Model Words-Driven Approaches for Duplicate Detection on the Web

Marnix de Bakker
marnixdebakker@zeelandnet.nl
Erasmus University Rotterdam
PO Box 1738, NL-3000 DR
Rotterdam, the Netherlands

Flavius Frasincar
frasincar@ese.eur.nl
Erasmus University Rotterdam
PO Box 1738, NL-3000 DR
Rotterdam, the Netherlands

Damir Vandic
vandic@ese.eur.nl
Erasmus University Rotterdam
PO Box 1738, NL-3000 DR
Rotterdam, the Netherlands

Uzay Kaymak
u.kaymak@ieee.org
Eindhoven University of Technology
PO Box 513, NL-5600 MB
Eindhoven, the Netherlands

ABSTRACT

The detection of product duplicates is one of the many challenges that Web shop product aggregators are facing. This paper presents two new methods to solve the problem of product duplicate detection. Both methods extend a state-of-the-art approach that uses the found model words in product titles to detect product duplicates. The first proposed method uses several distance measures to calculate distances between product attribute keys and values to find duplicate products when no matching product title is found. The second proposed method detects matching model words in all product attribute values in order to find duplicate products when no matching product title is found. Based on our experimental results on real-world data gathered from two existing Web shops, we show that the second proposed method significantly outperforms the existing state-of-the-art method in terms of F_1-measure, while the first method outperforms the existing state-of-the-art method in terms of F_1-measure, but not significantly.

Categories and Subject Descriptors

H.2.0 [**Database management**]: General; H.3.2 [**Information Storage and Retrieval**]: Information Storage—*record classification*; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*information filtering*

General Terms

Algorithms, Design

Keywords

Duplicate detection, Web shops, products, model words

1. INTRODUCTION

In the current Web era, a tremendous amount of products is sold online in a variety of Web shops. Despite the fact that different Web shops often sell the very same product, the information about this product can vary greatly over the different Web shops. Some Web shops display very detailed information, while others display only a small number of product attributes. One Web shop might only display properties like weight and manufacturer, while another might also display properties like color and dimensions. As a result, information about products tends to be scattered across different Web shops.

By aggregating data from various Web shops, one can obtain more detailed and accurate product information. Nevertheless, there is a significant problem that needs to be solved in order to combine product data from various Web shops. The problem is that a manual approach to product integration is infeasible, due to the vast amount of information, products, and Web shops. Therefore, this process needs to be automated. To automatically combine data from different websites, it is necessary to know how information from Web shops can be used to detect identical products, i.e., perform *duplicate detection*. Other product integration challenges, such as product schema alignment and product merging, are considered outside the scope of this paper.

Because the title of a product can vary on different Web shops, duplicate detection is not as simple as just finding product names that precisely match. This is especially true for electronic products like TV's. Take as an example the Web shop Bestbuy.com, which gives a TV the name 'Samsung - 40" Class / LCD / 1080p / 60Hz / HDTV', while the Web shop Newegg.com gives the very same product the name 'Samsung 40" 1080p 60Hz LCD HDTV LN40D503'. Because different Web shops can offer very different information about a product, duplicate detection on the Web is a challenging problem.

In this paper, we propose two solutions to product duplicate detection. A state-of-the-art solution [11] that has been previously used for this problem is based on extracting the so-called *model words* (words consisting of both numeric and alphabetic/punctuation characters) from the product names and comparing these to find duplicates. Model words are important in the process of duplicate detection, as information that is valuable for product duplicate detection (for instance product codes) often contains both alphabetic and numeric characters. The methods proposed in this paper extend this method and use it as a baseline. The first method that we propose combines the model words method for product names with accounting for similarities between product properties. The second proposed method not only uses model words for the product names, but also for the values of product properties.

The paper is organised as follows. In Section 2, we discuss related work on (product) duplicate detection. Section 3 presents two new methods for duplicate detection. The performance of the proposed methods and a state-of-the-art method are evaluated and compared in Section 4. Section 5 concludes the paper and suggests possible future work.

