An LSH-Based Model-Words-Driven Product Duplicate Detection Method

Aron Hartveld, Max van Keulen, Diederik Mathol, Thomas van Noort
Thomas Plaatsman, Flavius Frasincar, and Kim Schouten

Erasmus University Rotterdam
PO Box 1738, NL-3000 DR
Rotterdam, the Netherlands
{344544ah, 360314mk, 361103dm, 346877tn, 342789tp}@student.eur.nl
{frasincar, schouten}@ese.eur.nl

Abstract. The online shopping market is growing rapidly in the 21st century, leading to a huge amount of duplicate products being sold online. An important component for aggregating online products is duplicate detection, although this is a time consuming process. In this paper, we focus on reducing the amount of possible duplicates that can be used as an input for the Multi-component Similarity Method (MSM), a state-of-the-art duplicate detection solution. To find the candidate pairs, Locality Sensitive Hashing (LSH) is employed. A previously proposed LSH-based algorithm makes use of binary vectors based on the model words in the product titles. This paper proposes several extensions to this, by performing advanced data cleaning and additionally using information from the key-value pairs. Compared to MSM, the MSMP+ method proposed in this paper leads to a minor reduction by 6% in the F_1-measure whilst reducing the number of needed computations by 95%.

Keywords: Duplicate detection, min-hashing, locality sensitive hashing, Web shop products, Multi-component Similarity Method

1 Introduction

The amount of Web shops has rapidly grown over the last years and, along with the Web shops, the range of products available online increased drastically. Unfortunately, these Web shops might use different descriptions and representations for the same products. It is possible that a Web shop provides additional information on a certain product, while another Web shop does not. For example, *MediaMarkt* can provide information about the weight of a certain laptop, while *Bol.com* might give information about the operating system, but not the other way around. Furthermore, the price of the same product can differ between Web shops. Therefore, product information often varies across different Web shops. Getting the best price or finding a comprehensive description of the specifications of a product is very time consuming for customers, especially when they

have to search among different Web shops in order to find the same product. It is manually even impossible to find a complete overview of the product specifications and the prices among all available Web shops.

Different state-of-the-art duplicate detection methods exist, such as the Multi-component Similarity Method (MSM) used in [2]. However, these methods are very time consuming [2]. Especially when performing duplicate detection on a large amount of data, the running time can easily explode. Therefore, it is convenient to apply a pre-selection that provides candidate pairs and to only perform a duplicate detection method on this set of candidate pairs. One way of finding the candidate pairs is by using Locality-Sensitive Hashing (LSH) [10]. By applying the pre-selection with LSH, the amount of products that will be compared by the complex and time consuming duplicate detection method is reduced. Therefore, the time needed to find duplicates is also decreased.

The research proposed in this paper extends the one from [6]. In [6] the authors only use model words from the product title for duplicate detection. We propose to add information from the key-value pairs in the product descriptions in order to reduce the sparsity of the considered data and to lower the number of false negatives. Furthermore, we use data cleaning by detecting inconsistencies and improving the quality of our data. Lastly, in [6] the signature matrix is built by explicitly performing permutation over the rows. However, in practice, this is a very time-consuming approach. Therefore, in this paper, we use random hash functions to simulate involved permutations. The proposed method is called the Multi-component Similarity Method with Pre-selection+ (MSMP+), as the original method from [6] is called Multi-Component Similarity Method with Pre-selection (MSMP), stressing thus the link to the old method.

The structure of the paper is as follows, in Sect. 2, we describe related work in the fields of data cleaning, key-value pairs, model words, LSH, and duplicate detection methods. Next, in Sect. 3, a description of the duplicate detection method we propose in this paper is given. In Sect. 4, we evaluate our method and last, in Sect. 5 we give our conclusions and suggestions for possible further research.

2 Related Work

The number of Web shops increased enormously over the last years, which is accompanied by a large growth in online data. In order to integrate these data from different heterogeneous sources, it is crucial to use an efficient duplicate detection method. Duplicate detection methods for online data are discussed widely in previous literature. Fetterly et al. proposed an algorithm that detects duplicate Web pages by using Web crawlers, and tracked how clusters of duplicate documents evolve over time [8]. Furthermore, Henzinger did a study on different algorithms that identify duplicate Web pages [9]. She compared Broder et al.'s shingling algorithm, in which the similarity of a subset of shingles is computed based on the Jaccard similarity between two documents [3], and Charikar's random projection algorithm [4], and identified the shortages of these two approaches. She

proposed an algorithm that combines the quality of both algorithms and that improved the precision compared to the performances of the algorithms of [3] and [4], individually.

