

Avoiding Duplicate Records in a Database Using a Linguistic Quantifier Based Aggregation – A Practical Approach

Sławomir Zadrozny, Janusz Kacprzyk, *Fellow, IEEE*, and Grzegorz Sobota

Abstract— We show how Zadeh’s calculus of linguistically quantified propositions can be applied to avoid duplicate names in a publication database of a research institute. This problem, in its most general form, is studied in the literature by various communities. Here we focus on a specific scenario in which a need for its solution arises. Moreover, we make an attempt to apply fuzzy logic based concepts to solve it. The approach proposed yields promising results for the data for which it has been initially conceived and seems to be applicable also in a more general context. Its primary advantage is the ease and intuitiveness of customization. The main parameters take the form of linguistic quantifiers whose meaning is arguably familiar for an average user and which are fairly easy to be tuned to the changing requirements.

I. INTRODUCTION

THE promises of the Information Society are nowadays becoming the reality. More and more domains of our life are supported by information systems. They are getting more sophisticated, more reliable and even adaptive. Modern information systems usually have a complex structure, with many elements (subsystems) working together to accomplish goals set by the human user. Most often the weakest link in this chain of collaborating components is the human operator. A system processes data which is often acquired automatically but still there are many applications where the data has to be entered by a user. In such a case the errors are unavoidable. The most evident context in which it happens is the entering of data to a database. Often it involves massive amounts of data, more and less experienced users, etc. At the same time the quality of the content of a database is usually critical for the mission of the system. Thus any effort reducing chances for errors is worthwhile to be undertaken.

The problem of validating data has been addressed by many research communities, notably in the area of database

management systems. Rough, “syntactical” errors can be fairly easily eliminated by a proper use of masks, validation rules, etc. However there is a class of errors which do not lend themselves for such an easy eradication. Here the prominent role plays the problem of avoiding the assignment of different names to the same entity. This may easily happen due to a simple mistake, due to information integration from different sources, etc. A practical business example is the integration of a number of address databases of potential customers. The same person may appear in many databases and should be properly identified despite possibly different structures of integrated databases, different details of description, possible spelling errors, etc.

The identification of duplicate names is not a trivial task. Our approach should be primarily seen as an attempt to model the way a human being is dealing with such a task. We assume that the decision of a human being is based on the assessment of the truth of the proposition: “ Q (e.g. most) of important words in string 1 have a matching word in String 2”. This is the rationale of our approach.

This problem is studied under different names and in different contexts [3] as, e.g., *record linkage*, *data deduplication*, *name matching* etc. In the database cleansing context it boils down to the identification of several different records in a database representing the same entity. In its more general form it is assumed that records may have both different structures and different values of the fields.

In this paper we consider a more specific problem of matching values of text fields (strings), usually playing the role of a name of an entity represented by a given record. In the next section we state the problem in details focusing on its practical importance. Then we briefly survey the solutions known in the literature. Next we remind the principles of the Zadeh calculus of linguistically quantified propositions which is a theoretical foundation of the solution proposed in this paper. A detailed presentation of this solution follows and is exemplified by a numerical example in the next section.

II. THE PROBLEM

In the paper we consider a fairly simple system which is especially prone to various mistakes, inconsistencies, etc. in data. It is a system meant to support reporting of a research institute concerning the publications prepared by its employees. The system has been in use in its recent version for 10 years. It is used to automatically prepare a number of reports used both for reporting to some supervisory

J. Kacprzyk is with the Systems Research Institute, Polish Academy of Sciences, ul. Newelska 6, 01-447 Warsaw, Poland, and with the Warsaw School of Information Technology (WIT), ul. Newelska 6, 01-447 Warsaw, Poland (Corresponding author: phone: +(48)(22) 3810275; fax: +(48)(22) 3810103; e-mail: kacprzyk@ibspan.waw.pl).

S. Zadrozny is with the Systems Research Institute, Polish Academy of Sciences, ul. Newelska 6, 01-447 Warsaw, Poland (e-mail: zadrozny@ibspan.waw.pl).

G. Sobota is with the Warsaw School of Information Technology (WIT), ul. Newelska 6, 01-447 Warsaw, Poland

This work was partially supported by the Ministry of Science and Higher Education under Grant 3 T11C 052 27.

institutions as well as for various internal, decision making related, purposes. Its heart is a database of publications which is accessible over the Intranet via a Microsoft's Active Server Pages (ASP) driven Web interface. Every researcher is obliged to register his or her publications providing usual details such as the title, list of the authors, title of the edited volume or name of the conference. Among very different mistakes made by the users the one concerning the names of the conferences is especially cumbersome for the system administrator who is responsible for the quality of the data collected in the database. There are two tables in the database (managed by the Microsoft Access software) which are of interest here. One is the main RESULTS table collecting data on all different types of publications and other achievements of the institute employees. Some details, such as title, are recorded directly there. Some other details e.g., concerning the conference at which a paper was presented, are stored in a separate CONFERENCES table. Both tables are linked using usual referential integrity constraints.

