Journal of Intelligent Learning Systems and Applications, 2015, 7, 58-73
Published Online May 2015 in SciRes. http://www.scirp.org/journal/jilsa
http://dx.doi.org/10.4236/jilsa.2015.72006

Classifying Unstructured Text Using Structured Training Instances and an Ensemble of Classifiers

Andreas Lianos, Yanyan Yang

School of Engineering, University of Portsmouth, Portsmouth, UK
Email: andreas.lianos@port.ac.uk, linda.yang@port.ac.uk

Received 1 April 2015; accepted 23 May 2015; published 26 May 2015

Abstract

Typical supervised classification techniques require training instances similar to the values that need to be classified. This research proposes a methodology that can utilize training instances found in a different format. The benefit of this approach is that it allows the use of traditional classification techniques, without the need to hand-tag training instances if the information exists in other data sources. The proposed approach is presented through a practical classification application. The evaluation results show that the approach is viable, and that the segmentation of classifiers can greatly improve accuracy.

Keywords

Ensemble Classification, Diversity, Training Data

1. Introduction

This research proposes a novel ensemble classification methodology. The novelty of the method lies in the ability to train the classifiers with data that are in a different format from the values that they later classify. The proposed methodology is presented through a practical application, which serves as a consistent example and an evaluation framework.

This work is part of a research project that proposes a market-independent recommender system, focusing on identifying only a handful of products to serve as final recommendations. The aforementioned recommender system needs to gather information about user needs and then connect them to product attributes. For example relate "every day use of a camera", to a sufficient amount of "Megapixels". As this analysis is not done for every attribute, the need arises to realize which attributes have an important role in the decision making. To au-

How to cite this paper: Lianos, A. and Yang, Y. (2015) Classifying Unstructured Text Using Structured Training Instances and an Ensemble of Classifiers. *Journal of Intelligent Learning Systems and Applications*, **7**, 58-73.
http://dx.doi.org/10.4236/jilsa.2015.72006

tomatically acquire this knowledge, the short bullet-point lists that precede the product's extended description are used. **Figure 1** contains a snippet of a typical web-page that can help visualize this information. By examining the bullet-point lists of many products, one can observe which attributes are encountered more often, and thus deduce which ones are important for the whole product category. The challenge is to develop an appropriate methodology that can correctly match each bullet-point text to a specific attribute.

The difficulty of this task lies in the fact that there are no training data readily available for such bullet-point texts (bullets). With no known pairs of "bullet→attribute" to train classifiers, supervised classification methods cannot be used. However, what is available is the structured table of attributes that exists in the same web pages (**Figure 1**). This table has two pieces of information, the attribute's name and the attribute's value. Closely observing bullets reveals that they are vaguely constructed by using either the name component of the table, the value, or both. With this knowledge residing in the table, all that is needed is a methodology that can tap into it. It seems that there is a gap in the literature on how to tackle the aforementioned problem and utilize data that exist in a different format from the texts that need to be classified. At the same time using this data is crucial, as it allows the completion of the task without hand-tagging training instances (*i.e.* manually classifying bullets).

This work proposes a new classification approach (DICE, Diverse Classifiers Ensemble) that uses multiple different classifiers[1]. Each classifier has a unique feature set and can be trained from different training data, which consequently can be in different formats. That allows each classifier to capture a different "kind" of bullet, *i.e.* name based, value based, or combined. The results of the classifiers are integrated together, forming an ensem-

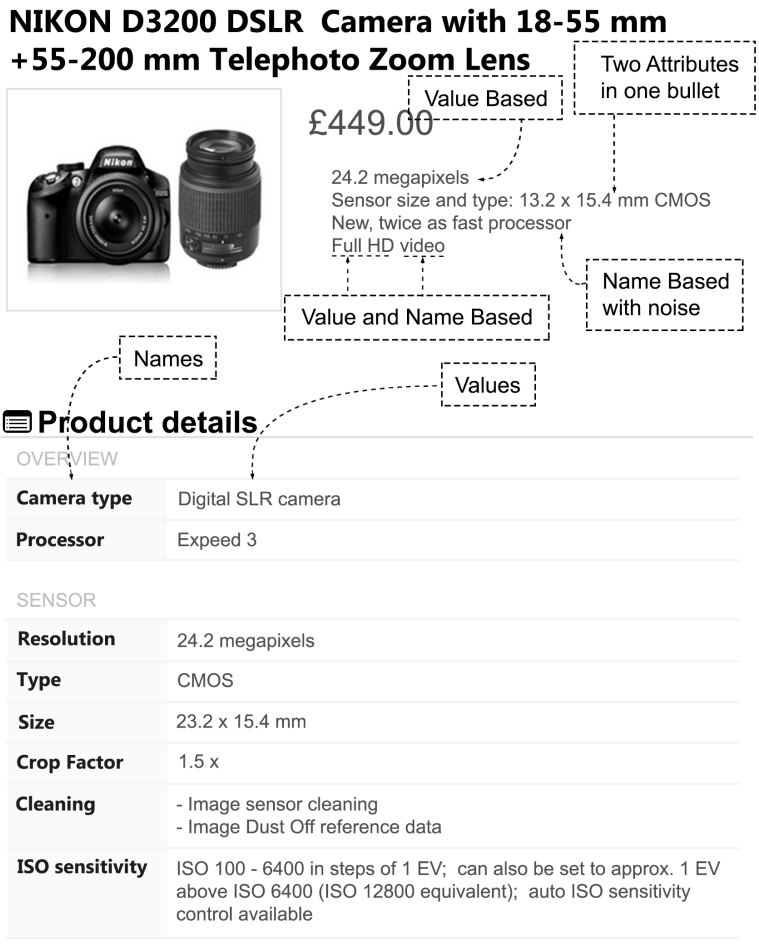

Figure 1. Snapshot from an actual web-page. Important areas are highlighted with dashed lines.

[1]For clarity, we will refer to the ensemble as "the ensemble" in contrast to "member classifiers". DICE will be used to describe the approach as a whole.

ble that produces the final result. The classifier being responsible to identify one kind of text has no information on how to treat the others. As such, classifiers are switched off for the ensemble when their opinion is not relevant. In essence, since there are no training instances for bullets as a whole, there is an attempt to find training instances that might contain just pieces of it. The ensemble then utilizes only the classifiers that have identified some pieces of useful information.

To demonstrate this approach a prototype of DICE has been developed. It is based on four classifiers with unique feature-sets specifically tailored for consumer electronics. Because this work utilizes training data that are not usable by other methods, there is no baseline for comparison. As such, the performance has been evaluated simply by measuring the accuracy (*i.e.* correctly classified bullets). In addition to standalone testing, the classifier has been put in context, where its ability to correctly identify the top 7 most important attributes has been measured. This information can then be used from the discussed recommended system.

