An Automated Approach to Product Taxonomy Mapping in E-commerce

Lennart Nederstigt, Damir Vandic, Flavius Frasincar

Abstract Due to the ever-growing amount of information available on Web shops, it has become increasingly difficult to get an overview of Web-based product information. There are clear indications that better search capabilities, such as the exploitation of annotated data, are needed to keep online shopping transparent for the user. For example, annotations can help present information from multiple sources in a uniform manner. This paper proposes an algorithm that can autonomously map heterogeneous product taxonomies for Web shop data integration purposes. The proposed approach uses word sense disambiguation techniques, approximate lexical matching, and a mechanism that deals with composite categories. Our algorithm's performance on three real-life datasets was compared favourably against two other state-of-the-art taxonomy mapping algorithms. The experiments show that our algorithm performs at least twice as good compared to the other algorithms w.r.t. precision and F-measure.

Key words: e-commerce, taxonomy mapping, word sense disambiguation

1 Introduction

The interchange of information has become much easier with the advent of the Web. This ease of sharing information has lead to a worldwide surge in activity on the Web. According to a recent study [13], the Web is doubling in size roughly every five years. The ever-growing amount of information stored on the Web poses several problems. Due to the vast amount of Web sites, it is becoming increasingly difficult to get a proper overview of all the relevant Web information. While traditional search engines help to index the Web, they do not understand the actual

Lennart Nederstigt (e-mail: len_nederstigt@xs4all.nl) · Damir Vandic (e-mail: vandic@ese.eur.nl) and Flavius Frasincar (e-mail: frasincar@ese.eur.nl)
Erasmus University Rotterdam, Econometric Institute

information on Web pages. This is due to the fact that most Web pages are geared towards human-readability, rather than machine-understandability. Differently than machines, humans are able to extract the meaning of words from the context of Web pages, but a machine cannot do so. This is particularly a problem when searching using words that can have multiple senses, like ‘keyboard’, which can either be a computer device or musical instrument. The search engine will include every page that contains the search term in the search results, regardless of whether it is actually relevant or not.

A Web domain in which this search problem manifests itself is e-commerce. There is virtually an endless amount of products available and just as many Web shops from which you could order them. There are clear indications that better search functionalities are needed in order to keep online shopping transparent for the user. A study on online shopping in the USA, performed in [4], indicates that more than half of the respondents had encountered difficulties while shopping online. Information was often found to be lacking, contradicting, or overloading the user. This emphasises the need for aggregating the information found in those Web shops and presenting it in a uniform way.

While existing price comparison sites, such as [12], already show aggregated information, they are often restricted in their use. Most price comparison sites only include regional price information and therefore compare only a limited amount of Web shops [14]. Furthermore, in order to take part in the comparison, Web shops often have to take the initiative and provide their data in a specific format that is defined by each price comparison site. This can be laborious because there is no standardised semantic format for exchanging information. In other words, sharing product information on the Web requires a significant amount of manual work.

A solution to the search problems encountered on the Web would be to annotate the data found on Web pages using standardized ontologies. In this way the data becomes fully understandable for computers as well. For e-commerce, there is already a standard ontology emerging, called GoodRelations [3]. Unfortunately, not that many Web pages have included a semantic description for their content so far. Furthermore, even when a semantic description is available, not every Web page might use the same ontology. That is why there is a need for algorithms that are able to map product ontologies to each other in a (semi-)automatic way.

In this paper we propose an algorithm for mapping product taxonomies. Taxonomies are the backbone of an ontology, as they contain the type-of relations. Our algorithm is based on the approach presented by Park & Kim [11]. The proposed algorithm can autonomously map heterogeneous product taxonomies from multiple sources to each other. Similar to the Park & Kim algorithm, our algorithm employs word sense disambiguation techniques to find the correct sense of a term using the semantic lexicon WordNet [9]. Differently than the Park & Kim algorithm, our algorithm considers for lexical matching various lexical similarity measures, like the Jaccard index and the Levenshtein distance. Our proposed algorithm also exploits the hierarchical structure of taxonomies by taking into account the distance between each candidate path and already existing mappings. In addition, a different similarity aggregation function is used to make the algorithm more robust against outliers.

