An Automatic Approach for Mapping Product Taxonomies in E-Commerce Systems

Lennart J. Nederstigt, Steven S. Aanen, Damir Vandić, and Flavius Frăsincar

Erasmus Universiteit Rotterdam
PO Box 1738, NL-3000 DR
Rotterdam, The Netherlands
len_nederstigt@xs4all.nl, ssaanen@gmail.com,
{vandic, frasincar}@ese.eur.nl

Abstract. The recent explosion of Web shops has made the user task of finding the desired products an increasingly difficult one. One way to solve this problem is to offer an integrated access to product information on the Web, for which an important component is the mapping of product taxonomies. In this paper, we introduce CMAP, an algorithm that can be used to map one product taxonomy to another product taxonomy. CMAP employs word sense disambiguation techniques and lexical and structural similarity measures in order to find the best matching categories. The performance on precision, recall, and the F_1-measure is compared favourably with state-of-the-art algorithms for taxonomy mapping.

Keywords: taxonomy mapping, e-commerce, lexical matching, word sense disambiguation

1 Introduction

The rapid expansion and increased reach of the Web over the last twenty years has significantly changed the way in which people exchange information. According to a study by Zhang et al. [22], the Web is growing exponentially, doubling in size roughly every five years. This unprecedented growth also means that the Web plays an increasingly important role in the world economy. The revenue of online shopping in the USA grew from \$7.4 billion in 2000 to \$34.7 billion in 2007 [7], almost a fivefold increase.

With the ever-increasing amount of information available on the Web, it is important that users remain able to access and search all information, as well as have easy and intuitive navigation possibilities. Computers already index Web pages for the use in search engines, but what they often cannot do is understand the actual content found on these Web pages. A human mind is still needed to interpret the data. This is due to the fact that most Web pages are geared towards human-readability rather than machine-usage. In order to fully utilise the power of the Web, it is important that Web information becomes understandable for computers as well.

The Semantic Web would allow computers to grasp the meaning of information on the Web and be able to act independently upon this information. This is done by annotating data, so that it becomes understandable for all systems sharing the same ontology [5]. GoodRelations [6] is a good example of an ontology that can be used to annotate Web resources with e-commerce or product information. For instance, with GoodRelations it is possible to formally specify the warranty, the delivery options, the payment methods, the currency, etc. It is also possible to specify product specific properties, e.g., the screen size of a mobile phone or the CPU speed of laptop.

Unfortunately, very few websites have included a semantic description of their published information, although there are definitely advantages to be gained by doing so. For instance, search engines could provide much more relevant search results if they would actually understand what is published on every website they index. Furthermore, there is evidence that annotation is urgently needed in order to make the Web more useful. According to the aforementioned study on online shopping in the USA [7], more than half of the surveyed users have encountered various frustrations and frictions while shopping online. Information was often found to be lacking, confusing, or there simply was an overload of information.

Integrating the information found on multiple websites would help people in finding the information that is relevant to them. In the field of e-commerce this means that the information about products, as offered by different online retailers, should be integrated. An important part of the integration is the mapping of the product taxonomy of one online retailer to that of another retailer. Aggregating these taxonomy structures enables the presentation of product information in a unified way to the user, largely alleviating frustrations caused by scattered information and failing search results.

However, because of the heterogeneity of product taxonomies, used by different online retailers, the aggregation of product information is not a straightforward process. The heterogeneity is caused by the fact that the construction of product taxonomies is a loosely defined process and as such, there is no uniform way for creating one. This results in characteristics of a product taxonomy that are difficult to handle, like categories composed of multiple terms, or a loosely defined hierarchy for type-of relations. Furthermore, product taxonomies tend to have a large level of depth, which makes category mapping a complex and difficult process.

In this paper we propose the *Category Mapping Algorithm for Products* (CMAP). This algorithm can be used to map product taxonomies from multiple sources to each other. We employ *word sense disambiguation* techniques, using WordNet [15], to find synonyms, hypernyms, and possible senses of category names. Furthermore, we use similarity measures, such as *Jaccard* [8], to determine the most likely candidate category to map to. In order to evaluate the performance of our algorithm, we compare its precision, recall, and the F_1-measure with state-of-the-art algorithms for taxonomy mapping, i.e., the algorithm proposed by Park & Kim [18] and the PROMPT algorithm [17].

2 Related Work

Recently, the merging of taxonomies from heterogeneous sources has been extensively studied, which has resulted in various approaches that have been proposed and implemented. In general, we can distinguish between schema-based and ontology-based matching. Schema matching is usually performed with the help of techniques that are trying to guess the meaning encoded in the schemas, whereas ontology matching systems primarily try to exploit knowledge explicitly encoded in the ontologies [20]. In other words, ontology matching works with structured data that the computer can understand, while schema matching has to deal with unstructured data that needs to be interpreted.