Our work contributes to the field of ensemble classification by proposing a methodology that can utilize training data that are not in the same format with the texts that need to be classified. This opens up a window to tapping into many data sources that were previously unusable. The proposed methodology differs dramatically from other ensemble classification approaches such as bagging [1] and boosting [2]. It uses classifiers that are inherently different from each other, rather than diversified instances of one base classifier. On the way, light has been shed on what constitutes a proper feature set for classifiers specifically tailored for consumer electronics. Finally, the presented empirical evaluation can be used as a starting point for future researches, by providing initial parameters.

In the market of consumer electronics the acquired knowledge can be used in multiple ways, such as forming meaningful comparisons, automatic product scoring, assistance in decision-making and personalized product recommendations. Making this information available outside specific markets and free from proprietary APIs, allows for cross-market solutions and third party implementations of the aforementioned tools. In addition, the methodology can be generalized and transferred to other domains that can utilize bespoke solutions. By allowing the use of training data that would be otherwise unusable, it would save a considerable amount of time and effort that would be used in hand-tagging training instances.

The next section discusses how previous research interacts with ours, and it offers some basic reference material. Section 3 is a thorough presentation of the proposed approach, followed by a detailed analysis of the feature set of each classifier and some specifics on their implementation. Section 4 presents the evaluation datasets. It also contains the results of the evaluation with the relevant discussion. Finally Section 5 summarizes some of the key points derived from this work, along with future plans and potential improvements.

2. Related Work

The problem of text classification is not new and there is ample literature relating to it (some very interesting resources are [3]-[5]). The most relevant application comes from information retrieval where documents are ranked according to their similarity to a given query. To demonstrate how this is relevant, attribute's names can be treated as documents and bullets as queries. Transferring the technique, attribute’s names can be ranked according to their similarity with the given bullet. The limited applicability of these techniques comes from the fact that a bullet may or may not contain the same words with the attribute’s name. For example the bullet “24 Megapixels” has no similarities with its respective attribute name “Resolution”. As a result looking only at the attribute’s name makes the correlation between them impossible, rendering a simple application of information retrieval methodologies ineffective.

To overcome the previous limitation we can borrow from a methodology used to match the fields of one database to those of another [6] [7]. This method examines the contents of the database (rather than the table names) to look for columns whose contents look similar. This however causes the opposite problem of the one previously mentioned, favoring the bullets that contain values over those that contain names. For example the bullet “Waterproof”, has no similarities with the respective values of the attribute which tend to be either “yes” or “no”. In addition, different attributes may have similar values. For example height and width both can be 10 cm, and waterproof and shockproof both may be either yes or no. This opens up a window for false positives, and as identified by their authors, is a major limitation inherent to the approach [7].

The reason both these methods have limited applicability, is because one cannot be sure which information is or is not contained in a bullet. At the same time there are no training instances for bullets, so that this information cannot be learned automatically. We merge the two approaches and develop an ensemble classification

model, in order to lift their limitations and utilize their strengths. Ensemble classification is a well established research area with stable results. Maclin and Opitz [8] have recently shown that the two predominant ensemble methodologies from the previous century, still outperform single classifiers. Ensemble classification strives to replace a single classifier with multiple—usually similar—classifiers, whose results are then recombined. The main diversification points of the various methods are a) how to obtain the member classifiers and b) how to combine their results.

The well proven methodologies such as bagging [1], boosting [2] [9], and their later variations, meet at a common point; the member classifiers are similar to each other. Even though this makes direct use of any such methodologies impossible, there is much to learn from the relevant research. Hansen [10] has shown that for an ensemble to be successful, member classifiers should be, among other things, diverse. "Diverse" in this context, means that the classifiers should make different errors given different values to classify. Dietterich [11] has later explained why this requirement is true. Drawing upon this idea the developed classifiers have completely distinct feature sets that do not overlap. In truth the classifiers are designed so that where one fails, the other excels.

With the way to obtain the member classifiers set, the interest shifts to the second point; the way that results of members are combined. The prevailing approach is plurality voting [10], or simple averaging depending on the type of task [12]. Simple averaging was also used in the initial implementation of bagging [1], but various studies have since suggested that weighted averaging might improve accuracy [13] [14]. Most weighted average methodologies however, either assume that the classifiers are similar, or that it is possible to calculate a measure of the effectiveness of the classifier by using part of the training data. None of these assumptions hold true for this work. In a nutshell, this happens because the classifiers are trained from the structured table of attributes, but then classify texts from the bullet-points. Since there is no prior knowledge of where bullet-points should be classified, the effectiveness of the classifier cannot be calculated without hand tagging data. Creation of artificial evaluation instances from the training data is guaranteed to suffer from overfitting.

That is not to say that existing methodologies have nothing to offer. Based on the most successful approaches, a weighted average method is used. Since the weights cannot be calculated from the data, a fixed weighted scheme is used instead. This is possible because the classifiers themselves are not created dynamically and their relation can be predetermined. Another approach suggests the use of the perceived confidence of the classifier as a start for calculating weights [15]. Borrowing from this idea, the results of certain classifiers are completely negated when their opinions are not relevant.

A lot of work has been done on the classification techniques themselves. Simple Bayesian classifiers have been found to perform very well, especially with small feature sets [16] [17]. Another popular approach to classification is Neural Networks. As Neural Networks attempt to find connections between features, they are particularly effective where features are strongly correlated. They are however expensive to train, especially as the number of features grow [18] [19]. Finally, Decision Trees is another widely used classification approach [20] [21]. For the interested reader Kotsiantis *et al*. [22] have written a thorough review of the most common classification techniques to date and Xhemali *et al*. [23] a comparison of Naïve Bays, Neural Networks and Decision Trees.

Reflecting on the related work, it becomes obvious that even though this research draws upon the ideas and findings of the existing literature, it cannot be directly compared with any of them. This work discusses how to tackle the problem of training data being in a different format from the values that they will later classify. Partially addressing this problem by creating multiple but different classifiers, it also examines if these can be used effectively in an ensemble. There seems to be little knowledge in the existing literature on how to address both these issues. Concerning the development of feature sets, no literature could be found specifically tailored for consumer electronics. Naumann *et al*. [7] discusses an application that matches fields of one database to those of another. The proposed feature set is rather generic, and as such it has been used as a starting point for value-based matching.