2. RELATED WORK

In this section we discuss related work that addresses the problem of (product) duplicate detection. Section 2.1 presents the state-of-the-art title model words method, which is our baseline. Section 2.2 addresses a product code-based method. The learnable string similarity method is discussed in Section 2.3 and last, in Section 2.4, filtering methods aimed at increasing the efficiency of duplicate detection are discussed.

2.1 The Title Model Words Method

In [11], model words from product names are used to identify duplicate products on the Web. To determine whether or not two products are identical, the authors use an algorithm that starts by calculating the word-based cosine similarity [10] between the two product names. If this similarity is above a threshold value, the products are considered duplicates. If the similarity is not greater than the threshold, the algorithm continues by extracting the model words from the names of the two products. Then, the algorithm checks if it can immediately conclude that the products are different. This is done by searching for word pairs (in each word pair there is one model word from the first product name and one from the second product name) where the non-numeric parts are approximately the same, but the numeric parts are different. A situation like this generally indicates different products, for example 'Samsung - 55" Class/ LED / 1080p / 120Hz / HDTV' vs. 'Samsung - 46" Class/ LED / 1080p / 120Hz / HDTV'. For this similarity check, the Levenshtein similarity [7] is used. If it finds such a word pair with different numeric parts and approximately the same non-numeric parts, the algorithm stops and gives as output that the two products are not duplicates.

If no such word pairs as previously described are found, the algorithm continues by calculating a new similarity between the product names. This similarity value is a combination of the cosine similarity between product titles and the average Levenshtein similarity between the two sets of words in the product titles. Subsequently, a check is performed to find model word pairs that are likely to be the same. This means that a model word pair has equal numeric parts and approximately equal non-numeric parts. If such pairs are found, the aforementioned cosine/Levenshtein similarity is then updated to increase the weight of the model words. Once this is done, the final step is to check if the final similarity value is greater than a threshold. If this is the case, then the algorithm returns true and the products are considered duplicates.

The methods proposed in this paper expand upon the title model words method that is described above. Different from the original title model words method, our methods use additional information from the product attributes to improve the effectiveness of duplicate detection. In addition, the newly proposed model words method uses not only model words from the title, but also found in the product attribute values.

2.2 The Product Code Based Method

In [6], the authors describe a method for duplicate detection using (model) words in the product titles. Their algorithm decides which products are duplicates using the similarity between product titles. An important element of this method for finding matching product titles is extracting product codes from product titles. These product codes are unique numbers for each product, designated by their manufacturer. To do this, the algorithm first removes common features such as dimensions, weight, voltage, etc., from the title. Then it filters the title, removing stop words and words that appear frequently in product offers of several different manufacturers. After this, the method generates candidates (generally consisting of up to three model words) for product codes, using a manually created list of regular expressions that capture knowledge on the syntactical structure of product codes. Last, it uses Web verification to check the correctness of the extracted candidates. This is done by submitting a query to a Web search engine for a candidate; the fraction of the results containing the corresponding manufacturer, with respect to all results, is then used to verify the correctness of each candidate.

The methods proposed in this paper are more flexible than those used in [6], since they do not need product codes to be contained in the title (in fact, in our TV data set, the majority of product titles do not contain a product code). Also, our proposed methods are fully automated, while in [6] the authors assume a manually generated list of regular expressions that capture knowledge on the syntactical structure of product codes. The methods in [6] use only information from the product titles, while our methods also use information from the product attributes, which grants us a much larger amount of information to use for duplicate detection.

2.3 The Learnable String Similarity method

In [2], methods similar to one of our proposed methods are presented. These methods use textual similarity measures, much like our first proposed method. One of their methods uses an extended variant of learnable string edit distance [9]; the other uses a vector-space based method with a support vector machine [3] for training. These methods are applied to databases and they can identify both duplicate fields and duplicate records (this corresponds with detecting duplicate attributes and products, respectively). However, to perform duplicate detection, these methods require the information in all records to be stored in the same way, i.e., the names of the fields are required to be the same for all records. Our proposed methods do not have this strict requirement. In fact, in the data set that we use, product attribute keys (which correspond to field names) from one Web shop are often different from the corresponding keys from the other Web shop. This is an important problem on the Web, which is addressed by our proposed methods, but not by the database methods from [2].

