Scalable Duplicate Detection for Web Shops using LSH and Gradient Boosting

Sem Koelewijn

Erasmus University Rotterdam. E-mail: 494615sk@student.eur.nl

Abstract. There exists a growing need for fast programs that can detect which identical products are available in different webshops. Duplicate detection algorithms using model words and locality-sensitive hashing have been proposed in the past, and frequently use clustering techniques to determine which products are the same. This paper investigates if a shingling approach combined with gradient boosting is a viable alternative. Although the initial filtering stage performs similarly to other algorithms using a model words-approach, the chosen classification method is found to perform significantly worse than previously proposed methods, with F1-scores that are 3.5 times smaller.

Keywords: Duplicate detection · web-shops · shingling · min-hashing · locality-sensitive hashing · gradient boosting

1 Introduction

For certain categories of products, online shopping is increasingly becoming the standard way through which purchases are done. With the World-Wide Web continuously expanding, there is a demand for convenient comparison between different webshops that offer the same item. Aggregating this information accurately is no trivial task, however, as the information webshops provide is almost never the same. (Near) duplicate detection algorithms have been developed in the past to combat this issue, but many struggle with the volume of data and the number of pairwise comparisons that need to be made [2, 3]. Newer algorithms usually make use of clever preprocessing and locality-sensitive hashing [4, 5]. Moreover, they usually employ (hierarchical) clustering solutions, many of which also tend to be slow. This paper will therefore investigate if (fast) classification models offer a good alternative. This will be done by applying altered techniques to the same dataset, which comprises descriptions of TVs in four different webshops. See [1] for the used code and dataset.

2 Related work

In the last decade, several papers have developed models that try to find duplicates in a dataset of TVs listed in webshops. For each TV, this dataset contains model IDs (to evaluate the performance), shop identifiers, product titles, and a

map of key-value pairs with a varying selection of features and their quantitative or qualitative feature descriptions. Examples include the number (or presence) of HDMI ports, aspect ratio and even warranty information. In total, there are 1624 TVs across four webshops, with 1262 unique model IDs. This means that we can find approximately 400 pairs of identical TVs in the data, out of roughly 1.3 million total pairs. The challenge is, however, to develop a model that finds these pairs without using the model IDs, as these are often not directly available.

[2] was one of the first papers to work with this dataset, and worked with measures such as TF.IDF to determine the similarity of an item compared to all other items in another webshop. [3] extended this to a model that can deal with multiple shops, but both had to employ many computational optimisation- and caching techniques to avoid excessive runtime or memory issues. [4] proposed the use of locality-sensitive hashing (LSH), which greatly reduced the amount of pairwise comparisons that had to be put through the model from [3], without sacrificing too much duplicate-identifying power. [5] further improved and results of [3, 4] through more elaborate preprocessing of all data using regular expressions instead of just the product titles. The preprocessing could be further improved by a more sophisticated technique called *blocking*, as featured in [6, 7]. Here, feature values are decomposed into quantitative and other supporting pieces of information, such as the measurement unit nits for screen brightness. These approaches made more use of shingling or q-grams instead of model words. Setting the blocking aside, using shingles usually involves less manual programming and can be executed quickly, and therefore might be a useful alternative to comparing model words in general. One aspect all these models share is that they make use of clustering techniques; leaving classification techniques somewhat understudied. These two points will therefore constitute what is changed to the general approach in related work. This method will now be described in more detail.

3 Method

The proposed algorithm will combine several elements from previous related papers in its first steps. In the first step, the dataset will be reformatted to extract relevant information and reduce computational time. Then, similarly to [4–6], min- and locality-sensitive hashing are used along with a heuristic to determine candidate pairs. These candidate pairs are then labelled according to whether they are an actual pair in the dataset, and similarity featured are computed. This data is then used to train a gradient boosting classifier.

3.1 Preprocessing

To properly analyse the dataset, the shared keys among all TVs are extracted: its model ID, shop, and title in shop, as well as a feature map, including more detailed information on the TV's specific features. Since the TV's titles will form the basis, these will be modified intensively. The demarcations for screen

size (in inch) and refresh rate (in Hz) are synchronised across the dataset, and any non-alphanumeric characters are removed. After this, the white spaces, and references to the brand of the TV or the shop it is in, are removed. The latter step is done such that these features will only be used in the classification step of the algorithm. For more information, see [5].