When a user enters information on his or her conference paper he or she is provided with a list of conferences taking place in the current year whose descriptions have been already entered by other users as it often happens that a number of researchers participate in the same conference. Nevertheless quite frequently it turns out that there are several records in the CONFERENCES table referring to the same conference.

A partial explanation for this "phenomenon" is that when it comes to registering papers with the system it often happens in a hurry at the end of the year – a usual deadline for completing one's record of achievements. Thus, usually this is the reason for not noticing by the users an entry for the conference on the list presented by the system. This would not be a big problem as long as the duplicates could be automatically identified by the system as having exactly the same description. However, usually this is not the case. In particular the field which theoretically should help to discover the duplicates, i.e., the name of the conference, has usually different form in the duplicate records. This is sometimes due to some typing errors but also due to ambiguous multipart long names of the conferences which are selectively and "creatively" interpreted by particular users. The system cannot prevent that using standard means, i.e., checking if the name of just registered conference already appears in the database. This line of defense works when the name given by the user is exactly the same as that earlier recorded in the database. However checking the content of the CONFERENCES table one can spot the following examples:

"Third Warsaw International Seminar on Soft Computing, WISSC-3'03"

and (1)

"3rd Warsaw International Semnar on Soft Computing"

or

"1st international conference on advances in medical signal and information processing (MEDSIP' 2000)"

and (2)

"International Conference on Advances in Medical Signal and Information Processing"

In the exemplary pairs of the conferences names (1) and (2) one may observe various problems: different wording of the same concept ("*Third*" and "*3rd*"), typing errors ("*Semnar*"), varying length of the names in a pair etc.

A similar problem – although to a lesser extent - concerns titles of edited volumes in which the institute employees place their papers, names of publishing houses or names of institutions at which the researchers have accomplished some achievements to be reported in the institute. This is a real nightmare for the administrator who has to maintain the database correcting improper entries and removing duplicates concerning the same event, institution etc. several times per year.

Several administrative measures were planned to avoid duplicate entries in a database – for example introducing a policy that the user cannot himself enter a conference description to the database but has to request in advance the administrator to enter it. However it was quickly realized that such measures make the system inconvenient to use and defies its whole mission. Thus it has been decided to supplement the system with a module helping to avoid duplicates. Namely, when a user tries to add a conference description to the database its name is checked in an intelligent way against the names of recorded conferences so as to discover potential duplicates and warn the user.

It is worth noticing that there are in fact two scenarios for the use of such a module. They may be referred to as *on-line* and *off-line*. The former boils down to a short description given above, i.e., newly added conference is checked against the existing ones. In the latter all conferences recorded in a database are checked against each other. The first mode of the execution is triggered by the user, the second by the administrator whose goal is to discover all potential duplicates. The type of the mode may have an influence on the construction of an algorithm. For example, in the first scenario it may be assumed that the database is clean, i.e., there are no duplicates. Then finding several potential matches for a newly added conference should be possibly interpreted as a mere consequence of the use of common (for conference names) words in the name of the conference rather than as the sign of the conference being a real duplicate. We will refer to that once again at the end of this paper.

The planned implementation of the module was preceded by a study of the existing approaches to the problem of a so called *deduplication*. Due to the experience of the current authors [11,12] they were also called for an advice. Finally it has been decided to implement a new intelligent algorithm –

after all, this is a research institute in the IT field! This paper presents an effort undertaken.

III. SOME KNOWN APPROACHES TO THE NAME MATCHING PROBLEM – A VERY BRIEF REVIEW

The problem defined in the previous section is a specific case of the broader problem of *data deduplication* whose various interpretations are listed in the introduction. Here we are concerned with name matching techniques which are a part of the solution to this problem. Their down-to-earth interpretation boils down to string comparison. Many techniques have been developed for this task over the years. Basically they consist in computing some kind of similarity (or dissimilarity) measure between two strings. Elmagarmid et al. [3] distinguish a few classes of such measures (metrics) which are meant for identifying strings which are distorted versions of each other. These classes correspond to various causes of this distortion. We will briefly discuss them referring the reader to [3, 2] for details and further references.