2.4 Filtering methods

Unlike the other approaches presented here, the main objective of the methods discussed in [13] is not to increase the quality of duplicate detection, but rather to increase the speed and scalability, so where other methods focus on increasing the effectiveness of duplicate detection, these methods focus on increasing the efficiency.

While assessing if two records are duplicates, most methods for duplicate detection compare each field from the first record to each field of the second record. On large data sets, this can cause the execution times to become very long. The focus of [13] is on shortening these execution times by reducing the amount of data that is considered. This is done by first *canonicalizing* each record: ordering the tokens (fields) within the record according to some global ordering. Then during the duplicate detection phase it is sufficient to only consider part of each of the tokens, for example the prefix: the first p tokens. If there is sufficient overlap in these parts of the two records, the two records are taken as candidates to be duplicates; if there is not sufficient overlap, the algorithm no longer needs to consider this record pair as potential duplicates. For duplicate candidates, a similarity measure is used to determine which of the candidate pairs are classified as duplicates.

Because the previously described algorithm is purely focused on improving the efficiency of duplicate detection, while the objective of our proposed methods is to improve the effectiveness of duplicate detection, we have not used this approach in the implementation of our methods. Nevertheless, improving the efficiency of our algorithm, possibly inspired by such an approach, is part of future work.

3. PRODUCT DUPLICATE DETECTION

The proposed methods build on the title model words method [11] and also use information other than title product information to improve the ability of the algorithm to detect duplicates. For all methods, we do not allow duplicate products from the same Web shop, since we assume that Web shops do not list the same product more than once. All product attributes are stored in key-value pairs (KVP's). An example of a key-value pair is: ('Weight', '16.9 lbs.'). Our algorithm is based on the assumption that we have products coming from two different Web shops. However, our approach remains generalizable, as the case with n Web shops can be reduced to $n-1$ applications of the two Web shops case. Furthermore, we assume that the product descriptions are using the same units of measurement, as the integration of different units of measurement is not the focus of this paper. The attribute distance method is presented in Section 3.1. The next section, Section 3.2, discusses the extended model words method.

3.1 The Attribute Distance Method

Our first proposed method uses model words to find similar product names, and distance measures to check matching attributes to detect duplicates. The pseudocode for the attribute distance method is given in Algorithm 1. It starts by assigning each product from the first Web shop to its own cluster, in order to prevent products from being clustered with products from the same Web shop, as mentioned at the beginning of Section 3. Subsequently, the algorithm loops over every product that is not from the first Web shop,

Algorithm 1 Attribute distance method

```
Require: The input: Sets $A$ and $B$ contain all products from two Web shops
Require: $T_{dist}$ is the threshold below which the distance between the current product and the best matching product is required to be for those two products to be considered duplicates
Require: $T_{key}$ is the distance threshold below which two keys are assumed to be equal
Require: $T_{match}$ is the threshold above which the number of key matches between two products is considered as a possible match
Require: $calcDist(q, r, dist_m)$ calculates the distance between strings q and r using distance measure $dist_m$
Require: clusterFull(b,j) returns true if cluster j already contains a product from the same Web shop as product b; otherwise, returns false
Require: key(q) returns the key from key-value pair (KVP) q; value(q) returns the value from KVP q
Require: matchingTitle(b,j) uses model words to check if the title of the current product (b) matches the title of a product in cluster j (using the method from [11]); if so, returns true; otherwise, returns false
1: Assign each product from the first Web shop (set A) to its own cluster, obtaining a set of clusters J
2: for all $b \in B$ do
3:   shortestDistance = $\infty$
4:   sameTitle = false
5:   for all $j \in J$ do
6:     if not clusterFull(j) then
7:       if matchingTitle(b,j) then
8:         sameTitle = true
9:         $cluster_{best} = j$
10:      else
11:        totalDistance = 0
12:        distance = $\infty$
13:        matchCount = 0
14:        for all KVP's q in b do
15:          for all KVP's r in j do
16:            $key_d = calcDist(key(q), key(r), dist_m)$
17:            if $key_d < T_{key}$ then
18:              $totalDistance = totalDistance + calcDist(value(q), value(r), dist_m)^2$
19:              $matchCount = matchCount + 1$
20:            end if
21:          end for
22:        end for
23:        if $matchCount > 0$ then
24:          $distance = \frac{\sqrt{totalDistance}}{2 * matchCount}$
25:        end if
26:        if $distance < shortestDistance$ & $matchCount > T_{match}$ then
27:          shortestDistance = distance
28:          $cluster_{best} = j$
29:        end if
30:      end if
31:    end if
32:  end for
33:  if sameTitle then
34:    Add current product to cluster $cluster_{best}$
35:  else if $shortestDistance < T_{dist}$ then
36:    Add current product to cluster $cluster_{best}$
37:  else
38:    Assign current product to a new cluster in J
39:  end if
40: end for
41: return J
```

