4

Basic techniques

The goal of ontology matching is to find the relations between entities expressed in different ontologies. Very often, these relations are equivalence relations that are discovered through the measure of the similarity between the entities of ontologies.

We present here some of the basic methods for assessing the similarity or the relations between ontology entities. By basic, we mean that these methods base their judgment on one particular kind of features of these entities. Chap. 5, in turn, shows how the results of these methods can be combined.

In this chapter, we first introduce basic concepts related to similarity (§4.1). Then, we consider basic methods following the 'kind of input' layer of the classification of Chap. 3: entity names (§4.2), structure (§4.3), extension (§4.4) and semantics (§4.5).

4.1 Similarity, distances and other measures

There are many ways to assess the similarity between two entities. The most common way amounts to defining a measure of this similarity. We present some characteristics of these measures.

Definition 4.1 (Similarity). *A similarity $\sigma : o \times o \rightarrow \mathbb{R}$ is a function from a pair of entities to a real number expressing the similarity between two objects such that:*

$$\begin{aligned} \forall x, y \in o, \sigma(x, y) \geq 0 & \qquad \textit{(positiveness)} \\ \forall x \in o, \forall y, z \in o, \sigma(x, x) \geq \sigma(y, z) & \qquad \textit{(maximality)} \\ \forall x, y \in o, \sigma(x, y) = \sigma(y, x) & \qquad \textit{(symmetry)} \end{aligned}$$

The dissimilarity is a dual operation. It is defined as follows.

Definition 4.2 (Dissimilarity). *Given a set o of entities, a dissimilarity $\delta : o \times o \rightarrow \mathbb{R}$ is a function from a pair of entities to a real number such that:*

$$\forall x, y \in o, \delta(x, y) \geq 0 \qquad \textit{(positiveness)}$$
$$\forall x \in o, \delta(x, x) = 0 \qquad \textit{(minimality)}$$
$$\forall x, y \in o, \delta(x, y) = \delta(y, x) \qquad \textit{(symmetry)}$$

Some authors consider a 'non symmetric (dis)similarity', [Tverski, 1977]; we then use the term non symmetric measure or pre-similarity. There are more constraining notions of dissimilarity, such as distances and ultrametrics.

Definition 4.3 (Distance). *A distance (or metric) $\delta : o \times o \longrightarrow \mathbb{R}$ is a dissimilarity function satisfying the definiteness and triangular inequality:*

$$\forall x, y \in o, \delta(x, y) = 0 \textit{ if and only if } x = y \qquad \textit{(definiteness)}$$
$$\forall x, y, z \in o, \delta(x, y) + \delta(y, z) \geq \delta(x, z) \qquad \textit{(triangular inequality)}$$

Definition 4.4 (Ultrametric). *Given a set o of entities, an ultrametric is a metric such that:*

$$\forall x, y, z \in o, \delta(x, y) \leq \max(\delta(x, z), \delta(y, z)) \qquad \textit{(ultrametric inequality)}$$

Very often, the measures are normalised, especially if the similarity of different kinds of entities must be compared. Reducing each value to the same scale in proportion to the size of the considered space is the common way to normalise.

Definition 4.5 (Normalised (dis)similarity). *A (dis)similarity is said to be* normalised *if it ranges over the unit interval of real numbers* $[0\ 1]$. *A normalised version of a (dis)similarity σ (respectively, δ) is denoted as $\overline{\sigma}$ (respectively, $\overline{\delta}$).*

It is easy to see that to any normalised similarity $\overline{\sigma}$ corresponds a normalised dissimilarity $\overline{\delta} = 1 - \overline{\sigma}$ and vice versa. In the remainder, we will consider mostly normalised measures and assume that a dissimilarity function between two entities returns a real number between $0.$ and $1.$

From the above definitions, the similarity and dissimilarity are complete functions that map pairs of entities to real numbers. An alternative representation for such a function on a finite set of entities is a *matrix* (see Example 4.14). The matrix has the advantage of being a finite data structure that can be exchanged between programs.

4.2 Name-based techniques

Some terminological methods compare strings. They can be applied to the name, the label or the comments of entities in order to find those which are similar. This can be used for comparing class names and/or URIs.

Throughout this section, the set $\mathbb{S}$ will represent the set of strings, i.e., the sequences of letters of any length over an alphabet $\mathbb{L}$: $\mathbb{S} = \mathbb{L}*$. The empty string is denoted as ϵ, and $\forall s, t \in \mathbb{S}$, $s + t$ is the concatenation of the strings s and t. $|s|$ denotes the length of the string s, i.e., the numbers of characters it contains. $s[i]$ for $i \in [1\ |s|]$ stands for the letter in position i of s.

Example 4.6 (Strings). The string 'article' is made of the letters a, r, t, i, c, l and e. Its length is 7 characters. 'peer-reviewed' and ' ' are two other strings (so '-' and ' ' are letters in the alphabet) and their concatenation 'peer-reviewed'+' '+'article' provides the string 'peer-reviewed article' whose length is 21.

A string s is the substring of another string t, if there exist two strings s' and s'', such that $s' + s + s'' = t$ (denoted as $s \in t$). Two strings are equal ($s = t$) if and only if $s \in t$ and $t \in s$. The number of occurrences of s in t (denoted as $s\#t$) is the number of distinct pairs s', s'', such that $s' + s + s'' = t$.

Example 4.7 (Substrings). The string 'peer-reviewed article' has the string 'review' as a substring because 'peer-'+'review'+'ed article'='peer-reviewed article'. The string 'homonymous' has three occurences of the string 'o', two occurences of the string 'mo' and only one occurence of the string 'nym'.

The main problem in comparing ontology entities on the basis of their labels occurs due to the existence of synonyms and homonyms:

Synonyms are different words used to name the same entity. For instance, Article and Paper are synonyms in some contexts;

Homonyms are words used to name different entities. For instance, peer as a noun has a sense 'equal' as well as another sense 'member of the nobility'. The fact that a word can have multiple senses is also known as *polysemy*.

Consequently, it is not possible to deduce with certainty that two entities are the same if they have the same name or that they are different because they have different names. There are more reasons than synonymy and homonymy why this could happen. In particular:

– Words from different languages, such as English, French, Italian, Spanish, German, Greek, are used to name the same entities. For instance, the word Book in English is Livre in French and книга in Russian.
– Syntactic variations of the same word often occur according to different acceptable spellings, abbreviations, use of optional prefixes or suffixes, etc. For instance, Compact disc, CD, C.D. and CD-ROM can be considered equivalent in some contexts. However, in some other contexts, CD may mean Corps diplomatique and in some others change directory.

These kinds of variations can occur within one ontology but can be even more frequent across ontologies. However, the way in which things are named remains very important in every day communication and names remain a good index of similarity or dissimilarity. Moreover, many different techniques have been designed for assessing the similarity of two terms notwithstanding the similarity or dissimilarity of the strings which denote them.

There are two main categories of methods for comparing terms depending on their consideration of character strings only (§4.2.1) or using some linguistic knowledge to interpret these strings (§4.2.2).

4.2.1 String-based methods

String-based methods take advantage of the structure of the string (as a sequence of letters). String-based methods will typically find classes Book and Textbook to be similar, but not classes Book and Volume.

There are many ways to compare strings depending on the way the string is viewed: for example, as an exact sequence of letters, an erroneous sequence of letters, a set of letters, a set of words. [Cohen *et al.*, 2003b] compares various string-matching techniques, from distance like functions to token-based distance functions. We discuss the most frequently used methods.

We distinguish between (i) normalisation techniques which are used for reducing strings to be compared to a common format, (ii) substring or subsequence techniques that base similarity on the common letters between strings, (iii) edit distances that further evaluate how one string can be an erroneous version of another, (iv) statistical measures that establish the importance of a word in a string by weighting the relation between two strings and (v) path comparisons.

Normalisation

Before comparing actual strings which have a meaning in natural language, there are normalisation procedures that can help improve the results of subsequent comparisons. In particular:

Case normalisation consists of converting each alphabetic character in the strings into their lower case counterpart. For example, CD becomes cd and SciFi becomes scifi.

Diacritics suppression consists of replacing characters with diacritic signs with their most frequent replacements. For example, replacing Montréal with Montreal.

Blank normalisation consists of normalising all blank characters, such as blank, tabulation, carriage return, or sequences of these, into a single blank character.

Link stripping consists of normalising some links between words, such as replacing apostrophes and blank underline into dashes or blanks. For example, peer-reviewed becomes peer reviewed.

Digit suppression consists of suppressing digits. For example, book24545-18 becomes book.

Punctuation elimination suppresses punctuation signs. For example, C.D. becomes CD.

These normalisation operations must be used with care for several reasons. In particular:

– they are often language-dependent, e.g., they work for occidental languages;
– they are order dependent: they do not guarantee to bring the same results when applied in any order;
– they can result in loosing some meaningful information; for example, carbon-14 becomes carbon or sentence separation, which is very useful for parsing, is lost;
– they may reduce variations, but increase synonyms. For example, in French livre and livré are different words respectively meaning book and shipped.

String equality

String equality returns 0 if the strings under consideration are not identical and 1 if they are identical. This can be taken as a similarity measure.

Definition 4.8 (String equality). *String equality is a similarity* $\sigma : \mathbb{S} \times \mathbb{S} \rightarrow [0\ 1]$ *such that* $\forall x, y \in \mathbb{S}$, $\sigma(x, x) = 1$ *and if* $x \neq y, \sigma(x, y) = 0$.

It can be performed after some syntactic normalisation of the string, e.g., downcasing, encoding conversion, accent normalisation.

This measure does not explain how strings are different. A more immediate way of comparing two strings is the Hamming distance which counts the number of positions in which the two strings differ [Hamming, 1950]. We present here the version normalised by the length of the longest string.

Definition 4.9 (Hamming distance). *The Hamming distance is a dissimilarity* $\delta : \mathbb{S} \times \mathbb{S} \rightarrow [0\ 1]$ *such that:*

$$\delta(s, t) = \frac{\left(\sum_{i=1}^{\min(|s|,|t|)} s[i] \neq t[i]\right) + ||s| - |t||}{\max(|s|, |t|)}$$

Substring test

Different variations can be obtained from the string equality, such as considering that strings are very similar when one is a substring of another:

Definition 4.10 (Substring test). *Substring test is a similarity* $\sigma : \mathbb{S} \times \mathbb{S} \rightarrow [0\ 1]$ *such that* $\forall x, y \in \mathbb{S}$, *if there exist* $p, s \in \mathbb{S}$ *where* $x = p + y + s$ *or* $y = p + x + s$, *then* $\sigma(x, y) = 1$, *otherwise* $\sigma(x, y) = 0$.

This is obviously a similarity. This measure can be refined in a substring similarity which measures the ratio of the common subpart between two strings.

Definition 4.11 (Substring similarity). *Substring similarity is a similarity* $\sigma : \mathbb{S} \times \mathbb{S} \rightarrow [0\ 1]$ *such that* $\forall x, y \in \mathbb{S}$, *and let t be the longest common substring of* x *and* y*:*

$$\sigma(x, y) = \frac{2|t|}{|x| + |y|}$$

It is easy to see that this measure is indeed a similarity. One could also consider a subsequence similarity as well. This definition can be used for building functions based on the longest common prefix or longest common suffix.

Thus, for example, the similarity between article and aricle would be $4/7 = .57$, while between article and paper would be $1/7 = .14$, and, finally, between article and particle would be $6/7 = .86$.

A prefix or suffix pre-similarity can be defined on this model from the prefix and suffix tests, which test whether one string is the prefix or suffix of another. These measures would not be symmetric. Prefix and suffix pre-similarity can be useful as a test for strings denoting a more general concept than another (in many languages, adding clauses to a term would restrict its range). For instance, reviewed article is more specific than article. It can also be used for comparing strings and similar abbreviations, e.g., ord and order.

The n-gram similarity is also often used in comparing strings. It computes the number of common n-grams, i.e., sequences of n characters, between them. For instance, trigrams for the string article are: art, rti, tic, icl, cle.

Definition 4.12 (n-gram similarity). *Let $ngram(s, n)$ be the set of substrings of s of length n. The n-gram similarity is a similarity $\sigma : \mathbb{S} \times \mathbb{S} \longrightarrow \mathbb{R}$ such that:*

$$\sigma(s,t) = |ngram(s,n) \cap ngram(t,n)|$$

The normalised version of this function is as follows.

$$\overline{\sigma}(s,t) = \frac{|ngram(s,n) \cap ngram(t,n)|}{\min(|s|,|t|) - n + 1}$$

This function is quite efficient when only some characters are missing.

Thus, for example, the similarity between article and aricle would be $2/4 = .5$, while between article and paper would be 0, and, finally, between article and particle would be $5/6 = .83$.

