Product Matching through Ontology Mapping in Comparison Shopping

Sangun Park[1], Wooju Kim[2], Sunghwan Lee[2], Siri Bang[2]

Abstract

Research on ontology merging and mapping is one of the most important issues in the Semantic Web because ontologies are developed and used by various sites and organizations respectively. Electronic commerce is the area that require ontology mapping on product comparison over different product classification taxonomies of various shopping malls. But, strict mapping strategy may lead a customer's configuration to search failure. Therefore we suggest a mapping algorithm for product matching that can provide more products by increasing sensitivity with reasonable decrease of specificity. We compared our algorithm with PROMPT based on experiments with 6 sets of source ontology and target ontology.

1. Introduction

Research on ontology merging and mapping is one of the most important issues in the Semantic Web environment because ontologies are developed and used by various sites and organizations respectively [3, 4, 6, 7, 11]. In electronic commerce area, each shopping mall has its own vocabulary and product hierarchy that cause a semantic interoperability problem [1, 14]. Gathering and merging product information from tremendous shopping malls in most product comparison sites depends on manual work by human. But, it is extremely inefficient to manage promptly changing information about products. That is, electronics commerce is the domain which essentially needs automatic ontology mapping on product names and attributes for efficient product search over multiple shopping malls.

Most research on ontology mapping [2, 5, 9] focuses on precision because incorrect matching among different ontologies can cause severe problems. PROMPT [11] is one of the approaches that adopt such conservative strategy with exact matching. But, product search in comparison shopping requires more flexible mapping between user's configuration and products. According to the Boston Consulting Group [13], 48% of all users have experienced unsatisfactory search results on desired products and 28% of all product purchase tryouts could not reach purchase because of search failure. If we use conservative and strict mapping on product search, customers will get only few products which exactly match with the configuration. Strict mapping strategy that may involve search failure is not desirable because customers want rich information on products.

Therefore, our research objective is to increase the number of matched products with the customer's configuration in automatic product mapping compared to the other ontology mapping

[1] The u-City Research Institute, Yonsei University, 134, Shin-Chon Dong, Seoul, South Korea, sangun.park@gmail.com

[2] Department of Information and Industrial Systems Engineering, College of Engineering, Yonsei University, 134, Shin-Chon Dong, Seoul, South Korea {wkim@yonsei.ac.kr;sunghwan@yonsei.ac.kr;siri27@nate.com}

approaches. This can be achieved by increasing recall rate with reasonable decrease in precision to satisfy the requirements of product search and comparison in electronic commerce domain.

2. Sensitivity and Precision

Precision can be calculated by dividing the number of correctly matched terms by the number of all matched terms as shown in (1) where *t_pos* is the number of correctly matched terms and *f_pos* is incorrectly matched terms [8]. Therefore, if one wants to enhance precision, the best way is to minimize incorrectly matched terms. That is the reason that most approaches of ontology mapping adopt conservative and strict strategy.

$$precision = \frac{t\_pos}{t\_pos + f\_pos} \qquad (1)$$

Meanwhile, *sensitivity* divides the number of correctly matched terms by the number of terms that should be matched, *pos* as shown in (2) [8]. Strict matching strategy tries to increase precision as much as possible in spite of low sensitivity. But, this strict strategy is not desirable in comparison shopping as we mentioned in Section 1.

$$sensitivity = \frac{t\_pos}{pos} \qquad (2)$$

Specificity is used with sensitivity together for classification performance measures and calculated as shown in (3) where *neg* is the number of terms that should not be matched and *t_neg* is the number of correctly not matched terms [8]. If we try to increase sensitivity by matching more products, specificity can be worse because correctly non-matched terms will decrease. Therefore, we use sensitivity and specificity in the performance evaluation and comparison of our algorithm and PROMPT.

$$specificity = \frac{t\_neg}{neg} \qquad (3)$$

Then, how to increase sensitivity compared to exact matching? The easiest way is using synonyms from WordNet [10]. By matching all synonyms of the given product, we can match more products and increase the chance of matching more correct products. But, it also can decrease precision because *f_pos* in (1) increases. So, using only synonym is not recommendable. In WordNet, a word has different senses and each sense has its own synonyms. If we can choose an appropriate sense of the given product from WordNet, it is possible to prevent precision from dropping too much by narrowing the synonym range.