assigning every product to a cluster with products that are found to be its duplicates, if any.

To find an appropriate cluster for a product, the algorithm loops over all clusters; for each product in these clusters, the algorithm first checks if the cluster is not 'full' which in this case means that it already contains a product from this second Web shop. If the cluster is full, this cluster is not considered anymore. After this, another check is performed, this time to assess if the title of the clustered product matches the title of the product we currently want to cluster, using the title model words method as in [11]. If the titles match, the current product is clustered with the clustered product with a matching title.

The attribute distance method extends the title model words method as follows. If no product with a matching title is found, the attribute distance method uses the information contained in the product attributes (stored as KVP's) to detect duplicates. For each combination of KVP's, it first calculates the distance between the two keys, if this distance is below a predetermined threshold (T_{key}), it continues by calculating the distance between the values of these two KVP's with matching keys. This distance is then squared. All squared distances between attribute values with matching keys are added up, after which the square root of this value is taken. This is then divided by two times the number of key matches, so that a pseudo-average attribute distance is obtained, which favors products with a higher amount of key matches.

When the loop is concluded (i.e., when either a matching title has been found or when the algorithm has gone through all clusters without finding a matching title), we decide which cluster the current product will be added to. If a matching title has been found, the current product is clustered with the matching product. Otherwise, the closest clustered product, i.e., the clustered product with the smallest attribute distance to the current product, is considered. If this attribute distance is below the threshold value (T_{dist}), the current product is clustered with this closest product. If the distance is above the threshold value, the conclusion is made that no duplicates of the current product are in the set of clusters, so a new cluster is made, containing only this product.

3.2 The Extended Model Words Method

The second method uses model words, not only to find similar names, but also to find similar attributes to detect duplicates. The pseudocode for the extended model words method is given in Algorithm 2. It starts by assigning each product from the first Web shop to its own cluster. Subsequently, the algorithm loops over every product that is not from the first Web shop, assigning every product to a cluster with products found to be its duplicates, if any. To find an appropriate cluster for a product, the algorithm starts by checking if the cluster is 'full' and if both products have matching titles, in the same way as in the previous method.

When no matching titles are found, the title model words method would conclude that the product has no duplicates and the attribute distance method would use distances between keys and values from the KVP's to detect duplicates. The extended model words method uses a different approach when no matching titles are found. The algorithm takes all model words from the attribute values of each clustered product and calculates the percentage of matching model words between that product and the product that is being clustered. Unlike the attribute distance method, this method does not use the keys, it uses only the values, taking all model words from those. Please note that we relax the model words definition here by allowing not only combinations of alphanumeric characters but also strings or numbers, alone.

The reason for the decision to only use the values from KVP's, is that data from various Web shops can be structured in very different ways; only investigating the values when their corresponding keys match, could (unnecessarily) limit the amount of information from the attributes that can be used to detect duplicates. For example: a certain TV from Bestbuy.com has the KVP: ('Product Weight', '19.1lbs. with stand (16.9lbs. without)').