It is possible, to add extra characters at the beginning and end of strings for dealing with too small strings.

Edit distance

Intuitively, an edit distance between two objects is the minimal cost of operations to be applied to one of the objects in order to obtain the other one. Edit distances were designed for measuring similarity between strings that may contain spelling mistakes.

Definition 4.13 (Edit distance). *Given a set Op of string operations ($op : \mathbb{S} \longrightarrow \mathbb{S}$), and a cost function $w : Op \longrightarrow \mathbb{R}$, such that for any pair of strings there exists a sequence of operations which transforms the first one into the second one (and vice versa), the edit distance is a dissimilarity $\delta : \mathbb{S} \times \mathbb{S} \longrightarrow [0\ 1]$ where $\delta(s,t)$, is the cost of the less costly sequence of operations which transforms s into t.*

$$\delta(s,t) = \min_{(op_i)_I; op_n(\dots op_1(s)) = t} \left(\sum_{i \in I} w_{op_i}\right)$$

In string edit distance, the operations that are usually considered include insertion of a character $ins(c, i)$, replacement of a character by another $sub(c, c', i)$ and deletion of a character $del(c, i)$. It can be easily checked that these operations are such that $ins(c, i) = del(c, i)^{-1}$ and $sub(c, c', i) = sub(c', c, i)^{-1}$. Each operation is assigned a cost and the distance between two strings is the sum of the cost of each operation on the less costly set of operations.

The Levenshtein distance [Levenshtein, 1965] is the minimum number of *insertions*, *deletions*, and *substitutions* of characters required to transform one string into the other. It is the edit distance with all costs equal to 1. The Needleman–Wunch distance [Needleman and Wunsch, 1970], in turn, is the edit distance with a higher costs for ins and del.

It can be proved that the edit distance is indeed a distance if $\forall op \in Op, w_{op} = w_{op^{-1}}$.

Example 4.14. The (rounded) Levenshtein distance table between the class labels of ontologies in Fig. 2.7 (p. 37):

	Science	Children	Book	Person	DVD	Textbook	Product	Pocket	Publisher	Popular	CD
Politics	0.75	1.00	0.88	0.88	1.00	1.00	0.75	0.75	0.67	0.75	1.00
Thing	0.71	0.75	1.00	1.00	1.00	0.88	1.00	1.00	0.89	1.00	1.00
Autobiography	0.92	0.85	0.85	0.92	1.00	0.85	0.92	0.92	0.85	0.85	1.00
Novel	0.86	0.88	0.80	1.00	1.00	1.00	0.86	0.67	0.89	0.71	1.00
Biography	1.00	0.89	0.78	0.89	1.00	1.00	0.89	0.89	1.00	0.89	1.00
Writer	0.86	0.75	1.00	1.00	1.00	0.88	0.86	0.83	0.67	0.86	1.00
Essay	1.00	1.00	1.00	0.83	1.00	1.00	1.00	1.00	0.89	0.86	1.00
Volume	0.86	0.75	0.83	1.00	1.00	1.00	0.71	0.83	0.78	0.71	1.00
LiteraryCritic	0.93	0.93	1.00	0.86	1.00	0.93	0.86	0.93	0.93	0.86	0.93
Poetry	0.86	0.88	0.83	0.83	1.00	0.88	0.71	0.67	0.89	0.71	1.00
Literature	0.80	0.90	1.00	0.80	1.00	0.90	0.80	0.90	0.90	0.80	1.00
Human	0.86	0.88	1.00	0.83	1.00	1.00	1.00	1.00	0.89	0.71	1.00

The closest names are Pocket and Novel, Pocket and Poetry, as well as Writer and Publisher and Politics and Publisher. These names are relatively far from each others (.67). So, in this case no correspondence can be found from such measures alone. However, the same measure on properties will obviously find the correspondence between author and author, for instance.

Other measures compute the cost of an edition operation as a function of the characters or substrings on which the operation applies. For that purpose, they use a cost matrix for each operation. A well known example of such a measure is the Smith–Waterman measure [Smith and Waterman, 1981] which was adapted to compute the distance between biological sequences based on the molecules that were manipulated. Other such measures are the Gotoh [Gotoh, 1981] and Monge–Elkan [Monge and Elkan, 1997] distance functions.

The Jaro measure has been defined for matching proper names that may contain similar spelling mistakes [Jaro, 1976, Jaro, 1989]. It is not based on an edit distance model, but on the number and proximity of the common characters between two strings. This measure is not a similarity because it is not symmetric.

Definition 4.15 (Jaro measure). *The Jaro measure is a non symmetric measure* $\sigma : \mathbb{S} \times \mathbb{S} \rightarrow [0\ 1]$ *such that*

$$\sigma(s,t) = \frac{1}{3} \times (\frac{|com(s,t)|}{|s|} + \frac{|com(t,s)|}{|t|} + \frac{|com(s,t)| - |transp(s,t)|}{|com(s,t)|}),$$

with

$$s[i] \in com(s,t) \textit{ if and only if } \exists j \in [i - (\min(|s|,|t|)/2\ i + (\min(|s|,|t|)/2]$$

and $transp(s,t)$ *are the elements of* $com(s,t)$ *which occur in a different order in* s *and* t.

For instance, if we again compare article with aricle, aritcle and paper, the number of common letters will respectively be 6, 7 and 1 (because in the last case, the 'e' in paper is too far away from that in article). The number of transposed common letters will be 0, 1 and 0 respectively. As a consequence, the similarities between these strings are: .95, .90 and .45.

This measure has been improved by favouring matches between strings with longer common prefixes [Winkler, 1999].

Definition 4.16 (Jaro–Winkler measure). *The Jaro–Winkler measure* $\sigma : \mathbb{S} \times \mathbb{S} \rightarrow [0\ 1]$ *is as follows:*

$$\sigma(s,t) = \sigma_{Jaro}(s,t) + P \times Q \times \frac{(1 - \sigma_{Jaro}(s,t))}{10},$$

such that P *is the length of the common prefix and* Q *is a constant.*

In this case, the similarity for the three strings compared to article with $Q = 4$ are: .99, .98 and .45. These measures only improve on the previous ones by explicitly providing a model of mistakes that penalises less the comparison.

Another similar measure is Smoa [Stoilos *et al.*, 2005] which is adapted to the way computer users define identifiers. It depends on common substring lengths and non common substring lengths, the second part being substracted from the first one. This measure has a value between -1 and 1.

Token-based distances

The following techniques come from information retrieval and consider a string as a (multi)set of words (also called bag of words), i.e., a set in which a particular item can appear several times. These approaches usually work well on long texts (comprising many words). For that reason, it is helpful to take advantage of other strings that are attached to ontology entities. This can be adapted to ontology entities as follows:

– By aggregating different sources of strings: identifiers, labels, comments, documentation, etc. Some systems go further by aggregating the tokens that correspond to connected entities [Qu *et al.*, 2006].
– By splitting strings into independent tokens. For example, InProceedings becomes In and Proceedings, peer-reviewed article becomes peer, reviewed and article.

Ontology entities are then identified with *bags of words* (or multisets) suitable for manipulation by using information retrieval techniques. Many different similarities or dissimilarities being applied to sets of entities can thus be applied to these bags of words. For example, the matching coefficient is the complement of the Hamming distance on sets (§4.4.1) and the Dice coefficient is the complement of the Hamming distance on multisets, i.e., using the union, intersection and cardinality of multisets instead of sets.

Original measures are those based on the corpus of such strings, i.e., the set of all such strings found in one of the ontologies or in both of them. These measures are no longer intrinsic to the strings to be compared but depend on the corpus.

They usually consider a bag of words s as a vector $\overrightarrow{s}$ belonging to a metric space V in which each dimension is a term (or token) and each position in the vector is the number of occurrences of the token in the corresponding bag of words. This is one way to represent multisets. Each document can be considered as a point in this space identified by its coordinate vector [Salton, 1971, Salton and McGill, 1983].

Once the entities have been transformed into vectors, usual metric space distances can be used: Euclidean distance, Manhattan distance (also known as city blocks) and any instance of the Minkowski distance (see also p. 123). We present here the cosine similarity which measures the cosine of the angles made by two vectors. It is very often used in information retrieval.

Definition 4.17 (Cosine similarity). *Given $\overrightarrow{s}$ and $\overrightarrow{t}$, the vectors corresponding to two strings s and t in a vector space V, the cosine similarity is the function $\sigma_V : V \times V \rightarrow [0\ 1]$ such that:*

$$\sigma_V(s,t) = \frac{\sum_{i\in|V|} \overrightarrow{s}_i \times \overrightarrow{t}_i}{\sqrt{\sum_{i\in|V|} \overrightarrow{s}_i^2 \times \sum_{i\in|V|} \overrightarrow{t}_i^2}}$$

Some more elaborate techniques use reduced spaces, like those obtained by correspondence analysis, in order to deal with a smaller dimension as well as to automatically map words of similar meanings to the same dimension. A famous example of such a technique, which is by using singular value decomposition, is known as latent semantic indexing [Deerwester *et al.*, 1990].

A very common measure is TFIDF (Term frequency-Inverse document frequency) [Robertson and Jones, 1976] which is used for scoring the relevance of a document, i.e., a bag of words, to a term by taking into account the frequency of appearance of the term in the corpus. It is usually not a measure of similarity: it assesses the relevance of a term to a document. It is used here to assess the relevance of a substring to a string by comparing the frequency of appearance of the string in the document with regard to its frequency in the whole corpus.

Definition 4.18 (Term frequency-Inverse document frequency). *Given a corpus C of multisets, we define the following measures:*

$$\begin{aligned}
\forall t \in \mathbb{S}, \forall s \in C, tf(t,s) &= t\#s && (term\ frequency)\\
\forall t \in \mathbb{S}, idf(t) &= log\left(\frac{|C|}{|\{s \in C; t \in s\}|}\right) && (inverse\ document\ frequency)\\
TFIDF(s,t) &= tf(t,s) \times idf(t) && (TFIDF)
\end{aligned}$$

Many systems use measures based on TFIDF. These measures compute, for each term in the strings, their relevance with regard to the corpus based on TFIDF. Then, they use vector space techniques for computing a distance between the two strings. There are several options for doing so depending on the selected space: this can be the whole corpus, the union of terms covered by the two strings or only the intersection of the terms involved in both strings. The most often used aggregation measure is the cosine similarity.

Path comparison

Path difference consists of comparing not only the labels of objects but the sequence of labels of entities to which those bearing the label are related. For instance, in the left-hand ontology of Fig. 2.7, the Science class can be identified by the path Product:Book:Science. In a first approximation, these can be considered as a particular way to aggregate tokens in an ordered fashion. A simple (and only) example is the one which concatenates all the names of the superclasses of classes before comparing them. So the result is dependent on the individual string comparison aggregated in some way.

Definition 4.19 (Path distance). *Given two sequences of strings, $\langle s_i \rangle_{i=1}^n$ and $\langle s'_j \rangle_{j=1}^m$, their path distance is defined as follows:*

$$\delta(\langle s_i \rangle_{i=1}^n, \langle s'_j \rangle_{j=1}^m) = \lambda \times \delta'(s_n, s'_m) + (1-\lambda) \times \delta(\langle s_i \rangle_{i=1}^{n-1}, \langle s'_j \rangle_{i=1}^{m-1})$$

such than

$$\delta(\langle\rangle, \langle s'_j \rangle_{j=1}^k) = \delta(\langle s_i \rangle_{i=1}^k, \langle\rangle) = k$$

with δ' being one of the other string or language-based distance and $\lambda \in [0\ 1]$.

For instance, we can take the string equality distance as δ', scoring 0 when the strings are equal, and .7 as λ. Then if we have to compare Product:Book:Science with Book:Essay:Science and Product:Cultural:Book:Science, the distances will respectively be: .273 and .09.

This measure is dependent on the similarity between the last element of each path: this similarity is affected by a λ penalty but every subsequent step is affected by a $\lambda \times (1-\lambda)^n$ penalty. So this measure takes into account the prefix, but the prefix can only influence the result to an extent which decreases as its distance from the end of the sequence increases. As can be seen, this measure is dependent on the rank of the elements to compare in the path. A more accurate, but expensive, measure, would choose the best match between both paths and penalise the items remote from the end of the path. Another way to take these paths into account is simply to apply them as a distance on sequences, such as described in [Valtchev, 1999].

Summary on string-based methods

The results given so far for these string comparisons are useful if people use very similar strings to denote the same concepts. If synonyms with different structures are used, this will yield a low similarity. Selecting pairs of strings with low similarity, in turn, yields many false positives since two strings can be very similar, e.g., Inproceedings and proceedings, and denote relatively different concepts. These measures are most often used in order to detect if two very similar strings are used. Otherwise, matching must use more reliable sources of information.

