Content Data Based Schema Matching

Marcin Szymczak, Antoon Bronselaer, Sławomir Zadrożny and Guy De Tré

Abstract A novel automatic method for detecting corresponding attributes in schemas based on content data is studied. More specifically, our proposed method for the detection of coreferent attributes in schemas is based on a statistical and lexical comparison of content data and detected coreferent tuples across multiple datasets, which increase the possibility of correct schema matching. We will show that knowledge of even a small number of coreferent tuples is sufficient to establish correct matching between corresponding attributes of heterogeneous schemas. The behaviour of the novel schema matching technique has been evaluated on several real life datasets, giving a valuable insight in the influence of the different parameters of our approach on the results obtained.

1 Introduction

The existence of coreferent content data (coreferent tuples, duplicates) which describe the same entity but in a different way across multiple, related databases significantly lowers data quality and should be avoided. However, a small number of coreferent tuples can be useful in the data integration process, which involves importing data from one source to another. Namely, coreferent tuples may be helpful

M. Szymczak (✉) · S. Zadrożny
Systems Research Institute, Polish Academy of Sciences, Newelska 6,
01-447 Warsaw, Poland
e-mail: szymczak@ibspan.waw.pl

S. Zadrożny
e-mail: zadrozny@ibspan.waw.pl

M. Szymczak · A. Bronselaer · G. De Tré
Department of Telecommunications and Information Processing, University Ghent,
St-Pietersnieuwstraat 41, 9000 Ghent, Belgium
e-mail: antoon.bronselaer@ugent.be

G. De Tré
e-mail: guy.detre@ugent.be

G. De Tré et al. (eds.), *Challenging Problems and Solutions in Intelligent Systems*, Studies in Computational Intelligence 634,
DOI 10.1007/978-3-319-30165-5_14

Table 1 Example of objects extracted from the source dataset *S*

Key	Name	Lon.	Lat.	Category	Address
1	Belfry & Cloth Hall	3.724911	51.053653	Tourist attract	Sint-Baafsplein, 9000 Ghent
2	Saint Bavo	3.797826	50.984194	Church	Sint-Baafsplein, 9000 Ghent
3	Cafe-Restaurant De Ster	4.050876	51.281777	Restaurant	Grotestraat 91, 7471 BL Goor
4	Het Kouterhof	3.665122	51.034331	Lodging	Stoopkensstraat 24, 3320 Hoegaarden
5	Borluut B&B	3.657992	51.018882	Lodging	Kleine Gentstraat 69, 9051 St-Denijs-Westrem
6	Gravensteen Hotel	3.719741	51.056485	Hotel	Jan Breydelstraat 35,9000,Ghent
7	Carlton Hotel	3.713951	51.036280	Lodging	Chartreuseweg 20, 8200 Brugge
8	Vlaamse Opera	3.722336	51.049746	Theater	Schouwburgstraat 3, 9000 Ghent

in establishing a true matching between the corresponding attributes of heterogeneous database schemas. This is known as the *schema matching* problem, which is the first step in data integration and is investigated in this paper.

1.1 Problem Illustration

As a motivating example let us consider the schema matching scenario of two datasets of points of interest (POIs) in which corresponding attributes in the source dataset *S* in Table 1 and the target dataset *T* in Table 2 have to be aligned as in Fig. 1.[1] The attributes "Key", "Name", "Lat", "Lon" and "Category" in Table 1 have to be matched to the attributes "ID", "POI", "Geo1", "Geo2" and "Type" in Table 2, respectively. It is obvious that matching techniques which are based on the attributes' names are not capable to establish all of these matchings. Semantical matching of corresponding attributes has to be established as coreferent attributes may have different names. Moreover, an attribute "Address" in the left part of Fig. 1 and in Table 1 is decomposed into a number of (sub)attributes: "Street", "City", "ZipCode" in the right part of Fig. 1 and in Table 2. Thus, a one-to-many matching between these attributes is required; namely, a concatenation function has to be applied to solve the *attribute*

[1]The order of datasets does not matter, i.e., there exists schema matching between corresponding attributes from the source dataset and the target dataset, and vice versa.

Table 2 Example of objects extracted from the target dataset T

ID	POI	Geo1	Geo2	Type	Street	City	ZipCode
1	Belfort en Lakenhalle	51.054898	3.721675	Bell tower	Emile Braunplein	Gent	9000 BE
2	Sint-Bavokerk	51.054898	3.721675	Church	Sint-Baafsplein	Gent	9000 BE
3	Cafe Theatre	51.049830	3.722015	Restaurant	Schouwburgstraat 5-7	Gent	9000 BE
4	Het Kouterhof	51.034379	3.665140	Hotel	Stoopkensstraat 24	Hoegaarden	3320 BE
5	Borluut Bed Breakfast	51.018938	3.657975	Hotel	Kleine Gentstraat 69	St-Denijs-Westrem	9051 BE
6	Hotel Gravensteen	51.056465	3.719741	Hotel	Jan Breydelstratt 35	Gent	9000 BE

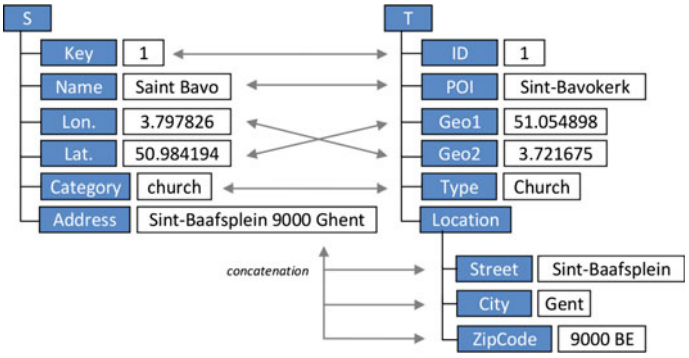

Fig. 1 Example of schema matching. *Arrows* represent matchings of coreferent schema attributes

granularity problem. In general also the *coverage* problem of matched different attributes exists, i.e., coreferent attributes do not necessarily completely have to represent the same information; for example, the attribute "Address" in Fig. 1 and in Table 1 does not contain information about the country. Moreover, due to errors, inaccuracies and lack of standard, coreferent data are not bound to be equal, i.e., the Belfry in Ghent has a different category in the considered tables. It should be clear that detected coreferent tuples do not guarantee perfect schema matching, i.e., the attributes "Name" and "Type" may contain similar values, e.g., *cafe* or *theatre*, which may mislead the matching system. Therefore, all of this makes the finding of coreferent data in schemas using content data a challenging task.

Examples of coreferent tuples are the objects described in the first, second, fourth, fifth and sixth rows in Tables 1 and 2, respectively. They have slightly different names, similar geographic coordinates and different categories and addresses, but they are still describing coreferent objects. These detected coreferent tuple pairs in the considered datasets are used to derive schema matching, known as *horizontal*

matching. The same or similar attribute values among coreferent tuple pairs imply coreference of the corresponding attributes of the schemas.

However, detecting coreferent tuple pairs without having knowledge about the correspondences between the attributes of heterogeneous schemas (known as schema alignment) is time-consuming and error prone. It requires the comparison of the values of each attribute from one schema with the values of each attribute from the other schema. Thus, one of the main challenges in the efficient detection of coreferent tuple pairs is the reduction of the set of attributes involved in the comparison to those that may correspond to each other. For this purpose our content data-based approach statistically and lexically compares the attributes' effective domains and selects potentially corresponding (coreferent) attributes which are called *candidate* attributes. This method is known as vertical matching. It significantly decreases the number of comparisons and increases the quality of coreferent tuple pairs detection. Candidate attributes give the first tips of the coreference among attributes, which is confirmed or rejected by detected coreferent tuples or even may be the basis to establish schema matching in case of a lack of coreferent tuples. However, it should be clear that vertical matching is necessary but not sufficient for efficient attribute coreference identification. Thus, our approach is a combination of vertical and horizontal schema matching methods used to establish the matching of corresponding attributes.

Many problems have to be addressed while devising such a schema matching algorithm. To sum up, the most important among them are the following:

- How can content data be useful in schema matching?
- How can one-to-one and one-to-many semantic matching be established between corresponding attributes?
- How can attribute granularity and the coverage matching problems occurrence be recognized?

1.2 Contributions

The objective of this paper is to propose a novel automatic semantical matching method of corresponding (coreferent) attributes in schemas based on data and metadata. More specifically, the detection of coreferent attributes in schemas is based on statistical and lexical analysis of content data and detected coreferent tuples across pairs of datasets, which increases the confidence in schemas matching. In other words, our method is a combination of vertical and horizontal schema matching techniques that applies possibilistic truth values (PTVs) and a kind of cardinality of a set of PTV to express the uncertainty about the matchings. Apart from this, our approach copes with the attribute granularity problem and the information coverage problem.

We will show that even a small number of coreferent tuples is sufficient to establish a correct matching between corresponding attributes of heterogeneous schemas. Such

methods can then later be used to improve the coreference detection of data described by schema which are considered as metadata of content data.

1.3 Outline

The remainder of this paper is organized as follows. In Sect. 2, an extensive overview of work related to the topic of this paper is provided. Next, in Sect. 3, some preliminary concepts are introduced that serve as a theoretical foundation of this paper. In Sect. 4.2, an overview of our novel content data-based schema matching algorithm is presented. In Sects. 5 and 6, the details of the algorithm are studied. In Sect. 7 an experimental study of the proposed methods and techniques is reported. Finally, Sect. 8 summarizes the most important contributions of this paper.

2 Related Work

Schema matching can be established by using different methods. Some methods use only data (e.g., duplicates [1, 10]), others use only metadata (e.g., schema information [17, 25–27], knowledge base [24]), whereas other methods use both data and metadata, e.g., [8, 9]. In this paper the content data-based schema matching approach is proposed which uses coreferent tuples. There is a large body of work on schema matching which uses content data [21]; for example, LSD [10] extracts information from a training set and consists of a learning and classification phase. More specifically, given a user-supplied mapping between schema elements, the learning step looks at content data to train the classifier, thereby discovering characteristic content data patterns and matching rules. Next, these patterns and rules can be applied to match other schema elements. Moreover, the approach in [18] captures valuable knowledge about the domain of the attribute. This approach uses regular expressions as a formalism to characterize a set of attribute values. Having these expressions, the corresponding attributes are detected by matching the regular expression of one attribute with the value of another attribute using the match function. In many cases it is still not clear which attributes correspond to each other. Thus, regular expressions are a valuable and useful tool but should be supported by other techniques. As opposed to most instance-based solutions which use summary information (e.g., average value) for attribute classification, we derive schema matching from detected coreferent tuples in the datasets. One schema matching approach using duplicates is ILA [19], which is a domain-independent program that learns the meaning of external information by explaining it in terms of internal categories. ILA considers a pair of objects as duplicates if both objects contain at least one attribute value in common and relies on a high extensional overlap (a number of coreferent tuples). In our opinion these assumptions are unrealistic.

IMap [8] is based on both schema and instance information as well as on a domain ontology and uses past matchings. The duplicates are identified by the user and only exact matches of attribute values are considered by a matcher. IMap copes with various attribute granularities, as in Chua et al. [6] and Lu et al. [16], but the focus is on numerical and differently scaled data (as opposed to our approach which focuses on textual data). Statistical analysis is employed to data in duplicates which are assumed to be identified by a common ID attribute. This means that at least one attribute is already aligned. The approach of Chua et al. [6] classifies attributes into domain classes (e.g., categorical) and forms attribute groups (sets of attributes from the same relation which may correspond one to another) based on predefined rules. Then, the correspondence scores of pairs of attribute groups are calculated. Finally, attributes are matched based on these scores. The approach of Lu et al. [16], one the other hand, uses correlation analysis techniques (supervised by the user) to identify attributes which are potentially semantically related; secondly, they apply regression analysis to generate the relevant conversion function that allows the attribute values of one database to be transformed into attribute values of the other database.

DUMAS [1], just as in our approach, drops several of the assumptions that were made in the above works: coreferent tuples are automatically detected using unaligned schemas; a few coreferent tuples being sufficient to establish schema matching (low extensional overlap). Moreover, it does not use any external source of information, such as an ontology; it is a content data-based approach. In contrast to our approach, DUMAS does not apply possibility theory and does not combine vertical and horizontal schema matching methods to detect coreferent tuples and establish schema matching.

3 Preliminaries

Before we present the details of elaborated algorithms we will first introduce some relevant basic concepts. We start with the multiset definition. Then, we more formally define the problem of coreference detection, which is the main problem addressed in this paper, and the cardinality of a set of pairs of coreferent objects.

3.1 Necessity Measure

In possibility theory, the certainty concerning the statement that the value of X is in A, denoted *Necessity*(X is A), is expressed by *the necessity measure* $N_X(A)$, defined with respect to a possibility measure $\Pi_X(A)$ as follows:

$$\textit{Necessity}(X \text{ is } A) \triangleq N_X(A) \triangleq 1 - \Pi_X(\overline{A}) \tag{1}$$

with $\overline{A}$ denoting the complement of the fuzzy set A [30].