Newegg.com also has information about this TV, only here, the information is structured in two different KVP's: ('Weight Without Stand', '16.9lbs.') and ('Weight With

Algorithm 2 New model words method

```
Require: The input: Sets $A$ and $B$ contain all products from two Web shops
Require: $T_{match}$ is the threshold above which the matching model words percentage must be for two products to be considered duplicates
Require: obtainModelWords(p) gives all model words from the values of the attributes of product p
Require: clusterFull(b,j) returns true if cluster j already contains a product from the same Web shop as product b; otherwise, returns false
Require: matchingTitle(b,j) uses model words to check if the title of the current product (b) matches the title of a product in cluster j (using the method from [11]); if so, returns true; otherwise, returns false
Require: matchingMWpercentage(C,D) returns the percentage of matching model words from two sets of model words
1: Assign each product from the first Web shop (set A) to its own cluster (set of clusters J)
2: for all $b \in B$ do
3:   $MW_{max} = 0$
4:   $MW = obtainModelWords(b)$
5:   for all $j \in J$ do
6:     if not clusterFull(j) then
7:       if $matchingTitle(b, j)$ then
8:         sameTitle = true
9:         $cluster_{best} = j$
10:      else
11:        $c = obtainModelWords(j)$
12:        $MWperc = matchingMWpercentage(MW, c)$
13:        if $MWperc > MW_{max}$ then
14:          $MW_{max} = matchingMWpercentage$
15:          $cluster_{best} = j$
16:        end if
17:      end if
18:    end if
19:  end for
20:  if sameTitle then
21:    Add current product to cluster $cluster_{best}$
22:  else
23:    if $MW_{max} > T_{match}$ then
24:      Add current product to cluster $cluster_{best}$
25:    end if
26:  else
27:    Assign current product to a new cluster in J
28:  end if
29: end for
30: return J
```

Stand', '19.1lbs.'). In this case, the attribute distance method would gain no information from these KVP's, because the keys do not match. The extended model words method, however, would find two matching model word pairs here (the model word '16.9lbs' and the model word '19.1lbs' would be found in the attributes of both TV's), which could enable this method to detect duplicates where the attribute value method would not be able to do so.

When the loop is concluded (i.e., when either a matching title has been found or when the algorithm has gone through all clusters without finding a matching title), we decide what cluster the current product will be added to. If a matching title has been found, the current product is assigned to the cluster with the matching product. Otherwise, the clustered product with the highest percentage of matching model words is considered. If this percentage is above a threshold value (T_{match}), the current product is clustered with the product with which this best match was found. If the percentage is below the threshold value, the conclusion is made that no duplicates of the current product are in the set of clusters, so a new cluster is made, containing only this product.

4. EVALUATION

In this section the results of the proposed approaches are evaluated. The methods discussed in Section 3 are compared against the basic title model words method, which they expand upon and against each other. To assess the performance of these methods, we use them to detect duplicates in our data set of TV's obtained from Best Buy [1] and Newegg [8] and calculate the F_1-measure, precision, and recall from the experiment results. The data set contains 282 TV's: 200 from Bestbuy.com and 82 from Newegg.com; each TV from Newegg.com has a duplicate in the data from Bestbuy.com. This means there are 82 pairs of duplicate TV's (so 164 TV's belonging to a duplicate pair) and 118 products that do not have a duplicate in the data set. To assess whether or not one method is better than another, we will run the algorithms on 20 random test sets of approximately 10% of all products (ensuring that there is a proportional amount of duplicates in these data sets, to ascertain that these smaller data sets are still representative for the original data set); using the remaining 90% of the data set as the training set each time to determine the method parameters. Then we calculate the F_1-measures and use a Wilcoxon signed rank test [12] to assess whether or not one method significantly outperforms the other. This section starts by evaluating each method separately. The title model words method, the attribute distance method, and the extended model words method are discussed in Sections 4.1, 4.2, and 4.3, respectively. In Section 4.4 all three methods are compared to each other.