There are several software packages for computing string distances. Table 4.1 provides a brief comparison of distances available in four Java packages: Simetrics[1], SecondString[2], the Alignment API[3] and SimPack[4]. A comparison of the metrics of the second package has been provided in [Cohen *et al.*, 2003b].

4.2.2 Language-based methods

So far we have considered strings as sequences of characters. When considering language phenomenon, these strings become texts (theoretical peer-reviewed journal article). Texts can be segmented into words: easily identified sequence of letters that are derived from an entry in a dictionary (theoretical, peer, reviewed, journal, article). These words do not occur in a bag (as used in information retrieval) but in a sequence which has a grammatical structure. Very often words, like peer, bear a meaning and correspond to some concepts, but the more useful concepts to be properly handled in a text are terms, such as peer-review, or peer-reviewed journal.

Terms are phrases that identify concepts; they are thus often used for labelling concepts in ontologies. As a consequence, ontology matching could take great advantage of recognising and identifying them in strings. This amounts to recognise the term Peer-reviewed journal in the labels scientific periodicals reviewed by peers (and not in journal review paper).

Language-based methods rely on using Natural Language Processing (NLP) techniques to help extract the meaningful terms from a text. Comparing these terms and their relations should help assess the similarity of the ontology entities they name

[1] http://www.dcs.shef.ac.uk/˜sam/stringmetrics.html
[2] http://secondstring.sourceforge.net
[3] http://alignapi.gforge.inria.fr
[4] http://www.ifi.unizh.ch/ddis/simpack.html

Table 4.1. String measures available in Simetrics, SecondString, Alignment API and SimPack Java packages.

Simetrics	SecondString	AlignAPI	SimPack
	n-grams	n-grams	
Levenshtein	Levenshtein	Levenshtein	Levenshtein
Jaro	Jaro	Jaro	
Jaro–Winkler	Jaro–Winkler	Jaro–Winkler	
Needleman–Wunch	Needleman–Wunch	Needleman–Wunch	
		Smoa	
Smith–Waterman			
Monge–Elkan	Monge–Elkan		
Gotoh			
Matching coefficient			
Jaccard	Jaccard		Jaccard
Dice coefficient			Dice coefficient
	TFIDF		TFIDF
Cityblocks			Cityblocks
Euclidean			Euclidean
Cosine			Cosine
Overlap			Overlap
Soundex			

and comment. Although these are based on some linguistic knowledge, we distinguish methods which rely on algorithms only and those which make use of external resources such as dictionaries.

Intrinsic methods: Linguistic normalisation

Linguistic normalisation aims at reducing each form of a term to some standardised form that can be easily recognised. Table 4.2 shows that the same term (theory paper) can appear under many different forms. The work in [Maynard and Ananiadou, 2001] distinguishes three main kinds of term variation: morphological (variation on the form and function of a word based on the same root), syntactic (variation on the grammatical structure of a term) and semantic (variation on one aspect of the term, usually using a hypernym or hyponym). Various subtypes of these broad categories are exemplified in Table 4.2. Multilingual variation, i.e., where the term variant is expressed in a different language, can be naturally added to these. Moreover, these types of variations can be combined in various ways.

Complete linguistic software chains have been developed for quickly obtaining a normal form of strings denoting terms. This is available through shallow parsers or part-of-speech taggers [Brill, 1992]. These usually perform the following functions:

Tokenisation: Tokenisation is the operation described in Sect. 4.2.1. It consists of segmenting strings into sequences of tokens by a tokeniser which recognises

Table 4.2. Variants of the term *theory paper* (adapted from [Maynard, 1999] and [Euzenat *et al.*, 2004a]).

Type	Subtype	Example
Morphological	Inflection	theory papers
	Derivation	theoretical paper
	Inflectional-Derivational	theoretical papers
Syntactic	Insertion	theory review paper
	Permutation	paper on theory
	Coordination	philosophy and theory paper
Morphosyntactic	Derivation-Coordination	philosophical and theoretical paper
	Inflection-Permutation	papers on theory
Semantic		foundational paper
Multilingual	French	article théorique

punctuation, cases, blank characters, digits, etc. For example, peer-reviewed periodic publication becomes ⟨peer, reviewed, periodic, publication⟩.

Lemmatisation: The strings underlying tokens are morphologically analysed in order to reduce them to normalised basic forms. Morphological analysis makes it possible to find flexion and derivations of a root. This involves suppressing tense, gender or number marks. Retrieving the root is called lemmatisation. Currently, systems can use some approximate lemmatisation techniques called stemming [Lovins, 1968, Porter, 1980] which strip suffixes from terms. For example, reviewed becomes review.

Term extraction: More elaborate technologies enable the extraction of terms from a text [Jacquemin and Tzoukermann, 1999, Bourigault and Jacquemin, 1999, Maynard and Ananiadou, 2001, Cerbah and Euzenat, 2001]. It is generally related to what is called corpus linguistics and requires a relatively large amount of text. Terminology extractors identify terms from the repetition of morphologically similar phrases in the texts and the use of patterns, e.g., noun^1 noun^2 $\rightarrow$ noun^2 on noun^1. This would recognise that the term theory paper is the same term as paper on theory.

Stopword elimination: The tokens that are recognised as articles, prepositions, conjunctions, etc. (usually words, such as to or a), are marked to be discarded because they are considered as non meaningful (empty) words for matching. For example, collection of article becomes collection article.

Once these techniques have been applied, ontology entities are represented as sets of terms, not words, that can be compared with the same techniques as presented before.

Extrinsic methods

Extrinsic linguistic methods use external resources, such as dictionaries and lexicons. Several kinds of linguistic resources can be exploited in order to find similarities between terms.

Lexicons. A lexicon, or dictionary, is a set of words together with a natural language definition of these words (see for instance those of Example 4.21). Of course, for a particular word, e.g., Article, there can be several such definitions. Dictionaries can be used with gloss-based distances (see below).

Multi-lingual lexicons. Multi-lingual lexicons are lexicons in which the definition is replaced by the equivalent terms in another language, e.g., Paper in English corresponds to Article in French. Such dictionaries can be very useful if ontology labels are expressed in different languages. They can be used for matching as well as for disambiguating terms, i.e., identifying their intended sense, before matching.

Semantico-syntactic lexicons. Semantico-syntactic lexicons and semantic lexicons are resources used in natural language analysers. They very often not only record names but their categories, e.g., non animate, liquid, and record the types of arguments taken by verbs and adjectives, e.g., to flow takes a liquid as subject and has no object. These are difficult to create and are not much used in ontology matching.

Thesauri. A thesaurus is a kind of lexicon to which some relational information has been added. It usually contains relations, named hypernym, e.g., Biography is a more general term than Autobiography, which is hyponym, synonym, e.g., Paper means the same as Article, antonym, e.g., practice is the opposite of theory. WordNet [Miller, 1995] is such a thesaurus which distinguishes clearly between word senses by grouping words into sets of synonyms (synsets).

Terminologies. A terminology is a thesaurus for terms, which very often contains phrases rather than single words. They are usually domain specific and tend to be less equivocal than dictionaries.

This is not an exhaustive nor an authorised description of linguistic resources but it provides a typology of the kinds of properties on which a similarity between terms can be assessed on a linguistic basis.

These resources can be defined for one language or be specific to some domain. In the latter case, they tend to be more adapted when texts or ontologies concern this domain because they retain specialised senses, or senses that do not exist in the everyday language. They may also contain proper names and common abbreviations that are used in the domain. For instance, a company could expand CD as Compact Disc, PO as Purchase Order instead of Post Office or Project Officer.

It is worth noting that linguistic resources are introduced in order to deal with synonyms (the fact that matching entities are named differently). By increasing the interpretation (sense) of words, they increase the chances of finding the matching terms (true positives). On the other side this also increases homonyms (the fact that more words are available for naming the matching entities) and the chances to match non matching terms (false positives). Dealing with this problem is known as word sense disambiguation [Lesk, 1986, Ide and Véronis, 1998]. Word sense disambiguation tries to restrict the candidate senses (and the candidate matches) from the context, especially by selecting the senses in relation to the other associated words and their senses.

We illustrate the use of external resources with the help of WordNet[5] [Miller, 1995, Fellbaum, 1998]. WordNet is an electronic lexical database for English (it has been adapted to other languages, see for instance EuroWordNet[6]), based on the notion of *synsets* or sets of synonyms. A synset denotes a concept or a sense of a group of terms. WordNet also provides an hypernym (superconcept/subconcept) structure as well as other relations such as meronym (*part of* relations). It also provides textual descriptions of the concepts (*gloss*) containing definitions and examples. We will denote WordNet as a partially ordered synonym resource.

Definition 4.20 (Partially ordered synonym resource). *A partially ordered synonym resource Σ over a set of words W, is a triple $\langle E, \leq, \lambda \rangle$, such that $E \subseteq 2^W$ is a set of synsets, $\leq$ is the hypernym relation between synsets and λ is a function from synsets to their definition (a text that is considered here as a bag of words in W). For a term t, $\Sigma(t)$ denotes the set of synsets associated with t.*

Example 4.21 (WordNet entry). We reproduce here the WordNet (version 2.0) entry for the word author. Each sense is numbered in superscript:

> author^1 *noun*: Someone who originates or causes or initiates something; *Example* 'he was the generator of several complaints'. *Synonym* generator, source. *Hypernym* maker. *Hyponym* coiner.
> author^2 *noun*: Writes (books or stories or articles or the like) professionally (for pay). *Synonym* writer^2. *Hypernym* communicator. *Hyponym* abstractor, alliterator, authoress, biographer, coauthor, commentator, contributor, cyberpunk, drafter, dramatist, encyclopedist, essayist, folk writer, framer, gagman, ghostwriter, Gothic romancer, hack, journalist, libretist, lyricist, novelist, pamphleter, paragrapher, poet, polemist, rhymer, scriptwriter, space writer, speechwriter, tragedian, wordmonger, word-painter, wordsmith, Andersen, Assimov...
> author^3 *verb.*: Be the author of; *Example* 'She authored this play'. *Hypernym* write. *Hyponym* co-author, ghost.

This resembles a traditional dictionary entry apart from the *Hypernym* and *Hyponym* features and the explicit mention of the considered sense. The hypernym relations for the senses of the words creator, writer, author, illustrator, and person are presented in Fig. 4.1.

There are at least three families of methods for using WordNet as a resource for matching terms used in ontology entities:

[5] http://wordnet.princeton.edu
[6] http://www.illc.uva.nl/EuroWordNet/

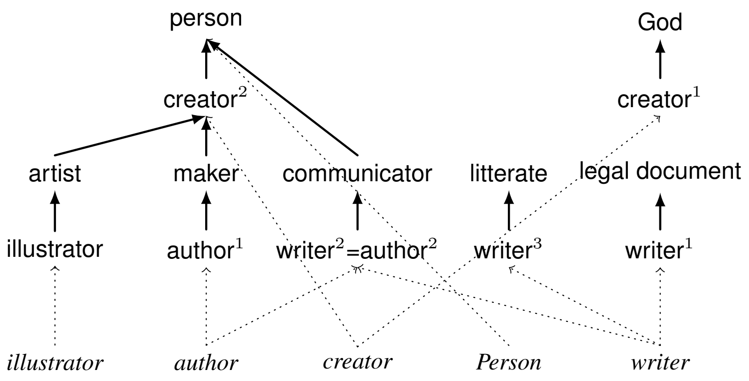

Fig. 4.1. The fragment of the WordNet hierarchy (limited to nouns) dealing with author, writer, creator, illustrator and person.

- considering that two terms are similar because they belong to some common synset;
- taking advantage of the hypernym structure for measuring the distances between synsets corresponding to two terms;
- taking advantage of the definitions of concepts provided by WordNet in order to evaluate the distance between the synsets associated with two terms.

A matcher based on WordNet can be designed by translating the (lexical) relations provided by WordNet to logical relations according to the following rules [Giunchiglia *et al.*, 2004]:

- $t \sqsubseteq t'$, if t is a hyponym or meronym of t'. For example, author is a hyponym of creator, therefore we can conclude that author $\sqsubseteq$ creator.
- $t \sqsupseteq t'$, if t is a hypernym or holonym of t'. For example, Europe is a holonym of France, therefore we can conclude that Europe $\sqsupseteq$ France.
- $t = t'$, if they are connected by synonymy relation or they belong to one synset. For example, writer and author are synonyms, therefore we can conclude that writer $=$ author.
- $t \perp t'$, if they are connected by antonymy relation or they are the siblings in the *part of* hierarchy. For example, Italy and France are siblings in the WordNet *part of* hierarchy, therefore we can conclude that Italy $\perp$ France.