3.2 Multisets

Within the context of this work, the framework of set theory (which is the basis of the relational model) will not suffice to present our approach. Instead, the more general framework of multisets (also called bags) will be used where necessary and the definitions by Yager [29] are adopted here. A multiset A over a universe U is defined by a function $A : U \to \mathbb{N}$. For each $u \in U$, $A(u)$ denotes the multiplicity (i.e., the number of occurrences) of u in A. The set of all multisets drawn from a universe U is denoted $\mathcal{M}(U)$. Yager has defined some basic operations on multisets. The j-cut of a multiset A is a regular set, denoted as A_j and is given by $A_j = \{u | u \in U \wedge A(u) \geq j\}$. Counterparts of classical set intersection, union operations and of the notion of subsethood are defined as follows:

$$\forall u \in U : (A \cup B)\,(u) = \max\,(A(u), B(u)) \tag{2}$$

$$\forall u \in U : (A \cap B)\,(u) = \min\,(A(u), B(u)) \tag{3}$$

$$A \subset B \equiv \forall u \in U : A(u) < B(u) \tag{4}$$

$$A \subseteq B \equiv \forall u \in U : A(u) \leq B(u). \tag{5}$$

The theory of multisets provides also an addition operator and a subtraction operator:

$$\forall u \in U : (A \oplus B)\,(u) = A(u) + B(u) \tag{6}$$

$$\forall u \in U : (A \ominus B)\,(u) = \max\,(A(u) - B(u), 0)\,. \tag{7}$$

The cardinality of a multiset A is calculated as the sum of all multiplicities:

$$|A| = \sum_{u \in U} A(u). \tag{8}$$

Finally, it is said that an element u belongs to the multiset A, denoted as $u \in A$, if $A(u) \geq 1$.

3.3 Object Coreference Detection

Considering a more abstract view of entity representation, denoting the universe of the ith feature of an object by U_i, we can model the universe O of objects by:

$$O = U_1 \times \cdots \times U_n. \tag{9}$$

Two objects $o_1 \in O$ and $o_2 \in O$ are said to be *coreferent* (denoted $o_1 \leftrightarrow o_2$) if and only if they describe the same real world entity.

Two elementary operators play an important role in establishing the coreference of objects: a *comparison operator* working at the level of object features (or metadata features, e.g., tags, paths) and an *aggregation operator* combining the comparison scores obtained for particular features.

Definition 1 (*Comparison operator*) A comparison operator on the universe O is defined by a function $\mathcal{C}$:

$$\mathcal{C} : O^2 \rightarrow \mathbb{L} \tag{10}$$

where $(\mathbb{L}, \leq)$ is a totally ordered and bounded lattice.

A comparison operator $\mathcal{C}$ compares (a feature of) two objects o_1 and o_2 and expresses the result of this comparison as a *matching degree*. This matching degree may be interpreted as expressing how certain it is that both objects are coreferent and belongs to a totally ordered and bounded lattice $\mathbb{L}$. In the case of probabilistic methods, $\mathbb{L}$ can be instantiated with the unit interval [0, 1] in order to express the result of comparison as a probability of coreference of the objects. Other practical examples of $\mathbb{L}$ are the set of truth values $\mathbb{B} = \{T, F\}$, where T and F denote full certainty of the match and mismatch, respectively or the set of possibilistic truth values (PTVs) [2].

In our approach we use PTVs to express the confidence (certainty) in the validity of the mappings produced by an algorithm. Hereby, a PTV is a normalized possibility distribution [30] defined over the set of Boolean values $\mathbb{B}$ [20]:

$$\text{PTV} \longmapsto \pi : \mathbb{B} \rightarrow [0, 1]$$

A PTV expresses the uncertainty about the Boolean value of a *proposition* p. We will often use the notation $\mu(T)$ and $\mu(F)$ instead of $\pi(T)$ and $\pi(F)$ assuming that the (un)certainty as to the truth of a proposition is expressed as "certainly true", "true or false" etc., represented by appropriate fuzzy sets in $\mathbb{B}$; e.g., respectively, $\mu(T) = 1$ and $\mu(F) = 0$, and $\mu(T) = \mu(F) = 1$, for the previous examples. In the context considered here, the propositions p of interest are of the form:

$$p \equiv o_1 \text{ and } o_2 \text{ are coreferent}$$

where o_1 and o_2 are two objects.

Let P denote a set of all propositions under consideration. Then each $p \in P$ can be associated with a PTV denoted $\tilde{p} = \big\{(T, \mu_{\tilde{p}}(T)), (F, \mu_{\tilde{p}}(F))\big\}$, where $\mu_{\tilde{p}}(T)$ represents the possibility that p is true and $\mu_{\tilde{p}}(F)$ denotes the possibility that p is false. In what follows, PTVs are often noted in couple notation as $(\mu_{\tilde{p}}(T), \mu_{\tilde{p}}(F))$. It is assumed that each PTV is normalized, which means that $\max(\mu_{\tilde{p}}(T), \mu_{\tilde{p}}(F)) = 1$. The domain of all possibilistic truth values is denoted $\mathcal{F}(\mathbb{B})$, i.e., is the fuzzy power set of (normalised) fuzzy sets over $\mathbb{B}$.

Let us define the order relation $\geq$ on the set $\mathcal{F}(\mathbb{B})$ by:

$$\tilde{p} \geq \tilde{q} \Longleftrightarrow \text{if}((\mu_{\tilde{p}}(F) \leq \mu_{\tilde{q}}(F)) \text{ and } (\mu_{\tilde{p}}(T) = \mu_{\tilde{q}}(T) = 1)) \text{ or } (\mu_{\tilde{q}}(T) \leq \mu_{\tilde{p}}(T)) \tag{11}$$

Moreover, two thresholds ($threshold_T$ and $threshold_F$) are employed to decide on object coreference. If $\mu_{\tilde{p}}(F)$ is lower than the $threshold_F$, then coreference is declared. If $\mu_{\tilde{p}}(T)$ is lower than the $threshold_T$, then a lack of coreference is declared. Finally, if both of the thresholds are exceeded then the coreference status is declared as being *unknown*.

Comparison of complex objects is usually a two-stage process. First, parts of objects, notably values of their features, are compared using a comparison operator. Thus, we extend our definition of the comparison operator (10) so as to make it applicable also to scalar feature values:

$$\mathcal{C}_i : U_i^2 \to \mathbb{L} \tag{12}$$

In this way a separate comparison operator $\mathcal{C}_i$ can be defined for each feature. Then, the results of those comparisons are aggregated to obtain an overall matching degree reflecting the coreference of the whole objects being compared. Therefore, another elementary operator, an *aggregation operator*, is needed.

Definition 2 (*Aggregation operator*) An aggregation operator on $\mathbb{L}$ is defined by a function $\mathcal{A}$:

$$\mathcal{A} : \mathbb{L}^n \to \mathbb{L} \tag{13}$$

where $(\mathbb{L}, \leq)$ is a totally ordered and bounded lattice.

For more information on aggregation operators the reader is referred to [5]. We assume an aggregation operator $\mathcal{A}$ to be idempotent:

$$\forall\, l \in \mathbb{L} : \mathcal{A}(l, l, \ldots, l) = l \tag{14}$$

Besides that, we assume that $\mathcal{A}$ is monotone in the following sense:

$$\forall\, (\boldsymbol{l}, \boldsymbol{l}') \in \mathbb{L}^n \times \mathbb{L}^n : \boldsymbol{l} \leq \boldsymbol{l}' \Rightarrow \mathcal{A}(\boldsymbol{l}) \leq \mathcal{A}(\boldsymbol{l}') \tag{15}$$

where the relation $\leq$ is generalized from $\mathbb{L}$ to vectors from $\mathbb{L}^n$ in a point wise way.

Based on the definition of these two elementary operators, a comparison of two objects can be generally written as:

$$\mathcal{C}(o_1, o_2) = \underset{i=1}{\overset{n}{\mathcal{A}}} \left(\mathcal{C}_i \left(u_{i1}, u_{i2}\right)\right) \tag{16}$$

where u_{i1} and u_{i2} denote the value of the ith feature of o_1 and o_2, respectively.

In our approach $\mathbb{L}$ is the space of all PTVs endowed with the relation given in (11). Aggregation of PTVs may be carried out using the *Sugeno integral for possibilistic truth values* as defined in [3]; cf. also [23] for the original, general definition of the Sugeno integral. This integral uses two *fuzzy measures* (γ^T and γ^F) which are defined below. Let us first remind briefly the definition of a *fuzzy measure*.

Definition 3 (*Fuzzy measure*) A *fuzzy measure* on a finite universe U is a set function $\gamma : \mathcal{P}(U) \to [0, 1]$ that satisfies the following properties:

$$\gamma(\emptyset) = 0 \tag{17}$$

$$\gamma(U) = 1 \tag{18}$$

$$A \subseteq B \Rightarrow \gamma(A) \leq \gamma(B) \tag{19}$$

Then, in the context considered here, the measure $\gamma^T(A)$ (resp. $\gamma^F(A)$) provides the assessment of certainty that two complex objects are (not) coreferent, given that the set of (metadata) features A are (not) coreferent. As required by the definition of fuzzy measures, γ^T and γ^F are monotonic and satisfy the boundary conditions of a fuzzy measure.

Definition 4 (*Sugeno integral for PTVs* [3]) Given a set of propositions $P = \{p_1, \ldots, p_n\}$ and a corresponding set of PTVs $\tilde{P} = \{\tilde{p}_1, \ldots, \tilde{p}_n\}$, let γ^T and γ^F be two fuzzy measures defined on P which satisfy the condition:

$$\forall Q \subseteq P : \min(\gamma^T(Q), \gamma^F(\bar{Q})) = 0 \tag{20}$$

where $\bar{Q}$ denotes the complement of Q.

Then the Sugeno integral of $\tilde{P}$ with respect to γ^T and γ^F is defined by:

$$S_{\gamma^{T,F}}(\tilde{P}) : \mathcal{F}(\mathbb{B})^n \to \mathcal{F}(\mathbb{B}) : \tilde{P} \mapsto \tilde{p}, \text{ where} \tag{21}$$

$$\mu_{\tilde{p}}(T) = 1 - \bigvee_{i=1}^{n} \mathrm{N}_{\tilde{p}_{(i)}}\left(F\right) \wedge \gamma^F\left(P_{(i)^F}\right) \tag{22}$$

and

$$\mu_{\tilde{p}}(F) = 1 - \bigvee_{i=1}^{n} \mathrm{N}_{\tilde{p}_{(i)}}\left(T\right) \wedge \gamma^T\left(P_{(i)^T}\right) \tag{23}$$

where $(\cdot)^T$ (respectively $(\cdot)^F$) is a permutation that orders the elements of $\tilde{P}$ non-increasingly (non-decreasingly), while $P_{(i)^F}$ and $P_{(i)^T}$ are sets of propositions p_j with, respectively, i largest values $\mu_{\tilde{p}_j}(F)$ and i largest values $\mu_{\tilde{p}_j}(T)$.

Remark. The motivation to use PTVs and the Sugeno integral is the following. We would like to show that taking into account similarity and dissimilarity of objects in each step separately may be advantageous. In fact, De Cooman [7] has shown in his formal analysis of PTVs that it is essential that possibilities for true and false can be measured separately. To this aim, the aggregated PTVs indicate both the coreference and the lack of coreference of paths/schemas. The choice for the Sugeno integral is motivated by the ability of the related fuzzy measures to model complex preferences in the regular case, making the Sugeno integral a very powerful and flexible aggregation operator [3]. The research on the aggregation of *bipolar* information (here: for and against the coreference) is not that developed in the literature and the Sugeno integral is a prominent example of an aggregation operator adopted for this setting. Besides that, the experimental results confirm that this is a promising choice. An alternative aggregator can be, e.g., an Ordered Weighted Conjunction (OWC) [2] which is in fact a special case of the Sugeno integral.

3.4 Cardinality of a Set of Pairs of Coreferent Objects

In this paper we will often use (multi)sets of Boolean propositions, certainty of truth of which will be expressed with a PTV associated with each proposition. We will then use the concept of a kind of the cardinality of such a (multi)set which counts those propositions fully certain to be true (i.e., with a PTV (1, 0) assigned) as 1, does not count at all propositions fully certain to be false (i.e., with a PTV (0, 1) assigned), and counts the remaining propositions to some degree belonging to [0, 1] and depending on how their PTVs are close to (1, 0) or (0, 1). In fact, this cardinality is similar to a fuzzy cardinality of fuzzy sets and will be expressed as a possibility distribution on the set of integers. We will denote this cardinality as $\pi_{\mathbb{N}}$ (provided that from the context it will be clear which set of propositions it concerns). We will call it also sometimes as a *fuzzy integer* due to the fact that its possibility distribution is assumed to be a convex function, in the same sense as membership functions of fuzzy numbers are assumed, i.e., every α-cut of this function (interpreted as a membership function of a fuzzy set) is an interval, i.e., contains all integers between the lowest and highest integers belonging to this α-cut.

In [12], a method is proposed to construct such a possibility distribution (fuzzy integer) for a (multi)set of propositions associated with PTVs $\tilde{P}$. In fact, this method may be treated as constructing a possibility distribution which expresses the possibility that an integer k represents the number of true propositions in P.

Definition 5 (*Cardinality of a set of PTV qualified propositions*) Let P be a multiset of independent Boolean propositions and let $\tilde{P}$ be the multiset of corresponding possibilistic truth values, i.e., $\forall p \in P : \tilde{p}$ is the PTV associated with p and expressing the (un)certainty as to the truth of p and let $\tilde{p}_{(i)}$ denote the ith largest possibilistic truth value with respect to the order relation defined by Eq. 11. The quantity of true

propositions in P is given by the following possibility distribution on the set of all integers (fuzzy integer):

$$\pi_{\mathbb{N}}(k) = \begin{cases} \mu_{\tilde{p}_{(1)}}(F), & k = 0 \\ \mu_{\tilde{p}_{(k)}}(T), & k = |P| \\ \min\left(\mu_{\tilde{p}_{(k)}}(T), \mu_{\tilde{p}_{(k+1)}}(F)\right), & \text{else.} \end{cases} \tag{24}$$

This definition states that $\pi_{\mathbb{N}}(k)$ is the minimum of the possibility that at least k propositions are true and the possibility that at least $|P| - k$ propositions are false.

Let us define an order relation $\prec_{sup}$ on the set of such possibility distributions (fuzzy integers).