Before applying a duplicate detection method it is crucial to define the input of the algorithm as binary vectors that represent the different products. In academic literature, there are several papers that introduce model words, for example [1,6]. Model words are defined as words that contain both numeric and alphabetic/punctuation tokens e.g. 12". These types of tokens often give valuable information for the duplicate detection, as they usually represent some unique aspects of a product. The model words are the input for an algorithm that creates a binary vector representation. In [6] only model words from the title are used to create binary vectors representing the products. These binary vectors are employed to find the candidate pairs by the LSH method. [6] does not use any form of data cleaning. However, data cleaning by removing errors and inconsistencies will increase the correctness of the data and avoid wrong conclusions, as argued in [12]. Another limitation of the work proposed in [6] is that this method does not use any information provided by the key-value pairs, while de Bakker et al. [1] show that the key-value pairs contain relevant information that can be used for the duplicate detection.

De Bakker et al. [1] introduce the Hybrid Similarity Method (HSM) for duplicate detection and extract model words from title and model words from the key-value pairs, which leads to superior duplicate detection results. An example of a key-value pair is: ('Weight', '20.5 lbs'). It is suggested that one should only use the product attribute values ('20.5lbs') and disregards the keys ('Weight'). For model words in the key-value pairs, de Bakker et al. [1] use a broader definition of model words. This definition also includes purely numeric tokens in addition to the mixed numeric/non-numeric tokens, e.g., '41.7' from '41.7 inches'. De Bakker et al. [1] show that model words of the key-value pairs can be useful when applying duplicate detection. Therefore, in our work we extend the binary vectors by also taking the information of the key-value pairs into account, and thus reducing the sparsity of the vectors.

After obtaining the binary vectors, LSH is a useful tool to reduce the dimensionality of the binary vectors [10]. This technique reduces the dimensionality of the data sets by mapping similar items of the signature matrix, constructed by minhashing, into the same buckets with a high probability. The LSH technique maximizes the probability of finding similar items in the same bucket and can be compared to the nearest neighbour search clustering algorithm [6].

LSH uses minhashing, in which the probability of a duplicate detection is maximized given a certain desired Jaccard similarity of two different sets. Cohen et al. [5] propose a minhashing function that defines a signature matrix that is constructed by randomly permuting the rows of the characteristic matrix and selecting for each column the first row index in which the column has a 1. Documents with similar signatures can be considered as similar and therefore become duplicate candidates. A disadvantage of using random permutations is that it is computationally intensive and the space that is required to store the

permutations is large [5]. The same approach is used by van Dam et al. [6], in which the same minhashing algorithm is used to define the signature matrix.

By combining the LSH algorithm with minhashing, duplicate detection becomes quick without losing a lot on recall and precision, because it retains the high Jaccard similarity items. Duan et al. [7] propose various LSH methods to improve the scalability of matching, in order to handle both large numbers of instances or match a large number of pairs efficiently. They propose to estimate the item similarity based on a small number of random hash functions and make use of the banding technique to avoid the quadratic complexity when comparing all the pairs. In this paper we plan a similar approach based on hash functions to simulate permutations and thus decrease the computation time and required space.

In order to find the final duplicates, after a pre-selection by LSH, the state-of-the-art product duplicate detection method MSM can be used. This method is described and employed in [2,6]. MSM uses a hierarchical clustering which leads to a relatively high F_1-measure, but simultaneously a large computation time.

3 Method Overview

Figure 1 gives a general overview of the approach used in our paper. We refer to our approach as MSMP+, as the approach of [6] is called the MSMP method. We start with data cleaning followed by extracting the model words from the product titles and key-value pairs. Thereafter, we create a binary product representation for each product. The next phase is to apply Locality Sensitive Hashing (LSH) to get candidate pairs and eventually MSM is used to get the final set of duplicates from the candidate pairs.

3.1 Data cleaning

To increase the correctness of the data and therefore the results, we use a data cleaning approach. The importance of data cleaning is mentioned for example in [12]. In the Web shop data, used for evaluation in this paper, some inconsistencies exist. These inconsistencies will lead to fewer found candidate pairs and therefore a higher number of false negatives. To efficiently correct for inconsistencies in the data, we used a frequency count of the model words. The most frequently occurring units are transformed into a standardized format. This transformation consists of three steps. First, all different representations of the units are transformed into one. E.g., the '"' sign for inch is normalized to 'inch'. In the second step, all upper-case characters are replaced by lower-case characters. Lastly, all spaces and non-alphanumeric tokens in front of the units are removed. The two main inconsistencies found are the representations of 'hertz' and 'inch'. Frequent variations of hertz and inch are transformed according to the steps described above. The results of these transformations of inch and hz are shown in Table 1. For example '23 Inch' becomes '23inch'.