Simple measures can be defined here (we only consider synonyms because they are the basis of WordNet synsets but other relationships can be used as well). The simplest use of synonyms is as follows:

Definition 4.22 (Synonymy similarity). *Given two terms s and t and a synonym resource Σ, the synonymy is a similarity $\sigma : \mathbb{S} \times \mathbb{S} \longrightarrow [0\ 1]$ such that:*

$$\sigma(s,t) = \begin{cases} 1 & \textit{if } \Sigma(s) \cap \Sigma(t) \neq \emptyset \\ 0 & \textit{otherwise} \end{cases}$$

This would consider that the similarity between author and writer is maximal (1.) and that between author and creator is minimal (0.).

Example 4.23 (Synonymy). The synonymy similarity between illustrator, author, creator, Person, and writer is given by the following table:

	illustrator	author	creator	Person	writer
illustrator	1.	0.	0.	0.	0.
author	0.	1.	0.	0.	1.
creator	0.	0.	1.	0.	0.
Person	0.	0.	0.	1.	0.
writer	0.	1.	0.	0.	1.

This strict exploitation of synonyms does not allow analysis of how far non synonymous objects are nor how close synonymous objects are. Since synonymy is a relation, all the measures on the graph of relations can be used on WordNet synonyms. Another measure computes the cosynonymy similarity.

Definition 4.24 (Cosynonymy similarity). *Given two terms s and t and a synonym resource Σ, the cosynonymy is a similarity $\sigma : \mathbb{S} \times \mathbb{S} \longrightarrow [0\ 1]$ such that:*

$$\sigma(s,t) = \frac{|\Sigma(s) \cap \Sigma(t)|}{|\Sigma(s) \cup \Sigma(t)|}$$

Example 4.25 (Cosynonymy similarity). The synonymy similarity between illustrator, author, creator, Person, and writer is given by the following table:

	illustrator	author	creator	Person	writer
illustrator	1.	0.	0.	0.	0.
author	0.	1.	0.	0.	.25
creator	0.	0.	1.	0.	0.
Person	0.	0.	0.	1.	0.
writer	0.	.25	0.	0.	1.

Some elaborate measures take into account that the terms can be part of several synsets and use a measure in the hyponym/hypernym hierarchy between synsets. A simple measure, known as edge-count, counts the number of edges separating two synsets in Σ (or the structural topological dissimilarity, see Sect. 4.3.2). More elaborate measures weight edge count with the position of synsets in the hierarchy. In particular, a measure developed specifically for WordNet is the one proposed by Wu and Palmer. It is presented in Sect. 4.3.2 because the hierarchy is, in this respect, similar to a class hierarchy. All measures defined in Sect. 4.3.2 can be used on the WordNet hypernym graph.

Other measures rely on an information theoretic perspective. They are based on the assumption that the most probable a concept, the less information it carries. So the information content of a concept is inverse to its probability of occurence. In the similarity proposed in [Resnik, 1995, Resnik, 1999], each synset (c) is associated with a probability of occurrence ($\pi(c)$) of an instance of the concept in a particular corpus. Usually, $\pi(c)$ is the the sum of the synset word occurrences divided by the total number of concepts. This probability is obtained from a corpus study. It is such that the more specific the concept, the lower its probability. The Resnik semantic similarity between two terms is a function of the more general synset common to both terms. It considers the maximum information content (or entropy), of the possible such synsets, taken as the negation of the logarithm of the probability of occurence.

Definition 4.26 (Resnik semantic similarity). *Given two terms s and t and a partially ordered synonym resource $\Sigma = \langle E, \leq, \lambda \rangle$ provided with a probability measure π, Resnik semantic similarity is a similarity $\sigma : \mathbb{S} \times \mathbb{S} \longrightarrow [0\ 1]$ such that:*

$$\sigma(s,t) = \max_{k; \exists c,c' \in E; s \in c \wedge t \in c' \wedge c \leq k \wedge c' \leq k} (-log(\pi(k)))$$

We do not provide examples of corpus-based similarity because the results are dependent on the corpus on which it is based (here for defining π). Examples of such measures based on the Brown corpus[7] are given in [Budanitsky and Hirst, 2006].

This measure uses the maximum, but one could have chosen instead an average or a sum of all the pairs of synsets associated with the two terms.

Other information-theoretic similarities depend on the increase of the information content measure from the terms to their common hypernyms instead of the shared information content. This is the case in the Lin information-theoretic similarity [Lin, 1998]. This method specifies the probabilistic degree of overlap between two synsets:

Definition 4.27 (Information-theoretic similarity). *Given two terms s and t and a partially ordered synonym resource $\Sigma = \langle E, \leq, \lambda \rangle$ provided with a probability π, Lin information theoretic similarity is a similarity $\sigma : \mathbb{S} \times \mathbb{S} \longrightarrow [0\ 1]$ such that:*

$$\sigma(s,t) = \max_{k; \exists c,c' \in \Sigma; s \in c \wedge t \in c' \wedge c \leq k \wedge c' \leq k} \frac{2 \times log(\pi(k))}{log(\pi(s)) + log(\pi(t))}$$

These similarities are not normalised.

A final way to compare terms found in strings through a thesaurus, like WordNet, is to use the definition (gloss) given to these terms in WordNet. In this case, any dictionary entry $s \in \Sigma$ is identified by the set of words corresponding to $\lambda(s)$. Then any measure defined in Sect. 4.2.1 can be used for comparing the strings [Lesk, 1986].

Definition 4.28 (Gloss overlap). *Given a partially ordered synonym resource $\Sigma = \langle R, \leq, \lambda \rangle$, the gloss overlap between two strings s and t is defined by the Jaccard similarity between their glosses:*

$$\sigma(s,t) = \frac{|\lambda(s) \cap \lambda(t)|}{|\lambda(s) \cup \lambda(t)|}$$

[7] http://nora.hd.uib.no/icame/

Example 4.29 (Gloss overlap). For computing the gloss overlap similarity between illustrator, author, creator, Person, and writer, we used the following treatments: take gloss for all senses and add the term name; suppress quotations ('...'); suppress empty words (or, and, the, a, an, for, of, etc.); suppress technical vocabulary, e.g., 'term'; suppress empty phrases, e.g., 'usually including'; keep categories, e.g., law; stem words. The gloss of author is given in Example 4.21.

The results have been taken as sets (not bags, so there is no repetition) of words and syntactically compared, yielding the following table:

	illustrator	author	creator	Person	writer
illustrator	1.	0.05	0.07	0.	0.02
author	0.05	1.	0.	0.	0.19
creator	0.07	0.	1.	0.06	0.02
Person	0.	0.	0.06	1.	0.04
writer	0.02	0.19	0.02	0.04	1.

This result is consistent with the previous measures since the only previously matching pair (author-writer) is still the highest scorer. This measure introduces new relations such as creator-illustrator, but still does not find the (possible) relation between creator and author. This is entirely related to the quality of glosses in WordNet.

Another example of building a matcher by using (WordNet) glosses includes counting the number of occurrences of the label of the source input sense in the gloss of the target input sense. If this number is equal to a threshold, e.g., 1, the less general relation can be returned. The reason for returning the less general relation is due to a common pattern of defining terms in glosses through a more general term. For example, in WordNet creator is defined as 'a person who grows or makes or invents things'. Thus, following this strategy we could find that creator $\sqsubseteq$ person. Some other variations of gloss-based matchers include considering glosses of the parent (children) nodes of the input senses in the WordNet *is a (part of)* hierarchy [Giunchiglia and Yatskevich, 2004]. The relations produced by these matchers depend heavily on the context of the matching task, and therefore, these matchers cannot be applied in all the cases [Giunchiglia *et al.*, 2006c].

Summary on linguistic methods

Many methods presented in this section have been implemented in the Perl package[8] WordNet::similarity [Pedersen *et al.*, 2004] and the Java package SimPack[9] (see Table 4.3). They have been thoroughly compared in [Budanitsky and Hirst, 2006].

Linguistic resources, such as stemmers, part-of-speech taggers, lexicons, and thesauri are invaluable resources since they allow the interpretation of the terms used in the expressions of ontologies. They provide a more accurate apprehension of these labels.

[8] http://wn-similarity.sourceforge.net/

[9] http://www.ifi.unizh.ch/ddis/simpack.html

Table 4.3. List of language measures based on WordNet and available in the wn-similarity Perl package and the SimPack Java package (some measures have not been presented yet).

WordNet::similarity	SimPack
Resnik	Resnik
Jiang–Conrath (1997)	
Lin	Lin
Leacock–Chodorow	Leacock–Chodorow
Hirst–St.Onge ([Saint-Onge, 1995])	
Edge count	Edge count
Wu–Palmer	Wu–Palmer
Extended Gloss Overlap	
Vector on gloss	

However, whenever the adequate resources are available for some language, they mainly open new possible matches between entities because they recognise that two terms can denote the same concept. Unfortunately, since they also recognise that the same term may denote several concepts at once, these techniques provide many possible matches from which to choose.

One way to choose among these representations is to take into account the structure of ontology entities in order to select the most coherent matches.

4.3 Structure-based techniques

The structure of entities that can be found in ontologies can be compared, instead of or in addition to comparing their names or identifiers.

This comparison can be subdivided into a comparison of the internal structure of an entity, i.e., besides its name and annotations, its properties or, in the case of OWL ontologies, the properties which take their values in a datatype, or the comparison of the entity with other entities to which it is related. The former is called internal (§4.3.1) and the latter is called relational structure (§4.3.2). The internal structure is the definition of entities without reference to other entities; the relational structure is the set of relations that an entity has with other entities. As expected, the internal structure is primarily exploited in database schema matching, while the relational structure is more important in matching formal ontologies and semantic networks.

4.3.1 Internal structure

Internal structure based methods are sometimes referred to as constraint-based approaches in the literature [Rahm and Bernstein, 2001]. These methods are based on the internal structure of entities and use such criteria as the set of their properties, the range of their properties (attributes and relations), their cardinality or multiplicity, and the transitivity or symmetry of their properties to calculate the similarity between them.

Entities with comparable internal structures or properties with similar domains and ranges in two ontologies can be numerous. For that reason, these kinds of methods are commonly used to create correspondence clusters rather than to discover accurate correspondences between entities. They are usually combined with other element-level techniques, such as terminological methods, and are responsible for reducing the number of candidate correspondences. They can be used with other approaches as a preprocessing step to eliminate most of the properties that are clearly incompatible.

For illustrating these methods we consider the properties associated with the Product and Volume entities in the example of Fig. 4.2 (the expected correspondences are given in Fig. 2.9, p. 48).

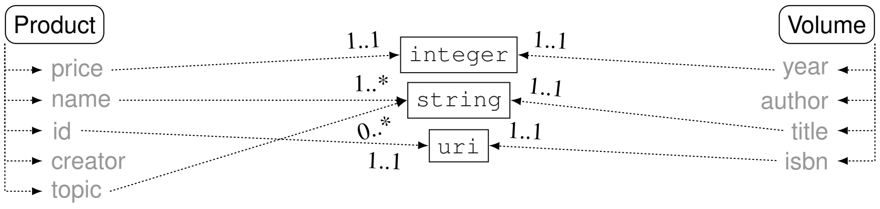

Fig. 4.2. Two sets of properties to be compared.

If we start from the elements of Fig. 4.2, there is no chance that pure terminological similarity methods find them very similar, though year and creator may appear the same to some edit distance methods. A linguistic method may be better able to find a relationship between creator and author.

Comparing the internal structure of ontology entities amounts to comparing their properties and composing the obtained result: the system can evaluate the similarity between all components considered next (names, keys, datatypes, domains, cardinalities) or multiplicities and combine the results. The combination operation is considered in Sect. 5.2, we focus here on the elementary comparison.

Property comparison and keys

In database schemas, unlike in formal ontologies, tables are provided with keys: a combination of properties whose values uniquely identify an object. For a Book, it would typically be the international standard book number (isbn), for a Person it can be his or her name, birth place and date.

This information is primarily very useful for recognising that two individuals are the same. Thus, keys are mostly used in extensional methods as a means to identify individuals and then apply methods on common set of instances (§4.4).

However keys can also be used for identifying classes: two classes identified in the same way are likely to represent the same set of objects. Moreover, even if two schemas use different keys for the same class, e.g., identifying Person with a social security number, there can be secondary keys that perform the same functions, e.g., that the social security number is also considered a key in the other class. So, when provided with keys, if they are highly compatible (similar names and types), it is plausible that the classes are equivalent.

For instance, if Product has id as a key and Volume has isbn as a key, it can be considered that these properties should correspond in case where the classes are the same. This can be considered possible because both properties have the same type (uri).