Also, a product has upper categories in its product hierarchy. We can use the hierarchy to choose appropriate matching products from the target ontology by comparing hierarchies of the source and target ontology.

In this paper, we propose an ontology mapping algorithm for product matching based on above two ideas. The algorithm searches for an appropriate product in a target ontology which matches with a given product of a source ontology.

3. Product Matching through Ontology Mapping

There are three steps in our ontology mapping algorithm. The first step is word sense disambiguation. This means choosing appropriate sense for a given product in a source ontology by comparing the product hierarchy and hypernym structure of each sense. The second step is searching for product candidates from a target ontology by using synonyms of the sense which was acquired at the first step. The last step is choosing the most appropriate product from candidates by comparing product hierarchies between the source ontology and the target ontology.

3.1. Word Sense Disambiguation for Product Categories

Selection of an appropriate sense for a given product is important to keep precision at a reasonable level. If we use synonyms of all senses of the product, it will decrease precision because incorrect matching can increase. Moreover, word sense disambiguation can enhance precision. For example, when a customer selects *notebook* which is a computer, a book for notes can be matched for *notebook* in addition to notebook computers. The basic idea of word sense disambiguation is comparing a product hierarchy and hypernym hierarchies of senses of the product in WordNet. The sense *notebook* that is a computer has a different hypernym hierarchy with that of a book for notes as shown in Fig. 1. By comparing the product hierarchy of ODP (Open Directory Project) [12] in the left column of Fig. 1 and hypernym hierarchies in WordNet of the right column, we can choose a proper sense for *notebook*.

The first step of disambiguation is searching for hypernyms from a hierarchy of a sense that match with upper categories of the product. For example, *notebooks* in ODP has three upper categories, *consumer electronics, computers,* and *systems*. Therefore, we search for matching terms from hypernyms of senses as shown in the formula (4).

$$cs(x, p) = \{h \mid h \in SYNSETS(x) \text{ and } h \in hypernyms(p)\} \quad \text{where } x \text{ is an upper category of the product hierarchy} \tag{4}$$

CS() returns a set of hypernyms that match to a given upper category x from a given sense hierarchy p. The following shows the results for upper categories of *notebooks*.

CS(system, sense_2) = {}
CS(computers, sense_2) = {computer}
CS(consumer electronics, sense_2) = {}