4.1 The Title Model Words Method

The title model words method uses two parameters, i.e., α and simThreshold (as described in Section 2.1). Both of these parameters can range from 0 to 1 and both affect how similar two titles have to be for their products to be considered the same. The higher α and simThreshold are, the more similar titles have to be for their products to be clustered together. Runs of the algorithm on the training set with various values (from 0 to 1 with a step size of 0.1) for these two parameters showed that high values (0.8 and 0.9) for both parameters tend to provide the best results; the results can be seen in Table 1.

Table 1: Means and standard deviations of the best values for each parameter over the 20 training sets for the title model words method

	Mean	Standard deviation
α	0.815	0.059
simThreshold	0.845	0.051

A somewhat surprising result is that the F_1-measure is almost always 0 when both α and simThreshold are 0.9, while the best F_1-measures are observed when these parameters take values close to 0.9. The cause of this is that when both parameters are 0.9, the similarity requirement for titles is so strict that no products are clustered together anymore. The title model words algorithm was run on the 20 test sets described earlier. The average value of the F_1-measure over these 20 runs was 0.357. The corresponding average precision and recall are 0.556 and 0.279, respectively.

4.2 The Attribute Distance Method

The attribute distance method uses 6 parameters. The first two parameters are α and simThreshold, the parameters used by the title model words method, which this method expands upon. Like the title model words method, this method performs best with high values for α and simThreshold (0.8 and 0.9). The third parameter, T_{key}, is the threshold below which the distance between two keys must be for them to be assumed equal. Increasing the value of this threshold increases the amount of information from the product attributes that is used, but this also makes it more likely that non-matching keys are considered equal, resulting in inaccurate distances. The best value for this parameter is 0.645 on average. The fourth parameter, T_{match}, is the minimum number of key matches that must be found between two products for them to be allowed as a possible match (if no matching product title is found). This is an integer that has been tested with values ranging from 1 to 5. This range was found to give useful results on a smaller test data set. The best value for this threshold is 1.95 on average, but this can vary greatly, since it has a standard deviation of 1.317.

The fifth parameter, $dist_m$, is the distance measure used to calculate the distance between two strings. Two distance measures were tested, i.e., the Jaro-Winkler distance measure [5] and the cosine distance measure [14]. On average, the cosine distance measure provided better results than the Jaro-Winkler distance measure. Last, T_{dist} is the threshold below which the distance between a product and the best matching product must be for them to be considered duplicates (when no matching title is found). The higher T_{dist} is set, the more clusters containing more than one product (i.e., duplicates) are made and the higher the risk of false positives (products clustered together while they are not duplicates). The best value for T_{dist} was found to be 0.1 when experimenting with values between 0 and 1 with a step size of 0.1. All of these parameter mean values and their standard deviations can also be found in Table 2.

The attribute distance algorithm was run on the 20 test sets described above. The average value of the F_1-measure over these 20 runs was 0.529, which is higher than the average F_1-value for the title model words method, which is

Table 2: Means and standard deviations of the best values for each parameter over the 20 training sets for the attribute distance method

	Mean	Standard deviation
α	0.82	0.062
simThreshold	0.855	0.051
T_{key}	0.645	0.167
T_{match}	1.95	1.317
T_{dist}	0.1	0

0.357. The average precision and recall of the attribute distance method are 0.531 and 0.556, respectively. When comparing these values with the corresponding ones from the title model words method, we observe that the values for the precision of these two methods are not far apart. We observe that the higher F_1-value of the attribute distance measure can be fully attributed to the higher recall.

4.3 The Extended Model Words Method

The extended model words method uses three parameters. The first two are α and simThreshold, the two parameters used by the title model words method, which is the basis for the extended model words method. For the extended model words method, the optimal value for α was found to be 0.9 in all cases, the optimal value for simThreshold is 0.87 on average, as shown in Table 3. The third parameter is T_{match}, which is the threshold above which the percentage of matching model words within the attributes of two products must be for them to be considered duplicates (note that this T_{match} is different from the threshold with the same name which is used in the attribute distance method). The higher this threshold, the stricter the requirement, so a lower T_{match} will provide more clusters. In all runs, the best results were achieved with a T_{match} of 0.3, when experimenting with values between 0 and 1 with a step size of 0.1.

