Journal of Web Engineering, Vol. 0, No. 0 (2017) 000–000
© Rinton Press

A FRAMEWORK FOR PRODUCT DESCRIPTION CLASSIFICATION IN E-COMMERCE

DAMIR VANDIC, FLAVIUS FRASINCAR
Econometric Institute, Erasmus University Rotterdam
P.O. Box 1738, 3000 DR Rotterdam, the Netherlands
{vandic, frasincar}@ese.eur.nl

UZAY KAYMAK
Department of Industrial Engineering & Innovation Sciences, Eindhoven University of Technology
P.O. Box 513, 5600 MB Eindhoven, the Netherlands
u.kaymak@ieee.org

Received (received date)
Revised (revised date)

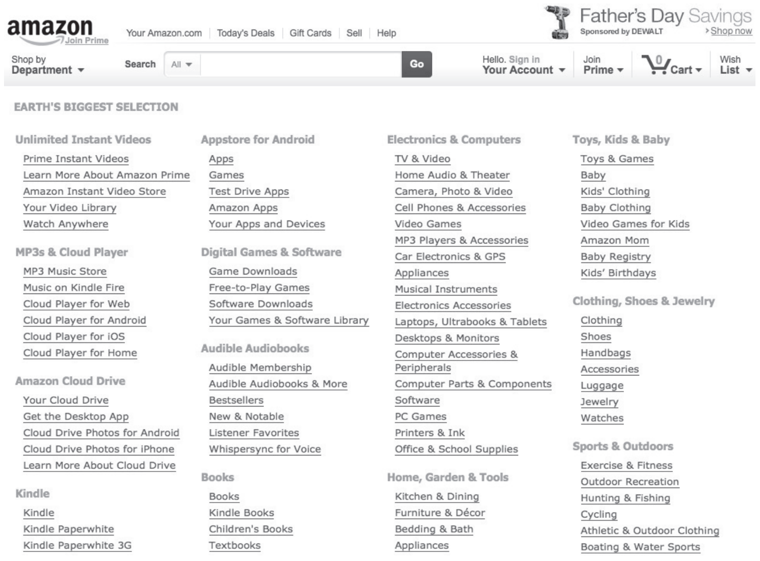
We propose the Hierarchical Product Classification (HPC) framework for the purpose of classifying products using a hierarchical product taxonomy. The framework uses a classification system with multiple classification nodes, each residing on a different level of the taxonomy. The innovative part of the framework stems from the definition of classification recipes that can be used to construct high-quality classifier nodes, using the product descriptions in the most optimal way. These classifier recipes are specifically tailored for the e-commerce domain. The use of these classifier recipes enables flexible classifiers that adjust to the taxonomy depth-specific characteristics of product taxonomies. Furthermore, in order to gain insight into which components are required to perform high quality product classification, we evaluate several feature selection methods and classification techniques in the context of our framework. Based on 3000 product descriptions obtained from Amazon.com, HPC achieves an overall accuracy of 76.80% for product classification. Using 110 categories from CircuitCity.com and Amazon.com, we obtain a precision of 93.61% for mapping the categories to the taxonomy of shopping.com.

Keywords: Product descriptions, hierarchical clustering, feature selection, e-commerce
Communicated by: to be filled by the Editorial

1 Introduction

The World Wide Web (WWW) has drastically changed the availability and exchange of information. Nowadays, consumers and businesses more often make use of e-commerce [25]. Most studies from literature focus on approaches that personalize the experience [14, 16, 40] and enhance the purchase decisions [42, 53] of users visiting Web shops. Product taxonomies are related to these research fields and have been widely used for the organization of many kinds of information on the Web. Product taxonomies help customers find relevant products and allow businesses to organize the offered product assortments. Figure 1 shows an example of the product taxonomy of Amazon.com, where the consumer has a good overview of the products that are offered.

Usually, there is no uniform description of the same products among different vendors on the Web, which forces business-to-business e-commerce to address the issues that arise



EARTH'S BIGGEST SELECTION

Unlimited Instant Videos
- Prime Instant Videos
- Learn More About Amazon Prime
- Amazon Instant Video Store
- Your Video Library
- Watch Anywhere

MP3s & Cloud Player
- MP3 Music Store
- Music on Kindle Fire
- Cloud Player for Web
- Cloud Player for Android
- Cloud Player for iOS
- Cloud Player for Home

Amazon Cloud Drive
- Your Cloud Drive
- Get the Desktop App
- Cloud Drive Photos for Android
- Cloud Drive Photos for iPhone
- Learn More About Cloud Drive

Kindle
- Kindle
- Kindle Paperwhite
- Kindle Paperwhite 3G

Appstore for Android
- Apps
- Games
- Test Drive Apps
- Amazon Apps
- Your Apps and Devices

Digital Games & Software
- Game Downloads
- Free-to-Play Games
- Software Downloads
- Your Games & Software Library

Audible Audiobooks
- Audible Membership
- Audible Audiobooks & More
- Bestsellers
- New & Notable
- Listener Favorites
- Whispersync for Voice

Books
- Books
- Kindle Books
- Children's Books
- Textbooks

Electronics & Computers
- TV & Video
- Home Audio & Theater
- Camera, Photo & Video
- Cell Phones & Accessories
- Video Games
- MP3 Players & Accessories
- Car Electronics & GPS
- Appliances
- Musical Instruments
- Electronics Accessories
- Laptops, Ultrabooks & Tablets
- Desktops & Monitors
- Computer Accessories & Peripherals
- Computer Parts & Components
- Software
- PC Games
- Printers & Ink
- Office & School Supplies

Home, Garden & Tools
- Kitchen & Dining
- Furniture & Décor
- Bedding & Bath
- Appliances

Toys, Kids & Baby
- Toys & Games
- Baby
- Kids' Clothing
- Baby Clothing
- Video Games for Kids
- Amazon Mom
- Baby Registry
- Kids' Birthdays

Clothing, Shoes & Jewelry
- Clothing
- Shoes
- Handbags
- Accessories
- Luggage
- Jewelry
- Watches

Sports & Outdoors
- Exercise & Fitness
- Outdoor Recreation
- Hunting & Fishing
- Cycling
- Athletic & Outdoor Clothing
- Boating & Water Sports

Fig. 1. Product categories from Amazon.com.

with heterogeneous information [27, 26], such as synonyms and homonyms. Similarly, the automatic classification of products is becoming more important as this enables companies to lower costs by spending less time on this task. Without automatic classification, one has to manually classify products and the cost of this process will keep increasing as the heterogeneous information on the Web keeps growing.

In this work, we investigate text classification techniques for the purpose of providing effective hierarchical product classification. We loosely define product classification as the task of 'assigning a product to an existing or new category, given a product description'. Hierarchical classification can be considered as a classification that takes the hierarchical structure of the taxonomy into account. In this work, the classification is determined to be static, i.e., we assume that the classification of products will not change over time.

Product descriptions on the Web usually contain information like the title, brand, features description, and (optionally) reviews of the product. If the product description is extracted from an existing system, it can also contain the category it was assigned in that system. On the Web, product descriptions are often not structured and not classified, i.e., the product category is missing or does not belong to a standard taxonomy [9]. There are various taxonomies that can be used for the purpose of product classification, including the United Nations Standard Products and Services Code (UNSPSC) standard [39].

The main focus of this paper is on the classification of products into an existing product taxonomy. The product taxonomy refers to a predefined hierarchy of product categories. The goal of this research is to evaluate text classification techniques for the purpose of effective product classification, and to provide a framework that deals with various issues encountered in practice that impede this process. For this system, we can identify three main requirements, (1) the classification of products to both internal and leaf nodes in the category hierarchy, thereby supporting classification to multiple nodes (multiple classification), (2) dealing with

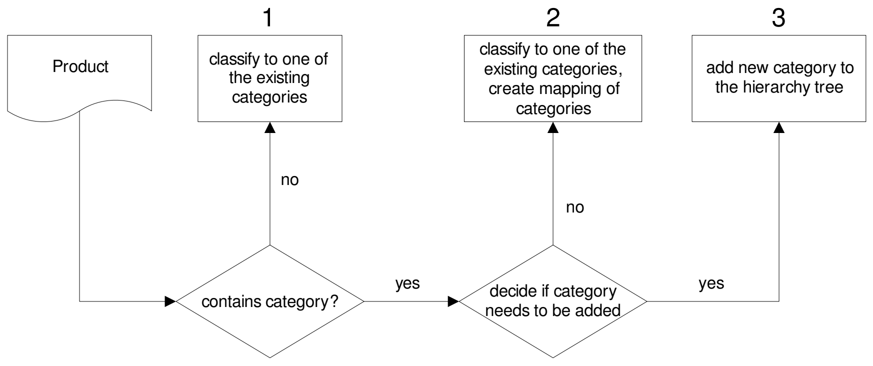

1
2
3
Product
classify to one of the existing categories
classify to one of the existing categories, create mapping of categories
add new category to the hierarchy tree
no
no
contains category?
yes
decide if category needs to be added
yes

Fig. 2. Three possible scenarios when a new product needs to be classified.

product descriptions that may contain a category, and (3) providing a decision algorithm to identify the cases where no matching category exists.