3. Methodology

3.1. Introduction

This work proposes DICE, a classification approach based on an ensemble of multiple and diverse classifiers. The aim of DICE is the ability to utilize training data that are not in the same format as the texts that need to be classified. The presented methodology is tailored around a specific problem, and the developed classifiers target

consumer electronics. The details of the approach are presented here to create a seamless example of the proposed approach. Our work does not present a generic way that can be transferred as-is to a different context. It does however suggest a way of tackling the aforementioned problem. Our discussion on the proposed methodology offers valuable insight on what to look for and how to effectively transfer this work. As it might be obvious, the feature sets of the classifiers are tailored around a specific problem and not form part of the generalized approach. In truth, this method can work with any number and type of classifiers. However, the specifics of each classifier are discussed for multiple reasons: a) it is an opportunity to discuss and explain what to look for when designing classifiers to work with this approach, b) It offers an insight on what an effective feature set is for consumer electronics and c) it is mandatory for the reproducibility of the results.

The proposed methodology is applied to data acquired directly from web-pages. Each web-page offers two pieces of information; the bullet-point list that is used as a short description of the product and the detailed table of attributes that usually follows. As discussed in Section 1, **Figure 1** shows a snippet of a webpage where both parts are clearly visible. The bullet-point lists of all products are added together, forming a single list that contains all the texts that will be later classified. The tables of attributes of all products are also added together. The combined table provides on the one hand the training data, and on the other hand the classes, *i.e.* the list of possible attributes a bullet might belong to. Essentially the classification process will try to match each bullet to an attribute.

To predict where a bullet belongs, each individual classifier will calculate a score for every possible attribute. Even though the attribute with the highest score is selected and becomes the opinion of the classifier, the classifier will calculate scores for all of them. Since the possible attributes are the same for all classifiers, the only difference in the outcome of the classifiers is the scores. The way that the score is calculated depends on the classification model (e.g. naive bays, neural networks, etc.) and does not affect the methodology. For the given context four classifiers like the above are developed, each using different criteria to decide to which attribute a bullet refers to. As a result four scores are calculated for every possible attribute, one from each classifier. The scores are added together using a fixed weighted scheme. Once added, the highest combined score becomes the opinion of the ensemble. **Table 1** shows an example of this procedure. It is interesting to note that the opinion of the ensemble may be different from the opinions of any of the member classifiers, as demonstrated in Table 1. This is an advantage of using weighted average over plurality voting.

As a result of the above procedure, every bullet is assigned to an attribute. Measuring the repetition of these attributes becomes the proposed metric of importance.

Before discussing the details of each classifier, it is mandatory to understand the nature of bullets. Bullets mainly consist of either the value of an attribute, the name, or both. For example, when browsing for cameras "4× optical zoom" explicitly mentions both the attribute's name (optical zoom) and its value (4×). However, if a tablet is "7 inch" only the value is mentioned, while the attribute's name (screen size) is implied. On the other hand "Waterproof" contains only the attribute's name and the value (yes or no) is completely omitted. Bullets may also contain connective or promotional words (such as: with, up to, new, amazing, etc.). These words constitute noise since they are not part of the training data. Some bullets may consist purely of noise words (e.g. Modern look, Amazing offer). Since there is no correct attribute for these bullets their classification is ipso facto impossible, introducing a baseline error rate. **Figure 1** is a screenshot of a web-page marked with further examples.

Table 1. Example classification of the bullet "24 Megapixels".

	Optical Zoom	Resolution	Flash Type
Classifier 1	0.5	0.7	**0.8**
Classifier 2	0.3	0.4	**0.5**
Classifier 3	**0.5**	0.4	0.1
Classifier 4	**0.4**	0.3	0.2
Ensemble	1.7	**1.8**	1.6

Each classifier provides a probability score for each attribute. Results are added together for the ensemble. Highlighted in bold are the highest scores of each classifier, representing the choice each classifier would make if working on its own. In this example, the ensemble would finally classify the bullet to "Resolution".

Training data are acquired from the structured table of attributes of product web-pages (see **Figure 1**). Each attribute translates to a triplet consisting of the category name, the attribute name and the value. For example a few attributes from **Figure 1** are the following:

(sensor) resolution = 24.2 megapixels

(general) size = 14 × 16 × 6 cm

In this paper we present the category in brackets. A category is a heading in the table that defines a group of features. In **Figure 1** the visible categories are "overview" and "sensor". Each pair of category and attribute name creates a unique class, e.g. "(sensor) resolution". If no category is found then none is used. Each attribute (and thus each class) can have multiple values, as these originate from different product web-pages. Because web-pages don't always follow the same format, the same attribute may appear multiple times under a slightly different name or a under a different category. This may result in the creation of different classes for the same attribute, e.g. "(memory) memory card" and "(general) memory card type". For the scope of this paper these attributes are treated as different classes.

3.2. The Classifiers

To utilize the multiple pieces of information contained in bullets four classifiers were developed. They have widely different feature sets, are implemented by different classification models and are trained with different data. The combined feature set of all classifiers is able to capture a wide range of characteristics of the bullets.

Separating the classifiers allows the modification of results separately for special cases, essentially pre-encoding knowledge into the system. Some rules are used in special cases that completely negate the results of a classifier. This is one of the major benefits of having distinct classifiers rather than one classifier that aggregates all the features. Another benefit is the ability to implement each classifier with the most appropriate classification model. Apart from the gain in accuracy, this also enhances performance. By keeping each classifier independent the process can be easily parallelised. It also allows the use of more efficient models where the complexities of a Neural Network are not needed. Finally, splitting the classifiers allows for training using different training data. This makes the separation not just beneficial, but mandatory, as it would be impossible to achieve otherwise. Essentially a bullet might contain data that exist in either training set, and one of the classifiers will be able to identify it.

3.2.1. The Name Classifier (NC)

The name classifier (NC) matches bullets that either consist of or contain the attribute's name. To create the feature set all possible stems from all attribute and category names are extracted. Each stem is then used to create a boolean feature for the classifier, e.g. "contains_optic", "contains_memor" and so on. As a result the classifier contains a variable number of features depending on the stems extracted from the training data. This makes the feature set very targeted for the given domain. Common stopwords are removed[2] and only words with 4 or more characters are considered. **Table 2** shows an example of the input text and the produced signature for an NC classifier with four features.

NC is trained using the names and categories of attributes. Essentially, NC is being taught that if a bullet contains any words of the class itself, then it belongs to that class. Using the category name makes it possible to identify bullets that need to mention the category in order to be meaningful. For example "3 cm sensor size" and "3 cm display size" ought to mention the category so as not to be confused. One training instance is created from

Table 2. Example signatures for the NC.