Datatype comparison

Property comparison involves comparing the property datatype (in OWL, this can be the range of the relation or a Restriction applied to the property in the class). Contrary to objects that require interpretations, datatypes can be considered objectively and it is possible to determine how close a datatype is to another (ideally this can be based on the interpretation of datatypes as sets of values and the set-theoretic comparison of these datatypes [Valtchev, 1999, Valtchev and Euzenat, 1997]).

We distinguish here between a datatype, which corresponds to the way the values are stored in a computer (like integer, float, string or uri), and a domain, which characterises a subset of a particular datatype (like [10 12] or '*book'). Datatypes are considered here and domains are addressed in the next section.

Datatypes are not fully disjoint, though there are rules by which an object of one type can be thought of as an object of another type and rules by which a value of some type can be converted in the memory representation of another type (known as *casting* in programming languages).

Ideally, the proximity between datatypes should be maximal when these are the same types, lower when the types are compatible (for instance, integer and float are compatible since they can be cast one into the other) and the lowest when they are non compatible. In addition, domain comparison should ideally be based on datatype comparison and the comparison of the sets of values covered by these domains. The compatibility between property datatypes can be assessed by using an underlying table lookup. An example of a part of such a table is given in Table 4.4.

Table 4.4. Part of a datatype compatibility table.

	char	fixed	enumeration	int	number	string
string	0.7	0.4	0.7	0.4	0.5	1.0
number	0.6	0.9	0.0	0.9	1.0	0.5

Such a table can be extracted, for languages like OWL, from the type hierarchy of XML Schema datatypes (see Fig. 4.3). In the example of Fig. 4.2, it can be considered that since a uri is a subclass of string, the isbn may be related to name.

Example 4.30 (Datatype comparison). In the example of Fig. 4.2, data type comparison would let us match price with year, both name and topics with title, and id with

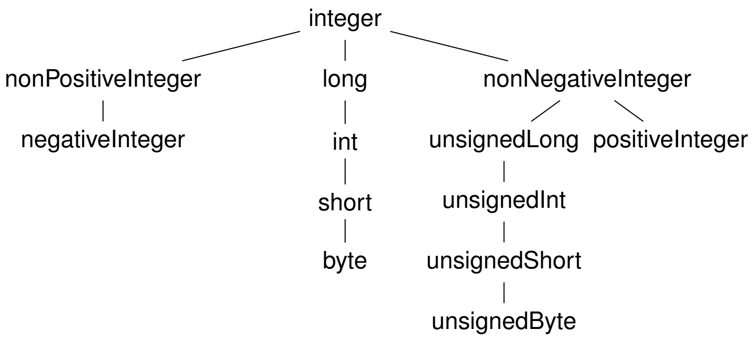

Fig. 4.3. Fragment of the XML Schema datatype hierarchy [Biron and Malhotra (ed.), 2004].

isbn. creator and author are left aside because they are object-valued properties. This comparison yields interesting results since it finds the expected matches. However, it also finds incorrect ones (price-year and topics-title) so these methods cannot be used in isolation.

Domain comparison

Depending on the entities to be considered, what can be reached from a property can be different: in classes these are domains while in individuals these are values. Moreover, they can be structured in sets or sequences. It is thus important to consider this fact in the comparison.

[Valtchev, 1999] proposes a framework in which the types or domains of properties must be compared on the basis of their interpretations: sets of values. Type comparison is based on their respective size, in which the size of a type is the cardinality or multiplicity of the set of values it defines. The distance between two domains is then given by the difference between their size and that of their common generalisation. This measure is usually normalised by the size of the largest possible distance attached to a particular datatype. We give here an instance of this type of measure.

Definition 4.31 (Relative size distance). *Given two domain expressions e and e' over a datatype τ, the relative size distance $\delta : 2^\tau \times 2^\tau \longrightarrow [0\ 1]$, is as follows:*

$$\delta(e, e') = \frac{|gen_\tau(e \vee e')| - |gen_\tau(e \wedge e')|}{|\tau|},$$

such that $gen_\tau(.)$ provides the generalisation of a type expression and $\vee$ and $\wedge$ correspond to the union and intersection of the types.

Example 4.32 (Relative size distance). Consider a property age in one class to be compared with the property age of three other classes (schoolchild, teenager and grown-up). The first property has a domain of $[6\ 12]$, while the others have respective domains expressed by: $[7\ 14]$, $[14\ 22]$ and ≥ 10. All these properties have datatype integer. The generalisation of these four domains are the domains themselves, the union with $[6\ 12]$ is respectively $[6\ 14]$, $[6\ 22]$, $[6\ +\infty[$, and the intersection is respectively $[7\ 12]$, $\emptyset$, and $[10\ 12]$. As a consequence, the distance will be respectively $3/|\tau|$, $17/|\tau|$ and $|\tau|-3/|\tau|$. This corresponds to some intuition that the distance between domains depends on the difference between the values they cover in isolation and in common.

There are three advantages of this measure. The most obvious one is that it is normalised. The second one is that it is totally general (it is not expressed in terms of integers). The third one is that it can easily be mapped to the usual measures that are often used.

Usually, a common generalisation depends on the type: it is a set for enumerated types and an interval for ordered types (it can also be a set of intervals). In the case of dense types, the size of a domain is the usual measure of its size (Euclidean distance can be used for real or floating point numbers). The case of infinite types has to be taken adequately (by evaluating the largest possible domain in a computer or by normalising with regard to the actual corpus) [Valtchev, 1999]. Normalising over the largest distance in the corpus, if possible, is often a good idea. Indeed, it is not reasonable, for example, to normalise the age of people with that of planets or their size even if they use the same unit. Another advantage of this framework is that it encompasses value comparisons which can be considered as singletons and compared with domains if necessary.

Comparing multiplicities and properties

Properties can be constrained by multiplicities (as they are called in UML). Multiplicities are the acceptable cardinalities of the set of values of a property (for a given object). Similar to compatibilities between datatypes, compatibility between cardinalities can be established based on a table look-up. An example of such a table for DTDs is given in Table 4.5, following the work in [Lee *et al.*, 2002].

Table 4.5. A cardinality compatibility table.

	*	+	?	none
*	1.0	0.9	0.7	0.7
+	0.9	1.0	0.7	0.7
?	0.7	0.7	1.0	0.8
none	0.7	0.7	0.8	1.0

In OWL, cardinalities or multiplicities are expressed through the minCardinality, maxCardinality and cardinality restrictions. Multiplicities can be expressed as an interval of the set of positive integers $[0\ +\infty[$. As such they are domains of the integer type. Two multiplicities are compatible if the intersection of the corresponding intervals is non empty. Any measure on the integer datatype can be used for assessing the similarity between multiplicities (see previous paragraph). However, in this case we choose a simpler distance inspired from the Jaccard similarity.

Values can be collected by a particular construction (set, list, multiset) on which cardinality constraints are applied. Again, it is possible to compare these constructed datatypes by comparing (i) the datatypes on which they are constructed and (ii) the cardinalities that are applied to them. For instance, sets of 2 and 3 children are closer to a set of 3 people than to a set of 10–12 flowers (if children are people). This technique is used in [Euzenat and Valtchev, 2004].

Definition 4.33 (Multiplicity similarity). *Given two multiplicity expressions* $[b\ e]$ *and* $[b'\ e']$*, the multiplicity similarity is a similarity between non negative integer intervals* $\sigma : 2^{\mathcal{T}} \times 2^{\mathcal{T}} \rightarrow [0\ 1]$*, such that:*

$$\sigma([b\ e],[b'\ e']) = \begin{cases} 0 & \text{if } b' > e \text{ or } b > e' \\ \dfrac{\min(e,e') - \max(b,b')}{\max(e,e') - \min(b,b')} & \text{otherwise} \end{cases}$$

For instance, if we have to compare multiplicity $[0\ 6]$ with $[2\ 8]$, $[8\ 12]$ and $[0\ +\infty]$, the comparison will respectively yield .5, 0. and $6/MAXINT$ (the latter is very low but remains non null because it is compatible with the initial multiplicity).

Example 4.34 (Multiplicity comparison). In the example of Fig. 4.2, multiplicity comparison can be used to further match id with isbn because they will both have a cardinality of $[1\ 1]$ and, unfortunately, will match price with year as well. However, is can also be used to prefer matching name rather than topic to title because they have the same multiplicities ($[1\ +\infty]$ instead of $[0\ +\infty]$).

Other features

Other internal structural factors have been considered in database schema matching. Since these are internal features, they can be very dependent on the knowledge model. For example, the work in [Navathe and Buneman, 1986] discusses such additional property characteristics as uniqueness, static semantic integrity constraints, dynamic semantic integrity constraints, security constraints, allowable operations and scale.

It is also possible in some languages to consider collection constructors, e.g., Set, List, Bag or multiset, Array, and their compatibility. It is then necessary to compare sets or lists of objects, e.g., the sequence of topics or the set of authors of a Book. In this case, general techniques can be used for assessing the similarity or distance between these sets depending on the similarity applying to the type of their elements. Concerning sets, these methods will be presented in Sect. 4.4.1 in the context of extension comparison. Concerning sequences, they can be adapted from some of the measures that have been presented in Sect. 4.2.1 which have considered strings as sequences of characters and paths as sequences of strings. In addition, Sect. 5.3.2 explains how to compare sets of objects with similarities.

In [Ehrig and Sure, 2004], it is proposed that the definition of a set of rules can be used for determining similarity between ontology entities. They point out that some features from OWL related to internal structure, such as symmetry and restrictions of values, could be used, but are discarded at the moment, as they do not have any wide distribution.

Summary on internal structure

Internal structure, including the names of entities, is very important for matching because it provides a basis on which algorithms can rely. The techniques for comparing them are efficient and easy to implement.

However, the internal structure does not provide much information on the entities to compare: many very different types of objects can have properties with the same datatypes. On the one hand, they can be used for eliminating incompatible correspondences and promoting compatible ones. On the other hand, it is always possible that different models of a concept use different, and incompatible, types. For these reasons, internal structure comparisons must always be used jointly with other techniques.

4.3.2 Relational structure

An ontology can be considered to be a graph whose edges are labelled by relation names (mathematically speaking, this is the graph of the multiple relations of the ontology: $\leq$, $\in$, $\perp$, :, $=$). Finding the correspondences between elements of such graphs corresponds to solving a form of the graph homomorphism problem [Garey and Johnson, 1979]. Namely it can be related to finding a maximum common directed subgraph.

Definition 4.35 (Maximum common directed subgraph problem). *Given two directed graphs $G = \langle V, E\rangle$ and $G' = \langle V', E'\rangle$, does there exist $F \subseteq E$ and $F' \subseteq E'$ and a pair of functions $f : V \to V'$ and $f^{-1} : V' \to V$ such that:*

- $\forall \langle u, v\rangle \in E|_F$, $\langle f(u), f(v)\rangle \in E'|_{F'}$;
- $\forall \langle u', v'\rangle \in E'|_{F'}$, $\langle f^{-1}(u'), f^{-1}(v')\rangle \in E|_F$;
- $\forall u \in V|_F$, $f^{-1}(f(u)) = u$;
- $\forall u' \in V'|_{F'}$, $f(f^{-1}(u')) = u'$;
- *there is no other $F \subseteq H \subseteq E$ and $F' \subseteq H' \subseteq E'$ satisfying these properties.*

Note that graph matching is another type of problem which is presented in Sect. 5.7.3.

In ontology matching, the problem is encoded as an optimisation problem (finding the isomorphic subgraphs minimising some distance like the dissimilarity between matched objects or maximising similarity). These subgraphs do not have to be maximal. Moreover, the problem is very often adapted for multipartite graphs separating classes from properties.

The similarity comparison between two entities from two ontologies can be based on the relations of these entities with the other entities in the ontologies: the more two entities are similar, the more their related entities should be alike. This remark can be exploited in several ways depending on the kind of relations considered. Moreover, given the transitive nature of some relations, it is natural to extend this remark through transitivity. Roughly, for each pair of relations, we can come up with 5 different ways of comparing the relations [Euzenat *et al.*, 2004a]:

r comparing the entities in direct relation through r;
r^- comparing the entities in the transitive reduction of relation r;
r^+ comparing the entities in the transitive closure of relation r;
r^{-1}comparing the entities coming through a relation r;
$r\uparrow$ comparing entities which are ultimately in r^+ (the maximal elements of the closure).

These relations are exemplified as follows:

Example 4.36 (Exploiting relations in an ontology). Given the left-hand ontology of Fig. 2.7, the relations based on subClass from Book are as follows:

$$\begin{aligned} subclass(\mathsf{Book}) = subclass^-(\mathsf{Book}) &= \{\mathsf{Science}, \mathsf{Pocket}, \mathsf{Children}\} \\ subclass^+(\mathsf{Book}) &= \{\mathsf{Science}, \mathsf{Pocket}, \mathsf{Textbook}, \mathsf{Popular}, \mathsf{Children}\} \\ subclass^{-1}(\mathsf{Book}) &= \{\mathsf{Product}\} \\ subclass\uparrow(\mathsf{Book}) &= \{\mathsf{Textbook}, \mathsf{Popular}, \mathsf{Pocket}, \mathsf{Children}\} \end{aligned}$$

Table 4.6 displays the different ways of comparing two ontology entities based on their relations with other entities. Of course, an approach can combine several of the above criteria [Mädche and Staab, 2002, Euzenat and Valtchev, 2004, Bach *et al.*, 2004].

