Model-Words-Driven Duplicate Classification using Minhashing and LSH

Roel Veth

Erasmus University Rotterdam
Rotterdam, the Netherlands
622593rv@student.eur.nl

Abstract. The number of products sold online is growing rapidly, with many duplicate products being sold. For websites aggregating these product it is a challenge to find these duplicates as efficiently and effective as possible. In this paper the dimensionality of the problem is reduced by Minhashing. The number of required comparisons is lowered by using candidate pairs found by Locality-Sensitive Hashing. The candidate pairs are classified as duplicates or non-duplicates as determined by their Jaccard similarity. While the resulting performance of this classification method is largely inferior to the clustering methods used in previous works, the method is computationally simple and fast.

Keywords: Duplicate detection, Locality-Sensitive Hashing, Minhashing

1 Introduction

With the rise of widely accessible internet the number of online shops has also increased dramatically. The ease of access of these shops can allow the consumer to be more price-sensitive. However, with the increase of shops the number of online product offerings can be overwhelming, making it hard for consumers to properly compare products. To solve this problem some websites have started aggregating products from different webshops. To properly compare the different products it is necessary to know which products are identical. However, each webshop contains a large amount of data on each of their many products. Together with the great number of different webshops it becomes computationally unfeasible to simply compare all products. Another problem is that different webshops provide different data on their products. Thus methods to lower the computational burden are required. Two of those methods are explored in this paper: Minhashing and Locality-Sensitive Hashing (LSH). Minhashing is used to lower the dimensionality of the comparison problem. To lower the number of comparisons LSH can be used to create a pre-selection of candidate-pairs. This allows to skip all comparisons between products which are not deemed candidate pairs, while losing only few true pairs. In past work the resulting candidate pairs have been grouped by Multi-component Similarity Method, which was an extension of a Hybrid Similarity Method[5]. This work was continued by [6], who succesfully lowered computation time by reducing the number of pairwise comparisons. Finally, [7] have extented the method in [6] by extending the data with additional key-value pairs from the TV data to lower the data sparsity to lower the number of false negatives. In this paper it is attempted to create a duplicate detection system based on classification, where pairs of products are classified as duplicates or non-duplicates instead of grouping similar products together. Also the result of band size in LSH is researched. In this paper data on televisions (TVs) from four different webshops is used, which has also been used by [5], [6] and [7]. First the data is described, after which a detailed explanation of the product comparison is given. Next the results are displayed and discussed. Finally the conclusions are drawn and further research is suggested. The programming code can be found at https://github.com/RoelVeth/CS-for-BA.git

2 Data

A database of TVs is used as data in this research. In total 1262 different TVs are offered on four different websites: Amazon.com[1], Newegg.com[4], BestBuy.com[2] and TheNerds.net[3]. In total 1624 different offerings are made, with some TVs being offered on two, three or four different webshops. Each of the 1624 TVs has the following data points: *Model ID*, *Shop*, *Title*, *URL* and a *FeaturesMap*. The *Title* is how the TV is presented on the website and thus contains quite some information on the product, for example: *Newegg.com - LG 26" 720p LCD TV 26LD350*. The *FeaturesMap* contains a multitude of properties of the TV. Which properties are given in the *FeaturesMap* differs for each TV, examples are *Brand* and *weight*. In the method utilized in this paper only *title*, *shop*, *model ID* and the *Brand* from the *FeaturesMap* are used. The *Model ID* is only used to evaluate the estimated classifications.

3 Method Overview

To generate the results a multitude of processes is completed. First the data is cleaned, as to prevent errors by typographic differences in the data. Secondly a signature matrix is created using Minhashing. Next a set of candidate pairs is defined and finally the candidate pairs are classified as duplicate or non-duplicate. To evaluate the classification results a bootstrapping approach is utilized.

3.1 Data Cleaning

The first step is data cleaning, here the different parts of the data used in this research are cleaned to ensure proper results. The *shop* and *model ID* data are kept unchanged. In the *Brand* data the string *'tv'* and any spaces are removed. The *Title* data was cleaned the most. First all letters were transformed to lowercase, next the symbols '(', ')', '[', ']', '{', '}', ':' and '/' were removed. In the *Titles* often the size of the screen is given in inches and the refresh rate in Hertz. However, these values have many different representations. The original representations and their final representation are displayed in table 1.