2 Related work

The algorithm we propose in this paper is based on the approach presented in [11], where the focus is more on product taxonomy mapping in particular, rather than ontology mapping. Due to this focus, it manages to achieve a higher recall than more general approaches for ontologies when mapping product taxonomies [11]. These approaches only map the classes when the similarity between these is very high. However, the design of product taxonomies is a subjective task, which makes their mapping a loosely-defined domain. Furthermore, the mapping of product taxonomies is aimed at reducing search failures when shopping online. Thus, in order not to lose potentially user-desired products from the Web shop presentation, it is better to map more product classes, even when classes are not very similar.

While the algorithm presented in [11] is suited for product taxonomy mapping, we found aspects that can be significantly improved. For instance, the algorithm does not consider the existence of composite categories, which are categories that consist of multiple concepts, e.g., ‘Movies, Music & Games’. Mapping these categories often fails, because the word sense disambiguation process is not applicable for these categories. This is due to the fact that the algorithm is unable to find the senses of the complete name in WordNet. Furthermore, the algorithm has difficulties disambiguating categories with short paths to the root, because of the lack of information content. This could be improved by also considering children and siblings of a category node when disambiguating. Another drawback of the algorithm is its bias towards mapping to short paths, which are sometimes proven to be too general.

Another approach for taxonomy mapping is Anchor-PROMPT, available in the PROMPT Suite [10]. This algorithm provides a (semi-) automatic ontology mapping process. As the performance of Anchor-PROMPT largely depends on the accuracy of the initial mappings that are provided (a requirement of the algorithm), it is not suitable for fully automatic ontology mapping [10]. Because the algorithm uses relatively simple lexical matching techniques to find the initial mappings, it would be better to manually create the initial mappings instead. This can become an issue when having to map many large product taxonomies. We have chosen to include this algorithm in our evaluation in order to investigate how a general ontology mapping algorithm performs in the context of product taxonomy mapping.

There are several other approaches that, despite the fact that their focus is on ontology mapping in general, are interesting to mention. The authors of [8] propose an algorithm to semi-automatically map schemas, using an iterative fixpoint computation, which they dubbed *similarity flooding*. In [6], the authors propose a semi-automatic ontology mapping tool, called Lexicon-based Ontology Mapping (LOM). The Quick Ontology Mapping (QOM) approach, presented in [2], is designed as a trade-off between the quality of a mapping and the speed with which a mapping can be performed. In [1], the authors present a system of Combination of Matching Algorithms (COMA++), capable of performing both schema and ontology mapping. Cupid [7] is a general-purpose schema matching algorithm. It is a hybrid matcher that exploits both lexical and semantic similarities between elements.

3 Algorithm

Before going into the algorithm details, we first present a high level overview of the algorithm, indicating the differences between our approach and the approach of Park & Kim. Our algorithm requires two inputs. The first input is a category and its path in the source taxonomy. The second input is the target taxonomy of categories to which the source category has to be mapped. Our algorithm starts with the preprocessing of the category name. It splits the name on ampersands, commas, and the word 'and', which results in a set containing multiple terms, called the *split term set*. This step is performed to enhance the word sense disambiguation process for categories that consist of multiple concepts, which are called *composite categories*.

The first major process (the same as in the Park & Kim approach) is the word sense disambiguation process. This process tries to determine the correct sense of the term in the leaf node of the source taxonomy by finding the term in WordNet (a semantic lexicon) [9]. The correct sense of the term can be found by comparing the hyponyms of each sense found in WordNet with all the ancestor nodes in the path of the source taxonomy. A path is a list of nodes from the root to the current node. Our algorithm repeats this process for each term in the *split term set*. The result of this process is the *extended term set*, which contains the original term and also its synonyms (if the algorithm was able to determine the correct sense of the term). Because our algorithm splits the original category name, we define the *extended split term set* as the set of all extended term sets (one for each split term).

