Identity Resolution

Mining Product Data from the Web

presented by

Bertsch, Matthias Helmuth (1428768)
Demirxhiu, Ersejda (1577700)
Gjuzi, Anjeza (1623356)
Koseoglu, Bengi (1619463)

submitted to the

Data and Web Science Group
Prof. Dr. Christian Bizer
University of Mannheim

July, 2018

Contents

1 Introduction **1**

2 Gold Standard and Preprocessing **2**

2.1 Gold Standard Creation 2

2.2 Preprocessing 3

3 Models **4**

3.1 Approach 1 4

3.2 Approach 2 5

3.3 Approach 3 8

3.4 Approach 4 10

4 Conclusion **12**

Chapter 1

Introduction

The Web offers us daily an increasing number of e-shops, each of which containing a broad variety of products. To understand different offers that come from multiple vendors it is necessary to make a comparison between these e-shops. As a result of this comparison we can determine whether some products are offered in more than one of these web pages and analyze the difference in the offer of several e-shops regarding the same product. 'Determining whether two offers refer to the same product involves extracting a set of features (product attributes) from the web pages containing the offers and comparing these features using a matching function' [**?**]. Identity resolution is the task of finding which of these correspondences between records describe the same real world entity. For our project we used crawled information about two product categories, bags and cameras. A product catalog was constructed using 50 seed products per each category. The information was gathered from crawling 21 websites for cameras and 16 websites for bags. As a result of our crawling process we gathered a total of 22979 camera product pages and 42856 bag product pages. In the crawled data we have features such as the name, description, page id, brand, price, product specific characteristics such as weight, color, size etc. However, most of the e-shops do not publish global product identifiers, which makes it more difficult to find the matching pairs for the products. For identifying which of these products are exactly the same, we applied several machine learning algorithms presented in this paper. These machine learning algorithms were initially applied to our manually annotated product matches in the gold standard, for evaluation and comparison to find which model has higher performance.

Chapter 2

Gold Standard and Preprocessing

To perform Identity Resolution and to evaluate the models that we want to apply to our entire corpus, we initially need a gold standard. For its construction, we manually annotate matching pairs of crawled pages with items in the product catalog. For the product catalogs, to distinguish between products, a unique ID is assigned manually. For the crawled pages, each product is identified through a page ID. To simplify the gold standard, to represent the matching pairs, we used only the corresponding IDs of the products that form the pairs. A binary representation (1 or 0) demonstrates whether the two IDs represent the same product or not.

2.1 Gold Standard Creation

To construct the gold standard, we manually found positive and negative matches from the crawled corpus for each entry of the product catalog and labeled that pair with 1 if the products are the same, we have a positive match, and 0 if the products are different, meaning we have a negative match. Initially we created a smaller gold standard to run the first approach as explained in 3.1 then refined it by adding corner cases. Following the example of [?] for gold standard creation for each entry in the product catalog we found at least one positive match from product pages. Since all products in the product catalog are different from each other, a page which is a positive match with the i^{th} product in the product catalog will definitely be a negative example for the other 49 products. With this heuristic we ended up with 120 positive and 5880 negative correspondences for cameras and with 114 positive and 5586 negative ones for bags. As a result we have a highly unbalanced dataset, with a large number of negative examples and just a few of positive ones. We did the mapping of the attributes of the products from product catalog with attributes of the product from the web pages. As general conclusion regarding gold standard creation, it was easier to find positive matches for cameras than for bags. We believe this comes as a result of the diversity of bags as a product category, considering the high number of producers and models. Even though we have crawled 42856 bag product pages which is nearly twice the number of crawled camera pages, we found more product overlap while creating cameras gold standard than when creating the bags gold standard.

2.2 Preprocessing

After creating the gold standards respectively for bags and cameras, we take the contents of the product pages and product catalog, preprocess and consider them as a bag of words. For this process different features such as product name initially, and adding price and brand afterwards were taken into consideration to see if adding more features would improve the performance of our models, giving us better matching results. Additionally we parsed the corresponding .HTML files using beautiful soup package in python while removing HTML tags and annotations. After parsing, we preprocessed the data by removing noise, tokenizing, normalizing and eliminating tokens that have more than 45 characters since the longest word in English vocabulary has 45 characters. In parallel we have done preprocessing of the extracted features annotated with schema.org attributes in rapid miner again we did tokenization, stemming and stopwords removal.

Chapter 3

Models

In order to tackle the problem at our hand, we decided to focus on four different approaches. While the first approach involves a very primitive way of calculating the similarity and defining a threshold based on f1 score, the second, third, and fourth option are more complex and consist of trained machine learning models. We exclusively worked with our gold standard in the first, second, and third approach, and in the fourth approach we included additional product identifier data. We had a highly unbalanced class problem as only 1 percent of our data consists of positive matches. Therefore, while evaluating the algorithms or deciding on the thresholds, we looked at the f1 scores.