Table 1. Transformation of representations 'Hertz' and 'Inch' representations. The representations were changed in the order displayed here. While the first and the last symbol for inch might appear similar, they are actually two different symbols which look equal in this font.

Original value	Final value
‘”’, ‘’ ”, ‘”’, ‘”, ‘ in’, ‘inches’, ‘-inch’, ‘ inch’, ‘”’	‘inch’
‘-hz’, ‘ hz’, ‘hertz’, ‘ hertz’	‘hz’

3.2 Signature Matrix

A signature matrix is used as representation of all TVs with reduced dimension. To develop a Signature matrix, first a characteristic matrix is required. A characteristic matrix contains the binary vectors of all products, to create these vectors a list of product elements is required. This list of product elements is a list of most words which can be found in the *Title* data. In the rest of this paper this list will be referred to as 'the modelwords list'. To ensure this list is not too large, the only words accepted in the modelwords list are those of a signature form. This form was designed by [6] and is defined by the regular expression (RegEx):

([a-zA-Z0-9]*(([0-9]+[ˆ0-9,]+)|([ˆ0-9,]+[0-9]+))[a-zA-Z0-9]*)

This expression recognizes three types of symbols, alphanumerical, numerical and special symbols. For a word to be accepted into the modelword list it requires at least two out of three. Examples are: `26inch`, `720p`, `26ld350` and `sb-6560hd`.

For each TV a binary vector is made, whose elements coincide with elements in the modelword list. If an element of the binary vector is '1', this indicates that this element is present in the *Title* of this TV. If the vector element is '0' the modelwords element is not present. For the characteristic matrix all binary vectors are placed side-by-side. In the data 1624 TVs are present and 1288 modelwords are found, thus the characteristic matrix is a binary representation of the data with 1288 rows and 1624 columns.

The signature matrix is generated from the characteristic matrix. The goal of this different representation is to decrease the number of rows, which will decrease computation time in future calculations. To create the signature matrix Minhashing is used. In this research the method described in chapter 3.3.5 of [8] is used. While it is not a very intuitive method it requires less computational power, which is very useful to make the code scalable for larger data sets. For this research the number of rows is decreased to 650, which is chosen as it is approximately half the number of model words and thus approximately halves the number of rows in the matrix.

3.3 Selecting Duplicate Candidates

To define the candidate pairs LSH is used. In LSH the signature matrix is split into b bands, each containing an equal number r of rows, thus $b \cdot r = N = 650$, where N is the number of rows in the signature matrix. For each band all columns are placed in buckets by a hash function. If the number of buckets is large enough the TVs will only be hashed into the same bucket when one or more of their bands are equal[8]. When TVs are placed in the same bucket they are classified as a candidate pair and will thus be examined further. TVs which are not classified as candidate pairs are not investigated further. TVs who have a Jaccard similarity equal to threshold $T_{Jaccard} = (1/b)^{(1/r)}$ have a 50% chance to be classified as candidates[8].

3.4 Classifying Duplicates

The classification of pairs as either a duplicate or a non-duplicate is a two-step process. In the first step the *Brand* and *Shop* data of the candidates are compared. If $Brand_A \neq Brand_B$ or $Shop_A = Shop_B$ TVs A and B are classified as non-duplicate. If $Brand_A = Brand_B$ and $Shop_A \neq Shop_B$ the second step follows. Here the Jaccard similarity between the binary vectors is calculated, this will result in a value between zero and one. This similarity is then compared to a threshold value T ($T \neq T_{Jaccard}$). If the similarity is larger than T the pair is classified as a duplicate.

3.5 Evaluation of Model

In this research the model is tested with different LSH properties. The values which are differed between tests are the number of bands, b, and the number of rows per band, r, while maintaining $b \cdot r = N$ with N the number of rows in the signature matrix. The values of r and b used can be found in table 2.

To evaluate the effectiveness of the model on new data a bootstrapping method is applied. For this method the data is split in training and test data. The training data is generated by drawing N random data points from the data set, here $N = 1624$ is the number of data points. When a data point is drawn and placed in the training set it is not removed from the data set, meaning it can be drawn again. When this happens the twice drawn data point is not added to the training set again. Asymptotically this results in a randomly drawn training set consisting of approximately 63% of the original data set. The data points which are not placed in the training set are placed in the test set. This is repeated 5 times, resulting in 5 random training sets and 5 corresponding test sets.