Definition 6 (*Sup-order of fuzzy integers*) For two fuzzy integers, $\tilde{n}$ and $\tilde{m}$, the order relation $\prec_{sup}$ is defined as:

$$\tilde{n} \prec_{sup} \tilde{m} \Leftrightarrow \sup \tilde{n}_\alpha < \sup \tilde{m}_\alpha \tag{25}$$

Hereby, $\tilde{n}_\alpha$ is the α-cut of $\tilde{n}$, which is treated here as a fuzzy set, and α is chosen such that:

$$\alpha = \sup\{x | \sup \tilde{n}_x \neq \sup \tilde{m}_x\} \tag{26}$$

4 Content Data-Based Schema Matching

Before we continue to describe our method for schema matching, first of all we should define the problem more formally.

4.1 Problem Definition

Within the scope of this paper it is assumed that entities from the real world are described as objects (tuples) which are characterised by a number of *attributes* (features). A *schema* $\mathcal{R}$ of a given dataset, which consists of tuples, is identified by a set of attributes A. For each attribute $a \in A$, let $dom(a)$ denote the domain of a (the set of possible values for attribute a) and let $dom'(a)$ denote the subset of $dom(a)$ comprising the values of a that are actually present in the (tuples of the) dataset.

Two datasets are considered. The source dataset over the schema $\mathcal{R}_S$ with the set of attributes $A_S = \{a_1^S, \ldots, a_n^S\}$ is denoted as S, while the target dataset over the schema $\mathcal{R}_T$ with the set of attributes $A_T = \{a_1^T, \ldots, a_m^T\}$ is denoted as T. The one-to-many schema matching is defined as follows.

Definition 7 (*One-to-many schema matching*) A relation M is a schema matching if:

$$M \subseteq 2^{A_S} \times 2^{A_T} \times \tilde{M} \tag{27}$$

where $M = \{m_i\} = \{(A'_S, A'_T, \tilde{m})\}$, $A'_S \subseteq A_S$, $A'_T \subseteq A_T$ and A'_S or A'_T is a singleton set, $\tilde{M}$ is the set of PTVs and $\tilde{m} \in \tilde{M}$ expresses the certainty degree to which A'_S matches A'_T.

Some additional properties may be associated with each matching $m \in M$. For example, the *local matching cardinality*, denoted $card^l_m$, is the number of matched attributes in m, i.e., $card^l_m = |A'_S| + |A'_T|$ (e.g., $card^l_m$ is equal to 2 for a one-to-one matching). Moreover, particular matchings may be classified to a *type*. The following matching types are distinguished:

- *full matching* (coverage level 1): corresponding attributes have the same meaning and cover completely the same concept, e.g., "Name" and "POI" or "Type" and "Category" in Fig. 1;
- *inclusion matching* (coverage level 0.5): corresponding attributes have partially the same meaning and do not cover completely the same concept, e.g., "Address" in the source S in Fig. 1 represents the address of a POI which consists of a street, house number, city and zip code, and this is a part of the concatenation of the attributes "Street", "City" and "ZipCode" in the target T in Fig. 1, which consists of the same information as address from the source but is extended by a country code. "Street" in the target T in Fig. 1 represents only a part of the address from the source. Thus, two sub-types of matching are considered: the *source is a part of the target* and the *target is a part of the source*, respectively;
- *has a common part matching* (coverage level 0.3): corresponding attributes have partially the same meaning, do not cover completely the same concept and are not an inclusion matching. E.g., the matching between "Address" in the source S and "ZipCode" in the target T in Fig. 1. "Address" represents the address of a POI which consists of a street name, house number, city and zip code without the country code; while "ZipCode" in the target T in Fig. 1 represents the zip code and country code of a POI, thus only the zip code is a common part.
- *unknown* (coverage level 0): if attributes do not match.
- *concatenation*. This is a special case of attribute matching which combines two or more attributes. Combining matching types might result in another matching type. For instance, a combination of two inclusion matchings may give a full matching (of attributes) or an inclusion matching. E.g., let assume inclusion matchings between attributes from the source S and the target T in Fig. 1: "Address" and "Street"; "Address" and "City"; "Address" and "ZipCode". Concatenation of these matchings gives a full matching.

The matching $m \in M$ can be interpreted as a one-to-one matching of corresponding attributes if the cardinalities of A'_T and A'_S are equal to 1 or as a one-to-many matching of corresponding attributes if the cardinalities of A'_T or A'_S are greater than 1.

Furthermore, in the context of a one-to-many schema matching M, we consider a set D of coreferent tuple pairs which is defined as follows.

Definition 8 (*A set D of coreferent tuple pairs*) A set D of coreferent tuple pairs consists of 4-tuples $d = (t^S, t^T, M_V, \tilde{d})$ where t^S and t^T are coreferent tuples from the source and target datasets, respectively, M_V is a set of attributes matchings for which there are coreferent values in both particular tuples t^S and t^T, and $\tilde{d}$ is a PTV representing the (un)certainty that two tuples $t^S \in S$ and $t^T \in T$ are coreferent.

4.2 Algorithm

The novel content data-based schema matching Algorithm 1 creates matchings between corresponding attributes A_S and A_T of the source dataset S and the target dataset T, respectively, using content data. Therefore, the inputs for the algorithm are the source and target datasets (S and T, respectively), and a set of parameters (P_V and P_H for each phase) which are used to establish a schema matching. The objective of our algorithm is to establish as many valid one-to-one or one-to-many schema matchings M for coreferent attributes as possible.

Algorithm 1 SCHEMAMATCHINGALGORITHM

Require: Dataset S, Dataset T, Parameters P_V, Parameters P_H
Ensure: Schema Matching M
1: $M_V \leftarrow$ getVerticalMatchings($\{dom'(a^S)\}_{a^S \in A_S}$, $\{dom'(a^T)\}_{a^T \in A_T}$,$P_V$)
2: $M \leftarrow$ getHorizontalMatchings(S,T,M_V,P_H)

The Algorithm 1 is composed of two main phases. First, vertical schema matchings are established by the method getVerticalMatchings which compares the domains of particular attributes (line 1 in Algorithm 1, which is further discussed in Sect. 5). Second, the established vertical matchings M_V are used to detect coreferent tuple pairs in the heterogeneous data sources which, in turn, constitute a basis to generate horizontal schema matchings M by using the method getHorizontalMatchings (line 2 in Algorithm 1, which is further discussed in Sect. 6). These steps are described in detail in the following sections.

5 Phase I: Vertical Matching

The first phase of our novel schema matching approach is the generation of one-to-one and one-to-many vertical matchings between corresponding attributes. These matchings are established by Algorithm 2 based on statistical analysis and lexical

comparison of attribute domains. Thus the input for the algorithm are the subsets of the domains consisting of these values that actually occur in tuples of respective datasets, $\{dom'(a^S)\}_{a^S \in A_S}$ and $\{dom'(a^T)\}_{a^T \in A_T}$, and also a set P_V of parameters which define the thresholds and submatcher settings and is detailed further on. This phase consists of three steps.

In the first step, "Statistical analysis of content data", the subsets of the attribute domains are statistically compared by the *statistical matcher*, and if particular subsets are coreferent, then the matching between their corresponding attributes is established (lines 2–18 in Algorithm 2); otherwise the attribute domains are lexically compared in the second step called "Overlapping" by the *lexical matcher* (lines 19–25 in Algorithm 2). More specifically, each pair of attributes is processed sequentially by the following techniques. First, the results of the statistical analysis of the subsets of the attribute domains (such as the analysing the average length, average values, called attribute properties) are compared, which is a relatively computationally non-expensive statistical technique. Second, only if coreference between two attributes is not declared then the intersection of the subsets of their domains, which are represented by multisets of terms, is calculated based on the equality relation, i.e., two terms are added to the intersection if they are equal. Thus, two attributes are considered as coreferent if a cardinality of the intersection exceeds threshold. Third, if coreference between the attributes is still not declared, then the subsets of their domains are calculated analogously to the second technique but based on the low-level string comparison technique [2] instead of the equality relation. This is the most computationally expensive method of the three, but it is also the most valuable because non-equal but coreferent terms can be detected. The established matchings are added to the set $M_V^{1:1}$ of the one-to-one schema matchings. Next, in the third step, called "Generalization", from the established one-to-one schema matchings in $M_V^{1:1}$, a one-to-many schema matching ($\in M_V^{1:n}$) is generated (line 28 in Algorithm 2). Finally, the vertical schema matching M_V is composed of the one-to-one schema matchings $M_V^{1:1}$ and the one-to-many schema matchings $M_V^{1:n}$ (line 29 in Algorithm 2). These steps are described in detail in the following subsections.

5.1 Step 1: Statistical Analysis of Content Data

In the first step the attribute domains of each schema are statistically analysed separately using predefined *Data Analysers* $P_V.AN$ (lines 2–8 in Algorithm 2). This returns a set of *properties* for each attribute which are considered as a basis for some heuristics for determining the coreference of attributes. There is a large body of work of such properties and heuristics [13–15, 22]. Thus, we give only some examples of such properties and also give an example of an application. These aspects are subject to further research and outside the scope of the work. The proposed examples of such heuristics are the following:

Algorithm 2 VERTICALMATCHINGALGORITHM

Require: $\{dom'(a^S)\}_{a^S \in A_S}$, $\{dom'(a^T)\}_{a^T \in A_T}$, Parameters P_V
Ensure: Schema Matching M_V
1: Schema Matching $M_V^{1:1} \leftarrow$ null
2: Properties $P_S[]$, $P_T[]$
3: **for all** $a^S \in A_S$ **do**
4: $P_S[a^S] \leftarrow$ getProperties($dom'(a^S)$,$P_V.AN$)
5: **end for**
6: **for all** $a^T \in A_T$ **do**
7: $P_T[a^T] \leftarrow$ getProperties($dom'(a^T)$,$P_V.AN$)
8: **end for**
9: **for all** $a^S \in A_S$ **do**
10: **for all** $a^T \in A_T$ **do**
11: Matching $m \leftarrow$ null
12: **if** compareStats($P_S[a^S]$,$P_T[a^T]$) $> P_V.thrStats$ **then**
13: $m.A'_S \leftarrow a^S$
14: $m.A'_T \leftarrow a^T$
15: $m.\pi_{\mathbb{N}} \leftarrow \pi^1_{\mathbb{N}}$
16: $M_V^{1:1} \leftarrow M_V^{1:1} \cup m$
17: **continue**
18: **end if**
19: $m.\pi_{\mathbb{N}} \leftarrow$ compareDom($dom'(a^S)$,$dom'(a^T)$,P_V)
20: FuzzyInteger $\pi^{thr}_{(dom^S,dom^T)} \leftarrow$ getThr($P_V.thrOverlap$)
21: **if** $\pi^{thr}_{(dom^S,dom^T)} \prec_{sup} m.\pi_{\mathbb{N}}$ **then**
22: $m.A'_S \leftarrow m.A'_S \cup a^S$
23: $m.A'_T \leftarrow m.A'_T \cup a^T$
24: $M_V^{1:1} \leftarrow M_V^{1:1} \cup m$
25: **end if**
26: **end for**
27: **end for**
28: Schema Matching $M_V^{1:n} \leftarrow$ getGeneralization($M_V^{1:1}$)
29: $M_V \leftarrow M_V^{1:1} \cup M_V^{1:n}$

- average, minimum and maximum length as a number of characters in a value without white spaces (numbers are considered as character strings, e.g., telephone numbers, bank accounts, etc.);
- average, minimum and maximum number of tokens for alphabetic and alphanumerical data types. Each value is tokenised, which results in a set of substrings which are called tokens. In most cases, the tokens are separate words. In our approach, the tokenisation of a value is equivalent to subdividing the value in a multiset of tokens and deleting all white spaces in a value;
- average, minimum and maximum value for numerical data types;
- data type: numerical (values contain only numbers), alphabetic (values contain only letters and special characters), alphanumerical (values contain any characters);

Next, attributes a^S and a^T that have similar properties are considered as potentially coreferent (candidate attributes, line 12 in Algorithm 2) and the established matching m between them is added to the set of matchings $M_V^{1:1}$ (lines 13–16 in Algorithm 2, similarity for all properties is assumed here). The statistical criteria are very strict, thus this matching is assigned a full certainty which is expressed by the fuzzy integer $\pi_{\mathbb{N}}^1$ that $\forall x \in \mathbb{N}\, \pi_{\mathbb{N}}^1(x) = 1$. The basis to decide if properties are similar is the similarity function. We use a simple function that calculates the similarity of properties for particular attributes as a normalised difference of property values. The returned values are within the unit interval [0, 1], where 1 means strong similarity and 0 means a complete lack of similarity. Properties with a similarity above threshold $P_V.thrStats$ are considered to be similar. The similarity function is defined by Eq. (28) and is applied for all properties, except for data type property which is considered similar only if compared data types are the same.

$$\text{simProp}(a^S, a^T) = 1 - \frac{|\text{propVal}(a^S) - \text{propVal}(a^T)|}{|\text{propVal}(a^S)| + |\text{propVal}(a^T)|} \tag{28}$$

Hereby $a^S \in A_S$ and $a^T \in A_T$, and *propVal* is a method which gets the value of a particular property, e.g., the maximum length of the values for an attribute a^S (or a^T).

Remark. Information from the schema, e.g., maximum value, etc., is not considered because it might be too general and may mislead the matching algorithm. For instance, let assume a database of students with an attribute "Age" of type INTEGER in the range of -2^{31} to $2^{31} - 1$. This statistical analysis of the values of the attribute "Age" which are actually present in the database may return a range of 20–29. This information can be more useful than data type restriction which is defined in the database schema.