The proposed Hierarchical Product Classification (HPC) framework requires a given category hierarchy, without existing product description associations. Given the above requirements, we can identify three scenarios that can occur when a new product has to be classified, as shown in Figure 2. A product description can contain a category that does not necessarily have to be present in the existing category taxonomy. If the product description contains a category, the system must be able to determine whether or not the product should be classified to an existing category in the hierarchy (scenario 2), or that it should be classified using the given category by adding it to the hierarchy (scenario 3), i.e., when there is no match. The classification algorithm should use, if available, the new given category in this decision, as it can provide valuable information concerning the classification of the product. If the product description does not contain a category, the task is to classify the product to one of the existing categories (scenario 1). The main focus of this paper is on scenarios 1 and 2. However, the proposed category mapping algorithm for scenario 2 can be used with a decision function to determine whether the product description 'fits' one of the categories present in the system or whether the computed mapping is false, i.e., whether it is necessary to perform the steps for scenario 3.

The contributions of this paper are threefold. First, we evaluate several classification techniques on large, real-world product description data sets, which has not been done before. Second, we propose a high precision algorithm that makes use of syntactic and semantic similarities in order to map a given product category to an existing taxonomy of product categories. Third, and last, this work gives a clear overview of which feature selection methods in product descriptions provide the most accurate classifications.

We start with a survey of the current literature on related classification techniques, feature selection methods, and evaluation techniques. We discuss these topics in Section 2. In Section 3 we present the details of the proposed framework. We provide an overview of the evaluation results in Section 4, where we assess the framework with real-world data from Amazon.com. Finally, in Section 5, we summarize our findings and give directions for future research.

2 Related Work

According to [51], automated text classification (TC) is a learning task, defined as assigning predefined category labels to new documents based on the likelihood suggested by a training set of labeled documents. Classification techniques can be categorized by two aspects. First, the difference between *flat* and *hierarchical* classifiers is that flat classifiers assign documents to categories at one level, i.e., there is no category hierarchy, as opposed to hierarchical classifiers, where the hierarchy of categories must be taken into account by the classifier. Second, a classifier can be an *independent binary* classifier or *m-ary* ($m > 2$) classifier. Given a document, an independent binary classifier makes a yes/no decision for each category, while an *m-ary* classifier typically consists of multiple classifiers (e.g., one for each category) and computes a ranked list of candidate categories for each document.

In the literature, we can find two main approaches for hierarchical text classification, i.e., the *big-bang* approach and the *top-down level-based* approach [36]. These two approaches are not tied to a specific classification technique because they only prescribe how one or more text classifiers should be used. In the *big-bang* approach, only a single classifier is used in the classification process. Given a document, the classifier assigns it to one or more categories in the category taxonomy. In the *top-down level-based* approach, one or more classifiers are constructed at each level of the category taxonomy and each classifier works as a flat classifier at its level. A document will first be classified by the classifier at the root level. It will then be further classified into one or more lower categories by their corresponding classifiers. This process continues until it reaches a final category that could be a leaf category or an internal category. Different types of classification techniques have been developed that can be used with both approaches. These include rule-based techniques [33, 32, 43], probabilistic approaches [15, 3, 22, 38], fuzzy [44, 20], support vector machine approaches [7, 36, 54, 11, 28], neural networks [21, 45, 30], and cluster-based techniques [19, 35].

A major issue for all text classification techniques is the high dimensionality of the feature space. Several feature selection methods exist that can be used in combination with a threshold to achieve a desired degree of term elimination. *Term frequency thresholding* (TF) is the simplest technique for vocabulary reduction. The frequency of each term is computed and a minimum threshold for this frequency is used to remove terms. *Information gain* (IG) is another feature selection method that is used frequently in the field of machine learning [24]. IG measures the number of bits of information obtained for category prediction by knowing the presence or absence of a term in a document. *Mutual information* (MI) is a criterion commonly used in statistical language modeling of words associations [4]. This criterion has by convention value zero when there is no document that contains the considered word pair. To use this criterion for feature selection, one can compute the *average* or *maximum* mutual information value for a term. Another popular statistical criterion is the χ^2 *statistic* (CHI). The χ^2 *statistic* measures the lack of independence between two words and can be compared to the χ^2 distribution with one degree of freedom. Like the MI, the CHI statistic has a value of zero when the two words are independent. This statistic can be used for feature selection in the same manner as MI (computing an average or maximum). The difference between CHI and MI is that CHI is a normalized value, and hence CHI values are comparable.

The authors of [52] have performed an empirical study, comparing different feature selection methods, including the TF, IG, and CHI methods. The authors used two m-ary classifiers, a k-Nearest Neighbor classifier (kNN) [47], and a regression based method named Linear Least Squares Fit mapping (LLSF) [50]. They consider recall and precision as performance measures. The authors conclude that IG and CHI provide the most effective aggressive term removal (up to 90% of the original feature space) without losing classification accuracy.

As previously discussed, there are roughly two approaches to classification, i.e., the top-down and big-bang approaches. Because top-down approaches use multiple classifiers, they address the issue of separating the noisy terms from the useful ones. This task is usually highly dependent on the location in the category hierarchy. For example, 'mobile phone' may be a good feature for a top level classification (e.g., *Electronics*), but becomes useless when drilled down to *Electronics/Mobile Phones*.

In [3], a typical top-down approach is proposed, where a Bernoulli model is assumed for the document generation. A method based on Fisher's discriminant indices is used for feature selection, which takes place at each category node. The authors compared their approach with a weighted one-level cosine classifier. Their approach showed better results with respect to the micro-averaged recall [49], i.e., 0.66 versus the 0.48 result of the cosine classifier.

[6] propose a system that classifies products using existing classification standards, such as UNSPSC [39]. The authors consider three methods and the main focus of the system is the business-to-business environment. For non-hierarchical classification the best result comes from the Naïve Bayes Classifier, i.e., 78%, outperforming the Vector Space Model (VSM) [31] and the kNN algorithm. We hypothesize that this performance can be increased by employing a top-down classification system where feature selection is performed on each node level, separately. For hierarchical classification the highest accuracy obtained is 38%.

[5] propose an approach where documents are classified only to leaf nodes of the category hierarchy. Classification is done by taking the weighted sum of feature occurrences that should be larger than the category threshold. The innovative contribution of this approach is the possibility of restructuring an initial hierarchy or building a new one from scratch, topics that are outside the scope of our approach.

[43] identify several issues with the *top-down level-based* approaches. Among other aspects, the *closeness of classification* is not addressed by these approaches, e.g., classifying a mobile phone, which belongs to the category 'Mobile Communications' as 'Electronics' is a smaller error compared to classifying it as 'Clothes'. For this reason, there are a number of approaches proposed in the literature that are designed using the Big-Bang approach. [45] propose a two-level classification, where their approach is characterized by a probabilistic framework. [30] present the design and evaluation of an approach based on the Hierarchical Mixture of Experts model. As our solution, this approach also uses a divide-and-conquer strategy to define smaller categorization problems based on a predefined hierarchical structure. With respect to accuracy, the approach of [30] shows better results compared to [48] and [18], where a nearest neighbor classifier and a linear classifier are used, respectively.

[37] proposes *Chimera*, an approach for classifying product descriptions that combines learning, rules (created by employees), and crowdsourcing. The authors argue that using rules (in conjunction with learning) is valuable and that research should focus more on helping analysts create and manage these more effectively. Although this approach provides interesting results, it is difficult to compare it with our approach. First, the system relies on significant human effort. For example, the system uses a manually curated list of 20,000 brands in the

classification step. Another example is the use of rules and crowdsourcing in the system. This makes it very difficult to compare this approach with ours, which is fully automatic. Second, the focus of the classification task seems to differ from ours. Whereas we propose a system for hierarchical product classification, i.e., using a deep multi-level taxonomy, the Chimera system focuses more on a large scale, flat, taxonomy, consisting of only two levels. The different scope makes a direct comparison with our solution unsuitable.

We can draw several conclusions from the literature overview. First, besides [6], none of the related work that aims to solve the same task as our approach focuses on specifically classifying product descriptions. The work in [6] has some significant limitations, as it compares only three methods (VSM, k-Nearest Neighbor, and Naïve Bayes), and more importantly, the results for hierarchical classification are not promising as the highest accuracy that is obtained is 38%. Second, there is no literature on feature selection for product descriptions. It is not clear which parts of a product description can be used for hierarchical classification of products. The paper aims to fill these gaps by thoroughly evaluating the effects of using the different parts of a product description in combination with well-known feature selection methods.

3 The HPC Framework

In this section we present the Hierarchical Product Classification (HPC) framework for classifying product descriptions using a hierarchical product category taxonomy. In the next sections the different components of the HPC framework are discussed in detail. The preparation of the data set is discussed in Section 3.1. In Section 3.2, we discuss the HPC classification system.

3.1 Data set processing

The preparation of the data set is part of the HPC framework, as product descriptions are usually very heterogeneous, especially with respect to the level of detail. For this reason, the HPC framework assumes that a product description has at least the following required parts: (1) product title (text), (2) brand of the product (nominal), (3) price of the product (number), and (4) description of the features of the product (text). To avoid the ambiguity of the term *product description*, we will introduce the term *features description* for description of the features of a product and *product description* will refer to the collection of all four parts (title, brand, price, and features description).