The first class of character-based measures addresses the typing errors, i.e., is meant to discover the match between such strings as, e.g., “international” and “interntional”. Here belongs a famous *edit distance* also known as the *Levenshtein distance*. Dissimilarity between two strings is computed as a minimal number of insertions, deletions and replacements of single characters needed to transform one string into another. Different weights may be associated with particular operations. Many extensions of this technique have been proposed making it possible to identify, e.g., the abbreviations of the same word. Some efficient algorithms mostly relying on the use of dynamic programming have been devised.

Another metric which is widely acclaimed as giving good results (at least for short strings) is the *Jaro distance metric* and its variations. This method consists in identifying common characters in both strings, i.e., characters from one string which have an identical counterpart, possibly slightly misplaced, in another string. The sets of common characters are computed separately for both strings – they usually are different. Finally the measure is computed as a function of these two sets of common characters. The *Jaro-Winkler* variant of this distance metric additionally takes into account the length of the longest common prefix of compared strings.

There are also *two level distance metrics* which are defined as follows:

$$sim(s,t) = \frac{1}{K} \sum_{j=1}^K \max sim'(a_i, b_j)$$

where strings s and t are represented by sets of its substrings (tokens), i.e., $s=\{a_i\}$ and $t=\{b_j\}$ and sim' is a secondary distance metric. For example, such a metric in which the secondary distance metric is *Jaro-Winkler* metric will be referred to as *Level2Jaro-Winkler*.

Another approach giving good results consists in using substrings (also known as n -grams) as the units of comparison between two strings – instead of single characters.

The second class of token-based measures is meant mainly to identify strings distorted by the misplacement, omission etc. of words, i.e., to discover the match between such strings as, e.g., “International Conference on Systems Sciences” and “15th International Conference on Systems Science”. Basically the methods in this class divide the string into tokens (words) and then apply various techniques of documents matching developed in the framework of the information retrieval such as the $tf \times IDF$ weighting or probabilistic approaches in the spirit of language modeling. The measures of the first class are also applicable here to neutralize typing errors.

Finally, there are methods addressing the problem of identifying strings which are misspelled due to their different written and phonetic forms. Here belongs a very popular *SOUNDEX* algorithm which basically works by computing a compressed form (code) of the string using some rules to some extent specific for a given natural language. Then two strings are identified if their codes are equal. There exist some variations of this general mode of operation.

There are also many hybrid methods combining advantages of different approaches mentioned above. For example, classical information retrieval related techniques are rather useless in case of spelling errors. However if we apply these techniques to n -grams instead of whole words, they become spelling-error-proof.

On a very abstract level some common feature of virtually all of abovementioned methods may be observed. Namely, usually some partial scores are computed reflecting the similarity (dissimilarity) of words (tokens) and whole strings which are later somehow aggregated to obtain the final measure.

Such an aggregation should be as human consistent as possible. Therefore, we assume that the decision of a human being is based on the assessment of the truth of the proposition: “ Q (e.g. most) of important words in string 1 have a matching word in String 2”. Its very essence is a soft aggregation of partial scores (results of substring comparisons),

In our approach we propose to employ the linguistic quantifier driven aggregation for this purposes. For example, as mentioned above, we will identify two strings as identical if, e.g., *most of their important words are very similar*. In the following sections we first briefly remind Zadeh’s proposal to model expressions of this form and next we propose how particular components of expressions of this type, such as *important word*, *similar word* may be interpreted. An approach similar in the spirit to the one proposed in this paper has been developed by the authors of [6] but it was meant for rather different purposes.

IV. ZADEH’S CALCULUS OF LINGUISTICALLY QUANTIFIED PROPOSITIONS – A TOOL TO AGGREGATE PARTIAL MATCHING DEGREES

As a tool we will use the Zadeh [9] *calculus of linguistically quantified propositions*. The propositions under consideration are of the following form:

Most elements of set X possess property S (3)

that may be formally expressed as follows:

$$\bigvee_{x \in X} S(x) \quad (4)$$

where Q is a *fuzzy linguistic quantifier* (e.g., "most"), $X = \{x_1, \dots, x_n\}$ is a universe of discourse, and $S(\cdot)$ is a property (of the elements of X) which is assumed fuzzy, and $\text{truth}(S(x_i)) = \mu_S(x_i)$. In a more general form the propositions (3) may be expressed as follows:

Most elements of set X possessing property F possess also property S (5)

that may be formally expressed as follows:

$$\bigvee_{x \in X} QF S(x) \quad (6)$$

where property F is fuzzy too, and $\text{truth}(F(x)) = \mu_F(x)$. If $\forall x \in X \mu_F(x) = 1$, i.e., $F = X$, then (5) and (6) reduce to (3) and (4), respectively.