Similar concepts from different taxonomies can be identified either by matching their corresponding terms or by examining their position in the structure of the taxonomy. Many algorithms employ term matching, which indicates how closely one term is related to another, either lexically or semantically. Various measures have been used for this purpose [8,9,11,19,21]. Some of the semantic similarity measures use a semantic lexicon, such as WordNet [15], or a shared upper ontology, such as DOLCE [4] or SUMO [16], to extract the meaning of a term. The path similarity, which indicates how closely two terms are related by analysing their context in the taxonomy structures, can be measured by using, for example, *neighbour matching* [2,3], *tree matching* [1,3,12], *path matching* [1,17], and *iterative fix-point computation* [14].

Besides algorithms that focus on automatic mapping, we find also semi-automatic approaches for taxonomy mapping in the literature. Chimaera [13] is an example of such an approach. Chimaera is a mapping tool that can be used either stand-alone or as part of the Ontolingua environment. It suggests, considering the structural similarity, potential mappings based on the lexical similarity between classes. LOM, which is an ontology mapping tool, uses different techniques in order to find the best match [10]. It analyses the lexical similarity between classes by using both exact and partial term matching. Furthermore, it refines the search for potential matches by using sets of synonyms and similar concepts. The synonym sets are found using WordNet and the similar concepts using the SUMO upper ontology.

Noy and Musen have created the PROMPT suite, which includes various tools that can be used for multiple-ontology management [17]. One of these tools is iPROMPT, which provides interactive ontology merging by letting the user decide on how to map a term. It assists the user by providing suggestions based on already defined mappings and the lexical similarity between terms. While the achieved results on recall and precision are quite good, it involves a lot of manual labour and is thus not suited for larger ontologies. AnchorPROMPT, an extension of iPROMPT and also part of the PROMPT suite, addresses this issue by exploiting the category position in the structure of a taxonomy for automatically defining the mappings. It automatically generates a larger set of identical concepts from a previously identified set. These pairs of related terms from the source ontologies, called *anchors*, can be created manually or generated by the system. This approach requires far less manual input than iPROMPT, but it is not specifically targeted at matching product taxonomies and also the precision drops when longer category paths are used.

Park & Kim suggest to map product taxonomies using a *word sense disambiguation* technique [18]. Given a category, they first try to find the correct sense and the synonyms associated with that particular sense. Using these synonyms, all category paths that have one of the synonyms as a leaf category are retrieved, i.e., the candidate paths. Then the algorithm uses the similarity between the taxonomy paths in order to find the best match. The algorithm can perform better than ontology mapping algorithms that are not specifically designed for matching heterogeneous product taxonomies. However, it does not perform well on nodes close to the root node of a category hierarchy, since it needs more information content, which is only present along longer paths. Furthermore, the algorithm has a bias towards mapping to general categories, which does not work well when dealing with categories composed of multiple product concepts, e.g., mapping 'Books' to 'Books, Movies, Music, Games', while this last category also has a subcategory called 'Books'.

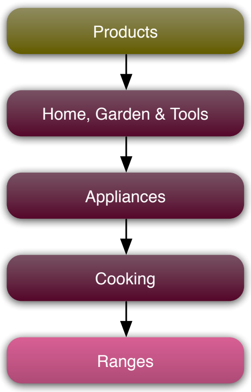
3 CMAP

This section explains the CMAP algorithm. Although the CMAP algorithm is based on the algorithm proposed by Park & Kim, there are important differences. CMAP is designed to address the specific issues that arise from the structure of product taxonomies, rather than focusing on schema matching in general. Furthermore, we present a simple, but powerful procedure, which employs lexical term matching and path similarity in conjunction with word sense disambiguation techniques, to determine the best mapping.

Both CMAP and the algorithm from Park & Kim start with searching for the correct meaning of category names from the source taxonomy. Using the synonyms of the correct sense gathered from that process, candidate categories that match the current source category are selected in the target taxonomy. The best out of these candidates is selected using two measures called *co-occurrence* and *order-consistency*. We explain these steps in detail in the rest of this section. The described process is being repeated for every category path in the source taxonomy.

Our approach also handles root category nodes differently than Park & Kim. CMAP assumes that the root category term has no meaning and should therefore not be used in the matching process. For example, the root nodes from the datasets used in this research are 'Online Shopping', 'Shopping' and 'Products'. For this reason, CMAP always maps the source root category to the target root category. Because the root category does not provide any information, it is skipped in computations that use category paths. Another important observation is, that all term matching within CMAP is case-insensitive, since the case of characters generally does not affect the meaning.

Many category names in existing datasets are actually composite categories, like 'Home, Garden & Tools', as shown in Fig. 1. This issue has not been investigated by Park & Kim. In CMAP, we address this issue by splitting the category string on predefined words (e.g., ‘and’) or characters (e.g., ‘,’). The result is called the *split term set*. For each element from this set, the word sense disambiguation process discussed next is applied separately.