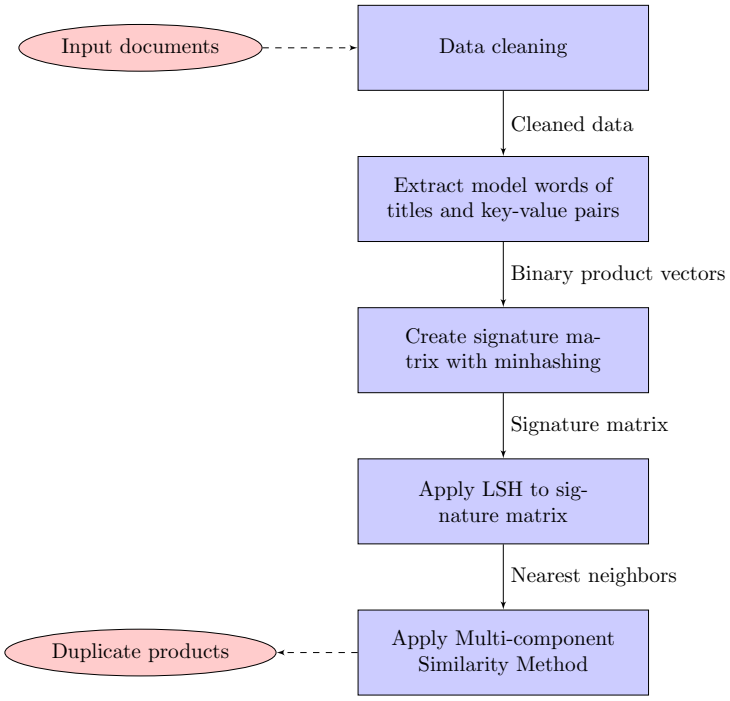

Fig. 1. General overview of MSMP+

Table 1. Transformation of frequent representations

Data value	Normalized value
'Inch', 'inches', '"', '-inch', ' inch', 'inch'	'inch'
'Hertz', 'hertz', 'Hz', 'HZ', ' hz', '-hz', 'hz'	'hz'

3.2 Signature matrix

In this paper, model words from the title and the key-value pairs as proposed in [2] are used to create binary vectors representing the product. The binary vectors obtained from these model words are used to create a signature matrix with minhashing.

Properties of products are often represented as key-value pairs. The titles in product data are relatively uniformly defined, whereas in the representation of the properties there seems to be more variation across Web shops. The weight of a television, for example, will in one Web shop have value '20.8 lbs' whereas in the other Web shop it will be represented as '20.8lbs'. Simply using the same definition for model words in the key-value pairs as in the title therefore does not seem logical since only the '20.8lbs' would be added, creating more dissimilarity between the binary vector representations of the products. To account for this we only add decimal numbers, that either stand alone like '20.8' or that have

a numerical part and a qualitative part like, '20.8lbs' where the 'lbs' part will be deleted so that in both cases the modelwords will only contain a numerical part. Decimal numbers are used since these numbers often show a certain measurement, which contains specific information about the products.

Some product titles, however, contain more information than others. Therefore we use the model words from the title and search for these in the key-value pairs. In the key-value pairs, information from the title is often repeated, thus if one duplicate contains some information in the title and another does not, this information can possibly be extracted from the key-value pairs.

For the model words we use the definitions from [2] and an extended definition. Model words as defined by [2] have the property that they contain both numeric and alphabetic/punctuation tokens. The model words are defined by the following regular expression:

$$ModelWord_{title} = \text{([a-zA-Z0-9]*(([0-9]+[\^0-9,]+)|([\^0-9,]+[0-9]+))[a-zA-Z0-9]*)}$$

This regular expression consists of several sub-patterns. The pattern [a-zA-Z0-9] recognises alphanumerical tokens, [0-9] numerical tokens, and [\^0-9,] special characters. Model words consist of at least two of these three types.

