A Hybrid Model Words-Driven Approach for Web Product Duplicate Detection

Marnix de Bakker, Flavius Frasincar, and Damir Vandic

Erasmus University Rotterdam
PO Box 1738, NL-3000 DR
Rotterdam, the Netherlands
marnixdebakker@zeelandnet.nl, {frasincar,vandic}@ese.eur.nl

Abstract. The detection of product duplicates is one of the challenges that Web shop aggregators are currently facing. In this paper, we focus on solving the problem of product duplicate detection on the Web. Our proposed method extends a state-of-the-art solution that uses the model words in product titles to find duplicate products. First, we employ the aforementioned algorithm in order to find matching product titles. If no matching title is found, our method continues by computing similarities between the two product descriptions. These similarities are based on the product attribute keys and on the product attribute values. Furthermore, instead of only extracting model words from the title, our method also extracts model words from the product attribute values. Based on our experimental results on real-world data gathered from two existing Web shops, we show that the proposed method, in terms of F_1-measure, significantly outperforms the existing state-of-the-art title model words method and the well-known TF-IDF method.

Keywords: entity resolution, model words, attribute distance, products

1 Introduction

In recent years, the amount of products sold through on-line shops has grown rapidly, as a result of the convenience the Web offers to consumers [9]. Although many Web shops sell the same products, the information about these products can differ greatly from Web shop to Web shop. For example, one Web shop might only have information about the weight and accessories, while others might also have information about the manufacturer and the dimensions of the product. Some Web shops only present general information, while others show a vast amount of specific information on a certain product. As these examples show, rather than being in one place, product information is often distributed across many Web shops.

The product information distribution on the Web forces consumers to spend considerable effort in order to find all their desired product information. Consumers can benefit greatly if data from different Web shops could be aggregated to form a more complete set of product information. However, aggregating

product data from different Web shops is a difficult task. The vast amount of products, product information, and Web shops makes it infeasible to perform product integration manually, therefore, this process has to be automated. To automatically aggregate data from various websites, it is necessary to perform *duplicate detection*, i.e., to determine, using data from Web shops, which product descriptions refer to the very same product. The scope of this paper does not encompass other product integration challenges such as product merging and product schema alignment.

Due to the diversity of names used for products by Web shops, duplicate detection is not as trivial as finding product names that exactly match. For instance, Bestbuy.com names a TV from LG as 'LG - 32" Class / LED / 720p / 60Hz / HDTV', while Newegg.com gives the very same product the name 'LG 32" Class (31.5" Measured) 720p 60Hz LED-LCD HDTV 32LV2400'. This example shows that in order to be able to determine whether two product descriptions (or titles) refer to the same product, one has to carefully analyze the syntax and semantics of the description. This makes product duplicate detection a challenging problem.

Several methods to perform duplicate detection are presented in this paper. The first one is the title model words method [10], which is a state-of-the-art Web product duplicate detection method that extracts the so-called *model words* from product names and compares these in order to detect duplicates. Model words are words that contain both numeric and alphabetic/punctuation characters. These type of words often give valuable information for the process of duplicate detection, as they usually represent some unique aspect of a product (e.g., a product code, the model number, etc.).

The method proposed in this paper extends the title model words method and uses it as a baseline. We also compare the results of our proposed method against the results of a method that uses the well-known TF-IDF approach [8] for duplicate detection. Our proposed method first uses the model words method for product names, then uses similarities between product attributes, and subsequently employs model words for the values of product properties.

The paper is organized as follows. In Sect. 2, we discuss related work on (product) duplicate detection. Sect. 3 discusses the proposed method for duplicate detection. We evaluate our method against the state-of-the-art title model words method and a TF-IDF-based method in Sect. 4. Last, Sect. 5 concludes the paper and suggests possible future work.