Example Let us consider the attributes from the source dataset in Table 1 and the target dataset in Table 2. The calculated properties of the subsets of the attribute domains from these datasets are presented in Tables 3 and 4, respectively. Next, the similarities between these properties are calculated by Eq. 28, e.g., for the attributes $a^S =$ "Lon" and $a^T =$ "Geo2" we obtain:

$$\begin{aligned}
&\text{MinLength:} && \text{simProp}(a^S, a^T) = 1 - \tfrac{|8-8|}{|8|+|8|} = 1 \\
&\text{MaxLength:} && \text{simProp}(a^S, a^T) = 1 - \tfrac{|8-8|}{|8|+|8|} = 1 \\
&\text{AvgLength:} && \text{simProp}(a^S, a^T) = 1 - \tfrac{|8-8|}{|8|+|8|} = 1 \\
&\text{MinValue:} && \text{simProp}(a^S, a^T) = 1 - \tfrac{|3.657992-3.657975|}{|3.657992|+|3.657975|} = 0.999998 \\
&\text{MaxValue:} && \text{simProp}(a^S, a^T) = 1 - \tfrac{|4.050876-3.722015|}{|4.050876|+|3.722015|} = 0.957691 \\
&\text{AvgValue:} && \text{simProp}(a^S, a^T) = 1 - \tfrac{|3.756594-3.701370|}{|3.756594|+|3.701370|} = 0.992595
\end{aligned} \tag{29}$$

Assuming that the threshold $P_V.thrStats$ is equal to 0.8, these attributes are considered as being coreferent because the similarity of all properties of these attribute domains

Table 3 Properties of attribute domains from the source dataset in Table 1

Property	Name	Lon.	Lat.	Category	Address
Min length	10	8	9	5	27
Max length	23	8	9	15	44
Avg length	14.86	8	9	8	31.36
Min #tokens	2	–	–	1	3
Max #tokens	4	–	–	2	5
Avg #tokens	2.38	–	–	1.13	3.86
Min value	–	3.657992	50.984194	–	–
Max value	–	4.050876	51.281777	–	–
Avg value	–	3.756594	51.064419	–	–
Data type	str	num	num	str	str-num

Num means numerical datatype, *str* means alphabetic data type, and *str-num* means alphanumeric datatype

Table 4 Properties of attribute domains from the target dataset Table 2

Property	POI	Geo1	Geo2	Type	Street	City	ZipCode
Min length	12	9	8	5	15	4	7
Max length	21	9	8	10	20	17	7
Avg length	16.17	9	8	8.83	18.17	7.17	7
Min #tokens	1	–	–	1	1	1	2
Max #tokens	3	–	–	2	3	1	2
Avg #tokens	2.16	–	–	1.17	2.16	1	2
Min value	–	51.018938	3.657975	–	–	–	–
Max value	–	51.056465	3.722015	–	–	–	–
Avg value	–	51.044901	3.701370	–	–	–	–
Data type	str	num	num	str	str-num	str	str-num

Num means numerical datatype, *str* means alphabetic data type, and *str-num* means alphanumeric datatype

exceeds 0.8. The same holds for the attribute pair “Lat” and “Geo1”. Thus, these two attribute pairs determine two one-to-one matchings which are added to the set $M_V^{1:1}$ of one-to-one matchings. However, the other attribute pairs are not coreferent based on the statistical information, therefore they are further processed in the next step of our algorithm.

5.2 Step 2: Overlapping

The attributes a^S and a^T, whose statistical properties are not similar enough, are considered in this step. More specifically, a lexical comparison of the subsets $dom'(a^S)$ and $dom'(a^T)$ of the attribute domains is conducted using soft strings comparison. For that purpose, the method compareDom is used which works as follows (lines 19–25 in Algorithm 2).

First, special characters that appear in the values of $dom'(a^S)$ and $dom'(a^T)$, i.e., dash, semicolon, dot, etc., are replaced by a space character, which results in strings of terms separated by space. This way, each attribute is described by a multiset of obtained terms (W_{a^S} and W_{a^T}, respectively). Next, the intersection I of these multisets is calculated according to formula (3). Thus, multiset I contains the common terms of W_{a^S} and W_{a^T}, which are assigned a PTV (1, 0) and are the basis for further checking whether the particular attributes a^S and a^T are coreferent or not. Namely, these associated PTVs multiplied by the term multiplicity form a multiset $\tilde{P}$ which is used to construct a possibility distribution $\pi_{\mathbb{N}}$ (a fuzzy integer) introduced in Definition 5.

The fuzzy integer $\pi_{\mathbb{N}}$ of intersection I reflects the possibility that two attribute domains are coreferent. Hence, if $\pi_{\mathbb{N}}$ is greater than the threshold $\pi^{thr}_{(dom^S, dom^T)}$ with respect to the order relation of Definition 6, then the attributes $a_S \in A_S$ and $a_T \in A_T$ are considered to be potentially coreferent (candidate attributes, line 21 in Algorithm 2), and the established matching m between them is added to the set $M^{1:1}_V$ of matchings (lines 22–24 in Algorithm 2). The threshold $\pi^{thr}_{(dom^S, dom^T)}$ is a fuzzy integer and is dynamically calculated by the method getThr (lines 20 in Algorithm 2). This threshold depends on the particular attribute domains and the predefined parameter $P_V.thrOverlap$, which specifies the percentage of domain terms that overlap. More specifically, $\pi^{thr}_{(dom^S, dom^T)}$ is constructed from n PTVs (1, 0), where n is calculated by the following equation:

$$n = \lfloor \min(|W_{a^S}|, |W_{a^T}|) \times P_V.thrOverlap \rfloor \tag{30}$$

However, if a fuzzy integer $m.\pi_{\mathbb{N}}$ of matching is not larger than the threshold $\pi^{thr}_{(dom^S, dom^T)}$, then the domains of the considered attribute a^S and a^T are analogously compared again, but the equalness relation, which decides on the coreference of the terms, is replaced by the low-level string comparison method proposed in [2]. This low-level comparison method estimates the possibility that two given terms (strings) are coreferent or not and is based on an approximation of *weak string intersections* which is the set of longest common subsequences. It uses the concept of a moving window to construct the intersection of the two input strings. More specifically, the algorithm starts at the beginnings of both strings of a pair and moves a window over each of them. Each time common characters are detected under the moving windows they are added to the intersection, which is the largest set (in terms of set cardinality) that is a subset of both strings.

For example, consider a pair of strings $s_1 = tracks$ and $s_2 = tracklist$. The construction of the intersection goes then as follows. We start with two one-character

wide windows. Initially each window is at the beginning of a respective string and contains a character *'t'*. This character is common so it is added to the intersection and both windows move to their next position. Similarly for *'r'*, *'a'*, *'c'* and *'k'*. In the next step the windows contain different characters, *'s'* and *'l'*, respectively. Thus, the window size is increased by one. This is repeated until the windows contain a common character, here *'s'*, or there are no more characters in both of the strings. Next, the common character is added to the intersection, windows are shrunk to one character and moved to the position where the common character was found increased by 1. This construction of the intersection is repeated until the windows reach the ends of strings. Finally, the resulting intersection is *'tracks'*. The non common characters are counted (considered as errors) and decrease possibility that steps are coreferent.

This method marks out four different types of errors during comparison. These are *prefix*, *suffix*, *gap* and *mismatch*. The *prefix* is an error where one of the input strings contains a prefix before the matched substring, for instance a letter *'d'* is a prefix for ***d****title* and *title*. Analogously for *suffix*. The *gap* consists of missing characters in the middle of a string, for instance *'li'* is a gap and *'t'* is a suffix for strings *track****list*** and *tracks*. Finally, a *mismatch* is an unmatched character in both strings. These errors have different importance and influence on the final matching result. Because of that, the importance of each error type is expressed by predefined weights between 0 and 1 and are problem dependent. The higher the weight of an error the lower degree of matching of two strings for which such an error occurs. In our case, where abbreviations are very popular, the crucial error types are *prefix* and *mismatch* so their weights are set to 1, *gap* has a weight 0.3 and *suffix* 0.1 is the minor error.

Our algorithm then compares pairs of strings from the domains of the considered attribute a^S and a^T. It generates PTVs which express the uncertainty about the coreference of the compared strings as described above. The possibility that a proposition p, stating that two strings are coreferent, is true ($\mu_{\tilde{p}}(T)$) and the possibility that p is false ($\mu_{\tilde{p}}(F)$) are calculated by the following equations:

$$\mu_{\tilde{p}}(T) = \frac{possT}{factor} \tag{31}$$

$$\mu_{\tilde{p}}(F) = \frac{possF}{factor} \tag{32}$$

where *possT*, *possF* and *factor* equal:

$$possT = \frac{|intersection|}{\max(s_1.length, s_2.length)} \tag{33}$$

$$possF = \sum_{i=0}^{|errors|} (errors_i.size \times w_i) \tag{34}$$

$$factor = \max(possT, possF) \tag{35}$$

where $|intersection|$ denotes the number of common characters, an $|errors|$ is the number of types of the errors, $errors_i.size$ is the number of the errors of a given type.

On the one hand, $possT$ is the ratio between the number of characters that are found to be common for a pair of strings (cardinality of the intersection) and the length of the longer string. On the other hand, $possF$ is computed as the sum of the product of the number of the errors of a given type (from errors that are found during comparison) and predefined weight w_i of specific error type. Finally, $factor$ is the maximum of $possT$ and $possF$ and is used to normalize both possibilities.

Two terms are considered as coreferent if $\mu_{\tilde{p}}(F)$ of the resulting PTV is lower than $\mu_{\tilde{p}}(F)$ of the predefined threshold $P_V.\tilde{thr}$ (see Sect. 3.3).

This terms comparison method thus takes into account misspellings and abbreviations and, moreover, has a low computational complexity. This technique was chosen due to its efficiency [2]. In the literature a multitude of algorithms for string comparison has been proposed and these may also be employed here. An example of an interesting survey concerning strings in general is [11]. An example of an approach employing fuzzy logic which might also be of interest to the reader is [31].

Example Let us consider the attribute $a^S =$ "Address" from the source dataset in Table 1 and the attribute $a^T =$ "City" from the target dataset in Table 2. Let us assume that these attributes have not been indicated as coreferent based on the analysis carried out in Step 1. After preprocessing the subset $dom'(a^S)$ of the attribute domain, the resulting multiset W_{a^S} contains the terms: "Sint-Baafsplein" (multiplicity 2), "9000" (3), "Ghent" (4), "Hoegaarden" (1), "St-Denijs-Westrem" (1), "Gentstraat" (1), etc. Whereas, the multiset W_{a^T} of the subset $dom'(a^T)$ of the attribute domain consists of the terms: "Gent" (4), "Hoegaarden" (1) and "St-Denijs-Westrem" (1). Next, the intersection I of the multisets W_{a^S} and W_{a^T} is calculated which contains two terms, "Hoegaarden" and "St-Denijs-Westrem", both with a multiplicity equal to 1 (the multiplicity of an element in the intersection of two multisets is the minimum of the multiplicities of that element in both multisets, see Sect. 3.2). Both returns are given an associated PTV (1, 0). Thus, the fuzzy integer of this intersection is constructed from the multiset of PTVs {(1,0); (1,0)} by Eq. 24.

Figure 2 (the top left-most graph) shows the multiset of possibilistic truth values, where a *circle* denotes the possibility of T and a *triangle* denotes the possibility of F. The derived possibility distribution $\pi_{\mathbb{N}}$ (the fuzzy integer) is shown below the possibilistic truth values. The middle graph of Fig. 2, in turn, shows the multiset of PTVs which are used to construct the threshold $\pi^{thr}_{(dom^S, dom^T)}$ which depends on the particular attribute domains and the predefined parameter $P_V.thrOverlap$ and is shown in the graph below the multiset of PTVs. Following the specification, the multiset of PTVs which is used to construct the threshold $\pi^{thr}_{(dom^S, dom^T)}$ consists of n PTVs equal to (1, 0), where n is calculated by Eq. 30 with $P_V.thrOverlap = 0.35$ and equals to $n = \lfloor \min(33, 6) \times 0.35 \rfloor = 2$. The fuzzy integer $\pi_{\mathbb{N}}$ is not larger than $\pi^{thr}_{(dom^S, dom^T)}$ (w.r.t. Definition 6), because the 1-cuts of both fuzzy integers have the same supremum equal 2. So, the domains of the considered attributes a^S and a^T are

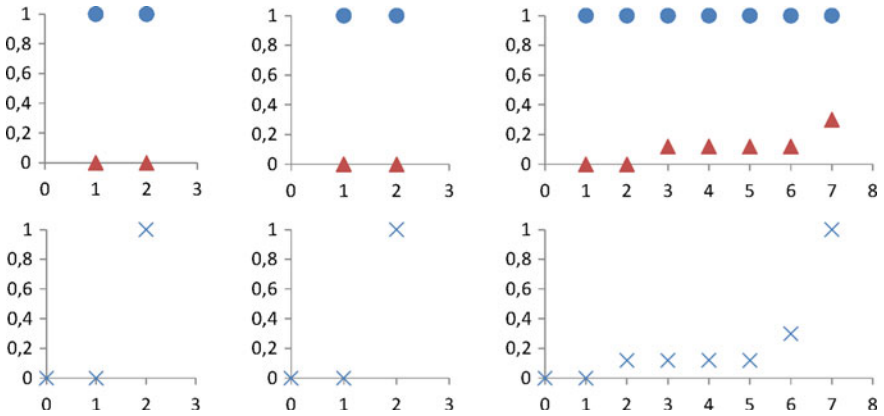

Fig. 2 Fuzzy integers derived from the possibilistic truth values of the attribute matching ("Address"; "City") based on the equality relation, the threshold $\pi^{thr}_{(dom^S, dom^T)}$ and the attribute matching ("Address"; "City") based on the low-level string comparison method

next compared using the low-level string comparison method with $\mu_{\tilde{p}}(F) = 0.5$ as the predefined threshold $P_V.\tilde{thr}$. That comparison returns an intersection I, which consists of the following elements: ("Gentstraat", "Gent") with an associated PTV (1, 0.3) and multiplicity 1; ("Ghent", "Gent"), (1, 0.12), 4; ("St-Denijs-Westrem", "St-Denijs-Westrem"), (1, 0), 1; and ("Hoegaarden", "Hoegaarden"), (1, 0), 1. Figure 2 (the right-most top graph) shows the multiset of PTVs {(1,0); (1,0); (1,0.12); (1,0.12); (1,0.12); (1,0.12); (1,0.3)}, which are used by Eq. 24 to construct a fuzzy integer $\pi_{\mathbb{N}}$ and is shown below the PTVs in Fig. 2. Now, it turns out that, the fuzzy integer $\pi_{\mathbb{N}}$ is larger than the threshold $\pi^{thr}_{(dom^S, dom^T)}$ (w.r.t. Definition 6), because the 1-cut of the right-most fuzzy integer has a higher supremum, equal 7, than the middle fuzzy integer threshold, which has the supremum 2. This is the same fuzzy integer threshold as above because it depends on the same particular attribute domains—we consider the same attributes. Thus, the attributes "Address" and "City" are considered as being potentially coreferent and the established matching m between them is added to the set $M_V^{1:1}$ of matchings.