More formally, we define the vocabulary of unique words of all alphabetic parts (i.e., title and features description) of a product description as the vector $\mathbf{w} = (w_1, w_2, \ldots, w_n)$. A product description d_i is then represented as

$$\left(\mathbf{x}^i_{title}, \mathbf{x}^i_{desc}, p^i, b^i\right) \in \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R} \times \mathbb{B} \tag{1}$$

where $\mathbf{x}^i_{title}$ and $\mathbf{x}^i_{desc}$ represent the title and features description of product description i, respectively. These vectors contain the counts for each word from the vocabulary. The term p^i represents the price and $b^i \in \mathbb{B}$ represents the brand, where $\mathbb{R}$ is the set of real numbers and $\mathbb{B}$ is the set of all known product brands. This definition is necessary as the HPC framework addresses each part of a product description differently. Furthermore, the set $C = \{c_1, c_2, \ldots, c_n\}$ represents all known product categories, with a total of n categories. The hierarchy is then represented as

$$H = \{(c_a, c_b) \,|c_a, c_b \in C \wedge c_a \leq c_b\} \tag{2}$$

where $\leq$ denotes the subsumption relationship. A set of product descriptions is denoted by $D = \{d_1, d_2, \ldots, d_m\}$, where $d_i \in D$ represents a product description i, i.e., $d_i = \left(\mathbf{x}^i_{title}, \mathbf{x}^i_{desc}, p^i, b^i\right)$. Also, we let the vector $\mathbf{y}$ denote the category mappings of the product descriptions. Consequently, $\mathbf{y}$ contains m values. We assume here that a product belongs only to one category (the most specific one).

In the data preparation process, there are the two main steps that are performed on the content of the product descriptions. First, all stop words are removed from the title and features description. The HPC framework does not define a stop word list, this has to be specified by the user. This enables the user of the system to perform fine adjustments to decide which words are considered stop words and which are not. The removal of stop words eliminates the noise stop words introduce. The accuracy of a classification algorithm often increases after the removal of stop words. Even though in our evaluations we have used a standardized stop word list, a more automated approach could be employed, such as the one proposed in [46].

After the stop words are removed, the remaining words of the product title and features description are stemmed. Many word stemming algorithms exist and the HPC framework does not restrict the usage of any particular stemming algorithm. The default stemming algorithm is the Porter stemming algorithm [29]. After the stemming process has completed, we have a set of product descriptions that are prepared for the classification system processes.

3.2 Classification system

The *classification system*, the core of the HPC framework, is used to classify product descriptions and it consists of a hierarchy of *classifiers nodes* (a hierarchy similar to the product taxonomy nodes, but without the product taxonomy leaves). A classifier node is a collection of classifiers that are trained on different parts of the product description. The classification system is based on the top-down approach. The reason for choosing the top-down approach is that it can select different features depending on the classifier location in the taxonomy. As mentioned earlier, features 'mobile' and 'phone' may be appropriate for a decision between *Electronics*, *Home & Garden*, and *Sports*, but become less useful when the classifier has to decide between the children of the category *Electronics/Mobile Phones*.

We propose the so-called K-level top-down approach, where $K > 1$. The parameter K is the highest level of the product taxonomy where classifier nodes will be placed. If $K = 2$, then the classification takes place on the first and second level of the taxonomy (i.e., levels 0 and 1). Figure 3 shows an example of a classification system with $K = 2$. We can see that the first classifier node decides between the categories 'Electronics' and 'Sports' (level 0). If 'Electronics' is chosen by the first classifier node, then the second classifier node has to classify to either 'Home', 'Communication', 'Knives', 'Mobile Phones', or 'Monitors'. In this case, this is the last classifier and therefore it classifies to the leaves of the sub-taxonomy. If 'Sports' was chosen, then another classifier (also on level 1) had to decide between 'Jackets' and 'Shoes'. In this case the leaves are also the children of the node 'Sports'.

In the HPC framework, classifier nodes are constructed by using *classifier recipes*. A classifier recipe contains the necessary information to construct a classifier node. It defines which classification techniques and feature selection methods are used for what parts of the product description. It is important to note that each level in the category hierarchy can have its own classifier recipe. Consequently, classifier nodes can differ from level to level in the category hierarchy. Figure 4 shows the structure of a classifier recipe. A classifier recipe consists of four components. The first two components each define a feature selector and a text classifier, which are used for the title and the features description. The third component is a classification algorithm that operates on the brand and price. The fourth component is a specialized algorithm that is used in the case that a category is present in the product description. In this paper, we propose and evaluate such an algorithm. For the brand and price, one can choose any classifier that takes as input one numerical and one categorical variable.

3.2.1 Constructing classifier nodes

For each node, a classifier recipe is used. The classifier node encompasses four classifiers that use different parts of the product description (i.e., (1) title, (2) description, (3) brand and price, and (4), [optional] category). In order to construct a classifier node, one needs to have a classifier recipe, a training set

$$D \subset \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R} \times \mathbb{B}, \tag{3}$$

and a target vector $\mathbf{y}$ where the values are taken from the set of categories C. A product description d_i, as discussed in Section 3.1, is represented as

$$\left(\mathbf{x}^i_{title}, \mathbf{x}^i_{desc}, p^i, b^i\right) \in D, \tag{4}$$

For both the title and the features description, a classifier recipe defines the feature selector and text classifier (first classifier and second classifier, respectively). A feature selector selects

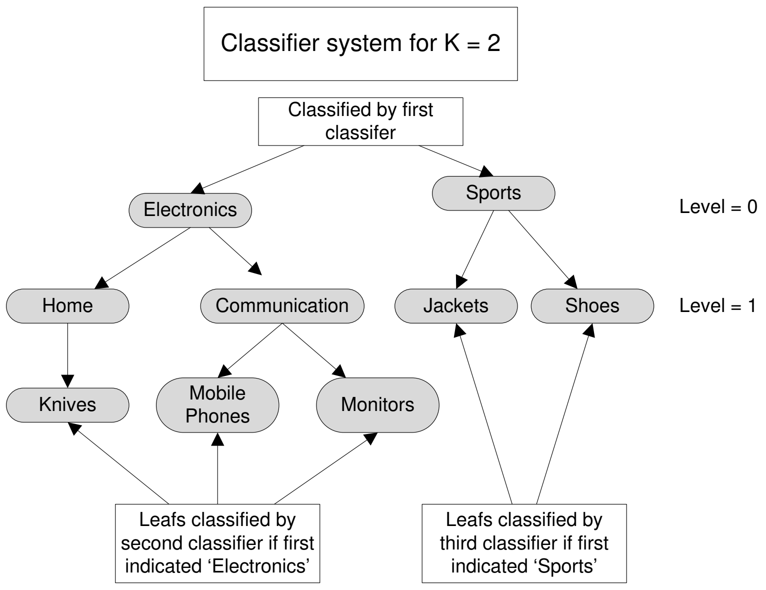

Classifier system for K = 2
Classified by first classifer
Electronics
Sports
Level = 0
Home
Communication
Jackets
Shoes
Level = 1
Knives
Mobile Phones
Monitors
Leafs classified by second classifier if first indicated 'Electronics'
Leafs classified by third classifier if first indicated 'Sports'

Fig. 3. K level top-down approach for $K = 2$.

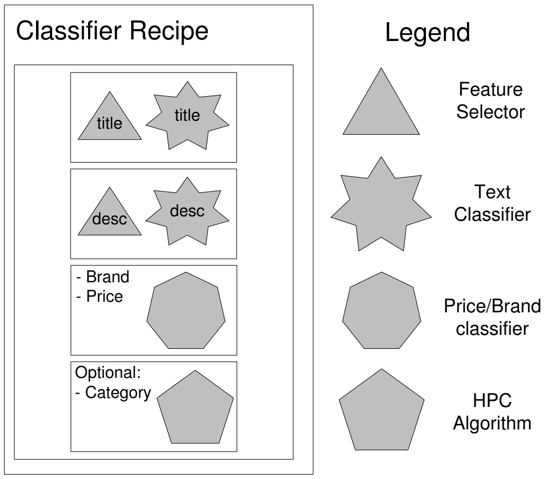


Fig. 4. The structure of a classifier recipe.

relevant features, given a feature matrix $\mathbf{X}$ (where each column represents one feature) and a target vector $\mathbf{y}$. The text classifier must be a function that takes as input a product description vector $\mathbf{x} \in \mathbb{R}^n$ and outputs one of the categories, predefined by the set C (see Section 3.1). In order to construct the classifiers for the title and description in the classifier recipe, the feature selector is first applied to the training sets $\mathbf{x}_{title}$ and $\mathbf{x}_{desc}$. Next, the classifier is trained on the training set through cross validation, to obtain reliable results and to prevent the classifier to overfit the data. The 'best' classifier is chosen, i.e., the one with the highest precision.

The third part of the classifier recipe defines the usage of the brand and the price of the product description for the purpose of classification. This classifier is trained using cross validation on the price and brand training data, and the 'best' classifier is chosen, i.e., the one with the highest precision. A classifier recipe defines a threshold β. If the precision of the best classifier, when considering only instances of the brand provided in a product description, is below β, then this classifier is not used because it is unreliable. This condition is determined at runtime. The recipe also defines a threshold parameter δ, this threshold represents the minimum number of instances in the training data set in order to use this classifier. The δ threshold ensures that this classifier is used only when there is enough data to make a reliable decision for the brand/price combination.

Product categories can have different names across systems but also different hierarchies can be used. For instance, 'Nintendo DS Games' can be a child of 'Games', where on another system it is a child of a more specific category 'Console Games'. The fourth and last part of a classifier recipe defines the classifier of the given category in a product description. It requires an algorithm that takes a string input (the given category) and outputs a list of possible matches, along with the corresponding scores (similar to an *m-ary* classifier). The score should be between 0 and 1 and the category with the highest score is the one which matches the best. The HPC framework defines a custom algorithm for this purpose. When there is no category given in the product description, this classifier is not used.