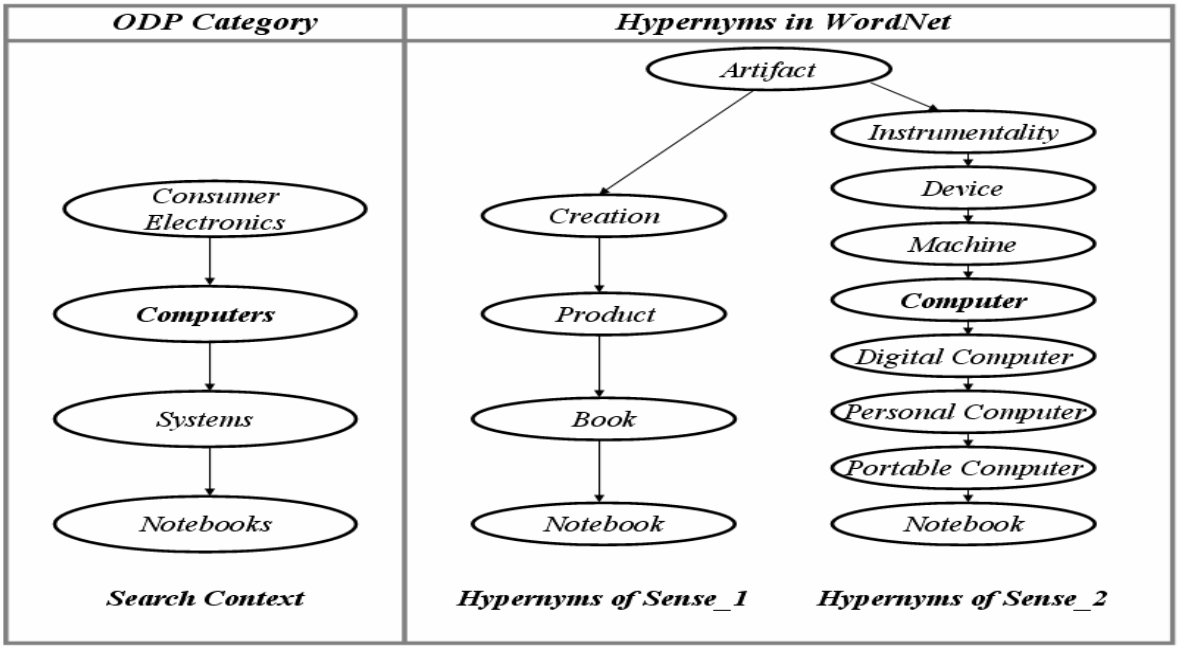

Fig. 1. A Product Hierarchy of ODP and Corresponding Hypernym Hierarchies in WordNet

The next step is calculating a measure represents the similarity between an upper category and a sense. If a matching hypernym is close to the sense, then the similarity is high because a closer hypernym is more important. The function *hypernymproximity()* returns the similarity by calculating a minimum distance between the matching hypernym and the base node of the sense in the hypernym hierarchy as shown in (5).

$$hypernymproximity(x, p) = \begin{Bmatrix} \dfrac{1}{Min\_dist(cs(x, p), base)} & if\ cs(x, p) \neq \phi \\ 0 & otherwise \end{Bmatrix} \quad (5)$$

The distance between the nodes is the number of arcs between them. For example, sense_2 of Fig. 1 has the term Computer in its hypernym hierarchy that matches with the upper category Computers. The number of arcs between Notebook and Computer in sense_2 is 4. Therefore, hyperproximity(computers, sense_2) is 1/4. We can calculate hyperproximity for all upper categories as follows.

hyperproximity(Systems, path 2) = 0

hyperproximity(Computers, path 2) = 1/4

hyperproximity(Consumer Electronics, path 2) = 0

The last step is calculating similarity between a product and senses. The function *pathproximity()* adds all *hyperproximity* of a given sense and divides it by the number of nodes of the product hierarchy as shown in (6).

$$pathproximity(p) = \frac{\sum_{x \in upper\_categories(base)} hypernymproximity(x, p)}{n} \quad (6)$$

For example, *pathproximity* of *sense_2* can be calculated as follows based on the above *hyperproximity* results:

$$pathproximity(sense\_2) = \left(\frac{0 + \frac{1}{4} + 0}{4} \right) = 0.0625$$

In our example, pathproximity of sense_1 is 0 because there is no matching hypernym for the upper categories in sense_1. Therefore, sense_2 is selected for the appropriate sense and we use the synonyms of sense_2, {notebook, notebook computer} in the remaining steps.

3.2. Generation of Candidates for the Best Matching Category Path

Once we found an exact sense for the product from WordNet, the next step is to search for the candidates for the best matching category path from a target ontology.

Searching for candidates uses the synonyms that acquired from the previous step. In our example, we will search for product categories that match with {notebook, notebook computer} in Amazon.com ontology. We could get 'Notebooks', 'Notebook', and 'All Notebooks' from product categories. After the completion of search, we need to delete redundant categories of the product. To do this, the algorithm generates serial hierarchies of the categories by extracting all upper categories. We generated three hierarchies in our example as follows:

/Product/ Electronics /Accessories/Computer Accessories/Notebook
/Product/Electronics/ Computers/Notebooks
/Product/Electronics/Computers/Notebooks/All Notebooks

3.3. Choice of the Best Matching Product Category

To choose the best matching product category, we designed two measures for the calculation of similarities between the given product hierarchy and candidates. One is *co-occurrence* and the other is *order-consistency*. The measure *co-occurrence* is the ratio of the number of common categories between a source hierarchy and a target hierarchy to the number of categories of the target hierarchy.

Fig. 2 shows the source hierarchy and two target hierarchies in our example. The two target hierarchies have two common matching categories, *Electronics* and *Computers*(*Computer Accessories*), in addition to *Notebooks*(*Notebook*) as shown in Fig. 2. But, the upper target

hierarchy has 4 categories and the lower has 5 categories. Therefore, *co-occurrence* of the target hierarchies are 3/4 and 3/5 respectively.