2 Related Work

In literature, we can find several duplicate detection algorithms. Some of these algorithms are used in our experiments as baseline methods, against which we benchmark our approach. The state-of-the-art method described in [10] is the first baseline method that we consider. This approach uses model words extracted from the names of products to perform product duplicate detection on the Web. For this purpose, an algorithm is used that starts by calculating the

word-based cosine similarity between two product names. If this similarity exceeds a predefined threshold α, the algorithm stops and the two products are considered duplicates. If the similarity is lower than the threshold, the algorithm continues by extracting model words from the two product names. Subsequently, the algorithm determines if it can immediately conclude that the products are not duplicates by looking for specific model word pairs (one from each product description). For these specific word pairs, the non-numeric parts have to be approximately the same (based on Levenshtein distance), but the numeric parts should be different. Finding such a model word pair gives support for the conclusion that we are dealing with two different products. Consider the two product names 'LG - 32" Class / LED / 720p / 60Hz / HDTV' and 'LG - 40" Class / LED / 720p / 60Hz / HDTV'. In this example we identified the pair of model words *32"* and *40"*. Based on the fact that their non-numeric parts are the same and their numerical parts are not, we conclude that these two names represent two different products.

If no word pairs meeting the previously described condition are found, the algorithm proceeds by computing a new similarity between the two product names. This similarity value is a weighted average of the average Levenshtein similarity between the two sets of words in the product titles and the word-based cosine similarity between the product names (product titles). After this similarity is computed, the algorithm checks if there are model word pairs that are likely to be the same, i.e., model word pairs that have approximately the same non-numeric parts and the same numeric parts. If there are such pairs, the aforementioned cosine/Levenshtein similarity value is updated with the model word pair similarity. After completing this step, the final step is to check if the final similarity value exceeds a predetermined threshold (β). If that is the case, the products are considered to be duplicates. The method proposed in this paper extends this method, using additional information from the product attributes to improve the duplicate detection process. Furthermore, in our approach we also use the similarity between pairs of product attribute names and pairs of their corresponding values. As a specific feature, it uses model words not only in the title, but also in the product attribute values.

The second baseline method that we use in our experiments is based on the the Term Frequency-Inverse Document Frequency (TF-IDF) method [8]. In such an approach, the term frequency is defined as the number of times that a word occurs in the attribute values of a product. Here, IDF is defined as the logarithm of the total number of products divided by the number of products in which the word occurs. The TF-IDF method uses the parameter δ, which is the minimum TF-IDF value in order for two products to be considered the same. These type of duplicate detection approaches are often used in offline duplicate detection areas [5], e.g, the approach presented in [3].

There are several other works that we can find in literature that aim for goals similar to ours. In [6], a method for duplicate detection using (model) words in the product titles is presented. The algorithm for this method detects duplicates using the similarity between product titles. An important aspect of

this method is that it extracts product codes from the product titles. These codes are numbers that are designated to products by their manufacturer and are unique for each product. The first step for extracting the product code from a product title is to remove common features such as weight, color, and dimensions from the title. Subsequently, the algorithm removes stop words and words that appear frequently in product offers of various manufacturers from the product title. The next step is to generate candidates, which often consist of up to three model words, for product codes. For this task, a manually created list of regular expressions that capture knowledge on the syntactical structure of product codes is used. The final step is based on Web verification to check the correctness of the extracted candidates. For this purpose, a query is submitted to a Web search engine for a candidate; the fraction of the results containing the corresponding manufacturer, with respect to all results, is used to check the correctness of each candidate.

The method presented in [6] is less flexible than our proposed method, since our method does not need product codes to be contained in the title (actually, the majority of product titles in the TV data set used in our experiments does not contain a product code). Also, our proposed method is 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. Like the title model words method, the method from [6] only uses information from the product titles, while our proposed method also uses information from the product attributes. By employing this extra information, we are better equipped to solve the problem of duplicate detection on the Web.

In literature we can also find entity resolution methods that focus on the textual similarity measures. For example, one of the methods proposed by the authors of [2], employs an extended variant of the learnable string edit distance. We have also encountered approaches where a vector-space based method with a support vector machine [4] is used for the training. These methods are applied to databases and they can identify both duplicate records and duplicate fields, which corresponds to detecting duplicate products and attributes, respectively, in our setup. A disadvantage of these methods is that they 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 method does not have this strict requirement. In fact, between different Web shops in the data set we use, there is a great deal of variation in product attribute keys (which correspond to field names) that represent the same information. This is a critical issue on the Web; it is addressed by our proposed method, but not by the database methods from [2].