3.1 Approach 1

First approach involves calculating similarity score between product catalog and product pages and then defining a threshold based on the calculated similarity to classify whether product pages and product catalog refers to the same product or not. For this approach, we initially built the structure in rapid miner and once we understand how to proceed, we transferred and rebuilt the whole process in python, as python gave us more flexibility. In approach 1, we applied Cosine and Jaccard similarity measures using different variables, which yielded a total of 9 different models. We first started with a simple model where we took only the name attribute from product catalog and took only the name attribute from the extracted schema.org annotations in product pages, as Bag of Words and TF-IDF to calculate Cosine and Jaccard similarities. Secondly, we decided to add price and brand information which is extracted from schema.org annotations in product pages and price and brand information available in product catalog, as BoW and TF-IDF, to see whether including more attributes will increase our prediction accuracy or not. After finding the best threshold and calculating f1 score, we found out that our prediction had indeed increased, even though not very significantly. That could be due to the fact that names as BoW is already a good identifier for a product, we can see the same situation in hierarchical classification. Lastly, we decided to take the whole HTML page that corresponds to the particular product page, preprocess and parse it and add it as tokens with BoW's or TF-IDF weightings and compare the similarity using Jaccard and Cosine with all the attributes available in product catalog also converted in BoW and TF-IDF. When html pages are added, we can see a significant drop in our predictions. We believe this is due to the fact that while adding html pages, we are also adding noise to our model. Same situation happened when we added html pages to our models in hierarchical classification. Another point worth mentioning is, HTML coupled with TF-IDF and cosine similarity wasn't able to distinguish between 1 and 0's and ended up with 0 accuracy. Overall, we reached a 0.91 F1 score in terms of matches for bags and 0.80 in terms of matches for cameras. You can see the corresponding thresholds and F1 score results in table 3.1.

3.2 Approach 2

Second approach involves taking the binary vectors of tokens between product pages and product catalog, assigning 1 if the token exists both in the product page and product catalog, and then applying classification machine learning algorithms in order to predict whether the item in the product catalog and the product page refer to the same product or not. This approach was applied only in python. We again started with the simplest model which is constructed by taking only names from product catalog and taking only names from the extracted schema.org annotations in product pages, tokenizing and assigning 1 if a particular token exists both in the product page and product catalog. Secondly, we added price and brand attributes in order to increase the algorithms prediction. Aligned with approach 1, when we added additional variables our prediction accuracy increased significantly in terms of F1 Score of 1 in decision tree and logistic regression. Thirdly, we added HTML pages in a similar manner described in Approach 1 and took all the available attributes from product catalog, and assigned 1 if a token exists both in the HTML of a product page and in all the attributes in the correspondent product in the product catalog. When we added HTML pages as tokens, we also experienced a drop in our prediction accuracies, which is also aligned with the results we saw

Algorithms	Threshold Cameras	Cameras F1 (0)	Cameras F1 (1)	Threshold Bags	Bags F1(0)	Bags F1(1)
Only Names+ BoW +Cosine	0.55	1	0.78	0.50	1	0.9
Only Names+ BoW +Jaccard	0.3	0.99	0.70	0.3	1	0.71
Only Names+ TF-IDF +Cosine	0.4	1	0.78	0.35	1	0.9
Names, Price, Brand+ BoW +Cosine	0.5	1	0.8	0.4	1	0.9
Names, Price, Brand+ BoW +Jaccard	0.3	0.99	0.71	0.2	1	0.91
Names, Price, Brand+ TF-IDF +Cosine	0.4	1	0.79	0.25	1	0.9
HTML+ BoW +Cosine	0.25	0.99	0.30	0.1	0.99	0.47
HTML+ BoW +Jaccard	0.0107	0.97	0.20	0.0087	0.99	0.02
HTML+ TF-IDF +Cosine	0.34	0.98	0.34	0	0	0.04

Table 3.1: Approach 1 Results

in approach 1. In here, in order to increase our predictions, we decided to apply f classification method and choose 100 variables before applying classification algorithms in order to find the relevant attributes. With f-classification, our results have indeed increased but they are still lower compared to applied algorithms with clean features. For each of the step, we applied four different machine learning algorithms, which includes decision trees, logistic regression, nearest centroid and random forest, which in return yielded with 16 different models. Compared to approach 1, approach 2 predictions are not as good. This could be due to the fact that, for option 1 we assigned probabilities manually. Therefore, we had the chance to assign a threshold smaller than 0.5 if it resulted with good predictions, whereas in this approach, the algorithm's predefined threshold is 0.5, and classifies products according to this threshold. The result of approach 2 can be found in table 3.2.