5.3 Step 3: Generalization

The last step of the vertical matching phase derives a one-to-many schema matching $M_V^{1:n}$ by using the method called getGeneralization based on one-to-one schema matching $M_V^{1:1}$ (line 28 in Algorithm 2). Afterwards, the vertical schema matching M_V is composed of the schema matching $M_V^{1:n}$ and $M_V^{1:1}$ (line 29 in Algorithm 2). The method getGeneralization is implemented by the Algorithm 3 which works as follows.

The input schema matching $M_V^{1:1}$, which is generated in steps 1 and 2 (Sects. 5.1 and 5.2, respectively), is the basis to generate a set $M'_{1:n}$ of one-to-many matchings by combining a number of one-to-one matchings which have the same attribute from A_S or A_T, i.e., a one-to-many matching has either the form: (line 1 in Algorithm 3):

$$\left(A, a_j^T\right), \text{ where } A \subseteq A_S \wedge |A| \geq 2 \wedge \forall a_i^S \in A \;\; \exists \left(a_i^S, a_j^T\right) \in M_V^{1:1} \tag{36}$$

or,

$$\left(a_i^S, A\right), \text{ where } A \subseteq A_T \wedge |A| \geq 2 \wedge \forall a_j^T \in A \;\; \exists \left(a_i^S, a_j^T\right) \in M_V^{1:1} \tag{37}$$

Afterwards, for each matching $m_{1:n} \in M'_{1:n}$, the *extended* domains are compared by the compareDom method (line 3 in Algorithm 3). This is done analogously as in the "Overlapping" step of Sect. 5.2. An extended domain is constructed by the method getDom and contains concatenated values of all attributes that are specified in the parameters, i.e., of all attributes forming the set A in $m_{1:n}$ (cf. (36) and (37)). The values are concatenated one by one and separated with a white space into a new value which belongs to the extended domain.

Next, *alternative* matchings $M_{alt}^{1:1} \subseteq M_V^{1:1}$ are selected by the method getAlternatives (line 4 in Algorithm 3). An alternative matching $m_{1:1} \in M_{alt}^{1:1}$ for $m_{1:n}$ should have at least one attribute in common with the matching $m_{1:n} \in M'_{1:n}$, i.e., (a_k^S, a_l^T) is an alternative matching with respect to (A, a_j^T) if $a_k^S \in A$ or $a_l^T = a_j^T$, and similarly for (a_i^S, A). Finally, if the fuzzy integer $\pi_{\mathbb{N}}$ of the one-to-many matching $m_{1:n}$ is larger (w.r.t. Definition 6) than the fuzzy integer of any alternative matching $m_{1:1} \in M_{alt}^{1:1}$ (line 6 in Algorithm 3), then the matching $m_{1:n}$ is added to the schema matching $M_V^{1:n}$ (line 7 in Algorithm 3).

Algorithm 3 GENERALIZATIONALGORITHM

Require: Schema Matching $M_V^{1:1}$
Ensure: Schema Matching $M_V^{1:n}$
1: Schema Matching $M'_{1:n} \leftarrow$ getCombination($M_V^{1:1}$)
2: **for all** Matching $m_{1:n} \in M'_{1:n}$ **do**
3: $m_{1:n}.\pi_{\mathbb{N}} \leftarrow$ compareDom(getDom($m_{1:n}.A'_S$),getDom($m_{1:n}.A'_T$))
4: Schema Matching $M_{alt}^{1:1} \leftarrow$ getAlternatives($M_V^{1:1}$,$m_{1:n}$)
5: **for all** Matching $m_{1:1} \in M_{alt}^{1:1}$ **do**
6: **if** $m_{1:1}.\pi_{\mathbb{N}} \prec_{sup} m_{1:n}.\pi_{\mathbb{N}}$ **then**
7: $M_V^{1:n} \leftarrow M_V^{1:n} \cup m_{1:n}$
8: **break**
9: **end if**
10: **end for**
11: **end for**

Remark. This generalization is specified for alphanumerical data, where numerical data are considered as character data. For numerical data more sophisticated concatenation method (such as aggregation or transformation function, e.g., a function

Table 5 Example of the vertical schema matching $M_V^{1:1}$

Matching m	Source	Target
1	Name	Type
2	Name	POI
3	Lon.	Geo2
4	Lat.	Geo1
5	Category	Type
6	Key	Id
7	Address	ZipCode
8	Address	Street
9	Address	City

which calculates average value) of values of all attributes that are specified in the matching parameters, i.e., of all attributes forming the set A in $m_{1:n}$ (cf. (36) and (37)), is required and it is out of the scope of this paper.

Example Let us consider the one-to-many matching $m_{1:n} \in M'_{1:n}$ as a combination of the one-to-one matching $M_V^{1:1}$ in Table 5. Namely, $m_{1:n}$ establishes a matching of the attribute $a^S =$ "Address" ($a^S \in A_S$) from the source dataset in Table 1 and the attributes $A'_T =$ {"Street", "City", "ZipCode"} from the target dataset in Table 2 ($A'_T \subseteq A_T$). This matching is derived as a combination of the one-to-one (candidate, alternative) matchings 7, 8 and 9 of Table 5.

First, the extended domains are constructed by using the method getDom. These domains contain the values of all attributes forming the set A in a one-to-many matching; cf. (36) and (37). The extended domain $dom_{ext}(A'_S) = dom(a^S) = dom$("Address") contains the values: "Sint-Baafsplein, 9000 Ghent", "Grotestraat 91, 7471 BL Goor", "Jan Breydelstraat 35, 9000, Ghent", etc. The extended domain $dom_{ext}(A'_T) = dom_{ext}$({"Street", "City", "ZipCode"}) contains the concatenated values: "Emile Braunplein Gent 9000 BE, "Sint-Baafsplein Gent 9000 BE", "Jan Breydelstraat 35 Gent 9000 BE", etc. Both extended domains $dom_{ext}(A'_S)$ and $dom_{ext}(A'_T)$ are compared by the compareDom method just as in the "Overlapping" step in Sect. 5.2. More specifically, after preprocessing the attribute domains, the multiset $W_{A'_S} = W_{a^S}$ contains the terms: "Sint-Baafsplein" (multiplicity 2), "9000" (3), "Ghent" (4), "Hoegaarden" (1), "St-Denijs-Westrem" (1), "Gentstraat" (1), etc. The multiset $W_{A'_T}$ contains the terms: "BE" (6), "Gent" (4), "9000" (4) "Hoegaarden" (1), "St-Denijs-Westrem" (1), "Sint-Baafsplein" (1), etc. Next, the intersection I of these multisets is determined which contains the terms: "Kleine" with associated PTV (1, 0) and multiplicity 1; "Gentstraat", (1, 0), (1); "24", (1, 0), (1); "5", (1, 0), (1); "69", (1, 0), (1); "3320", (1, 0), (1); "9051", (1, 0), (1); "Schouwburgstraat", (1, 0), (1); "9000", (1, 0), (4); "Stoopkensstraat", (1, 0), (1); "Sint-Baafsplein", (1, 0), (1); "Jan", (1, 0), (1), "St-Denijs-Westrem", (1, 0), (1); "Hoegaarden", (1, 0), (1). All associated PTVs are (1, 0) because all the compared terms are equal. Next, the fuzzy

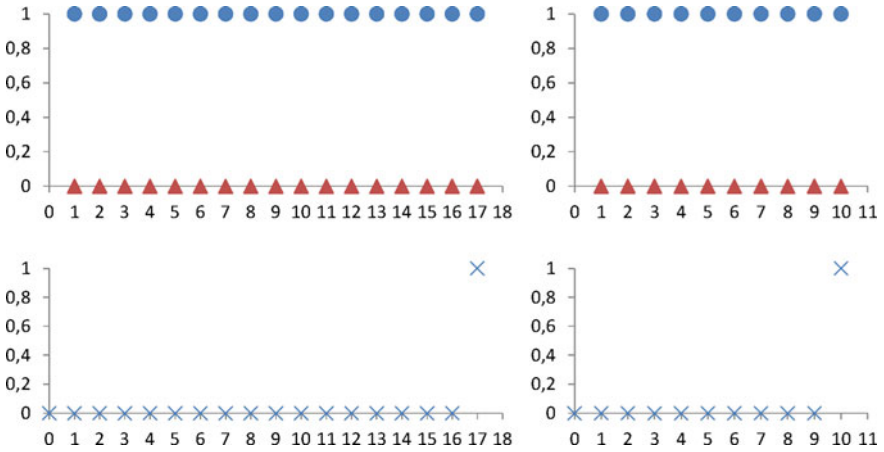

Fig. 3 Fuzzy integers derived from possibilistic truth values of the attribute matching ("Address"; "Street", "City", "ZipCode")

integer expressing the cardinality of the multiset of matching terms is constructed from the resulting multiset of PTVs multiplied by the term multiplicity by Eq. 24.

Figure 3 shows the set of possibilistic truth values, where a *circle* denotes the possibility of T and *triangle* denotes the possibility of F. The derived possibility distribution $\pi_{\mathbb{N}}$ (the fuzzy integer) is shown below the possibilistic truth values.

Next, *alternative* matchings $M_{alt}^{1:1} \subseteq M_V^{1:1}$ are selected by the method getAlternatives, i.e., one-to-one matchings that have at least one attribute in common with the constructed one-to-many matching $m_{1:n}$. Namely, ("Address"; "ZipCode"), ("Address"; "City"), and ("Address"; "Street"). Finally, it turns out that the fuzzy

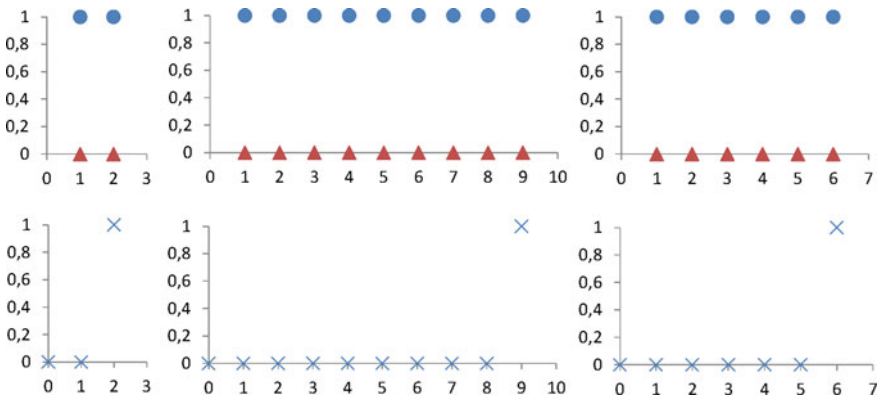

Fig. 4 Fuzzy integers derived from possibilistic truth values of the alternative attribute matchings for the attribute "Address": ("Address"; "City"), ("Address"; "Street") and ("Address"; "ZipCode")

integer $\pi_{\mathbb{N}}$ related to the one-to-many matching $m_{1:n}$ is larger than the fuzzy integer (w.r.t. Definition 6) related to each alternative matching in Fig. 4, because the 1-cut of the fuzzy integer resulting from concatenation $m_{1:n}$ has a higher supremum (17 in Fig. 3) than the fuzzy integers of the alternative matchings (2, 9, 6, respectively in Fig. 4). So, the matching $m_{1:n}$ is added to the schema matching $M_V^{1:n}$.

Finally, the union of sets of matchings $M_V^{1:1}$ and $M_V^{1:n}$ forms the final set of candidate matchings M_V, which is the basis to detect coreferent tuples and to establish the final matching between corresponding attributes in the next phase.

6 Phase II: Horizontal Matching

In this phase, the candidate vertical schema matching M_V from the previous phase is used to establish the horizontal schema matching M by the method getHorizontalMatchings (line 2 in Algorithm 1). More specifically, in step 1 of this phase the vertical schema matching M_V is used to efficiently detect coreferent tuples across heterogeneous data sources (Sect. 6.1). This significantly reduces the number of comparisons and the complexity of the approach and in turn is the basis for generating the final schema matching in the second step of this phase (Sect. 6.2). The following subsections describe both steps of the horizontal matching phase.

6.1 Step 1: Coreferent Tuples Detection

The coreferent tuples detection Algorithm 4 for schema matching searches for the n-most coreferent tuple pairs D across tuples in the source dataset S and the target dataset T using the candidate vertical schema matching M_V as follows. First, each tuple from the source is compared with each tuple from the target. More precisely, the values of the corresponding attributes, which are matched by M_V, are compared by the method compareTuples (line 3 in Algorithm 4) which inter alia calculates the possibility that two given tuples are coreferent (expressed by a PTV denoted as $\tilde{d}$) and returns a pair of coreferent tuples d. The details of this comparison are presented in Algorithm 5 and described in the next Paragraph "Tuples comparison". Next, coreferent tuple pair d is added to the set D of coreferent tuple pairs (line 4 in Algorithm 4). Finally, the detected coreferent tuple pairs $d \in D$ are sorted by $\tilde{d}$ using Eq. 11 and the $P_H.n$ most coreferent tuple pairs are the result of this algorithm (line 7 and 8 in Algorithm 4, respectively). Using a fixed threshold on the matching degree would be unreasonable because the (un)certainty of tuples coreference varies along with the number of corresponding attributes [1], i.e., if only a few attributes are truly coreferent then the certainty will be low. Thus, instead, our method ranks coreferent tuple pairs by their PTVs and gets the n-most coreferent tuple pairs. It has to be clear that the goal is not to detect all coreferent tuples. These coreferent tuple

pairs serve as the basis to establish horizontal schema matching, what is discussed in the next Sect. 6.2.