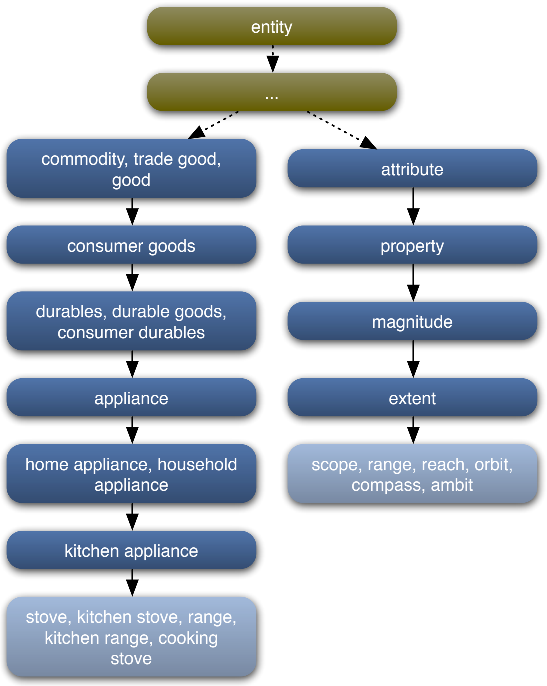

entity
...
commodity, trade good, good
consumer goods
durables, durable goods, consumer durables
appliance
home appliance, household appliance
kitchen appliance
stove, kitchen stove, range, kitchen range, cooking stove
attribute
property
magnitude
extent
scope, range, reach, orbit, compass, ambit

Fig. 1. Hierarchy for ‘Ranges’ in the Amazon taxonomy

Fig. 2. Two sense hierarchies for the term ‘ranges’ in WordNet

3.1 Word Sense Disambiguation

A core concept of both algorithms is finding the correct sense of a category term from the source taxonomy. WordNet [15], a semantic lexicon, is used for obtaining all possible senses of a category from the source taxonomy. In order to find the correct sense, we have to match the full path of a category to the hierarchy of hypernyms for each sense. For instance, if the path from the source taxonomy is ‘Computers/Notebook’, we can deduce that the correct sense of ‘Notebook’ would be a laptop in this case, rather than a notepad. By finding the correct sense of a term we can also use its synonyms, which enhances the ability to identify category matches, even when the category names are not lexically similar. This is particularly important for matching product taxonomies, because there is no standard for category names, so each taxonomy might be using different words to express the very same product category. For example, one taxonomy might have a category called 'Notebook', while another might have 'Laptop'. In both cases the meaning is the same, but they are impossible to match using just lexical similarity measures.

Using WordNet, different meanings for the category (split) term are acquired, in the form of their hypernym structure. We call this the sense hierarchy. Figure 2 shows two senses for the term 'ranges'. The synonyms per sense for 'ranges' are at the bottom in light blue (light grey in black and white printing). The goal of the word sense disambiguation process is to end up with only one set of these synonyms, namely the one for the correct source category sense. Note that the word sense disambiguation process only uses the current source category, and the WordNet information about the possible meanings. The target taxonomy does not play a role in the algorithm yet.

In order to decide which meaning fits best to the source category that has to be mapped, a function is employed that finds matches between an upper category (an ancestor of the current node) and a sense hierarchy excluding the leaf node (denoted by list S). Upper categories use dark purple (dark grey in black and white printing) nodes in Fig. 1. The sense hierarchy represents one of the meanings of the current category, like one out of the two paths in Fig. 2. The function is given by:

$$\text{computeSimilarity}(t, S) = \{x | x \in H, H \in S \text{ and } \text{baseform}(t) \in H\} \tag{1}$$

$$\begin{aligned} \text{where } t &= \text{an upper category to match} \\ H &= \text{the set of synonyms of one hypernym} \\ S &= \text{a sense hierarchy (one sense from WordNet)} \end{aligned}$$

where `baseform()` is a function which returns the lemma of a term using WordNet. The result of the function is a set of matches between an upper category, and a sense hierarchy (one meaning of the current category).

Using Equation (1), a measure can be defined that represents how well an ancestor of the source category fits to a certain sense hierarchy. In other words, how well does an upper category fit to one of the possible meanings of the source category? The match between an upper category and an element from the sense hierarchy that is closest to the sense leaf, is seen as most important match. Therefore, `hyperProximity()` uses the distance between the closest match and the leaf to compute a measure for the fit given by:

$$\text{hyperProximity}(t, S) = \begin{cases} \frac{1}{\min\limits_{x \in C}(\text{dist}(x, b))} & \text{if } C \neq \emptyset \\ 0 & \text{if } C = \emptyset \end{cases} \tag{2}$$

$$\begin{aligned} \text{where } t &= \text{an upper category to match} \\ S &= \text{a sense hierarchy (one sense from WordNet)} \\ C &= \text{computeSimilarity}(t, S) \\ b &= \text{base term (leaf) of the sense hierarchy } S \end{aligned}$$

Function `min()` gives the minimum of a set of values, and `dist()` computes the distances between two nodes in the sense hierarchy. The distance can be explained as the number of arcs that are being traversed when navigating from node to node in a sense hierarchy. In the Ranges example, the hyperproximity for parent 'Appliances' and the left sense hierarchy in Fig. 2, would be $\frac{1}{3}$. This is because the node in the sense hierarchy closest to the base, is 'appliance'.

