Ontology-Aided Product Classification: A Nearest Neighbour Approach

Alastair A. Abbott and Ian Watson

Department of Computer Science,
University of Auckland,
Auckland, New Zealand
aabb009@aucklanduni.ac.nz, ian@cs.auckland.ac.nz

Abstract. In this paper we present a k-Nearest Neighbour case-based reasoning system for classifying products into an ontology of classes. Such a classifier is of particular use in the business-to-business electronic commerce industry, where maintaining accurate products catalogues is critical for accurate spend-analysis and effective trading. Universal classification schemas, such as the United Nations Standard Products and Services Code hierarchy, have been created to aid this process, but classifying items into such a hierarchical schema is a critical and costly task. While (semi)-automated classifiers have previously been explored, items not initially classified still have to be classified by hand in a costly process. To help overcome this issue, we develop a conversational approach which utilises the known relationship between classes to allow the user to come to a correct classification much more often with minimal effort.

1 Introduction

In business-to-business (B2B) e-commerce, the problem of classifying products for the purpose of maintaining electronic product catalogues (EPCs) can be particularly difficult and costly. Since different suppliers often use different product classification standards, products from different suppliers need to be reclassified into the standard required [8]. This is a problem not only affecting businesses which need to maintain catalogues of products available from various suppliers, but also in the greater e-procurement industry. Well maintained and classified EPCs are of particular importance given the movement to electronic business practices and allow for accurate spend-analysis and increased efficiency in both finding and selling products [14].

The gold-standard solution would be to have a standardised classification system valid across all domains; this is the primary goal of the UNSPSC (United Nations Standard Products and Services Code) schema.[1] However, due to differing requirements of businesses the UNSPSC standard might not be suitable for everyone, and other classification schemes such as eCl@ss[2] are also widely used.

[1] http://www.unspsc.org
[2] http://www.eclass-online.com

A. Ram and N. Wiratunga (Eds.): ICCBR 2011, LNAI 6880, pp. 348–362, 2011.

Further, the process of transferring existing catalogues to a different schema might simply be too costly to be justifiable. As such, even for those businesses using the UNSPSC standard, products from other suppliers must be re-classified into this standard making this a perpetual issue.

The manual classification of products into a standard such as the UNSPSC is a time-consuming and costly process: some studies have indicated it can take 5–10 minutes to classify an item into a complex standard such as the UNSPSC, primarily because of the large number of similar sounding classes [6]. Automated or semi-automated systems to perform this task are a promising (and perhaps inevitable) solution since this task fits well within the Machine Learning domain, and have significant potential to cut cost and time [2]. Some existing automatic classification systems have been developed, primarily using statistical methods [5,8]. In this paper we develop a k-Nearest Neighbour (kNN) case-based reasoner to aid this classification task. Since the accuracy of such systems (particularly on difficult products) is not sufficient to allow fully automated classification, we use the ontological structure of the classification hierarchy to aid a conversational approach to classification. This dramatically reduces the cost to classify new items, and allows an accurate classification to be achieved with minimal user interaction.

2 Background

2.1 The UNSPSC Schema

The UNSPSC is a four-level hierarchical classification schema [14]. It was designed by the United Nations and Dun & Bradstreet as a universal system to classify goods and services across all domains and ease spend analysis, the finding and purchasing of products, and the maintenance of product catalogues. Each class, or "commodity," is assigned a unique 8-digit code and the schema is designed so that every product or service should fit uniquely into a class.

The levels of the schema are shown in Fig. 1, and two of the eight digits of the commodity code come from each level [14]. For example, in Fig. 2 the commodity "Staplers" would have a commodity code of 44121615.