In order to meet the above requirements, we propose the Category Mapping algorithm, which is also employed in [41]. The goal of the Category Mapping algorithm is to identify to which existing product category the given product category should be mapped. There are two difficulties with this process. First, one has to deal with syntactic variations and with semantic variations. The syntactic variations are for example singular/plural forms, abbreviations, and typographical mistakes. The semantic variations are synonyms and homonyms. In order to deal with these issues we developed an algorithm which is able to determine the correct category for a product with high precision. Before we give the details of the algorithm, we need to explain existing text similarity measures and other similarity functions that are used in the algorithm.

The Levenshtein distance [17] is a metric for measuring the amount of difference between two strings (i.e., the so-called edit distance). The Levenshtein distance between two strings is given by the minimum number of operations needed to transform one string into the other, where an operation is an insertion, deletion, or substitution of a single character. We denote it by alv_{ij}, which is the absolute Levenshtein distance between strings i and j. The HPC framework uses the *normalized* Levenshtein distance, which is a function of the absolute Levenshtein distance. We use the notation lv_{ij}, which is the normalized Levenshtein distance between strings i and j. The normalized Levenshtein distance is defined as

$$\mathrm{lv}\left(i,j\right) = \frac{\mathrm{alv}(i,j)}{\max(\mathrm{length}(i),\mathrm{length}(j))} \tag{5}$$

The normalized Levenshtein distance addresses the issue of short string lengths. If you have two strings, of both length 24, then an absolute Levenshtein distance of 3 is not large. However, with two strings of length 6 this distance is quite large as it is 50% of the tag length. According to the absolute Levenshtein distance these two distances are the same. But the normalized Levenshtein distances are in this case 0.125 and 0.5. This indicates that, according to the normalized Levenshtein distance, the two pairs of strings do not have the same distance, i.e., the first pair is more similar.

The function $\mathrm{calcCosineSim}\left(A,B\right)$ is used to compute the cosine similarity between two sets of words A and B, and it is defined as follows:

$$\mathrm{calcCosineSim}\left(A,B\right) = \frac{|A \cap B|}{\sqrt{|A|}\sqrt{|B|}} \tag{6}$$

With $\mathrm{avgLvSim}\left(A,B\right)$, the average Levenshtein similarity between two sets of words can be computed. Using the normalized Levenshtein distance function $\mathrm{lv}\left(i,j\right)$ for words i and j, we can give the definition of the function $\mathrm{avgLvSim}\left(A,B\right)$, where A and B are sets of words, as following:

$$\mathrm{avgLvSim}\left(A,B\right) = \sum_{a\in A}\sum_{b\in B}\left(1-\mathrm{lv}\left(a,b\right)\right)\frac{\mathrm{length}\left(a\right)+\mathrm{length}\left(b\right)}{\sum\limits_{a\in A}\sum\limits_{b\in B}\mathrm{length}\left(a\right)+\mathrm{length}\left(b\right)} \tag{7}$$

Algorithm 1 shows the steps taken to find a matching product category, given a new category name. It requires to have an existing set of categories C. The algorithm also requires to have the set Y of synonyms/syntactic variations of the provided category name. For this purpose, we use WordNet [8] to gather the category synonyms. The process starts by combining the category name, which needs to be mapped to an existing category, with all syntactic variations and synonyms of that category name, obtained from WordNet, in one set

Z (line 1). After that, the empty set S is created (line 2). In lines 3 through 8, the set S is filled. For each combination between a category from the set Z and a category from the set C, the category names are cleaned. The cleaning of category names is necessary in order to remove any 'noise'. For example, some users write in words 'Camera and Photography' and others might write the abbreviated form 'Camera & Photography'. We solve this issue by replacing occurrences of both 'and' and '&' by a space character. After the category names are cleaned, the similarity between them is computed and added to the set S. The similarity is stored as a pair together with the category from the set C (the set of existing categories). The function that is used to calculate the similarity between two cleaned category names is given by:

$$\text{getCatSim}(A, B) = \lambda \cdot \text{calcCosineSim}(A, B) + (1 - \lambda) \cdot \text{avgLvSim}(A, B) \tag{8}$$

where A and B are sets of words. The function $\text{calcCosineSim}(A, B)$ is defined by Equation 6 and $\text{avgLvSim}(A, B)$ is defined by Equation 7. The sets A and B are obtained by splitting a category name on the space character. This is achieved by using the function $\text{cleanAndSplit}(\cdot)$, which also 'cleans' the category names, i.e., it replaces the word 'and', the word 'or', the character '&', as well as parentheses, comma's, and other special characters, with a space character. When all combinations are processed and the set S is filled, a category needs to be chosen. The category with the highest score in the set S is selected as the matching product category. If multiple categories exist with the highest score, then the average cosine similarity between the feature vectors of each category and the product description is computed, and the category with the highest cosine similarity is chosen. If the highest score is below γ, then this classifier is not used in the process of classification.

Algorithm 1: The category matching algorithm.

Input : The new category c to be matched to an existing category (text).
Output : The best matching category with the corresponding computed similarity.
Data : The set C (set of categories).
The set Y (synonyms of the new category c).

```
1  $Z = Y \cup \{c\}$;
2  $S = \{\}$;
3  // for each category pair from $Z$ and $C$, compute their similarity
4  foreach $z$ in $Z$ do
5      foreach $c'$ in $C$ do
6          $A = \texttt{cleanAndSplit}(z)$;
7          $B = \texttt{cleanAndSplit}(c')$;
8          $S = S \cup \{(c', \texttt{getCatSim}(A, B))\}$;
9      end
10 end
11 return $\{(r, m) \in S | \forall (y, n) \in S : n \leq m\}$
```

Algorithm 2: The HPC system construction process.

```
1  $CF = \{cf_{-1}\}$ // set of classifier nodes with top-level classifier
2  $Q$ = empty queue ;
3  **foreach** $c$ **in** $C_{top}$ **do**
4      enqueue $(Q, (c, 0))$;
5  **end**
6  $i = 0$;
7  **while** notEmpty $(Q)$ **do**
8      $(c, L)$ = dequeue $(Q)$;
9      **if** $L = K - 2$ **then**
10         $cf_i$ = classifier at node $c$ trained on **leaf categories** under $c$, using recipe for
           level $L$;
11     **else**
12         $cf_i$ = classifier at node $c$ trained on **children categories** of $c$, using recipe for
           level $L$;
13         **foreach** $ch$ **in** children$(c)$ **do**
14             enqueue $(Q, (ch, L + 1))$;
15         **end**
16     **end**
17     $CF = CF \cup \{cf_i\}$;
18     $i = i + 1$;
19 **end**
```

3.2.2 Constructing the classification system

In the previous section, we discussed the different parts of a classifier recipe and how we construct one classifier node, given a set of labeled product descriptions. In order to construct a complete HPC classification system, at least one recipe is needed. As we will see in Section 4, it is advisable to use different classifier recipes on each level. In this section we discuss the design and implementation of the complete HPC classification system.

Algorithm 2 shows the basic steps to construct an HPC classification system. The algorithm starts by creating a classifier for level -1, this level is one level above the level where the top-level categories reside (level 0). The first root level classifier always exists, independent of the value of K. The root level classifier is added to the set of classifier nodes. This task is performed in lines 1 through 3. From line 4, the algorithm starts a breadth-first traversal through the category hierarchy, creating classifier nodes where necessary and stopping when it hits a leaf category node or the current level has exceeded $K - 2$. The breadth-first traversal function performs a check for each category, starting with the top-level categories. If the level of the category node is equal to $K - 2$, then a classifier is created which is trained on the leaf category nodes of that category. If this is not the case, then an intermediary classifier node is created and trained on the children of the current category node, its children are also added to the queue to be visited. One should note that whenever a classifier node is created, the corresponding classifier recipe for that level (which the category node resides on) is used.

3.2.3 Classification Algorithm

The classification process starts at the root level classifier node, which classifies the product description to one of the top-level categories. Next, the algorithm continues the classification until the classification results in a category leaf node or the maximum classification depth has been reach (represented by the K parameter). The algorithm chooses the next classifier node based on the previous classification.

In each classifier node, the classification is performed by following a simple voting system. The algorithm classifies a single product description into one of the existing categories from the set C, as defined in Section 3.1. The algorithm starts by asking each component to cast a vote on the target category, i.e., each component performs classification, outputting one product category. The next step is to check if there is a category in the set S which has the highest amount of votes. If there exist such a category, then that is the category which is returned as the best match. In the case that no such category exists, the classifier node flags the product description as unclassifiable. In this case, Scenario 3 would be useful to consider, i.e., there is a need for modifying the existing category hierarchy by adding one or more new categories to the hierarchy. However, this scenario is out of the scope of this paper.

The example shown in Figure 3 highlights these steps for $K = 2$. Because $K = 2$, there can be only 2 classification steps. The first classifier node decides between the categories 'Electronics' and 'Sports' (level 0) and the second classifier node, regardless of the outcome of the first classifier, will classify the product description to one of the leaves. These leaves are 'Knives', 'Mobile Phones', or 'Monitors' in case the first classifier chose 'Home' or 'Communication', and Jackets or Shoes in case the first classifier chose for 'Sports'.