The extended model words algorithm was also run on the 20 test sets described at the beginning of Section 4. The average value of the F_1-measure over these 20 runs was 0.607, which is higher than the corresponding F_1-measure for the other methods. The average values of the precision and recall are 0.637 and 0.597 respectively, so the F_1-value, the average precision and average recall of the extended model words method are all higher than the corresponding values of both other methods.

Table 3: Means and standard deviations of the best values for each parameter over the 20 training sets for the extended model words method

	Mean	Standard deviation
α	0.9	0
simThreshold	0.87	0.047
T_{match}	0.3	0

4.4 Comparison of All Methods

To compare the performance of the three methods, we use the values of the F_1-measure over the aforementioned 20 test sets for each method. The average F_1-values over these 20 test sets are presented in Table 4. The average F_1-value of the attribute distance method is higher than that of the title model words method. The average F_1-value for

Table 4: Average F_1-value, precision and recall over the 20 test sets for each method

Method	Average F_1-measure	Average precision	Average recall
Title model words	0.357	0.556	0.279
Attribute distance	0.529	0.531	0.556
New model words	0.607	0.637	0.597

the extended model words method is higher than those of the other two methods. The difference between the average F_1-value of this method and that of the title model words method is quite large; the difference between the F_1-value of the extended model words and that of the attribute distance method is much smaller.

However, to assess whether or not these differences are significant, we perform Wilcoxon signed rank tests. The results of these tests will allow us to determine if any method significantly outperforms the others. The results of these tests are displayed in Table 5. The p-values resulting from the Wilcoxon signed rank test to compare the title model words method to the attribute distance method are 0.082 and 0.923. This shows that we can not prove a significant difference in performance between these two methods at a 0.05 significance level (although the attribute distance method does outperform the title model words method at a 0.10 significance level). The p-values obtained when comparing the extended model words method to the attribute distance method are 0.285 and 0.727, both clearly not significant. The p-value for the Wilcoxon signed rank test to assess whether or not the extended model words method outperforms the title model words method is 0.002, so the extended model words method has significantly better performance than the title model words method, but does not significantly outperform the attribute distance method.

Table 5: The one-sided p-values for the Wilcoxon signed rank test, calculated to determine whether or not a method outperforms the others ($\mu_{row} < \mu_{column}$)

p-values	Title model words	Attribute distance	New model words
Title model words	X	0.082	0.002
Attribute distance	0.923	X	0.285
New model words	0.999	0.727	X

Table 6 presents the means and standard deviations of the execution times for the three methods. These results were obtained from runs on the 20 test sets with the parameters that were found to be the best on their corresponding training sets. The title model words method clearly has the shortest mean execution time: 109 milliseconds, this was expected since our two proposed methods use the algorithm from the title model words method as their base and extend it. The mean execution time of the extended model words method is longer: 563 ms, this is because the extended model words method considers not only the title, but also the product attribute values. The attribute distance method has the longest mean execution time: 1730 ms, this is because the attribute distance method not only uses product titles and product attribute values, but also product attribute names (keys), which increases the execution time.

Table 6: Means and standard deviations of the execution times (in milliseconds) over the 20 test sets for each method

Method	Mean	Standard deviation
Title model words	109	26
Attribute distance	1730	385
New model words	563	91

5. CONCLUSIONS

In this paper we have proposed new solutions to the problem of product duplicate detection on the Web. We started with the title model words method, a state-of-the-art solution to the problem of product duplicate detection on the Web [11]. We have devised two new methods to solve the problem of duplicate detection, both of which extend the title model words method. The first new method is the attribute distance method, which uses the information in the product attributes, using distance measures to determine if products are duplicates when no matching title is found. The second new method is the extended model words method: this method not only uses model words to detect matching titles, but also searches for matching model words in the product attribute values to detect duplicates.

For the evaluation we have used a real-world data set of TV's from two existing Web shops, which contains duplicates as well as non-duplicate products. The results show that the extended model words method significantly outperforms the existing state-of-the-art title model words method with respect to the F_1-measure. With respect to the F_1-measure, the attribute distance method does not outperform the extended model words method, yet it does outperform the title model words method, but not significantly.