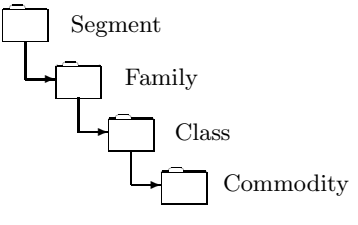

Fig. 1. The UNSPSC class hierarchy structure

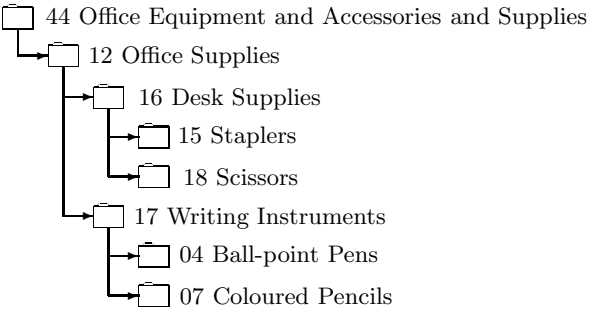

Fig. 2. An example (partial) snapshot of the UNSPSC schema

2.2 Previous Work

Ding et al. developed a system for automatic product classification into the UNSPSC schema called GoldenBullet [5]. Using a Naïve-Bayes approach, Ding et al.'s classifier correctly classified products with an accuracy of 78% when using the top result returned by the classifier. The correct classification was in the top 10 results a slightly higher 88% of the time. Ding et al. also explored using a kNN classifier as well as a vector-space model classifier, but found these were much less accurate. Unfortunately, the nature of the product catalogue they used is not specified other than that it consisted of 40,000 items primarily in the electronics domain. Crucial factors such as the quality of the data or spread of classes encompassed were not discussed.

Another classifier is described in [8], which also uses the Naïve-Bayes method, and obtained a similar accuracy of 86% based on the top result returned. Their data was obtained from an e-procurement database maintained by the Public Procurement Services of Korea, a government run organisation. This uses a classification schema based on the UNSPSC hierarchy, and the database that was used contained almost 500,000 items spread over 7960 classes.

Since the accuracy of a classifier is heavily reliant on the dataset used, the lack of information or availability of datasets used for these classifiers makes comparison of results extremely difficult. Certain types of classifier may be inclined to work better on particular types of dataset. Further, as was noted in [8], the quality of the dataset used will significantly affect the accuracy.

3 Classifier Development

In this paper we detail a project carried out in collaboration with with Unimarket,[3] a locally based e-procurement business. In order to provide a good e-procurement service it is necessary to maintain accurate catalogue data, something which necessarily entails working with a range of classification systems from various suppliers. This is crucial for the facilitation of B2B trading and

[3] `http://www.unimarket.co.nz`

accurate spend analysis services where expenses across different classes of commodities and services must be readily compileable.

No suitable datasets using the UNSPSC schema were available from Unimarket or elsewhere, and instead data from Amazon using their classification hierarchy were used for testing and development. While we could have looked at transferring the data into the UNSPSC hierarchy, this migration process is primarily a mapping rather than a classification task and is beyond the scope of this project—it cannot be done completely automatically from scratch. Either an initial mapping between classes across schemata is required, or an initial subset of data would need to be manually classified before a (semi)-automated system could complete the task. Rather, the task of the project was to develop a tool to classify new products into the classification schema. This could not only be used to complete the migration to the UNSPSC hierarchy, but is necessary for catalogue maintenance where it would be of use in the ongoing task of adding new products from suppliers correctly and efficiently into the hierarchy.

3.1 Solution Outline

In this paper a kNN Instance Based Learner [16] is explored as a proof-of-concept for a new classification system. In existing systems such as GoldenBullet the significant 10–20% of misclassified items require a costly manual classification procedure to be applied. As a result, a considerable degree of user interaction is inevitable and the system cannot reasonably run in a completely autonomous fashion; in practice the manual classification of the 10–20% of products the automated system struggles with is costly and a limiting factor. To improve on this and handle more constructively the difficult cases, our proposed learner uses a conversational approach to classification. This allows much greater accuracy to be achieved at the cost of some minor user interaction for each classification—an overall gain when compared to classifying difficult items manually from scratch.

To achieve this aim, an ontology of the hierarchical relationship of classes is used, allowing intelligent questions to be asked based on the inferred relationships of candidate classification solutions, in turn aiding the classification of classes and improving the accuracy in a "mostly-automated" approach. The goal of the automated classifier is hence to classify correctly with a minimal number of questions; inevitably some difficult cases will be as bad as the worst case possibility, but this can be drastically reduced when compared to existing approaches. This technique is similar to the method of critiquing [9] in Case Based Reasoning (CBR) recommender systems.

