Paper Topic: Scalable Product Duplicate Detection

The product information distribution on the Web forces consumers to spend considerable effort in order to find the desired product information. Consumers can greatly benefit if data from different Web shops could be aggregated to form a more complete set of product information. However, aggregating product data from different Web shops is a difficult task. The vast amount of products, product information, and Web shops makes it unfeasible to manually perform product aggregation, therefore, this process has to be automated. To automatically aggregate data from various Web sites, it is necessary to perform duplicate detection, i.e., to determine, using data from Web shops, which product descriptions refer to the very same product.

In this assignment you need to address the scalability issue for duplicate product detection using data coming from several Web shops. Due to the size of the data, as well as the number of comparisons to be performed, one needs to propose a scalability solution for the involved computations by reducing the number of comparisons using the approximation technique of Locality Sensitive Hashing (LSH). The approach is to be evaluated for both efficacy and efficiency on an available real-world data set containing a large number of products.

The dataset to be used is available from: https://personal.eur.nl/frasincar/datasets/TVs-all-merged.zip as a JSON file that contains 1,624 descriptions of televisions coming from four Web shops: Amazon.com [1], Newegg.com [2], Best-Buy.com [3], and TheNerds.net [4]. Finding the product duplicates can be approached as a classification problem, where pairs of products are considered duplicates/non-duplicates, or as a clustering problem, where clusters of products are considered duplicates, as, for example, in the Multi-component Similarity Method (MSM) [5]. The maximum number of duplicates, which is 4 in the considered dataset, is not known to the duplicate detection method.

The description of a product contains a header with information as the shop (e.g., Newegg.com), title (e.g., “Sharp 70" 1080p 120Hz LED-LCD HDTV - LC70LE650U”), and modelID (e.g., “LC70LE650U”), and a set of key-value pairs (e.g., “brand: Sharp”, “maximum resolution: 1920 x 1080”, “refresh rate: 120 Hz”, “screen size: 7”). The modelID is available only for training and for checking if the duplicates are properly found, i.e., the proposed algorithm has no access on the modelID for finding the duplicates when testing, but it is available for the evaluation of the algorithm (as the gold annotations). The reason is that the modelID is missing on many Web sites, which is what makes finding the product duplicates an interesting problem to solve.

The main focus of this assignment is on producing a scalable solution for product duplicate detection. This means that coming up with a very complex duplicate detection algorithm and a sophisticated similarity measure is less important here than coming up with an approach that scales well given the large number of products and pairs of these that need to be considered. By scaling well it is meant to have a slightly less effective method while improving greatly on its efficiency.

As evaluation technique, the bootstrapping method (drawing instances with replacement as many as the original number of instances) needs to be used. For each bootstrap you obtain approximately 63% of the original data as training data, while the remaining data (out-of-sample) is considered as the test dataset. For robustness you need to consider at least 5 bootstraps in the evaluation procedure. To evaluate the performance of the algorithm the average (across bootstraps) F_1-measure needs to be used. The F_1-measure is the harmonic mean between precision and recall.

The interpretation of true positive (TP), false positive (FP), true negative (TN), and false negative (FN) in this work is slightly different from usual. TP is given by the pairs of products that are predicted to be duplicates and are real duplicates. FP is given by the pairs of products that are predicted to be duplicates but are real non-duplicates. TN is given by pairs of products that are predicted to be non-duplicates and are real non-duplicates. Last, FN is given by the pairs of products that are predicted to be non-duplicates but are real duplicates.

In previous work we have proposed to use model words (words consisting of both numeric and non-numeric parts, e.g., "120Hz") from title in LSH to reduce the number of product comparisons to be made by a hierarchical clustering algorithm resulting in the method Multi-component Similarity Method with Preselection (MSMP) [6]. This work has been further extended by generalizing the model words (i.e., allowing also words with just a numeric part, e.g., "70") and using these from both title and key-value pairs (more precisely the value part of the key-value pairs) in LSH resulting in the Multi-component Similarity Method with Preselection+ (MSMP+) [7].

In this project you need to use LSH, but you are free to choose the product representations that you want as long as they are different from the previous work. For evaluating the performance of your scalability solution you need to use pair quality (number of duplicates found/number of comparisons made), pair completeness (number of duplicates found/total number of duplicates), and F_1^*-measure, which is the harmonic mean between pair quality and pair completeness. As there is a trade-off between pair quality and pair completeness, a discussion needs to be made on how pair quality, pair completeness, F_1^*-measure, and F_1-measure change for different fractions of comparisons (number of comparisons made/total number of possible comparisons). You are also free to decide on the method used to find the duplicates (a classification or a clustering-based solution), including the similarity measure (if needed).

As programming language you can use any programming language you like (e.g., R, Python, Java) and the GitHub link to your code needs to be given in the paper. Please make sure that you have added to the GitHub repository a README file explaining what this project is about, the structure of your code, and how to use the code. You can use any programming library you find useful (e.g., classification, clustering) besides one for LSH, which you need to implement yourself. The paper needs to be written in English, obey the Springer Lecture Notes in Computer Science style, and not be longer than 6 pages.

References

[1] Amazon.com, Inc. http://www.amazon.com
[2] Newegg Inc. http://www.newegg.com
[3] Best Buy Co., Inc. http://www.bestbuy.com
[4] Computer Nerds International, Inc. http://www.thenerds.net (now defunct)
[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., 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: 30th International Conference on Advanced Information Systems Engineering (CAiSE 2018). Lecture Notes in Computer Science, vol. 10816, pp. 149-161. Springer (2018)