As can be observed from Table 4.6, some features have type String and can be compared with the techniques proposed in Sect. 4.2.1. However, those with type Class or Property really induce a graph structure. Moreover, the values which are labelled by Set(·) are more difficult to deal with because this means that many edges labelled by the feature will appear in the graph. The last part of the table is, in fact, relevant to the extensional methods that will be presented in Sect. 4.4.

There are three types of relations that have been considered so far in relational structure techniques: taxonomic relations, mereologic relations and all the involved relations. These are considered below.

Taxonomic structure

The taxonomic structure, i.e., the graph made with the subClassOf relation, is the backbone of ontologies. For this reason, it has been studied in detail by researchers and is very often used as a comparison source for matching classes.

Table 4.6. Features on which comparison of ontology entities can be made. The table reads: Two *Entities* are similar if their *Features* are similar. This table is an adapted version of tables reported in [Ehrig, 2007], [Euzenat *et al.*, 2004a] and [Euzenat and Valtchev, 2004].

Entity	Feature	OWL	Type
Class	name	rdf:label	String
	id	rdf:ID	String
	comments	rdf:comment	String
	same classes	owl:sameClassAs	Set(Class)
	properties	*property*	Set(Property)
	ultimate properties	*property*$\uparrow$	Set(Property)
	direct superclasses	owl:subClassOf$^-$	Set(Class)
	direct subclasses	owl:subClassOf^{-1-}	Set(Class)
	superclasses	owl:subClassOf*	Set(Class)
	subclasses	owl:subClassOf^{-1*}	Set(Class)
	ultimate subclasses	owl:subClassOf$^{-1}\uparrow$	Set(Class)
	direct instances	rdf:type^{-1*}	Set(Individual)
	instances	rdf:type^{-1-}	Set(Individual)
Property	name	rdf:label	String
	id	rdf:ID	String
	comments	rdf:comment	String
	same properties	owl:samePropertyAs	Set(Property)
	domain/range	rdfs:domain/rdfs:range	Class
	direct superproperties	rdfs:subProperty$^-$	Set(Property)
	direct subproperties	rdfs:subProperty^{-1-}	Set(Property)
	superproperties	rdfs:subProperty*	Set(Property)
	subproperties	rdfs:subProperty^{-1*}	Set(Property)
Individual	name	rdf:label	String
	id	rdf:ID	String
	comments	rdf:comment	String
	same individuals	owl:sameAs	Set(Instance)
	direct classes	rdf:type$^-$	Set(Class)
	classes	rdf:type*	Set(Class)
	properties	*property*	Set(Property)

There have been several measures proposed for comparing classes based on the taxonomic structure. The most common ones are based on counting the number of edges in the taxonomy between two classes. The structural topological dissimilarity on a hierarchy [Valtchev and Euzenat, 1997] follows the graph distance, i.e., the shortest path distance in a graph taken here as the transitive reduction of the hierarchy.

Definition 4.37 (Structural topological dissimilarity on hierarchies). *The structural topological dissimilarity $\delta : o \times o \to \mathbb{R}$ is a dissimilarity over a hierarchy $H = \langle o, \leq \rangle$, such that:*

$$\forall e, e' \in o, \delta(e, e') = \min_{c \in o}[\delta(e, c) + \delta(e', c)]$$

where $\delta(e, c)$ is the number of intermediate edges between an element e and another element c.

This corresponds to the unit tree distance of [Barthélemy and Guénoche, 1992], i.e., with weight 1 on each edge. This function can be normalised by the maximal length of a path between two classes in the taxonomy:

$$\overline{\delta}(e, e') = \frac{\delta(e, e')}{\max_{c,c' \in o} \delta(c, c')}$$

Example 4.38 (Structural topological dissimilarity). We provide the examples of this section based on the taxonomy in Fig. 4.1. We consider that each term corresponds to a class (all senses are considered together) and there exists a top of the hierarchy (on top of Person, litterate, legal document and God).

	illustrator	author	creator	Person	writer
illustrator	0.	.8	.4.	.6	1.
author	.8	0.	.4	.6	0.
creator	.4	.4	0.	.2	.6
Person	.6	.6	.2	0.	.4
writer	1.	0.	.6	.4	0.

Again, this corroborates the WordNet data that the closest classes are writer and author.

The results given by such a measure are not always semantically relevant since a long path in a class hierarchy can often be summarised as an alternative short one.

A similar measure is the one of Leacock–Chodorow [Leacock *et al.*, 1998] which is function of the length of the shortest path. It has been introduced for lexicographic taxonomies (§4.2.2). A more elaborate distance of this kind is known as the Wu–Palmer similarity [Wu and Palmer, 1994]. This distance takes into account the fact that two classes near the root of a hierarchy are close to each other in terms of edges but can be very different conceptually, while two classes under one of them which are separated by a larger number of edges should be closer conceptually.

Definition 4.39 (Wu–Palmer similarity). *The Wu–Palmer similarity $\sigma : o \times o \longrightarrow \mathbb{R}$ is a similarity over a hierarchy $H = \langle o, \leq \rangle$, such that:*

$$\sigma(c, c') = \frac{2 \times \delta(c \wedge c', \rho)}{\delta(c, c \wedge c') + \delta(c', c \wedge c') + 2 \times \delta(c \wedge c', \rho)}$$

where ρ is the root of the hierarchy, $\delta(c, c')$ is the number of intermediate edges between a class c and another class c' and $c \wedge c' = \{c'' \in o; c \leq c'' \wedge c' \leq c''\}$.

Example 4.40 (Wu–Palmer similarity). The Wu–Palmer similarity also provides a figure in coherence with WordNet structure.

	illustrator	author	creator	Person	writer
illustrator	1.	.5	.67	.4	.29
author	.5	1.	.67	.4	1.
creator	.67	.67	1.	.67	.4
Person	.4	0.4	.67	1.	.5
writer	.29	1.	.4	.5	1.

The upward cotopic similarity applies the Jaccard similarity to cotopies. It has been described in [Mädche and Zacharias, 2002] and is as follows:

Definition 4.41 (Upward cotopic similarity). *The upward cotopic similarity* $\sigma : o \times o \longrightarrow \mathbb{R}$ *is a similarity over a hierarchy* $H = \langle o, \leq \rangle$*, such that:*

$$\sigma(c, c') = \frac{|UC(c, H) \cap UC(c', H)|}{|UC(c, H) \cup UC(c', H)|}$$

where $UC(c, H) = \{c' \in H; c \leq c'\}$ *is the set of superclasses of c.*

Example 4.42 (Upward cotopic similarity). In this case, because all senses count in the cotopy (and not the closest one in terms of path), the result is different from other measures: creator benefits from its position as a superclass of author and illustrator for scoring better than the usual writer-creator pair because they have too many unrelated senses.

	illustrator	author	creator	Person	writer
illustrator	1.	.37	.43	.4	.18
author	.37	1.	.43	.29	.36
creator	.43	.43	1.	.4	.18
Person	.4	.29	.4	1.	.25
writer	.18	.36	.18	.25	1.

These measures cannot be applied as they are in the context of ontology matching since the ontologies are not supposed to share the same taxonomy H, but this can be used in conjunction with a resource of common knowledge, such as WordNet. For that purpose, it is necessary to develop these kinds of measures over a pair of ontologies. In [Valtchev, 1999, Euzenat and Valtchev, 2004], this amounts to using a (local) matching between the elements to be compared (for instance, the hierarchies).

Beside these global measures that take into account the whole taxonomy for assessing the similarity between classes, there are non global measures that have been used in the ontology matching contexts. These measures usually take advantage of the 'direct' part of Table 4.6. Below are some of these measures:

Super or subclass rules: These matchers are based on rules capturing the intuition that classes are similar if their super or subclasses are similar. For example, if superclasses are the same, the actual classes are similar to each other. If subclasses are the same, the compared classes are also similar [Dieng and Hug, 1998, Ehrig and Sure, 2004]. This technique has at least two drawbacks: (i) when there are several sub or superclasses, then, without care, they would all be mapped into the same one, so it is necessary to have some other discriminating features, and (ii) the similarity between the sub or super classes will rely in turn on that of their super or subclasses. This turns this problem into yet another global similarity problem.

Bounded path matching: Bounded path matchers take two paths with links between classes defined by the hierarchical relations, compare terms and their positions along these paths, and identify similar terms. This technique has been introduced in Anchor-Prompt (§6.1.9). For example, in Fig. 2.9, if Book corresponds to Volume and Popular corresponds to Autobiography, then the elements along the paths (Science on one side and Biography and Essay on the other side) must be carefully considered for correspondence. For instance, for deciding that Essay is more general than Science. This technique is primarily guided by two anchors of paths and uses alternative techniques for choosing the best match.

Mereologic structure

The second well known structure after the taxonomic structure is the mereologic structure, i.e., the structure corresponding to a *part-of* relationship. The difficulty for dealing with this kind of structure is that it is not easy to find the properties which carry a mereologic structure. For example, a class Proceedings can have some whole-part relations with a class InProceedings, but it will be expressed through a property communications. These InProceedings objects will in turn have a mereologic structure which is expressed through sections property.

However, if it is possible to detect the relations that support the part-of structure, this can be then used for computing similarity between classes: they will be more similar if they share similar parts. This is even more useful when comparing extensions of classes because it can be inferred that objects sharing the same set of parts will be the same.

Relations

Beside two previous kinds of relations, one can consider the general problem of matching entities based on all their relations. Classes are also related through the definitions of their properties (like author and creator in Fig. 4.2). These properties are also edges of a graph and if they are found similar, they can be used for finding that classes are similar. However, contrary to taxonomic and mereologic structures, the relation graph can contain circuits. How to handle these will be considered in Sect. 5.3. We consider here similarities.

The similarity between nodes can also be based on their relations. For example, in one of the possible ontology representations of schemas of Fig. 2.7, if the Book class is related to the Human class by the author relation in one ontology, and if the

Volume class is related to the Writer class by the author relation in the other ontology, then knowing that classes Book and Volumes are similar, and that relations author and author are similar, we can infer that Human and Writer may be similar too. The similarity among relations in [Mädche and Staab, 2002] is computed according to this principle.

This can be applied to a set of classes and a set of relations. It means that if we have a set of relations $r_1 \dots r_n$ in the first ontology which are similar to another set of relations $r'_1 \dots r'_n$ in the second ontology, it is possible that two classes, which are the domains of relations in those two sets, are similar too.

This principle can also be extended to the composition of relations, i.e., instead of considering only the relations asserted at a class, one can consider their composition with relations starting at the domain of this relation. For instance, instead of considering the author relation, one will consider the author·firstname, the author·lastname, or the author·nationality relations.

One of the problems of this approach is that it is based on the use of similarity of relations to infer the similarity of their domain classes or their range classes. This introduces circularity in the computation of similarity. There are several ways to overcome this circularity. As a first alternative, the similarity on relations can be based on their labels using techniques developed in Sect. 4.2.1. As a second alternative, if relations are organised in a taxonomy, then methods considered in the previous subsection can be used as well.

Finally, two extreme solutions, that use the relations for reaching nodes but not for actually matching, are considered by the following approaches:

Children. The similarity between nodes of the graph is computed based on similarity of their children nodes, that is, two non leaf entities are structurally similar if their immediate children sets are highly similar. A more complex version of this matcher is implemented in [Do and Rahm, 2002].

Leaves. The similarity between nodes of the graphs is computed based on similarity of leaf nodes, that is, two non leaf schema elements are structurally similar if their leaf sets are highly similar, even if their immediate children are not [Madhavan *et al.*, 2001, Do and Rahm, 2002]. This is very well adapted to comparing document schemas.

Summary on relational structure

Matching ontologies from their relational (or external) structure is very powerful because it allows all the relations between entities to be taken into account. This must be grounded on other tangible properties, which is why it is often used in combination with internal structural methods and terminological methods.

It is worth considering what are the important relations before using such techniques. The most commonly used structure is the taxonomy because it is the backbone of ontologies and has usually received a lot of attention from designers. In some fields, the mereology relations are as important as taxonomic ones. However, they are difficult to identify because contrary to the subClass relation, they can bear any other name.