The approach taken in developing our classifier was to build a CBR classifier [1]. In this setup, products in the catalogue are stored as *cases* in the *case-base*. An item to be classified is regarded as a query, and a global similarity metric is used to find the most similar cases in the case-base, and classify the query based on these. We opted to use the kNN algorithm for the global similarity metric, which retrieves the k cases (products) which are most similar to the query based on the similarity of individual attributes. These k results will then be used to guide the conversational procedure, and with care can be made

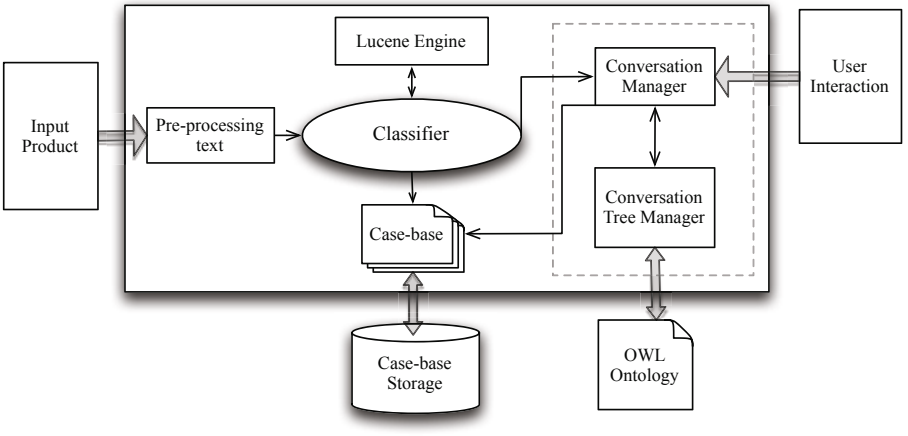

Fig. 3. Architecture diagram of the CBR system

to return an optimal range of queries allowing the conversational procedure to be both accurate and efficient (in terms of number of questions asked).

An OWL (Web Ontology Language) ontology[4] storing the hierarchy of product classes was created from Amazon's existing classification hierarchy. For the UNSPSC hierarchy, OWL format ontologies are already available for use. The ontology stores the hierarchical relationship between different classes which will be utilised by the classifier in the conversational stage.

In order to implement the CBR classifier, the open-source Java CBR development framework jCOLIBRI[5][4,10] was used. jCOLIBRI allows for flexible development of various types of CBR systems. It includes functionality for pre-defined and user-defined global and local similarity metrics, has built-in (and extendable) functionality for testing the accuracy of the CBR systems, and functionality to integrate with OWL ontoligies via the OntoBridge[6] framework [12]. This allowed the class associated with a product in the case-base in our case representation to be easily and dynamically linked into the hierarchy of classes stored in the ontology file. The overall design of the system is shown in Fig. 3.

3.2 Data

Data were collected from Amazon[7] through the *Amazon Product Advertising API*.[8] These data are hierarchically structured and representative of a large number of classes. Since Amazon allows products to be classified under more than one category, care was taken to only sample a product once so the situation

[4] http://www.w3.org/TR/owl-ref/
[5] http://gaia.fdi.ucm.es/projects/jcolibri/
[6] http://gaia.fdi.ucm.es/grupo/projects/ontobridge/
[7] http://www.amazon.com
[8] http://affiliate-program.amazon.com/gp/advertising/api/detail/main.html

could be treated as a single-label scenario. This hints at some of the difficulties of classification as a product may reasonably fit into more than one category even in a schema such as the UNSPSC, but our aim was to develop a classifier suitable for such a single-label hierarchy. The Amazon class hierarchy is not as consistent as the UNSPSC hierarchy: classes at the same depth in the hierarchy may not be of comparable generality. However, given the size of the case-base and the reasonably good quality of the data, an automated system should be able to function fairly well, and the conversation stage should help deal with some of the issues with the class structure. This reiterates the benefits of adopting a unified schema such as the UNSPSC, and the classifier is only expected to function better on such an organised hierarchy.

An important step was to determine which attributes should be used for classification. The Amazon data is attribute-rich, but most attributes were deemed irrelevant for classification—indeed, too many attributes can be detrimental if they are not relevant enough to the classification. The majority of attributes were deemed to be irrelevant either by mere consideration, or were found to contribute little, if at all, to the accuracy of the classifier during initial testing. This selection was performed manually with the aim of developing a general model, and care was also taken to choose attributes which one would expect to be present in most kind of catalogues to increase the portability of the system. The attributes which were eventually determined to be useful for classification were:

Store (supplier). Often the supplier will give a good indication of the type of product supplied.

Product Name. Clearly this contains information about the class a product should belong to.

Product Description. As for *Product Name*.