The linguistic quantifier can be absolute (e.g., "about 5", "more or less 100", "several"), and relative (e.g., "a few", "more or less a half", "most", "almost all").

In the framework of the Zadeh calculus of linguistically quantified statements the truth values of the statements (6) are calculated using the concept of ΣCount , a scalar cardinality of fuzzy sets: $\Sigma\text{Count}(F) = \sum_{x \in X} \mu_F(x)$. The truth values are calculated as follows. The properties S and F are interpreted as fuzzy sets in X . The quantifiers are modeled by fuzzy sets in the real line or the interval $[0,1]$ for the absolute and relative quantifiers, respectively. We will use only relative quantifiers or, more specifically, proportional, nondecreasing quantifiers. A fuzzy set representing such a linguistic quantifier Q may be exemplified by the following membership function:

$$\mu_Q(x) = \begin{cases} 1 & \text{for } x \geq 0.8 \\ 2x - 0.6 & \text{for } 0.3 < x < 0.8 \\ 0 & \text{for } x \leq 0.3 \end{cases} \quad (7)$$

Then, due to Zadeh [9]:

$$\text{truth}_{x \in X}(QF S(x)) = \mu_Q \left[\frac{\sum_{i=1}^n (\mu_F(y_i) \wedge \mu_S(y_i))}{\sum_{i=1}^n \mu_F(y_i)} \right] \quad (8)$$

where \wedge denotes the minimum operator.

The membership function (7) may be treated as parametrized by two parameters A and B – in (7) they are equal 0.3 and 0.8, respectively. Thus later we will refer to Q as $Q(A,B)$.

The basic Zadeh's approach to modeling linguistic quantifiers is very clear and intuitively appealing and is the most widely used. There are many other possible approaches, e.g., via the use of the OWA operators [7].

V. LINGUISTIC QUANTIFIER GUIDED AGGREGATION FOR NAMES MATCHING

The problem considered in the paper in the assumed context boils down to a definition of the similarity measure of two strings representing names (titles) of some entities. Our goal is to define such a measure which is intuitive and easily customizable for a human operator. In the abstract form the basic version of the proposed measure may be expressed as follows:

$$\text{sim}(S1, S2) = \text{truth}(Q \text{ of important words in } S1 \text{ have a matching word in } S2) \quad (9)$$

where $\text{sim}(S1, S2)$ denotes the similarity measure between two compared strings $S1$ and $S2$. In order to make this definition operational we have to further define the elements set in (9) in italics. In what follows we will denote the symbols of measures, membership functions, etc. using small letters and their corresponding properties, fuzzy sets, etc. with the same symbols using the capital letters (cf., e.g., cimp and CIMP on the next page).

Quantifier Q is to be selected from among a list of quantifiers defined in the system. A good starting point may be the one defined by (7). In fact, this is exactly the choice of a quantifier which makes our proposed definition of similarity easily and intuitively customizable. We will refer back to this issue after discussing the remaining components of (9).

The degree of importance of a word will be determined by two factors: its position in the string and its uniqueness in the whole collection of the names under consideration. It may be argued that the words closer to the beginning of the string are more important in the following sense. If they do appear in the second string then there are higher chances that both names (strings) refer to the same entity. Otherwise, these chances are lower. However it should be taken into account that some words are very common for the names of a given class of entities. In our case the names of conferences very often contain the words "conference", "international" etc. Thus their influence on the judgment regarding the similarity of strings has to be reduced. In information retrieval theory (cf., e.g., [1]) there are two basic mechanisms which are relevant for this purposes. Firstly, so-called stopwords are usually removed from the documents in the preprocessing step. These are primarily the words which do not carry any meaning on their own – such as the articles, connectives etc. One should be very careful applying this technique in the context considered here as the names of entities are rather short and each word may be fairly important for the comparison among them. Another technique is more promising for our purposes and consists in assigning weights to particular words in such a way that words appearing (at least once) in many documents are assigned lower weights than those which are relatively rare. The most popular formula for calculating the weights in such a way is the famous *inverse document frequency (IDF)*, which has the following form:

$$\text{IDF}(w_i) = \log N / n_i \quad (10)$$

where: w_i is a word, N is the total number of documents in a given collection, and n_i is the number of documents in this collection containing the word w_i . Formula (10) is explicitly used in the weighting schemes of the vector space model and implicitly, with a slight variation, in the probabilistic model – two leading generic models considered in the information retrieval theory.