Using the extended split term set that is obtained from the word sense disambiguation process, the algorithm analyses the target taxonomy and determines which paths are considered to be candidate paths for the mapping of the path from the source taxonomy. It does that by searching for paths that end with leafs that contain at least half of the terms in the extended (split) term set.

In order to determine which candidate path is the best path to map to, both algorithms compute the *co-occurrence* and *order-consistency* similarities for each path. The *co-occurrence* expresses the level of overlap between the source taxonomy path and one of the candidate target paths, while disregarding the hierarchy. The order-consistency is the ratio of common nodes (nodes that occur in both the source path and the candidate path) that appear in the candidate path according to the hierarchical order in the source path. Our algorithm adds a third measure, the *parent mapping distance*, which is the normalised distance in the target taxonomy between a candidate path and the path to which the parent in the source path was mapped to.

Using the similarity measures obtained in the previous step, the algorithms determine the best path to map the source path to. While Park & Kim use the arithmetic mean of the co-occurrence and order-consistency to obtain the overall similarity, our algorithm uses the harmonic mean (including the parent mapping distance). The path with the highest overall similarity is selected as the path to map to, assuming that the overall similarity is higher than a configurable threshold. If it fails to reach this threshold, or if no candidate paths were found, the algorithm of Park & Kim will not map the source path for a given input category. In this situation, our algorithm will map the source path to the same path in the target taxonomy to which its parent is mapped. If the parent of the source path was not mapped, our algorithm will also not map the source path.

Word Sense Disambiguation. As discussed previously, our algorithm splits composite categories and the split term set contains the individual terms that result from this process. This means that rather than using the entire category term for the word sense disambiguation process, like the algorithm of Park & Kim does, it will perform this process separately for each term in the split term set. Other than that, the implementation of this part of the algorithm does not differ from the implementation used by Park & Kim. Both algorithms enhance their ability to perform a correct mapping by first trying to determine the correct sense of a category term from the source taxonomy. This is useful, because it helps to identify semantically similar categories from different taxonomies, even when they are not lexically similar. For instance, if the path from the source taxonomy is ‘Computers/Notebook’, we can deduce that the correct sense would be a laptop in this case, rather than a notepad. We could then include the word ‘laptop’ in the search terms that are used for identifying candidate paths in the target taxonomy. This might yield better candidate paths than only searching using the term ‘notebook’.

In order to find the meaning that fits most closely to the source category that needs to be mapped, the algorithm identifies matches between an upper category, i.e., an ancestor of the current node from the source taxonomy, and a sense hierarchy obtained from WordNet. This is done by finding the set of matching lemmas between an upper category in the source taxonomy and a sense hierarchy defined by hypernym relations. By comparing each upper category from the source taxonomy with all sense hierarchy nodes that are in the set of matching lemmas, we can measure how well each upper category of the source taxonomy fits to each sense hierarchy. As the information content per node in a sense hierarchy increases when a node is closer to the leaf, we aim to find the match with the shortest distance to the sense hierarchy leaf. The similarity score increases when this distance is shorter, it is defined as:

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

where t is an upper category to be matched, C is the set of matching lemmas, and ℓ is the leaf of the sense hierarchy S. The `dist()` function computes the distance between each matching lemma x (of a synonym of a hypernym) and the leaf node ℓ in the sense hierarchy. The distance is given by the number of edges that are being traversed when navigating from the node with the matching lemma to the leaf node in the sense hierarchy.

After we have determined the hyperproximity between each upper category from a source path and a particular sense hierarchy from WordNet, we compute the overall similarity between an entire source category path and a sense hierarchy of the (split) category term. This is done by computing the average hyperproximity between all upper categories of a source path and one sense hierarchy from WordNet. Park & Kim use a different approach here, their algorithm divides the hyperproximities of each upper category by the length of the entire source category path, including the leaf node. This does not lead to a proper average, as the Park & Kim algorithm does not compute the hyperproximity between the leaf node and the sense hierarchy. Once the path-proximity between the source path and each of the possible senses of the source category term has been computed, we can determine which of the found senses fits best. This is done by selecting the sense hierarchy that has the highest average path proximity.