Bullet	Produced signature, assuming the following feature set			
	{contains_optic,	contains_zoom,	containts_waterproof,	contains_flash}
12× optical zoom	{true,	true,	false,	false}
Waterproof up to 10 m	{false,	false,	true,	false}
24.4 megapixel	{false,	false,	false,	false}

[2]Based on a list common stop words, as found at https://github.com/arc12/Text-Mining-Weak-Signals/wiki/Standard-set-of-english-stopwords.

the name only and one from the combination of the name and category. **Table 3** shows an example of the training instances created from the attributes "(sensor) size" and "(display) size".

NC is implemented with a Naive Bayes classifier since there are only two different training instances for each class. As a result there are no underlying patterns to be discovered. Bullets that do not contain any words from the lexicon look exactly the same from the scope of the classifier, and are erroneously regarded similar (see last example in **Table 2** where the bullet is purely value based, and all features are set to false). On such occasions the results of NC are completely negated and not used for the ensemble. It is recommended to identify similar scenarios for every classifier, and negate the results when the classifier has no meaningful information to make a decision.

3.2.2. The Value Classifier (VC)

The Value based classifier (VC) matches bullets that have similarities with the value rather than the name of an attribute. It uses a variance of features and the core idea is to match letter and symbol patterns. The first set of features consists of finding the presence for each of the following characters:

abcdefghijklmnopqrstuvwxyz<>,.?;:'&()/*-+="

This produces boolean features such as "contains_a", "contains_b", etc. The specific occurrence of capital letters is excluded because they create false correlations. The SWC discussed later is instead responsible to capture these letters. Five more generic features are also used. These are the number of digits, the number of integers (as full non-decimal numbers, not as a one by one digit), the number of words (*i.e.* the number of whitespace characters minus 1), the total number of upper-case letters, and the total length of the value.

Finally, because direct count of each digit separately creates more noise than discriminating power, number ranges are used instead and the count shifts to how many times a number is found in any range. This process is called discretization and is a form of dimensionality reduction [24]. Dougherty *et al*. [25] has found that the performance of Naive Bayes is significantly improved when features are discretized regardless of the method. A number of discretization techniques were tested, *i.e.* K-means clustering, equal width binning and equal frequency binning, with the latest performing the best during pilot testing [26] [27].

Pilot testing for bin sizes between 40 and 80 has shown that the final accuracy fluctuates only around 7%, and the final number chosen was 60. However, if the training data are not enough to adequately fill all 60 bins this number is automatically reduced until all bins have at least 3 values. To create the bin ranges, the unique occurrence of numbers is being used. This is mandatory, as in the opposite scenario multiple bins might be filled with only one number. Using bands of numbers allows capturing of numbers of similar magnitude even if those appeared very rarely in the training data, or did not appear at all. To illustrate, VC is able to see the relation between "16 Megapixels" and "15.9 Megapixels" even if one of these values was never present in the training data (assuming both these numbers end up in the same bin). We recommend similar techniques for continuous data where identifying close number makes sense. However, discretization can create false positives if the main source of numbers is not continuous numbers, such as telephone numbers or versions.

Pilot testing has shown that using a Multilayer Perceptron (MLP) is a significant improvement over a Decision Tree and a Naïve Bayes classifier[3]. This holds true even when the Kernel Based Distribution [28] is used in Naive Bays to compensate for non-normal distributions of measurements, which tends to be the case with the acquired data. The use of letters as features creates strong connections between them, practically allowing the

Table 3. Creation of training instances for the NC.

Training instance		Class
size	→	(sensor) size
sensor size	→	(sensor) size
size	→	(display) size
display size	→	(display) size

Training instances for the attributes "(sensor) size" and "(display) size".

[3]Based on pilot testing performed by the authors.

classifiers to detect words. This is a plausible explanation as to why the MLP performs better. VC is trained using all the values of all the attributes.

3.2.3. Units Classifier (UC)

The Units Classifier (UC) finds the units in which an attribute is measured. This is achieved by getting the next word or symbol after a number, if any number is present, regardless of whether it is an actual unit or not. Scanning all the values of all attributes a list with all the possible different units for the domain is populated. The nominal feature "measured_in" is then constructed using the contents of that list as possible options (e.g. measured_in = [cm, hours, megapixel]). The possible options contain very little noise which is almost limited to number delimiters such as dashes, slashes or commas. Without loss of generality, these common elements can be removed to reduce the chance of false positives.

-*/?;,.()[]{}\|~

There is nothing to indicate that this parameter should change, even if the methodology transfers to completely different domains. A dictionary is used to combine different expressions of the same unit (e.g. double quotes with inch and inches, centimetre with cm and so on). Because the list of units is populated by the training data, it is very targeted to the given domain, and thus very effective.

Two more options are manually added to the list of units; one for the case where no units seem to be present, and one for the case where units are present but UC is not aware of them. The later happens when a unit did not exist in the training data, and in essence this option represents "all other units". A second feature is finally used detecting if an attribute is boolean or not. In case more than one units are present only the latest one is used. **Table 4** shows an example of the input text and the produced signature.

In case a bullet falls under any of the two special cases, and is not boolean, its results are negated during the combination. This is because too many classes donot contain any units, and as far as UC can detect they all look alike, while in truth their only similarity is that they contain no units. This is also the main reason UC needs to be separated from VC, highlighting once again the power of having multiple separate classifiers.

The predicting power of UC lies in the fact that generally, no more than a couple of classes will use the same units. However, once limited to these few classes there is no way of knowing which one is correct. That, along with the negation of results for values with no units, nicely supplements the value classifier. As the decision is based on two mutually exclusive features (it either contains a unit, or is boolean), a Naive Bayes classifier was used. UC is trained with all the values of all attributes. As UC will adapt its features to the given training set, it should be possible to transfer the whole classifier to completely different problems with virtually no changes.

3.2.4. Special Words Classifier (SWC)

The Special Words Classifier is used to detect the special words and acronyms often used in product descriptions. In essence it discriminates between the normal and the possibly special use of letters and numbers. The number of each of the following characters is used to create the feature set

[A-Z] [a-z] [0123456789.-]

For example, "numOf_A", "numOf_B", and so on.

To identify the special use of character in values, values are first split in white-spaces and the parts are treated individually. Some simple rules are used to determine special versus regular use of each character. Initially all bracket symbols are removed from the value. Then single character parts and parts that are a valid number (integer or decimal) are discarded. Then special use of each character is assumed a priori, and negated in the following cases:

Table 4. Example signatures for the UC.