In a training set the duplicates are estimated several times, each time using another threshold T for classifying candidates as duplicates. To review the performance of the classification the *True Positives* (TP), *False Positives* (FP) and

False Negatives (FN) are determined using the *Model ID* data. With these values the performance measure, the F_1-score, can be calculated. The F_1-score is given by the harmonic mean between the *Recall* and the *Precision*, as defined by equation 1:

$$F_1 = 2\frac{Precision \cdot Recall}{Precision + Recall},\ Precision = \frac{TP}{TP + FP},\ Recall = \frac{TP}{TP + FN} \quad (1)$$

The threshold T which gives the best F_1-score on the training set is then applied to the corresponding test set to find the F_1-score on the test set. This is then repeated for all training sets. The model performance for parameters (r, b) is given by the average of the five F_1-scores achieved on the five test sets.

Table 2. The different values of the variables used during testing. Since b and r are bound by $b \cdot r = N = 650$ they are presented together.

Variable	**Values**
(b, r)	(10, 65), (13, 50), (25, 26), (26, 25), (50, 13), (65, 10)
T	0.10, 0.20, 0.30, 0.40, 0.50, 0.60, 0.70, 0.80, 0.90

4 Results

In this section the results are displayed and discussed. In table 3 the F_1-scores on training and test data are displayed. It is clear that the scores rise with decreasing $T_{LSH} = (1/b)^{(1/r)}$, and we see the highest F_1 test score for $(b, r) = (65, 10)$. This is expected, as less pairs are falsely discarded during LSH, thus lowering the false negatives number. However, what is quite interesting is that for $b = 13$, 50 and 65 we see an F_1-score on the test set which is higher than on the training set. This could be caused by the relatively low number of bootstraps. With only 5 bootstraps the results could still lack in robustness. Compared to the results of previous works using MSM based clustering we see that the F_1-scores are much lower. This is no surprise given that the Jaccard similarity, while it is computationally quick, is a quite crude similarity measurement. The utilized part of the data is also smaller compared to works such as [7], who used more key-value pairs from the *FeaturesMap* in the similarity measurements. While not researched, it is expected that with increasing b the fraction of comparison (FOC) increases since more candidate pairs are expected, this would also be in line with results from previous work. We have seen that in our researched range this increases the F_1-score. Another measure which was not researched in this paper, is the F_1^*-score, which is the harmonic mean of pair quality (PQ) and pair completeness (PC). An increase in FOC is suspected to lead to an increase in PC as it does for the F_1-score. Simultaneously it would likely lead to a decrease in PQ, as the number of candidate pairs compared most likely increases faster than the number of true duplicates found. This would lead to a optimal value of the F_1^*-score for a certain FOC.

Table 3. Results for different values of (b, r) and their respective implied LSH-threshold T_{LSH}. Here $F_1(training)$ denotes the average score on training data, $F_1(test)$ the average score on test data.

b	10	13	25	26	50	65
r	65	50	26	25	13	10
T_{LSH}	0.965	0.950	0.884	0.878	0.740	0.659
$F_1(training)$	0.048	0.027	0.050	0.060	0.092	0.136
$F_1(test)$	0.026	0.030	0.033	0.034	0.125	0.171

5 Conclusions

In this paper the previous work of [7] is continued, but with a classification approach instead of a clustering approach. Also the key-value pairs, except for *Brand* and *Shop*, are not taken into account. As a classification method the Jaccard similarity is compared to a threshold. While computationally fast, the F_1-scores are inferior to the clustering methods as proposed in previous works. We have seen that increasing the fraction of comparisons (FOC) increases the F_1-scores. In continued work it could be interesting to research this further. Different performance measure, such as the F_1^*-score, could also be researched to find for which fraction of comparisons the performance maximises.

References

1. Amazon.com, inc., `http://www.amazon.com/`
2. Best buy co., inc, `http://www.bestbuy.com/`
3. Computer nerds international, inc., `http://www.thenerds.net/`
4. Newegg inc., `http://www.newegg.com/`
5. 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)
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. Hartveld, A., Keulen, M., Mathol, D., Noort, T., Plaatsman, T., Frasincar, F., Schouten, K.: An LSH-Based Model-Words-Driven Product Duplicate Detection Method, pp. 409–423 (01 2018)
8. Leskovec, J., Rajaraman, A., Ullman, J.D.: Mining of Massive Datasets. Cambridge University Press, USA, 2nd edn. (2014)