A primary case base consisting of over 23,000 items was obtained, with items classified into approximately 140 classes (over 3 top-level classes). Sample data is shown in Table 1. Around a third of the items were identified to be near-identical to other items (varying perhaps, for example, only in the colour of the item), an effect which would skew the results making the accuracy appear falsely high. Since such items are likely to be added to the catalogue (and classified) all at once, fair testing of the system should work with only one of each such item in the case-base. Further, this can have a strong detrimental impact on the diversity of results obtained by the kNN search and thus affect the accuracy of the conversational stage. This is likely to be a problem in many real-world systems, so a reasonable approach is to only use one of each group of items as a representative in the case-base for the classification procedure. Indeed, this approach is motivated by footprint-based retrieval [13], and although differs in its goal it is similarly based on the assumption that the joint competence of such groups of cases is no different from that of the representative chosen.

So that testing would more accurately mimic the real world situation where such similar items would be classified together, the dataset was pre-processed. The pre-processing removed all but one of these nearly identical items—i.e. those

Table 1. A snapshot of the Amazon data

ID	Store	Name	Description	Class
B002EA1EZE	Ultimark	ultimark pre inked stock message stamp faced...	ensure efficient communication and provide clear instructions...	ID490540011
B000V27TCO	Sato	sato industrial labels...	by sato industrial labels per roll	ID490540011
B002ZHQ3KW	Discount Rubber...	self inking christmas rubber stamp...	spread a little holiday cheer with christmas rubber stamps...	ID490540011
B0006HX78Y	Pendaflex	pendaflex pressboard end tab guides...	pt pressboard includes a xyz and mc tabs	ID490540011
B00007LP9W	Avery	avery clip style rigid laserink jet badges...	each kit includes clear plastic badge holders and sheets of...	ID490540011

from the same supplier with similarity above an empirically determined threshold. This similarity was determined using the same similarity metrics as were used for classification, and a conservative approach was taken as it is better to accidentally remove unique items from the case-base than to leave similar items in the case base, potentially skewing results.

All textual data (which is the primary focus in classification) was stripped of punctuation, numbers and product dimensions (e.g. width, height, weight etc.) automatically, as descriptions often contained these technical details of a product that are of little value in determining the class a particular item belongs to.

3.3 Classification Details

The kNN algorithm was used as the global similarity metric of choice, assigning a value in $[0, 1]$ to a pair of cases based on their similarity. This is calculated as the weighted sum over the local similarity metrics of the supplied query attributes. If i indexes the n attributes of the cases and f_i is the local similarity metric for the ith attribute, the kNN similarity between two cases Q and T is calculated as

$$\text{Similarity}(Q, T) = \frac{\sum_{i=1}^{n} f_i(Q_i, T_i) \times w_i}{\sum_{i=1}^{n} w_i} \ .$$

The k items with the highest similarity to the query are selected, and the conversational stage of the classification procedure is entered. In our setup we chose to use a value of $k = 10$, which seemed to strike a good balance between the number of choices the user was given based on these results, and the accuracy of the classification.

The weighting used placed relative weights of *Store*: 2, *Product Name* and *Product Description*: 5 each, determined to give the best results in testing. In cases where data was missing for a particular attribute, the attribute is given a weighting of 0 so that it is not taken into consideration. This was empirically determined to produce better results than treating this as a local similarity value of 0, which would instead indicate that the cases were actually dissimilar with respect to this attribute.

As for the global similarity metric, the local similarity metrics f_i assign a value in $[0, 1]$ to a pair of cases based on the similarity of the ith attribute. The local similarity metrics used for each attribute were as follows:

Store (supplier). A similarity of 1 is assigned if and only if the suppliers of the cases are identical. In all other cases a score of 0 is used, as the similarity between suppliers cannot otherwise be quantified without auxiliary knowledge on the type of products various suppliers supply.

Product Name. Since this is a textual attribute, textual similarity methods must be used. For this application it is necessary to employ domain independent textual retrieval similarity methods. We opted to use the Lucene textual comparison engine,[9] based on information retrieval techniques, for which jCOLIBRI includes built-in functionality [11]. The Lucene textual comparison is detailed more fully in the following paragraphs.

Product Description. As for *Product Name*, the Lucene textual comparison engine was used.

Since two out of three attributes consist entirely of textual data (and these are the most heavily weighted attributes, accounting for 10/12ths of the weighting), the textual comparison methods are critical and hence deserve further discussion. While textual CBR usually refers to the situation in which the cases themselves are stored as blocks of text [15], techniques from textual CBR are equally applicable for use in classifying products when combined with a kNN global similarity function [11]. As such, textual methods suitable for the type of data which product descriptions consist of should be chosen appropriately.