Bullet or training instance	Produced signature, assuming the following feature set	
	{[megapixels, x, cm, mm, inch, hot, no_units_found, unknown_unit],	is_boolean}
12 Megapixels	{megapixels,	false}
Yes	{no_units_found,	true}
Waterproof up to 10 m	{unknown_unit,	false}
2 AA batteries last 360 shots	{shot,	false}

- Numbers or letters that represent a number with units (e.g. 9600 p)
- Numbers or letters in a "something-by-something" format, regardless of units in the end (e.g. 100 × 20, 1536 × 9600 p)
- Letters from fully lowercase words, or words that only start with a capital (e.g. optical, Optical)
- Symbols at the start or end of words (e.g. the dot at the end of "5× optical zoom." has no special meaning)

Table 5 presents some example cases where special character meaning is assumed.

SWC is particularly effective in capturing versions, models, technology acronyms etc. The classifier could be used as-is in any context that might contain similarly modeled special words. It also demonstrates how to utilize characteristics of the text that are relevant to the given domain. For this classifier a Multilayer Perceptron was preferred because the connection between features can actually identify whole words rather than individual letters. SWC is trained with all the values of all attributes. Bullets that contain no special words would be falsely related to attributes whose values tend not to contain any special words. To avoid false correlations, the results of SWC are completely negated in the above scenario.

3.2.5. The Ensemble

To combine the results a fixed weighted scheme can be used. In essence, much like the negation of results, it is a form of knowledge pre-built into the system. The weights are set empirically to specifically match consumer electronics. Any of the discussed parameters can be adjusted to optimize results for completely different domains. However, that would require manually labeling pilot data so that adjustments could be evaluated. The final weights used are the following:

NC: 6.50, VC: 0.25, UC: 2.00, SWC: 1.25

It should be noted that different classifiers give results of different magnitudes, and thus the weights cannot be used as an indicator of the importance of each classifier.

3.3. Pre-Filtering of Classes and Training Data

Since each discovered attribute translates to a class, the possible classes are defined directly by the raw data. This might cause problems during classification as the original list of classes might contain a lot of noise. To deal with this issue three filters have been developed that remove invalid classes and trim noisy data. The first filter is looking for outliers in the values, removing values particularly long in comparison to the average value length of that attribute. An interquartile range test is used to define many outliers in one pass [29]. The main values filtered are ones that contain explanatory texts. For example the class "(features) built-in flash" has typical values of yes or no. If one value is "yes, guide number of 12 m @ 100ISO", then it is removed from the training data. As outliers are defined in contrast to the other values, removing them and repeating the procedure might discover new outliers.

The second filter removes classes with less than 1% occurrence. This mainly removes classes that are most commonly found under a different name. It also removes classes that are misspelled, or that are made ad-hoc for a specific product. Some examples of such cases are "(general) size" that is misspelled, or "(sensor) flash" that is incorrectly placed under the "sensor" category. This filter greatly reduces the number of possible outcome classes making the classification task easier and increasing accuracy. At the same time however, accuracy will go down since some texts will no longer have a correct class to be classified. The nature of the data or the problem should be examined to decide if this filter should be applied or not. Data with high number of rare classes,

Table 5. Example cases where special character meaning is assumed.

Bullet or training instance	Parts where special meaning is assumed. The respective characters would be true in the bullet's signature.
H.234 movie recording	H.234
Triple XD engine	XD
250 cd/m^2	cd/m^2
Conventional 4:3	4:3
2 ms gray-to-gray	gray-to-gray

and low number of texts that should classify to these classes would benefit the most. Of course, if all classes need to be maintained then this filter cannot be used. In this context the algorithm is looking for the most important attributes. By definition these should have a significant occurrence between product pages, greatly benefiting for the use of this filter.

The final filter transforms or erases classes with an average value length over a given threshold. Initially the filter attempts to identify if the value can be split into multiple smaller parts. If a pattern is found then the value is split and each part is regarded as a separate value. For example, the class "(sensor) ISO modes" mainly contains values that list all modes that a camera supports (e.g. ISO 100/ISO 200/ISO 400/ ISO 800/ISO 1600/ISO 3200). These are split into individual pieces and treated as separate values. If a pattern is not found the whole class is disregarded. This filter is effective because by definition bullets are small texts, so such particularly long values are never encountered. The evaluation data backs this up. On the Camera dataset more than 80% of the retrieved bullets have a length of less than 20 characters, and a total average length of 17 (before any filtering). The same filter is applied to the extracted bullets, and particularly long bullets are removed. The application of this filter depends on the nature of the problem. If the problem is interested in identifying only a specific type of classes, then the rest can be removed to improve accuracy. This is effectively what this filter does for our work, since classes with long and textual values are not relevant.

4. Evaluation

To evaluate the presented methodology a demonstration system was developed in Java. The classifiers were implemented using the well-known WEKA framework for machine learning and data mining [30]. Two different datasets were used for evaluation, *i.e.* Cameras and TVs. Data were obtained using live commercial websites, namely Argos (190 products), Curries (30), Jessops (116) and Tesco (407) for Cameras, and Curries (207), Pixmania (225) and Tesco (300) for TVs. All available products were retrieved from each web site[4]. To retrieve the web pages a web crawler was developed. The category's entry page is seeded to the crawler that then downloads all product pages consecutively. A custom extractor is used for each site to identify the needed information. Specifically, the extractor identifies the bullets, as well as the category headings and pairs of the attribute's name and value. A lot of research has been done on generic extraction frameworks. Each framework emphasizes on different aspects, but none is capable of identifying and labeling all the needed pieces of information for this work. Ghani *et al*. [31] have developed a methodology that can extract pairs of attributes-values from web-pages. Even though their work could be applied, for the creation needs of the evaluation datasets additional information is needed (*i.e.* bullets and category headings). As a result manual extraction was preferred, in the sake of efficiency. **Table 6** presents some more information about the created datasets. An interesting observation is that the exact same bullet is used 2.8 times for cameras, and 8.21 times for TVs.

In regards to the pre-filtering of data, two iterations were found more than adequate for outlier removal, with the first iteration alone removing 96% of total values removed. A third iteration does not remove any additional values. As the maximum length reduces, the filter that removes long values improves the accuracy. However, the reduction in length increases the risk for erroneously discarded bullets. The maximum allowed length has been set to 40 which permits 95% of bullets to go through[5].

Since training and evaluation data are in a different format, overfitting is not directly possible. For the same reason K-fold cross validation cannot be applied. Instead, a random X portion of the web-pages is used to provide the training data, and all the bullets are always used for evaluation. We run tests for X values of 5%, 10%, 20%, 30%, 40% and 50%. We repeat each test 30 times, and present the averages.

Table 6. Evaluation datasets.

Set	Web-pages	Attributes	Total values	Bullets	Unique bullets
Cameras	743	297	18,799	930	327
TV	732	420	21,259	1659	202

Total values are the number of values all attributes have if put together. All information is before any filtering takes place.

