Scalable Product Duplicate Detection

Owen Lin (547132)

Erasmus University Rotterdam

Abstract. This paper provides a scalable method for duplicate detection using MinHash, Locality-Sensitive Hashing and Agglomerative Clustering as an extension of existing papers covering the same topic. The method achieves an $F1$-score of around 0.26 averaged over bootstrap samples.

Keywords: Duplicate detection · Local Sensitivity Hashing · E-commerce

1 Introduction

In a world where consumer consumption is rapidly transitioning from offline to online stores, it only makes sense that similar or identical goods and services are offered by multiple online vendors. Transparency is of greatest importance for consumers to make a well-informed purchase decision. Also, firms need to know the pricing strategy of their competitors to position themselves. It is therefore important to develop a method that detects the same goods or services offered on different sites. One way is to use the product information on the web pages. However, it is unfeasible to compare them all. This paper will provide a scalable duplicate detection solution by exchanging some efficacy for efficiency. This will be applied to a data set consisting of televisions offered on various international web shops. The python script along with other resources for this paper is provided here: https://github.com/owen1in/duplicate-detection.
In the next section Related Work, relevant literature is discussed. The method is formulated and applied in Methods. Then, the results are presented in Evaluation and finally, we arrive at Conclusion.

2 Related Work

Many papers have proposed possible methods to detect product duplicates. The Multi-component Similarity Method (MSM) proposed by [4] uses the similarity between the products' features and titles to cluster products from multiple web shops. This method improved the $F1$ accuracy by a significant margin compared to existing solutions like the Title Model Words Method. [5] used MSM, based on extracted model words from the product's title, in combination with min-hashing and Locality-Sensitive Hashing (LSH) to reduce the computational load in the MSM, which led to the MSMP, where the 'P' refers to the pre-selection done by LSH. Model words are used such that an accurate binary vector representation of the products can be made. Their work is further extended by [1] where they also use a product's features for model words.

3 Methods

The method starts with data cleaning followed by extracting the model words. Then, min-hashing is performed and LSH is applied to the resulting signature matrix. From the obtained candidate pairs, duplicates are clustered using an agglomerative clustering algorithm.

3.1 Data Pre-processing

Data Cleaning The data set consists of 1624 TVs offered on 4 different web shops, of which 399 are duplicates. The method starts with cleaning the TVs' titles to increase the information retention of the model words in the next step as well as to increase uniformity. First, the curly and square brackets, parentheses and other redundant punctuation characters are removed. Then, various representations of units are standardized. For example, the TV's diagonal length is often denoted in inches. However, this is not consistent within and between sites and can be denoted as 60 inch, 60 inches or 60", to name a few. Therefore, inches, pounds and hertz are standardized to "inch", "lb" and "hz", respectively.

Model Words Consecutively, extraction of model words from only the titles is executed, which is beneficial for the algorithm's efficiency since it shrinks the universal set compared to when the key-value pairs of the features are used as well. These model words consist of non-space separated strings that are a combination of alphabetical and numeric characters, which could include punctuation marks, as per [1]. This universal set is extended with a general list of TV brands to improve accuracy. [1] The model words are used to create a characteristic matrix, where the column corresponds to a TV and a row corresponds to a model word. Let's say there are C models and R model words. These form the binary matrix B with elements $b_{r,c} \in \{0, 1\}$, where $r = 1, ..., R$ and $c = 1, ..., C$. Then, $b_{r,c} = 1$ indicates that model word r is in the title of model c and $b_{r,c} = 0$ means that the title does not contain the model word. 1300 model words are obtained, hence the binary matrix has $R = 1300$ rows and $C = 1624$ columns for the entire data set.

Min-hashing To increase efficiency while losing minimal efficacy, B's row dimension is reduced by 60%, while maintaining the similarity between products, using min-hash signatures as described in [2]. It will result in a signature matrix S with dimensions $M = R * 40\%$ by C with elements denoted by $s_{m,c}$, whose values are initialized by ∞. So, it has 40% of the rows of B, but it still has the same number of columns, where each column still corresponds to a TV. To create the signatures, the process starts with creating M random hash functions: $h_m(x) = (ax + b)\ mod(p)$, $m = 1, ..., M$, where a and b are random integers and p is a random prime number of at least M to avoid collision. Then, for row $r = 1, ..., R$ in binary matrix B, if a column in a row r equals one, that is $b_{r,c} = 1$, the corresponding values in column c in S are replaced by $h_m(r)$ if $s_{m,c} < h_m(r)$ for every row m.