The extended definition of the model words includes decimal numbers and allow these decimal numbers to have an optional non-numeric part. These model words are defined by regular expression:

$$ModelWord_{key-valuepairs} = \text{(\^\textbackslash d+(\textbackslash .\textbackslash d+)?[a-zA-Z]+\$|\^\textbackslash d+(\textbackslash .\textbackslash d+)?\$)}$$

This regular expression contains of two parts split by the or sign "|". The part \^\d+(\.\d+)?[a-zA-Z]+$ finds decimal numbers followed by an alphabetic characters. The second pattern \^\d+(\.\d+)?$ finds all the decimal numbers without alphabetic characters.

After identifying these value-based model words, the non-numerical part of these model words is deleted as mentioned earlier. By doing so, we make sure that more similarity is created between products that have the same value for a property, but have a different representation of that value.

We formalize the procedure of obtaining binary vectors in a similar way as [6], but with the additional use of the models words of the key-value pairs. Let P be the set of product descriptions corresponding to the N products we consider in our data. Furthermore, let $title(p)$ denote the title of product $p \in P$ and $values(p)$ denote the set value attributes of the key-value pairs of a product $p \in P$. The procedure of obtaining binary vectors is shown in Algorithm 1. In the first part we initialize MW_{title} as the set containing all model words from titles and MW_{value} as the set containing all the model words from the value attributes of all product descriptions. Secondly, for every product p we define a binary vector b^p by setting element i equal to 1 if the title or a value attribute of product p contains model word $i \in MW_{title}$, or if a value attribute contains a model word $i \in MW_{value}$, and 0 otherwise.

Algorithm 1 Obtaining Binary Vectors

$MW = \emptyset$
for all products $p \in P$ **do**
  for all model words $mw_{title} \in title(p)$ **do**
    $MW = MW \cup \{mw_{title}\}$
  end for
  for all model words $mw_{value} \in values(p)$ **do**
    $MW = MW \cup \{mw_{value}\}$
  end for
end for
for all products p in P **do**
  for all model words $mw \in MW$ **do**
    if $mw_{title} \in title(p) \vee mw_{title} \in values(p) \vee mw_{value} \in values(p)$ **then**
      $b^p_{mw} = 1$
    else
      $b^p_{mw} = 0$
    end if
  end for
end for
return b^p for all $p \in P$

3.3 Defining Duplicate Candidates

After the extraction of the model words and the conversion into binary vectors, we can apply LSH. By applying LSH one can reduce the high dimensionality of the original data set. In addition, the number of computations is reduced. The LSH algorithm maps possible similar items from the original data set into the same bucket. Items within the same bucket can therefore be seen as duplicate candidates. The number of duplicate candidates is much smaller than the original number of input items and therefore the number of computations done by MSM is reduced, which is important because of the previous large running time.

Minhash Signatures. In order to apply the LSH technique efficiently, one can replace the large set of binary codes with a smaller set. These representations are called *signatures*. The final goal of these signatures is to compare them and to give an accurate and fast estimation of the Jaccard similarity of two sets. An effective way to reduce the large set is by applying *minhash signatures*. This technique computes a signature for each set, so that similar documents have similar signatures and dissimilar documents are not likely to have similar signatures. It picks a list of permutations and computes a minhash signature for each set in the data. The permutations are generated randomly using random hash functions. The hash functions are of the following form:

$$h_{a,b}(x) = (a + bx) mod(p) \tag{1}$$

in which a and b are random integers and p a random prime number ($p > k$), where k is the dimension of the new vectors [11].

The number of instances in the reduced set (signature matrix) is therefore equal to the number of instances in the characteristic matrix, but the number of rows r is reduced to k. A high number of min-hashes gives more stable results. Since computing similarity with MSMP+ takes up most of the computation time we set the amount of min-hashes to be equal to 50% of the total size of the binary signature vector. We reduce the number of rows by 50% and therefore, k is half of the value of r.

Locality-Sensitive Hashing. In order to find pairs with large similarity efficiently, LSH can be used [13]. LSH divides the signature matrix M into b bands with r rows for each band. The b and r must be chosen in such a way, that the following equation holds:

$$n = r * b, \tag{2}$$

with n the length of the columns of signature matrix M. In each band, the columns are hashed and divided into buckets. Items are hashed several times, because of the usage of multiple bands. The products that are hashed to the same bucket at least one time are now categorized as candidate pairs, which will be later checked by the MSM algorithm. The LSH algorithm reduces the number of candidate pairs that has to be compared by the MSM algorithm.

The relation between the false positives and the false negatives can be represented by the threshold t. An approximation of this threshold is:

$$t \simeq (1/b)^{1/r}. \tag{3}$$

A higher threshold reduces the false positives and increase the false negatives, while a lower threshold reduces the false negatives and increases the false positives.

