Scalable Product Duplication Detection using Locality Sensitive Hashing

Ralph Hermeling

Erasmus School of Economics
513098rh@student.eur.nl

Abstract. Due to the exponential growth of e-commerce a vast amount of products are being sold online. To aggregate the information about these online products a scalable and effective duplicate detection algorithm is required. In this paper we address the scalability issue for duplicate product detection using Locality Sensitive Hashing (LSH) for data coming from several Web shops. This paper uses q-grams from the product title to make binary vectors. It achieves 10 times worse Pair Quality than existing methods such as MSMP(+).

Keywords: Duplicate detection · Min-hashing · Locality sensitive hashing · Web shop products · Q-grams

1 Introduction

Consumers search the web meticulously to find the desired product information. An algorithm that could aggregate all the relevant information for them would be greatly beneficial. The task of aggregating products however requires you to know which products are duplicates and this is a difficult assignment. Especially when online product information is scattered across the web. It is unfeasible to do it manually. Therefore the main focus of this paper is to address the scalability issue for duplicate detection using data from several Web shops. The code I wrote for this paper can be found here.

2 Related Work

E-commerce growth continues and is increasing due to the demand shock caused by the corona pandemic. This results in a huge amount of online data. It is only possible to aggregate the data coming from multiple sources if one has an scalable and efficient duplicate detection algorithm. Duplicate detection algorithms are extensively researched in previous literature. For example [2] employs LSH in order to reduce the amount of possible duplicates that can be used as an input for their Multi-component Similarity with Pre-selection+ Method (MSMP+). The LSH they performed uses min-hashing to create signatures from product representations. The results of LSH are therefore dependent on the binary vectors that represent the different products. The authors of [1,4] utilize Model Words,

which are words that consist of a numerical and alphabtical/punctuation part e.g. 1080p. Model words often contain unique features of the product and are therefore invaluable for duplicate detection. In [2] model words are extracted from the title and the product description (key-value pairs). This results in a decrease by 95% of needed computations. In this paper I only use q-grams from the title to make shingles. [2] use q-grams for their similarity computation but not for creating the binary vectors. A q-gram is a contiguous sequence of string with length q and it supports partial word-matching. As [3] states this makes it robust in a context of short, informative descriptions with abbreviations on Web shops.

3 Method

The LSH approach used in this paper consists of a three-step process. First we convert product titles into q-grams, which are then used to create product representations via binary vectors. Then I use min-hashing to create signatures which my LSH process takes as input. LSH outputs candidate pairs and in this way reduces the number of comparisons.

3.1 Data cleaning

To increase the chances of data properties colliding I perform the same data cleaning as [2]. This approach starts by transforming frequent variations of 'inch' and 'hertz' to their normalized counterparts which are 'inch' and 'hz', respectively. An overview of frequent representations is given in the Appendix. Thereafter all uppercase-characters are set to be lower case. Lastly I remove all white space and non-alphanumeric characters from the title, because I assume that sharing white space does not indicate whether two products are more likely to be duplicates.

3.2 Signature matrix

In this paper, q-grams from the title are used to create binary vectors representing the product. I use the following workflow to obtain binary vectors. Let N denote the set of products in our data and P the set of corresponding product descriptions. Moreover, let $title(p)$ denote the title of product $p \in P$. First I initialize a empty set called the vocabulary V. Thereafter for every $title(p)$, we build q-grams. The set of q-grams belonging to product p is denoted by V_p. Then for every product p we add V_p to V. The resulting vocabulary V is the union of all V_p. The procedure to obtain binary vectors for every $p \in P$ is as follows. First define a binary vector b^p with a length of $|V|$. If q-gram $v \in V$ is present in V_p then element b^p_v is equal to 1 and 0 otherwise.