Algorithm 4 CoreferentTupleDetectionAlgorithm

Require: Dataset S, Dataset T, Schema Matching M_V, Parameters P_H
Ensure: n-most Coreferent Tuple Pairs D
1: **for all** Tuple $t^S \in S$ **do**
2: **for all** Tuple $t^T \in T$ **do**
3: Coreferent tuple pair $d \leftarrow$ compareTuples(t^S,t^T,M_V,P_H)
4: $D \leftarrow D \cup d$
5: **end for**
6: **end for**
7: $D \leftarrow$ sort(D)
8: $D \leftarrow$ getMostCoreferent($P_H.n$,D)

Tuples Comparison A comparison of two tuples is conducted by Algorithm 5 and works as follows. First, the input tuples t^S and t^T from the source dataset and the target dataset, respectively, form a pair of candidate coreferent tuples ($d.t^S$, $d.t^T$) (line 1 and 2 in Algorithm 5, respectively). Second, a comparison of attribute values for each matching $m \in M_V$ computed in the previous step works as follows. An initial matching $d.m$ is initialized as a copy of a matching m (line 4 in Algorithm 5). Next, for each attribute(s) $m.A'_S$ ($m.A'_T$) of a candidate matching $m \in M_V$ the value(s) v^S or v^T from tuples $d.t^S$ and $d.t^T$ are respectively extracted (lines 5 and 6 in Algorithm 5). In case of 1:n matching of attributes (see Sect. 5.3), the extracted values can be vectors of values $v^S[]$ or $v^T[]$ whose coordinates are concatenated into v^S or v^T before they are compared. Afterwards, the extracted values v^S and v^T are compared using a data type-specific method which estimates the possibility $d.\tilde{m}$ that two given values are coreferent (line 7 in Algorithm 5). More precisely, a numerical and an alphanumerical matchers are considered as follows.

Numerical matcher. Numerical values are compared by a method which is based on the difference (*diff*) of the considered values and a difference threshold ($P_H.thrDiff$). The difference threshold is a real number which defines the maximum allowed difference of values, and depends on the range of values of a particular attribute and the predefined parameter $P_H.thrNum$, which specifies the percentage of difference for average range of the considered attributes $m.A'_S$ and $m.A'_T$ (1:n attribute matching does not apply for numerical data, thus, $m.A'_S = a^S$ and $m.A'_T = a^T$). More specifically, $P_H.thrDiff$ is calculated by the following equation:

$$P_H.thrDiff = \left\lfloor \frac{\text{range}(dom'(a^S)) + \text{range}(dom'(a^T))}{2} \times P_H.thrNum \right\rfloor \quad (38)$$

where range is a difference between the maximum and minimum value of $dom'(a^S)$ or $dom'(a^T)$ from the source or the target. If the difference *diff* between values is smaller than the difference threshold $P_H.thrDiff$, then the possibility that a propo-

sition p stating that the two values are coreferent is true ($\mu_{\tilde{p}}(T)$) equals 1, and the possibility that p is false ($\mu_{\tilde{p}}(F)$) is a fraction of the difference *diff* and the difference threshold $P_H.thrDiff$; otherwise, a lack of coreference is declared, i.e., $\mu_{\tilde{p}}(T) = 0$ and $\mu_{\tilde{p}}(F) = 1$.

Alphanumerical matcher. Alphanumerical values are transformed into sets of substrings which are in most cases separate words. The string is split at the position of a white space, comma, dot or other special character. The usefulness of this approach follows from the fact that character-based methods are typically not well suited for longer strings. Next, the substrings are compared with one another by the low-level string comparison method [2] (see Sect. 5.2). This gives PTVs which are aggregated by the Sugeno integral [3, 4, 23]. Aggregation results in a single PTV which reflects the possibility that two given values are coreferent (see Sect. 3.3). More specifically, the aggregation operator for the comparison of two values, v^S and v^T, is defined by the Sugeno integral for PTVs, where:

- $P = \{p_i\}$ is a set of propositions stating coreference of pairs of substrings,
- $\tilde{P}$ is the set of selected PTVs corresponding to the above-mentioned propositions representing the uncertainty about their truth values computed by the low-level string comparison method,
- the fuzzy measure γ^T is defined by:

$$\gamma^T(Q) = \sum_{j=1}^{k} w_j, \qquad Q \subseteq P, Q = \{p_1, \ldots, p_k\} \tag{39}$$

where w_j is the weight of the jth pair (s_S^j, s_T^j) of substrings, computed by:

$$w_j = \frac{1}{|P|} \tag{40}$$

- the fuzzy measure γ^F is defined by

$$\gamma^F(Q) = \begin{cases} 1 \text{ if } Q = P \\ 0 \text{ otherwise} \end{cases} \tag{41}$$

what is implied by condition (20) is that for each Q which is a subset of P.

Moreover, pairs of substrings with the selected PTVs are grouped into two sets. The first set contains substrings which have an associated PTV that is larger than $P_H.\tilde{thr}$ w.r.t. Eq. 11, and are called coreferent tokens. The second set contains substrings which have an associated PTV that is not larger than $P_H.\tilde{thr}$ and are called non-coreferent tokens. These two sets are the basis for deriving the type of matching. The type *type* of matching $d.m$ (see Sect. 4.1) is specified based on the number of coreferent and non-coreferent tokens (substrings) of the values v^S and v^T by a method which is presented in the Sect. "Matching type of tuples" (line 8 in Algorithm 5).

Next, the matching $d.m$ is added to the set $d.M_V$ of matchings for the coreferent tuples pair d.

Finally, the set $d.M_V$ contains matchings $d.m$ of coreferent attributes A'_S and A'_T for the particular tuples $d.t^S$ and $d.t^T$. Each matching $d.m \in d.M_V$ has an associated PTV ($d.\tilde{m}$), which expresses the possibility that attributes A'_S and A'_T are coreferent for the particular tuples $d.t^S$ and $d.t^T$ based on their values, and is the basis for further checking whether the particular tuples $d.t^S$ and $d.t^T$ are coreferent or not. Namely, these associated PTVs form a multiset $d.\tilde{M}_V$ and are aggregated by the Sugeno integral [3, 4, 23]. The aggregation returns a single PTV $\tilde{d}$ which reflects the possibility that two given tuples are coreferent (line 11 in Algorithm 5). More specifically, the aggregation operator for the comparison of two tuples, $d.t^S$ and $d.t^T$, is defined by the Sugeno integral for PTVs, where:

- $P = \{p_i\}$ is a set of propositions stating coreference of attributes A'_S and A'_T for the particular tuples $d.t^S$ and $d.t^T$, represented by $d.M_V = \{d.m_i\}$,
- $\tilde{P}$ is the set of PTVs corresponding to the above-mentioned propositions, represented by $d.\tilde{M}_V$,
- the fuzzy measure γ^T is defined by:

$$\gamma^T(Q) = \sum_{j=1}^{k} w_j, \qquad Q \subseteq P, Q = \{p_1, \ldots, p_k\} \tag{42}$$

where w_j is the weight of the jth pair (A'_S, A'_T) of attributes for the particular tuples, computed by:

$$w_j = \forall d.\tilde{m} \in d.\tilde{M}_V : w_{d.\tilde{m}} = \frac{1}{|d.M_V|}. \tag{43}$$

These weights are equal for each PTV and depend on the number of matchings.

- the fuzzy measure γ^F is defined by

$$\gamma^F(Q) = \begin{cases} 1 \text{ if } Q = P \\ 0 \text{ otherwise} \end{cases} \tag{44}$$

what is implied by condition (20) is that for each Q which is a subset of P.

Matching Type of Tuples The matching types, which are defined in Sect. 4.1, depend on the factors *ratioS* and *ratioT*. These factors represent the completeness of each matching $d.m \in d.M_V$ for the particular tuples t^S and t^T and are based on the number of coreferent tokens (substrings) of compared values $v^S \in t^S$ and $v^T \in t^T$. The factor *ratioS* (*ratioT*) is the fraction of the number of coreferent tokens over the number of tokens of the value v^S (v^T, respectively). Thus, the following conditions have to be considered:

- if $ratioS = 0$ and $ratioT = 0$ then $m \in M$ is an "unknown" matching
- if $ratioS = 1$ and $ratioT = 1$ then $m \in M$ is a "full" matching

Algorithm 5 COMPARETUPLESALGORITHM

Require: Tuple t^S, Tuple t^T , Schema Matching M_V, Parameters P_H
Ensure: Coreferent tuple pair d
1: $d.t^S \leftarrow t^S$
2: $d.t^T \leftarrow t^T$
3: **for all** Matching $m \in M_V$ **do**
4: Matching $d.m \leftarrow m$
5: var $v^S \leftarrow d.t^S[m.A'_S]$
6: var $v^T \leftarrow d.t^T[m.A'_T]$
7: $d.\tilde{m} \leftarrow$ compare(v^S, v^T, $P_H.\tilde{thr}$, $P_H.thrNum$)
8: $d.m.type \leftarrow$ mType(v^S, v^T)
9: $d.M_V \leftarrow d.M_V \cup d.m$
10: **end for**
11: $\tilde{d} \leftarrow$ aggregate($d.\tilde{M}_V$)

- if $ratioS = 1$ and $ratioT \neq 1$ then $m \in M$ is a "source is part of target" matching
- if $ratioS \neq 1$ and $ratioT = 1$ then $m \in M$ is a "target is part of source" matching
- if $ratioS \neq 1$ and $ratioT \neq 1$ then $m \in M$ is a "a common part" matching

Example Let us consider coreferent tuple pairs detection for the following case. The input is the vertical matching M_V of Table 6, which is the basis for detecting coreferent tuple pairs across the source and target datasets, of Tables 1 and 2, respectively. Each tuple from the source dataset is compared to each tuple in the target dataset which results in the set D of coreferent tuple pairs. For example, let us consider that tuple t^S

Table 6 Example of the vertical schema matching M_V

Matching m	Source	Target
m_1	Name	Type
m_2	Name	POI
m_3	Name	Type, POI
m_4	Name, Category	Type
m_5	Lon.	Geo2
m_6	Lat.	Geo1
m_7	Category	Type
m_8	Key	Id
m_9	Address	ZipCode
m_{10}	Address	Street
m_{11}	Address	City
m_{12}	Address	ZipCode, Street
m_{13}	Address	ZipCode, City
m_{14}	Address	Street, City
m_{15}	Address	ZipCode, Street, City

in row 2 in Table 1 is compared to tuple t^T in row 2 in Table 2. First, the values of the matched attributes (by M_V) are compared in sequence, e.g., the value "Saint Bavo" of the attribute "Name" in t^S is compared by the alphanumerical matcher to the following values in t^T w.r.t. the matchings m_1, m_2 and m_3 in Table 6: "Church" (of the attribute "Type"), "Sint-Bavokerk" (of the attribute "POI") and "Church Sint-Bavokerk" (of the concatenation of the attributes "Type" and "POI"). This comparison results in $d.M_V$ which contains all the attributes matchings $m \in M_V$, namely $d.m \in d.M_V$ (with associated PTVs $d.\tilde{m}$ which form a multiset $d.\tilde{M}_V$) for the particular tuples t^S and t^T, e.g., $d.\tilde{m}_1 = (0, 1)$, $d.\tilde{m}_2 = (1, 0.12)$, $d.\tilde{m}_3 = (1, 0.16)$, etc. Next, the matching type for each $d.m \in d.M_V$ for particular tuples is derived: $d.m_1.type$ is an "unknown" matching (no common tokens), $d.m_2.type$ is a "full" matching (all tokens are common), $d.m_3.type$ is a "source is part of target" matching (all tokens from the source are common, but not all from the target), etc. Next, the PTVs in $d.\tilde{M}_V$ are aggregated by the Sugeno integral with equal weights $w = 1/15$ (calculated by Eq. 43). This results in a single PTV $\tilde{d}$ that equals $(1, 0.5)$ which reflects the possibility that the two given tuples t^S and t^T are coreferent.

Finally, the detected coreferent tuple pairs D are sorted by $\tilde{d}$, and the $P_H.n$ (in our case 3) most coreferent tuple pairs are returned. Namely, the pairs of tuples from rows 2, 4 and 5 of Tables 1 and 2 are returned as the 3 most coreferent tuple pairs and are used to establish the final schema matching, what is discussed in the next section.

6.2 Step 2: Schema Matching

The n most coreferent tuples pairs of D detected in the previous step and the vertical schema matching M_V established in the first phase of our approach are used to infer the final horizontal schema matching M. Our novel approach is implemented by Algorithm 6, which works as follows. First of all, for each matching $m \in M_V$, the cardinality $card_m$ of this matching is calculated (lines 2–6 in Algorithm 6). This cardinality is the number of coreferent tuple pairs whose values of attributes $m.A'_S$ and $m.A'_T$ are coreferent. Hereby coreference is considered if $\mu_{\tilde{p}}(F)$ of $d.\tilde{m}$ is lower than $\mu_{\tilde{p}}(F)$ of the predefined threshold $P_H.thr\tilde{D}up = (0.5, 0.5)$ w.r.t. Eq. 11.