Last, there are approaches that focus on the scalability of the duplicate detection process, instead of the effectiveness. This is the case for the work done in [12]. These methods focus on increasing the efficiency, while other methods focus on improving the effectiveness of duplicate detection. Most methods for duplicate detection compare each field from the first record to each field of the second record during the process of assessing if two records are duplicates. This

can be a problem when working with large datasets, because it can cause the execution times to become very large. By reducing the amount of data that is considered, the methods in [12] aim to lower these execution times. The first step towards this objective is *canonicalizing* each record: ordering the tokens (fields) within the record according to some global ordering. After this is done, during the duplicate detection phase, it suffices to only consider part of each of the tokens, for example the suffix: the last p tokens. The two records are taken as candidates to be duplicates if there is sufficient overlap in these parts of the two records. If that is not the case, then it is no longer necessary for the algorithm to consider this record pair as potential duplicates. A similarity measure is used to determine which of the candidate pairs are classified as duplicates. As the focus of this algorithm is on improving the efficiency of duplicate detection, while the objective of our proposed method is to improve the effectiveness of duplicate detection, we have not used this approach as a reference method. Nevertheless, optimizing the efficiency of our algorithm, as suggested by this approach, is part of our future work.

3 The Hybrid Similarity Method

As previously mentioned, our proposed method, the hybrid similarity method, extends the title model words method [10]. While the title model words method only uses information from the product titles, our method also exploits information from the product attributes. Intuitively, this extra information should lead to duplicate detection results superior to those of the title model words method. All product attributes are stored in key-value pairs (KVP's). An example of a key-value pair is: ('Weight', '20.5 lbs.').

We assume that there are no duplicate products from the same Web shop, an assumption that is also made for the other methods. This assumption is based on the belief that Web shops do not list their products more than once. Furthermore, data integration within a Web shop is out of the scope of this paper, as the context in which the duplicate detection is performed is not the same (i.e., in these cases, there are varying details about the structure of product descriptions).

The pseudocode of the hybrid similarity method applied to two different Web shops is given in Algorithm 1. The method starts by assigning each product from the first Web shop to its own cluster, in order to prevent products from being grouped with products from the same Web shop. After this, it loops over every product from the second Web shop and searches for duplicates among the clustered products. If such a duplicate is found, the product is clustered with the found duplicate.

To find an appropriate cluster for a product, the algorithm first loops over all clusters. For each product in these clusters, the algorithm then checks if the cluster is not 'full', which in this case means that it already contains a product from the second Web shop. If the cluster is full, it is not considered any more. If the cluster is not full, the title model words method (the baseline) is used to check

if the title of the current product matches with the title of the clustered product. If such a title match is found, the two products are considered duplicates and are clustered. The algorithm then continues with the next unclustered product. If no title match is found, the algorithm proceeds to use information from the product attributes to detect if the two products are duplicates.

To use the information from the product attributes, a similarity measure is constructed. This measure is based on two different methods. The first method loops over all combinations of KVP's from both products. If it finds a pair of matching keys, the similarity between the corresponding attribute values is calculated. The similarity measure can be any text similarity measure; in this paper we have performed tests using the cosine similarity and the Jaro-Winkler similarity measure. When the loop has ended, the average of all these value similarities is computed. This average similarity forms the first part of the similarity measure.

The second part of the similarity measure starts by analyzing all KVP's in which no matching keys were found. Then, it extracts all model words from the values of these attributes and combines them in two sets (one for each product). Here, we use a broader definition of model words, i.e., a definition that also includes purely numeric words in addition to the mixed numeric/non-numeric words. Subsequently, the percentage of matching model words between the two sets is calculated. This matching model words percentage forms the second part of the similarity measure.

We should stress that for this part of similarity measure we only use the product attribute values and disregards the keys. The reason for this is that data from various Web shops can be structured in very different ways; only investigating the values when their corresponding keys match, could (unneces-