3.4 Multi-component Similarity Method

The final method we use after LSH is the MSM. MSM is a hierarchical adpoted single linkage clustering method that makes use of a specific function in order to calculate the similarity of two products. This similarity function consists of three parts. The first part compares matching key-value pairs. The overlapping q-grams are used as a similarity measure. Alternatively, the cosine- and Jaro-Winkler measure could be used, but these are sensitive to misspellings and are token-based. The q-gram uses tokens of q characters, in this case $q = 3$. These are taken from a sliding window from the left to the right of the string. This calculated similarity is added to the final similarity of the two products with a learned weight.

The second part consists of the key-value pairs that were not matched in the first part. For these pairs the HSM method [1] is used. This method uses the

model words and calculates the percentage of matches. This calculated similarity will again be added with a learned weight to the final similarity of the two products.

The final part of the similarity function uses the Title Model Words Method (TMWM) [14], which employs the model words from the product titles. This is also added after using a learned weight to the final part of the similarity function of two products.

MSM uses an adapted hierarchical single linkage clustering. This algorithm is performed on the dissimilarity matrix, containing the dissimilarities between products. Some of the dissimilarity values are manually set to infinity. This is true for products of the same Web shop, products that have different brands, using a list of television brands from the Web [15] and for products that are not considered to be candidate pairs by the LSH method. The remaining dissimilarity values are computed by using the same similarity function for each pair of products, as was mentioned before. The adapted hierarchical single linkage clustering is performed on the dissimilarity matrix, where the distances between clusters are defined as the shortest distance between a pair of products from these clusters. A cluster that contains a pair of objects with distance infinity will also have distance infinity. This iterative process will continue to merge the two nearest clusters until the distance exceeds a certain threshold. The clusters obtained can be seen as clusters containing duplicates.

4 Evaluation

In order to evaluate the performance of our method, a data set containing information about televisions sold in four different Web shops is used. The aim of the method is to find duplicate televisions across these Web shops with a high precision and as few comparisons as possible. This is done by using a preselection with LSH before we apply the actual duplicate detection method MSM. We compare the results of our method denoted as MSMP+ with the results of the method proposed by [6] denoted as MSMP.

4.1 Data

The four Webshops in our dataset are international Web shops, namely `www.amazon.com`, `www.bestbuy.com`, `www.thenerds.net` and `www.newegg.com`. The data set contains information of 1629 different televisions in total. All products are represented with a title, as well as additional information stored in key-value pairs. The representation of the title can be seen as a summary of the product as it contains information of several properties of the product. An example of a television product title is: 'Philips 4000 Series 29\" Class 2812\" Diag. LED 720p 60Hz HDTV 29PFL4508F7 - Best Buy'. This representation gives you information of the brand, television type, resolution refresh rate, and the Web shop that sells the television.

Furthermore, all product descriptions contain other properties in the key-value pairs, such as: 'shop', 'url', and 'ModelID'. If the 'ModelID' for two different products is the same, the products can be considered as duplicates. ModelID's are often not present, which makes the approach of our method useful, but for evaluation purposes we have selected a data set where these ModelID's are present to have a high quality gold standard. We use the comparison made with ModelID's as a benchmark to evaluate our method. The number of properties, represented as keys, and the keys themselves are different across products. For example, one product might contain information on the HDMI-input, while another product might not have this information represented.

4.2 Evaluation Methods

To evaluate the performance of the methods, different metrics are used. In the first part we evaluate the LSH results and in the second part the MSM results. The LSH-method is evaluated with two metrics: Pair Quality (PQ) and Pair Completeness (PC). These are defined as, respectively:

$$PQ = \frac{D_f}{N_c}, \tag{4}$$

where D_f is the amount of duplicates found, and N_c is the number of comparisons made.

$$PC = \frac{D_f}{D_n}, \tag{5}$$

where D_f is the amount of duplicates found and D_n is the total amount of duplicates.

To evaluate the complete method, the F_1-measure is used. The F_1 measure is the harmonic mean of PQ and PC. Its optimal value is 1 and its lowest value is 0. The corresponding formula is:

$$F_1 = \frac{2 * PQ * PC}{PQ + PC}. \tag{6}$$

4.3 LSH performance

In order to achieve a consistent result we make use of a procedure that is called bootstrapping. Bootstrapping relies on random sampling with replacement. This re-sampling method allows us to evaluate the performance of the method on every bootstrap. In total, 100 bootstraps are performed and around 60% of the products in the data set are used in every bootstrap. This way, each bootstrap contains roughly 1000 products. The final performance of every measure is computed as the average over all bootstraps.