[4]Webpages retrieved 2/2/2014.
[5]Presented numbers are averaged over both datasets.

Table 7 and **Table 8** show the data gathered for each X, for Cameras and TVs respectively. Roughly 50% of the total attributes (297) can be discovered from just 5% of the webpages, rising to 92% for 50% of the webpages. TVs follow a similar trend with 60% for X = 5%, and 90% for X = 50% (420 maximum attributes). This is easily explained as attributes repeat often between webpages, especially between webpages of the same source. As they repeat, the most important ones can be discovered earlier, since they are mentioned in more webpages. A detailed evaluation of the filtering mechanism is not part of the paper. However, it is interesting to note that as X increases filtering is able to identify and discard not just more attributes, but a bigger portion of the attributes. This supports the previous explanation that important attributes are discovered early. It further suggests that with more attributes the mechanism adapts and the perception of what is important changes.

For the purposes of evaluation three lists were manually created.

- ListA notes the correct classes for each bullet. Bullets might have more than one correct class as they might directly refer to more than one attributes. For example, the bullet "3 inch LCD screen" contains both "(screen) size" and "(screen) type".
- ListB groups classes that refer to the same attribute but have different names. For example "(screen) type" and "(viewfinder) type". The criteria for this annotation were that the values of the attribute must be in a similar format. For example the classes "(flash) type" and "(other) flash" are not merged, as the typical values of the first are "built-in, pop-up, etc", and the values of the second are "yes/no".
- ListC holds a directed graph of classes that contain other classes. For example "(sensor) type and size" contains the distinct classes of "(sensor) type" and "(sensor) size". The criteria for this annotation were that the parent class should offer at least all the information contained in any given child.

4.1. Classification Accuracy

To measure the accuracy of the classification ListA is expanded to include all similar classes from ListB. If the classification result exists in this expanded ListA, it is counted as correct. To continue the previous examples, if "3inch LCD screen" is classified to either "(screen) size", "(screen) type" or "(viewfinder) type", it is counted as correct.

Table 7. Training data gathered for the camera dataset.

Percentage of Webpages (X)	Attributes before Filtering	Total Values before Filtering	Attributes after Filtering	Total Values after Filtering	Attribute Reduction	Values Reduction
5%	149	967	130	954	13%	1%
10%	193	1845	157	1818	19%	1%
20%	222	3752	168	3674	24%	2%
30%	244	5596	176	5463	28%	2%
40%	260	7476	183	7302	30%	2%
50%	274	9620	182	9361	33%	3%

Table 8. Training data gathered for the TV dataset.

Percentage of Webpages (X)	Attributes before Filtering	Total Values Before Filtering	Attributes after Filtering	Total Values after Filtering	Attribute Reduction	Values Reduction
5%	253	1082	241	1028	5%	5%
10%	293	2085	279	2016	5%	3%
20%	328	4389	312	4320	5%	2%
30%	354	6489	334	6439	6%	1%
40%	370	8718	350	8646	6%	1%
50%	379	10667	356	10505	6%	2%

Figure 2 summarizes the accuracy of the classifiers over 30 iterations, for different amounts of training data (X), for both data sets. For example in **Figure 2(b)**, the first column of SWC shows that when trained with 5% of the available training data, SWC can predict the correct outcome 35% of the times. Subsequent columns show that the results of SWC remain more or less stable, regardless of the amount of training data.

"Max Ensemble" represents the potentially correct classifications of the ensemble, if weights were readjusted per iteration to achieve the maximum possible correct classifications. As expected, the chosen weights do not give the best possible results in every iteration. However, by comparing "Ensemble" to "Max Ensemble" it can be observed that results are consistently close. The gap between actual and potential accuracy reduces noticeably with more training data. This demonstrates how a fixed weighted scheme is an appropriate approach to combine the results of a highly diverse, not dynamically created ensemble.

It is immediately obvious from both charts that the ensemble has consistently better accuracy than any standalone classifier, for both data sets, for any amount of training data (X). Specifically, the improvement over the best performing classifier ranges from 144% to 168% for Cameras, and from 195% to 214% for TVs[6], depending on X. Please note that the Camera dataset has a baseline error rate of 16% due to noisy bullets with no respective classes, capping the maximum possible correct classifications to 84%. The equivalent number for TVs is only 2%[7].

Another interesting observation for the Camera dataset is that even through 3 out of 4 classifiers degrade their results as X increases, the ensemble improves. This is made possible by using weighted average rather than plurality voting. As explained in the methodology, this gives to the ensemble the ability to consider all the results, and not just the first picks of every classifier. Consequently, it can make a more informed decision. This is demonstrated in the results by the higher accuracy of the ensemble in comparison to any of the individual classifiers.

Since there are no other existing methodologies to tackle this problem, there is no baseline for comparison and results will need to be judged per se. To decide on whether the achieved accuracy is adequate, we further perform the following tests.

4.2. Method Effectiveness

Apart from the classification accuracy, the effectiveness of the whole procedure is also evaluated in context. To achieve this, we measure the existence of the top 7 attributes identified as important, in the list of the top 7 actually important attributes.

The list of actually important attributes occurs by measuring the repetition of each correct class from ListA. Utilizing ListB similar classes are merged. Finally, because the bullets that refer to a child class will refer to the

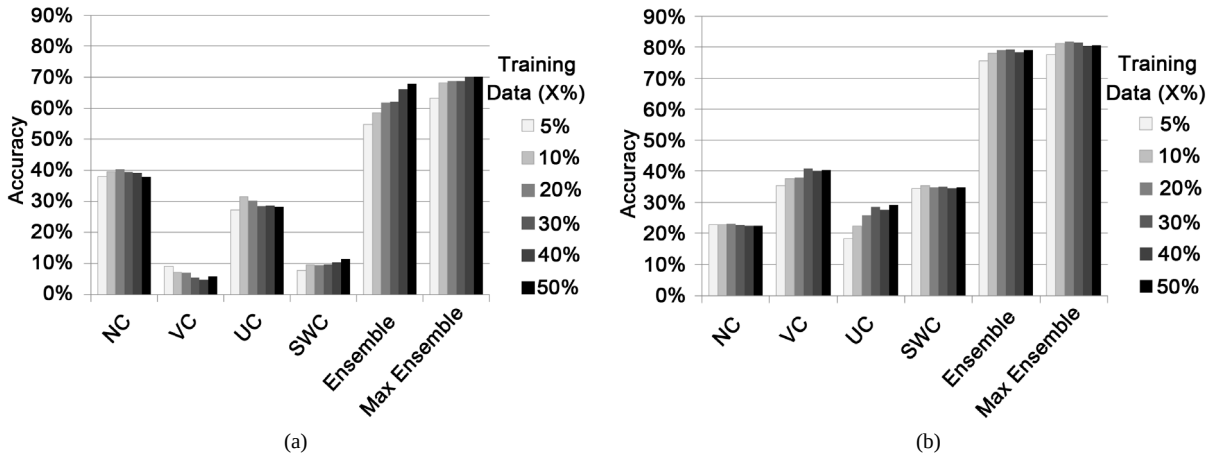

Figure 2. Correct predictions of each classifier for the camera dataset (a) and TV dataset (b), for different percentages of training data (X). (a) Camera classifiers; (b) TV classifiers.