Next, if a particular matching m returns most of the coreferent tuple pairs (line 7 in Algorithm 6), i.e., $m.card_m/|D|$ is greater than the predefined threshold $P_H.thrMajority$, then m is added to the final schema matching M (line 8 in Algorithm 6). Next, the propositions evaluated using PTVs of the matching m across all coreferent tuple pairs of D (i.e., $\forall d \in D : d.\tilde{m}$) are aggregated by the Sugeno integral. This results in a possibility degree $\tilde{m}$ (PTV), which expresses the uncertainty of that matching $m \in M$ (line 9 in Algorithm 6) [3, 4, 23] (see Sect. 3.3).

More specifically, the aggregation operator for matching $m \in M$ is defined by the Sugeno integral for PTVs, where:

- $P = \{p_i\}$ is a set of propositions stating coreference of attributes $m.A'_S$ and $m.A'_T$ across all coreferent tuple pairs of D, represented by $D[m] = \{d_i.m\}$,
- $\tilde{P}$ is the set of PTVs corresponding to the above-mentioned propositions, represented by $\tilde{D}[m] = \{d_i.\tilde{m}\}$,
- the fuzzy measure γ^T is defined by:

$$\gamma^T(Q) = \sum_{j=1}^{k} w_j, \qquad Q \subseteq P, Q = \{p_1, \ldots, p_k\} \tag{45}$$

where w_j is the weight of the jth duplicate d_j, computed by:

$$w_j = \frac{1}{|D|}. \tag{46}$$

These weights are equal for each PTV $d_j.\tilde{m}$ and depend on the number of coreferent tuple pairs.

- the fuzzy measure γ^F is defined by

$$\gamma^F(Q) = \begin{cases} 1 \text{ if } Q = P \\ 0 \text{ otherwise} \end{cases} \tag{47}$$

what is implied by condition (20) is that for each Q which is a subset of P.

Moreover, a possibility degree $\tilde{m}$ of that matching $m \in M$ can be used to resolve schema matching *conflicts*. The schema matching M contains conflicts if there exists more than one matching $m \in M$ (called alternative matching) for any attribute $a^S \in A_S$ or $a^T \in A_T$. This means that a matching which is donated with the larger PTV than alternative matchings is preferable and another alternative matchings should be removed. However, the conflict resolution is subject to further research and outside the scope of this paper.

Finally, the matching types of the matching $m \in M$ across all coreferent tuple pairs D are unified by the method unifyType in line 10 in Algorithm 6. This method returns for each $m \in M$ the most popular matching type across all coreferent tuple pairs D. In the case of indistinguishable matching types, i.e., if maximum frequency of matching type for matching $m \in M_V$ over all coreferent tuple pairs of D is not unique, the matching type with the predefined lowest coverage level is selected (see Sect. 4.1). For example, if full matching (with coverage level 1) and inclusion matching (with coverage level 0.5) types are specified for the same number of coreferent tuple pairs then inclusion matching type is selected. The unified matching types inform about matching completeness and can be used to resolve schema matching conflicts but it is out of the scope of this paper.

Example Let us consider the coreferent tuple pairs in D which were detected in the previous step. The set D of coreferent tuples pairs consists of 3 pairs, each composed of rows 2, 4, 5 from Tables 1 and 2. The matching cardinality $card_m$ is calculated for each matching $m \in M_V$ in Table 6. For example, the $card_m$ of the matching

Algorithm 6 HORIZONTALMATCHINGALGORITHM

Require: Coreferent tuple pairs D, Schema Matching M_V, Parameters P_H
Ensure: Schema Matching M
1: **for all** Matching $m \in M_V$ **do**
2: **for all** Coreferent tuple pair $d \in D$ **do**
3: **if** $d.\tilde{m} > P_H.thr\tilde{D}up$ **then**
4: $card_m \leftarrow card_m + 1$
5: **end if**
6: **end for**
7: **if** $card_m/|D| > P_H.thrMajority$ **then**
8: $M \leftarrow M \cup m$
9: $\tilde{m} \leftarrow$ aggregate($\tilde{D}[m]$)
10: $m.type \leftarrow$ unifyType($D[m].type$)
11: **end if**
12: **end for**

$m_{10} = \{$"Address"; "Street"$\}$ equals 2 because only the attribute values of two tuple pairs are coreferent for the threshold $P_H.thr\tilde{D}up$ equal to (0.5, 0.5). More specifically, the value "Stoopkensstraat 24, 3320 Hoegaarden" is similar to "Stoopkensstraat 24", and "Kleine Gentstraat 69, 9051 St-Denijs-Westrem" is similar to "Kleine Gentstraat 69". The certainty as to their similarity is expressed for both of them by a PTV (1, 0.33). The similarity of values "Sint-Baafsplein, 9000 Ghent" and "Sint-Baafsplein" is expressed by a PTV (1, 0.5), but this does not exceed the threshold. In contrast, $card_m$ of the matching $m_{15} = \{$"Address"; "Street", "City", "ZipCode"$\}$ is equal to 3, because the attribute values of all coreferent tuple pairs are coreferent. This confirms that the concatenation of attributes makes sense.

Next, if $card_m$ of m satisfies the majority condition in line 7 of Algorithm 6, then the matching m is added to M. The predefined threshold $P_H.thrMajority = 0.3$ and $|D| = 3$, thus if $card_m$ of m is greater than 0.9, then m is considered as a matching in M. This means that M contains only matchings which are confirmed by at least one coreferent tuple pair. Matchings m_4, m_9 and m_{11} in Table 6 are not included in M because they do not satisfy this condition. Next, the PTVs for matching m across all coreferent tuple pairs D are aggregated by the Sugeno integral. For the matching m_{10}, the PTVs (1, 0.33), (1, 0.5), (1, 0.33) are aggregated with equal weights $w = 1/|D| = 1/3$ to a single PTV equal to (1, 0.33). This PTV reflects the possibility that the attributes "Address" and "Street" are coreferent. Finally, the matching types of the matching $m \in M$ across all coreferent tuple pairs D are unified by the method unifyType. For the matching m_{10}, the unified matching type is "t is part of s" (target is part of source) because the matching type of all considered coreferent tuple pairs is "t is part of s".

This gives us the schema matching M in Table 7 with alternative matchings (conflicts) for some of the attributes, e.g., m_{10}–m_{15} in Table 7. These conflicts can be resolved based on cardinality ($card_m$), certainty ($\tilde{m}$) and/or type of matching but it is out of the scope of this paper.

Table 7 Example of the schema matching M with alternative matchings

Matching m	Source	Target	$card_m$	$type_m$	$\tilde{m}$
m_1	Name	Type	1	t is part of s	(1, 0.67)
m_2	Name	POI	1	Has a common part	(1, 0.5)
m_3	Name	Type, POI	1	Has a common part	(1, 0.6)
m_5	Lon	Geo2	2	Full matching	(1, 0.33)
m_6	Lat	Geo1	2	Full matching	(1, 0.33)
m_7	Category	Type	1	Full matching	(1, 0.67)
m_8	Key	id	3	Full matching	(1, 0)
m_{10}	Address	Street	2	t is part of s	(1, 0.33)
m_{12}	Address	ZipCode, Street	3	Has a common part	(1, 0.03)
m_{13}	Address	ZipCode, City	1	Has a common part	(1, 0.5)
m_{14}	Address	Street, City	3	t is part of s	(1, 0.12)
m_{15}	Address	ZipCode, Street, City	3	s is part of t	(1, 0.12)

7 Evaluation and Discussion

In this section we describe an experimental evaluation of our method which shows the influence of the parameters and the benefits of using content data (compared to schema information-only-based methods).

7.1 Datasets

To illustrate the proposed approach we consider different real-world datasets, respectively, containing information about 'compact discs', 'restaurants' and 'points of interest'.

Compact disc (CD) data are contained in two datasets which are defined in two schemas. The first schema is extracted from FreeDB[2] and consists of 8 attributes. The second schema is extracted from Discogs[3] and consists of 24 attributes, of which 6 have been identified manually as being coreferent with the FreeDB schema attributes and act as the ground truth for our experiments. The FreeDB dataset contains 124 tuples which are extracted from the CD dataset,[4] while the Discogs dataset contains

[2]FreeDB, http://www.freedb.org/.

[3]Discogs, http://www.discogs.com/data/.

[4]http://hpi.de/naumann/projects/repeatability/datasets/cd-datasets.html.

132 tuples which are extracted from Discogs.[5] The number of coreferent tuples in these datasets is equal to 33, which are detected manually.

Restaurant data are represented by two famous datasets [28]. One dataset stems from the on-line guide 'Zagat', while the other dataset stems from the on-line guide 'Fodor'. Zagat contains 331 tuples and Fodor contains 533 tuples, where 112 coreferent tuples are counted (i.e., 112 restaurants occur in both lists). These datasets are defined in two pairs of schemas, called R1 and R2, respectively. Both schemas of the first pair R1 consist of 6 attributes, of which all 6 have been identified manually as being coreferent, i.e., each attribute in the Fodor schema corresponds to exactly one attribute in the Zagat schema, and vice versa. More specifically, 6 one-to-one matchings are established. The Fodor schema in the second schemas pair R2 is identical to the Fodor schema in the first schema pair R1. But the Zagat schema in the second schemas pair R2 consists of 5 attributes, of which the attribute "street-city" is a concatenation of the attributes "street" and "city". Thus, each of the 4 attributes in the Fodor schema corresponds to exactly one attribute in the Zagat schema, and the concatenation of the attributes "street" and "city" in the Fodor schema corresponds to the attribute "street-city" in the Zagat schema. So, four one-to-one matchings and one-to-many matching are present. The truly coreferent schema attributes act as ground truth for our experiments.

Points of interest data are represented by two datasets which are defined in two schemas. The first dataset is made available by the Belgian company RouteYou,[6] which is an on-line provider of cycling routes. In order to support their routing algorithms, RouteYou manages a database with POIs. This database is defined by a schema which consists of the attributes latitude, longitude, POI name, POI category, POI internal name (which is a copy of the POI name extended by location information) and the language in which the name and category are given. An important characteristic of the given POI database is that data is mostly contributed by independent users of the website. Hereby, a user can pinpoint a location on the map, type in the name of the POI he/she wants to add and associate one of the predefined POI categories to it. From the complete POI database, we inferred one dataset by selecting tuples in English and in a specific area: the center of Ghent. This resulted in the RouteYou dataset which consists of 136 tuples. The second dataset contains 945 tuples which were extracted from the Google Maps database and are related to the same specific area. The tuple extraction has been done using the Google Places API.[7] The resulting dataset is defined by a schema which consists of the attributes id, name, vicinity, lat, lng, googleId and type, of which 6 have been identified manually as being coreferent with the attributes of the RouteYou dataset.

Table 8 contains a summary of all datasets considered in the experiments. The number of attributes in the data varies between 5 and 24 (column 2 in Table 8), while the number of truly coreferent attributes varies between 5 and 6 (column 5 in Table 8). The number of detected coreferent tuple pairs varies between 33 and 112 (column

[5] Discogs, http://www.discogs.com/data/.

[6] http://www.routeyou.com.

[7] Google Places, http://developers.google.com/places/.

Table 8 Real-world datasets

Datasets	# attributes	# tuples	# dup	# coreferent attr.
S: CD.FreeDB	8	124	33	6
T: CD.Discogs	24	132		
S: R1.Fodor	6	533	112	6
T: R1.Zagat	6	331		
S: R2.Fodor	6	533	112	5
T: R2.Zagat	5	331		
S: POI.RouteYou	9	136	51	6
T: POI.Google	7	945		

4 in Table 8), while the number of tuples varies between 124 and 945 (column 3 in Table 8).

7.2 Evaluation Setting

To determine the quality of our approach, we compared its result against the manually derived results. Based on the standard confusion matrix, we will consider the following three sets. The first set, denoted as B, contains the truly coreferent objects which are discovered by our approach, i.e., so-called true positives. The second set, denoted as A, contains truly coreferent objects which are not identified, i.e., so-called false negatives. The last set, denoted as C, contains objects which are falsely identified as coreferent, i.e., so-called false positives.

Precision is defined as the fraction of truly coreferent objects among all objects classified by a given algorithm as being coreferent:

$$\text{Precision} = \frac{|B|}{|B| + |C|}. \tag{48}$$

Recall is another important quality measure which in our case can be defined as the fraction of true positive objects among all coreferent objects present in a test dataset:

$$\text{Recall} = \frac{|B|}{|A| + |B|}. \tag{49}$$

7.3 Experiment: Configuration of Parameters of the Vertical Matcher

Goal. Our vertical matching Algorithm 2 in Sect. 5 employs the parameters $P_V.thrStats$ and $P_V.thrOverlap$. $P_V.thrStats$ specifies the threshold above which statistical properties of attribute domains are considered as similar. $P_V.thrOverlap$ specifies the percentage of the domains overlap. This experiment evaluates the impact of these parameters on the precision and recall of the established vertical schema matching.

Procedure. For the parameter $P_V.thrStats$, a range from 0 to 1 with a step equal to 0.01 is considered. For the parameter $P_V.thrOverlap$, a range from 0 to 1 with a step equal to 0.1 is considered. Mean recall and precision for each value of $P_V.thrStats$ and $P_V.thrOverlap$ over all datasets are calculated. Statistical matcher is executed before the lexical matcher, thus, the overlap threshold does not have to be considered.

Result. Figure 5 shows the mean precision and recall over all datasets for the different values of the parameter $P_M.thrStats$ (uninterested results are omitted). For $P_M.thrStats$ values between 0.08 and 0.96 the statistical comparison of content data gives the highest precision of matching. In this case, precision is more important than recall because non matched truly coreferent attributes can be matched by the lexical matcher. Besides that, the criteria of the statistical matcher are strict (all properties have to be similar) because statistical information may mislead the matcher, i.e., there can exist domains which have similar properties but may describe non coreferent attributes. We choose $P_M.thrStats$ equal to 0.9 for the further evaluations.