The number of found candidate duplicate-pairs depends on the size n of the signature matrix and the threshold value t as described in Sect. 3. A higher

value for t will lead to fewer candidate-duplicate pairs and therefore to fewer comparisons by MSM. However, a higher threshold value t will also lead to a smaller amount of found candidate duplicate-pairs and a potential higher number of false negatives. The aim of this research is to lower the number of comparisons made by MSM, while still finding a large amount of the duplicates. Therefore we run the algorithm for different values of t. A trade-off is made in order to define the best value of t, such that the PQ and the PC are optimized. The fraction of comparisons is defined as the candidate duplicate-pairs found by LSH divided by the total number of possible comparisons. The fraction of comparisons is therefore directly related to the threshold value t. By adding more candidate duplicate-pairs, the PC will increase. However, by doing more pairwise comparisons, the running time of MSM increases as well. The threshold value t, executed for 100 bootstraps, characterizes this trade-off. We vary the value of t from 0 to 1 with a step size of 0.05. All results are averaged over the bootstraps.

In order to compare the performance of the LSH part of our MSMP+ method with the MSMP method, we plot the PQ, PC and F_1 measure against the fraction of comparisonsof the methods in Fig. 2, 3 and 4, respectively (after LSH, check only pairs that are in the same bucket). Moreover, these figures report the separate results for the MSMP method with data cleaning (clean), the MSMP method with data cleaning and model words from title and also the model words from the key-value pairs found in the tile (values), and the MSMP+ method this method contains all the elements from above plus the decimal model words as explained in section Method Overview.