4 Evaluation

In this section, we evaluate the proposed framework and its components. The goal is to find what the best approach is for classifying product descriptions. First, in Section 4.1, we give an overview of the data collection process for the evaluation of the HPC framework. We also briefly discuss how we implemented the HPC framework for the purpose of this evaluation. Then, in Section 4.2, we give an extensive evaluation of the HPC framework, which includes a discussion of the results for the considered feature selection methods and the classifications algorithms.

4.1 Data Collection

For the evaluation of the feature selection methods and classification algorithms, we collected a large data set of product descriptions. The product descriptions are obtained from Amazon.com, using the Amazon Web Services (AWS) API [1]. This process was implemented in Java. The product category taxonomy that is used in the evaluation is constructed from existing Amazon.com categories. Because Amazon.com contains many product categories (around 120,000), we have chosen to use only a subset from all these categories. For the evaluation of the category mapping algorithm, we used data sets from CircuitCity.com and Amazon.com.

There are in total 319 product categories in the constructed product taxonomy, which is a simplified but representative view of the original taxonomy. The categories are located in a hierarchical taxonomy that consists of 4 levels. On the first level, there are 4 categories: 'Electronics', 'Office Products', 'Musical Instruments', and 'Clothing'. In order to have enough data for the training and testing of the classification models, the data set of product descriptions is collected in such a way that the minimum number of products per category is 200. The total number of collected product descriptions is 419,832, with 18,206 unique brands. Unfortunately, only 235,105 products are annotated with a brand. The same holds for the price; only 201,519 product descriptions contain a price. In order to speed-up the retrieval of the product descriptions, a multi-threaded crawler was developed in order to fetch and process the product descriptions.

4.2 Results

In this section, we discuss the evaluation of the different aspects of the HPC framework. We first evaluate the HPC framework for scenario 1, i.e., when no category is present in the product description and the product description needs to be classified to one of the existing categories. Then, we focus on the performance of the feature selection and classification algorithm components, as well as the overall performance of the HPC framework. Next, we evaluate the proposed algorithm for scenario 2, i.e., when a category is present in the product description. This consists of an evaluation of the proposed category mapping algorithm.

Although we do not show graphs for every pair of a feature selection algorithm and a classification algorithm (due to the high number of combinations), we stress that we evaluated all possible combinations for both the title property and the features description property. In the text we sometimes refer to these results by numbers instead of graphs.

4.2.1 Feature Selection Approaches

The four feature selection methods that are evaluated are Term Frequency (tf), Mutual Information (mi), Information Gain (ig) and Chi Square (chi). The reason for choosing these feature selection methods is that Information Gain and Chi Square have shown good results in the literature [52]. Furthermore, in a general text categorization context, the Term Frequency method performs surprisingly well as well, while the Mutual Information was found to perform badly [52]. We want to investigate if these findings also hold when the employed data set consists of product descriptions.

Figures 5 and 6 show us a comparison of all pairs of the four feature selection methods for the title property and features description property, respectively. Given a feature selection size, each comparison is based on the number of same features that have been selected by the corresponding two feature selection approaches. On the x-axis of the figures, the feature selection sizes are shown. On the y-axis the ratio between the number of same features and the total selected features is shown. For performance reasons, the comparison is performed on a subset of the data set with 3000 product descriptions. The reason why the x-axis range in Figure 5 is lower than the x-axis range in Figure 6 is because the title property contains less features to choose from than the features description property.

There are several interesting findings that follow from these two figures. First, we observe that the Information Gain and the Chi Square method have a high overlap in selected features for both the title and features description property. This is in line with findings of other studies, where Information Gain and Chi Square have been found to be highly correlated in terms of accuracy [34, 52]. For the title property, at a feature selection size of 50, more than 45 features are the same ($ratio > 0.90$). Second, the results suggest that the Term Frequency method selects features similar to those from Information Gain and Chi Square only for larger total number of selected features. This indicates that the findings of [52] (i.e., a strong correlation between Term Frequency, Information Gain, and Chi Square) partially applies also to product descriptions. Third, we observe that the Mutual Information shows low ratios for all methods. Only when the feature selection size is 100 or larger, the Mutual Information and Term Frequency method start showing a resemblance in their feature selection process. Last, the results of the comparisons for the features description property, shown in Figure 6, indicate that the ratio pair ordering is the same as the ratio pair ordering of the product title. We do notice that most of the ratios are lower than the ones for the product title, which suggests that the product descriptions are more heterogeneous than product titles and that this causes more variation between the methods.

Besides analyzing the relatedness of the different feature selection methods, we also analyzed the performance of each feature selection method. For this we use the accuracy metric from information retrieval, which is in our context equal to the precision because we always classify a product description and we consider this to be a *positive*. Figures 7 and 8 give us an overview of the accuracy of the feature selection methods for several feature selection sizes. The goal is here to compare feature selection methods, which means that we need to fix the classifier for now. Later on, we will discuss the different combinations of feature selection methods and classification algorithms and their performance. We chose to use the Naïve Bayes classifier in this case because it is fast, and more importantly, it requires no parameters to be set. This is useful because the performance of the Naïve Bayes is then affected only by the used feature selection method. The results for these experiments are obtained by performing a five-fold cross-validation procedure on a data set of 5,000 products and the four main root categories.

From the results we obtain three interesting findings. First, the Information Gain and Chi Square methods are relatively similar and have the highest accuracy. For the title property, as shown in Figure 7, we observe slightly higher accuracy values for the Chi Square method,

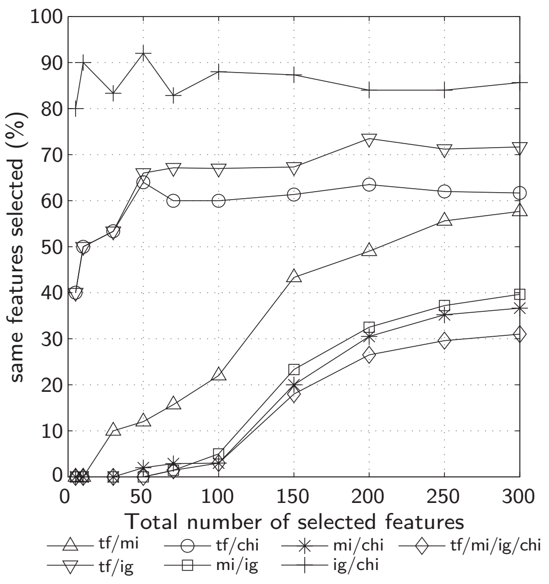


Fig. 5. Comparing feature selection methods similarity for the title property.

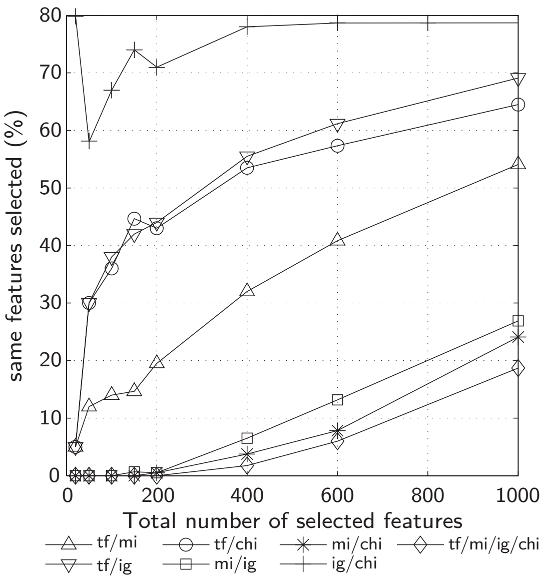


Fig. 6. Comparing feature selection methods similarity for the features description property.

but we have found these differences not to be significant at a 95% confidence level, using a paired t-test. On the other hand, for the features description property, shown in Figure 8, the difference between the Information Gain and Chi Square is significant at a 95% confidence level. We can conclude that the Information Gain shows significantly better results for the features description property. The reason for this is most likely that the heterogeneity in the features description values makes it difficult for the Chi Square feature selection method to measure the degree of independence between the selected features and the categories. At the same time, with its higher performance, the Information Gain method seems to be able to more easily measure the reduction in entropy when knowing the feature.

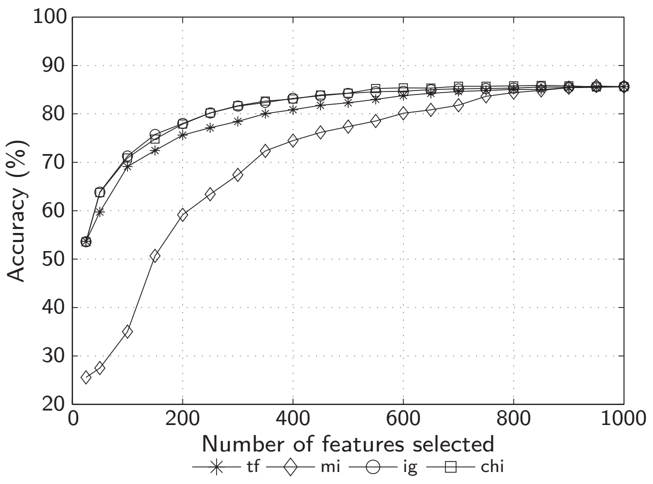


Fig. 7. Comparing feature selection methods on accuracy for the title property.