Candidate Path Identification. The resulting extended (split) term set of the word sense disambiguation process is used to identify candidate paths in the target taxonomy. A candidate path is a path in the target taxonomy that is marked by the algorithm as a potential target path to map the current source category to. In order to find the candidate paths, the algorithms compare the terms in the extended (split) term set with the paths in the target taxonomy.

The algorithm proposed by Park & Kim first compares the root node of the target taxonomy with the extended term set. If none of the terms in the extended term set is a substring of the currently examined category in the target taxonomy, the algorithm considers the children of the current category. Otherwise, if at least one of the terms in the extended term set is a substring of the currently examined category, that category is marked as a candidate path. In addition, the algorithm will no longer consider the children of that path as a potential candidate. The algorithm of Park & Kim assumes that if a more general category already matches the term, it is likely to be a better candidate path than a longer (more specific) path. However, this does not always hold true, as there are many composite categories in product taxonomies that split the multiple concepts in subcategories one level lower. For instance, the composite category ‘Music, Movies and Games’ in the Amazon.com product taxonomy has a subcategory called ‘Music’. Therefore, differently than the algorithm of Park & Kim, our algorithm continues to search the entire target taxonomy for candidate paths, even when an ancestor of a path was already marked as a candidate path.

As our algorithm splits the original category name if it is a composite category, it could occur that multiple extended term sets have to be compared with the category names in the target taxonomy. This means that our algorithm has to perform the matching for each extended term set in its extended split term set. This matching process returns a true/false value for each extended term set: true if one of terms is a substring of the currently examined category term, and false if none of the terms is a substring. If at least half of the boolean values is true, we consider the path of the current target category as a candidate path.

Aggregated Path Similarity Score. Once all the candidate paths in the target taxonomy have been identified, we need to determine which one of them fits best to the source paths. In order to calculate the measure of fit, we need to calculate an aggregated similarity score for each candidate path. In the algorithm proposed by Park & Kim, the aggregated score is composed of two similarity measures, the *co-occurrence* and the *order-consistency*. Our algorithm adds an extra measure, called the *parent mapping similarity*. Furthermore, it extends the co-occurrence measure by splitting terms and using the extended (split) term set of the correct sense. The algorithm proposed by Park & Kim uses only the original term or the synonyms of all the senses for the original term for calculating these similarity measures.

The co-occurrence is a similarity that measures how well each candidate path fits to the source category path that is to be mapped. It computes the overlap between two category paths, while disregarding the order of nodes in each path. This is done by applying a lexical matching function that is based on the average of Levenshtein and Jaccard similarities to each pair of categories from the source and candidate paths. The co-occurrence is defined as:

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

where P_{src} and P_{targ} are the nodes from the current source path and candidate path, respectively. The `maxSim()` function computes the maximum similarity between a single category name, either from the source or candidate path, and the entire path from the other taxonomy. It compares the single category name with all the nodes in the other path, using extended (split) term sets, obtained in the same way as in the candidate path selection.

The co-occurrence is useful for computing the lexical similarity between the source path and a candidate path from the target taxonomy. However, it disregards the order in which these nodes occur in the path. Therefore, we need an additional measure, which takes the order into account. Park & Kim define this measure as the order-consistency. The measure checks whether the common nodes between the paths appear in the same order. First of all, a list of matching nodes, called the *common node list*, between the two paths has to be obtained. The function `common()` adds a node to the list, if it can match the category term of a node, or one of the synonyms of the category term, with a node, or one of its synonyms, from the other path. In this function, all found senses from WordNet for the terms are taken into consideration. The resulting *common node list* is then used by `precRel()` to create binary node associations. These binary node associations denote a precedence relation between two nodes, which means that the first node occurs before the second node in the hierarchy of the source path. For every element in the *common node list*, pairs of node names from the source path are created. The `consistent()` function uses the precedence relations to check whether these precedence relations between two nodes also hold true for the candidate path, i.e., whether the two categories in the candidate path occur in the same order as the same two categories in the source path. If a precedence relation holds true also for the candidate path, this function returns the value 1, otherwise it returns 0. Using the aforementioned functions, the function for the order-consistency is given by:

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

where P_{src} and P_{targ} are the nodes from the current source path and candidate path, respectively, and C is $\texttt{common}(P_{\text{src}}, P_{\text{targ}})$. The denominator in the above fraction is the number of possible combinations of two nodes, which can be obtained from the *common nodes list*. Therefore, the order-consistency is the average number of precedence relations from the source path that are consistent with the candidate path.

The co-occurrence and the order-consistency both measure the similarity between the source path and a candidate path, computing the degree of category overlap and hierarchical order similarity, respectively. However, we can also exploit our knowledge of how the parent of the source node was mapped in order to find the best candidate path. As the current source node is obviously closely related to its parent, there is a considerable chance that the best candidate path is closely related to the target category to which the parent of the source node was mapped as well. That is why we include a third measure, called the *parent mapping distance*, which is the distance (difference) in the target taxonomy between a candidate path and the path to which the parent in the source path was mapped to.

Once the various similarity scores have been calculated, we can compute the aggregated similarity score for each candidate path. The algorithm proposed by Park & Kim computes the arithmetic mean of the co-occurrence and the order-consistency. However, the arithmetic mean is not very robust to outliers. The harmonic mean is more appropriate for aggregating the similarities, as it has a bias towards very low values, mitigating the impact of large outliers. Using the overall similarity measures for each candidate path, we can determine which one of the candidates is the best. We use a threshold to determine the minimal score needed to perform a mapping.

4 Evaluation

This section compares the performance of our algorithm against the performance of the algorithm by [11] and Anchor-PROMPT [10]. We evaluate the performance of the algorithms using three real-life datasets. The largest dataset, containing over 44,000 categories, was obtained from the Open Directory Project (ODP), available at http://dmoz.org. This dataset is relatively large, which makes it interesting for evaluation purposes, as it shows how well the algorithms scale when mapping large product taxonomies. The second dataset was obtained from Amazon.com, the largest online retailer in the USA. We have selected over 2,500 different categories in total with paths that have a maximum depth of five levels. The last dataset was obtained from Overstock.com, a large online retailer from the USA. It contained just over 1,000 categories with a maximum depth of four levels and it has a comparatively broad and flat taxonomy structure with many composite categories, which makes word sense disambiguation difficult.

Table 1 shows the average results of the six mappings per algorithm. As one can notice, our algorithm performs better on both recall and the F_1-measure than Anchor-PROMPT and the algorithm of Park & Kim. The recall has increased from 16.69% for Anchor-PROMPT and 25.19% for Park & Kim to 83.66% for our algorithm. Despite the clear improvement in recall, our algorithm actually performs slightly worse on precision than the algorithm of Park & Kim. The precision dropped from 47.77% to 38.28%. The high recall of our algorithm can be attributed to the fact that we are better able to deal with composite categories. The observed improvement in recall is also due to the usage of an approximate lexical matching function instead of an exact one for order consistency. As we perform more mappings to target categories, our precision declines slightly compared to that of the algorithm of Park & Kim.

Table 1 Comparison of average results per algorithm

Algorithm	Precision	Recall	F_1-measure	Computation time
Anchor-PROMPT	28.93%	16.69%	20.75%	0.47 s
Park & Kim	47.77%	25.19%	32.52%	4.99 s
Our algorithm	38.28%	83.66%	52.31%	20.71 s

Anchor-PROMPT maps more conservatively due to the fact that it is geared towards ontology mapping in general. Making classification mistakes in product taxonomy mapping is less severe than in most ontology mapping problems, because in an e-commerce domain it is considered more important to map many categories with some imprecision rather than only mapping a few categories with high precision. This is also reflected by the fact that the optimal thresholds for both our algorithm and the algorithm of Park & Kim were found to be very low. The average optimal similarity threshold, which determines whether a mapping is performed, is 0.025 for Park & Kim and 0.183 for our algorithm. In order to obtain the optimal thresholds, we have used a subset set for each mapping data set. We can conclude from the results that algorithms that are specifically tailored to mapping product taxonomies perform better than ontology mapping algorithms (such as Anchor-PROMPT) within this domain.