Until now the measurements only used the relation between an upper category and a sense for the source category. However, we need to be able to measure how well a complete source category path, like the one depicted in Fig. 1 (denoted by P), fits one particular sense (hierarchy) S of the category (split) term. The measure for this is given by:

$$\text{pathProximity}(P, S) = \sum_{x \in P} \frac{\text{hyperProximity}(x, S)}{|P| - 1} \tag{3}$$

$$\begin{aligned} \text{where } P &= \text{list of nodes from the source category path} \\ S &= \text{a sense hierarchy (one sense from WordNet)} \\ x &\neq b \\ b &= \text{base term (leaf) of the sense hierarchy } S \end{aligned}$$

It is the average hyperproximity for all upper categories with the current sense hierarchy. This differs from the Park & Kim algorithm, which divides by the total number of nodes (not only ancestors), which does not lead to an average.

Since 'Appliances' is the only upper category matching with something from the sense hierarchies, the path-proximity for the left sense hierarchy in Fig. 2 is $\frac{1}{9}$. This is because the denominator is 3 ($|P| - 1$), i.e., CMAP does not use the root node in the calculation. The path-proximity for the sense hierarchy on the right in the figure is 0 since there are no matches.

Selecting the Proper Sense. Equation (3) measures the fit of the source category path to one possible sense of the category term. This must be extended to be able to deal with multiple possible senses. This is done by computing the path-proximity for all possible senses of the source category (split) term. The sense with the highest path-proximity is picked as the correct sense. In our example, the first (left in the figure) sense would be chosen, which is the correct conclusion.

Last, only the original source category term (or terms in case of a composite category) and the synonyms of the correct sense are used (or synonyms of the correct senses), i.e., the extended split term set. For our example depicted in Fig. 2, that would be the left node in light blue (light grey in black and white printing). According to Park & Kim's algorithm, when none of the senses fit, or when the category (split) term is not known in WordNet, the process should be stopped and the source category should be mapped to nothing. However, while testing the algorithm of Park & Kim, we discovered that this would result in low performance on recall. That is why, for evaluation purposes, we have implemented their algorithm in such a way, that the algorithm takes the source category name itself as the "extended" term set, and continue the process. CMAP does this as well, but then for the (possibly) split version of the (composite) category, i.e., the "extended" split term set.

3.2 Candidate Path Selection

Using the extended (split) term set that from the word sense disambiguation process, candidate target paths can be collected. This step only uses the target taxonomy. CMAP simply walks through every single category from the target taxonomy. If the split target category term is a superset of the extended split term set, the path of that category is being marked as a candidate. For example, source category 'Home and Garden' would fit into candidate category 'Home, Garden & Tools', but not the other way around.

Park & Kim, as they do not handle composite categories, check only if the source category term (or its synonyms) is contained in the target category. In addition, they claim that from all candidates, it is needed to remove all more specific ones. In other words, each candidate path of which an ancestor also exists as separate candidate, is removed. However, as explained in Sect. 2, this results in errors with composite categories. CMAP therefore gives every target category, despite its ancestors, the same chance of winning the match with the source category.

Now, there is a collection of candidate paths for the category mapping. To determine which candidate fits best, the co-occurrence and order-consistency will be measured for each of the target paths. The extended (split) term set is only being used for the candidate path selection. In the rest of the algorithm it will not be used again.

3.3 Co-occurrence

The co-occurrence is a measure that defines how well the current source category path fits to a candidate target path, while ignoring the order of the nodes in each path. It expresses the level of overlap between two category paths, i.e., the source taxonomy path and one of the candidate target paths.

The co-occurrence measure is based on the function `maxSim()`, which computes the similarity between one category name and another category path (either source or target). This similarity is defined as the highest value of the Jaccard index (for sets of characters), measured on each pair of the category name and a category name from the given category path. For the computation of the Jaccard index, we use a morphological processor in conjunction with WordNet to find the lemma of each category name. This improves the accuracy of the Jaccard index as it now ignores, for example, singular and plural forms of words.

Using `maxSim()`, we define the co-occurrence as following:

$$\text{coOccurrence}(P_{\text{src}}, P_{\text{targ}}) = \left(\sum_{t \in P_{\text{targ}}} \frac{\text{maxSim}(t, P_{\text{src}})}{|P_{\text{targ}}|} \right) \cdot \left(\sum_{t \in P_{\text{src}}} \frac{\text{maxSim}(t, P_{\text{targ}})}{|P_{\text{src}}|} \right)$$

$$\begin{aligned} \text{where } P_{\text{src}} &= \text{list of nodes from the current source path} \\ P_{\text{targ}} &= \text{list of nodes from a candidate target path} \\ t &= \text{a category term} \end{aligned}$$

As we can notice from the above equation, the co-occurrence consists of a product of two terms that look similar at first sight. The difference between the two terms is the way of comparing, in the first term we compare each node from the target path to the source path and in the second term we compare each node from the source path to the target path. For each pair of a node and a path (either source or target), we compute the similarity using the `maxSim()` function. Both the left and right term in the co-occurrence equation are the average of this similarity for each node that is processed.