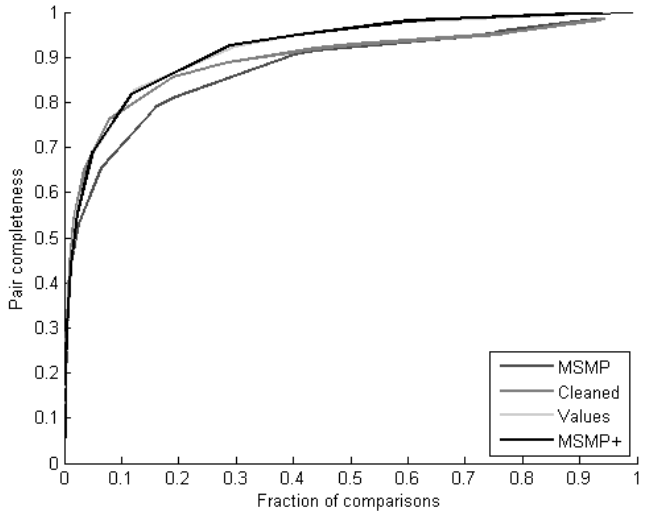

Fig. 2. Pair Completeness for different fractions of comparison

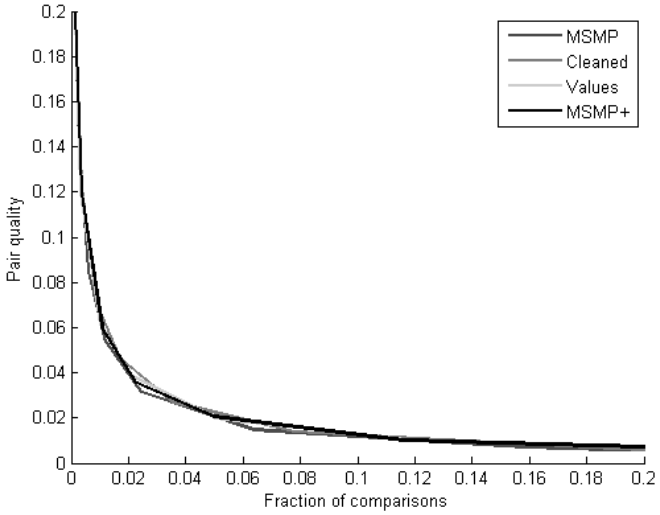

Fig. 3. Pair Quality for different fractions of comparisons

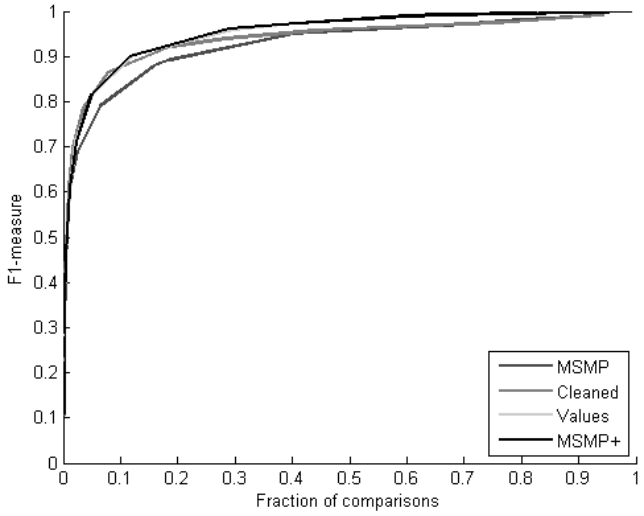

Fig. 4. F_1-measure of MSMP+ compared to the MSMP for different fractions of comparisons

From Fig. 2 it can be shown that the MSMP+ method outperforms the MSMP method. For example by performing 10% of the number of pairwise comparisons a PC of 80% is achieved with the MSMP+ method, while with the MSMP method there is only a PC of 70%. By calculating the area under the curve using the trapezoid method with the midpoint rule, it can be shown that MSMP+ method has an area under the curve improvement of 12.2% compared to the MSMP method. Furthermore we see that cleaning the data improves MSMP for the lower threshold values and using the key-value pair method does this for the higher threshold values. Finally there is only a slight change adding the decimal numbers.

In Fig. 3 an overview for the PQ against the fraction of comparisons can be found. Although the differences between the different methods of improvement are hard to distinguish, there is a 9.2% improvement of the area under the graph for MSMP+ against MSMP.

Finally we make use of the so-called F_1 measure to compare the results of the LSH part in both methods in Fig. 4. There is a 9.3% improvement of the area under the graph, in this case. Same as in Fig. 2, cleaning the data improves MSMP for the lower threshold values and the key-value pair method does this for the higher threshold values.

We conclude that the LSH part of the MSMP+ method proposed in this paper significantly outperforms the MSMP method on all the evaluation metrics: PC, PQ, and F_1.

4.4 Performance of MSM

In order to evaluate the performance of the MSMP+ method compared to the results of MSMP, the F_1-measure is used again. The results are obtained by running the MSMP+ method for different fixed threshold values t, where every run consists of 10 bootstraps. Fig. 5, shows the results. The set of parameters used in the MSMP+ algorithm is optimized over a grid of parameters. For every value t, the best performing parameter is selected.

By applying the LSH pre-selection method before performing the duplicate detection method MSM, we reduced the number of pairwise comparisons with respect to the number of comparisons done by the MSM method. Note that the fraction of pairwise comparisons done by MSM is equal to 1 if all comparisons are performed. As shown in Fig. 5, we compare the F_1-measure obtained by the MSMP+ and MSMP method to the F_1-measure of the MSM method (benchmark). The F_1-measure of the benchmark is equal to 0.525 [2]. As the graph shows, the F_1-measure decreases as the fraction of comparisons decreases. However, a large decrease in faction of comparisons leads to only a small reduction of the F_1-measure. From Fig. 5 we can conclude that the MSMP+ method outperforms the MSMP method. If we look at a reduction of 95% of pairwise comparisons for example, the MSMP method leads to an F-1 measure of 0.46, while the MSMP+ method for this reduction leads to a F-1 measure of 0.49. The MSMP+ method has an improvement of the area under the curve for the F_1-measure of 7.8% compared to the MSMP method.

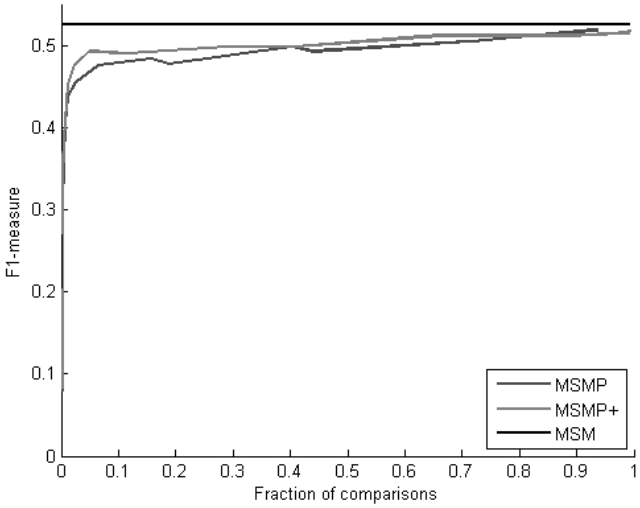

Fig. 5. MSMP+ compared to MSM

To summarize, the results show that LSH is an efficient method to significantly reduce the number of pairwise comparisons and therefore the computation time, while still finding an high number of duplicates. Also the MSMP+ method significantly outperforms the MSMP method.