The relational structure raises the problem of which part influences what: there is usually a mutual influence between each of the related parts. This is the reason why, beside the similarity equations used for comparing the entities, it is necessary to have an iterative algorithm. This is considered in Sect. 5.3.

4.4 Extensional techniques

When individual representations (or instances) are available, there is a very good opportunity for matching systems. When two ontologies share the same set of individuals, matching is highly facilitated. For example, if two classes share exactly the same set of individuals, then there can be a strong presumption that these classes represent a correct match.

Even when classes do not share the same set of individuals, these allow the grounding of the matching process on tangible indices which do not change easily. For instance, titles of Books do not have any reason to change. So if titles of Books are different, then these are most certainly not the same books. Then, matching can be again based on individual comparisons.

We thus divide extensional methods into three categories: those which apply to ontologies with common instance sets, those which propose individual identification techniques, before using the previous ones, and those which do not require identification, i.e., which work on heterogeneous sets of instances.

4.4.1 Common extension comparison

The easiest way to compare classes when they share instances is to test the intersection of their instance set A and B and to consider that these classes are very similar when $A \cap B = A = B$, more general when $A \cap B = B$ or $A \cap B = A$. The work in [Larson *et al.*, 1989, Sheth *et al.*, 1988] discussed how relationships and entity sets can be integrated primarily based on the set relations: equal ($A \cap B = A = B$), contains ($A \cap B = A$), contained-in ($A \cap B = B$), disjoint ($A \cap B = \emptyset$) and overlap. The problem is the ability to handle faults: small amounts of incorrect data may lead the system to draw a wrong conclusion on domain relationships. Moreover, the dissimilarity has to be 1 when none of these cases apply: for instance, if the classes have some instances in common but not all.

A way to refine this is to use the Hamming distance between two extensions: it corresponds to the size of the symmetric difference normalised by the size of the union.

Definition 4.43 (Hamming distance). *The Hamming distance between two sets is a disimilarity function $\delta : 2^E \times 2^E \to \mathbb{R}$ such that $\forall x, y \subseteq E$:*

$$\delta(x, y) = \frac{|x \cup y - x \cap y|}{|x \cup y|}$$

This version of the symmetric difference is normalised. Using such a distance in comparing sets is more robust than using equality: it tolerates some individuals being misclassified and can still produce a short distance.

It is also possible to compute a similarity based on the probabilistic interpretation of the set of instances. This is the case of the Jaccard similarity [Jaccard, 1901].

Definition 4.44 (Jaccard similarity). *Given two sets A and B, let $P(X)$ be the probability of a random instance to be in the set X. The Jaccard similarity is defined as follows:*

$$\sigma(A, B) = \frac{P(A \cap B)}{P(A \cup B)}$$

This measure is normalised and reaches 0 when $A \cap B = \emptyset$ and 1 when $A = B$. It can be used with two classes of different ontologies sharing the same set of instances.

Formal concept analysis

One of the tools of formal concept analysis (FCA) [Ganter and Wille, 1999] is the computation of the concept lattice. The idea behind formal concept analysis is the duality between a set of objects (here the individuals) and their properties: the more properties are constrained, the fewer objects satisfy the constraints. So a set of objects with properties can be organised in a lattice of concepts covering these objects. Each concept can be identified by its properties (the intent) and covers the individual satisfying these properties (the extent).

In ontology matching, the properties can simply be the classes to which the individuals are known to belong and the technique is independent from the origin of the entities, i.e., whether they come from the same ontology or not. From this data set, formal concept analysis computes the concept lattice (or Galois lattice). This is performed by computing the closure of the instances$\times$properties Galois connection. This operation starts with the complete lattice of the power set of extent (respectively, intent) and keeps only the nodes which are closed under the connection, i.e., starting with a set of properties, it determines the corresponding set of individuals, which itself provides a corresponding set of properties; if this set is the initial one, then it is closed and is preserved, otherwise, the node is discarded. The result is a concept lattice, like the one computed in Fig. 4.4 from the table.

For instance, let us start with the table of Fig. 4.4. The table displays a small set of instances and the classes they belong to (from both ontologies). The right-hand side of Fig. 4.4 displays the corresponding concept lattice. From this lattice the following correspondences can be extracted:

Science $=$ Essay	Science $\geq$ Biography	Essay $\geq$ Popular
Science $\geq$ Autobiography	Popular $=$ Biography	Popular $=$ Autobiography
Literature $\geq$ Pocket	Novel $=$ Pocket	

The result is not accurate. However, it is possible to weight these results by first eliminating the redundant correspondences and by providing a confidence according to the size of the extent covered by the correspondence.

	Book	Science	Popular	Pocket	Essay	Biography	Autobiography	Literature	Novel	Poetry
My life	√	√	√		√	√	√			
Logic	√	√			√					
La chute	√			√				√	√	
Mes propriétés	√							√		√

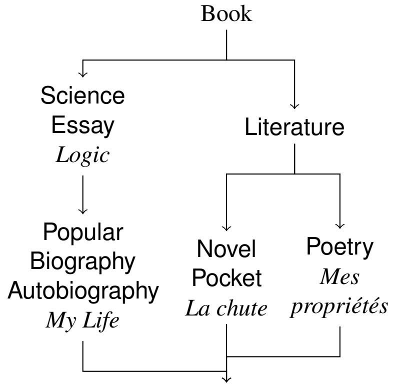

Fig. 4.4. A 'formal context' and the corresponding concept lattice.

4.4.2 Instance identification techniques

If a common set of instances does not exist, it is possible to try to identify which instance from one set corresponds to which other instance from the other set. This method is usable when one knows that the instances are the same. This works, for example, when integrating, two human resource databases of the same company, but does not apply for those of different companies or for databases of events which have no relations.

A first natural technique for identifying instances is to take advantage of keys in databases. Keys can be either internal to the database, i.e., generated unique surrogates, in which case they are not very useful for identification, or external identification, in which case there is high probability that these identification keys are present in both data sets (even if they are not present as keys). In such a case, if they are used as keys, we can be sure that they uniquely identify an individual (like isbn).

When keys are not available, or they are different, other approaches to determine property correspondences use instance data to compare property values. In databases, this technique has been known as record linkage [Fellegi and Sunter, 1969, Elfeky *et al.*, 2002] or object identification [Lim *et al.*, 1993]. They aim at identifying multiple representations of the same object within a set of objects. They are usually based on string-based and internal structure-based techniques (§4.2 and §4.3.1).

If values are not precisely the same but their distributions can be compared, it is possible to apply global techniques. This case is covered in the next section.

4.4.3 Disjoint extension comparison

When it is not possible to directly infer a dataset common to both ontologies, it is easier to use approximate techniques for comparing class extensions. These methods can be based on statistical measures about the features of class members, on the similarities computed between instances of classes or based on a matching between entity sets.

Statistical approach

The instance data can be used to compute some statistics about the property values found in instances, such as maximum, minimum, mean, variance, existence of null values, existence of decimals, scale, precision, grouping, and number of segments. This allows the characterising of the domains of class properties (§4.3.1) from the data. In practice, if dealing with statistically representative samples, these measures should be the same for two equivalent classes of different ontologies.

Example 4.45 (Statistical matching). Consider two ontologies with instances. The analysis of numerical properties size and weight in one ontology and hauteur and poids in the other reveals that they have different average values but the same coefficient of variation, i.e., standard deviation divided by mean, which, in turn, reveals comparable variability of size and hauteur on the one hand and weight and poids on the other hand. This is typically what happens when values are expressed in different units. The ratio of average values of size/hauteur is 2.54 and that of weight/poids is 28.35.

These values have been established based on the whole population. They can be used for comparing the statistical characteristics of these properties in the classes of the ontologies. For instance, the average value of the size property for the Pocket class significantly differs from that of the global population and, once divided by 28.35, is very close to that of the Livredepoche class (also differing from the whole population in the same manner). Hence, these two classes could be considered as similar.

Other approaches, like [Li and Clifton, 1994], propose methods that utilise data patterns and distributions instead of data values and domains. The result is a better fault tolerance and a lower time-consumption since only a small portion of data values are needed due to the employment of data sampling techniques. In general, applying internal structure methods to instances allows a more precise characterisation of the actual contents of schema elements, thus, more accurately determining corresponding datatypes based, for example, on the discovered value ranges and character patterns.

These methods have, however, one prerequisite: they work better if the correspondences between properties are known (otherwise they could match different properties on the basis of their domain). This is already a matching problem to be solved.

Similarity-based extension comparison

Similarity-based techniques do not require the classes to share the same set of instances, though they can still be applied in that case. In particular, the methods based on common extensions always return 0 when the two classes do not share any instances, disregarding the distance between the elements of the sets. In some cases, it is preferable to compare the sets of instances. This requires a (dis)similarity measure between the instances that can be obtained with the other basic methods.

In data analysis, the linkage aggregation methods allow the assessment of the distance between two sets whose objects are only similar. They thus allow us to compare two classes on the basis of their instances.

Definition 4.46 (Single linkage). *Given a dissimilarity function* $\delta : E \times E \longrightarrow \mathbb{R}$*, the single linkage measure between two sets is a disimilarity function* $\varDelta : 2^E \times 2^E \longrightarrow \mathbb{R}$ *such that* $\forall x, y \subseteq E$*,* $\varDelta(x, y) = \min_{(e,e') \in x \times y} \delta(e, e')$.

Definition 4.47 (Full linkage). *Given a dissimilarity function* $\delta : E \times E \longrightarrow \mathbb{R}$*, the complete linkage measure between two sets is a disimilarity function* $\varDelta : 2^E \times 2^E \longrightarrow \mathbb{R}$ *such that* $\forall x, y \subseteq E$*,* $\varDelta(x, y) = \max_{(e,e') \in x \times y} \delta(e, e')$.

Definition 4.48 (Average linkage). *Given a dissimilarity function* $\delta : E \times E \longrightarrow \mathbb{R}$*, the average linkage measure between two sets is a disimilarity function* $\varDelta : 2^E \times 2^E \longrightarrow \mathbb{R}$ *such that* $\forall x, y \subseteq E$*,* $\varDelta(x, y) = \frac{\sum_{(e,e') \in x \times y} \delta(e,e')}{|x| \times |y|}$.

Other linkage measures have been defined. Each of these methods has its own benefits, e.g., maximising shortest distance, minimising longest distance, minimising average distance. Another method from the same family is the Hausdorff distance measuring the maximal distance of a set to the nearest point in the other set [Hausdorff, 1914]:

Definition 4.49 (Hausdorff distance). *Given a dissimilarity function* $\delta : E \times E \longrightarrow \mathbb{R}$*, the Hausdorff distance between two sets is a disimilarity function* $\varDelta : 2^E \times 2^E \longrightarrow \mathbb{R}$ *such that* $\forall x, y \subseteq E$*,*

$$\varDelta(x, y) = \max(\max_{e \in x} \min_{e' \in y} \delta(e, e'), \max_{e' \in y} \min_{e \in x} \delta(e, e'))$$

Matching-based comparison

The problem with the former distances, but average, is that their value is a function of the distance between one pair of members of the sets. The average linkage, on the other hand, has its value function of the distance between all the possible comparisons.

Matching-based comparisons [Valtchev, 1999] consider that the elements to be compared are those which correspond to each other, i.e., the most similar one.

To that extent, the distance between two sets is considered as a value to be minimised and its computation is an optimisation problem: that of finding the elements of both sets which correspond to each others. In particular, it corresponds to solving a bipartite graph matching problem (§5.7.3).

Definition 4.50 (Match-based similarity). *Given a similarity function* $\sigma : E \times E \longrightarrow \mathbb{R}$*, the match-based similarity between two subsets of* E *is a similarity function* $MSim : 2^E \times 2^E \longrightarrow \mathbb{R}$ *such that* $\forall x, y \subseteq E$*,*

$$MSim(x, y) = \frac{\max_{p \in Pairings(x,y)} \left(\sum_{\langle n,n' \rangle \in p} \sigma(n, n') \right)}{\max(|x|, |y|)},$$

with $Pairings(x, y)$ *being the set of mapping of elements of* x *to elements of* y*.*

This match-based similarity already requires an alignment of entities to be computed. It also depends on the kind of alignment that is required. Indeed, the result will be different depending on whether the alignment is required to be injective or not. The match-based comparison can also be used when comparing sequences [Valtchev, 1999].

Summary on extensional techniques

Knowing extension information is invaluable for ontology matching because this provides information that is independent from the conceptual part of the ontology. Indeed, ontologies are views of the world and this is the reason why there can be numerous different ontologies on the same topic (and the reason why they have to be matched). Extension information is supposed to be less prone to variability and can be used to accurately match classes.

This extension information is even more useful when a set of individuals characterised in both ontologies is available. This provides an easy way to compare the overlap between two classes.