Algorithm 1 Hybrid similarity method

Require: The input: Sets A and B contain all products from two Web shops
Require: γ is the threshold similarity for two keys to be considered equal, δ is the product distance threshold that determines whether two products are identified as duplicates
Require: calcSim$(q, r, measure)$ calculates the similarity between strings q and r using similarity measure $measure$
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 clustered product j (using the method from [10]); if so, returns true; otherwise, returns false
Require: exMW(p) returns all model words from the values of the attributes (except for those where a key match was found) from product p
Require: mw(C, D) returns the percentage of matching model words from two sets of model words

```
Assign each product from the first Web shop (set A) to its own cluster, obtaining a set of clusters J
for all b ∈ B do
    bestSimilarity = −1
    for all j ∈ J do
        if not clusterFull(j) then
            if matchingTitle(b, j) then
                Assign product b to cluster j
            else
                sim = 0
                avgSim = 0
                m = 0 {number of matches}
                for all KVP's q in b do
                    for all KVP's r in j do
                        keySim = calcSim(key(q), key(r), measure)
                        if keySim > γ then
                            sim = sim + calcSim(value(q), value(r), measure)
                            m = m + 1
                        end if
                    end for
                end for
                if m > 0 then
                    avgSim = sim / m
                end if
                mwPerc = mw(exMW(b), exMW(j))
                hSim = θ * avgSim + (1 − θ) * mwPerc
                if hSim > bSim then
                    bSim = hSim
                    bestCluster = j
                end if
            end if
        end if
    end for
    if bSim > δ then
        Add current product to cluster bestCluster
    else
        Assign current product to a new cluster in J
    end if
end for
return J
```

sarily) limit the amount of information from the attributes that can be used to detect duplicates. For example, a particular TV from Bestbuy.com has the KVP: ('Product Weight', '19.1lbs. with stand (16.9lbs. without)'). Newegg.com has information about this TV as well, only here, the information is structured in two different KVP's: ('Weight Without Stand', '16.9lbs.') and ('Weight With Stand', '19.1lbs.'). In this case, the first part of the similarity measure would

gain no information from these KVP's, because the keys do not match. The second part, however, would find two matching model words here (the model word '16.9lbs' and the model word '19.1lbs'), which would aid the algorithm to determine whether the two names represent the same product.

The last element required for the similarity measure is θ, which is the weight given to the first part of the similarity measure. This weight is based on the number of key matches: it is calculated by dividing the number of key matches by the number of KVP's in the product with the smallest amount of KVP's. Intuitively, the higher the number of matching keys, the greater the importance of the key-based similarity measure. The formula for the similarity measure is defined as:

$$hybridSimilarity = \theta * avgSim + (1 - \theta) * mwPerc \quad (1)$$

where $avgSim$ is the average similarity based on the matching keys (the first part) and $mwPerc$ is the matching model words percentage (the second part).

When the algorithm has looped over all clustered products, it has to decide to which cluster the current product will be added. The algorithm identifies the closest clustered product, i.e., the clustered product for which the value of the hybrid similarity measure is the highest. If this value is higher than the threshold value δ, the current product is clustered with this closest product. If this value is less than δ, the algorithm concludes that the product has no duplicates and a new cluster that contains only this product is created.

4 Evaluation

In this section, the results of the investigated approaches are evaluated. Our proposed method is compared against the basic title model words method and the TF-IDF method. To assess the performance of these methods, we use them to detect duplicates in a data set of TV's that is obtained from Best Buy [1] and Newegg [7]. As evaluation measures we use 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 run the algorithms on 20 random test sets of approximately 10% of all products. We make sure that there is a proportional (with respect to size) amount of duplicates in these datasets, to ascertain that these smaller datasets are still representative. We have used the remaining 90% of each data set as the training set to determine the method parameters. Then, we calculate the F_1-measures and use a Wilcoxon signed rank test [11] 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 TF-IDF method, and the hybrid similarity method are discussed in Sect. 4.1, 4.2, and 4.3, respectively. In Sect. 4.4 we compare the results from all three methods.

4.1 The Title Model Words Method

The title model words method uses the two parameters α and β. Both of these parameters are thresholds that 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 β are, the more similar titles have to be for their products to be clustered together. Training the algorithm on the 20 training sets showed that high values (0.8 and 0.9) for both parameters tend to provide the best results. The training was performed by letting each parameter range from 0 to 1 with steps of 0.1. Table 1 summarizes the findings of these runs. A somewhat surprising result is that the F_1-measure is almost always 0 when both α and β are 0.9, while the best F_1-measures are observed when these parameters take values close to, but smaller than 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 any more.

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
β	0.845	0.051

The title model words algorithm was run on the 20 test sets described earlier, with the corresponding (training set) optimized parameters. 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 TF-IDF method

The TF-IDF method has only the parameter δ, which represents the minimum TF-IDF value for two products to be identified as equal. The TF-IDF algorithm was trained using values ranging from 0.1 to 0.9 for the parameter δ (with steps of 0.1). In all training sets, the best value for δ was found to be 0.1.