In our approach we combine the position of the word in the string with its normalized IDF measure. The normalization consists in dividing the measure of a given word by the highest IDF measure of the other words appearing in the same string. The combination of both components contributing to the importance measure of a word is done using a t -norm, notably via the minimum operator. Thus the formula for the importance of a word may be expressed in the natural language as follows:

$$\text{wimp}(w,S) = \text{truth}(\text{The word } w \text{ is } \textit{near} \text{ the beginning of the string } S \text{ and is } \textit{unique} \text{ in the collection of the strings}) \quad (11)$$

where the nearness to the beginning is modeled by the fuzzy set NEARBS such that:

$$\mu_{\text{NEARBS}}(w, S) = 1 - f(\text{wpos}(w,S) / |S|)$$

where $\text{wpos}(w,S)$ is a position of the word w in the string S (counting from 0) and $|S|$ denotes the length of the string (in words), and f is a scaling function, which in our preliminary experiments is assumed for simplicity to be the function $f(x)=x$. We have also experimented with another version of the scaling function:

$$f(x) = Q(x) \quad (12)$$

where Q is a linguistic quantifier defined by its membership function given by (7).

The (degree of) uniqueness of a word is measured using the IDF formula (10) and thus the overall formula for the importance corresponding to (11) is as follows:

$$\text{wimp}(w_i,S) = \mu_{\text{NEARBS}}(w_i,S) \wedge \text{IDF}(w_i) / \max_w \text{IDF}(w,S) \quad (13)$$

Now we have to clarify the concept of a word in the string S_1 having a matching word in the string S_2 , cf. the formula (9). Here and later on by defining a concept we will mean its natural language like definition with a formula for calculating the extent to which a particular element satisfies the definition, and this extent is expressed in terms of a truth value. Briefly speaking w_2 in the string S_2 is a matching word for w_1 in the string in S_1 if w_1 is *similar* to w_2 and the positions of both words in their strings are *close*. Similarity of the words is in turn defined using the following natural language expression:

$$\text{wsim}(w_1,w_2) = \text{truth}(Q \text{ of } \textit{important} \text{ characters in } w_1 \text{ have a } \textit{matching} \text{ character in } w_2) \quad (14)$$

where the *importance* of a character ch_1 in the word w_1 and the concept of having a matching character ch_2 in the word w_2 is defined, somehow in a recurrent way, similarly to the

corresponding concepts regarding the whole strings and appearing in (9). This will be discussed in the next paragraphs after formalizing the expression (14). Thus let us assume that we have a measure $\text{cmatch}(c,w_1,w_2)$ expressing the degree to which a character c in the word w_1 has a matching character in the word w_2 as well as a measure $\text{cimp}(c,w)$ expressing the degree of importance of the character c in the word w . Then we are in a position to calculate the truth degree of the formula (14) using the formula (8) which takes now the following form:

$$\text{wsim}(w_1,w_2) = \text{truth}(\bigvee_{c \in w_1} \text{CIMP} \text{ CMATCH}(c)) = \mu_Q \left[\frac{\sum_{i=1}^{|w_1|} (\mu_{\text{CIMP}}(c_i) \wedge \mu_{\text{CMATCH}}(c, w_1, w_2))}{\sum_{i=1}^n \mu_{\text{CIMP}}(c_i)} \right] \quad (15)$$

Then the concept of a word in the string S_1 having a matching word in the string S_2 , denoted as $\text{wmatch}(w,S_1,S_2)$, is finally modeled by the following natural language expression:

$$\text{wmatch}(w_1,S_1,S_2) = \text{truth}(\text{There exists a word } w_2 \text{ in } S_2 \text{ matching the word } w_1)$$

what is formally expressed as:

$$\text{wmatch}(w_1,S_1,S_2) = \text{truth}(\exists w_2 \in S_2 \text{ WSIM}(w_1,w_2)) = \max_{w_2 \in S_2} \text{wsim}(w_1, w_2) \quad (16)$$

where the existential quantifier \exists is modeled by the maximum operator.

The degree of importance of a character c in a word w , denoted as $\text{cimp}(c,w)$, will be determined first of all by its position in the string: the closer the character to the string beginning the more important it is. There may be other reasonable factors to differentiate the importance of the characters in a string. For example it may be argued that consonants are more important than vowels when comparing two strings. In our experiments we have tested also such an extended definition of the character importance. However basically the importance of a character is modeled using the concept of nearness to the beginning of the word:

$$\text{cimp}(c,w) = \text{truth}(\text{The character } c \text{ is } \textit{near} \text{ the beginning of the word } w) \quad (17)$$

which in turn is modeled by the fuzzy set NEARBW:

$$\mu_{\text{NEARBW}}(c, w) = 1 - g(\text{cpos}(c,w) / |w|)$$

where $\text{cpos}(c,w)$ is a position of the character c in the string w (counting from 0) and $|w|$ denotes the length of the word (in characters), and g is a scaling function, which in our preliminary experiments is assumed to be the function $g(x)=x$. Also some experiments with $g=f$, where f is defined as in (12) were undertaken.