In particular, textual similarity functions must be domain independent, as the product data is not confined to a particular domain. Further, suppliers do not necessarily provide structured paragraphs of information—the data may consist of lists, technical data and other unstructured text. Hence, semantic similarity functions would also be inappropriate. This limits the ability to compare cases, but is an inherent feature a classifier for this problem must cope with. As such, the obvious option is to use a statistical approach.

The Lucene engine used is based on a combination of the Vector Space Model from the Information Retrieval domain and a Boolean model [7,11]. It is a keyword based search method, which analyses the relative frequencies of the terms in the query in the cases in the case-base. In order to perform this efficiently, the case-base is pre-processed on startup to extract the statistical information required, and then the search query is analysed against this [7,11].

Advanced techniques and modifications were also investigated, such as checking if nodes in the ontology were present as words in the text, as this is a strong indicator for narrowing the search. However, the effect of this was not significant, especially given the extra cost incurred in the search procedure. Given the large weighting the textual attributes hold extra effort spent implementing more advanced techniques would pay off in accuracy in real world deployments of such a classifier. In this paper, however, we are focusing on using the ontology

[9] `http://lucene.apache.org`

to guide the conversational stage, and for this the Lucene engine, which is built into the jCOLIBRI framework, is sufficient.

The final, important part of the classification process is, of course, the conversational stage of the process. As can be seen in Fig. 3, this acts as a sub-unit of the system mediating the input from the user with the ontology, case base and classification engine. It is this stage which differentiates this proposal most strongly from existing proposals (although also based on different classification algorithms), and has the most significant impact on the most costly phase of classification: when the top or most common result is not the correct classification. The ontology which represents the hierarchy of relationships between classes can be regarded as a rooted tree where the depth of each leaf is identical, leaf nodes correspond to individual classes (commodities in the UNSPSC hierarchy), and internal nodes correspond to higher level classes. The root itself can be regarded as a "Thing" class, although presumably nothing is classified under this category directly. Given the k cases returned by the kNN algorithm, the $m \leq k$ classes present in these results are extracted as potential classes.

The nodes corresponding to these classes are identified within the ontology tree, and the subtree that these induce is extracted. In other words, all classes which are not an ancestor of one of the m candidate classes are removed from the tree. In general, however, this still leaves many classes which are not one of the m candidate classes in the extracted subtree. To reduce this to a minimal structure, all edges between nodes (except the root) of degree two are contracted, i.e. non-candidate nodes which are not "important" to the subtree structure are removed. This leaves a tree which contains only the ontological relationship between the m candidate classes, and no others. In general, this is no longer a complete tree, although the average depth of a node is much lower than before. All candidate classes have maximal depth in the original ontology, but in the extracted tree, they can be direct children of the root "Thing" class.

Once the candidate subtree has been extracted, the conversational stage proceeds as follows: Starting at the root, the user is presented, as options, all the children of the root. If the user selects an option which is a leaf class, the classification is complete and the item can be stored in the case-base with the chosen class. If the chosen option is not a leaf class, the user is then presented the children of the chosen class as options. This proceeds until either a classification is made, or the user chooses an option specifying that none of the presented options are correct.

In this case that a classification cannot be made based on the top k results, the general approach is to resort to manual classification. The idea of the conversational approach utilising all k of the top k results (rather than the most common or top one) significantly reduces how often this occurs. However, if the user makes some choices before they determine that none of the sub-classes presented are correct, this means that the top k results contained items which, while they were not in the correct class, were somewhat similar to the query in the fact that they belong to the same more general classes. This information can be used to speed up manual classification by only requiring the user to classify