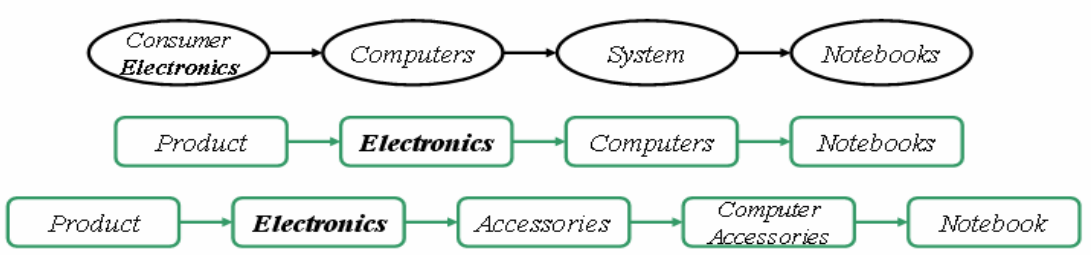

Fig. 2. The Source Hierarchy and the Target Hierarchies

However, co-occurrence is not enough to represent similarity because co-occurrence cannot measure orders of categories in the hierarchy. As shown in Fig. 3, Y1 and Y2 have the same co-occurrence value, but Y1 is more similar to Search Context than Y2 because the order of categories in Y1 is the same with the order in Search Context while the order of Y2 is different. The other measure *order-consistency* compares this order of categories. The order in Search Context is subdivided into precedence relations that are binary relations of two nodes. For example, There are three precedence relations, {a, b}, {b, c}, and {a, c} in Search Context. Y1 keeps all the precedence relations while Y2 keeps only two.

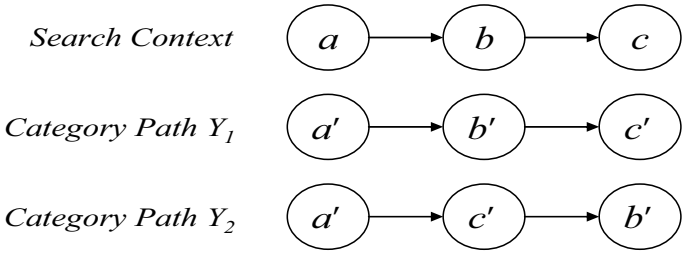

Fig. 3. An Example Search Context and Category Paths

The final similarity between a source product and a target product is the average of co-occurrence and order-consistency. We choose a threshold on the similarity to determine whether we match the source product with the target product or not. We expect that the matching result will be changed by controlling not only the ratio of co-occurrence and order-consistency to the similarity but also the threshold.

4. Empirical Evaluation and Results

In this section, we compare the mapping results between our algorithm and PROMPT. PROMPT compares two different taxonomies and automatically recommends the matching terms by using synonyms [11]. Furthermore, PROMPT enables users to do direct mapping. However, it is

impossible to perform direct mapping in product search. Therefore, we consider only the recommended values by PROMPT

4.1. Experiment Design for Evaluation

To conduct an experiment, we selected two well-known shopping malls –Amazon.com and Buy.com – and ODP [12]. In the case of Amazon.com and Buy.com, product category mapping is fairly easy because they provide similar product categories. However, shopping categories of ODP use different product names and classifications from the shopping malls. Therefore, we can expect that product category mapping between ODP and shopping malls will show lower performance than between two shopping malls.

We constructed product ontologies from Amazon.com, Buy.com, and ODP respectively for our experiment. The product ontology of Amazon.com consists of 136 nodes, Buy.com consists of 225 nodes, and ODP consists of 133 nodes. A set of the experiment consists of one source ontology and one target ontology. For example, Amazon.com for the source ontology and Buy.com for the target ontology is one set. Therefore, there are 6 sets in the experiment. As the results of the experiment, we obtained 12 sensitivity and specificity pairs for our algorithm and PROMPT.

4.2. Experiment Results

Table 1 shows the performance results on sensitivity and specificity. On average, sensitivity of our algorithm is better than PROMPT by 46.5% and worse by 24.3%. It shows that our objective is successfully achieved. The maximum and minimum differences of sensitivity are 67.6% and 24.8% respectively while the maximum and minimum differences of specificity are -37.6% and -5.6% respectively.