The last concept we have to formalize is that of a character in the word w_1 having a matching character in the word w_2 , cf. the formula (14). We consider c_2 in the word w_2 as a matching character for c_1 in the word w_1 if c_1 and c_2 are identical and their positions in their strings are *close*.

Closeness of the positions is in turn modeled by the fuzzy set CLOSE such that:

$$\mu_{CLOSE}(i, j, w_1, w_2) = 1 - h(\text{abs}(i-j)/(|w_1|+|w_2|))$$

where $\text{abs}(x)$ is an absolute value of x , $|w_i|$ is the length of the word w_i and h is a scaling function which in our experiments is assumed to be $h(x)=x$. Now we can define the formula for $\text{cmatch}(c, w_1, w_2)$ to be as follows:

$$\text{cmatch}(c, w_1, w_2) = \max_{c_2 \in w_2} ((c_2 = c) \wedge \mu_{CLOSE}(cpos(c, w_1), cpos(c_2, w_2), w_1, w_2))$$

Finally we can express the measure of similarity between two string $S1$ and $S2$ as the truth degree of proposition (9) which, by using formula (8), takes the following form:

$$\text{sim}(S1, S2) = \text{truth} \left(\bigwedge_{w \in S_1} \text{WIMP WMATCH}(w, S1, S2) \right) = \mu_Q \left[\frac{\sum_{i=1}^{|S_1|} (\mu_{WIMP}(w_i) \wedge \mu_{WMATCH}(w, S1, S2))}{\sum_{i=1}^{|S_1|} \mu_{WIMP}(w_i)} \right] \quad (18)$$

Let us summarize the steps needed to calculate the similarity degree given by (18) in the form of the following pseudocode:

```

res = 0;
sumimp = 0;

foreach w1 in S1
{
  imp = wimp(w1, S1);
  bestmatch = 0;
  foreach w2 in S2
  {
    m = wsim(w1, w2);
    if (m > bestmatch)
      {bestmatch = m};
  }
  res = res + tnorm(bestmatch, imp);
  sumimp = sumimp + imp;
}
res = Q(res/sumwi);

```

where the functions $wimp$ and $wsim$ correspond to the measure defined in this section and the functions Q and $tnorm$ implement the concept of a linguistic quantifier and t -norm, respectively. The details of these functions are obvious and will be omitted here.

The basic structure of the proposed algorithm for the names matching is fairly similar to some of those mentioned in the literature. Namely there are three levels: the level of strings, words and characters. The partial scores of similarity are computed on the levels of words and characters and later aggregated on the higher level. Our main goal was to check how linguistic quantifiers may be applied to guide the aggregation process on different levels. Thanks to their intuitive interpretation they very good serve the purposes of

customization of the whole procedure. Namely, changing the quantifiers employed in (9) and (14) one may relax and harden the requirements set for the matching of the whole strings and words, respectively. For example, changing the *most* quantifier in (14) to *almost all* will require more (*important!*) characters to match in order to declare two words as matching. Such a change may be executed by the system administrator in a response to the user complaints that the lists of potential duplicates shown by the system are too long and missing the point. Customizing the system by using such intuitive measures may be very comfortable for an administrator who is not an IT specialist, what is in fact the case in the system considered in this paper.

For the discussion of the proposed solution some practical aspects should be taken into account. Namely three scenarios of its use may be considered:

- I. by a regular user when the database may be assumed to be clean, i.e., the administrator of the system has just checked the list of the conferences removing existing duplicates and correcting improperly entered descriptions,
- II. by a regular user when the database should be assumed to possibly contain a number of duplicates as it has not been checked by the administrator for some time,
- III. by the administrator when the goal is to discover all potential duplicates existing in the database unlike in the previous two scenarios where a list of potential duplicates for a given record is to be returned.

Most interesting is the distinction between the two first scenarios. It is worth noticing that the concept of cleanliness of the database may be assumed fuzzy and associated with a membership function related to the period of time elapsed since the last check done by the administrator. The shorter (longer) this period for a given database the higher (lower) its membership degree to the set of clean databases. In fact, time should be here rather understood in terms of the number of records added to the table under consideration (in the considered system this is the table of conferences and only additions to it are allowed for regular users thus we do not have to take into account possible modifications). Thus the cleanliness of the database may be taken as another parameter of the proposed algorithm. For example if in a clean database a relatively long list of potential duplicates is discovered for a given record entered by the user it may rather be a result of a non-specific name (of a conference in our case) rather than of a real duplication of data.