This process is best explained with an example. Consider a source path 'Shopping/Books/Science' from the ODP taxonomy and a candidate target path 'Products/Books/Magazines/Science' from the Amazon taxonomy. The first category in both paths is not used as we do not process the root nodes of taxonomies. This means that we start with 'Books', 'Magazines', and 'Science' from the Amazon path, i.e., the first term in the co-occurrence equation in this example. Because we have a full match for 'Books' and 'Science', we have `maxSim`('Books', {'Books','Science'}) = 1 and `maxSim`('Science', {'Books', 'Science'}) = 1. For 'Magazines', we have `maxSim`('Magazines', {'Books', 'Science'}) = 0.25 as the best match, because there are 3 characters in the intersection and 12 characters in the union of the two character sets 'Magazines' and 'Science'. Thus, the first term of the co-occurrence measure from Equation **??** is in this example $(1 + 1 + 0.25)/3 = 0.75$.

The same process is then repeated except that we start with 'Books' and 'Science' from the ODP path, i.e., the second term in the co-occurrence equation in this example. From this it follows that the second part is $(1 + 1)/2 = 1$, as there is a perfect match for both the categories 'Books' and 'Science'. Finally, we multiply both terms in order to obtain the co-occurrence measure, which is $0.75 \times 1 = 0.75$.

The Park & Kim algorithm has a similar definition for the co-occurrence measure, but instead of using the Jaccard index for the similarity, they use a function called `termMatch()`. If one of either words is contained within the other, `termMatch()` returns the value of the length of the smallest string divided by the length of the largest one. We find that this approach is too restricted too work well with a broad range of categories in product taxonomies.

3.4 Order-Consistency

The co-occurrence measure compares the similarity between nodes in source and candidate target paths. However, in order to find the best possible mapping it is also important that these nodes appear in the same order in both paths. The procedure that computes the order-consistency starts with the function `common()`, which searches for matching nodes between the source path and the current candidate target path. For every match, where also synonyms of the original term can be used, it adds the matching nodes to the common list.

Function `precedenceRelations()` uses the common list to find binary node associations of which the first node hierarchically comes before the second node in the source path. For every element in the common list, it creates pairs of node names from the source path. The first node in the pair should always occur before the other node in the source path. The order of the pairs themselves in the precedence relation set is not important.

Checking whether one precedence relation is applicable to the candidate target path as well is done by the function `consistent()`. If so, it returns the value 1, otherwise it returns 0. The order-consistency is given by:

$$\text{orderConsistency}(P_{\text{src}}, P_{\text{targ}}) = \sum_{r \in R} \frac{\text{consistent}(r, P_{\text{targ}})}{\binom{\text{length}(C)}{2}} \tag{4}$$

$$\begin{aligned} \text{where } P_{\text{src}} &= \text{list of nodes from the current source path} \\ P_{\text{targ}} &= \text{list of nodes from a candidate target path} \\ C &= \text{common}(P_{\text{src}}, P_{\text{targ}}) \\ R &= \text{precedenceRelations}(C, P_{\text{src}}) \\ r &= \text{one precedence relation} \end{aligned}$$

The denominator is the number of possible combinations to choose two out of the common nodes. Therefore, the order-consistency is the average number of precedence relations from the source path that are consistent with the candidate target path.

3.5 Overall Similarity

For every candidate target path, the co-occurrence and order-consistency are computed. The Park & Kim algorithm takes the candidate path that has the highest average of these measures as the measure for overall similarity. CMAP uses the following function to determine the overall similarity:

$$\begin{aligned} \text{overallSimilarity}(P_{\text{src}}, P_{\text{targ}}) &= (\text{orderConsistency}(P_{\text{src}}, P_{\text{targ}}) + t) \\ &\quad \cdot \text{coOccurrence}(P_{\text{src}}, P_{\text{targ}}) \end{aligned} \tag{5}$$

$$\begin{aligned} \text{where } P_{\text{src}} &= \text{list of nodes from the current source path} \\ P_{\text{targ}} &= \text{list of nodes from a candidate target path} \\ t &= \text{the similarity threshold} \end{aligned}$$

The similarity threshold is a parameter for both algorithms that functions as a cut-off for the similarity measure for which the source path will not be mapped. CMAP uses this in the overall similarity measure to give candidate paths that have a co-occurrence of 1 and an order-consistency of 0 a chance to pass, since in that situation the overall similarity will not be below the cut-off value. The reason for this is that in the source taxonomy, children of the root node (e.g., 'Shopping/Books') will always have an order-consistency of 0 because there are no precedence relation pairs to be found, i.e., the root is not used in the process. However, it could well be that a candidate path like 'Products/Books' would fit, even though the order-consistency is 0. The order-consistency offset with the threshold is multiplied by the co-occurrence to make the two measures more dependent on each other. When one of them is very low, while the other is very high, the resulting measure would be smaller than when the average would be used. It ensures that candidate paths with both a fair order-consistency and co-occurrence will better score overall than candidate paths which only score well for one of the measures. This will prevent candidate paths that only have a small set of common categories (in the correct order) from being picked as the correct path based on their high score for order-consistency. In both algorithms, the candidate target path with the highest overall similarity, if passing the similarity threshold, will be chosen for the mapping.