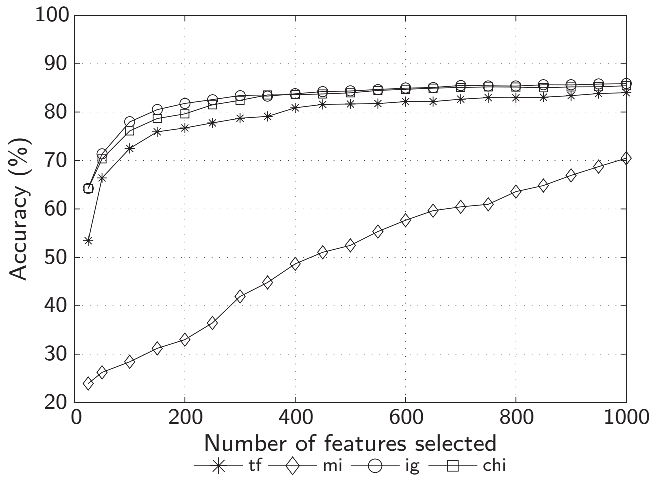


Fig. 8. Comparing feature selection methods on accuracy for the features description property.

Second, we observe that the Term Frequency method performs better than the Mutual Information method. The weakness of the mutual information criterion, i.e., that the values are strongly influenced by the marginal probabilities of terms, is validated by these results. [52] reported similar findings on a Reuters news data set. The authors found that the Information Gain and Chi Square methods give the best accuracy, but that the Term Frequency method, although performing worse, is highly correlated with the two. The authors suggest to use the Term Frequency method because the trade-off between effectiveness and computational cost, compared to the Information Gain and Chi Square method, is in favor of the Term Frequency method. Our results support this claim, both for the title property as for the features description property.

Third, when considering the general influence of the number of features selected (x-axis), we observe that for the title property, the performance increases more gradually than for the features description property. For example, for the title property, the accuracy does not change anymore at approximately 600 features, while for the features description property, the accuracy barely changes after 400 selected features. This indicates that the title property is more sensitive to the number of selected features than the features description property.

4.2.2 Classification Algorithms

In the previous section, we focused on the feature selection algorithms. In this section we focus on the evaluation of the different classifier components. We only present the results for the k-Nearest Neighbor classifier and the Support Vector Machines classifier. We already presented the results for the Naïve Bayes algorithm in the previous section.

K-Nearest Neighbor classifier. The initial impression of the k-Nearest Neighbor (kNN) classifier is that it does not perform very well. We find that the kNN classification technique is not able to deal appropriately with the product description data. This is different from what other authors have found, where kNN was able to deliver acceptable performance for the purpose of general text categorization [10].

In order to obtain valid accuracy values, we again performed a five-fold cross validation procedure on the top level categories, with 3000 training product descriptions. Figures 9 and 10 show the accuracy results for two different configurations with the kNN classifier. The figures show the accuracy for different values of k, i.e., the number of selected neighbors, and for two features selection methods. Overall, the variance of the results is low, i.e., there is not much difference between the feature selection methods, except for the Mutual Information method, which performs far worse than the others. This is illustrated by Figure 9, where we can see that the Mutual Information reaches a level of 0.55 accuracy when 200 features are selected. In contrast, we found that other feature selection methods only need 50 features to obtain such an accuracy (or higher). The same holds for the features description property, i.e., only the performance is even worse, with an accuracy of 0.45 at 400 selected features.

The results also show us that sometimes less is more, e.g., Figure 10 shows us that the accuracy is higher when 200 features are selected than when 400 features are selected. The only exception is the Mutual Information, which shows an approximately linear relationship

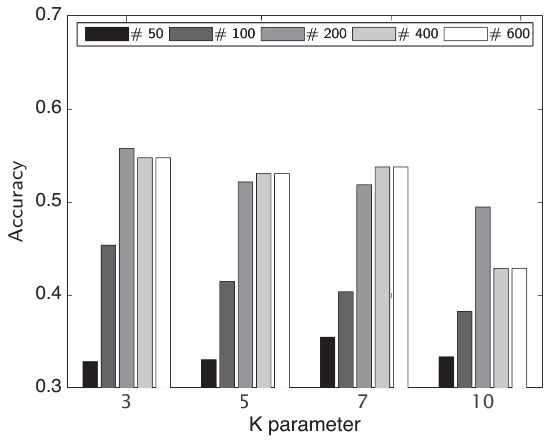


Fig. 9. Summary of the accuracy on the title property for the kNN classifier, using the Mutual Information features selection method.

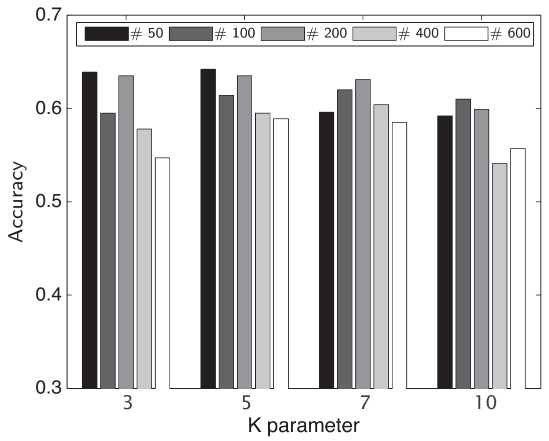


Fig. 10. Summary of the accuracy on the features description property for the kNN classifier, using the Chi Square features selection method.

(which eventually diminishes) between the number of selected features and the accuracy.

Furthermore, we observe that when k increases, the kNN classifier gets more sensitive for modeling noise. Figure 10 shows us that at $k = 7$, the accuracy for selected feature counts 400 and 600 is higher than at $k = 10$. It seems as if the kNN picks up noise from the extra selected features because of the higher k value.

Support Vector Machines. For the evaluation of the Support Vector machines we performed the same cross validation procedure (five-fold, with 3000 training samples). We have chosen for the SVM one-against-one approach. With the one-against-one approach, one needs to train more SVMs than with the one-against-all approach. Our experiments showed that the one-against-one approach performs better with respect to accuracy. That is why we provide a thorough evaluation of the one-against-one approach (with many parameter combinations). Further, we fixed the choice of the kernel. We choose the Radial Basis function (RBF) kernel [2] because the RBF kernel was found to give the best performance on text categorization [13]. Following from our experimental setup, we have to optimize only two parameters: the box constraint in the dual form notation of the SVM definition, and the γ parameter, which determines the width of the RBF kernel.

The general results of the SVM classifier are better than the kNN method, as we found the highest accuracy to be 78.07%. We observe that the SVM classifier is very sensitive to the two parameters. Typical behavior is shown in Figure 11. We notice that a value $\gamma = 1$ is not suitable as the accuracy does not exceed 60% and the accuracy drops as the number of selected features increases. [13] reports an optimal γ of 0.6. Our results show that the optimal value of γ for our data set is much larger, somewhere around 50. The higher value of γ indicates that the classification model required for our task is relatively complex, i.e., the influence of single features can be quite large. The reason for this is that the employed data set is to a relatively high degree semi-structured when compared to traditional text classification data sets (such as news articles or blog posts).

The most optimal configuration of the SVM classifier, for the features description property,

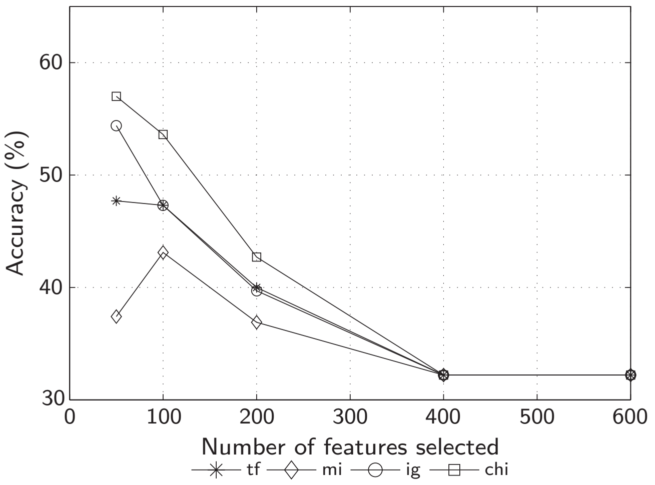


Fig. 11. Accuracy on the title property for the SVM classifier (with $\gamma = 1$, box constraint = 100).

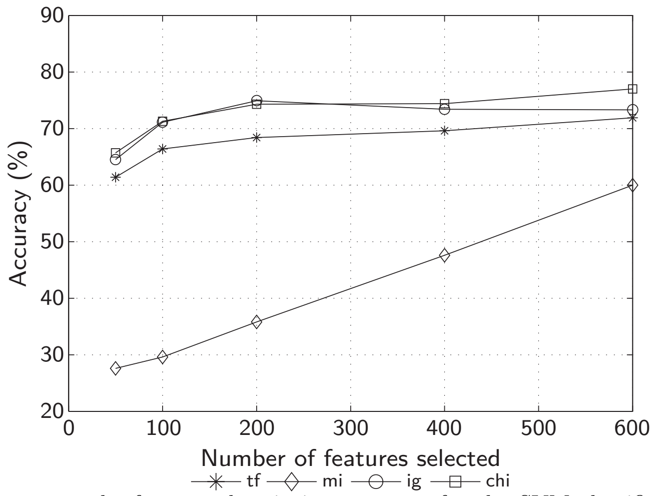


Fig. 12. Accuracy on the features description property for the SVM classifier (with $\gamma = 50$, box constraint = 100).

is for $\gamma = 50$ and box constraint = 100. The accuracy results for this configuration are shown in Figure 12. We observe that the overall accuracy is higher for the features description property, most likely because of the extra terms it contains compared to the title property, which helps the classifier to obtain an accurate classification.