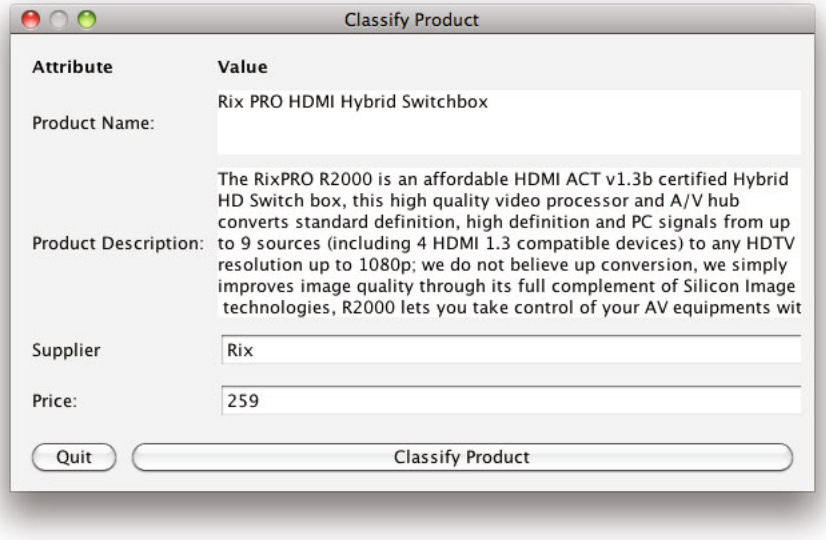

Fig. 4. A sample query for a product to be classified

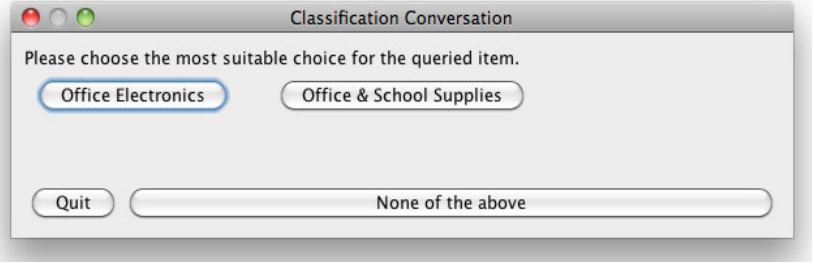

Fig. 5. A view of the conversational stage of classification

within this subtree, aiding the manual classification somewhat. The benefit of this is, unfortunately, a little harder to quantify as it needs to be measured in time taken by a user—something harder to approximate from the stats alone.

In Figs. 4 and 5 some stages of the classification procedure for the developed classifier are shown. In Fig. 4, the query can be seen for the product to be classified—in this case a switchbox, an item of office hardware. Once the kNN classifier determines the top k classes, the conversational stage begins. Figure 5 shows the question asked at a particular point in the conversational stage for this particular item.

4 Results

As previously discussed, the Amazon classification hierarchy is not ideal and is inconsistent in some instances. Specifically, in some categories the classes are

Table 2. Datasets used for classification

Dataset	Description
DS1	Full dataset, not pre-processed
DS2	Full dataset, pre-processed
DS3	Revised dataset, not pre-processed
DS4	Revised dataset, pre-processed

much more specific than others, and categories with large amounts of overlap (e.g. "Office Lighting" and "Lamps"). The UNSPSC hierarchy is much more consistent and less ambiguous, so to try and eliminate the difficulty this would cause for the classifier a second slightly smaller dataset of 13,000 items with some manual cleaning was also used. Specifically, for testing purposes some over-specific or ambiguous classes were removed, and the generality of the classes remaining match more closely (although still only roughly) that of the UNSPSC hierarchy.

Datasets both with and without these classes removed were tested, and a summary of the datasets used is given in Table 2. We also tested the effect of the pre-processing to remove near-identical items, so a total of four datasets were tested.[10]

Testing was performed using 10-fold cross-validation: the case-base is randomly split into 10 folds, and in turn each fold is used for querying the case-base consisting of the other 9 folds. The complete case base was used in all results, although randomly chosen subsets of each fold were used for preliminary testing and are an necessity if testing is to be performed on larger cases bases since classification of each item took approximately 1 second. Indeed, the classification time is $O(n)$ for kNN retrieval [3], so this will only further increase the testing time for larger case bases.

Four test metrics were used:

"kNN" accuracy. This is the standard metric for kNN classifiers, and records the percentage of queries for which the most common class among the k returned was the correct classification. Since the purpose of the project is to use and assess a conversational approach where all k results are utilised, this is probably not the most accurate metric, but gives a useful idea of the quality of the classifier.

"top-one" accuracy. The percentage of queries for which the top result was the correct classification. This metric is largely included to aid comparison to the previous product classifiers which used this metric.

"top-k" accuracy. This records the percentage of queries for which one of the top k results was the correct classification. This indicates the item would have been successfully classified in the conversational stage by the user, and is a more important metric given the application. A value of $k = 10$ was used in our testing.

[10] All four datasets and the class ontology file are available for download from `http://www.cs.auckland.ac.nz/~aabb009/resources/AmazonProductData.zip`. We invite others to test their classifiers and try and improve upon our results.