3.2 Pair Filtering

To determine which pairs of TVs qualify as candidate pairs, or conversely, to filter out pairs of TVs that are dissimilar, LSH is used on the titles of the TVs in the webshops. As mentioned in [4], LSH is an attractive technique due to its ability to detect (highly) similar items instead of only exact duplicates.

The process of LSH consists of three steps. All titles are first converted to binary column vectors. The resulting binary matrix then has its rows permuted n times and each time, it is checked from top to bottom for each column at which row the first 1 is found. This generates a $n \times m$ signature matrix. This signature matrix is split up into b bands, each with r rows such that $b \cdot r = n$. In each band, it is then checked if there are any identical pairs of columns. For a worked example, see [4]. This procedure is replicated here, except for the generation of the binary matrix. In this paper, the binary matrix is not made by extracting a set of model words from each string, but rather, a sliding window of k-shingles (or also often called q-grams), where k is a hyperparameter. This approach is instead similar to [6]. After determining the set of unique shingles across all titles, the binary matrix is then generated by checking for each shingle if it is present in a title.

Based on the results of [6], the possible hyperparameter space is set to $k \in \{2, 3, ..., 7\}$. Another related set of hyperparameters are the number of hashes and bands for our signature matrix. The number of hashes in the signature matrix is set to 420, or approximately equal to the square root of the number of items in the complete dataset. This is done to sufficiently reflect potential variation in the min-hashing process. The number 420 is chosen specifically, since with this number r can be set to many different values, such that b is also an integer. Since preliminary testing often yielded no candidate pairs if bands of 15 or more rows were created, we set the hyperparameter space of our number of bands to $r \in \{b \leq 420 | 420/b \in \{1, 2, ...14\}\}$, for ten options in total.

Next to applying LSH, it is determined for each pair of TVs if they are in the same shop. Only if they are not, the pair can actually be considered a candidate pair. This is a simplifying assumption that greatly reduces the number of pairs in one instant and only filters out about 2% of the actual pairs.

3.3 Classification

Once it has been determined which pairs of TVs could potentially be related, we can investigate these candidate pairs more closely and compute a number of similarity measures. For training purposes, it is checked if the candidate pairs have identical model IDs. This will be the binary classification label. In total,

five classification features will be generated, though this could likely be extended further. Firstly, we will measure the similarity of the title descriptions as well as the feature map descriptions of the two TVs labelled as a candidate pair. For the titles, we have already made related computations in our LSH steps. For the feature map, consisting of key-value pairs, we computed the weighted average of the similarity of all combinations of values of both feature maps, weighted by the similarity of the keys. This is similar to [3], except here no minimum key similarity threshold is applied.

We will measure the similarity of two descriptions though the Jaccard similarity measure. The Jaccard similarity of any two sets is defined to be equal to the ratio of the number of items in the union over the number of items in the intersection of the two sets. In our case, the sets are represented by boolean matrices, with 1 indicating presence in the set. For the titles, the binary matrix from LSH is used to compute the degree of similarity, with the length of the shingles equal to k. For the feature maps, new binary matrices are (temporarily) created by shingling the two key and the two value descriptions, from which the Jaccard index is then computed analogously for each combination of keys/values. Since the descriptions for keys and values are much shorter than the titles, large shingling lengths will likely yield low similarity scores for near-duplicates. As such, we let the hyperparameter space for the key-shingles $p \in \{2, 3, 4, 5\}$ and value-shingles $q \in \{1, 2, 3, 4\}$.

Next to these similarity features, which take real values from 0 to 1, we introduce three binary features based on elements of the title descriptions that were extracted in the preprocessing stage. These binary features indicate respectively whether the brand, screen size, and refresh rate of the TVs marked as a candidate pair are the same. Though these elements could be clear heuristics for pair filtering, the preprocessing might have introduced false negatives here. As such, these features are relegated to this part of the algorithm. In case the title does not provide this required information for either of the two TVs, the feature is set to 1 as a precaution.