The TF-IDF method was also run on the 20 test sets described previously. The average value of the F_1-measure was 0.201, the average precision was 0.433, and the average recall was 0.133. When comparing these results to the corresponding value from the title model words methods, we notice that they are all lower than those of the title model words method.

4.3 The Hybrid Similarity Method

The hybrid similarity method uses 5 parameters. The first two parameters are α and β. These are used in the process of finding matching titles and as such, they are the same as in the title model words method. For these parameters,

we have used the same ranges as before for the training process (0.1 to 0.9 with a 0.1 step size). Similar to the title model words method, the hybrid similarity method achieves the best results with high values for α and β: in the 20 test runs, none of the values for these two parameters were lower than 0.8, in fact, the best values for α proved to be 0.9 in all test runs. The third parameter is γ, the threshold that determines when two keys are considered equal. Like the other thresholds, the optimal value for this parameter is also quite high: 0.825 on average. The fourth parameter is the used similarity measure, which determines the similarity between a pair of keys or values. In our tests, we have used the cosine similarity measure and the Jaro-Winkler similarity value. The reported results were obtained using the cosine similarity, as the results obtained using the cosine similarity value were better than those obtained with the Jaro-Winkler similarity measure. The fifth and final parameter is δ, which is the threshold similarity that determines whether two products are considered equal. Like with the other parameters, the training was performed by letting the values range from 0.1 to 0.9 with a step size of 0.1. The best values of δ from the 20 test runs were either 0.2 or 0.3. The average of these optimal values was 0.276. The means and standard deviations of the four numerical parameters of this method are shown in Table 2.

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

	Mean	Standard deviation
α	0.9	0
β	0.88	0.041
γ	0.825	0.125
δ	0.267	0.044

The average F_1-measure obtained by running the hybrid similarity algorithm over the previously described 20 test sets was 0.656. The corresponding average values of the precision and recall were 0.741 and 0.647, respectively. All three of these performance measures are clearly higher than the corresponding values for the other two methods. In the next section we analyze whether the measured differences are statistically significant.

4.4 Comparison of All Methods

The main metric we use to compare the performance of the three considered methods is the F_1-measure. The precision and recall will also be stated. As mentioned previously, these tests are performed on the 20 test sets mentioned before. The average values of these three performance measures are shown in Table 3. This table shows that all three performance measures of the title model words method are higher than those of the TF-IDF method. The hybrid similar-

ity method has higher average values than both other methods for all of these performance measures.

Table 3. 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
TF-IDF	0.201	0.433	0.133
Hybrid Similarity	0.656	0.741	0.647

Wilcoxon signed rank tests are performed to check whether or not these differences are significant. For these tests, we use a significance level of 0.05. Table 4 shows the p-values for the performed comparisons, along with the corresponding hypothesis. What stands out about these test results is that several of these p-values are equal to zero or one, indicating that there are some very clear significance results. For instance, the tests to determine whether or not the title model words method and the TF-IDF method outperform the hybrid similarity method both result in a p-value of one. This means that these two methods do not significantly outperform the hybrid similarity method. The table also shows that the TF-IDF method does not significantly outperform the title model words method, with a p-value of 0.989. The title model words method does significantly outperform the TF-IDF method, with a p-value of 0.049. The table shows very clearly that the hybrid similarity method outperforms the other two methods, as both p-values are 0 (these values are rounded to three decimals; without rounding, these p-values are 0.0001 for the title model words method and 0.000002 for the TF-IDF method).

Table 4. The one-sided p-values for the Wilcoxon signed rank test, calculated to determine whether or not a method outperforms the others ($H_0 : \mu_{row} = \mu_{column}, H_A : \mu_{row} < \mu_{column}$)

p-values	Title model words	TF-IDF	Hybrid similarity
Title model words	X	0.989	0
TF-IDF	0.049	X	0
Hybrid similarity	1	1	X

The means and standard deviations of the execution times of the three methods are presented in Table 5. To obtain these results, the three methods were run on the 20 test sets, using the parameters which provided the best performance (in terms of F_1-measure) on the training set. The title model words is the fastest, with an average of 109 ms. This finding is in line with our expectations, since this method only uses the product titles. As a result, it does not spend any

Table 5. 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
TF-IDF	233	34
Hybrid similarity	3108	613

time on the product attributes. Also, the hybrid similarity method extends this method, so the execution times of that method can not be smaller than those of the title model words method. The TF-IDF method has larger execution times than the title model words method, with an average of 233 ms. However, the hybrid similarity method has by far the largest execution times: 3108 ms on average. The fact that the execution times of this method are so much larger than those of the TF-IDF method are most likely caused by the fact that the TF-IDF method only looks at the product attribute values, while the hybrid similarity method uses the product title, the product attribute names, and the product attribute values.