5 Conclusion

This paper proposes an algorithm suitable for automated product taxonomy mapping in an e-commerce environment. Our proposed algorithm takes into account the domain-specific characteristics of product taxonomies, like the existence of composite categories and the syntactical variations in category names. The algorithm we propose is based on the algorithm of Park & Kim [11]. Similar to the Park & Kim approach, we employ word sense disambiguation techniques in order to find the correct sense of a term. Differently than the Park & Kim algorithm, we consider various lexical similarity measures for lexical matching, like the Jaccard index and the Levenshtein distance. Furthermore, our algorithm exploits the hierarchical structure of taxonomies and uses a different similarity aggregation function, in order to make the algorithm more robust against outliers.

We have shown that our algorithm performs better than other approaches when mapping product taxonomies. Its performance on mapping three real-life datasets was compared favourably with that of Anchor-PROMPT [10] and the algorithm proposed by [11]. It manages to significantly increase both the recall and the F_1-measure of the mappings. We have also argued that recall is more important than precision in the context of online shopping.

For future work, we would like to employ a more advanced word sense disambiguation technique, such as the one proposed by [5]. It would also make sense to consider using word category disambiguation, also known as part-of-speech tagging, for the words found in a category name. Often the meaning changes when the part-of-speech is different. For example, ‘Machine Embroidery’ and ‘Embroidery Machines’, referring to machine-made embroidery and a machine which makes embroidery, respectively. By differentiating between adjectives and nouns, it is possible to identify these two meanings.

References

1. Aumueller, D., Do, H.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. Ehrig, M., Staab, S.: QOM - Quick Ontology Mapping. In: International Semantic Web Conference 2004 (ISWC 2004), vol. LNCS-3298, pp. 683–697. Springer (2004)
3. Hepp, M.: GoodRelations: An Ontology for Describing Products and Services Offers on the Web. In: 16th International Conference on Knowledge Engineering: Practice and Patterns (EKAW 2008), vol. LNCS-5268, pp. 329–346. Springer (2008)
4. Horrigan, J.B.: Online Shopping. Pew Internet & American Life Project Report **36** (2008)
5. Lesk, M.: Automatic Sense Disambiguation using Machine Readable Dictionaries: How to tell a Pine Cone from an Ice Cream Cone. In: 5th Annual ACM SIGDOC International Conference on Systems Documentation (SIGDOC 1986), pp. 24–26. ACM (1986)
6. Li, J.: LOM: A Lexicon-based Ontology Mapping Tool. In: 5th Workshop Performance Metrics for Intelligent Systems (PerMIS 2004) (2004)
7. Madhavan, J., Bernstein, P.A., 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)
8. 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 Computer Society (2002)
9. Miller, G.A.: WordNet: A Lexical Database for English. Communications of the ACM **38**(11), 39–41 (1995)
10. Noy, N.F., Musen, M.A.: The PROMPT Suite: Interactive Tools for Ontology Merging and Mapping. International Journal of Human-Computer Studies **59**(6), 983–1024 (2003)
11. 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)
12. Shopping.com: Online Shopping Comparison Website. http://www.shopping.com (2011)
13. Zhang, G.Q., Zhang, G.Q., Yang, Q.F., Cheng, S.Q., Zhou, T.: Evolution of the Internet and its Cores. New Journal of Physics **10**(12), 123027 (2008)
14. Zhu, H., Madnick, S., Siegel, M.: Enabling Global Price Comparison through Semantic Integration of Web Data. International Journal of Electronic Business **6**(4), 319–341 (2008)