3.3 Min-hashing

Minhashing allows us to convert our sparse binary vectors into dense vectors. To create a signature of length g one needs g hash functions. A hash function takes the following form $h_{a,b} = (a + bx) mod(p)$, where a and b are random integers and p a random prime number ($p > g$) [2]. One attractive property of min-hash signatures is that they approximate Jaccard similarity and this results in similar documents having similar signatures. In this way some information is still retained while simultaneously reducing the dimension. The amount of rows of the new input matrix remains the same as it still is the amount of products but the amount of columns reduces to g. I have chosen $g \approx 500$ to compare my results with [2]. This choice of g leads to a decrease of 83.3% of the original binary vector size.

3.4 Locality-Sensitive Hashing

To find pairs with large similarity efficiently I use LSH [2]. It takes as input a signature matrix M and hashes segments of each signature and looks for hash collisions. This approach is called the banding method. It splits my signatures into sub-parts called bands b, where r corresponds to the amount of rows each band b has. The b and r are chosen in such a way that the following equation always holds $g = b * r$, where g denotes the length of the signature matrix M. In this way each band is processed through our hash function and not the full signature. This allows for signatures that share even some similarity to end up in the same bucket. In other words given g we now have g opportunities to find matching sub-vectors between our signature vectors. Thus if sub-vectors match once or more then they are considered as candidate pairs. This also affects the false positive rate. Let t represent the relationship between the false positives and the false negatives. We use the same approximation as [2] for this threshold t: $t \approx (1/b)^{(1/r)}$. A lower threshold results in relatively lower false negatives and more false positives, whereas a higher threshold reduces the false positives and increases the false negatives.

4 Evaluation

4.1 Data

The data consists of 1624 unique televisions listed on four different international Web shops as described in [2]. For every television there is a corresponding title, modelID, shop, and features map. A features map consists of key-value pairs and each pair can differ across television observations. As explained by [2] the title functions as a summary of the product since it contains information about multiple characteristics of the product. I use the modelID property to identify the product duplicates and evaluate my LSH.

4.2 Evaluation Methods

I use the same statistics as [2] to evaluate LSH: Pair Quality (PQ) and Pair Completeness (PC). These are defined as, respectively: $PQ = \frac{D_f}{N_c}$, where D_f corresponds to the amount of duplicates found, and N_c to the amount of comparisons made. $PC = \frac{D_f}{D_n}$, where D_n corresponds to the total amount of duplicates. I use the F_1* measure to evaluate the complete LSH method. It corresponds to the harmonic mean of PQ and PC and is defined as follows:$F_1* = \frac{2 * PQ * PC}{PQ + PC}$.

4.3 LSH performance

To achieve a consistent result we use the same procedure as [2], which utilizes bootstrapping. In total I perform 100 bootstraps and the training data for every bootstrap consists of approximately 60% of the total amount of products. This is roughly 1,000 products. Then the final performance is equal to the average of all bootstrap runs. As stated in [2] the number of found candidate duplicate-pairs depend on the size n of the signature matrix and the threshold value t described in section 3.3. A higher t leads to fewer candidate-duplicate pairs and lesser comparisons. In contrast a high threshold t will result in a smaller amount of found candidate duplicate-pairs and therefore a higher false negative rate. The aim of this paper is to reduce the number of comparisons. Thus we run the algorithm for multiple values of t and try to find the optimal value for t, such that PQ and PC are maximized. In every graph the fraction of comparisons is given on the x-axis. It is defined to be the number of candidate duplicate-pairs found divided by the total number of possible comparisons. As stated by [2] the threshold t is directly related to this statistic. I vary the value t from 0 to 1 with a step size of 0.05. The size of the q-grams is set equal to 3.

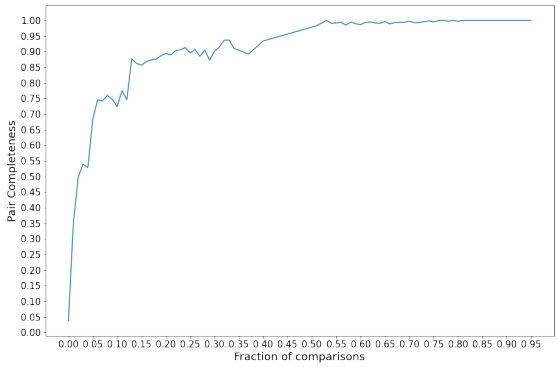

Fig. 1. Pair Completeness for different fractions of comparison.