Despite the fact that the attribute distance method not only uses title model words, but also additional information from the product attributes, we have found that the attribute distance method does not significantly outperform the title model words method. The likely explanation of this result is that the way in which data is represented differs greatly between the two Web shops in our data set. If a key from one Web shop does not match the corresponding key from the other shop (which occurs quite frequently in our data set), the information that could be gained from the value belonging to these keys remains unused. Because of this, a great deal of information can be lost, leaving only a small, sometimes insufficient amount of information to use for duplicate detection. This has a negative effect on the performance of the attribute distance method.

The fact that major differences in keys between one source and the other exist, was our main motivation to implement the extended model words method. This method disregards the attribute keys and only analyses the attribute values, thereby avoiding the problem that affects the attribute distance method. Although the attribute distance method does not significantly outperform the title model words method, this paper does provide a new method that outperforms the existing state-of-the-art title model words method.

As future work we would like to experiment with additional string distance measures such as the Jaccard [4] or Levenshtein [7] distance measures. Also we would like to investigate an ontology-based approach for duplicate detection where domain background knowledge can be used to aid duplicate detection. For instance, knowing the range of a property can support the value matching step in our first proposed duplicate detection algorithm.

Another possible approach that we would like to explore is a hybrid one, combining elements from the attribute distance method and the extended model words method; for instance by using distance measures to find similar keys, as in the attribute distance method, and then finding matching model words from the attribute values. Also we would like to improve the efficiency of the proposed algorithms using optimization methods from existing work [13].

Last, we would like to experiment with other distance-based methods. An example of this could be the use of the cosine similarity on TF-IDF vectors that are obtained from the attribute values of a product. At a later stage, one could also experiment with feature selection methods.

6. REFERENCES

[1] Best buy. `http://www.bestbuy.com`.

[2] M. Bilenko and R. Mooney. Adaptive Duplicate Detection Using Learnable String Similarity Measures. In *Proceedings of the 9th International Conference on Knowledge Discovery and Data Mining (KDD 2003)*, pages 39–48. ACM, 2003.

[3] C. Cortes and V. Vapnik. Support-Vector Networks. *Machine Learning*, 20(3):273–297, 1995.

[4] P. Jaccard. Distribution de la Flore Alpine dans le Bassin des Dranses et dans Quelques Régions Voisines. *Bulletin de la Société Vaudoise des Sciences Naturelles*, 37:241–272, 1901.

[5] M. Jaro. Advances in record-linkage methodology as applied to matching the 1985 census of tampa, florida. *Journal of the American Statistical Association*, 84(406):414–420, 1989.

[6] H. Köpcke, A. Thor, S. Thomas, and E. Rahm. Adaptive Duplicate Detection Using Learnable String Similarity Measures. In *Proceedings of the 15th International Conference on Extending Database Technology (EDBT 2012)*, pages 545–550. ACM, 2012.

[7] V. I. Levenshtein. Binary Codes Capable of Correction Deletions, Insertions, and Reversals. *Soviet Physics Doklady*, 10(8):707–710, 1966.

[8] Newegg. `http://www.newegg.com`.

[9] E. Ristad and P. Yianilos. Learning string-edit distance. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 20(5):522–532, 1998.

[10] G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval*. McGraw Hill, 1983.

[11] D. Vandic, J. W. J. van Dam, and F. Frasincar. Faceted Product Search Powered by the Semantic Web. *Decision Support Systems*, 53(3):425–437, 2012.

[12] F. Wilcoxon. Individual Comparisons by Ranking Methods. *Biometrics Bulletin*, 1(6):80–83, 1945.

[13] C. Xiao, W. Wang, X. Lin, J. X. Yu, and G. Wang. Efficient similarity joins for near-duplicate detection. *Transactions on Database Systems (TODS)*, 36(3):15, 2011.

[14] M. Zaki. *Introduction to Data Mining*. Springer, 2003.