"average-depth". This is the average depth of the conversation tree. In other words, it records the average number of questions the user would have to be asked to successfully classify the product based on the returned results. If none of the top k results are the correct classification, the maximum possible depth is recorded to represent unsuccessful completion of the conversation stage.

Note that the *average-depth* metric needs to be interpreted with the relevant value of k in mind. Since the maximum depth is equal to the number of levels in the schema (3 for the Amazon schema used, 4 for the UNSPSC schema), even if the whole ontology was used for the conversational stage (as opposed to the extracted subtree), the depth would still be very small. However, the number of options (answers for the questions) presented to the user would by overwhelming, and would not be practical. Thus, the *average-depth* scores need to be interpreted with it in mind that no more that k (10 in our case) options will ever be presented to the user, and usually much less than this.

The results of these tests are presented in Table 3, and are discussed in detail in the following section.

Table 3. Results of the evaluation of the classifier

Dataset	*kNN*	*top-one*	*top-k*	*average-depth*
DS1	80%	82%	95%	1.31
DS2	68%	69%	92%	1.51
DS3	77%	77%	94%	1.38
DS4	71%	70%	94%	1.46

5 Discussion

As can be seen from the evaluation, the *top-one* metric shows that top result returned is the correct classification around 70% of the time on the dataset DS4, which we consider to be the dataset for which the results will best reflect the real world accuracy. This is less than the *top-one* accuracy obtained by the Naïve classifiers GoldenBullet [5] and the one described in [8] by around 15%. However, since the datasets are different it is rather difficult, if not impossible, to compare results meaningfully. It is likely that the fact that items can reasonably belong to more than one class in the Amazon hierarchy, as well as the similarity and lack of consistency between class definitions limits the *top-one* accuracy from being better than we obtained, and a more structured classification schema such as the UNSPSC should work better with the classifier.

The results for the *top-k* metric are very promising. In DS4, for 94% of items, one of the top 10 results was a correct classification, indicating that the conversation stage would have resulted in a correct classification. This result is higher than obtained by the GoldenBullet [5] classifier for *top-10* accuracy, although the other Naïve Bayes classifier [8] does not quote any results for *top-10* accuracy. Because of the nature of the intended functionality of this classifier and

the fact that this represents the percentage of instances for which classification is successful in the conversational stage, we believe this is the more informative accuracy measure for our classifier.

The *average-depth* results are fairly stable. These show that even though 10 results were returned by the kNN search, the user only had to answer 1.4 questions on average to classify the item. This is a significant improvement over presenting the user with the top classes and asking them to pick the correct one. This should result not only in a saving of time, but by asking simple questions and having the user look over less options this should reduce the rate of human error in classification, resulting in a more accurate product catalogue.

As we can see, the *top-10* accuracy of the classifier increases slightly for DS4 compared to DS2. This is consistent with our prediction that the accuracy of the classifier is negatively affected by the inconsistencies within the Amazon class system, and while there will always be ambiguities and difficult classifications in such a dataset—this is part of the problem the classifier should solve—but it is unlikely to be made worse on the UNSPSC dataset, and if anything we would expect the more logical structure of the hierarchy to be beneficial.

Also note that the accuracy decreases when pre-processing is applied. This is not a negative effect, but rather the "false" positives of nearly identical items being classified correctly are removed, meaning the accuracy metric reflects the actual accuracy of the system in a more realistic manner. This effects the *top-one* accuracy much more than the *top-10* accuracy, since the top result should generally remained unchanged after pre-processing (recall one of the multiple similar cases are retained), whereas the top ten results cover a wider range of classes rather than being populated with cases similar to the query.

An interesting point is the suitability of various values of k for use in the classifier. We chose $k = 10$ because it seemed to be a good balance between accuracy and number of questions to answer. The value of k used will impact the *top-k* accuracy and thus the average number of questions that must be answered by the user, so should be considered carefully in a commercial application. A larger value of k will result in slightly more user interaction (on average) for each classification, but would increase the accuracy and thus decrease the instances in which costly complete manual classification is required. Thus, the value of k should be chosen to balance the time required to classify an item from scratch and the time spent on each individual item to be classified. This will depend on the size and nature of the dataset, and classification schema used.

Taking the results and relevant considerations into account, we believe the classifier is a successful proof-of-concept, and shows much promise to be used in a commercial setting. The primary issue in a company such as Unimarket wishing to adopt such a classifier is that it already requires their data to be classified into a useful classification schema, at least in part. If this is the case, then the classifier described can aid in significantly reducing the cost of classifying new items and maintaining the quality of the electronic product catalogue for efficient business practices.