Figure 1 shows that my approach achieves a PC of approximately 75% by performing 10% of the number of pairwise comparisons. This is 5% worse than the MSMP+ method described in [2], but 5% better than the regular MSMP method from [3]. The PC line of my method follows the same general trend as in [2].

Figure 2 shows that my algorithm performs 10 times worse than MSMP(+) [2,3]. For a fraction of comparison that is approximately 0, my method achieves a PQ of 0.017, whereas MSMP+ achieves a PQ of 0.2. In other words, my method requires 10 times more comparisons to find the same amount of duplicates. This indicates that signatures created by using q-grams do not retain the same level of information as Model Words from [2,3]. It can be caused by two things. First I have picked $g \approx 500$ and this reduces the binary vector size by 83.3%, as a result the titles and their corresponding signatures do not transfer over the same level of similarity as Model Words do. Secondly q-grams are too general and are not able to capture unique product characters in the same compact way Model Words do.

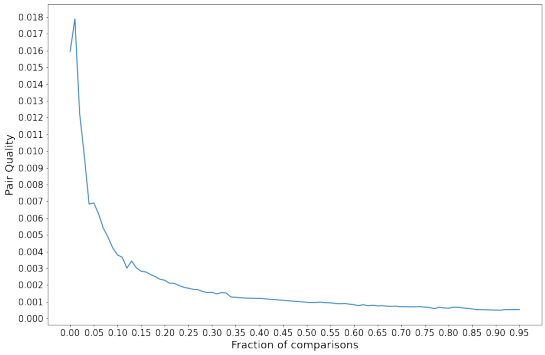

Fig. 2. Pair Quality for different fractions of comparisons.

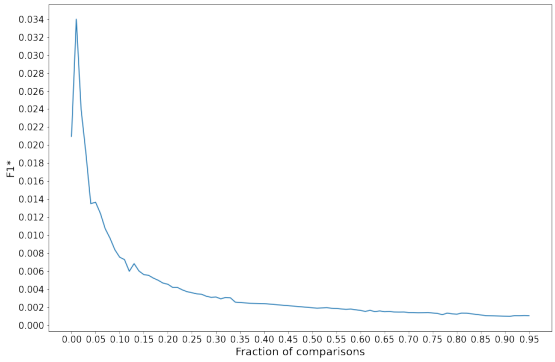

Fig. 3. F1* for different fractions of comparisons.

Lastly Figure 3 shows that F1* attains its maximum value for a fraction of comparisons equal to approximately 2.5% and thereafter rapidly declines to

0.002. This can be explained by the fact that F1* is the harmonic mean between PC and PQ. The PQ attains its maximum in the same fraction of comparisons range but thereafter steeply declines. The value of the PC increases for every fraction of comparisons. Therefore F1* steeply declines to approximately zero.

5 Conclusion

This paper proposes an alternate approach to acquire model words that is by creating q-grams from the title. It achieves better values for PC for the same fractions of comparisons as MSMP and is slightly more worse than MSMP+. However, MSMP(+) performs 10x better with respect to PQ. This indicates that q-grams signatures do not retain similarity at the same level Model Words do in [2,3]. Since our aim is to reduce the amount of possible comparisons by using LSH I do not recommend using q-grams, since it finds 10 times less duplicates in the same amount of comparisons as Model Words do.

References

1. De Bakker, M., Frasincar, F., Vandic, D.: A hybrid model words-driven approach for web product duplicate detection. In: International Conference on Advanced Information Systems Engineering. pp. 149–161. Springer (2013)
2. Hartveld, A., van Keulen, M., Mathol, D., van Noort, T., 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)
3. 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)
4. 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)

Appendix

Table 1. Transformation of frequent representations

Data value	Normalized value
‘Inch’, ‘inches’, ‘”’, ‘-inch’, ‘ inch’, ‘inch’	’inch’
‘Hertz’, ‘hertz’, ‘Hz’, ‘HZ’, ‘ hz’, ‘-hz’, ‘hz’	’hz’