5 Conclusions

In this paper we propose a method called MSMP+ in order to reduce the number of calculations involved in product duplicate detection on the Web. This method is an extension of the MSMP method described in paper [6]. The MSMP method uses a LSH pre-selection method before performing the duplicate detection method MSM. Before applying the LSH pre-selection method, it is necessary to define all products as binary vectors. In the MSMP method this is done by only using the model words that appear in the title. In the MSMP+ method, we first start with data cleaning followed by extracting the model words from the product titles and key-value pairs. Then we create a binary product representation for each product, apply LSH, and lastly perform te MSM algorithm.

When we only look at the performance of LSH the MSMP+ method has improved the area under curve compared to MSMP of pair completeness and pair quality with 12.2% and 9.2% respectively. The area under the curve of the F_1-measure is improved by 9.3%. After performing MSM on the candidate duplicates, the MSMP+ method has improved the area under the F_1-measure curve by 7.8%, compared to the MSMP method.

We conclude that the MSMP+ method outperforms the MSMP method on all evaluation criteria and therefore the MSMP+ method is a valuable extension of the MSMP method.

References

1. de Bakker, M., Frasincar, F., Vandic, D.: A hybrid model words-driven approach for web product duplicate detection. In: 25th International Conference on Advanced Information Systems Engineering (CAISE 2013). vol. 7908, pp. 149–161. Springer (2013)
2. van Bezu, R., Borst, S., Rijkse, R., Verhagen, J., Frasincar, F., Vandic, D.: Multi-component similarity method for web product duplicate detection. In: 30th Symposium on Applied Computing (SAC 2015). pp. 761–768. ACM (2015)
3. Broder, A.Z., Glassman, S.C., Manasse, M.S., Zweig, G.: Syntactic clustering of the web. Computer Networks and ISDN Systems 29(8), 1157–1166 (1997)
4. Charikar, M.S.: Similarity estimation techniques from rounding algorithms. In: Thirty-Fourth Annual ACM Symposium on Theory of Computing (STOC 2002). pp. 380–388. ACM (2002)
5. Cohen, E., Datar, M., Fujiwara, S., Gionis, A., Indyk, P., Motwani, R., Ullman, J.D., Yang, C.: Finding interesting associations without support pruning. IEEE Transactions on Knowledge and Data Engineering 13(1), 64–78 (2001)
6. van Dam, I., van Ginkel, G., Kuipers, W., Nijenhuis, N., Vandic, D., Frasincar, F.: Duplicate detection in web shops using LSH to reduce the number of computations. In: 31th ACM Symposium on of Applied Computing (SAC 2016). pp. 772–779. ACM (2016)
7. Duan, S., Fokoue, A., Hassanzadeh, O., Kementsietsidis, A., Srinivas, K., Ward, M.J.: Instance-based matching of large ontologies using locality-sensitive hashing. In: 11th International Semantic Web Conference (ISWC 2012). vol. 7649, pp. 49–64. Springer (2012)
8. Fetterly, D., Manasse, M., Najork, M.: On the evolution of clusters of near-duplicate web pages. Journal of Web Engineering 2(4), 228–246 (2003)
9. Henzinger, M.: Finding near-duplicate web pages: a large-scale evaluation of algorithms. In: 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2006). pp. 284–291. ACM (2006)
10. Indyk, P., Motwani, R.: Approximate nearest neighbors: towards removing the curse of dimensionality. In: Thirtieth Annual ACM Symposium on Theory of Computing (STOC 1998). pp. 604–613. ACM (1998)
11. Leskovec, J., Rajaraman, A., Ullman, J.D.: Mining of massive datasets. Cambridge University Press (2014)
12. Rahm, E., Do, H.H.: Data cleaning: Problems and current approaches. Bulletin of the IEEE Computer Society Technical Committee on Data Engineering 23(4), 3–13 (2000)
13. Slaney, M., Casey, M.: Locality-sensitive hashing for finding nearest neighbors. IEEE Signal Processing Magazine 25(2), 128–131 (2008)
14. Vandic, D., Van Dam, J.W., Frasincar, F.: Faceted product search powered by the Semantic Web. Decision Support Systems 53(3), 425–437 (2012)
15. Wikipedia: The free encyclopedia: `http://wikipedia.org/wiki/List_of_television_manufacturers`