Table 1. Performance Results on Sensitivity and Specificity

Experimental Set	Sensitivity		Specificity	
	Our Algorithm	PROMPT	Our Algorithm	PROMPT
Amazon → Buy	96.9%	61.7%	56.4%	91.1%
Amazon → ODP	93.3%	25.7%	78.9%	84.5%
Buy → Amazon	93.5%	56.0%	61.0%	94.8%
Buy → ODP	97.2%	40.6%	69.5%	89.6%
ODP → Amazon	92.9%	36.0%	50.5%	88.1%
ODP → Buy	85.7%	60.9%	70.5%	84.7%
Average	**93.3%**	**46.8%**	**64.5%**	**88.8%**

Fig. 4 shows the comparison graph on sensitivity and Fig. 5 shows the comparison on specificity. Our algorithm shows the best performance in product matching from Amazon.com to ODP compared to PROMPT with 67.6% of sensitivity increase and only 5.6% of specificity decrease. It will be the best if we can improve sensitivity and specificity at the same time. But, it is hard to accomplish. Therefore, we need to balance them in order to fulfill requirements of the given

domain. At least in electronic commerce domain, we expect that our algorithm is better than PROMPT.

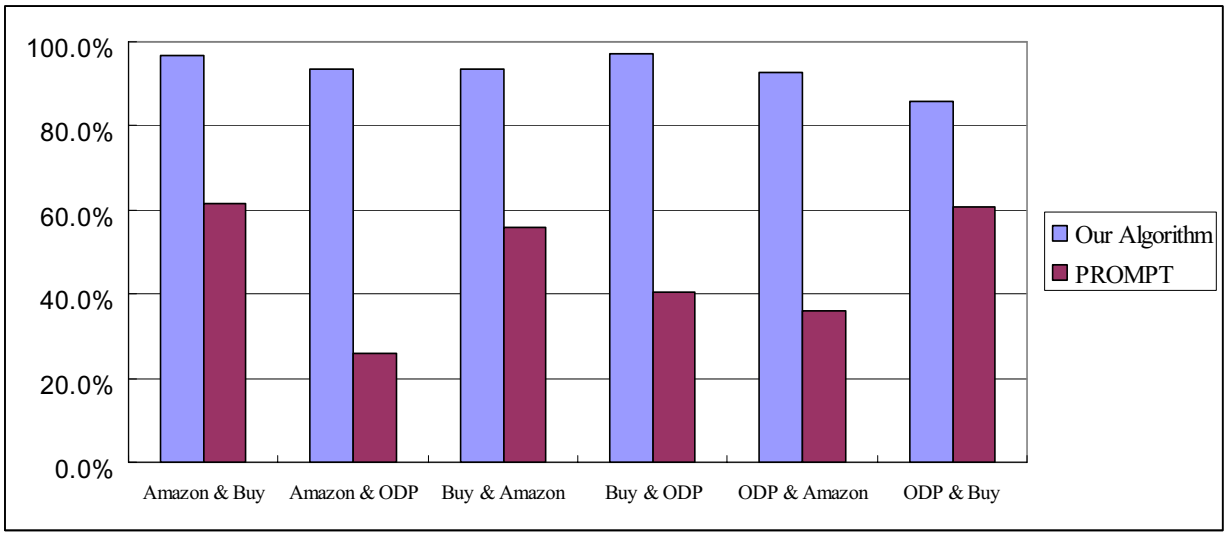

Fig. 4. Comparison of Sensitivity

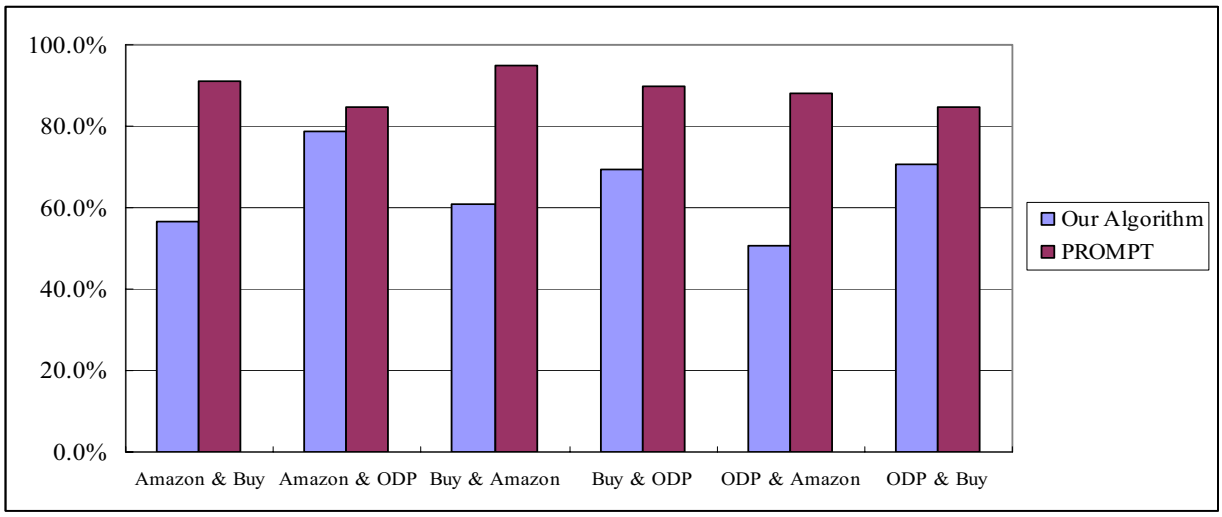

Fig. 5. Comparison of Specificity

5. Conclusion

In this paper, we proposed an ontology mapping algorithm that provides efficient product matching between heterogeneous product classifications. And, we performed a comparative evaluation between our algorithm and PROMPT with 6 experimental sets. The experiment results showed that our algorithm is more effective than PROMPT in product comparison of electronic commerce domain.

There is an interesting future research issue. Sensitivity and specificity can be changed by controlling not only the ratio of *co-occurrence* and *order-consistency* to the similarity but also the threshold as we described in Section 3. We expect that we can find the optimal values of the parameters – the ratio and the threshold. We are planning to conduct experiments finding the optimal values.

There are huge application areas of ontology mapping, but it is inefficient to apply the same mapping strategy to all the areas. We suggested a mapping algorithm which is appropriate for product comparison in electronic commerce domain by increasing sensitivity. We expect that our research contributes the practical application of ontology mapping on electronic commerce.