Figure 6 shows the mean precision and recall obtained over all datasets for different values of the parameter $P_M.thrOverlap$ in the lexical matcher. For $P_M.thrOverlap$ values equal to 0.3 and 0.4 the lexical comparison of content data gives matchings with the highest precision and recall. The established matchings are the basis to detect coreferent tuple pairs, which in turn are used to derive the final schema matching,

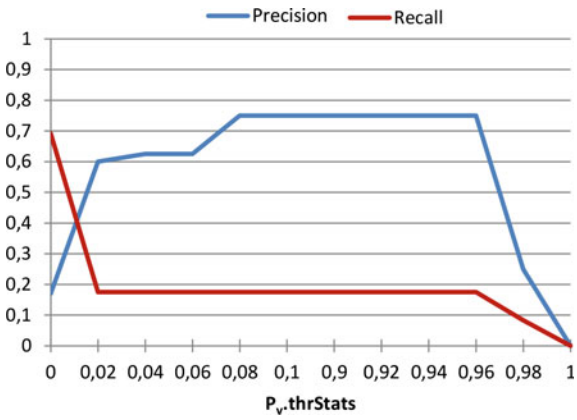

Fig. 5 Mean precision and recall over all datasets in function of $P_V.thrStats$ and $P_M.thrOverlap$ equal to 0.3

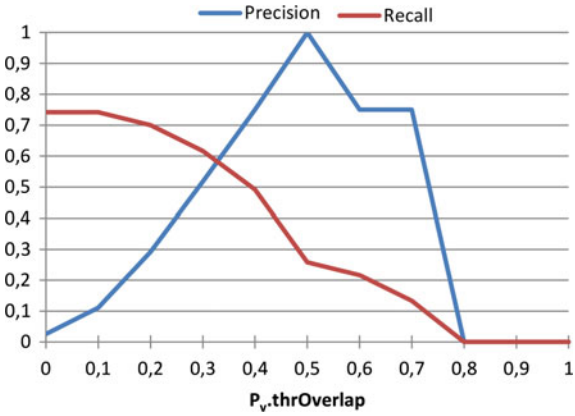

Fig. 6 Mean precision and recall over all datasets in function of $P_V.thrOverlap$ and $P_M.thrStats$ equal to 0.9

thus, the method has to derive as many as possible matchings at the expense of precision—the non coreferent matchings are eliminated by the horizontal matcher. Thus, $P_M.thrOvelap$ equal to 0.3 is selected for the further evaluations, because the smaller percentage of domain terms that overlap is easier to satisfy.

The combination of the matchings which are established by statistical and lexical matchers gives an average precision equal to 0.58 and recall equal to 0.79 over all datasets for $P_M.thrStats$ equal to 0.9 and $P_M.thrOvelap$ equal to 0.3.

7.4 Experiment: Configuration of Parameters of the Horizontal Matcher

Goal. The goal of this experiment is to show the impact of the number $P_H.n$ of coreferent tuple pairs, which is the basis to establish the schema matching, and the parameter $P_H.thrMajority$ of our horizontal matching algorithm in Sect. 6 on the precision and recall of the established schema matching. $P_H.thrMajority$ is a threshold which specifies that a matching is considered as a correct matching by the horizontal matcher.

Procedure. For the parameter $P_H.n$, a range from 1 to 10 is considered. For the parameter $P_H.thrMajority$, values 0.25, 0.5, 0.75 and 1 are considered. $P_H.thrMajority$ equal to 0.25 (0.5, 0.75 or 1) means that if any vertical matching $m \in M_V$ between the particular attributes is repeated by a quarter (two quarters, three quarters or all, respectively) of detected coreferent tuple pairs then m is added to the set M of horizontal matchings. The mean precision and recall for each value of $P_H.thrMajority$ and $P_H.n$ over all datasets are calculated.

Result. Figure 7 shows the mean precision and recall over all datasets for different values of the parameters $P_H.thrMajority$ and $P_H.n$.

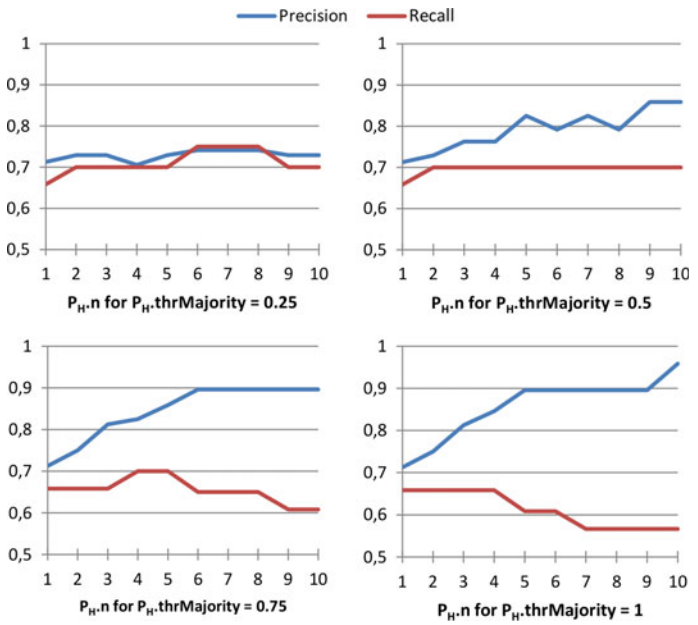

Fig. 7 Mean precision and recall over all datasets for different $P_H.thrMajority$ in function of $P_H.n$

Setting $P_M.thrMajority$ to 0.5 or 0.75, and $P_H.n$ to 5 or 6 is sufficient to establish the schema matching with high precision and recall. More coreferent tuple pairs ($P_H.n$ greater than 6) do not increase the precision significantly or can even decrease the recall for a large value of $P_H.thrMajority$, because $P_H.thrMajority$ equal to 1 forces that a particular matching has to be confirmed by all the selected coreferent tuple pairs. However, this may be unrealistic and difficult to satisfy. Besides that, our horizontal matcher is based on the n most coreferent tuples pairs so using many coreferent tuple pairs may result in coreferent tuples pairs having assigned low certainty (because the values of some attributes may not be coreferent). Thus, it is recommended to use only a few coreferent tuple pairs but then those that are assigned the highest certainty and $P_M.thrMajority$ between 0.5 and 0.75.

7.5 *Benefits of Using Content Data*

Content data-based schema matching approaches are able to establish semantical matchings of corresponding schema elements in those situations where the schema information-only-based methods can be ineffective. For example, a schema matching method which is based only on element names is not able to create a matching if the names of coreferent attributes are synonyms. This means that content data add additional information about the particular attribute which can be used to better infer the

semantics of the attribute and, as a consequence, create a matching between coreferent attributes. Moreover, attribute names may even mislead the schema matching methods which are only based on schema information. Content data are a powerful and valuable source of information which can be used to considerably improve schema matching. In contrast, approaches which are based on schema information only can establish matchings of attributes which are not coreferent based on the content data, e.g., "id" attributes.

8 Conclusion

In this paper, a content data based schema matching algorithm has been proposed as a way to construct proper matching between corresponding schema elements of heterogeneous datasets. The algorithm is especially useful in cases where finding the correspondences between the schema elements based on schema information only is difficult or impossible. Our novel technique employs possibilistic truth values, to express certainty of matchings, similarities etc., and fuzzy integers, to express cardinalities of sets of true propositions based on the certainty of their truth expressed using PTVs. This allows us to explicitly cope with the (un)certainty of semantical one-to-one and one-to-many schema matchings which are set up by our novel techniques in an automated fashion. As a consequence, solutions to the attribute granularity problem and the data coverage problem are proposed. The behaviour of the novel schema matching algorithm has been evaluated on several real life datasets, thus providing us with a valuable insight into the influence of the different parameters. Moreover, it has been shown what are the advantages of the proposed approach compared with a schema matching approach based on schema information only.

Acknowledgments This contribution is supported by the Foundation for Polish Science under International PhD Projects in Intelligent Computing. Project financed from The European Union within the Innovative Economy Operational Programme 2007–2013 and European Regional Development Fund. This work was also partially supported by the National Science Centre (contract no. UMO-2011/01/B/ST6/06908).

References

1. Bilke, A., Naumann, F.: Schema matching using duplicates. In: Proceedings of the 28th International Conference on Data Engineering (ICDE) (2005)
2. Bronselaer, A., De Tré, G.: A possibilistic approach on string comparison. IEEE Trans. Fuzzy Syst. **17**(1), 208–223 (2009)
3. Bronselaer, A., De Tré, G.: Properties of possibilistic string comparison. IEEE Trans. Fuzzy Syst. **18**(2), 312–325 (2010)
4. Bronselaer, A., Hallez, A., De Tré, G.: Extensions of fuzzy measures and the sugeno integral for possibilistic truth values. Int. J. Intel. Syst. **24**(2), 97–117 (2009)

5. Calvo, T., Mayor, G., Mesiar, R. (eds.): Aggregation Operators: New Trends and Applications. Physica-Verlag GmbH, Heidelberg (2002)
6. Chua, C.E.H., Chiang, R.H.L., Lim, E.P.: Instance-based attribute identification in database integration. VLDB J. **12**(3), 228–243 (2003). Oct
7. de Cooman, G.: Towards a possibilistic logic. In: Ruan, D. (ed.) Fuzzy Set Theory and Advanced Mathematical Applications, International Series in Intelligent Technologies, vol. 4, pp. 89–133. Springer, US (1995)
8. Dhamankar, R., Lee, Y., Doan, A., Halevy, A., Domingos, P.: imap: discovering complex semantic matches between database schemas. In: Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data, ACM Press (2004)
9. Do, H.h., Rahm, E.: Coma—a system for flexible combination of schema matching approaches. In: Proceedings of the VLDB 2002, pp. 610–621 (2002)
10. Doan, A., Domingos, P., Levy, A.Y.: Learning source description for data integration. In: WebDB (Informal Proceedings), pp. 81–86 (2000)
11. Elmagarmid, A., Ipeirotis, P., Verykios, V.: Duplicate record detection: a survey. IEEE Trans. Knowl. Data Eng. **19**(1), 1–16 (2007)
12. Hallez, A., De Tré, G., Verstraete, J., Matthé, T.: Application of fuzzy quantifiers on possibilistic truth values. In: Proceedings of EUROFUSE EURO WG on Fuzzy Sets, pp. 252–254. EXIT (2004)
13. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning. Springer Series in Statistics. Springer New York Inc, New York (2001)
14. Jain, A.K., Duin, R.P.W., Mao, J.: Statistical pattern recognition: a review. IEEE Trans. Pattern Anal. Mach. Intell. **22**(1), 4–37 (2000). Jan
15. Little, R.J.A., Rubin, D.B.: Statistical Analysis with Missing Data. Wiley, New York (1986)
16. Lu, H., Fan, W., Goh, C.H., Madnick, S., Cheung, D.: Discovering and reconciling semantic conflicts: a data mining prospective. In: Proceedings of IFIP Working Conference on Data Semantics (DS-7) (1997)
17. Madhavan, J., Bernstein, P.A., Rahm, E.: Generic schema matching with cupid. In: Proceedings of the 27th International Conference on Very Large Data Bases. pp. 49–58. VLDB '01, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2001)
18. Mehdi, O.A., Ibrahim, H., Affendey, L.S.: Instance based matching using regular expression. Procedia CS **10**, 688–695 (2012)
19. Perkowitz, M., Doorenbos, R.B., Etzioni, O., Weld, D.S.: Learning to understand information on the internet: an example-based approach. J. Intel. Inf. Syst. **8**(2), 133–153 (1997). Mar
20. Prade, H.: Possibility sets, fuzzy sets and their relation to Lukasiewicz logic. In: Proceeding of 12th Int Symp on Multiple-Valued Logic. pp. 223–227 (1982)
21. Rahm, E., Bernstein, P.A.: A survey of approaches to automatic schema matching. VLDB J. **10**(4), 334–350 (2001). Dec
22. Reiss, R.D., Thomas, M.: Statistical analysis of extreme values: with applications to insurance, finance, hydrology and other fields. Birkhuser Basel, 3rd edn. (2007)
23. Sugeno, M.: Theory of Fuzzy Integrals and its Applications. Ph.D. thesis, Tokyo, Japan (1974)
24. Szymczak, M., Koepke, J.: Matching methods for semantic annotation-based XML document transformations. In: K. Atanassov, et al. (Eds.), New Developments in Fuzzy Sets, Intuitionistic Fuzzy Sets, Generalized Nets and Related Topics. Applications. Volume II. pp. 297–308. SRI PAS (2012)
25. Szymczak, M., Zadrożny, S., De Tré, G.: Coreference detection in XML metadata. In: Pedrycz, W., Reformat, M. (eds.) Proceedings of 2013 Joint IFSA World Congress NAFIPS Annual Meeting. pp. 1354–1359 (2013)
26. Szymczak, M., Bronselaer, A., Zadrożny, S., De Tré, G.: Semantical mappings of attribute values for data integration. In: Proceedings of NAFIPS 2014. pp. 1–8. IEEE (2014)
27. Szymczak, M., Zadrożny, S., Bronselaer, A., De Tré, G.: Coreference detection in an XML schema. Inf. Sci. **296**, 237–262 (2015)
28. Tejada, S., Knoblock, C., Minton, S.: Learning object identification rules for information integration. Inf. Syst. **26**(8), 607–633 (2001)

29. Yager, R.: On the theory of bags. Int. J. Gen. Syst. **13**(1), 23–27 (1986)
30. Zadeh, L.: Fuzzy sets as a basis for a theory of possibility. Fuzzy Sets Syst. **100**, 9–34 (1999). Apr
31. Zadrożny, S., Kacprzyk, J., Sobota, G.: Avoiding duplicate records in a database using a linguistic quantifier based aggregation—a practical approach. In: Proceedings of FUZZ-IEEE. pp. 2194–2201 (2008)