[6]Due to constraints in processing power, the MLPs of VC and SWC were replaced with Naive Bayesian classifiers, for the TV dataset only.
[7]149 bullets out of a total of 930 for cameras. 29 out of for 1659 TVs.

parent class as well, parent classes are removed if any of their children are found (to make room for more classes). The list of identified attributes occurs by simply summing up the results of the classification, and thus remains independent of any human interaction. Both lists are finally trimmed to 7 elements[8]. Each identified attribute is checked for existence in the list of actual attributes. Attributes might exist either directly or by any of their children. Existence is also assumed if classes are found under different names.

Table 9 and **Table 10** contain the average results over 30 iterations for different amounts of training data (X). With 5% of the training data used the ensemble correctly identifies 75% of the top attributes for cameras and 81% for TVs. This implies that overfitting is indeed not an issue, since accuracy is already at high with just 5% of the data used. Increasing X offers a double benefit, as it improves both accuracy and stability. The rate of improvement is in line with the total correct classifications previously discussed. From a different perspective the top 7 most important attributes are always included in the top 11 identified, for both sets, for every iteration. To demonstrate the impact this has on practical applications, a developer (or better yet, a crowdsourcing system) looking to identify the 7 most important attributes, can now only examine 11, rather than 297 or 420 that were originally found in each set.

5. Discussion and Future Work

This work has shown that using highly diverse classifiers creates synergy, and dramatically improves results. It has also demonstrated that separated classifiers have multiple advantages. Each classifier can be implemented by the most appropriate model, which results in reduced training times and improved accuracy. In addition it allows training from different parts of the dataset, or from completely different and diverse datasets. This can be particularly useful in cases where training data are not readily available, as it allows training from data that are in a different format. The negation of results for certain classifiers is also an important technique, effectively using the classifier only when its results are relevant. Finally, filtering the training data is an essential step, as it reduces noise with minimal impact on the total training values.

The evaluation has shown that using a custom ensemble of classifiers is a feasible solution to the problem. Each classifier can target a specific subset of characteristics that the text might exhibit. This is effectively what

Table 9. Attributes identified as important that are actually important (camera dataset).

Percentage of training data (X)	Found in top 7	Relative standard deviation
5%	5.27 (75%)	16%
10%	5.43 (78%)	13%
20%	5.63 (80%)	10%
30%	5.63 (80%)	9%
40%	5.83 (83%)	6%
50%	5.76 (82%)	8%

Table 10. Attributes identified as important that are actually important (TV dataset).

Percentage of training data (X)	Found in top 7	Relative standard deviation
5%	5.70 (81%)	14%
10%	5.70 (81%)	10%
20%	5.63 (80%)	14%
30%	5.83 (83%)	10%
40%	5.73 (82%)	8%
50%	5.83 (83%)	6%

[8]Seven was chosen as a representative number based on the study of Miller [32], which identifies it as the maximum number before information overloading beings to appear.

allows the use of different training sources, since each classifier needs training only in a specific subset of characteristics, even if those characteristics will not be present in every text. The ensemble can correctly identify more bullets than each classifier would individually. This is mainly due to the negation of results, since the ensemble can decide without being affected by the opinions of the classifiers that do not have enough data to make a decision. The evaluation of the suggested method in context provides a better understanding of its power. Results show how this approach can be used to minimize the need for human interaction while maintaining zero error rates. Reducing the potential options by over 96% while still maintaining all the correct results, even in a worst case scenario, shows that this approach has a lot to offer.

In addition, the feature set of each classifier has successfully captured its respective aspect, providing some insight on what can be used as an effective feature set for product classification. Combining heuristic methods with existing classification techniques can utilize hidden characteristics of the available data. The algorithm offers the opportunity for tuning in various stages, providing valuable flexibility. Essentially, a targeted version of the system could be developed per product category or domain.

It is important to realize that even though the needs of this work were different, the developed classifiers are able to classify arbitrary texts to explicit classes. This allows the methodology to be used for other problems, such as consolidating multiple fields of a database or multiple databases to one. The presented methodology requires human involvement limited to the tuning of parameters before executing it. Default parameters for consumer electronics are presented in this paper. The outcome of the ensemble can be further improved if access to human intelligence is assumed. For example, in the case of this work the top 11 attributes can be presented to an expert so the top 7 can be selected. This is a significant improvement over presenting every attribute of the category. In the case of merging database fields, it would allow the suggestion of fields that might need to be merged.

Since there are misclassified values there is room for improvement. Potentially, this can be achieved with better weight tuning and more heuristic procedures during the combination. Methodologies that use the confidence of the classifier as a starting point for the weight might be applicable. The presented methodology only identifies one attribute from each bullet. It would seem beneficial to define this mapping as one to many, to best capture the information conveyed by the bullet. The whole algorithm can be further improved with even more targeted feature sets, since the potential for hidden characteristics in values is endless. As demonstrated, classifiers can be tailored to specific problems. In the presented methodology SWC captures characters with special meaning. Consumer electronics often contain a lot of acronyms, but in their absence the effectiveness of SWC would diminish. This however points to the idea that different problems might have different characteristics that can be exploited. The approach taken allows adding or removing classifiers with virtually no changes to other parts, creating yet another adjustment point.

It is in the immediate plans of this work to make practical use of the above methodology in a product recommender system. The automatic knowledge acquired will be filtered through a crowdsourcing system to further improve results. With the achieved level of accuracy this can happen with minimal involvement from the crowd. The final results will be then used for automatic product scoring, using only those attributes that have been identified as important.

6. Conclusion

This paper examines if an ensemble of bespoke and diverse classifiers can utilize training data that are not usable by traditional classification techniques. This has been achieved by developing multiple classifiers with unique feature sets. Each feature set targets different characteristics of the available training data, which carry enough information to provide classification of high accuracy, when combined in an ensemble. This segmentation allows each classifier to train on completely different data sources, since only a portion of the features needs to be available. The experimental results show that the developed approach is viable, and that the segmentation of classifiers can greatly improve accuracy. Application of the presented methodology will be required to define the relation between classifiers.