A surprising result is that the best accuracy of the Naïve Bayes classifier is higher than the best accuracy of the SVM classifier. As shown in Figures 7 and 8, for both the title and features description property, the Naïve Bayes classifiers obtain an accuracy well above 80%, while the accuracy of the best SVM does not exceed 80%. Studies from the past have shown that the Naïve Bayes classifier can achieve comparable performance as the SVM classifier [12] and our findings provide further evidence for this claim.

4.2.3 Evaluating the HPC Framework

In this section, we evaluate the HPC framework as a whole. Table 1 shows us the accuracy of a $K = 3$ HPC classification system. The results are obtained through cross validation and 3000 training product descriptions. In this case, both for the title and features description, a Naïve Bayes classifier is used with an Information Gain feature selector set to select 400 features. The Price/Brand classifier is implemented using Quadratic Discriminant Analysis (QDA) [23]. This method aims to classify the category given the brand and the price. The price is first transformed by applying the natural logarithm. This classifier is trained only on categories that contain the brand.

Table 1 show the accuracy for each considered level of the category hierarchy. The first level is the level where the root product categories reside. For example, we can see that the Naïve Bayes classifier on the title has achieved an accuracy of 74.55% on the first level. The 'Total' column indicates the total accuracy for a level. This is the accuracy of the system, the other column are referred as the accuracy of the individual classifiers.

One might expect that the prices across product categories for a certain brand follow a particular distribution, and are thereby useful as an input for a classifier, however, this

Table 1. Accuracy for $K = 3$ classification system, with Naïve Bayes for title and features description, and Information Gain with select count equal to 400.

Level	Total	Title	Features desc.	Price/Brand
0	82.87%	74.55%	85.48%	13.21%
1	64.76%	69.37%	60.31%	13.24%
2	79.94%	82.65%	83.86%	34.58%

Table 2. Accuracy for $K = 3$ classification system, with Naïve Bayes for title, SVM ($\gamma = 50$, box constraint=100) for features description, and Information Gain with select count equal to 400.

Level	Total	Title	Features desc.	Price/Brand
0	75.04%	74.55%	72.92%	13.21%
1	64.35%	62.14%	66.00%	15.36%
2	81.42%	83.45%	77.63%	37.23%

Table 3. Accuracy for $K = 3$ classification system, with Naïve Bayes for title and features description, and Information Gain with select count equal to 1000.

Level	Total	Title	Features desc.	Price/Brand
0	83.52%	74.55%	86.13%	13.21%
1	64.45%	69.37%	60.80%	13.21%
2	82.42%	82.97%	84.11%	34.58%

assumption fails for the Amazon.com data set. As we can see in Table 1, the precision on level 0 is 13.21%. The Price/Brand property has also been evaluated with other classifiers, such as logistic regression and neural networks, although the results remained the same. From these results, we can conclude that the price and brand are not usable in this context.

We can further observe that at level 0, the feature description is the best property to choose, as the accuracy of the classifier on this property is 85.48%. For the second level, the title is the best property to be chosen as it has the highest accuracy on that level. One possible explanation for this is that at the second level, the model words from the title boost the classifier more than they do on the first level, where the classification is more coarse-grained. Finally, on the last level, the classifier on the feature description performs the best.

Table 2 shows us an example where the accuracy, on level 0, of the system is higher than the individual classifiers. In this case, the best classifiers is trained on the title property, giving a 74.55% accuracy. Levels 1 and 2 show different behavior than in Table 1, as on level 1 the features description property scores better and on the third level the title property.

Last, Table 3 shows us a system where only Naïve Bayes classifiers are used on the title and features description and 1000 features are selected by the Information Gain method. This combination gives the best results, with a 83.52% accuracy on level 0. For levels 1 and 2, similar results are obtained as for the classifier in Table 1, i.e., the title scores better on level 1 and the features description scores better again on level 2, with an average accuracy of 76.80%.

Table 4. An excerpt of the golden standard category mappings.

Original category	**#1 choice**	**#2 choice**	**#3 choice**	**#4 choice**
Blu-Ray & DVD Players	Blu-ray Players	DVD Players	DVD Drives	Car DVD Players
Networking & Internet	Networking	Networking Hub and Switches	Other Network Devices	
Power Supplies	System Power Supplies			
Webcams	Web Cameras	Digital Cameras		
Memory/Ram	Random Access Memory (RAM)	Computer Memory	Memory Cards	

Table 5. Results of the category mapping algorithm using $\gamma = 0.80$.

Manual mapping	**Percentage assigned to**
1st choice	77.27%
2nd choice	6.36%
3rd choice	8.18%
4th choice	0.91%
5th choice	0.91%
1th, 2nd, 3rd, 4th, or 5th choice	93.63%
Misclassification	6.37%

4.2.4 The Category Mapping Algorithm

For the evaluation of the Category Mapping algorithm, we collected 110 unique categories from CircuitCity.com and Amazon.com. After collecting these categories we manually mapped the collected categories to the Shopping.com categories. We do not provide only one mapping per category, but a list of possible correct mappings for all 110 categories. The first category on the list is the best choice, the second was the second best, etc. Table 4 shows some examples of these manual annotations. This manual mapping is used to benchmark our algorithm.

The goal of the algorithm is to map categories as much as possible to categories specified in the first chosen category, but there is always some subjectivity involved. For instance, for the mapping of 'Blu-Ray & DVD Players' one could assign 'DVD players' as first choice while one could also assign 'Blu-Ray Players' as a first choice.

The algorithm for category mapping has only the threshold parameter γ (not to be confused with the SVM γ parameter). In order to obtain the optimal value, we experimented with values between 0 and 1 with a step size of 0.05. Using this procedure, we determined that $\gamma = 0.80$ gives the best results. Table 5 shows the results for the algorithm with $\gamma = 0.80$ on the 110 categories. We observe that only 6.37% of the 110 categories are not correctly mapped to one of the corresponding manually chosen Shopping.com categories. This yields that 93.63% of the categories are correctly mapped to one of the corresponding five manually assigned categories. 77.27% of the 110 categories, which is 82.53% of total percentage correctly classified categories (93.63%), are mapped to the first manually chosen category.

5 Conclusions and Future Work

This paper proposes the Hierarchical Product Classification framework for the purpose of product classification using a product category taxonomy. The framework defines a classification system with K levels that is used to classify a product description to one of the leaves. The innovative part of the framework stems from several aspects. First, the framework uses *classification recipes* to construct classifier nodes. The classification recipes allow for flexible classifiers, i.e., different classifiers and different feature selection can be used on each of the levels of the product category taxonomy. Furthermore, in order to provide a more complete picture of the components needed to perform high quality product classification, we have evaluated several feature selection methods and classification techniques.

From the obtained results we can draw several conclusions. First, we have found the k-Nearest Neighbor algorithm to be unsuitable as an independent classifier. Besides the computational cost, the accuracy is too low to be useful in practice. Furthermore, we have shown that with our product data set, the Naïve Bayes classifier can perform better than Support Vector Machines, obtaining an average accuracy of 76.80% for product classification. When considering the properties of a product description, we have found that the features description provides better predictors for the top levels but that the title provides better predictors for the lower levels, except for the last level, where the features description gives again better results.

In the case that a product description contains a category, we make use of our proposed Category Mapping algorithm, which is a novel algorithm that makes use of semantic and syntactic matching. The algorithm achieves a precision of 93.63% on a manually mapped test set. It makes use of the average Levenshtein similarity in order to deal with syntactic variations of product categories and the cosine similarity for semantic similarity. WordNet is used to obtain a set of synonyms for each word in the product category, increasing the search space, and thus, the recall.

In future work we want to further investigate the interaction effects between the classification algorithms and the feature selection methods. One approach would be to research different combination strategies at different levels of the product category taxonomy. Another approach would be to use ensemble techniques to combine classifiers, i.e., a classifier on the title, a classifier on the features description, and a classifier on both the title and description. A third option is to use ensemble techniques to combine and evaluate the category mapping algorithm with the previously presented text classifiers.

Acknowledgment

Damir Vandic is supported by an NWO Mosaic scholarship for project 017.007.142: *Semantic Web Enhanced Product Search (SWEPS).*

References

1. Amazon.com. AWS - Amazon Web Services, 2017. `http://aws.amazon.com/`.
2. C. M. Bishop. *Pattern Recognition And Machine Learning.* Springer-Verlag, 2007.
3. S. Chakrabarti, B. Dom, R. Agrawal, and P. Raghavan. Using Taxonomy, Discriminants, and Signatures for Navigating in Text Databases. In *Proceedings of the 23rd International*

Conference on Very Large Data Bases, pages 446–455. Morgan Kaufmann Publishers Inc., 1997.