In order to further discuss the proposed solution the following should be noted:

- the proposed measure of similarity given by (18) is not symmetric,
- the proposed measure makes it possible to order known strings (here: names of the conferences) so that a potential duplicate for a given string goes first; however it does not give a clear cue which strings from the beginning of the list should be still considered as potential duplicates and which already not.

The first observation may be seen as a nuisance but in fact may be quite reasonable to accept and helpful. This asymmetry will manifest itself, e.g., when comparing a string with another one which is its prefix. The first longer string will match the second to a lower degree than the second will match the first. This may be a proper behavior in the scenario I mentioned in the previous page. Namely, it may be assumed that correct names of the conferences are usually longer than their duplicates entered by regular users – people are often lazy! Thus there is really a higher chance that a shorter name is a duplicate of the longer than the opposite, what is properly reflected by the asymmetry of the proposed measure. Of course, this rationale may work or not for a given community of the system users. If it is necessary the proposed measure may be made symmetric by taking, e.g., the minimum or the maximum of the two degrees $\text{sim}(S1,S2)$ and $\text{sim}(S2,S1)$.

The second observation possibly requires additional efforts to make the system more successful in avoiding the duplicates. The simplest solution is to present the user with an ordered list of all conferences known in the system arranged in such a way that at the top are those assessed as most similar to the one entered by the user, i.e., those with the highest value of the similarity measure (18). However, if the lists are getting longer and longer the users may become reluctant to consult them and they will prefer to enter the conference than to check if it already exists in the system. In such a case a technique known as the *thresholding strategy* [8,12] may help to decide where to cut a list and make it more user-friendly. A simple thresholding strategy consists in assuming a similarity threshold value. Then only these pairs of the conference names are treated as duplicate whose similarity degree is higher than assumed threshold value.

VI. NUMERICAL RESULTS

We have implemented a prototype versions of our algorithm in PERL and C#. We run a few experiments in the real system whose CONFERENCES table contained 923 records. A few of them were removed during preprocessing as they were incomplete and never used in fact. All conference names were capitalized for the purposes of these experiments.

We experimented with different settings of the parameters of our algorithm. The best results reported below were obtained for the following setting:

- word importance is computed using the scaling function (12), where $Q(0.1, 0.75)$ is used (cf. (7))
- the quantifiers in (15) and (18) take the form $Q(0.2,0.85)$
- $\max(\text{sim}(S1,S2),\text{sim}(S2,S1))$ is taken as the final similarity degree

Our basic experiments cover different scenarios mentioned in the previous section. In scenario III all records are compared with each other. In order to make the experiment in line with the pragmatics of the real system only conferences taking place in the same year are compared. This yields 44980 pairs of conferences to be compared.

Among them there are 123 pairs of actual duplicates. In this scenario we have compared results of our algorithm with the following algorithms discussed in [2]: TFIDF, Jaro, Jaro-Winkler, Level2Jaro-Winkler. All these algorithms, including our, yield as a result of the comparison of two names a number from the interval [0,1]. Thus we evaluate their effectiveness computing the F1 measure ($F1=(2*Precision*Recall)/(Precision+Recall)$) for various threshold values equal 0.5, 0.6 and so on, and choosing the highest value of F1 measure for given algorithm. Thus for each algorithm the following steps are executed for each threshold value. The similarity of all 44980 pairs of conference names is computed. Then all pairs for which the computed similarity degree is higher than the threshold value are considered as candidate-duplicates. The precision is computed as the fraction of actual duplicates among candidate-duplicates and the recall is computed as the fraction of candidate-duplicates being actual duplicates among all actual duplicates (i.e., 123). The results of this comparison are collected in Table I.

TABLE I
COMPARISON OF SELECTED ALGORITHMS USING F1 MEASURE

Algorithm	F1
Level2Jaro-Winkler	0.6754
TFIDF	0.6204
Our algorithm	0.5198
Jaro	0.4268
Jaro-Winkler	0.3247

In the second tested scenario (referring to the scenario II mentioned in previous section) only conference names being actual duplicates are concerned. There are 179 of them in the database. Algorithms are evaluated in the following steps. Each of 179 names is in turn treated as a name NM of the conference just entered by the user and is compared with the remaining 178 names. These 178 names are ordered non-increasingly according to their similarity degree to NM computed by given algorithm. Then the measure of the Average Precision At Seen Relevant Documents (AtSeen), cf., e.g., [1], is computed, i.e., positions of all actual duplicates of NM (there is at least one!) are found in the ordering of all names and the average precision measure at all these position is computed. The overall effectiveness measure of given algorithm is the averaged AtSeen measure over all 179 names treated in turn as NM . The results of the comparison are shown in Table II.