Locality-Sensitive Hashing An effective approach to LSH for min-hash signatures is to divide S into l bands of length k rows and to hash every column's k-length vector l times for every band. [2] This way, any pair of TVs have l opportunities to end up in the same bucket and in case that happens at least one time, the pair is considered as a duplicate candidate pair. For simplicity, it is assumed that two TVs only hash to the same bucket if they are identical in at least one band. Since $M = l * k$ must hold, a choice of k directly determines l. Hence, it is sufficient to focus solely on k. To find a good balance between false positives and false negatives, we can refer to the threshold approximation: $t \approx (\frac{1}{M/k})^{\frac{1}{k}}$ which is increasing in k. A lower k means a pair of TVs have more chances to share a bucket and for every band, it only needs to uniformly agree in fewer rows. This will lead to more candidate pairs, hence more pairs need to be compared. Therefore, a low threshold leads to more false positives and fewer false negatives while the opposite is true for a higher threshold. To obtain flexible results, the last band's length is set equal to $Mmod(k)$ in case M is not divisible by k, which could also lead to more false positives since its length is at most k.

3.2 Clustering

Dissimilarity Matrix A list of candidate pairs has been produced from the Data Pre-processing. A dissimilarity matrix with C rows and C columns, which correspond to the televisions, is initialized with zeros on the diagonal and ∞ as its off-diagonal elements. Then, for every candidate pair (cp_1, cp_2), their Jaccard Dissimilarity (JD) is calculated using their binary vector representations from B:

$$JD(B, cp_1, cp_2) = 1 - \sum_{r=1}^{R} \frac{\mathbb{1}_{(b_{r,cp_1}=b_{r,cp_2}=1)}}{\mathbb{1}_{(b_{r,cp_1}\neq 0 \ and \ b_{r,cp2}\neq 0)}}$$

, where $\mathbb{1}$ represents the indicator function. This distance replaces the corresponding elements to obtain a symmetric in the dissimilarity matrix.

Agglomerative Clustering Using the dissimilarity matrix as an input for the unsupervised agglomerative clustering algorithm implemented by [3], pairs of clusters are merged recursively based on a distance threshold (dt). Complete linkage is used where the maximum distances between all observations of the two sets are computed.

4 Evaluation

Different metrics, averaged over 10 bootstrap samples for stable results, are used to evaluate the candidate pairs produced by LSH, and the final duplicates produced by the clusters. One bootstrap sample consists of roughly 60% of the entire data set by drawing $C = 1624$ observations with replacement and only keeping the unique draws. Candidate pairs are evaluated on pair quality $p^q = \frac{CP^{TP}}{CP^{total}}$, which is the number of candidate pairs that are true duplicates divided by the total number of candidate pairs proposed by LSH, and pair completeness $p^c = \frac{CP^{TP}}{D}$, where D represent the real total number of duplicates in the bootstrap sample. They are considered together in the $F1^*$-measure:

$$F1^* = \frac{2 * p^q * p^c}{p^q + p^c}$$

Then, the clusters are evaluated by the $F1$-measure:

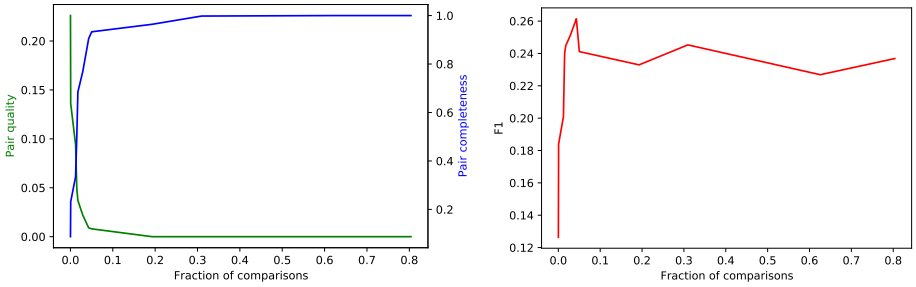
$$F1 = \frac{2 * P * R}{P + R}$$

, where $P = \frac{TP}{TP+FP}$ represent the precision of the clusters and $R = \frac{TP}{TP+FN}$ the recall. "T" and "F" refer to True and False, and "P" and "N" stand for Positive and Negative.
For every bootstrap sample, the length of the signature matrix S is roughly $M = 380$. This is somewhat shorter than expected because the rows that only contain 0 in B are removed. This can happen since some model words are not present in the title of the models in the bootstrap sample. Then, hyperparameter tuning is done for the distance threshold (dt). Combinations for rows $k = \{5, .., 10\}$, which on average correspond to thresholds $0.4 \leq t \leq 0.7$ for LSH, and distance thresholds $dt = \{0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5\}$ are tested. The optimal dt is found to be 0.2. This means that candidate pairs with dissimilarities of at least 20% will not be clustered together.
Then, using $dt = 0.2$, the relationship between the number of comparisons made and the p^q, p^c and $F1$ is examined. We will use the fraction of comparisons, defined to be the number of comparisons made in LSH, divided by the total possible number of comparisons one can make. The latter equals $\frac{n(n-1)}{2}$, where n is the size of the bootstrap sample.

Results The graphs in figure 1 illustrate the results. Observe that, in the left graph, the pair completeness increases rapidly while the pair quality decreases rapidly. This trade-off can be explained by the fact that the number of true duplicates in the sample is rather low. Naturally, the more comparisons are made, the more duplicates end up in the candidate pairs list produced by LSH. However, the rate of finding duplicates flattens, while the length of the candidate pairs list, thereby the number of comparisons, grows, leading to a lower pair quality. This leads to the shape of the observed curves, where we observe an

"elbow" at a fraction of comparisons around 0.05. The $F1^*$ stays close to zero for every fraction of comparisons, since p^q and p^q "take turns" taking values close to 0, hence $F1^*$ itself also stays close to zero and is therefore not presented.
Looking at the right graph of figure 1, which presents the $F1$-scores as a function of the fraction of comparisons, it can be observed that the curve also has an elbow when around the length of the candidate pairs list is around 5% of the total number of possible comparisons and flattens after that point.
Everything considered, it can be seen that a fraction of comparisons higher than 5% does not lead to a significant increase in the $F1$-measure. Also, the pair quality and pair completeness, p^q and p^c, have converged to approximately 0 and 1, respectively. Keep in mind that the higher the fraction of comparisons, the more computational power and time are needed. In practice, this means that k should not go lower than a certain value, since a lower k leads to more candidate pairs, hence more comparisons.

Fig. 1. The relationships of $F1$, pair quality and pair completeness with the fraction of comparisons



5 Conclusion

This paper proposes a scalable method for duplicate product detection by reducing the rows of the characteristic matrix, obtained from extracting model words, by 60%. Then for LSH, $k = 6$ is found to be the optimal length of a single band. This corresponds to a fraction of comparisons of 0.043, a pair quality of 0.009 and a pair completeness of 0.905. Values lower than $k = 6$ are discouraged since it is expected that this will not lead to major improvements while using more time and energy. Using the optimized value for the distance threshold of $dt = 0.2$, an $F1$-score of 0.261 is achieved. Since this paper was restricted in computational power, efficiency was often prioritized over efficacy. In a world where technologies are quickly evolving, more complex methods for duplicate detection can be explored that improve the accuracy of the duplicates. This is left for future research.

References

1. Hartveld, A., Keulen, M.v., Mathol, D., Noort, T.v., Plaatsman, T., Frasincar, F., Schouten, K.: An lsh-based model-words-driven product duplicate detection method. In: International Conference on Advanced Information Systems Engineering. pp. 409–423. Springer (2018)
2. Leskovec, J., Rajaraman, A., Ullman, J.D.: Mining of massive data sets. Cambridge university press (2020)
3. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research **12**, 2825–2830 (2011)
4. Van Bezu, R., Borst, S., Rijkse, R., Verhagen, J., Vandic, D., Frasincar, F.: Multi-component similarity method for web product duplicate detection. In: Proceedings of the 30th annual ACM symposium on applied computing. pp. 761–768 (2015)
5. 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: Proceedings of the 31st Annual ACM Symposium on Applied Computing. pp. 772–779 (2016)