4. K. W. Church and P. Hanks. Word Association Norms, Mutual Information, and Lexicography. *Computational Linguistics*, 16(1):22–29, 1990.
5. S. D'Alessio, K. Murray, R. Schiaffino, and A. Kershenbaum. The Effect of Using Hierarchical Classifiers in Text Categorization. In *Proceedings of 6th International Conference Recherche d'Information Assistee par Ordinateur*, pages 302–313, 2000.
6. Y. Ding, M. Korotkiy, B. Omelayenko, V. Kartseva, V. Zykov, M. Klein, E. Schulten, and D. Fensel. GoldenBullet: Automated Classification of Product Data in E-commerce. In *Proceedings of the 5th International Conference on Business Information Systems*, 2002.
7. S. Dumais and H. Chen. Hierarchical classification of Web content. In *Proceedings of the 23rd Annual International Conference on Research and Development in Information Retrieval*, pages 256–263. ACM, 2000.
8. C. Fellbaum, editor. *WordNet: An Electronic Lexical Database (Language, Speech, and Communication).* The MIT Press, May 1998.
9. D. Fensel, Y. Ding, B. Omelayenko, E. Schulten, G. Botquin, M. Brown, and A. Flett. Product Data Integration in B2B E-Commerce. *IEEE Intelligent Systems*, 16(4):54–59, 2001.
10. E.-H. Han, G. Karypis, and V. Kumar. Text Categorization Using Weight Adjusted k-Nearest Neighbor Classification. In *Proceedings of the 5th Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 53–65. Springer-Verlag, 2001.
11. P.-Y. Hao, J.-H. Chiang, and Y.-K. Tu. Hierarchically svm classification based on support vector clustering method and its application to document categorization. *Expert Systems with applications*, 33(3):627–635, 2007.
12. J. Huang, J. Lu, and C. X. Ling. Comparing Naive Bayes, Decision Trees, and SVM with AUC and Accuracy. In *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on*, pages 553–556. IEEE, 2003.
13. T. Joachims. Text Categorization with Support Vector Machines: Learning with Many Relevant Features. In *Proceedings of the European Conference on Machine Learning*, pages 137–142. Springer-Verlag, 1998.
14. Y. S. Kim, B.-J. Yum, J. Song, and S. M. Kim. Development of a recommender system based on navigational and behavioral patterns of customers in e-commerce sites. *Expert Systems with Applications*, 28(2):381–393, 2005.
15. D. Koller and M. Sahami. Hierarchically Classifying Documents Using Very Few Words. In *Proceedings of the 14th International Conference on Machine Learning*, pages 170–178. Morgan Kaufmann Publishers Inc., 1997.
16. Y.-H. Lee, P. J.-H. Hu, T.-H. Cheng, and Y.-F. Hsieh. A Cost-sensitive Technique for Positive-Example Learning Supporting Content-Based Product Recommendations in B-to-C E-commerce. *Decision Support Systems*, 53(1):245 – 256, 2012.
17. V. I. Levenshtein. Binary Codes Capable of Correction Deletions, Insertions, and Reversals. *Soviet Physics Doklady*, 10(8):707–710, 1966.
18. D. D. Lewis, R. E. Schapire, J. P. Callan, and R. Papka. Training Algorithms for Linear Text Classifiers. In *Proceedings of the 19th Annual International Conference on Research and Development in Information Retrieval*, pages 298–306. ACM, 1996.

19. T. Li, S. Zhu, and M. Ogihara. Hierarchical Document Classification Using Automatically Generated Hierarchy. *Journal of Intelligent Information Systems*, 29(2):211–230, 2007.
20. C.-F. Lin and S.-D. Wang. Fuzzy Support Vector Machines. *IEEE Transactions on Neural Networks*, 13(2):464–471, 2002.
21. C.-H. Lin and H. Chen. An Automatic Indexing and Neural Network Approach to Concept Retrieval and Classification of Multilingual (Chinese-English) Documents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 26(1):75–88, Feb 1996.
22. A. McCallum, R. Rosenfeld, T. M. Mitchell, and A. Y. Ng. Improving Text Classification by Shrinkage in a Hierarchy of Classes. In *Proceedings of the 15th International Conference on Machine Learning*, pages 359–367. Morgan Kaufmann, 1998.
23. G. J. McLachlan. *Discriminant Analysis and Statistical Pattern Recognition.* Wiley, 2004.
24. T. Mitchell. *Machine Learning.* McGraw Hill, 1996.
25. S. Mulpuru, V. Boutan, C. Johnson, S. Wu, and L. Naparstek. Forrester Research eCommerce Forecast, 2014 to 2019. `https://goo.gl/6b1fh3`, 2017.
26. L. J. Nederstigt, D. Vandic, and F. Frasincar. A lexical approach for taxonomy mapping. *Journal of Web Engineering*, 15(1&2):84–109, 2016.
27. W. K. Ng, G. Yan, and E.-P. Lim. Heterogeneous Product Description in Electronic Commerce. *SIGecom Exchanges*, 1(1):7–13, 2000.
28. N. Oza, J. Castle, and J. Stutz. Classification of Aeronautics System Health and Safety Documents. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 39(6):670–680, Nov 2009.
29. M. F. Porter. An Algorithm for Suffix Stripping. *Readings in information retrieval*, pages 313–316, 1997.
30. M. E. Ruiz and P. Srinivasan. Hierarchical Text Categorization Using Neural Networks. *Information Retrieval*, 5(1):87–118, 2002.
31. G. Salton, A. Wong, and C. S. Yang. A Vector Space Model for Automatic Indexing. *Communications of the ACM*, 18(7):613–620, 1975.
32. M. Sasaki and K. Kita. Rule-Based Text Categorization Using Hierarchical Categories. In *Proceedings of the 1998 IEEE International Conference on Systems, Man, and Cybernetics*, volume 3, pages 2827–2830, 1998.
33. F. Shih and S.-S. Chen. Adaptive Document Block Segmentation and Classification. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 26(5):797–802, Oct 1996.
34. S. R. Singh, H. A. Murthy, and T. A. Gonsalves. Feature Selection for Text Classification Based on Gini Coefficient of Inequality. In *Proceedings of the Fourth International Workshop on Feature Selection in Data Mining (FSDM 2010)*, volume 10, pages 76–85, 2010.
35. M. Steinbach, G. Karypis, and V. Kumar. A Comparison of Document Clustering Techniques. 00 034, University of Minnesota, 2000.
36. A. Sun and E. P. Lim. Hierarchical Text Classification and Evaluation. In *Proceedings of the 2001 IEEE International Conference on Data Mining*, pages 521–528. IEEE Computer Society, 2001.

37. C. Sun, N. Rampalli, F. Yang, and A. Doan. Chimera: Large-scale classification using machine learning, rules, and crowdsourcing. *Proceedings of the VLDB Endowment*, 7(13):1529–1540, 2014.
38. K. Toutanova, F. Chen, K. Popat, and T. Hofmann. Text Classification in a Hierarchical Mixture Model for Small Training Sets. In *Proceedings of the 10th International Conference on Information and Knowledge Management*, pages 105–113. ACM, 2001.
39. UNSPSC.org. United Nations Standard Products and Services Code, 2017. `http://www.unspsc.org`.
40. D. Vandic, S. S. Aanen, F. Frasincar, and U. Kaymak. Dynamic facet ordering for faceted product search engines. *IEEE Transactions on Knowledge and Data Engineering*, 29(5):1004–1016, 2017.
41. D. Vandic, J.-W. van Dam, and F. Frasincar. Faceted Product Search Powered by the Semantic Web. *Decision Support Systems*, 53(3):425–437, 2012.
42. H. Wang, Q. Wei, and G. Chen. From Clicking to Consideration: A Business Intelligence Approach to Estimating Consumers' Consideration Probabilities. *Decision Support Systems*, 56(0):397 – 405, 2013.
43. K. Wang, S. Zhou, and S. C. Liew. Building Hierarchical Classifiers Using Class Proximity. In *Proceedings of the 25th International Conference on Very Large Data Bases*, pages 363–374. Morgan Kaufmann, 1999.
44. T.-Y. Wang and H.-M. Chiang. Fuzzy Support Vector Machine for Multi-class Text Categorization. *Information Processing & Management*, 43(4):914–929, 2007.
45. A. S. Weigend, E. D. Wiener, and J. O. Pedersen. Exploiting Hierarchy in Text Categorization. *Information Retrieval*, 1(3):193–216, 1999.
46. W. J. Wilbur and K. Sirotkin. The Automatic Identification of Stop Words. *Journal of information science*, 18(1):45–55, 1992.
47. Y. Yang. Expert Network: Effective and Efficient Learning from Human Decisions in Text Categorization and Retrieval. In *Proceedings of the 17th Annual International Conference on Research and Development in Information Retrieval*, pages 13–22. Springer-Verlag New York, Inc., 1994.
48. Y. Yang. An Evaluation of Statistical Approaches to MEDLINE Indexing. In *Proceedings of the American Medical Informatics Association Annual Fall Symposium*, pages 358–362, 1996.
49. Y. Yang. An Evaluation of Statistical Approaches to Text Categorization. *Information retrieval*, 1(1-2):69–90, 1999.
50. Y. Yang and C. G. Chute. An Example-Based Mapping Method for Text Categorization and Retrieval. *ACM Transactions on Information Systems*, 12(3):252–277, 1994.
51. Y. Yang and X. Liu. A Re-examination of Text Categorization Methods. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 42–49. ACM, 1999.
52. Y. Yang and J. O. Pedersen. A Comparative Study on Feature Selection in Text Categorization. In *Proceedings of the Fourteenth International Conference on Machine Learning*, pages 412–420. Morgan Kaufmann Publishers Inc., 1997.
53. Y. C. Yang. Web User Behavioral Profiling for User Identification. *Decision Support Systems*, 49(3):261 – 271, 2010.

54. H. Yu, J. Yang, and J. Han. Classifying Large Data Sets Using SVM's with Hierarchical Clusters. In *Proceedings of the 9th International Conference on Knowledge Discovery and Data Mining*, pages 306–315. ACM, 2003.