Algorithms	Cameras F1 (0)	Cameras F1 (1)	Bags F1(0)	Bags F1(1)
Only Names+ Decision Tree	0.99	0.71	1	0.8
Only Names+ Logistic Regression	1	0.72	1	0.8
Only Names+ Nearest Centroid	0.93	0.16	0.94	0.18
Only Names+ Random Forest	0.99	0.63	1	0.74
Names, Price, Brand +Decision Tree	1	0.77	1	0.79
Names, Price, Brand +Logistic Regression	1	0.73	1	0.77
Names, Price, Brand +Nearest Centroid	0.92	0.15	0.94	0.19
Names, Price, Brand +Random Forest	0.99	0.57	1	0.74
HTML + Decision Tree	0.98	0.16	0.99	0.17
HTML + Logistic Regression	0.99	0.24	0.99	0.15
HTML + Nearest Centroid	0.68	0.01	0.59	0.03
HTML + Random Forest	0.99	0.03	0.99	0.03
HTML + Decision Tree + F Classification	0.99	0.23	0.99	0.18
HTML + Logistic Regression + F Classification	0.99	0.30	0.99	0.05
HTML + Nearest Centroid + F Classification	0.81	0.07	0.72	0.05
HTML + Random Forest + F Classification	0.99	0.2	0.99	0.08

Table 3.2: Approach 2 Results

3.3 Approach 3

Third approach involves calculating similarity between a particular attribute in product catalog and its same correspondence extracted from the product pages by feature extraction team and assigning these calculated similarities as weights to the attributes which then feeds into machine learning algorithms. Feature extraction team extracted 16 variables in cameras and 8 in bags, but not all the extracted attributes were complete as most of them had missing values. Therefore, we first preprocessed the data by dropping attributes if they had more 50 percent missingness and by filling them with the most frequent values if they had less than 50 percent missingness. After this preprocessing, we ended up with 11 attributes to compare in cameras and 7 attributes in bags. For the first 8 models, we calculated Jaccard similarity score by treating every column as text, and applied decision tree, logistic regression, nearest centroid and random forest algorithms. We first applied these algorithms on unbalanced dataset. The results from the algorithms are higher compared to approach 2 for cameras, but still lower than approach 1. Bags in general resulted with worse than all other approaches with every way we tried, since name wasn't within the features that were extracted by the feature extraction team.

Additionally, in order to overcome the issue of balancing as mentioned at the beginning of the chapter, we decided to balance the dataset by removing duplicated rows and applied the algorithms to the balanced dataset. With this approach, we were able to increase the prediction accuracy of bags but in return lost prediction power in cameras. For the other 8 models, we decided to use Gaussian Kernel in order to compare two integer values rather than treating them as text, for string value comparisons, we kept on using Jaccard similarity. Using Gaussian Kernel as a similarity measure to compare two integer values, has increased our predictions in cameras and lowered in bags in unbalanced dataset.

To summarize, in terms of bags we reached bad results since name attribute which is one of the most important attributes used to identify a product wasn't extracted for bags and consequently, didn't include it in the model. For cameras, we reached a prediction result of 0.8 using Gaussian Kernel and random forest which is the highest result in approaches which used machine learning. Overall, balancing didn't help in improving our prediction.

Algorithms	Cameras F1 (0)	Cameras F1 (1)	Bags F1(0)	Bags F1(1)
Jaccard + Decision Tree (Unbalanced)	1	0.77	0.99	0.21
Jaccard + Logistic Regression (Unbalanced)	1	0.75	0.99	0.15
Jaccard + Nearest Centroid (Unbalanced)	0.97	0.38	0.94	0.12
Jaccard + Random Forest (Unbalanced)	1	0.79	0.99	0.18
Jaccard + Decision Tree (Balanced)	0.95	0.73	0.79	0.26
Jaccard + Logistic Regression (Balanced)	0.95	0.75	0.83	0.05
Jaccard + Nearest Centroid (Balanced)	0.79	0.57	0.72	0.41
Jaccard + Random Forest (Balanced)	0.93	0.68	0.73	0.23
Jaccard & Gaussian Kernel + Decision Tree (Unbalanced)	1	0.79	0.99	0.03
Jaccard & Gaussian Kernel+ Logistic Regression (Unbalanced)	1	0.77	0.99	0.02
Jaccard & Gaussian Kernel+ Nearest Centroid (Unbalanced)	0.97	0.38	0.94	0.09
Jaccard & Gaussian Kernel+ Random Forest (Unbalanced)	1	0.8	0.99	0.03
Jaccard & Gaussian Kernel+ Decision Tree (Balanced)	0.93	0.68	0.56	0.14
Jaccard & Gaussian Kernel+ Logistic Regression (Balanced)	0.94	0.71	0.8	0
Jaccard & Gaussian Kernel+ Nearest Centroid (Balanced)	0.77	0.55	0.7	0.38
Jaccard & Gaussian Kernel+ Random Forest (Balanced)	0.92	0.68	0.55	0.17