With the classification data defined, we will run use a (standard) gradient boosting classifier to develop classification rules. We will split our dataset randomly into two parts through bootstrapping without replacement, where the (unique) sampled data points (usually around 1000) will be used to train the gradient boosting classifier, and the remainder will go through the same initial steps, though its labels will be predicted without using the model IDs and using solely the decision rules of the classifier instead.

4 Evaluation

To test our model, we run 10 bootstraps for each combination of hyperparameters. Due to limited computing power, we do this separately for k and r and check the performance of our filtering step first. We then check, for the optimal combination of k and r, all combinations of p and q in the classification step. The performance of the duplicate detection method are measured using

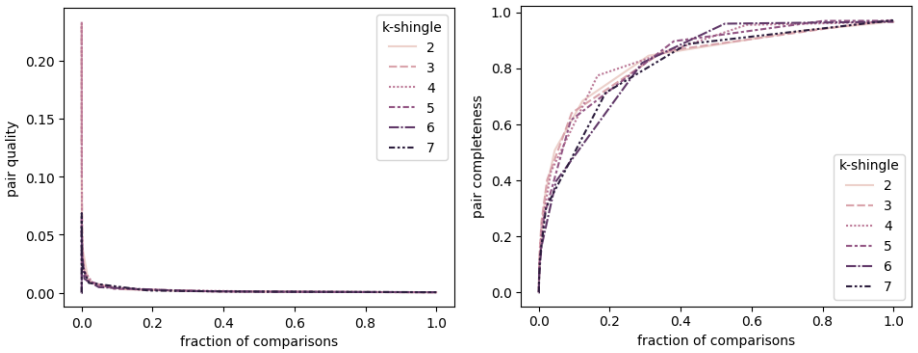

Fig. 1: Precision and recall of LSH-method at different degrees of filtering

the F1-scores, the harmonic mean of precision and recall. For the pair filtering performance, we report the provisional F1*-score, treating all candidate pairs as predicted pairs. [5] denotes precision and recall for this step as pair quality and pair completeness, respectively.

Figure 1 shows how the components of the F1*-score change depending on the amount of candidate pairs out of all possible pairs in the bootstrap. These results are similar to those reported in [4, 5], and show LSH can effectively filter out non-duplicates while retaining only relatively few, and can keep the majority of actual pairs after dropping 90% of the possible combinations. Moreover, it can be seen that the shingling length has a limited impact on this. Figure 2 shows that the combined score only using LSH is still low, but that the shingle length k does cause clear differences in the score depending on r. As the shingle length is reduced, the number of unique shingles across all titles is also much lower, and the range of values in the signature matrix is also shorter, making it more likely that bands have duplicate columns. Since pair quality dominates the F1*-score, making bands larger is needed to balance out this effect. The optimal combination of (k, r) is found to be (4, 7).

Using this combination, we again repeatedly run the entire algorithm for all combinations of p and q. Figure 3 shows the resulting F1-scores. It must be noted that these scores are only really influenced by the recall, as for all combinations the precision was equal to 1 for at least eight bootstraps. None of the specific combinations perform very well, though the combination $p = 4, q = 3$ seems to perform the best on average, with a mean of 0.1223 and a standard deviation of 0.0433 (the horizontal line in the boxes highlights the median). Compared to the related work in [4, 5], where scores of 0.46 and 0.49 were reported, this is about three and a half times worse.

5 Conclusion

In this paper, the potential of a straightforward classification model to detect duplicate items using shingling was evaluated using F1-scores, after applying