5 Conclusions and Future Work

This paper proposes a new hybrid similarity method in order to solve the challenging problem of product duplicate detection on the Web. Our approach extends the state-of-the-art title model words method for duplicate detection, presented in [10]. The title model words algorithm only uses information from the product titles, but our hybrid similarity method also exploits product attribute information. In this way, the detection of duplicate products is made possible for the cases where the title model words method may have missed some of the true model words. In our proposed method, the product attributes are used to construct a weighted similarity, consisting of two parts. The first part is obtained by first detecting pairs of matching product keys from the two products and then computing the average similarity between the attribute values corresponding to these matching keys. For the second part we extract the model words from all key-value pairs where no matching keys were found and use the percentage of matching model words from the two products as the similarity. We compare our proposed method against both the title model words method and the well-known TF-IDF method.

To assess the performance of the three duplicate detection methods, we use a real-world dataset of televisions from two Web shops. This dataset contains both duplicate and non-duplicate products. We use the F_1-measure to assess the performance of the three methods. From the results that we obtained, we can conclude that the title model words method significantly outperforms the TF-IDF method. More importantly, we can conclude that the hybrid similarity method significantly outperforms the other two methods. The reason for this result is that the hybrid similarity method uses more information, i.e., the title

model words method only uses information from the product titles, the TF-IDF method only uses information from the product attribute values, but the hybrid similarity method uses information from the product titles, the product attribute values, and the product attribute keys.

As future work we would like to assess the performance of the hybrid similarity method using additional string distance measures such as the Levenshtein or Jaccard distance measures. Another research topic that we would like to pursue, would be to use an ontology-based approach for duplicate detection, in which duplicate detection can be aided by domain background knowledge. For instance, knowledge about the range of a property can support the key matching step. Also, we would like to investigate the possibility of improving the efficiency of our proposed method using optimization methods from existing work [12].

Acknowledgments

The authors of this paper are partially supported by an NWO Mozaiek scholarship (project 017.007.142) and the Dutch national program COMMIT.

References

1. Best Buy Co., Inc.: `http://www.bestbuy.com`
2. Bilenko, M., Mooney, R.: Adaptive Duplicate Detection Using Learnable String Similarity Measures. In: Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2003). pp. 39–48 (2003)
3. Bilenko, M., Mooney, R.: Adaptive Name Matching in Information Integration. IEEE Intelligent Systems 18(5), 16–23 (2003)
4. Cortes, C., Vapnik, V.: Support-Vector Networks. Machine Learning 20(3), 273–297 (1995)
5. Elmagarmid, A., Ipeirotis, P., Verykios, V.: Duplicate Record Detection: A Survey. IEEE Transactions on Knowledge and Data Engineering 19(1), 1–16 (2007)
6. Köpcke, H., Thor, A., Thomas, S., Rahm, E.: Tailoring Entity Resolution for Matching Product Offers. In: Proceedings of the 15th International Conference on Extending Database Technology (EDBT 2012). pp. 545–550 (2012)
7. Newegg Inc.: `http://www.newegg.com`
8. Salton, G., Fox, E., Wu, H.: Extended Boolean Information Retrieval. Communications of the ACM 26(11), 1022–1036 (1983)
9. Thomas, I., Davie, W., Weidenhamer, D.: Quarterly Retail e-commerce Sales 3rd Quarter 2012. U.S. Census Bureau News (2012)
10. Vandic, D., van Dam, J., Frasincar, F.: Faceted Product Search Powered by the Semantic Web. Decision Support Systems 53(3), 425–437 (2012)
11. Wilcoxon, F.: Individual Comparisons by Ranking Methods. Biometrics Bulletin 1(6), 80–83 (1945)
12. Xiao, C., Wang, W., Lin, X., Yu, J.X., Wang, G.: Efficient Similarity Joins for Near Duplicate Detection. ACM Transactions on Database Systems (TODS) 36(3), A:1–A:40 (2011)