When no good candidate target path has been found, CMAP, unlike the algorithm of Park & Kim, does try to provide a mapping. When the parent of the current source category could be mapped earlier, the source category is being mapped to the same target path. For instance, if 'Shopping/Books/Braille' can not be mapped, it will be mapped to 'Products/Books', since that is a more general category in which it also fits (i.e., the mapping of the parent 'Books'). However, if the parent had the same problem with mapping, and is also mapped to its parent, the current source category will not be mapped. This is done in order to prevent too extreme generalisations from happening.

The algorithm basically follows the complete procedure from word sense disambiguation, to candidate selection, to finding the best of the candidate target paths using the two measures, for each source category path (from root category to leaf category) from the taxonomy, with a chosen similarity threshold.

4 Evaluation

We compare CMAP with the algorithm from Park & Kim and PROMPT, which are the most similar and relevant projects for our goals. We first discuss the design of the evaluation and what measures were considered during the experiments.

4.1 Evaluation Design

For the purpose of evaluating our algorithm, three real-life product taxonomies were collected. The first product taxonomy contains over 2,500 categories and is from Amazon.com, one of the largest online retailers in the world. The second dataset contains over 1,000 categories and is from Overstock.com (O.co), which is a big retailer that has RDFa-tagged product pages for the GoodRelations [6] ontology. The third dataset is the largest one, containing 44,000 product categories, and is from the Open Directory Project (ODP). ODP is the largest human-edited directory of the Web. All data was collected using custom-built HTML DOM or RDFa crawlers. Using these three datasets we can run six different mappings, one for each combination of datasets. To be able to evaluate a mapping, it is necessary to do the mappings by hand as well. Since the datasets are too large for this, we have taken a random sample of five hundred nodes from each dataset. For every node, we ensure that its ancestors are also in the sample set. A mapping is always performed from a sampled dataset to a full dataset. In order to prevent any bias, the manual mappings were performed collectively by three independent individuals. The main experiment consisted of computing the mappings for all combinations of datasets (using different parameters) for each algorithm. For CMAP and Park & Kim a different overall similarity threshold was used, ranging from 0.0 to 1.0 in steps of 0.05. The optimal thresholds were obtained using a hill-climbing procedure with the F_1-measure as the optimization criterion. PROMPT has one important parameter, i.e., the maximum path length that is considered. Using the same procedure as for the algorithm of Park & Kim, we found that the optimal value for this parameter is 2. Finally, the evaluation results were obtained by comparing the generated mappings with the manual mappings.

We decided to use the well-established measures from information retrieval for our evaluation, i.e., precision, recall, accuracy, and specificity. For our evaluation, the common definitions cannot be used since we do not have one 'positive' class, but as many 'positive' classes as the number of correctly mapped paths in the target taxonomy. We do have only one 'negative' class, i.e., the action of mapping to nothing. The true and false positives (TP and FP) are therefore defined as the correct and incorrect mappings to a path, respectively, and the true and false negatives (TN and FN) as the correct and incorrect mappings to null (nothing), respectively. The precision, recall, accuracy, and specificity are then defined as:

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \qquad \text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$\text{accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}} \qquad \text{specificity} = \frac{\text{TN}}{\text{FP} + \text{TN}}$$

In order to have an overall score for each algorithm, we use the F_1-measure, which is the harmonic mean of the precision and recall.

4.2 Results

As shown in Table 1, CMAP performs better on average for both precision and recall when compared with the algorithm of Park & Kim and PROMPT. Furthermore, this increase in performance does not come at the expense of a significant amount of additional computation time. The increase from 4.99 seconds to 5.82 seconds for the average computation time per mapping, on an Intel Core 2 Duo P8800 at 2.66GHz with 4GB of RAM, is mostly caused by a large number of composite categories in the Amazon and Overstock datasets. These category names are split and processed individually by CMAP rather than processing the whole category name at once, which explains the slight increase in average computation time. Although we have not performed a formal computational complexity analysis due to the intricacies of the proposed algorithm, our results indicate the CMAP has linear time complexity w.r.t. the number of category nodes in the target path. CMAP maps 500 categories to ODP within 8 seconds, while the same 500 categories are mapped in approximately 0.2 seconds to Overstock.com. PROMPT is the fastest amongst all algorithms, at the expense of a lower precision and recall.