TABLE II
COMPARISON OF SELECTED ALGORITHMS USING ATSEEN MEASURE

Algorithm	AtSeen
Level2Jaro-Winkler	0.8343
TFIDF	0.8120
Our algorithm	0.7488
Jaro	0.6245
Jaro-Winkler	0.6092

First results obtained seem to be promising. In both scenarios our algorithms works sufficiently well. In particular the AtSeen effectiveness measure shown in Table II is fully satisfactory for the practical case considered. As there are rarely more than 3 duplicates concerning the same conference then the 0.75 level of this measure means that on a short 5 item list of most similar names indicated by the algorithm all actual duplicates are always present.

As mentioned earlier the main advantage of the proposed approach is high interpretability of its parameters as, in fact, almost all measures underlying the proposed algorithm are expressed in (quasi)-natural language and then represented using Zadeh's calculus of linguistically quantified propositions.

VII. CONCLUDING REMARKS

We showed how linguistic quantifiers may be applied in the construction of a name matching algorithm to be used to discover the duplication of names in records of a publication database. One of the main advantages of their use is here the ease and intuitiveness with which a configuration of the system may be executed. In this respect further effects may be obtained by the direct use of the *protoform* concept [10,5]. The formula (18) refers to a hierarchy of linguistically quantified propositions whose building blocks, notably the linguistic quantifiers, may be selected depending on the current need. The interpretation of this hierarchy in terms of the protoform may support "navigation" in it.

Of course further experiments on different datasets are needed to prove advantages of the proposed algorithm. Moreover, there are still some more parameters that may be tuned yielding possibly even better results. A general scheme of parameters tuning has to be conceived and will be a subject of our further research.

REFERENCES

- [1] Baeza-Yates R. and Ribeiro-Neto B., Modern information retrieval. ACM Press and Addison Wesley, 1999.
- [2] Cohen W.W., Ravikumar P. and Fienberg S.E., A Comparison of String Distance Metrics for Name-Matching Tasks. In Proceedings of IJCAI-03 Workshop on Information Integration on the Web (IIWeb-03), Acapulco, Mexico, 2003, pp. 73-78
- [3] Elmagarmid A.K, Ipeirotis P.G. and Verykios V.S., Duplicate record detection. IEEE Transactions on Knowledge and Data Engineering, vol. 19, no. 1, 2007, pp. 1-16.
- [4] Kacprzyk J. and Zadrozny S., Computing with words in intelligent database querying: standalone and Internet-based applications, Information Sciences, 2001, vol. 34, pp. 71-109.
- [5] Kacprzyk J. and Zadrozny S., Linguistic database summaries and their protoforms: towards natural language based knowledge discovery tools. Information Sciences, 2005, vol. 173, pp. 281-304.
- [6] Szczepaniak, P. S. and Niewiadomski, A., Internet search based on text intuitionistic fuzzy similarity. In Intelligent Exploration of the Web, P. S. Szczepaniak, J. Segovia, J. Kacprzyk, and L. A. Zadeh, Eds. Studies In Fuzziness and Soft Computing. Physica-Verlag GmbH, Heidelberg, Germany, 2003, 96-102.
- [7] Yager R.R. and Kacprzyk J., The Ordered Weighted Averaging Operators: Theory and Applications. Kluwer, Boston, 1997.
- [8] Yang Y., A study on thresholding strategies for text categorization. In Proceedings of ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'01), 2001, 137-145.
- [9] Zadeh L.A., A computational approach to fuzzy quantifiers in natural languages. Computers and Maths. with Appls. 9, 1983, pp. 149 - 184.
- [10] Zadeh L.A., A prototype-centered approach to adding deduction capabilities to search engines - the concept of a protoform. In Proceedings of the Annual Meeting of the North American Fuzzy Information Processing Society (NAFIPS 2002), pp. 523- 525.
- [11] Zadrozny S. and Kacprzyk J., On the application of linguistic quantifiers for text categorization. In Proceedings of International Conference on Fuzzy Information Processing, Beijing, China, 2003, vol. 1, 435-440.
- [12] Zadrozny S. and Kacprzyk J., Computing with words for text processing: an approach to the text categorization. Information Sciences, 176(4), 2006, 415-437.