Table 3.3: Approach 3 Results

3.4 Approach 4

We learned individual models in this approach that are trained on additionally collected data and evaluated on our gold standard. Thus, we only used the gold standard for evaluation in this option. Our additional data is collected by searching for product identifiers (e.g. GTIN) on the web that correspond to the products in our product catalog. While searching for product identifiers, we have collected five additional pages for each camera in the product catalog and three additional pages for each bag in the product catalog. This resulted in a larger corpus than our gold standard. Generally, as previously mentioned it is much more difficult to find additional pages for bags, because bags are more diverse than cameras and thus usually offered in less, but more specialized, shops.

It is now possible to construct a corpus of training data out of the additionally collected pages. Generally, this is done by following the approach described in the creation of the gold standard (cf. Chapter 2). Subsequently, we performed similar preparation steps than in the previous options. We, for example, removed extracted features with too many missing values and filled missing values in the remaining extracted features with the most frequent value. Then, we computed the Jaccard similarity between the corresponding features of the product catalog and the extracted features of the additional pages. Thus, we obtained a particular number of entries for each camera in the product catalog (i.e. $\# = 250$) and each bag in the product catalog (i.e. $\# = 150$). Obviously, this resulted in five positive labels for each camera in the product catalog and three positive labels for each bag in the product catalog. Afterwards, we split the resulting training data to learn individual models for each product in the product catalog. Our models were then evaluated with respect to the gold standard. We followed the same methods, took the same features, and applied the same similarity measure to construct our test data out of the features of the product catalog and the extracted features of the gold standard.

Generally, we achieved the best configuration for each model by performing a grid search on the training data. Then, we took the best configuration and trained a model with this configuration on the training data. Again, we performed this optimization step individually for each product in the product catalog. While we were able to take measures to balance our training data for the cameras (i.e. dropping duplicate entries), this is impossible for the bags, because there we have too few differently looking entries with respect to the positive label. Subsequently, we were able to evaluate our models on the test data which is left unbalanced. We learned four different models (i.e. *DECISION TREE*, *LOGISTIC REGRESSION*, *RAN-*

DOM FOREST, *GRADIENT BOOSTING*) to perform identity resolution in this option. Our results are provided in Table 3.4 and report the average performance with respect to the individual models in each category.

	DECISION TREE	*LOGISTIC REGRESSION*	*RANDOM FOREST*	*GRADIENT BOOSTING*
0 (cameras)	0.99	0.99	0.99	0.99
1 (cameras)	0.66	0.58	0.68	0.69
0 (bags)	0.99	0.99	0.99	0.99
1 (bags)	0.03	0.02	0.03	0.03

Table 3.4: F_1 (#4)

If we take a look at the obtained results, we observe that our models perform worse than in the previous option. Generally, we observe this result, because the most promising shops are the shops we have actually crawled for. Therefore, the quality of the pages contained in the additional data is usually worse than the quality of the pages contained in our gold standard. This issue is much more severe for the bags where we have much more diversification in the way of presenting bags to the customer. We also have to consider that features extracted with feature extraction techniques might be wrong. Again, this is much more severe with respect to bags, because we have, for example, no extraction of the name. Therefore, we achieve a much better performance with respect to the cameras than with respect to the bags.

Chapter 4

Conclusion

The Web offers us a high number of products. In order to understand if different e-shop offer the same products and also to compare the offer varieties we need to perform Identity Resolution. Firstly, we created a gold standard; for bags we had 114 positive matching pairs and 5586 negative ones, and for cameras we had 120 positive pairs and 5880 negative ones. To find the matches between the products in our crawled pages and the product catalog, we used four different approaches. For the first approach we calculated the similarity score between product catalog and product pages and then defined a threshold based on the calculated similarity. Second approach involves taking the binary vectors of tokens between product pages and product catalog, assigning 1 if the token exists both of them. For these approaches we initially used the name attribute as Bag of Words, and then added price and brand attributes. When adding price and brand we concluded that our prediction had slightly increased. When .html pages are added in both cases, we can see a significant drop in our predictions, as while adding .html pages, we are also adding noise to our model. Third approach involves calculating the similarity between a particular attribute in product catalog and its same correspondence extracted from the product pages. The results from the models in the third approach for cameras had better results than the second approach but lower than the first one. As for the bags, they generally had the worst results in this approach than all the others, as the name attribute which has high importance, wasn't extracted. In the fourth approach, we learned models to collect additional data by searching for product identifiers (e.g. GTIN) on the web that correspond to the products in our product catalog, therefore generating 5 additional pages for cameras and 3 for bags. For this approach we had the highest results with Gradient Boosting for cameras. For bags however, the results are almost the same for all the models.