Table 1. Comparison of average results per algorithm

Algorithm	Precision	Recall	F_1-measure	Computation time
PROMPT	19.82%	10.62%	0.1350	0.47 s
Park & Kim	37.89%	17.93%	0.2415	4.99 s
CMAP	41.82%	26.03%	0.3180	5.82 s

The recall is important for the matching of product taxonomies, because it is better to map many categories with some possible imprecision rather than only mapping a few categories with high precision in this field. The main objective of mapping product taxonomies is to reduce the number of search failures, when the user cannot find the products he is interested in, rather than making sure that no errors are being made. Although the recall performance is relatively low, we do improve over the method of Park & Kim and PROMPT.

The average precision has increased from 37.89% for the algorithm of Park & Kim to 41.82% for CMAP. Our increased precision is favoured by the new overall path similarity measure, which ensures that both order consistency and co-occurrence need to have high values for the proposed mappings. Also, the higher precision and recall can be attributed at least partially to the fact that CMAP will map a category to the mapping of its parent when no match can be found. This is intuitive for product taxonomies, because a very specific product category might not exist in the other taxonomy, but there might be a more general category to which the algorithm can map. Furthermore, splitting composite category names enables us to find more candidate paths to map to, which also greatly improves the recall. For the average recall, our algorithm outperforms the algorithm of Park & Kim with a difference of 8.10 percent point, i.e., our average recall is 45% higher than that of Park & Kim. Besides the ability to deal with composite categories, the high recall can be attributed also to the use of the Jaccard index for matching terms. The algorithm of Park & Kim uses an

Table 2. Best results for Park & Kim algorithm

Mapping	Precision	Accuracy	Specificity	Recall	F_1-measure	Threshold
Amazon → ODP	29.10%	20.80%	40.63%	11.47%	0.1645	0.10
Amazon → O.co	47.15%	31.80%	60.84%	17.37%	0.2539	0.00
ODP → Amazon	27.07%	42.48%	64.47%	15.93%	0.2006	0.05
ODP → O.co	31.89%	27.66%	38.54%	20.07%	0.2464	0.00
O.co → Amazon	54.29%	32.20%	45.21%	26.84%	0.3592	0.00
O.co → ODP	37.86%	26.60%	47.90%	15.92%	0.2241	0.00
Average	37.89%	30.26%	49.60%	17.93%	0.2415	

Table 3. Best results for CMAP

Mapping	Precision	Accuracy	Specificity	Recall	F_1-measure	Threshold
Amazon → ODP	31.44%	25.60%	26.29%	25.09%	0.2791	0.05
Amazon → O.co	69.94%	44.60%	66.23%	34.97%	0.4663	0.30
ODP → Amazon	27.06%	41.68%	56.64%	21.60%	0.2402	0.15
ODP → O.co	39.61%	31.26%	50.53%	19.61%	0.2624	0.10
O.co → Amazon	46.53%	32.00%	37.93%	28.83%	0.3561	0.20
O.co → ODP	36.32%	29.40%	34.15%	26.10%	0.3037	0.15
Average	41.82%	34.09%	45.30%	26.03%	0.3180	

Table 4. Best results for PROMPT

Mapping	Precision	Accuracy	Specificity	Recall	F_1-measure
Amazon → ODP	7.74%	14.40%	29.56%	4.04%	0.0531
Amazon → O.co	35.59%	29.00%	57.54%	13.08%	0.1913
ODP → Amazon	8.08%	31.26%	43.48%	9.04%	0.0853
ODP → O.co	15.51%	20.84%	32.19%	10.90%	0.1280
O.co → Amazon	41.95%	27.80%	39.52%	21.92%	0.2879
O.co → ODP	10.07%	18.80%	39.02%	4.75%	0.0645
Average	19.82%	23.68%	40.22%	10.62%	0.1350

exact lexical match procedure for this purpose, which does not take the syntactic variations between terms into account.

Tables 2, 3, and 4 show the detailed results for each combination of data sets, where in the Tables 2 and 3 the column 'Threshold' contains the different overall similarity thresholds. Compared to the algorithm of Park & Kim, the only mapping for which CMAP scores lower on the F_1-measure is the mapping from Overstock to Amazon. We can see in Tables 2 and 3 that the F_1-measure drops from 0.3592 to 0.3561. Table 4 shows that the results of PROMPT for the F_1-measure are inferior to both CMAP and the algorithm of Park & Kim.

5 Conclusions and Future Work

This paper proposes CMAP, an algorithm suitable for mapping product taxonomies. The main focus in the development of CMAP was to improve an existing mapping algorithm by accounting for issues that are specific for product taxonomies. It achieves this objective by enhancing the algorithm proposed by Park & Kim with composite category splitting, using Jaccard index as similarity measure instead of exact lexical matching, using a different measure for overall similarity, and by mapping to a more general target category when no match can be found for a source category. The algorithm was tested on three different datasets and its performance was compared with that of PROMPT and the algorithm of Park & Kim. When categories are mapped from one product taxonomy to another, the results show that a higher average precision, recall, and F_1-measure are obtained using CMAP than by using the other state-of-the-art algorithms.