Acknowledgements. We thank Unimarket for providing us with data and helping fund this project, as well as S. Datt and the anonymous referees for comments which helped improve this paper.

References

1. Aamodt, A., Plaza, E.: Case-based reasoning: Foundational issues, methodological variations, and system approaches. Artificial Intelligence Communications 7(1), 39–59 (1994)
2. Abels, S., Hahn, A.: Reclassification of electronic product catalogs: The "apricot" approach and its evaluation results. Informing Science Journal 9, 31–47 (2006)
3. De Mántaras, R.L., McSherry, D., Bridge, D., Leake, D., Smyth, B., Craw, S., Faltings, B., Maher, M.L., Cox, M.T., Forbus, K., Keane, M., Aamodt, A., Watson, I.: Retrieval, resuse, revision and retention in case-based reasoning. The Knowledge Engineering Review 20(3), 215–240 (2005)
4. Díaz-Agudo, B., González-Calero, P.A., Recio-García, J.A., Sánchez-Ruiz-Granados, A.A.: Building CBR systems with jCOLIBRI. Science of Computer Programming 69, 68–75 (2007)
5. Ding, Y., Korotkiy, M., Omelayenko, B., Kartseva, V., Zykov, V., Klein, M., Schulten, E., Fensel, D.: GoldenBullet in a nutshell. In: Proceedings of FLAIRS 2002 (2002)
6. Grabowski, H., Lossack, R., Weißkopf, J.: Datenmanagement in der Produktentwicklung (Data management in product development). Hanser-Verlag (2002)
7. Hatcher, E., Gospodnetić, O., McCandless, M.: Lucene in Action, 2nd edn. Action Series. Manning Publications Co. (2004)
8. Kim, Y., Lee, T., Chun, J., Lee, S.: Modified Naïve Bayes classifier for e-catalog classification. In: Lee, J., Shim, J., Lee, S.-g., Bussler, C., Shim, S. (eds.) DEECS 2006. LNCS, vol. 4055, pp. 246–257. Springer, Heidelberg (2006)
9. McGinty, L., Smyth, B.: Improving the performance of recommender systems that use critquing. In: Mobasher, B., Anand, S.S. (eds.) ITWP 2003. LNCS (LNAI), vol. 3169, pp. 114–132. Springer, Heidelberg (2005)
10. Recio-García, J.A., Sánchez-Ruiz, A.A., Díaz-Agudo, B., González-Calero, P.A.: jCOLIBRI 1.0 in a nutshell: A software tool for designing CBR systems. In: Petridis, M. (ed.) Proccedings of the 10th UK Workshop on Case Based Reasoning, pp. 20–28. CMS Press, University of Greenwich (2005)
11. Recio-García, J.A., Díaz-Agudo, B., González-Calero, P.A.: Textual CBR in jCOLIBRI: From retrieval to reuse. In: Wilson, D.C., Khemani, D. (eds.) Proceedings of the ICCBR 2007 Workshop on Textual Case-Based Reasoning: Beyond Retrieval, pp. 217–226 (August 2007)
12. Recio-García, J.A., Díaz-Agudo, B., González-Calero, P.A., Sánchez-Ruiz, A.A.: Ontology based CBR with jCOLIBRI. In: Ellis, R., Allen, T., Tuson, A. (eds.) Proceedings of AI 2006, The Twenty-sixth SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence, Applications and Innovations in Intelligent Systems XIV, pp. 149–162. Springer, Cambridge (2006)
13. Smyth, B., McKenna, E.: Footprint-based retrieval. In: Althoff, K.D., Bergmann, R., Branting, L.K. (eds.) ICCBR 1999. LNCS (LNAI), vol. 1650, pp. 343–357. Springer, Heidelberg (1999)

14. UNSPSC: Better supply managment with UNSPSC. Electronic, http://www.unspsc.org/AdminFolder/Documents/adopting-unspsc.pdf (retrieved January 14, 2011)
15. Weber, R.O., Ashley, K.D., Brüninghaus, S.: Textual case-based reasoning. The Knowledge Engineering Review 20(3), 255–260 (2005)
16. Wettschereck, D., Aha, D.W., Mohri, T.: A review and empirical evaluation of feature weighting methods for a class of lazy learning algorithms. Artificial Intelligence Review 11, 273–314 (1997)