponding MAPs of Experiment 2 for all three datasets. However, this cannot implies the confidences of similarity predicates have a higher impact on precision than the reliabilities of attributes. The reason is that for all the datasets, all attributes are more or less having the same reliability. Our datasets do not have any attribute having the characteristic similar to the situation of Challenge 2 in Section 1. As a result, we perform one more set of experiments: we reformat the dataset so that the data contained in the attributes Author Names and Conference Place are shuffled completely across all records, i.e. they do not give any useful identity information at all. Due to the limited space, we cannot show this figure. Yet, the different between "No Reliable" and "Committee" becomes larger (i.e. compare with Figure 5 the effectiveness of "No Reliable" deteriorated because 'No Reliable" ignores the reliabilities of attributes), whereas the different between "No Reliable" and "No Confidence" becomes less (i.e. compare with Figure 5 the effectiveness of "No Confidence" improved because some attributes are clearly not reliable and we consider this factor in "No Confidence").

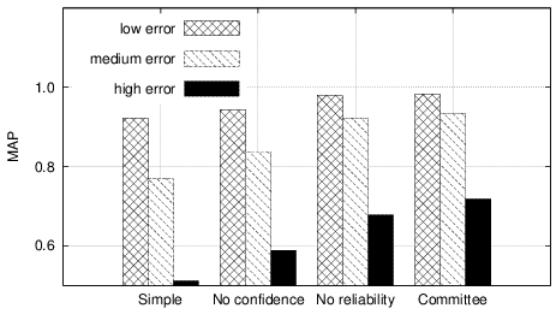

Figure 5: Sensitivity of different components

6 Conclusion

In this paper, we proposed a generic framework that combines a set of similarity predicates to form a committee. We apply this framework to solve the entity resolution problem. To the best of our knowledge, this is the first piece of study about similarity predicate committee. To formulate an effective similarity predicate committee, we need to model two elements, namely, confidences of similarity predicates and reliabilities of attributes. We have explained how they can be modeled on the fly dynamically based on a particular query record without any training data. Extensive experiments are conducted and encouraging results are obtained.

References

[1] R. Ananthakrishna, S. Chaudhuri, and V. Ganti. Eliminating fuzzy duplicates in data warehouses. In Proceedings of the Very Lage Databases (VLDB), 2002.

[2] O. Benjelloun, H. Garcia-Molina, Q. Su, and J. Widom. Swoosh: A generic approach to entity resolution. Technical Report March, Stanford University, 2005.

[3] A. Chandel, O. Hassanzadeh, N. Koudas, M. Sadoghi, and D. Srivastava. Benchmarking declarative approximate selection predicates. In Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD'07), 2007.

[4] S. Chaudhuri, V. Ganti, and R. Kaushik. A primitive operator for similarity joins in data cleaning. In Proceedings of the 22nd International Conference on Data Engineering (ICDE'06), 2006.

[5] W. W. Cohen. Integration of heterogeneous databases without common domains using queries based on textual similarity. In Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD'98), 1998.

[6] W. W. Cohen, P. Ravikumar, and S. E. Fienberg. A comparison of string distance metrics for name-matching tasks. In Proceedings of IJCAI-03 Workshop on Information Integration on the Web (IIWeb'03), 2003.

[7] J. B. Copas and F. J. Hilton. Record linkage: statistical models for matching computer records. Journal of the Royal Statistical Society, 3(153), 1990.

[8] H.-H. Do and E. Rahm. Coma 1 7 a system for flexible combination of schema matching approaches. In Proceedings of the Very Lage Databases (VLDB), 2002.

[9] I. P. Fellegi and A. B. Sunter. A theory for record linkage. Journal of the American Statistical Association, 64(328), 1969.

[10] G. P. C. Fung, J. X. Yu, HaixunWang, D. W. Cheung, and H. Liu. A balanced ensemble approach to weighting classifiers for text classification. In Proceedings of 6nd IEEE International Conference on Data Mining (ICDM'06), 2006.

[11] L. Gravano, P. G. Ipeirotis, H. V. Jagadish, N. Koudas, S. Muthukrishnan, and D. Srivastava. Approximate string joins in a database (almost) for free. In Proceedings of 27th International Conference on Very Large Data Bases (VLDB'01), 2001.

[12] S. Guha, N. Koudas, A. Marathe, and D. Srivastava. Merging the results of approximate match operations. In Proceedings of the Very Lage Databases (VLDB), 2004.

[13] D. Gusfield. Algorithms on Strings, Trees and Sequences: Computer Science and Computational Biology. Cambridge University Press, Cambridge, 1997.

[14] M. A. Hernández and S. J. Stolfo. The merge/purge problem for large databases. In Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD'95), 1995.

[15] M. A. Hernández and S. J. Stolfo. Real-world data is dirty: Data cleansing and the merge/purge problem. Data Min. Knowl. Discov., 2(1), 1998.

[16] M. A. Jaro. Advances in record linkage methodology as applied to matching the 1985 census of tampa. Journal of the American Statistical Association, 1984.

[17] C. Li, B. Wang, and X. C. Yang. Vgram: Improving performance of approximate queries on string collections using variable-length grams. In Proceedings of the 33rd International Conference on Very Large Data Bases (VLDB'07), 2007.

[18] J. Li and A. Deshpande. Consensus answers for queries over probabilistic databases. In Proceedings of the Twenty-Eigth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS'09), pages 259–268, 2009.

[19] D. C. Montogomery and G. C. Runger. Applied Statistics and Probability for Engineers. John Wiley & Sons, Inc., 2nd edition, 1999.

[20] S. Sarawagi and A. Kirpal. Efficient set joins on similarity predicates. In Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD'04), 2004.

[21] S. Tejada, C. Knoblock, and S. Minton. Learning domain-independent string transformation for high accuracy object identification. In Proceedings of the ACM SIG Knowledge Discovery and Data Mining (KDD), 2002.

[22] W. E. Winkler. The state of record linkage and current research problems. US Bureau of the Census, 1999.

[23] W. E. Winkler. Overview of record linkage and current research directions. Technical Report Statistics 2006-2, U.S. Bureau of the Census, Statistical Research, Room 3000-4, 2006.

[24] I. H. Witten and E. Frank. Data Mining: Practical Machine Learning Tools and Techniques. Morgan Kaufmann, 2nd edition, 2005.