Acknowledgements

This work is supported by academic research funding from the University of Portsmouth.

References

[1] Breiman, L. (1996) Bagging Predictors. *Machine Learning*, **24**, 123-140. http://dx.doi.org/10.1007/BF00058655

[2] Schapire, R.E. (1990) The Strength of Weak Learnability. *Machine Learning*, **5**, 197-227. http://dx.doi.org/10.1007/BF00116037

[3] Manning, C.D., Raghavan, P. and Schütze, H. (2008) Introduction to Information Retrieval, Vol. 1. Cambridge University Press, Cambridge. http://dx.doi.org/10.1017/CBO9780511809071

[4] Sebastiani, F. (2002) Machine Learning in Automated Text Categorization. *ACM Computing Surveys*, **34**, 1-47. http://dx.doi.org/10.1145/505282.505283

[5] Forman, G. (2003) An Extensive Empirical Study of Feature Selection Metrics for Text Classification. *Journal of Machine Learning Research*, **3**, 1289-1305.

[6] Dasu, T., Johnson, T., Muthukrishnan, S. and Shkapenyuk, V. (2002) Mining Database Structure; or, How to Build a Data Quality Browser. *Proceedings of the* 2002 *ACM SIGMOD International Conference on Management of Data*, Madison, 3-6 June 2002, 240-251. http://dx.doi.org/10.1145/564691.564719

[7] Naumann, F., Ho, C.T., Tian, X., Haas, L. and Megiddo, N. (2002) Attribute Classification Using Feature Analysis. *Proceedings of the International Conference on Data Engineering*, San Jose, 2002, 271-271. http://dx.doi.org/10.1109/icde.2002.994725

[8] Maclin, R. and Opitz, D. (2011) Popular Ensemble Methods: An Empirical Study. ArXiv11060257

[9] Freund, Y., Schapire, R.E., *et al*. (1996) Experiments with a New Boosting Algorithm. *International Conference on Machine Learning*, **96**, 148-156.

[10] Hansen, L.K. and Salamon, P. (1990) Neural Network Ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **12**, 993-1001. http://dx.doi.org/10.1109/34.58871

[11] Dietterich, T.G. (2000) Ensemble Methods in Machine Learning. In: *Multiple Classifier Systems*, Springer, Berlin, 1-15. http://dx.doi.org/10.1007/3-540-45014-9_1

[12] Opitz, D.W. and Shavlik, J.W. (1996) Actively Searching for an Effective Neural Network Ensemble. *Connection Science*, **8**, 337-354. http://dx.doi.org/10.1080/095400996116802

[13] Perrone, M.P. and Cooper, L.N. (1992) When Networks Disagree: Ensemble Methods for Hybrid Neural Networks. DTIC Document.

[14] Rogova, G. (1994) Combining the Results of Several Neural Network Classifiers. *Neural Networks*, **7**, 777-781. http://dx.doi.org/10.1016/0893-6080(94)90099-X

[15] Maclin, R. and Shavlik, J.W. (1995) Combining the Predictions of Multiple Classifiers: Using Competitive Learning to Initialize Neural Networks. *Proceedings of the* 14*th International Joint Conference on Artificial Intelligence*, Montreal.

[16] McCallum, A. and Nigam, K. (1998) A Comparison of Event Models for Naive Bayes Text Classification. *Proceedings of the AAAI*-98 *Workshop on Learning for Text Categorization*, Madison, 1998, 41-48.

[17] Friedman, N., Geiger, D. and Goldszmidt, M. (1997) Bayesian Network Classifiers. *Machine Learning*, **29**, 131-163. http://dx.doi.org/10.1023/A:1007465528199

[18] Zhang, G.P. (2000) Neural Networks for Classification: A Survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C*: *Applications and Reviews*, **30**, 451-462.

[19] Ruck, D.W., Rogers, S.K. and Kabrisky, M. (1990) Feature Selection Using a Multilayer Perceptron. *Neural Network Comput*, **2**, 40-48.

[20] Quinlan, J.R. (1986) Induction of Decision Trees. *Machine Learning*, **1**, 81-106. http://dx.doi.org/10.1007/BF00116251

[21] Rokach, L. and Maimon, O.Z. (2008) Data Mining with Decision Trees: Theory and Applications. World Scientific Publishing Co., Inc., Singapore.

[22] Kotsiantis, S., Zaharakis, I. and Pintelas, P. (2007) Supervised Machine Learning: A Review of Classification Techniques. *Frontiers in Artificial Intelligence and Applications*, **160**, 3.

[23] Xhemali, D., Hinde, C.J. and Stone, R.G. (2009) Naïve Bayes vs. Decision Trees vs. Neural Networks in the Classification of Training Web Pages. *International Journal of Computer Science Issues*, **4**, 16-23.

[24] Liu, H. and Motoda, H. (1998) Feature Selection for Knowledge Discovery and Data Mining. Springer, Berlin. http://dx.doi.org/10.1007/978-1-4615-5689-3

[25] Dougherty, J., Kohavi, R. and Sahami, M. (1995) Supervised and Unsupervised Discretization of Continuous Features. *Proceedings of the* 12*th International Conference on Machine Learning*, Tahoe City, 9-12 July 1995, 194-202.

[26] Catlett, J. (1991) On Changing Continuous Attributes into Ordered Discrete Attributes. *Machine Learning—EWSL*-91

Lecture Notes in Computer Science, **482**, 164-178.

[27] Kerber, R. (1992) Chimerge: Discretization of Numeric Attributes. *Proceedings of the* 10*th National Conference on Artificial Intelligence*, San Jose, 12-16 July 1992, 123-128.

[28] John, G.H. and Langley, P. (1995) Estimating Continuous Distributions in Bayesian Classifiers. *Proceedings of the* 11*th Conference on Uncertainty in Artificial Intelligence*, Montreal, 18-20 August 1995, 338-345.

[29] Natrella, M. (2010) NIST/SEMATECH e-Handbook of Statistical Methods.

[30] Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P. and Witten, I.H. (2009) The WEKA Data Mining Software: An Update. *ACM SIGKDD Explorations Newsletter*, **11**, 10-18. http://dx.doi.org/10.1145/1656274.1656278

[31] Ghani, R., Probst, K., Liu, Y., Krema, M. and Fano, A. (2006) Text Mining for Product Attribute Extraction. *ACM SIGKDD Explorations Newsletter*, **8**, 41-48. http://dx.doi.org/10.1145/1147234.1147241

[32] Miller, G.A. (1956) The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information. *Psychological Review*, **63**, 81-97. http://dx.doi.org/10.1037/h0043158