References

[1] Benetti, H., D. Beneventano, S. Bergamaschi, F. Guerra, and M. Vincini, An Information Integration Framework for E-Commerce, IEEE Intelligent Systems 17(1) (2002).

[2] Castano, S., A. Ferrara, and S. Montanelli, H-MATCH: An Algorithm for Dynamically Matching Ontologies in Peer-based Systems, Proc. of the 1st VLDB Int. Workshop on Semantic Web and Databases (SWDB 2003), Berlin, Germany (2003).

[3] Decker, S., M. Erdmann, D. Fensel, and R. Studer: Ontobroker, Ontology Based Access to Distributed and Semi-Structured Information, Proc. of DS-8, Semantic Issues in Multimedia Systems, Boston, MA, USA, ed., R & et.al. Meersman (1999) 351-369.

[4] Ehrig, M. and S. Sraab, QOM: Quick Ontology Mapping, Lecture Notes in Computer Science, 3298 (2004) 683-697.

[5] Ehrig, M. and Y. Sure, Ontology Mapping - An Integrated Approach, Lecture Notes in Computer Science, No. 3053 (2004) 76-91.

[6] Gal, A., G. Modica, and H. Jamil, Improving Web Search with Automatic Ontology Mapping, Mississippi State Univ, Working Paper (2003).

[7] Guarino, N., C. Masolo, and G. Vetere, OntoSeek: Content-Based Access to the Web, IEEE Intelligent Systems, 14(3) (1999) 70-80.

[8] Han, J. and M. Kamber, Data Mining: Concepts and Techniques, Morgan Kaufmann Publishers (2000) 325-326.

[9] Kalfoglou, Y. and M. Schorelmmer, Ontology mapping: the state of the art, The Knowledge Engineering Review, 18(1) (2003) 1-32.

[10] Miller, G. A., WordNet a Lexical Database for English, Communications of the ACM, 38(11) (1995) 39-41.

[11] Noy, N.F. and M.A. Musen, The PROMPT Suite: Interactive Tools for Ontology Merging and Mapping, International Journal of Human-Computer Studies, 59(1) (2003) 983-1024.

[12] Open Directory Project, http://www.dmoz.com, 2006.

[13] Pecaut, D., M. Silverstein, and P. Stanger, Winning the Online Consumer Insights into Online Consumer Behavior, A Report by the Boston Consulting Group, <http://www.bcg.com>, 2000.