The algorithm could be improved by not only analysing the ancestors of a category and the hypernyms of its sense, but also its children and hyponyms. This resolves the issue with short paths, where the word sense disambiguation process is often inconclusive because it simply cannot obtain enough information about the correct sense from the path. The word sense disambiguation process could also be further improved by splitting composite categories of the upper categories in a path as well, rather than just splitting the current category. Furthermore, the results from the word sense disambiguation process are now only used for selecting candidate paths, but they could also be used for the co-occurrence and order-consistency measures. Last, in future work we plan to distinguish between nouns and adjectives in category names. This would allow for more precise mapping, because nouns are generally more important for product similarity than adjectives.

Acknowledgment This work is sponsored by the Netherlands Organisation for Scientific Research (NWO) Mozaiek project 017.007.142: Semantic Web Enhanced Product Search (SWEPS).

References

1. Aumueller, D., Do, H., Massmann, S., Rahm, E.: Schema and Ontology Matching with COMA++. In: ACM SIGMOD International Conference on Management of Data 2005 (SIGMOD 2005). pp. 906–908. ACM (2005)
2. Castano, S., Ferrara, A., Montanelli, S., Zucchelli, D.: Helios: A General Framework for Ontology-based Knowledge Sharing and Evolution in P2P Systems. 14th International Workshop on Database and Expert Systems Applications (2003)
3. Ehrig, M., Staab, S.: QOM - Quick Ontology Mapping. In: International Semantic Web Conference 2004 (ISWC 2004). pp. 683–697. Springer (2004)
4. Gangemi, A., Guarino, N., Masolo, C., Oltramari, A., Schneider, L.: Sweetening ontologies with dolce. In: Knowledge Engineering and Knowledge Management: Ontologies and the Semantic Web, Lecture Notes in Computer Science, vol. 2473, pp. 223–233. Springer Berlin / Heidelberg (2002)

5. Gruber, T.: A Translation Approach to Portable Ontology Specifications. Knowledge Acquisition 5(2), 199–199 (1993)
6. Hepp, M.: Goodrelations: An ontology for describing products and services offers on the web. In: Knowledge Engineering: Practice and Patterns (EKAW 2008), Lecture Notes in Computer Science, vol. 5268, pp. 329–346. Springer (2008)
7. Horrigan, J.: Online Shopping. Pew Internet & American Life Project Report 36 (2008)
8. Jaccard, P.: The Distribution of the Flora in the Alpine Zone. New Phytologist 11(2), 37–50 (1912)
9. Levenshtein, V.: Binary Codes Capable of Correcting Deletions, Insertions, and Reversals. In: Soviet Physics Doklady. vol. 10, pp. 707–710 (1966)
10. Li, J.: LOM: A Lexicon-based Ontology Mapping Tool. In: Performance Metrics for Intelligent Systems 2004 (PerMIS 2004) (2004)
11. Lin, D.: An Information-Theoretic Definition of Similarity. In: 15th International Conference on Machine Learning (ICML 1998). pp. 296–304. Morgan Kaufmann Publishers Inc. (1998)
12. Madhavan, J., Bernstein, P., Rahm, E.: Generic Schema Matching with Cupid. In: 27th International Conference on Very Large Data Bases (VLDB 2001). pp. 49–58. Morgan Kaufmann Publishers Inc. (2001)
13. McGuinness, D., Fikes, R., Rice, J., Wilder, S.: The Chimaera Ontology Environment. In: 17th National Conference on Artificial Intelligence (AAAI 2000). pp. 1123–1124. AAAI Press (2000)
14. Melnik, S., Garcia-Molina, H., Rahm, E.: Similarity Flooding: A Versatile Graph Matching Algorithm and its Application to Schema Matching. In: 18th International Conference on Data Engineering (ICDE 2002). pp. 117–128. IEEE (2002)
15. Miller, G.: WordNet: A Lexical Database for English. Communications of the ACM 38(11), 39–41 (1995)
16. Niles, I., Pease, A.: Towards a Standard Upper Ontology. In: International Conference on Formal Ontology in Information Systems 2001 (FOIS 2001). pp. 2–9. ACM (2001)
17. Noy, N., Musen, M.: The PROMPT Suite: Interactive Tools for Ontology Merging and Mapping. International Journal of Human-Computer Studies 59(6), 983–1024 (2003)
18. Park, S., Kim, W.: Ontology Mapping between Heterogeneous Product Taxonomies in an Electronic Commerce Environment. International Journal of Electronic Commerce 12(2), 69–87 (2007)
19. Resnik, P.: Using Information Content to Evaluate Semantic Similarity in a Taxonomy. In: 14th International Joint Conferences on Artificial Intelligence (IJCAI 1995). pp. 448–453 (1995)
20. Shvaiko, P., Euzenat, J.: A Survey of Schema-Based Matching Approaches. Journal on Data Semantics IV 3730, 146–171 (2005)
21. Yang, Y., Liu, X.: A Re-examination of Text Categorization Methods. In: 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 1999). pp. 42–49. ACM (1999)
22. Zhang, G., Zhang, G., Yang, Q., Cheng, S., Zhou, T.: Evolution of the Internet and its Cores. New Journal of Physics 10, 123027 (2008)