There are situations, however, in which data instance information is not available. This can be caused by the unavailability of data (connection data to a web service is not available) or for confidentiality reasons. In such a situation, the other techniques are the only possible ones.

4.5 Semantic-based techniques

The key characteristics of semantic methods is that model-theoretic semantics is used to justify their results. Hence they are deductive methods. Of course, pure deductive methods do not perform very well alone for an essentially inductive task like ontology matching. They hence need a preprocessing phase which provides ‘anchors’, i.e., entities which are declared, for example, to be equivalent (based on the identity of their names or user input for instance). The semantic methods act as amplifiers of these seeding alignments.

We thus include in semantic techniques particular methods for anchoring the ontologies (§4.5.1). They are based on the use of existing formal resources for initiating an alignment that can be further considered by deductive methods (§4.5.2).

4.5.1 Techniques based on external ontologies

When two ontologies have to be matched, they often lack a common ground on which comparisons can be based. In this section we focus on using intermediate formal ontologies for that purpose. These intermediate ontologies can define the common context or background knowledge [Giunchiglia *et al.*, 2006c] for the two ontologies to be matched. The intuition is that a background ontology with a comprehensive coverage of the domain of interest of the ontologies to be matched helps in the disambiguation of multiple possible meanings of terms.

This common ground can often be found by relating the ontologies to external resources. These resources can differ on three specific dimensions:

Breadth: whether they are general purpose resources or domain specific resources. By using specialised resources, e.g., the Formal Model of Anatomy in medicine, one can be sure that the concepts in the contextualised resources can be matched accurately to their corresponding concepts in the ontology. However, by using more general resources there is more probability that an alignment already exists and can be exploited right away.

Formality: whether they are pure ontologies with semantic descriptions or informal resources such as WordNet. By using formal resources, e.g., DOLCE or the Formal Model of Anatomy, it is possible to reason within or across these formal models in order to deduce the relation between two terms. By using informal resources, e.g., WordNet, it is possible to extend the set of senses that are covered by a term and to increase the number of terms which can express these concepts. There is thus more opportunity to match terms.

Status: whether these resources are considered as references such as ontologies, thesauri or they are sets of instances or annotated documents that are shared.

Since non pure ontological resources such as WordNet have been considered in Sect. 4.2.2 and extensional resources have been dealt with in Sect. 4.4.1, we concentrate here on using external formal ontologies.

Contextualising ontologies can typically be achieved by matching these ontologies with a common upper-level ontology that is used as external source of common knowledge, e.g., Cyc [Lenat and Guha, 1990], Suggested Upper Merged Ontology (SUMO) [Niles and Pease, 2001] or Descriptive Ontology for Linguistic and Cognitive Engineering (DOLCE) [Gangemi *et al.*, 2003].

Example 4.51 (Using upper-level ontologies as background knowledge). An experiment has been carried out by expressing fishery resources (such as databases and thesauri) within the DOLCE upper level ontology [Gangemi, 2004]. The goal was to merge these resources into a common Core Ontology of Fisheries. It has involved transforming manually the resources into lightweight ontologies expressed with respect to DOLCE and then using reasoning facilities for detecting relations and inconsistencies between entities of this ontology.

An approach proposed in [Aleksovski *et al.*, 2006] works in two steps:

Anchoring (also known as contextualising) is matching ontologies o' and o'' to the background ontology o. This can be done by using any available methods presented in this book, usually non sophisticated ones.

Deriving relations is the (indirect) matching of ontologies o' and o'' by using the correspondences discovered during the anchoring step. Since concepts of ontologies o' and o'' become a part of the background ontology o via anchors, checking if these concepts are related, can be therefore performed by using a reasoning service (§4.5.2) in the background ontology. Intuitively, combining the anchor relations with the relations between the concepts of the reference ontology is used to derive the relations between concepts of o' and o''.

Example 4.52 (Using domain specific formal ontologies as background knowledge). Suppose we want to match the anatomy part of the CRISP[10] directory to the anatomy part of the MeSH[11] meta-thesaurus. In this case the FMA ontology[12] can be used as background knowledge which gives the context to the matching task. The result of anchoring is a set of matches with three different kinds of relations: $\equiv$, $\preceq$, $\succeq$ between concepts from FMA, and CRISP or MeSH.

For example, the concept of brain from CRISP, denoted by Brain_{CRISP}, could be easily anchored to the concept brain of FMA, denoted by Brain_{FMA}. Similarly, the concept of head from MeSH, denoted by Head_{MeSH}, could be anchored to a background knowledge concept Head_{FMA}. In the reference ontology FMA there is a *part of* relation between Brain_{FMA} and Head_{FMA}. Therefore, we can derive that Brain_{CRISP} is a part of Head_{MeSH}.

Since the domain specific ontology provides the context for the matching task, the concept of Head was correctly interpreted as meaning the upper part of the human body, instead of, for example, meaning a chief person. This is not so straightforward as can be shown by replacing FMA with WordNet: in WordNet the concept of Head has 33 senses (as a noun). Finally, once the context of the matching task has been established, as our example shows, various heuristics, such as string-based techniques, can improve the anchoring step.

There are some other techniques which attempt at using not one context ontology but as many as possible. These ontologies are typically taken from the web, selected for relevance, i.e., that they contain enough matches with the initial ontologies, and the result is a consensus between the results provided with these ontologies [Sabou *et al.*, 2006a].

Once these initial alignments have been obtained, they can be exploited further by deductive techniques.

4.5.2 Deductive techniques

The basis of the semantic techniques are the merging of two ontologies and the search for correspondences A such that $o, o' \models A$. Of course, this can apply only if A can be considered as a formula of the language. For instance, this can apply if it is a subsumption relation between two entities e and e': $e \sqsubseteq e'$. These semantic techniques can also be used for testing the satisfiability of alignments (§2.5.4), in particular, for discarding alignments which lead to an inconsistent merge of both ontologies.

Examples of semantic techniques are propositional satisfiability, modal satisfiability techniques, or description logic based techniques.

[10] http://crisp.cit.nih.gov/
[11] http://www.nlm.nih.gov/mesh/
[12] http://sig.biostr.washington.edu/projects/fm/

Propositional techniques

An approach for applying propositional satisfiability (SAT) techniques to ontology matching includes the following steps [Giunchiglia and Shvaiko, 2003a, Bouquet and Serafini, 2003, Giunchiglia *et al.*, 2004, Shvaiko, 2006]:

1. Build a theory or domain knowledge ($Axioms$) for the given input two ontologies as a conjunction of the available axioms. The theory is constructed by using matchers discussed in the previous sections, e.g., those based on WordNet, or those using external ontologies (§4.5.1).
2. Build a matching formula for each pair of classes c and c' from two ontologies. The criterion for determining whether a relation holds between two classes is the fact that it is entailed by the premises (theory). Therefore, a matching query is created as a formula of the following form:

   $$Axioms \rightarrow r(c, c')$$

   for each pair of classes c and c' for which we want to test the relation r (within $=$, $\sqsubseteq$, $\sqsupseteq$, $\perp$). c and c' are also sometimes called contexts.
3. Check for validity of the formula, namely that it is true for all the truth assignments of all the propositional variables occurring in it. A propositional formula is valid if and only if its negation is unsatisfiable, which is checked by using a SAT solver.

SAT solvers are correct and complete decision procedures for propositional satisfiability, and therefore, they can be used for an exhaustive check of all the possible correspondences. In some sense, these techniques compute the deductive closure of some initial alignment [Euzenat, 2007].

Example 4.53 (Propositional logic relation inference).

Step 1. Suppose that classes images and Europe belong to one ontology, while another ontology has classes pictures and Europe (as well). A matcher which uses WordNet can determine that images $=$ pictures. Also many other matchers can find that classes of Europe in both ontologies are identical, i.e., Europe $=$ Europe. Then translating the relations between classes under consideration into propositional connectives in the obvious way results in the following $Axioms$:

$$(\mathsf{images} \leftrightarrow \mathsf{pictures}) \wedge (\mathsf{Europe} \leftrightarrow \mathsf{Europe})$$

Step 2. Suppose c is defined as Europe $\sqcap$ images which intuitively stands for the concept of European images, while c' is defined as pictures $\sqcap$ Europe which intuitively stands for the concept of pictures of Europe. Let us also suppose that we want to know if c is equivalent ($\leftrightarrow$) to c'. Thus, this matching task requires constructing the following formula:

$$((\mathsf{images} \leftrightarrow \mathsf{pictures}) \wedge (\mathsf{Europe} \leftrightarrow \mathsf{Europe})) \rightarrow$$
$$((\mathsf{Europe} \wedge \mathsf{images}) \leftrightarrow (\mathsf{Europe} \wedge \mathsf{pictures}))$$

Step 3. Negation of this formula turns out to be unsatisfiable, and therefore, the equivalence relation holds. See also Chap. 9 for a detailed discussion of this example.

Notice that this technique, beside pruning the incorrect correspondences, also discovers the new ones between complex concepts. In the example above c is defined by combining (taking intersection of) such atomic concepts as Europe and images. And, similarly for c'. These are simple examples of complex concepts being bounded by the expressive power of a propositional language. The relation between such complex concepts as (Europe $\wedge$ images) and (Europe $\wedge$ pictures) was not available after the first step, and has being discovered as a result of deduction.

This technique can only be used for matching tree-like structures, such as classifications, taxonomies, without taking properties or roles into account. Modal SAT can be used, as proposed in [Shvaiko, 2004], for extending the methods related to propositional SAT to binary predicates.

Description logic techniques

In description logics, the relations, e.g., $=$, $\sqsubseteq$, $\sqsupseteq$, $\perp$, can be expressed with respect to subsumption. The subsumption test, can be used to establish the relations between classes in a purely semantic manner. In fact, first merging two ontologies (after renaming) and then testing each pair of concepts and roles for subsumption is enough for matching terms with the same interpretation (or with a subset of the interpretations of the others) [Bouquet *et al.*, 2006].

Example 4.54 (Description logic relation inference). Consider two minimal description logic ontologies:

$$\text{Micro-company} = \text{Company} \sqcap \leq_5 \text{employee}$$

meaning that a Micro-company is a Company with at most 5 employees and

$$\text{SME} = \text{Firm} \sqcap \leq_{10} \text{associate}$$

meaning that a SME is a Firm with at most 10 associates. The following initial alignment (expressed in description logic syntax) includes:

$$\text{Company} = \text{Firm}$$
$$\text{associate} \sqsubseteq \text{employee}$$

It expresses that Company is equivalent to Firm and associate is a subclass of employee. This obviously entails:

$$\text{Micro-company} \sqsubseteq \text{SME}$$

i.e., Micro-company is a subclass of SME.

There are other uses of description logic techniques which are relevant to ontology matching. For example, in a spatio-temporal database integration scenario, as first motivated in [Parent and Spaccapietra, 2000] and later developed in [Sotnykova *et al.*, 2005], the inter-schema correspondences are initially proposed by the integrated schema designer and are encoded together with input schemas in the $\mathcal{ALCRP}(\mathcal{S}_2 \oplus \mathcal{T})$ language. Then, description logic reasoning services are used to check the satisfiability of the two source schemas and the set of inter-schema correspondences. If some objects are found unsatisfiable, the inter-schema correspondences should be reconsidered. A similar approach in the context of alignment debugging has also been investigated in [Meilicke *et al.*, 2006].

Summary on semantic techniques

As it was mentioned in the beginning, semantics techniques cannot find the correspondences alone. However, they are invaluable when correspondences are generated in order to ensure the completeness, i.e., find all the correspondences that must hold, and the consistency, i.e., find correspondences that lead to inconsistency, of the alignment.

Only a few of these techniques have been developed so far (usually, databases had only simple semantic theories so these techniques were not developed in this field). However, with the improvement of deductive tools for dealing with semantic web languages, we believe that we will see more systems using semantic-based techniques.

An important challenge of these techniques is their integration with inductive techniques. Indeed, completing alignments and finding inconsistencies is a crucial step. However, once deductive techniques have been applied, their results might be considered as an input to inductive techniques. For example, for finding more correspondences from the completion or for selecting alternative correspondences instead of inconsistent ones. This theme deserves to be further investigated.

4.6 Summary

We have discussed basic techniques that can be used for building correspondences based on terminological (§4.2), conceptual (§4.3), extensional (§4.4) and semantic (§4.5) arguments. This classification of techniques is a natural one since each of these deals with a partial view of ontologies.

There are many such techniques and our goal was not to present them all. It was rather to propose a panorama of the most used ones so far and to show the direction they take. There is still much work going on in finding better methods in each of these directions.

We have also observed that all these techniques cannot be used in isolation, but that each of them can take advantage of the results provided by the others. Another part of the art of ontology matching relies on selecting and combining these methods in the most adequate way. Combinations of basic matchers is the topic of the next chapter.