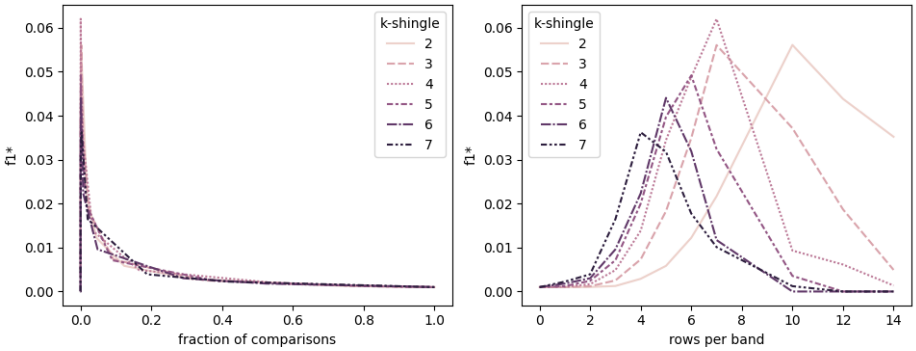

Fig. 2: F1*-scores at different degrees of filtering and rows per band

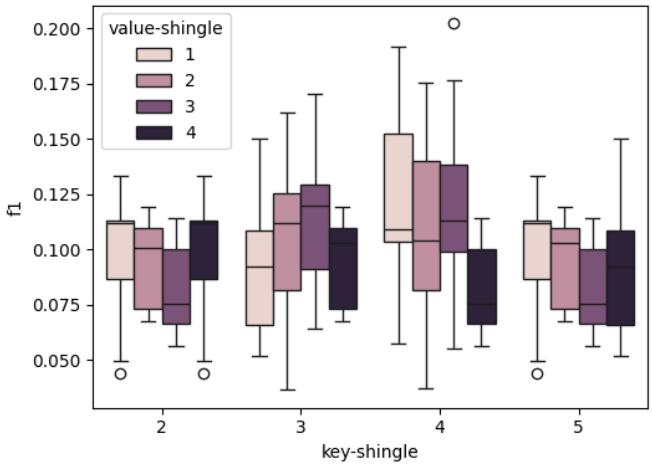

Fig. 3: F1-scores of full model, with title shingle length 4 and 7 rows per band.

locality-sensitive hashing as a filtering procedure. The results have shown us that, although the filtering displayed similar performance to that of previous work, the classification model struggles to balance precision and recall due to falsely detecting items as duplicates, resulting in a low F1-score. Given that gradient boosting is currently one of the better binary classification techniques, this poses a challenge for the use of a classification approach in general. The approach still has value where avoiding false negatives is desired, but if false positives are undesirable, clustering techniques have shown to be more promising in earlier works. Future works could further investigate the causes of degeneracy, for instance by applying and comparing various other binary classification techniques or developing more feature data. Since the classification model depends on the data it is tasked with classifying, newly developed techniques for preprocessing could also be implemented.

References

1. Koelewijn S. (2023): Duplicate detection algorithm for a dataset with TVs in webshops, using LSH and GB. Available at {https://github.com/SemK89/CSfBA-23-Duplicate-Detection}.
2. De Bakker, M., Frasincar, F., Vandic, D. (2013): A hybrid model words-driven approach for web product duplicate detection. In Proceedings of the 25th international conference on Advanced Information Systems Engineering (CAiSE 2013). pp. 149–161.
3. Van Bezu, R., Borst, S., Rijkse, R., Verhagen, J., Frasincar, F., Vandic, D. (2015): Multi-component similarity method for Web product duplicate detection. In: 30th ACM Symposium on Applied Computing (SAC 2015). pp. 761–768.
4. Van Dam, I., van Ginkel, G., Kuipers, W., Nijenhuis, N., Vandic, D., Frasincar, F. (2016): Duplicate detection in web shops using LSH to reduce the number of computations. In: 31st ACM Symposium on of Applied Computing (SAC 2016). pp. 772–779.
5. Hartveld, A., van Keulen, M., Mathol, D., van Noort, T., Plaatsman, T., Frasincar, F., Schouten, K. (2018): An LSH-based model-words-driven product duplicate detection method, In: 30th International Conference on Advanced Information Systems Engineering (CAiSE 2018). Lecture Notes in Computer Science, vol. 10816, pp. 149-161.
6. Van Rooij, G., Sewnarain, R., Skogholt, M., Van der Zaan, T., Frasincar, F., Schouten, K. (2016): A Data Type-Driven Property Alignment Framework for Product Duplicate Detection on the Web. In 17th Web Information Systems Engineering Conference (WISE 2016(1)). pp. 380-395.
7. Valstar, N., Frasincar, F., Brauwers, G. (2021): APFA: Automated product feature alignment for duplicate detection. In Expert Systems with Applications Volume 174, Article 114759.